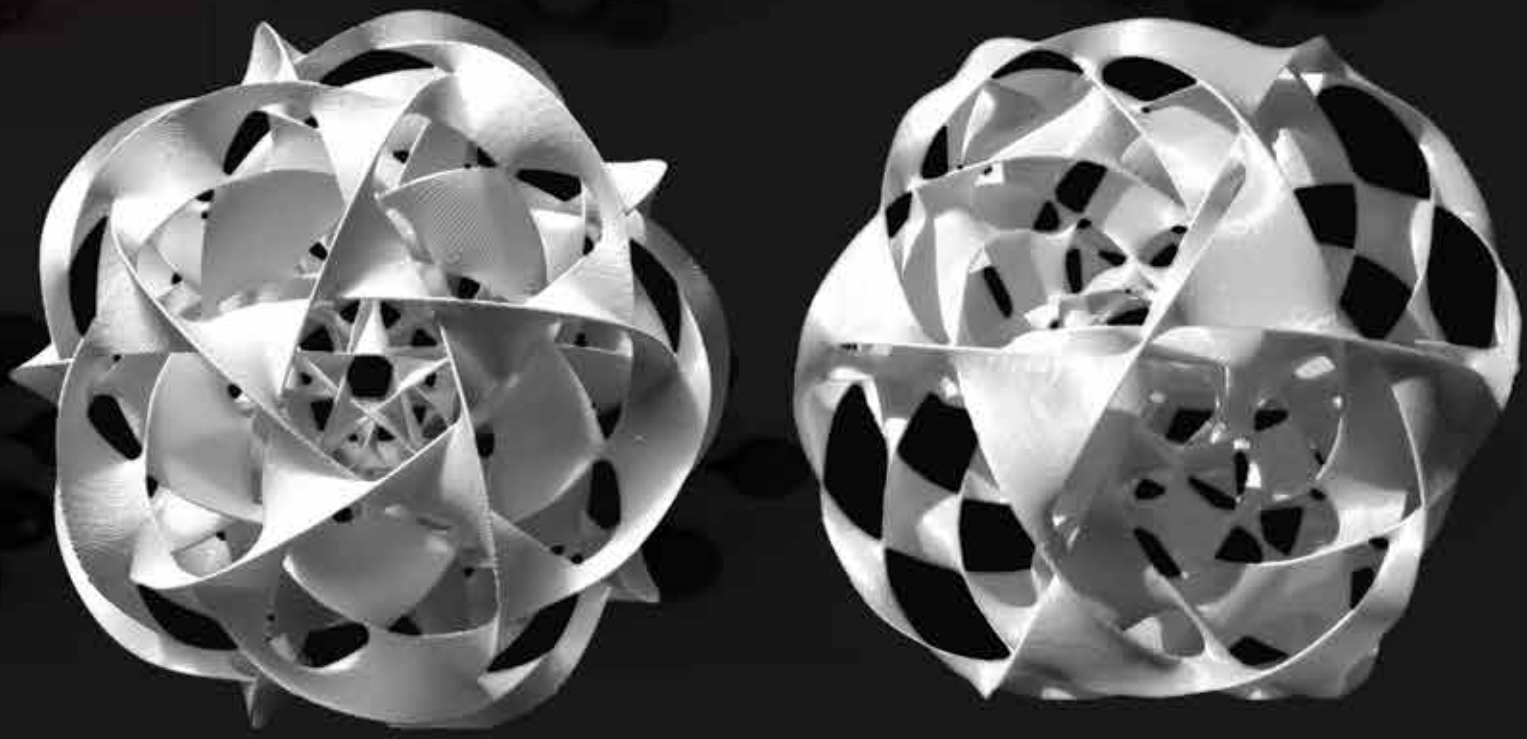
HYPERSEEING

The Publication of the International Society of the Arts, Mathematics, and Architecture

SUMMER 2019
www.isama.org

The Proceedings of the SMI 2019 Fabrication & Sculpting Event (FASE)



Special Issue on Shape Modeling International 2019

17-21 June 2019 Vancouver Canada

HYPERSEEING
Special Issue on SMI 2019

International Geometry Summit 2019

Shape Modeling International 2019
Fabrication and Sculpting Event

Eighteenth Interdisciplinary Conference of the International Society of the Arts, Mathematics, and Architecture

Vancouver, Canada
June 17-21, 2019

FASE Conference Chairs

Ergun Akleman (Texas A&M University)
Karina Rodriguez Echavarria (University of Brighton)

FASE Paper Chairs

Negar Kalantar (California College of the Arts)
Vinayak Raman Krishnamurthy (Texas A&M University)
Konrad Polthier (FU Berlin)

FASE Program Committee

Valery Adzhiev
Bournemouth University

Melinos Averkiou
University of Cyprus

Moritz Baecher
Disney Research

Hujun Bao
Zhejiang University

Gill Barequet
Technion - Israel Institute of Technology

Loic Barthe
IRIT - Université de Toulouse

Amit Bermano
Princeton University

Silvia Biasotti
IMATI-CNR

Michael Birsak
KAUST

Adrien Bousseau
INRIA

Vladimir Bulatov
Shapeways, Inc

Daniela Cabiddu
IMATI-CNR

Marcel Campen
RWTH Aachen University

Nathan Carr
Adobe Systems Inc.

Raphaëlle Chaine
LIRIS Université Lyon

Siddhartha Chaudhuri
IIT, Bombay

Weikai Chen
University of Hong Kong

Jianer Chen
Texas A&M University

Francesco De Comite
University of Sciences and Technology, Lille

Bailin Deng
Cardiff University

Olga Diamanti
Autodesk

Ye Duan
University of Missouri at Columbia

Jeremie Dumas
New York University

Gershon Elber
Technion- Israel Institute of Technology

Bianca Falcidieno
IMATI-CNR

Robert Fathauer
Tessellations Inc.

Greg N. Frederickson
Purdue University

Oleg Fryazinov
Bournemouth University

Paul Gailiunas
Bridges Organization

Xifeng Gao
Florida State University

Abel Gomes
University of Beira Interior

Eric Guérin
LIRIS

Takeo Igarashi
The University of Tokyo

Ioannis Ivrissimtzis
Durham University

Alec Jacobson
University of Toronto

Bih-Yaw Jin
National Taiwan University

Bert Juettler
Johannes Kepler University

Vladimir Kim
Adobe Systems

Robert Krawczyk
Illinois Institute of Technology

Hamid Laga
The University of Tokyo

Manfred Lau
City University of Hong Kong

Xin Li
Louisiana State University

Jyh-Ming Lien
George Mason University

Ligang Liu
U. of Science and Technology of China

Lin Lu
Shandong University

Stephen Luecking
DePaul University

James Mallos
Sculptor

Luigi Malomo
IST-CNR

Marcel Morales
Université de Grenoble I

Marcos Novak
University of Califonia, Santa Barbara

James Palmer
Northern Arizona University

Alexander Pasko
Bournemouth University

Giuseppe Patanè
IMATI-CNR

Jean-Philippe Pernot
Arts et Métiers ParisTech
Nico Pietroni
IST-CNR

Helmut Pottmann
Vienna University of Technology

Rinus Roelofs
Sculptor

Damien Rohmer
Ecole Polytechnique

Asla Medeiros Sa
FGV Escola de Matemática Aplicada

Faramarz Samavati
University of Calgary

Henry Segerman
Oklahoma State University

Kenneth Sloan
University of Alabama at Birmingham

Shinjiro Sueda
Texas A&M University

John Sullivan
Technische Universität Berlin

Carlo H. Séquin
University of California, Berkeley

Kenshi Takayama
National Institute of Informatics

Jean-Philippe Vandeborre
IMT Lille Douai

Juraj Vanek
Arevo Inc.

Remco Veltkamp
Utrecht University

Tom Verhoeff
Eindhoven University of Technology

Charlie Wang
Delft University of Technology

Brian Wyvill
University of Victoria

Yong-Liang Yang
University of Bath

Qingnan Zhou
Adobe Systems

Bo Zhu
Dartmouth College

CNR: Consiglio Nazionale delle Ricerche, Italy

FASE Supported by

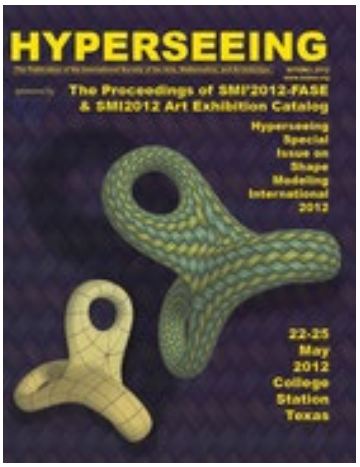
California College of the Arts
FU Berling
University of Brighton
Texas A&M University

CONTENTS

Proceedings of Fabrication and Sculpting Event

History & Future: Fabrication and Sculpting Event

Ergun Akleman and Karina Rodriguez Echavarria
FASE Conference Chairs



There are at least two aspects to shape modeling: theoretical and practical. The mathematical and theoretical aspects of shape modeling have traditionally been supported by the SMI conference. With the Fabrication and Sculpting Event our goal is to include more hands-on, application-oriented ways by designers and sculptors who construct sophisticated real physical shapes. The Fabrication and Sculpting Event has its own program committee, and the accepted papers are published in Hyperseeing. With FASE, we hope to attract practitioners who might usually be less inclined to write papers containing formal algorithms or mathematical proofs, but who nevertheless have important things to say that are of interest to the shape modeling community and who also might provide visually stimulating material.

The Fabrication and Sculpting Event (FASE) started as an experiment in expanding the scope of shape modeling international (SMI) conference in 2012. We also had another FASE event in SMI'2013. There were very positive responses to the Fabrication and Sculpting Event papers and presentations both 2012 and 2013. Although we skipped FASE in SMI'2014, based on the success of earlier events, we continued the FASE event in both 2015 and 2016.

In 2013, Nat Friedman, the chair of the International Society of the Arts, Mathematics, and Architecture (ISAMA), asked us if we could organize the event as an annual ISAMA conference. The SMI steering committee unanimously agreed with the suggestion. As a result, the event can now be considered also as the Eighteenth Interdisciplinary Conference of ISAMA.

The ISAMA conference has a rich history. The first Art and Mathematics Conference (AM 92) was organized by Nat Friedman at SUNY-Albany in June, 1992. This conference was followed by annual conferences AM93-AM97 at Albany and AM 98 at the University of California, Berkeley, co-organized with Carlo Sequin. ISAMA was founded by Nat Friedman in 1998 along with the ISAMA publication Hyperseeing co-founded with Ergun Akleman in 2006. In addition, the Art/Math movement has taken

off with the formation of many additional conferences and organizations. In particular, we mention the very successful conference Bridges organized by Reza Sarhangi in 1998 and the excellent Bridges Proceedings. The significance of the art/math movement is now recognized internationally and in particular by the extensive art/math exhibit at the annual Joint Mathematics Meeting of the American Mathematical Society and the Mathematical Association of America organized by Robert Fathauer.

The main difference with other math/art conferences is that FASE focuses solely on 3D shapes. We invite submissions mainly from practitioners such as sculptors and architects to describe their methods. We expect that such papers and the following discussions can provide new problems, issues and questions for theoretical shape modeling research.

In the last few years, there has been an increased interest in these topics by the communities working at the intersection between engineering, fabrication and computing. This is partially driven by the ability to easily acquire and represent 3D shapes from the real world as well as the availability of digital fabrication technologies, such as additive manufacturing. These technologies comprise a combination of programmable digital tools, processes, materials and equipment which allow the creation of physical objects of complexities not achievable by traditional manufacturing processes.

Thus, we foresee that FASE will play an important role in the future in order to: i) provide a forum for these communities to gather user requirements, exchange ideas, knowledge, and expertise; as well as ii) provide a publication outlet to showcase the work of researchers and practitioners across the world.

Preface: Fabrication and Sculpting Event 2019

Negar Kalantar, Vinayak Raman Krishnamurthy and Konrad Polthier
FASE paper chairs

For this year's Fabrication and Sculpting Event, we solicited papers that pose new questions and motivate further research in designing, fabrication and sculpting. Topics should be useful, for example, in the following areas: fabrication of digital models; advanced manufacturing techniques such as additive manufacturing, laser cutting or CNC milling; interactive or procedural design of manufacturable shapes; interconnections of complex modeling and fabrication processes; and visually stimulating fabrication techniques or printed structures.

Thus, the scope of FASE is at the intersection of shape modeling and fabrication methods/algorithms, and papers may focus on both the digital/theoretical and the physical domain or just one of these domains – as long as the connection to the other domain is clear. It is not a requirement that the techniques presented in the paper involve computation as such, but they need to have a clear algorithmic or mathematical element.

We received nine submissions this year. Three of them were accepted as long full papers (max. 16 pages) and two were accepted as short papers (max. 8 pages). The full and short papers will be presented on Friday, June 21st between 10:20-12:05 in SMI&FASE Session at Joseph & Rosalie Segal Centre 1400-1410. The five accepted papers span a wide range of topics and views on the fabrication process of various artistically interesting artifacts. They include papers on design of shapes, including methods for designing complex star-cinder sculptures, bead sculptures which model many microscopic molecular structures faithfully, and symmetrical models of mathematical knots. Fabrication papers focus on the design and fabrication of data-drive anamorphic sculpture, decorative knots and fluids simulations.

Furthermore, three submissions were accepted as Posters and they will be presented and exhibited at IGS poster session Wednesday, June 19th between 14:30-15:30 in IGS Poster Fast Forward and Public Display at Djavad Lobby.

We wish to thank the program committee members and authors for their participation in the SMA/ISAMA 2019 Fabrication and Sculpting Event. We hope that new ideas and partnerships will emerge from the FASE papers that can offer a glimpse into a much larger territory and the event can enrich interdisciplinary research in Shape Modeling. We also hope that the attendees of SMI 2019 enjoy this event of the conference.

Combining Procedural Modeling and Interactive Graphical Editing for the Design of Abstract Geometrical Sculptures

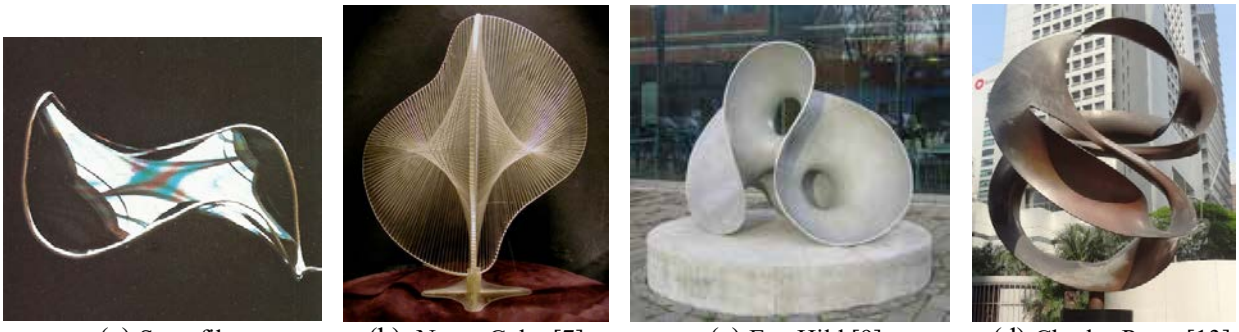
Carlo H. Séquin and Toby Chen
CS Division, University of California, Berkeley
E-mail: sequin@cs.berkeley.edu

Abstract

We describe an experimental CAD tool aimed at creating intricate 2-manifold surfaces suspended by complex tangles of border curves. This exploratory system aims at combining the best of two worlds: high-level procedural modeling for capturing various symmetries and replications of geometrical entities, and interactive graphics editing to define the surface elements that would be difficult to specify procedurally in an initial design file. The key issue under investigation is how to best capture the geometrical changes made in the graphics domain and reintroduce them into the procedural design file, which, in the end, is supposed to define the whole sculpture, while maintaining the flexible parameterization contained in the original specification.

1. Introduction

Many of the abstract geometrical sculptures by artists such as Naum Gabo [7], Eva Hild [9], or Charles Perry [13] define intricate 2-manifold surfaces suspended by complex tangles of border curves (Fig.1). We found it very difficult to model such geometries with commercial or public-domain CAD tools such as 3ds Max [1], Maya [11] or Blender [2].



(a) Soap film (b) Naum Gabo [7] (c) Eva Hild [9] (d) Charles Perry [13].

Figure 1: *Soapfilm-like surfaces suspended on smooth border curves.*

Creating the many interlinked border curves, while maintaining perfect symmetry, seems almost impossible with a typical on-screen, interactive, graphical modeling tool. It is much more convenient to define these curves in a procedural manner, for instance with a set of properly parameterized B-spline curves. These parameters can then be used to fine-tune the border curves as well as the suspended surface(s), while inspecting the complete sculpture in its (almost) final form.

On the other hand, it is even more difficult to define all the surface elements in their final form procedurally, that is, individually specifying the set of face vertices needed for all surface facets in code or some other user interface. For this part of the design task, an interactive graphics interface is much more appropriate for selecting suitable segments on some of the border curves and defining some surface patches between them. Most of the above mentioned, commercial CAD tools offer some ways of connecting curve segments with "bridges" or by extrusions. But, typically, these surface elements are captured in some low-level, non-hierarchical form and cannot be integrated into the original hierarchical, procedural description of the border curves. Some of the tools even force the user to convert the parametric description into a flat mesh before any mesh-level edits can be made. This is a particularly severe shortcoming, when the sculpture

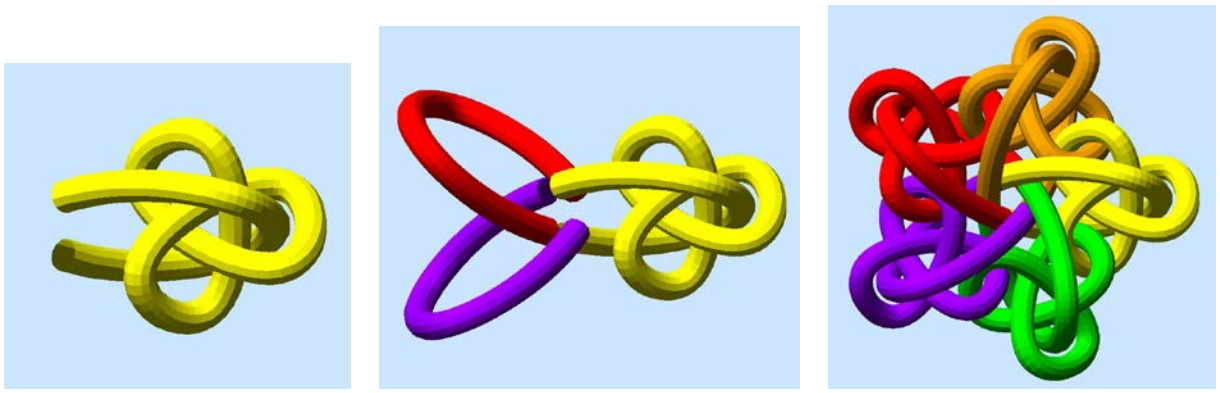
under construction exhibits a lot of symmetry, such a Perry's *Star Cinder* [14] (Fig.6a). In a sculpture like this, with 60-fold symmetry, it should be sufficient to define only $1/60^{th}$ or $1/30^{th}$ of the overall surface and then let the procedurally specified symmetries construct the remainder of the surface. For this purpose, it is desirable to have the interactively added parts integrated into the original high-level description. If the needed prototypical surface elements are placed into the proper location in the procedural description, all necessary replications of these elements will happen automatically.

With the specific goal of reconstructing sculptural surfaces like the ones created by Eva Hild (Fig.1c) [9] or by Charles O. Perry (Fig.1c) [13] or (Fig.6a) [14], we have started the development of a CAD tool, called NOME (Non-Orientable Manifold Editor), that helps us to create and optimize such designs. Our goal is to combine the best features of high-level procedural modeling for capturing various symmetries and replications of geometrical entities, with direct interactive graphics editing for specifying needed surface elements. The key challenge is to let the graphical editing step not be the final design phase, but to re-integrate these edits into the initial procedural file so that it now specifies the sculpture in its current form. The designer should be able to pick up this file at a later time and continue the editing and optimization of the geometry, while still enjoying the full functionality of all the various parameterizations that were included in the original specification. Achieving this goal is surprisingly difficult, and it accounts for the fact that this development has stretched over the last three years [6][19]. While the evolving NOME tool has been good enough to design the intended complex geometrical 2-manifolds, as demonstrated by some examples later in the paper, the integration between procedural descriptions and graphical editing is far from being solved perfectly. This paper tries to bring some of the pending issues to the forefront and describe our current thinking about them.

2. NOME Files

NOME is a design-specific language, akin to that of a single-assignment programming language. It follows the normal paradigm of most hierarchical graphics scene descriptions. It defines *points* by their (*x*,*y*,*z*) coordinates, and then uses such points to define *edges*, *polylines*, *faces*, and *meshes*. *Instances* of several such elements can be combined in a *group*. Instances of such groups can then be placed in multiple locations in the scene, after suitable optional transformations, including *non-uniform scaling*, *rotation around an arbitrary axis*, and *translation*. For instance the 5-fold rotational (C_5) symmetry around the chosen (0 0 1) *z*-axis, exhibited by the model shown in Figure 2c, is expressed with the following statements:

```
instance   ID_0   MODULE  yellow       end_instance
instance   ID_1   MODULE  orange   rotate(0 0 1)(72)   end_instance
instance   ID_2   MODULE  red      rotate(0 0 1)(144) end_instance
instance   ID_3   MODULE  purple   rotate(0 0 1)(216) end_instance
instance   ID_4   MODULE  green    rotate(0 0 1)(288) end_instance
```



(a) One MODULE *(b) connections between neighbors* *(c) five MODULEs joined*

Figure 2: *Cinquefoil with five interlinked trefoils.*

Of course, "MODULE" (Fig. 2a) may itself be a hierarchical construct, consisting of instances of other, smaller groups; for instance, MODULE could contain one complete trefoil and 1/5th of the cinquefoil:

```
group MODULE
  instance  ID_5  One-fifth-of_cinquefoil  end_instance
  instance  ID_6  Complete_trefoil-knot   end_instance
end_group
```

It may be difficult to define the exact geometry of MODULE (Fig.2a) up front in a procedural manner. The cinquefoil cannot be completed (Fig.2b), nor the trefoil properly dimensioned and positioned, until they can be judged in the context of the complete 5-fold assembly (Fig.2c). Thus, it is desirable to specify the MODULE geometry with some built-in parameters, which may concern non-uniform scaling, its placement in 3D space, or the details by which the cinquefoil loop in MODULE is connecting with its two neighbors (Fig.2b) to form, in the end, the complete cinquefoil.

The key mechanism for parameterization is through alteration of any numerical value in the NOME specification. Wherever a numerical parameter is needed for a NOME statement, e.g. to specify the *x*-, *y*-, or *z*- coordinates of a vertex, or the radius, *r*, of some circle, an expression must be supplied. This can simply be a number, or it may entail operations involving complex algebraic expressions and several numerical variables that can be adjusted interactively with corresponding *sliders* [6]. Whenever the current state of design is saved, the latest values of these sliders are also saved, and the necessary updates are made in the initial procedural description.

If MODULE, and all other geometry, are defined procedurally, and thus the whole design emerges from a procedural text file, this approach presents no real problems, and can be realized within many CAD environments, ranging from Rhino3D [15] to Berkeley SLIDE [16]. Difficulties arise, when a designer wants to introduce new elements into the design that are difficult to specify in the initial procedural description, such as some surface elements between arbitrary segments of different border curves. Such facets are more naturally introduced by clicking on sample vertices along some of the border curves. The challenge is to find a way to integrate these interactively defined design extensions into the original program that was used as the start of the design.

Many computer-aided design or modeling tools combine some procedural input with subsequent direct manipulation in a graphics display. Only a few tools try to integrate the changes made in the interactive graphics environment into the original procedural description. Most advanced in this respect is the *Sketch-n-Sketch* system [5][8], where a generator program is constructed based on the user's graphical drawing and editing operations. For our current application, where we want to construct complex 3D object with high symmetry, rather than 2D vector graphics designs, we always start with a procedural description that captures the basic structure and all its symmetries; normally this framework is not changed interactively.

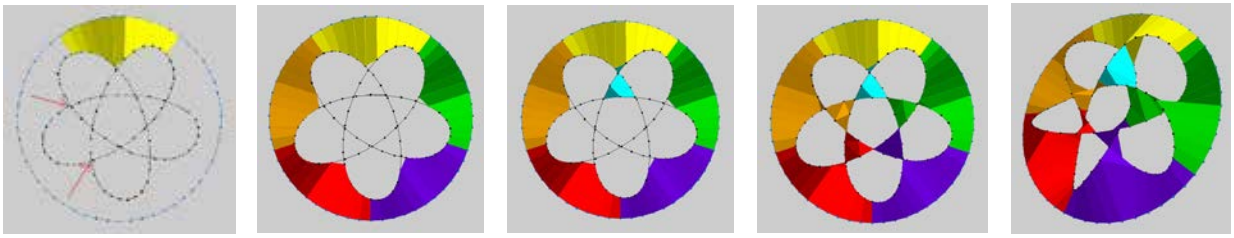
The designer typically starts by specifying a symmetrical tangle of border curves or a set of strategically place anchor faces that mark the main symmetry axes. Then, additional faces are added to define the topology of the final surface. NOME is crafted specifically to support the capability of constructing stretchable surfaces between procedurally defined geometrical features. This capability of linking up an arbitrary collection of vertices to form a stretchable face is made possible by assigning every geometrical entity, down to the vertices at the leaves of the scene tree, a unique hierarchical identifier. Every *vertex*, *object*, or *instance-call* is given an *ID*, i.e., a unique identifying label. The name for an arbitrary vertex then is constructed by concatenating all these labels along the path from the root of the scene tree to its final leaf instance. For vertices created by a built-in generator, such as the *torus* generator, or the *B-spline* generator used to define border curves, the last part of the vertex name is produced inside the corresponding generator. New facets, generated in an interactive, graphical editing session, are then described by reference to these composite textual labels, and can thus be added and referenced unambiguously in the original NOME design file.

A few higher-level operations make it easier to connect larger segments of border curves. Two multi-segment regions of border curves can be selected, and a zippering operation can then be invoked that forms a bridging ribbon between these two borders, composed of quadrilaterals and possibly some extra triangles, that prevent the connecting quadrilaterals from becoming too skewed [6].

In general, we want to save all changes that a designer makes in the interactive graphics environment in the initial procedural specification, while changing this file as little as possible. When a designer reopens this file at a later time to make further design enhancements, the file should look as familiar as possible. At intervals convenient to the designers, we let them save the state of a design. At those times, the most recent changes made are saved as an appendix to the original NOME file. Typically, the designer then reopens this file and moves some of the newly created facets to an appropriate place in the hierarchical description, so that these facets are now replicated with the symmetries specified in the original design; the displayed surface will then reappear more fully instantiated. This switching back and forth between two quite different views of the emerging design has proven quite powerful and convenient for creating the type of intricate 3D models presented below.

3. NOME Workflow Example

This section presents a simple tutorial example to describe a typical workflow in NOME for the creation of a soap-film-like surface spanning a cinquefoil surrounded by a circular ring (Fig.3). The border curve of the cinquefoil is specified as a cubic B-spline with three geometry parameters that makes it easy to adjust the extent and the width of the lobes, as well as the closest distance between the knot curve and its 5-fold symmetry axis. There is also a parameter that describes the amplitude of the undulation in the *z*-direction and thus controls the separation of the knot curve branches at the locations of the projected knot crossings (red arrows in Fig.3a). In the shown example, the B-spline is sampled at 90 uniformly spaced parameter values, and the circle is represented as a 55-sided polygon; both of these values are tied to sliders and can be changed interactively. These particular values are higher than they need be to define the final 2-manifold well enough so that it looks good after a few levels of refinement by Catmull-Clark subdivision [4]; but they provide enough clickable locations so that the designer can readily insert facets with a variety of shapes and aspect ratios. The designer need not make use of all the available sample points on the border curves.



(a) *(b)* *(c)* *(d)* *(e)*

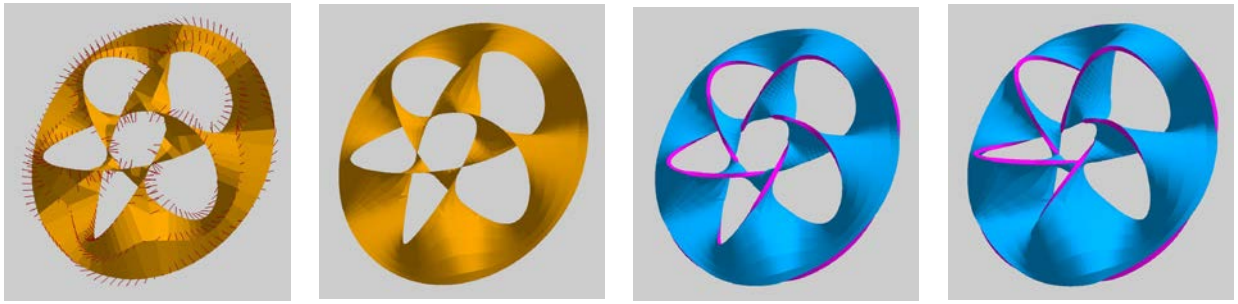
Figure 3: *Incremental construction of a 2-manifold suspended by the given two border curves.*

Another consideration in choosing the sampling density on each curve concerns the formation of nicely twisted saddles at all points where there are projected knot crossings. Ideally, one would like to have a pair of sample points as close as possible to the point of minimal distance in the skewed crossing of the two curves (red arrows in Fig.3a). Finally, the number of sampling point should respect the symmetry of the emerging design. The sample vertices must lie in the same respective locations on all the corresponding lobes of a given knot, so that any partial geometry that is subjected to the specified symmetry operations will have its vertices land on corresponding vertices on the target lobes. This is important in order to obtain a clean merger between the various partial geometry pieces, entered manually or generated automatically, into a single cohesive 2-manifold.

With a good choice of sampling points, the designer can now click on sets of (typically four) vertices and define them as facets in the forthcoming surface (Fig.3a). Quadrilaterals are the preferred facet shapes,

when the surface is later refined with Catmull-Clark subdivision [4]. A good start is to place the most clearly needed facets on the inside of the outermost rim, connecting them to the dominant lobes of the cinquefoil knot (Fig.3a); these facets then form parts of a continuous outer ribbon that will eventually travel around the whole sculpture. When roughly one fifth of the circumference has been filled in, it is time to save the current state of design. In the saved NOME file, a list of the newly added facets will appear as an appended small new mesh plus an *instance* call to that mesh. In this text file, the designer can now readily add four more instance calls with different rotation angles around the *z*-axis, in increments of 72°, as in the example in Section 1. Giving the five instances different colors, makes it easy to see, whether the added facets properly compose the whole circumference ribbon (Fig.3b), whether there is still a gap to be filled, or whether some of the facets in the different instances already overlap. Those flaws are then easily corrected, and the designer can resume an interactive editing session based on the modified NOME file.

The next step is to fill in the inner ring of five "triangular" regions (Fig.3c). Newly added facets can now either be included into the ribbon mesh created in the first editing session, or they can form a separate mesh that is replicated five times on its own, and which may be colored differently (e.g., cyan). The latter approach allows one to temporarily turn off some or all of the instances of the first mesh, to enhance the visibility into the interior of an emerging sculpture. This is not much of an issue in this 2.5D example, but becomes important in the complex, nested, 3D geometries discussed in Section 4. Eventually, all the facets have been placed, and a complete hierarchical specification of the desired 2-manifold has been generated (Fig.3d). Figure 3e provides an oblique view of this same assembly to yield a better understanding of the 3D nature of the twisted saddle geometries emerging at the border-curve cross-over points. The NOME-code describing the state of design shown in Figure 3d can be found on-line [17].



(a) (b) (c) (d)

Figure 4: *Smoothing and offsetting. (a) subdivision=2 with normal vectors; (b) subdivision=3; (c) offset surface generation; (d) adjusting some geometry parameters.*

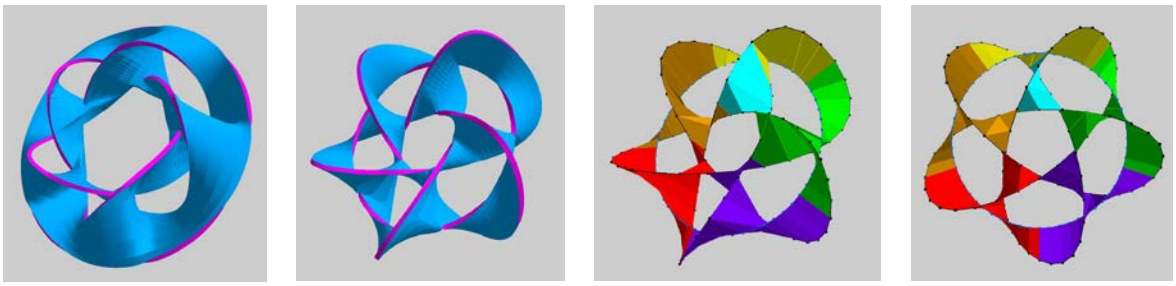
Now it is time to do a *merge* operation. In this process, in a bottom-up, recursive manner, all singly-used *border edges* of local pieces of the 2-manifold are tested for coincidence. If both pairs of end-vertices of a pair of border edges are within some small *epsilon*-distance of one another, these vertices and the edge in between will be joined. Here the initial, precisely defined, 5-fold symmetrical, procedural specification contained in the original NOME file plays an important role. It guarantees, that the vertices of the few prototype faces added graphically in one fifth of the sculpture, after replication and transformation, will align properly with the border-curve vertices in the other four sectors of the sculpture, so that the merger can be carried out without any problems.

An efficient, hierarchically flat, winged-edge data structure is now constructed to represent the emerging 2-manifold (Fig.4a,b). This data structure is used subsequently to perform mesh-refinement through two to four steps of Catmull-Clark subdivision [4]. This finely tessellated surface can be shown with or without surface normal vectors displayed at all vertices. This is the moment to inspect visually the generated surface for proper connectivity and for any undesirable spots with unacceptably high curvature. Furthermore, as long as we rely simply on Catmull-Clark subdivision for surface refinement and smoothing, the way the designer selects quadrilaterals affects the final geometry of the surface. Constructing long and narrow quads with an aspect ratio of 3 or more, will typically hamper the formation of a nicely balanced

saddle as one would find in a perfect minimal surface. Such flaws will show themselves at this stage and can be ameliorated by selecting facets with lower aspect ratios and by slightly changing some of the geometrical parameter values.

If the shape of the surface is satisfactory, the designer can now further smooth the surface with additional steps of subdivision and subsequently give it some "physicality" by constructing (blue) offset surfaces on both sides of the smoothed 2-manifold (Fig.4c). Along the border curves, the two surfaces are connected to one another with strips of small (red) quadrilateral facets.

Now there is a final opportunity to optimize the aesthetic appeal of the sculpture in its completed form. Perhaps the z-undulation amplitude of the knot may be enhanced to give the whole design more of a 3D character than was present in the earlier 2.5D configuration, which was more convenient for initial surface construction. A larger undulation (Fig.4d) will in general increase the distances between crossing border curve segments and soften the saddle geometries in the regions of the projected knot crossings. Being able to still change the geometric parameters of the knot curves, permits the designer to resolve some specific local trouble spots, as may be caused by two thickened surface elements getting too close to one another, or some saddles becoming too twisted. It is the responsibility of the designer adjusting the various sliders to watch out for unintended, bad side-effects, where solving one local problem will cause another one somewhere else. In Figure 5a the inner hole of the knot has been enlarged dramatically; but this now causes the inner five branches of the border curve to pass through the surface itself. This is not acceptable for the current design where we want to obtain an intersection-free 2-manifold.



(a) (b) (c) (d)

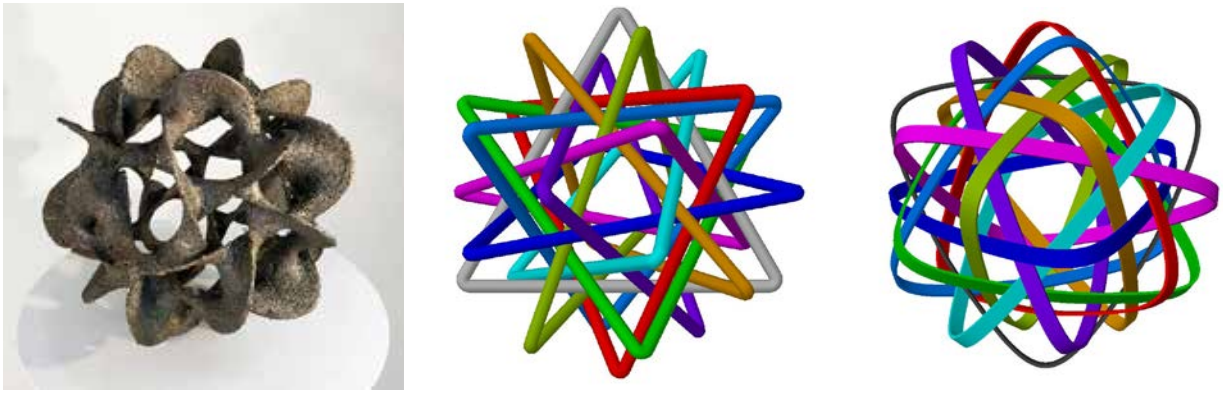
Figure 5: *Parameterized topological changes: (a) excessive enlargement of central hole; (b) topology change due to a reduced rim radius; (c) hierarchical view of new topology; (d) more geometrical changes: narrowing the cinquefoil lobes.*

In general, these adjustment operations are amazingly robust. In Figure 5b, the radius of the outer rim circle has been reduced so much that it now travels through the lobes of the cinquefoil knot. This forces this outer ribbon, which before was a gently undulating annulus, to transform into a twisted ribbon with five full twists. However, the result is still a proper 2-manifold without any self-intersections. At this point, the designer can readily switch back to a hierarchical view of the current design (Fig.5c,d) and possibly start another editing session to make further topological or geometrical changes.

4. A More Challenging Design

Here is a more complex 3D design that uses the power of NOME to the fullest. It was inspired by Charles O. Perry's *Star Cinder* [14] (Fig.6a). It is based on an *Orderly Tangle* [10] of 30 equilateral triangles with (oriented) icosahedral symmetry (Fig.6b). In Perry's sculpture, the pointy triangles have been softened into smoother triangular loops (Fig.6c), and a soap-film-like surface has been draped over the whole assembly of ten interlinked border curves. An inspection of this geometry shows that all the surface geometry is concentrated in a spherical crust in the outermost 30% of the circumsphere of this assembly. Our goal was to create a new generation of "Star Cinders" with a 2-manifold that will fill a larger fraction of the volume of the sphere, by forming several nested and tightly coupled spherical shells. This first requires more border

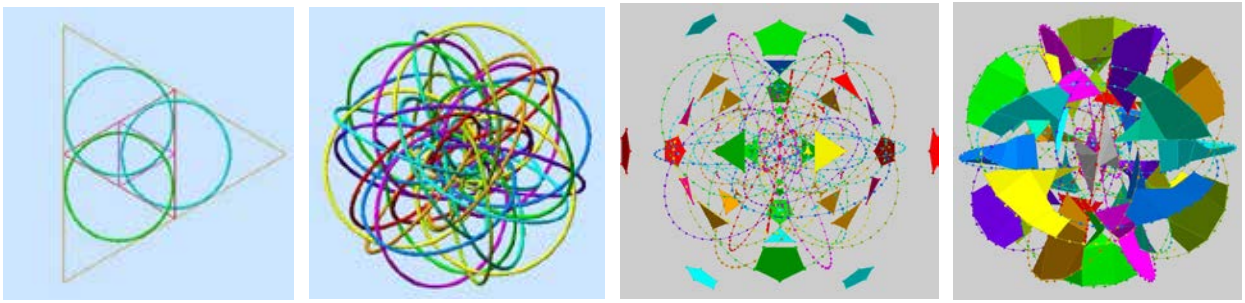
curve segments closer to the center of the sphere. This can be achieved by replacing all triangular loops with more complicated knots with 3-fold symmetry. A fist experiment was done with simple trefoil knots. An even denser tangle of border curves near the center emerges when the trefoil loops are replaced with (3,3)-torus knots, corresponding to sets of 3 symmetrically interlinked circles (Fig.7a).



(a) (b) (c)

Figure 6: *(a) Perry's "Star Cinder;" (b)"0rderly Tangle" of 10 triangles; (c) smooth triangular loops.*

These *"Triple-Loops"* are placed with the same icosahedral symmetry, and they now provide many border curve segments near the center of the sphere. Because of the high degree of symmetry exhibited by the 30 circles (Fig.7b), the pay-off from a hierarchical representation is correspondingly high. Each circle is associated with three geometric parameters (its *radius*, its *tilt-angle* with respect to the symmetry plane of the torus knot, and its *distance* from the center); they are adjusted to avoid any direct curve intersections and to maximize the separation of neighboring border curve segments. The challenge now is to create an intersection-free 2-manifold that uses all curve segments as border loops and maintains the overall icosahedral symmetry. To guide the designers in this task, triangular and pentagonal markers have been placed on the corresponding symmetry axis of the whole assembly (Fig.7c).



(a) (b) (c) (d)

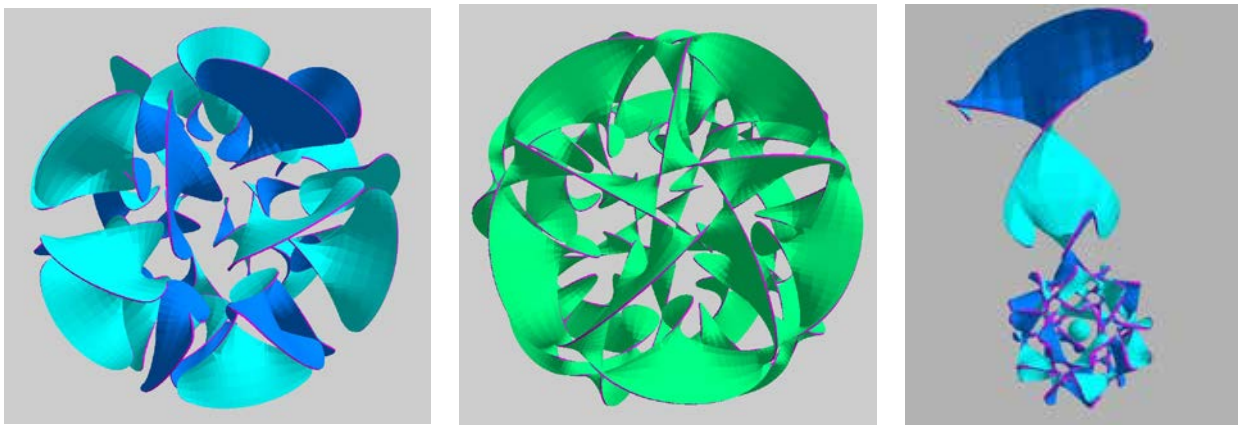
Figure 7: *(a) One Triple-Loop; (b) 10 Triple-Loops; (c) sampled NOME curves; (d) added facets.*

A good way to start manifold construction is by filling in the 30 clearly visible outer lobes by connecting the outermost curve segments in a mostly radial direction to the next inner curve segments encountered (Fig.7d and 8a). This explicit, manual construction needs to be done for only one single lobe, if the symmetry operations among the edges of an icosahedron have been captured properly in the transformations expressed in the 30 instance calls to the primary lobe. Pretty soon, a complete outer framework of twisted ribbons emerges (Fig.8b). On this framework, it is not too difficult to see where radial connections to the inner shells can be made (Fig.8c).

The inner portions are more difficult to connect properly; the tangle of border curves here is much denser; the various curve segments are always seen against much more cluttered surroundings; and the outer

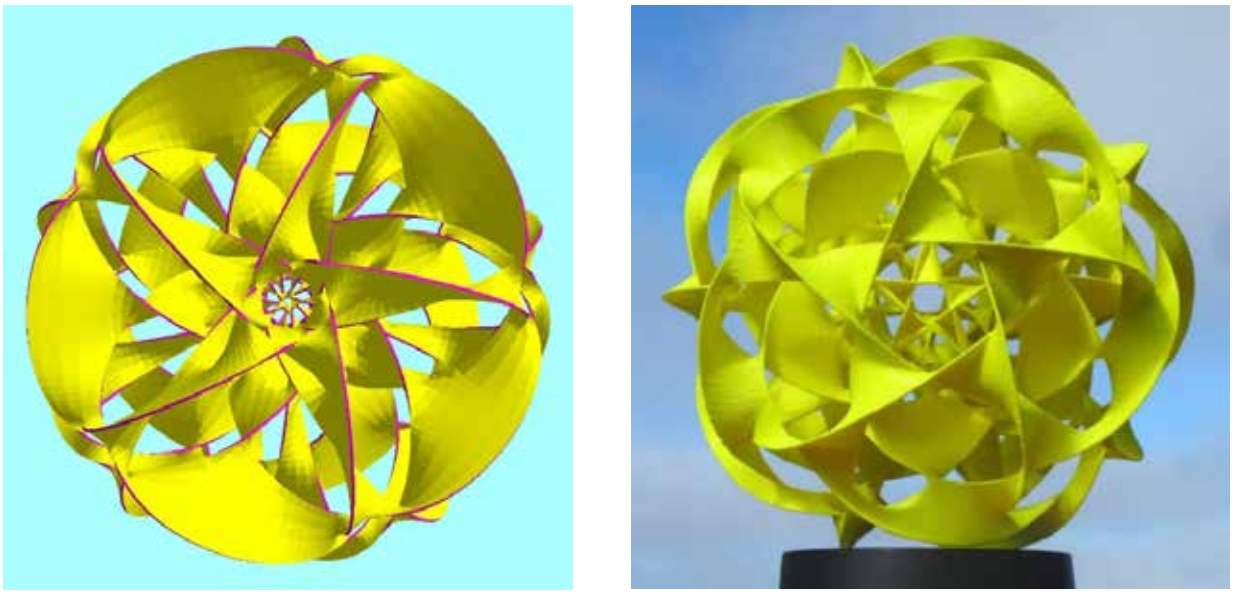
framework blocks visibility to the locations where new facets need to be added. Here the hierarchical procedural description provides some convenient help. Some of the listed instance calls that generate the 30-fold symmetry, or some of the facets in the basic mesh segment that is repeated 30 times, can readily be commented out, leaving just enough displayed geometry to serve as reference and to provide the needed set of vertices for the next set of facets to be added, while providing good visibility to the inner parts.

The next phase focuses on the construction of the innermost shell of the complete 2-manifold. For this phase, much of the outer framework may be turned off; however, it is advisable to keep visible the area where the prototypical lobe mesh was edited, so that the new inner mesh portions are constructed directly below it, and those regions can eventually be joined and connected to one another (Fig.8c).



(a) (b) (c)

Figure 8: *Construction of a 3-level Star Cinder: (a) 30 outer lobes, (b) laterally connected into the outermost shell; (c) one prototypical radial stem connecting to the innermost shell.*



(a) (b)

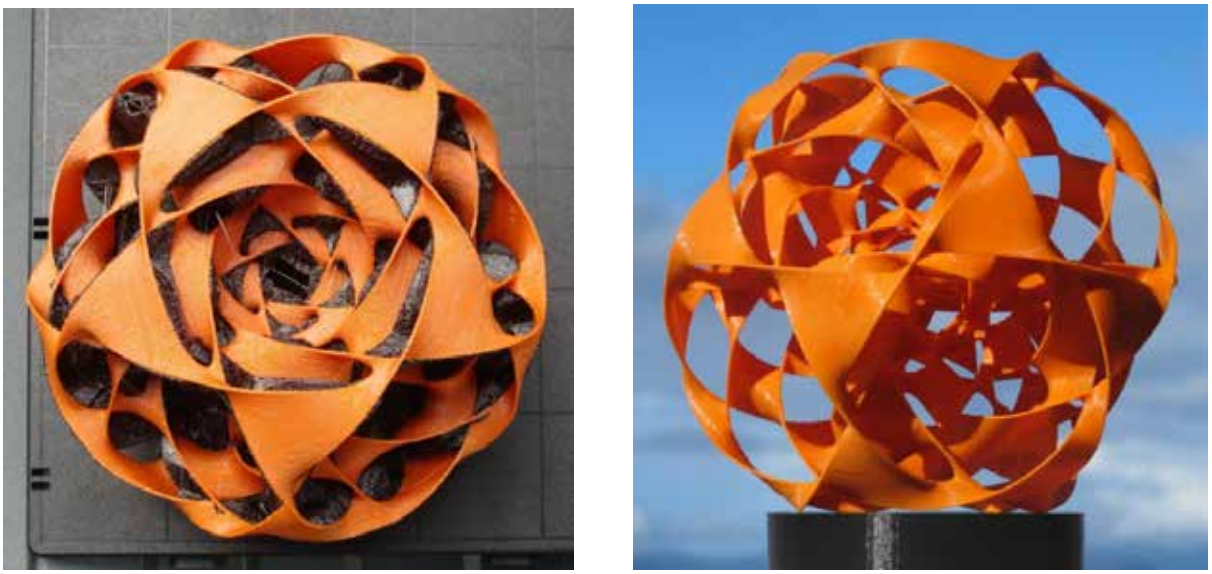
Figure 9: *3-level Star Cinder: (a) complete CAD model; (b) 3D ABS print (18cm diameter).*

Figure 9a shows the complete CAD model with all three nested shells of lateral connections. Figure 9b shows a 3D print made in ABS plastic on a *Dimension 1200* machine from Stratasys [18], after some of the parameters that define the radius and the tilt of the 30 border circles have been slightly changed so as to

produce narrower outer flanges and a somewhat enlarged center section. The internal support structure needed during layered fabrication consisted of wax (P400SR), which then was removed cleanly by immersing the 3D print in a concentrated sodium hydroxide solution for several hours.

This was actually a rather difficult design. Near the center of the sphere, the dense tangle of the 30 border circles made it difficult to figure out how to place the representative quadrilaterals and what border curves they should connect. About eight trial iterations were needed: placing some tentative facets; then letting the procedural specification instantiate them with the full icosahedral symmetry; and then removing them again if they resulted in some inacceptable intersections. The process stretched through 3 or 4 session over a couple of days; it also involved making slight changes to the border circles, hoping that this would make it easier to see where connecting facets should be placed.

The problem is that there is no unique solution to creating a fully symmetrical 2-manifold in such a tangle of border curves. A slight change in the tilt angle of the 30 circular border curves with respect to the corresponding dodecahedron edges will readily accommodate a topologically quite different soap-film surface. The result shown in Figure 10 now has five nested shells, but of somewhat simpler geometry. This model was also fabricated on a *Dimension 1200* fused-deposition-modeling machine from Stratasys, using ABS plastic with soluble wax support.

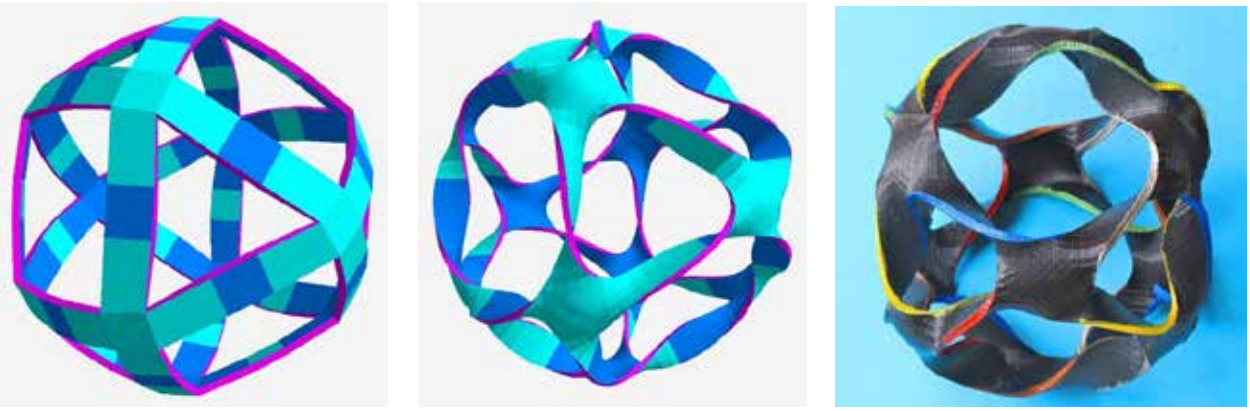


(a) (b)

Figure 10: *5-level Star Cinder: (a) 3D ABS print coming off the printer; (b) with support removed.*

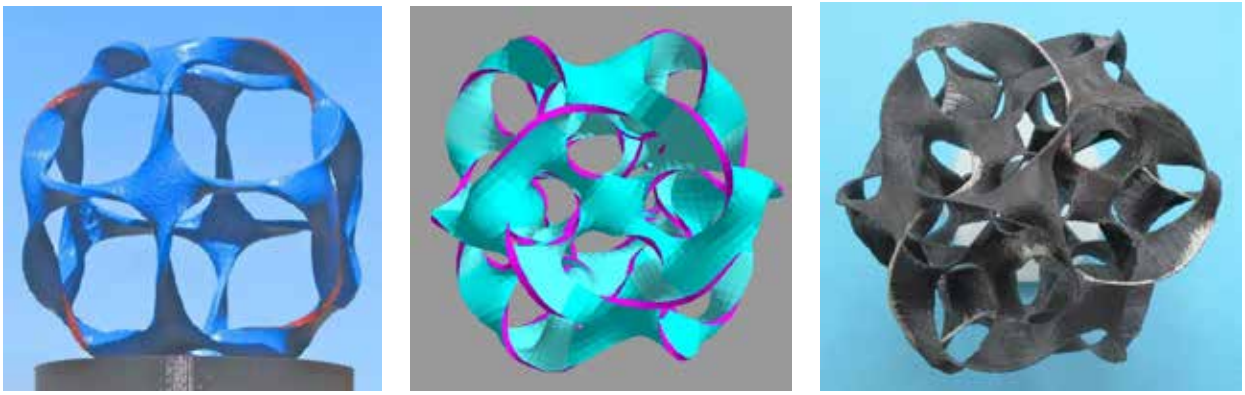
5. A Different Approach to Making *Star-Cinder* Geometries

Because of the difficulties with placing appropriate surface elements that result in a proper 2-manifold with no self-intersections and which exhibits the full symmetry of the oriented icosahedron, we also looked for other ways to design such multi-shell geometries. An "obverse" approach starts with a nested framework of twisted ribbons following the edges of a semi-regular polyhedron and then tries to interconnect concentric shells with additional ribbon elements running in the radial direction. These radial connections are particularly easy to define between two nested shells that are geometrical duals of one another. As one example, we started with a ribbon structure following a cuboctahedron (Fig.11a). All the ribbons connecting neighboring vertices were twisted through 180° (Fig.11b). In this approach, the topology of the surface is preconceived; on the other hand the resulting structure of the border curves is far from clear. Even on the simple geometry of the cuboctahedron frame, it was difficult to visualize the shape of individual border loops. Clarity was gained by physically painting different border loops in different colors. On the twisted cuboctahedron frame, there are 6 border loops with 4-fold symmetry (Fig.11c; see yellow color).



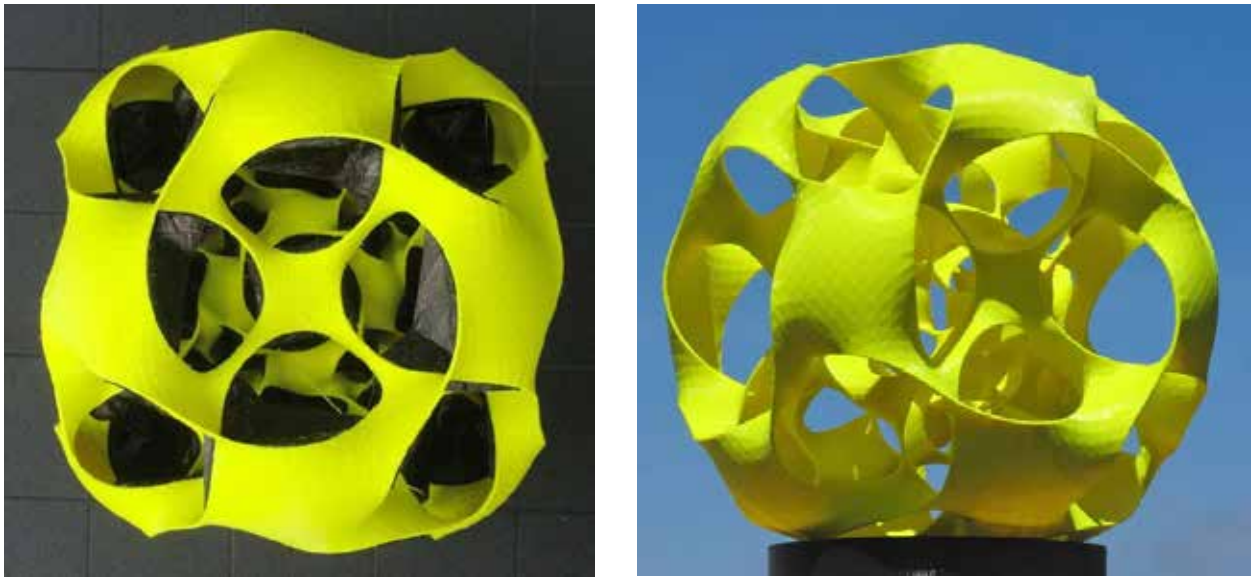
(a) (b) (c)

Figure 11: *Cuboctahedron frame: (a) with flat ribbons; (b) with twisted ribbons; (c) 3D print.*



(a) (b) (c)

Figure 12: *(a) Rhombic dodecahedron frame; (b) combined CAD model; (c) 3D print of 2-level surface.*



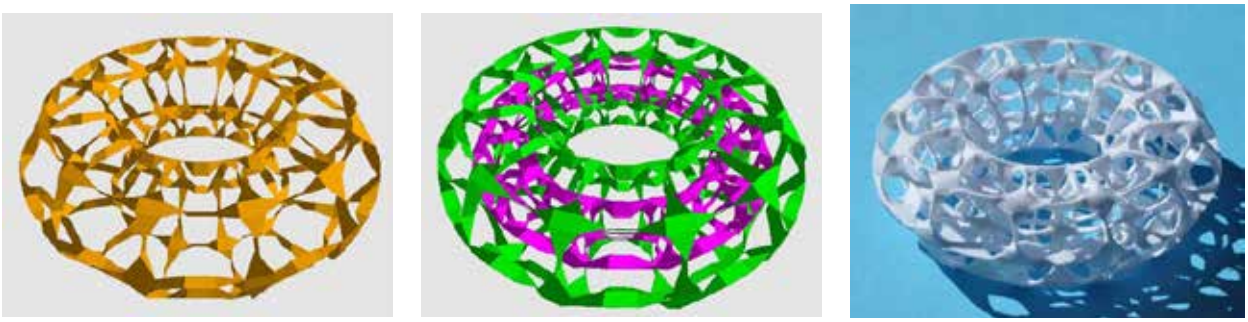
(a) (b)

Figure 13: *3 shells of twisted ribbon frames: (a) as it comes off the printer; (b) with support removed.*

The dual structure is the rhombic dodecahedron. On its twisted frame structure, there are also 6 border loops with 4-fold symmetry (Fig.12a). Next, a scaled-down version of it is placed inside the cuboctahedron frame. At the lined-up edge midpoints, both dual ribbons crossing at right angles have a radially oriented tangent. Thus, it is easy to connect them with a short ribbon segment that twists through 90° (Fig.12b). On the resulting combined 2-shell surface (Figs.12c), the border loops have 3-fold rotational symmetry.

With this approach, it is easy to make star cinders with three or more levels of nested shells. Once the twisted ribbon frames have been generated for a pair of dual polyhedra, more multi-shell surfaces can readily be constructed by nesting scaled version of the two shells in an alternating sequence. Once the two levels of connecting radial tabs have also been defined, those quadrilaterals can also readily be replicated with simple uniform scaling. Figure 13 shows a 3-level structure with a cuboctahedron frame on the outside. This model was also fabricated with soluble support. Figure 13a shows how it emerges from the printer, with the dark wax support still in place; Figure 13b shows the final sculpture after the support structure has been washed away.

This approach is not limited to shells of genus zero. In Figures 14, the same technique has been applied to a torus surface. The surface is partitioned into an array of 12□6 quadrilateral facets, and the edges in this wire frame are replaced with twisted ribbons (Fig.14a). A skinnier torus with the same major radius is placed inside, with the tessellation of its surface shifted by half a facet size in both directions of its surface coordinate system. Thus again, the two frames can be seen as duals of one another with corresponding edges crossing at right angles (Fig.14b). At these crossing points, where the set of ribbons on the outer, as well as on the inner torus, have surface tangents that point in the radial direction, connecting flanges between the outer and the inner frames are introduced; they are just single quads that twist through 90° as they sweep in the radial direction. In this case, two sets of six quads had to be introduced in the graphical display, oriented in the longitudinal direction as well as in the latitudinal direction, respectively. Both sets follow one minor circular loop in the torus. The set of those 12 facets is then put into the group that replicates one such minor-loop assembly twelve times around the *z*-axis (Fig.14c). With this approach, we knew exactly what had to be done, and the whole process took only about five hours, spread over two sessions in two consecutive days.



(a) (b) (c)

Figure 14: *Twisted torus frames: (a) outer torus; (b) concentric tori; (c) 3D print of 2-level surface.*

6. NOME Development

After a couple of somewhat ad-hoc implementations of NOME, the front-end of the current NOME system is structured like a compiler. A parser scans the textual input and constructs an abstract syntax tree (AST), whose nodes are the statements and expressions. From this information, NOME builds the directed, acyclic scene graph (DASG). During the rendering process, the DASG is expanded into the fully instantiated scene tree. All nodes in the DASG have pointers back to the AST to remember where they came from. Through these pointers, any edits that a user makes interactively can be traced back to the input NOME file and can be recorded with minimal deviations from what the user wrote originally.

Update Efficiency

Typically, our hierarchical descriptions comprise a few hundred to a few thousand vertices. Today's (lap-top) computers are fast enough to rebuild the whole scene from scratch within a fraction of a second. Thus, a continuous slider change will appear as a smooth deformation of the overall assembly. However, after the *merging* operation and a few levels of subdivision, the instantiated scene tree is much larger, and it may contain hundreds of thousands of vertices. Even small parameter changes may dislocate previously coinciding edges, and thus prevent a merger of border edge segments. For consistency, the merge operation has to be redone after the hierarchical geometry description has been updated. Merging, subdivision, and offsetting are costly operations, and here it does pay to determine exactly what portions of the scene geometry must be reconstructed.

In the current NOME, a dependency graph co-exists with the DASG and shares many of the same nodes. Every object in the dependency graph may have one or more inputs and outputs. For example, a *point* object has an output that tells its downstream connections about the *position* and the *name* of the point. In order to maintain an interactive frame rate, we separate the update process into two phases: *marking dirty* and the *actual update*. Every object output has a flag that denotes dirtiness, indicating the need to update it before its values can be used. Whenever an output is marked dirty, the dirty flag propagates downwards to all inputs that it is connected to. Each of those object inputs then decides, based on a predefined function per input, whether to propagate the dirty flag further down the dependency graph. As an example: Any user interaction with a slider will mark its *value* output dirty. A *point* object, that happens to use said value, in turn marks its *position* output dirty, but its *name* need not be marked dirty. The dirty flags then propagate to all faces and meshes that use that point, until, eventually, they reach the scene tree nodes associated with the meshes. During the second phase, the scene tree is traversed again, and any object that previously had been marked dirty is now recalculated from the latest clean nodes in the graph.

During a scene update, certain objects may become deleted. For example, when the user lowers the segment count for a cylinder, many strips of vertices and faces will simply disappear, invalidating all constructs that are dependent on them. Clearly, the program should gracefully notify the user of this fact – and not just crash! In addition to the dirty flag, we also maintain a *validity* flag that tracks the status of the most recent update for each object. The *validity* flag propagates downwards during the update process, and an object will delay its update actions as long as it sees that one of its upstream objects is still invalid.

Deleted Geometry

One of the issues that gave us a headache when trying to re-integrate graphical edits into the original procedural NOME file was deleting some geometry. Adding new geometry does not seem to cause any problems; it can simply be added at the end of the original NOME file in as compact a form as possible. The designer can then move these additional NOME statements to a suitable location in the hierarchical description, so that the new elements become part of an overall symmetrical replication. However, deleting geometry is a different matter!

The designer may realize at some point that the procedural symmetry operation places some faces on top of one another and that some of the prototype faces that were introduced earlier may have to be removed. The graphical operations of selecting such faces and marking them for removal, can easily achieve this. The situation is different, if a designer wants to cut an opening into a torus that was produced by a procedural generator routine. If it is just an isolated facet, the situation is as above: the face can be deleted, and all vertices stay in place. The question arises, whether vertices that are no longer needed to support any face, when an extended collection of faces are being deleted, should remain in place or should also be deleted from the data structure. Our current view is that those vertices stay in place and that the "deleted" faces are simply marked *invisible*. This has the advantage that any other constructs that may make use of some of these vertices remain unaffected.

The same question becomes more critical, if some geometry higher up in the hierarchy is being deleted. What if a whole boundary curve is deleted? Will all facets attached to it become obsolete? To minimize

this problem, NOME has no operation in its graphical interface to remove anything except faces. But a corresponding problem still exist when a user changes the number of sampling points along a curve via an interactive slider. If vertex #49 no longer exists, because the number of sampling points has been changed from 50 to only 25, constructs referring to vertex #49 will become ill-defined. The designer will then have to choose a different anchor point. In the current implementation, any geometry that relies on vertex #49 is temporarily marked *invalid* and not displayed until the said vertex somehow shows up later.

7. Potential NOME Enhancements

It may seem rather tedious for the designer to have to refer back to the NOME file in order to integrate some newly added facets into the overall hierarchical structure, or to introduce a new parameterized vertex to provide additional support for a large connecting membrane between multiple border curve segments. During the development of NOME, we have contemplated ways of integrating such geometry-changing operations into the graphical editing domain. Mostly we have concluded that this would be rather challenging, and, in the end, still fairly limiting. Any additional editing capabilities would require a corresponding set of menu options, and the user could then do only what these menus offer to do. Letting the designer go back to the text-based description offers a flexible and less constrained way of enhancing a given design. However, some possible enhancements are currently being discussed and may be implemented in the near future:

Adding Localized Spline Sampling Points:

In the vicinity of a close, skewed crossing of two border curves, where the surface will have to produce a twisted saddle, it is desirable to have sampling points on both border curves near the closest approach of the two curves. Appropriate new sample points could be introduced on demand by the designer. However, there are some conceptual difficulties with naming these vertices and with maintaining full symmetry in the overall design. So far, we have found it easier to just increase the overall sampling density on the border curve and then let the designer use only a subset of these vertices to define new facets and bridging surfaces.

Smoothing of Border Curves:

In the merged, hierarchically-flat scene description, a sequence of border vertices in a coarse polyhedral description of a 2-manifiold could be redefined as the control vertices of a (cubic) B-spline, and this new border curve could then be sampled with the same number of points as there were original "control" vertices. The old border vertices are then replaced with the corresponding new sample points, which now form a smoother border curve. All surface elements are automatically re-attached to the new border curve. This smoothing process could be repeated.

Generating Smoother Surfaces

The final surface geometry depends on the way in which the designer picks facets to define the surface topology. It is desirable to reduce this dependency. There are a few techniques through which this goal can be achieved in the future. In 1992 Henry Moreton described a system for fairing an arbitrary polyhedral surface by computing tangency constraints at all the vertices and then replacing all edges with smooth curve segments that all meet at the vertices with G1-(tangent)-continuity. All original facets get replaced by more complex curved patches [12].

Ken Brakke has developed the *Surface Evolver* [3] to approximate very closely true minimal surfaces. It would be nice to have this capability integrated in NOME and use it in place of the much simpler Catmull-Clark subdivision [4]. This integration will not be trivial, and its use will require additional user interaction. Brakke's surface refinement is not totally automatic; the user will have to use some judgement to initiate alternate refinement steps such as: *equiangulation*, *vertex averaging*, or *jiggling*. Also, the border curves need to be parameterized in a form that is compatible with Brakke's surface refinement process.

8. Discussion and Conclusions

To make possible the design of the complex, multilevel *Star-Cinder* sculptures, we needed an environment that combines parameterized, procedural specification of border curves with an interactive graphical way to define the surface topology suspended on those curves. A key challenge was to maintain an easy-to-understand, high-level description of the current state of the sculpture, which included the graphical edits made through the last interactive session.

The real bottleneck impeding the speed of some of the more complex surfaces remains at the conceptual design level. The times spent in clicking on vertices to add new facets, or switching back and forth between the graphical display and the NOME text file, or fixing up this text file to be in its most succinct form with a clean hierarchical structure, – they all pale in comparison with the thinking time spent observing the current display, rotating it, zooming in and out, in order to figure out the right place to insert the next facet. Thus, while adding additional, higher-level, graphical editing capabilities may be nice, they will not have a big impact on speeding up overall design time.

The development of NOME has stretched over the last three years [19],[6]. It started out as some special "scaffolding" to realize the design of some complex geometrical 2-manifolds; but then it raised some broader conceptual issues about combining procedural specifications with graphical editing capabilities. At this time, the NOME system has been used almost exclusively by the authors of this paper, The presented options may not be the best final solution, but they constitute a reasonable, practical approach for designing some rather complex 2-manifolds. The current development state is presented here to open up a broader discussion on how the integration between two quite different domains may best be achieved.

References

[1] Autodesk. 3DS MAX – https://www.autodesk.com/products/3ds-max/overview

[2] Blender. "Home of the Blender Project." – https://www.blender.org/

[3] K. Brakke. "Surface Evolver." Experimental Mathematics, 1 (2): pp 141–165 (1992) – https://en.wikipedia.org/wiki/Surface_Evolver

[4] E. Catmull and J. Clark. "Recursively generated B-spline surfaces on arbitrary topological meshes." *Computer-Aided Design* 10 (1978), pp 350-355.

[5] R. Chugh, B. Hempel, M. Spradlin, J. Albers. "Programmatic and Direct Manipulation." SIGPLAN 2016 – https://arxiv.org/abs/1507.02988

[6] G. Dieppedalle. "Interactive CAD Software for the Design of Free-form 2-Manifold Surfaces." Tech Report (EECS-2018-48), 5/10, 2018. – https://www2.eecs.berkeley.edu/Pubs/TechRpts/2018/EECS-2018-48.html

[7] N. Gabo. "Translucent Variation on Spheric Theme." – https://www.guggenheim.org/artwork/1381

[8] B. Hempel and R. Chugh. "Semi-Automated SVG Programming via Direct Manipulation." UIST 2016 – https://arxiv.org/abs/1608.02829

[9] E. Hild. "EVA HILD (Home Page)." – http://evahild.com/

[10] A. Holden. "Orderly Tangles." Columbia University Press, New York, 1983.

[11] Maya. "Computer Animation & Modeling Software." – https://www.autodesk.com/products/maya/overview

[12] H. P. Moreton and C. H. Séquin. "Functional Optimization for Fair Surface Design." Computer Graphics 26, 2, July 1992. – https://people.eecs.berkeley.edu/~sequin/CS284/TEXT/p167-moreton.pdf

[13] C. O. Perry. "Dual Universe." – http://www.charlesperry.com/sculpture/dual-universe

[14] P. Perry. "Selected Works 1964-2011." Page 168. The Perry Sudio, Norwalk, CT (2011).

[15] Rhino3D. "Rhinoceros." – https://www.rhino3d.com/

[16] J. Smith. "SLIDE design environment." (2003). – http://www.cs.berkeley.edu/~ug/slide/

[17] C. H. Séquin. "Disk(5,3)" – http://people.eecs.berkeley.edu/~sequin/TALKS/2019/Disk_5_3_CODE.txt

[18] Stratasys. "Dimension 1200es." – http://usglobalimages.stratasys.com/Main/Files/Machine_Spec_Sheets/PSS_FDM_Dim1200es.pdf

[19] Y. Wang. "Robust Geometry Kernel and UI for Handling Non-orientable 2-Mainfolds." UCB Tech Report (EECS-2016-65), 5/12, 2016. – https://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-65.html

Bead Sculptures and Bead-Chain Interlocking Puzzles Inspired by Molecules and Nanoscale Structures

Bih-Yaw Jin
Dept. of Chemistry,
National Taiwan University
Taipei, Taiwan 10617, ROC
byjin@ntu.edu.tw

Chia-Chin Tsoo
National Center for High-
Performance Computing
R&D 6th Road, Hsinchu, ROC
chchts@gmail.com

Abstract

Mathematical beading can be utilized for the construction of aesthetically pleasing sculptures inspired by a rich variety of shapes and forms in the nanoworld. Depending on the shapes of beads utilized, the resulting bead sculptures can be classified into two broad categories: (1) hard-sphere open packing models based on spherical beads, which include finite fullerenes, carbon nanotubes, carbon nanotori, curved three-dimensional graphitic surfaces, and space-filling tetrahedral zeolite structures and many inorganic coordination complexes such as extended metal atom chains; (2) truss-like frameworks constructed from tubular bugle beads, which can mimic the polyhedral representations of microscopic inorganic structures and many macroscopic space frames. Moverover, the hard-sphere open packing model for any polyhedral hydrocarbon of general formula $C_{2n}H_{2n}$ can be achieved by cross-linkings of n pre-made five-bead chains without using any beading process. The assembly processes of certain symmetric polyhedral molecules, such as three Platonic hydrocarbons including tetrahedrane, cubane, and dodecahedrane, are similar to solving interlocking puzzles. Hence, these bead-chain models can also be regarded as a new type of take-apart put-together puzzle.

Introduction

Chemistry is the scientific discipline that deals with compositions, structures, properties of substances and the changes they undergo. Three dimensional structures depicting the arrangements of atoms inside a molecule or the larger-scale hierarchical organization of constituents of nanomaterials are particularly important for a thorough understanding of their physical and chemical properties. Through the advancement of various experimental techniques such as X-ray diffraction, electron microscopy, a large variety of structures and organizations of molecules, aggregates and nanomaterials have been determined. Model building, which has been proven to be a powerful approach for unravelling many important molecular structures in the past, also provides a simple way for understanding spatial relationships for both novices and experts alike.

In this paper, we describe two general approaches to the construction of bead models of molecules and nano structures. [1, 2, 3] In the first approach, we apply systematically the standard right angle weave (RAW) technique (a.k.a. figure-eight stitch) commonly used in the beading community for the construction of a large variety of sculptures inspired by nanoscale molecular structures. We have successfully built many interesting bead sculptures including the 3-connected 3D nets such as cage-like fullerenes, carbon nanotori, helically coiled carbon nanotubes, carbon torus knots, high-genus fullerenes, hyperbolic fullerenes, and triply periodic minimal surfaces, together with many 4-connected 3D nets such as diamond, ices, zeolites, and silicate minerals. [3, 4]

These bead models based on spherical beads can be regarded as the macroscopic realization of valence sphere models for the corresponding molecules, in which the spatial arrangement of spherical beads approximates the three-dimensional electron density distributions. The hard-sphere repulsion among beads and elastic cords that hold beads together mimics the nature of chemical bonds in alkanes, fullerenes, zeolites and many other nanostructures. The saturation and directional character of chemical affinity fall out naturally in bead models. Moveover, constructing the bead model of a molecule with mathematical beading can

be regarded as an analog computation. The result of the computation is the molecule's approximate three-dimensional electron density profile. [5, 6] In addition to spherical beads, we have also utilized tubular bugle beads for constructing many skeletal polyhedral models that can be regarded as physical models of microscopic inorganic structures and macroscopic octet-truss systems. The nodes in these skeletal structures can have connectivity or coordination numbers up to twelve, which is common in many octet-truss structures. [7, 8]

The second approach for the construction of bead models involves no beading at all. Constructing a bead sculpture with mathematical beading technique is usually a slow and tedious process. One needs not only to become familiar with the basic beading techniques, but also to know the algorithmic details for a specific sculpture. While the eventual completion of an aesthetically pleasing sculpture may prove that the effort of the whole process is worthwhile, many repetitive beading steps require a great deal of patience and practice. Therefore, it would be interesting if one can avoid the beading processes, but still keep the essential idea of the hard-sphere open packing in the bead models. Indeed, it is possible to simplify the complicated task of beading for certain molecules into two stages: First, the preparation of the simplest possible beaded structures, namely one-dimensional chains that consist of n beads; second, the assembly of these pre-made bead chains through suitable cross-linkings. The resulting cross-linked bead-chain models, if correctly manipulated, can be utilized to represent valence sphere models of various linear, ring-like, tree-like, and polycyclic alkane molecules, together with many inorganic systems such as perovskite, and so on.

A particularly interesting family of molecules that can be constructed by bead chains is the polyhedrane compound, a cage-like hydrocarbon molecule of general formula $\mathrm{C}_{2n}\mathrm{H}_{2n}$, which has a skeleton corresponding to a polyhedron. A few examples are tetrahedrane, cubane, dodecahedrane, and related prismanes. Moreover, the construction processes for these symmetric compounds have a strong resemblance to solving nontrivial interlocking puzzles such as Burr puzzles: namely starting from a given set of bead chains, each containing a specific number of beads. The goal is to assemble them into a target molecular structure through a sequence of cross-linkings. For instance, two bead chains, each consisting of five beads, can be assembled into the valence sphere model of a tetrahedrane molecule, while four five-bead chains can be put together into the valence sphere model of a cubane molecule. We call this new kind of puzzles as Bead-Chain Puzzles (abbreviated as BC-Puzzles). [9]

Techniques and Algorithms: Degree Three Polyhedra

In this section, we describe the basic beading techniques and procedures for making a bead model. [5, 10] First of all, one needs to prepare the raw materials needed for constructing beaded molecules which are just beads and threads (fishing lines or elastic cords). Beads are made in many materials and come in different colors, textures, and sizes ranging from several millimeters to about a centimeter and usually contain a thin channel for stringing. The sizes of beads we used for constructing bead sculptures in this paper are mostly around 6 mm to 10 mm. Every bead sculpture shown here is constructed by beads with the same size. In principle, the same beading technique be easily generalized to make sculptures consisting of different sizes of beads. In order to construct a robust bead sculpture, it is crucial to choose threads with right thickness. A simple criteria for a degree three (cubic) polyhedral graph such as a fullerene is to choose the thickness of threads in such a way that they can go through the thin channel of beads at least three times, but no more than four times. The reason for this choice will become clear after we describe the techniques for making a bead sculpture.

Figure 1 gives the complete instruction on using the RAW technique to make a beaded cube which consists of twelve beads sitting on the midpoints of twelve edges. The whole procedure has six steps and each step consists of creating a four-member ring that represents a square face of a cube.

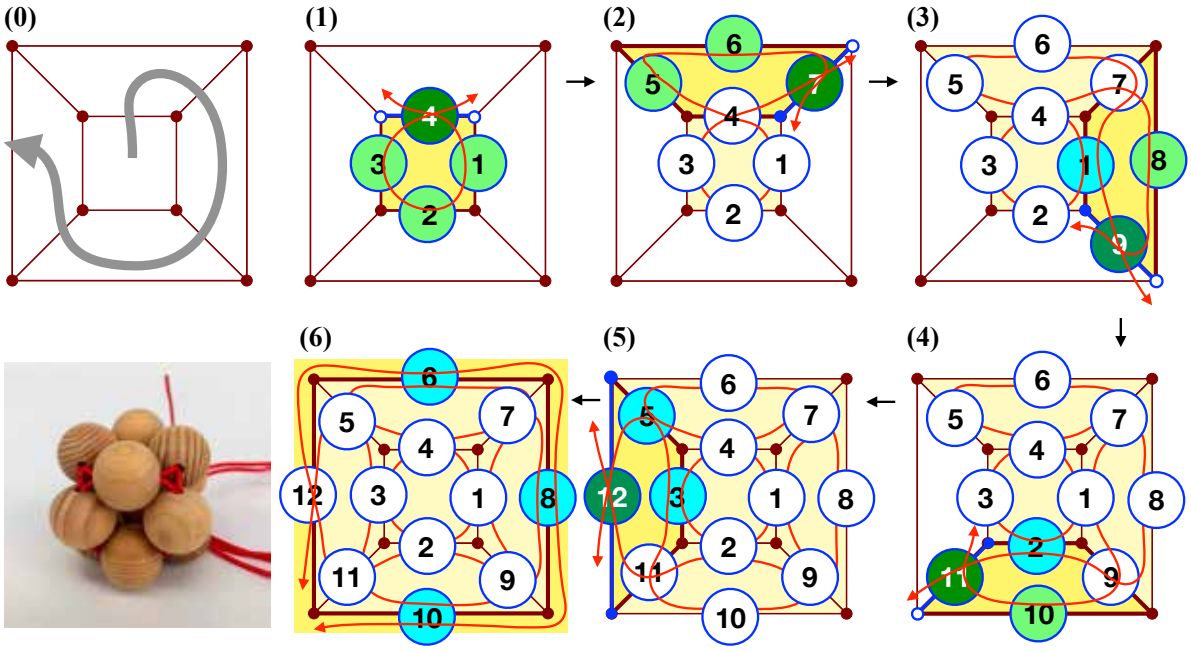


Figure 1: Constructing process of a beaded cube illustrated by using Schlegel diagrams. At each step, blue circles denote the beads just fished and green circles the beads just added. The dark green circle corresponds to the last bead just added and is labeled by the largest numeral. It is also the position where the crossing of two ends of thread take place and will be denoted as the *crossing bead*.

1. (a) String four beads ($\#1 \sim \#4$) in the middle of the thread.
   (b) Take the right end and insert it through the furthest bead ($\#4$) in the opposite direction and pull the thread tight. This makes the first square. For convenience, beads are labeled in ascending order according to the sequence they are added.
2. (a) Add three beads ($\#5 \sim \#7$) on the left end of the thread.
   (b) Take the opposite end and insert it through the furthest bead ($\#7$) in the opposite direction and pull the thread tight. This makes the second square. The first two rings look like a figure eight.
3. (a) In order for the next square to be connected with the two previous existing rings, the thread must go through bead $\#1$ in the neighboring ring. Insert the end of the thread at the junction through the closest bead ($\#1$)–this is called *fishing* and this end is denoted as the *fishing end* or *fishing line*. Notice that one will always add beads to the other end (*non-fishing end*) in the following step. Thus, the fishing and non-fishing ends will always alternate and both ends should decrease in length roughly equally.
   (b) Add two beads on the non-fishing end.
   (c) Take the other end and insert it through the furthest bead ($\#9$) in the opposite direction and pull the lines tight. This makes the third square. This structure can be viewed as a hemisphere now.
4. (a) Go fishing ($\#2$).
   (b) Add two beads ($\#10$, $\#11$) on the non-fishing end.
   (c) Take the other end and insert it through the furthest bead ($\#11$) in the opposite direction and pull the thread tight. This makes the fourth square.
5. (a) Go fishing ($\#3$, $\#5$).
   (b) Add the last bead ($\#12$) on the non-fishing end.

(c) Take the other end and insert it through bead #12 in the opposite direction and pull the thread tight. This makes the fifth square.

6. Go fishing (through beads #6, #8, and #10) and complete the whole structure.

The whole procedure can be summarized in a tabular beading form as given in Table 1. At each step, decision should be made on which neighboring beads needs to be fished first, and then beads can be added to the non-fishing end in order to keep the beading following along the spiral direction. Although we usually view the spherical beads as edges of the corresponding polyhedron, one can also connect the center of neighboring beads and visualizes the bead model as a rectified polyhedron, in this case, a cuboctahedron. If we follow the RAW technique and keep attention to pulling tight two ends of the thread around the crossing bead at the end of each step, a robust beaded structure will be created automatically.

Step i	1	2	3	4	5	6
# of beads fished, f_i	0	0	1	1	2	3
# of beads added, n_i	4	3	2	2	1	0
# of beads in a ring, p_i	4	4	4	4	4	4

Table 1: Tabular beading form for creating a beaded cube. The last row corresponds to its spiral face-Hamiltonian path, which can be codified by the sequence "444444".

According to the many workshops we have given, the most difficult step for beginners in constructing a beaded structure is *fishing*. The trick is to examine two ends of the thread that just coming out of the crossing bead every time one just finished a ring. Notice that the crossing bead at each step is always the last bead that has been just added! In Figure 1, a blue line segment is used to denote the channel direction of the crossing bead at the end of each step. The two ends of this segment correspond to two neighboring vertices of the degree three polyhedron. For instance, when the square 2 (the second step in Figure 1) is just complete, the particular end of the line that points to the direction of the solid blue circle must be the fishing line, while the other end pointing to the vertex denoted by empty blue circle has one edge without any bead on it must correspond to the non-fishing line. So the rule for deciding whether *fishing* should be performed or not is to check if the corresponding vertex (as indicated by the solid blue dot in the second step of Figure 1) is surrounded by three beads. If it is surrounded by three beads, go fishing, meaning that the cyan circle labeled by #1 as shown in the third step in Figure 1 should be fished before adding new beads.

The same beading method can be easily extended to other degree three polyhedron that contains a spiral face-Hamiltonian path, which visits each face once and only once along the spiral. For instance, a spiral face-Hamiltonian path exists for the great rhombicuboctahedron and can be codified as "8 46464646 84848484 64646464 8". Without employing its Schlegel diagram or even the corresponding tabular beading form, this code already gives enough information for creating its bead model. One simply follows this code and start to construct a bead model by making one polygon after another either clockwise or counterclockwise. Special attention has to be paid to *fishing* before adding new beads at each step to ensure that the condition of a degree three polyhedron is satisfied. When the last ring is formed, an overall structure that gives a faithful three-dimensional bead representation for the corresponding polyhedron will be automatically created. In Figure 2, we show the bead models for four degree three polyhedra.

More generally, the algorithm for making any degree three polyhedron that contains a spiral face-Hamiltonian path specified by the code $p_1p_2 \dots p_N$, where p_i is the number of sides of the ith polygon along the spiral, can be written as follows:

1. $i = 1$

   (a) Add $n_1 (= p_1)$ beads to the string.

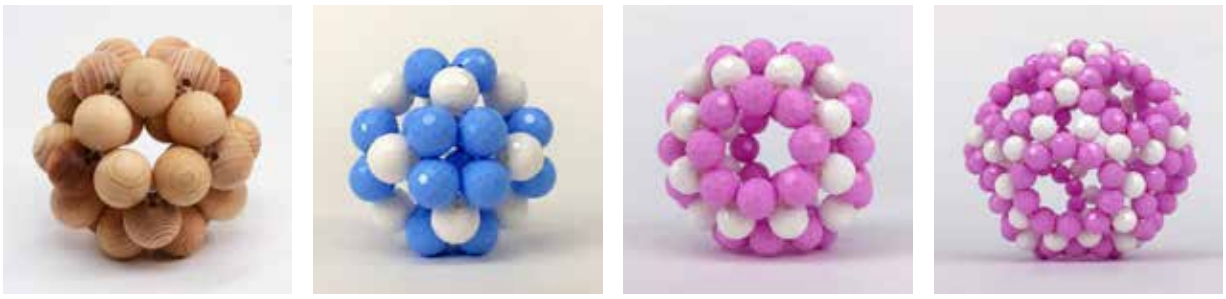


Figure 2: Bead models for dodecahedron, truncated octahedron, great rhombicuboctahedron, and great rhombicosidodecahedron.

(b) Form a ring with p_1 beads by crossing both ends of the string at the crossing bead labeled by the largest numeral (i.e. the furthest bead or the last bead just added).

2. $i = i + 1$
   (a) Fishing: $f_i = 0$, where f_i is the number of beads fished at the step i.
      i. Fish the next bead if the corresponding vertex is saturated (or surrounded) by three beads, $f_i = f_i + 1$.
      ii. Repeat 2(a)i until all neighboring saturated vertices are fished.
   (b) Add $n_i = p_i - f_i - 1$ beads to the non-fishing end of the string.
   (c) Form a ring that contains p_i beads (p_i-gon) by crossing both ends of the string at the bead labeled by the largest numeral.
3. Repeat 2 until all polygons are formed.

One can easily see that each bead is stitched exactly two times if the corresponding graph contains a face-Hamiltonian path. Therefore, one must choose the thickness of a thread such that it can go through each bead at least two times. However, without further handling of the remaining thread still outside the bead sculpture, the bead sculpture just finished can loosen up easily. The trick in preventing this happening is to string the remaining part of threads lying outside into the bead structure again until the whole sculpture becomes tightly bound. As a result, the channels of some beads will be filled by threads three or even four times.

Bead Sculptures: Curved graphitic structures

The first approach to the construction of bead models is based on the RAW techniques of mathematical beading introduced in previous section. According to the coordination numbers, there are three different broad families of nanostructures that can be constructed using the mathematical beading technique. The first family consists of any cage-like polyhedral fullerene, which is a degree three (cubic) polyhedral graph with only pentagonal and hexagonal faces, and the generalization to various curved graphitic structures that allow non-hexagonal rings other than pentagons. For simple cage-like polyhedral fullerenes C_v without holes (i.e. $g = 0$, where g is the genus), the number of pentagonal faces is always 12, the number of hexagonal faces is $(v - 20)/2$, and the number of edges (or beads) is $3v/2$. The bead model of a fullerene with v carbon atoms comprises exactly $3v/2$ beads since beads stand for chemical bonds instead of atoms.

To make any fullerene, weave face by face with beads using the RAW according to its pentagonal spiral code which specifies a spiral face-Hamiltonian path that visits each polygonal face once and only once. For instance, the spiral code for C_{60}, [1 7 9 11 13 15 18 20 22 24 26 32], gives the positions of twelve pentagons

in the spiral sequence of twelve pentagons and twenty hexagons. By using a prescribed spiral code, one can create a realistic bead model of any cage-like fullerene.

In addition to the cage-like fullerenes, beads can also be used to construct other more complicated graphitic structures such as carbon nanotori, high-genus fullerenes, and so forth. According to our experience, beads can be used to construct robust physical models of simple and complicated carbon nanostructures that are otherwise inaccessible. In Figure 3, we show a variety of curved graphitic structures which include finite and infinite 3-connected graphitic surfaces, starting from the simplest C_{60} to carbon nanotori, helically coiled carbon nanotubes, carbon torus knots, high-genus fullerenes, and triply periodic minimal surfaces with negative Gausssian curvatures. [11]

Figure 3: Curved graphitic structures. First row: Buckyball (C_{60}), endcapped carbon nanotube C_{80}, fullerene with lollipop shape, HIV virus, carbon nanotorus C_{120}; Second row: curved nanotori, helically coiled carbon nanotube, carbon trefoil knot, dodecahedral high-genus fullerene, genus-3 carbon tetrahedron; third row: fused C_{60} cube, carbon tetrapod, single periodic minimal surface, I-WP surface, $(1,0)$-P-surface; Fourth row: $(1,1)$-P-surface, $(2,1)$-P-surface, Chen-Gackstatter surface, a genus-2 surface, intersection for four graphene sheets with four lines of sp^3 defects.

Bead Sculptures: Zeolites and tetravalent graphs

The second family that can be constructed by mathematical beading comprises tetravalent periodic structures with the degree (or valency) of a vertex equal to four, which include diamond, ices, clathrate hydrates,

zeolites, and many minerals. [12, 13, 14] In Figure 4, we give a few examples of bead models corresponding to various zeolite frameworks. Zeolites are porous aluminosilicates, widely used as efficient catalysts, adsorbents, and ion exchangers in petrochemical industries and in our daily life. Mathematicians have predicted that millions of hypothetical zeolite structures are possible. However, only 245 structures have been discovered and synthesized so far.

Figure 4: Zeolites specified by 3-letter Framework Type Codes. First row: SOD (Sodalite), LTA, FAU, EMT, RHO; Second row: KFI, AST, TSC, ATN, BCT; Third row: LTL, MOZ, SAS, ABW, BEC; Fourth row: MEP, MTN, DOH, MWW (MCM-22), SSF.

Zeolites are based on the primary tetrahedral building unit, TO_4, where the central tetrahedrally bonded atom, T, is usually either a silicon or aluminum cation, surrounded by four oxygen anions. By linking these tetrahedra together, it is possible to build larger building units such as truncated octahedra, dodecahedra, etc. When these building units are linked together to form a zeolite framework, they characteristically yield periodic cavities and channels throughout the structure. It is a challenging work to construct all these structures with beads since the beading is an inherently slow sequential process. Right now, we have succeeded in making bead models for more than forty zeolite framework types.

Bead Sculptures: Inorganic structures with valencies higher than four

We have also applied successfully the technique of mathematical beading to the construction of skeletal polyhedral models with the degree of a vertex higher than four for many nanoscale inorganic structures or macroscopic octet-truss systems by using long tubular bugle beads. [7, 8] If bugle beads with the same length are adopted, all eight convex deltahedra can be constructed and the resulting structures are rigid as a consequence of the Cauchy rigidity theorem. Among these convex deltahedra, tetrahedron and octahedron are two most common structural motifs occurring in the molecular transition metal chemistry, inorganic solids, and many mineral structures.

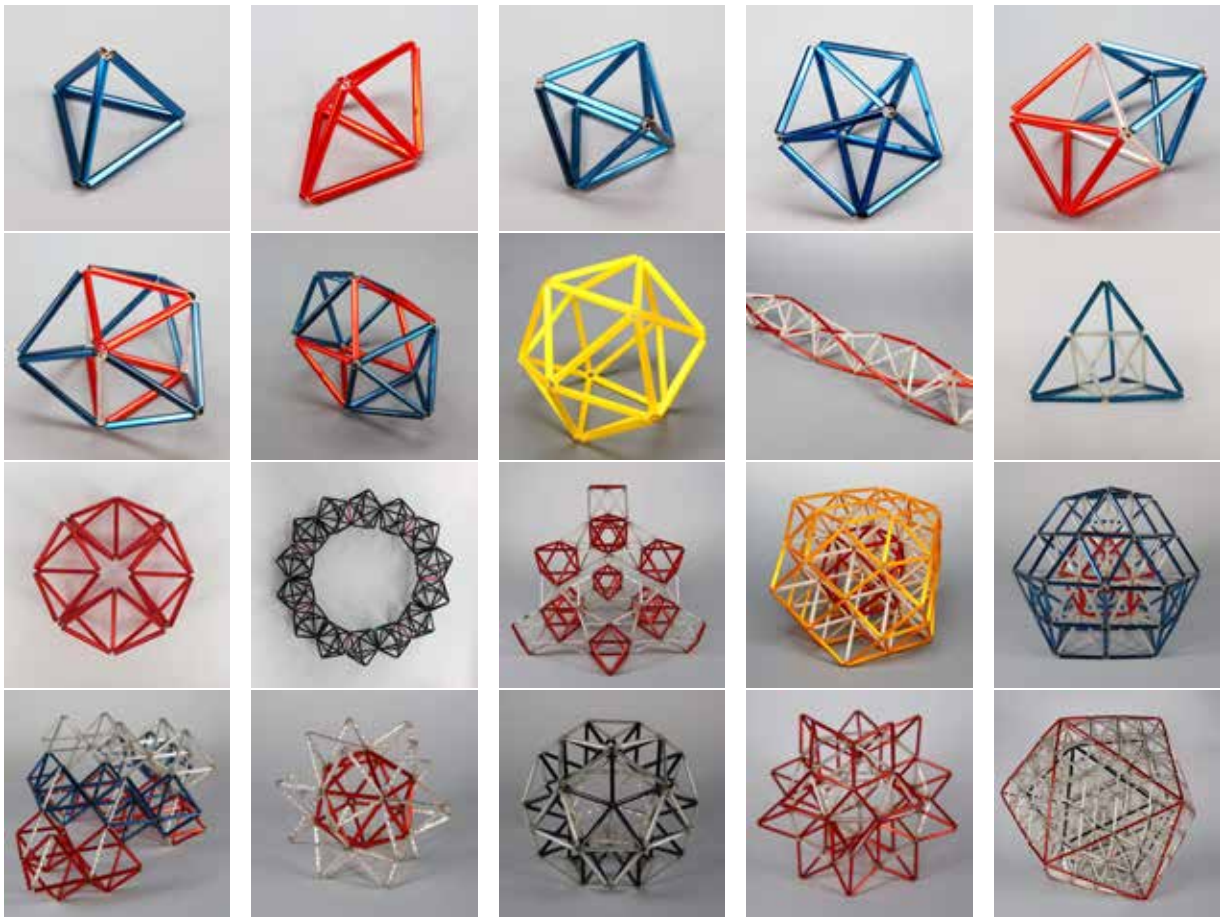


Figure 5: Skeletal truss models. First row: tetrahedron, trigonal bipyramid, octahedron, pentagonal bipyramid, snub disphenoid; Second row: Triaugmented triangular prism, gyroelongated square bipyramid, Icosahedron, Boerdijk-Coxeter helix; Third row: Kaleidocycle with eight regular tetrahedra, and Kaleidocycle with 14 gyroelongated square bipyramids, octet-truss consisting solely of octahedra, hexagonal close packed octet truss, cubic close packed octet truss; Fourth row: Anatase TiO_2 structure, elevated icosahedron, compound of icosahedron and 20 octahedra, compound of icosahedron and rhombic hexecontahedron, frequency-4 Mackay icosahedron.

Other deltahedra can sometimes become important in nano world. For instance, the B_{12} motifs in the extended structure of α-rhombohedral boron are found to be icosahedra. Many larger molecular structures can be described by linked deltahedra through sharing common vertices, edges, or faces. If bugle beads with different lengths are used, a richer structural variety of skeletal polyhedral sculptures can also be built.

Figure 5 shows a few skeletal truss models for inorganic systems and space frameworks based on linking different numbers of deltahedra.

Bead-Chain Construction Set and Interlocking Puzzles

Making a bead model for a large molecule is inherently a slow sequential process, and can take a lot of time to complete except for certain modularizable systems which could be constructed by many people together. In this section, we show that it is possible to create the same valence sphere models without any beading, at least for certain molecules such as polycyclic cage-like hydrocarbon with the chemical formula $C_{2n}H_{2n}$, or polyhedrane. Starting from several pre-made building blocks, the valence sphere models for these polyhedrane molecules can be assembled through cross-linkings and the tedious beading procedure can be avoided completely.

Consider the simplest polyhedrane, a cubane molecule with the formula C_8H_8. This molecule consists of eight carbon atoms arranged at the corners of a cube, with one hydrogen atom attached to each carbon atom. [15] Also there are twelve CC single bonds and eight CH bonds in this molecule. Notice that the bead model for this molecule corresponds to its valence sphere packing model. Atoms are not explicitly shown, while twenty chemical bonds are represented by twenty beads. To make a bead model for this molecule with a finite number of linear bead chains through cross-linkings, the beads corresponding to eight CH bonds must be located at chain ends. Hence, exactly four bead chains are required in order that a chain can start from one CH bond and end at another CH bond. Hence, CH bonds are represented by terminal beads. To use as few basic building blocks as possible, we choose $12/4 = 3$ non-terminal beads for each chain. Thus, a single type of five-bead chain is sufficient for making a bead model of a cubane molecule. Interestingly, the assembly process of a cubane model from four five-bead chains is nontrivial since the many possible ways of making cross-linkings among four five-bead chains exist. In many ways, assembling the bead model of a cubane molecule bears a resemblance to solving an interlocking puzzle. Therefore, we call this model as the BC-Cube (Bead-Chain Cube) puzzle as shown in Figure 6 and 7a.

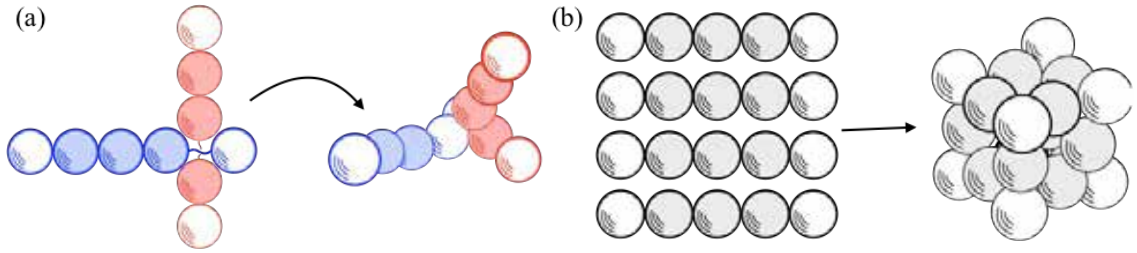


Figure 6: (a) Cross linking of two bead 5-bead chains. (b) A BC-Cube can be assembled from four five-bead chains through eight cross linkings.

Figures 7b and 7c shows the bead-chain models of the other two Platonic hydrocarbon molecules, namely tetrahedrane and dodecahedrane, which can be assembled from two and ten five-bead chains, respectively. Bead-chain models for two of seven Archimedean polyhedranes, truncated tetrahedrane ($C_{12}H_{12}$) and truncated icosahedrane ($C_{60}H_{60}$), are shown in Figures 7d and 7e, respectively. More generally, the carbon skeletons of polycyclic hydrocarbon with formula $C_{2n}H_{2n}$ can be regarded as cubic graphs in which all vertices have degree three. If we ignore CH bonds, each carbon atom is bonded exactly to three neighboring carbon atoms. A cross-linking between two bead chains introduces a local tetrahedral arrangement for four beads. To satisfy the requirement that carbon atoms are located on the polyhedral skeleton and hydrogen atoms pointing outward, one needs exactly n five-bead chains for making a bead model of the corresponding polyhedrane molecule. Finding a procedure of making a bead-chain model for a polyhedrane with five-bead

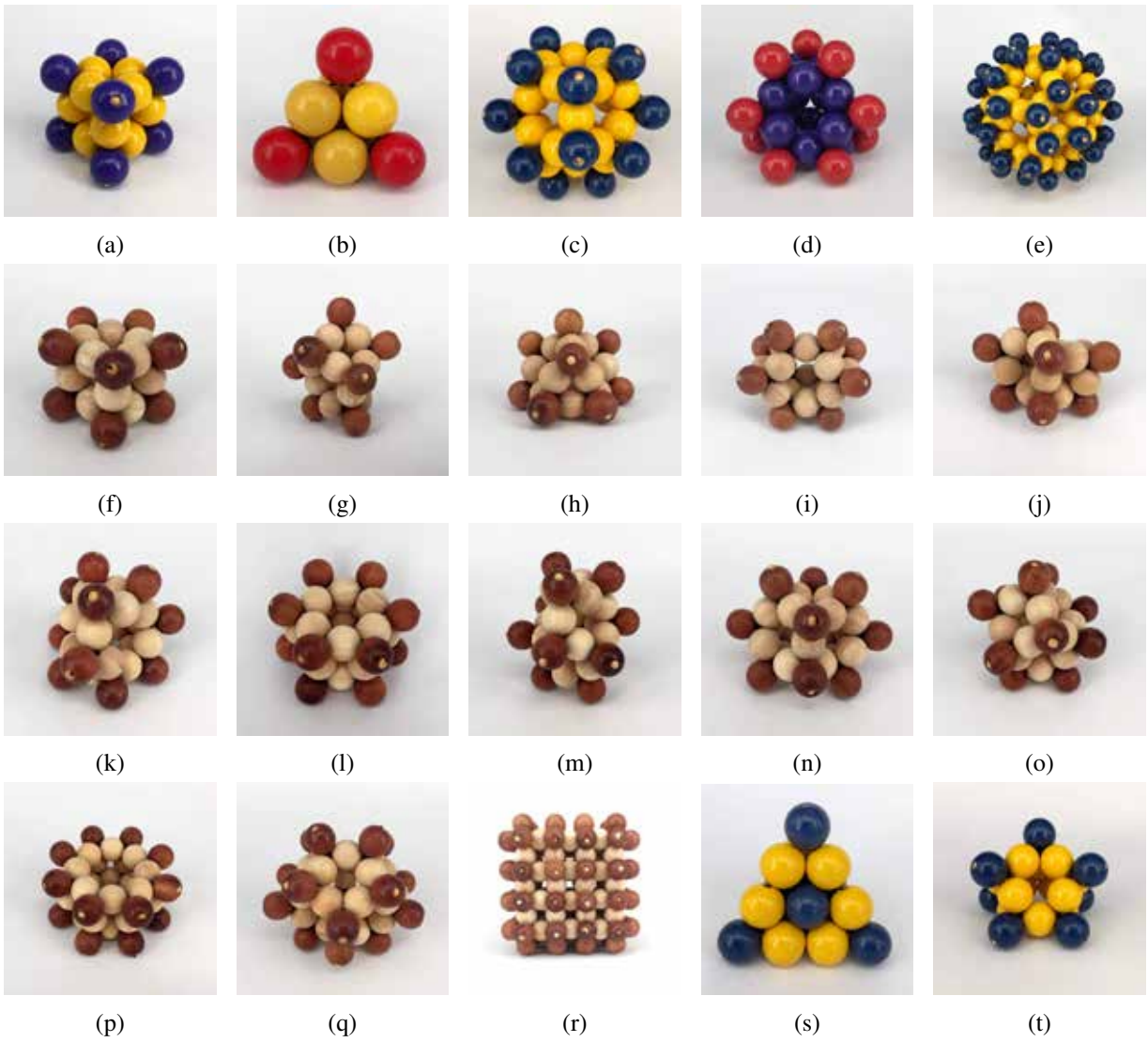


Figure 7: Bead chain construction set. First row: BC-Cube (C_8H_8) with four five-bead chains, BC-Tetrahedron (C_4H_4) with two five-bead chains, BC-Dodecahedron ($C_{20}H_{20}$) with 10 five-bead chains, BC-truncated tetrahedron ($C_{12}H_{12}$) with 6 five-bead chains, BC-Buckyball ($C_{60}H_{60}$) with 30 five-bead chains; Second row: fiver isomers of $C_{10}H_{10}$; Third row: five isomers of $C_{12}H_{12}$; Fourth row: $C_{16}H_{16}$, $C_{18}H_{18}$, Perovskite structure constructed by 48 five-bead chains, close-packed tetrahedral arrangement of twenty spheres, another isomer of C_8H_8 that contains two triangular faces and two pentagonal faces.

chains is equivalent to finding a way to draw its skeletal polyhedron with n non-overlapping strokes and each stroke covers exactly three edges.

Since the total number of cubic graphs grows exponentially as a function of n, there is a rich variety of possible structures that can be constructed by this new kind of bead chain construction set. In Table 2, we list the number of unlabeled cubic and cubic polyhedral graphs up to 22 vertices. Whenever an n-stroke solution for a polyhedrane exists, the corresponding polyhedral bead-chain model can be constructed.

Table 2: Number of unlabeled cubic polyhedral and triangle-free cubic polyhedral graphs with $2n$ nodes, with $n = 2, 3, \ldots, 11$.

Vertex	4	6	8	10	12	14	16	18	20	22
Cubic	1	2	5	19	85	509	4060	41301	510489	7319447
Polyhedral	1	1	2	5	14	50	233	1249	7595	49566

Moreover, if we allow cross-linkings of three bead chains at a single position, more complicated nanostructures that contain local octahedral arrangements of six spheres can also be created. For instance, Figure 7r gives a bead model for a portion of the perovskite structure that contains $4 \times 4 \times 4 = 64$ unit cells. A total of 48 five-bead chains with 64 triple-chain crossings are used. Finally, notice that the same four five-bead chains, which we used for making the bead model of a cubane molecule, can also be used to make another two interesting structures, one is a close-packed tetrahedral arrangement of twenty spheres (Figure 7s) and another one corresponds to a different isomer of C_8H_8 (Figure 7t)! We leave this as a puzzle for readers.

Conclusion

In conclusion, chemistry is rich with many interesting structures which can be employed for constructing aesthetically pleasing bead sculptures. With suitable interpretations for the meaning of beads and strings, these sculptures can model many microscopic molecular structures faithfully. More importantly, we believe that the molecular and structural diversities at the nanometer scale can also give us inspirations for finding new forms and shapes of sculptures and architectures.

Two different approaches for constructing bead sculptures, one by the mathematical beading and the other through the bead-chain cross-linking, are presented in this paper. The first approach based on mathematical beading can be applied to a large variety of nano structures to create fascinating bead sculptures. The disadvantage is that the mathematical beading is inherently a slow and sequential process. Great patience is required if one wishes to construct a bead sculpture with thousands of beads. The second approach based on the cross-linking of pre-made bead chains involves no beading at all. We demonstrate that the valence sphere model for any polycyclic hydrocarbon molecule of general formula $C_{2n}H_{2n}$ with carbon atoms located on the nodes of corresponding cubic graph can be assembled from n linear five-bead chains through n cross-linkings, which can be regarded as a new type of interlocking puzzle.

Acknowledgements

We acknowledge the financial support from the Ministry of Science and Technology, Taiwan, R.O.C.

References

[1] B.-Y. Jin, C. Chuang, C.-C. Tsoo. "Constructing Molecules with Beads: The Geometry of Topologically Nontrivial Fullerenes." *Bridges Conference Proceedings*, Pecs, Hungary, Jul. 24–28, 2010, pp. 391–394.

[2] C. Chuang, B.-Y. Jin, C.-C. Tsoo. “Designing Sculptures Inspired by Symmetric High-Genus Fullerenes with Mathematical Beading.” *Bridges Conference Proceedings*, Coimbra, Portugal, Jul. 27-31, 2011, pp. 523–526.

[3] C. Chuang, B.-Y. Jin, W.-C. Wei, C.-C. Tsoo. “Beaded Representation of Canonical P, D, and G Triply Periodic Minimal Surfaces.” *Bridges Conference Proceedings*, Towson, Maryland, USA, Jul. 25-29, 2012, pp. 503–506.

[4] C.-C. Tsoo, C. Chuang, B.-Y. Jin. “Mathematical Beading as Molecular Analog Computation: An Example from Beaded Sierpiński Buckyball.” *Bridges Conference Proceedings*, Enschede, the Netherlands, Jul. 27-31, 2013, pp 487-490.

[5] C. Chuang, B.-Y. Jin, C.-C. Tsoo, N. Y.-Wa Tang, M. P. S. Cheung, L. A. Cuccia. “Molecular Modeling of Fullerenes with Beads.” *J. Chem. Edu.* 2012, 89 (3), pp 414-416.

[6] C. Chuang, B.-Y. Jin. “Construction of Sierpiński Superfullerenes with the Aid of Zome Geometry: Application to Beaded Molecules.” *Bridges Conference Proceedings*, Enschede, the Netherlands, Jul. 27-31, 2013, pp 487–490.

[7] A. F. Wells, Structural Inorganic Chemistry, Oxford University Press, 1984.

[8] C.-C. Tsoo and B.-Y. Jin. “Designing Skeletal Polyhedral Sculptures Inspired by Octet-Truss Systems and Structural Inorganic Chemistry with Bugle Beads.” *Bridges Conference Proceedings*, Waterloo, Canada, Jul. 27–31, 2017, pp. 483-486.

[9] B.-Y. Jin. “Bead-Chain Construction Set and Interlocking Puzzle Inspired by Polyhedranes.” *Bridges Conference Proceedings*, Linz, Austria, Jul. 16-20, 2019.

[10] K. Horibe, B.-Y. Jin, C.-C. Tsoo. “From Sangaku Problems to Mathematical Beading: A Hands-on Workshop for Designing Molecular Sculptures with Beads.” *Bridges Conference Proceedings*, 2014, 503-508.

[11] C. Chuang, B.-Y. Jin. “Torus knots with polygonal faces.” *Bridges Conference Proceedings*, Seoul, Korea, Aug. 14-19, 2014, pp 59–64.

[12] Ch. Baerlocher and L. B. McCusker, Database of Zeolite Structures: `http://www.iza-structure.org/databases/`

[13] C.-C. Tsoo and B.-Y. Jin. “Molecular Modeling of Four-Connected Zeolite Frameworks with Mathematical Beading.” *Bridges Conference Proceedings*, Jyväskylä, Finland, Aug. 9–13, 2016, pp. 375-378.

[14] Y.-J. Fan, B.-Y. Jin, C.-C. Tsoo. “Designing Beaded Sculptures Inspired by Clathrate Hydrates.” *Bridges Conference Proceedings*, Stockholm, Sweden, Jul. 25-29, 2018, pp 555–558.

[15] “Cubane” in *Wikipedia: The Free Encyclopedia*. `https://en.wikipedia.org/wiki/Cubane`.

Data-spatialized Pavilion: Introducing a Data-driven Design Method based on Principles of Catoptric Anamorphosis

S. Vahab Hosseini[1], Hessam Djavaherpour[2], Usman R. Alim[3],
Joshua M. Taron[4], and Faramarz F. Samavati [5]

University of Calgary, Alberta, Canada
[1]seyedvahab.hosseini@ucalgary.ca, [2]hessam.djavaherpour@ucalgary.ca, [3]ualim@ucalgary.ca
[4]jmtaron@ucalgary.ca,[5]samavati@ucalgary.ca

Abstract

Data spatialization is a design technique through which data is used to create architectural spaces. It does not necessarily preserve the legibility of the represented data, but rather focuses on the spatial qualities that can be gained from the data. As a consequence, data in such data-driven spaces tend to be represented in abstract forms. By means of a method of spatial representation that has historically been used in art and architecture, we produce a data-spatialized architecture that preserves data legibility. This research aims to introduce a method for the design of a data-driven pavilion that represents data spatially through catoptric (mirror-assisted) anamorphosis. The major contribution lies at the underexplored intersection of data spatialization and perspectival representation, where the input data defines the physicality of the pavilion and simultaneously remains readable. In this work, a set of environmental datasets from North America – including elevation, precipitation, temperature, and population – is used to generate an anamorphic structure. The spatialized datasets can be updated by means of illuminating the components of the pavilion. Based on the result, this design methodology provides an accurate data representation in an anamorphic data-driven public space.

Introduction

Anamorphic projection attracts our attention by manipulating our spatial perception. As a subset of optically illusive perspectival representation, it highlights the gap between the representation of an object and its reality [21]. In other words, anamorphic projection has the potential to attract its audience and engage them in understanding the discrepancy between the representing and the represented [22]. What motivates this research is to harness the power of anamorphic projection in an architectural design using the aforementioned anamorphic attributes.

In architectural design, the advent of powerful computational techniques has opened up new horizons with the potential to develop new spatial and communicative possibilities by using “data" – i.e., *the input data* – as a design concept. This approach has led to the introduction of data spatialization, which provides architects with the opportunity to physicalize data at an architectural scale. Data-spatialized architectures, however, mandate that data be distributed within a three-dimensional space at an architectural scale [19]. Therefore, data in these spaces become abstract and are not as legible as may be desired. Such abstracted data may become hidden in the built environment, which renders the built environment mute when it could otherwise express itself in visually legible and socially meaningful ways.

To tackle this issue, this paper introduces a data-spatialization design technique that, while maintaining data legibility, employs data in the design of a pavilion through the use of catoptric (mirror-assisted) anamorphosis. The applied methodology maintains the two-dimensional representation of the input data within an anamorphic data-spatialized pavilion. It is based on dissecting the principles of a reversed curvilinear perspectival representation known as catoptric anamorphosis. This representation technique has historically been used to translate broad and often unperceivable images/data into a comprehensible form at the level of

small-scale objects. The objective of catoptric anamorphosis is to communicate wide-angle spatial qualities within a specific field of view. The role of the mirror is to reveal a hidden message (both two-dimensional and three-dimensional) that may otherwise be distributed and therefore illegible in the three-dimensional space.

In this paper, we establish a proof of concept by first designing an anamorphic sculpture of the Mona Lisa painting in a scaleless environment. Next, an architecture is introduced in the form of an anamorphic data-driven pavilion. Hypothetically, any easy-to-grasp type of data is applicable to this specific architecture. In this regard, we use environmental data since it can be understood by a broad range of audiences and engages with public awareness. Our data-spatialization process takes a series of environmental datasets of North America, including elevation, precipitation, temperature, and population data. Since elevation tends to remain consistent over time, it is used to build brick-like components mounted over the shell of the pavilion. The geometry of the shell of this pavilion has to be modified and improved in a way that it is fully reflected through the mirror. Such modification prevents any failure in maintaining the representation of data viewed through the anamorphic projection, and is handled by the design code. To provide an opportunity to update or swap datasets displayed by the anamorphic pavilion, all datasets, including the elevation data, undergo a process to uniquely illuminate every single brick of the pavilion. Such illumination helps in enhancing the dynamic ambience of this publicly-accessible privileged space.

Our main contribution is to introduce and develop a design methodology for a data-spatialized pavilion where the represented data remains legible, thanks to an anamorphic projection. This technique is expected to trigger public awareness towards the principles of catoptric anamorphosis, as a historical method of spatial representation in art and architecture. In a public space, such as the proposed pavilion, the audience would communicate with a series of represented environmental issues as a consequence of understanding how this projection works and how it is applied in building the pavilion.

Background

Designers can augment data-driven spaces in terms of legibility of the input data, manifested in the actual built environment. Within the scope of this research, this objective is accomplished by:

1. briefly reviewing the principles of catoptric anamorphosis projection and data visualization, and
2. employing data spatialization in a way that preserves the legibility of the input data in the built environment.

Toward this end, this paper focuses on three primary domains to investigate precedents and relevant projects: catoptric anamorphosis projection, data visualization, and data spatialization.

Anamorphosis

As an illusory technique of linear and curvilinear perspectival representation in the field of art and architecture, anamorphic projection highlights the gap between the physical form of an object and its visual appearance [21]. Wright describes anamorphosis as an illegible two-dimensional artform that becomes meaningful when viewed from one particular point in the space or by using a special device [29]. The special device is most commonly a cylindrical, spherical, conical or polyhedral mirror [7, 5] that captures data beyond a human's normal cone of vision [30, 28].

The objective of anamorphic design is not limited to creating a deformed object that is supposed to be read clearly from one viewpoint. In some cases, its aim is to reorient and/or reorder spatial elements to suggest a perfectly composed geometric space. One astonishing example is in St. Peter's Square in Rome where the application of anamorphic design overrides the perception of the oval composition of the space into that of a circle from a privileged point in the space [21].

Anamorphic projection has also been the subject of a number of research projects within the domains

of art, architecture and mathematics. For example, a refined technique for the design and fabrication of anamorphic projection on complex surfaces is introduced in the work of Di Paola *et al.* [5]. In addition, Jovanovic *et al*. [16] explore an application of robotic arms in automatically generating anamorphic structures from a two-dimensional image. Another work approaches the subject using a ray-casting technique, and provides a comprehensive framework to generate anamorphic effects [4]. Optically Illusive Architecture (OIA) takes an anamorphic approach toward architecture that suppresses the three-dimensionality of the space from a privileged point [9]. Through the lens of mathematics, Hickin [8] explains his method for creating two-dimensional anamorphic effects, bundled with an in-depth process to implement the catoptric anamorphosis. Likewise, a similar research project provides a method to generate anamorphosis for three-dimensional objects by utilizing a variation of a simple projective map known within the computer graphics literature as *collineation* [7].

Data Visualization

Modern uses of data must contend with a massive, heterogeneous, and dynamic volume of information produced as a result of an ongoing data deluge [12], extensive portions of which are difficult to understand and analyze in their raw format. However, the integration of human judgment together with visual representations of the data turns this data overload into an opportunity [17]. Such data should be transformed into graphical and visual representations, which are referred to as data visualizations [28, 17, 18]. Data visualization allows a broad range of users to understand information concealed within the data by providing mental models of information [17, 27].

Research shows that an appropriate method of visualization improves users' cognition in the perception and exploration of data [25]. It also helps improve the efficiency of information retrieval and memorability of data [14, 23]. A visualization, once paired with a three-dimensional space, can influence spatial perception skills. Moreover, it has the potential to expand the public's exploration and understanding of critical, complex data via inclusion of data in sculptures and architectural installations [15].

Data Spatialization

As computational methods continue to develop, architects are taking advantage of the many data streams available to them during the design process [3, 6]. Using computational design tools to generate novel architectural forms and spatial opportunities bridges the field of data visualization and architecture, and forms the basis of data spatialization [19, 10]. Such representations of data additionally have an aesthetic appeal as an external aid for visual thinking [13], and are referred to as data sculptures [13, 20]. These data-based physical artifacts aim to augment the audience's understanding of data and any socially relevant issues that underlie it [31].

Several important preceding works have enriched the initial research and design development of this paper, with emphasis on catoptric representation, data visualization, and data-driven spatial effects. For instance, *Cloud Gate* in Chicago, Illinois, by Anish Kapoor is a sculpture that represents the surrounding buildings and the city's skyline, however, in a distorted way (Figure 1a). As far as this research is concerned, Cloud Gate triggers a critical question with regards to how its geometry contributes to the deformation of the cityscape. In other words, how should the geometry be designed if it is supposed to flawlessly represent its surroundings?

Living Light by David Benjamin, The Living^{TM}, is a sculptural canopy in a public park in Seoul, South Korea, that displays air quality in the region. This project aims to raise public awareness by combining real-time data and dynamic lighting. At any given time, if the day's air quality is better when compared to the previous year, a panel on the sculpture corresponding to the region is illuminated [2] (Figure 1b). Living Light raises a key question in terms of the role of data in forming the physicality of the canopy, calling into question whether the input data necessarily limit the variation of the geometry of the canopy.

(a) **(b)** **(c)** **(d)** **(e)**

Figure 1: *From left to right: (a) Cloud Gate, Anish Kapoor, 2006. (b) Living Light, The Living, 2007. (c) Centennial Chronograph, Marcus, 2014. (d) Data Moiré, Alvin Huang, 2016. (e) Weather Report, Swackhamer, 2017.*

Centennial Chromagraph is also a project in data spatialization, which "oscillates between representational and atmospheric readings" i.e., it occupies both the role of communicative and sculptural installation [19] (Figure 1c). However, this project represents an abstract form of data, and does not support updates to the data and interaction possibilities for users. *Data Moiré* uses the same approach and merges the territories of data spatialization in order to articulate a vast quantity of data as a spatial experience [10] (Figure 1d). Although the resulting project provides an architectural feature that enhances the spatial experience, it does not provide a readable representation of the input data to its audience.

Weather Report [24] is another example of data spatialization that relies on a design driven by data and driven by users (Figure 1e). This project uses a set of two illuminated balloon walls, and serves as an example of a successful *democratization of visualizations* [12, 20, 31, 11, 26] in a spatial context. While one of the walls represents quantitative real-time weather data, the other gives its audience the chance to intuitively design a visualization based on their recollections and memories of the weather data, i.e. a qualitative representation of data. As a spatial structure, it provides a novel method for user interaction, however, the design could have been elaborated further in order to allow spatial relationships with its surrounding environment.

While the projects above aim to spatialize specific types of input data in accordance with certain design parameters, they tend to undermine one key factor: readability of the data. In other words, the resulting embodiment requires a description, label, or legend to be decoded and recognized. This research aims to address this shortcoming by designing an anamorphic pavilion whose spatially-visualized data is accurate, dynamic, and easily recognizable by its audience.

Methodology

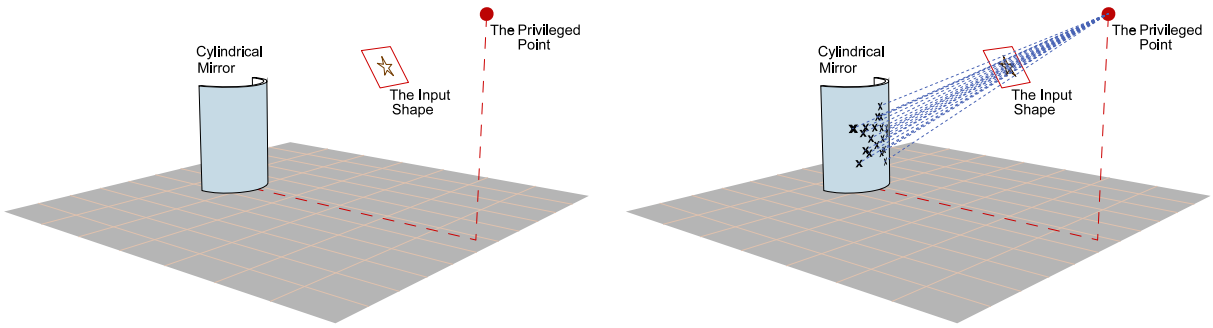
This project reverse-engineers the reflection law by means of a Grasshopper® code to simulate catoptric anamorphosis. Briefly, the code takes a two-dimensional curve as its input and generates a corresponding deformed curve, whose original form is revealed when placed in front of a particular mirror. The next step is to generate two-dimensional catoptric anamorphic images. Finally, the code is developed further to convert images into three-dimensional anamorphic objects.

The first experiment to three-dimensionalize an image is conducted using the Mona Lisa painting. Afterwards the research proceeds on to environmental datasets from North America to represent different types of engaging information. The results are expected to be legible when viewed from a privileged space through a cylindrical mirror.

The Pseudocode

The law of reflection indicates that rays of light travel through linear paths. When reaching a perfectly reflective surface, they bounce back at an angle identical to the incoming vector with respect to the surface

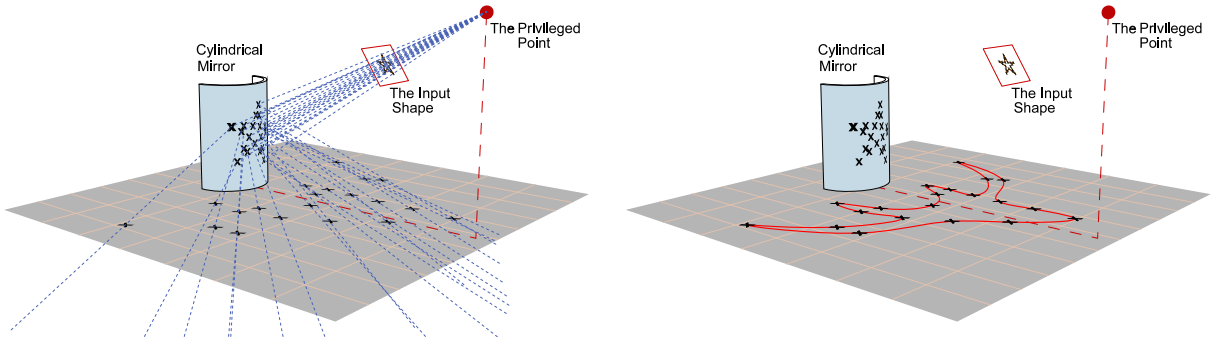
normal at the point of intersection [5, 4, 8]. The code reverses this phenomenon by first receiving a privileged point, a surface to serve in the role of mirror, a 2D or 3D space where the anamorphic object is formed, and the input curve (Figure 2a). Next, it converts the input shape into a series of points. Having placed the input shape between the privileged point and the mirror, the rays, extending from the viewpoint to the shape points, intersect with the mirror. In other words, the mirror acts as a *picture volume* [9] that hosts the intersection points (Figure 2b).



(a) **(b)**

Figure 2: *(a) The scene setup with the privileged point, the input shape, and the cylindrical mirror. (b) The input shape converted into a series of points and projected onto the mirror.*

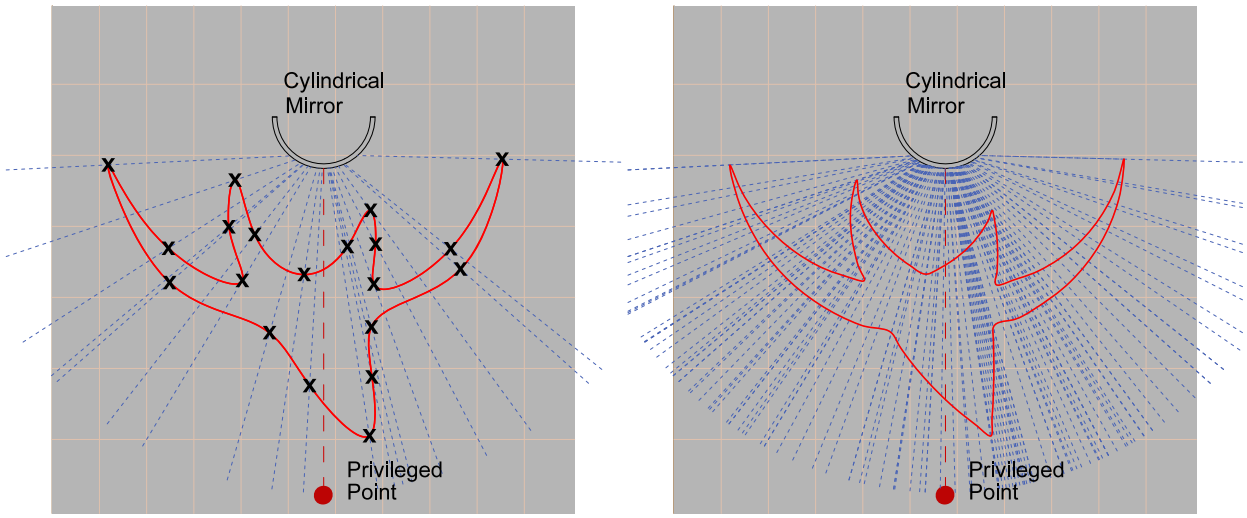
The law of reflection then allows the rays – extending from the privileged point to the corresponding intersection points on the mirror – to be treated as incoming rays of light and bounced/reflected off the mirror (Figure 3a). The intersection of these reflected paths with any arbitrary surface, i.e., the World XY plane here, constitutes a series of points. By interpolating a parametric curve through these points, an output shape is produced that reveals the original two-dimensional shape through the mirror when viewed from the privileged point (Figure 3b). In this project, we implemented interpolation through a Non-uniform rational B-spline (NURBS) periodic curve with a degree of three [1]. Interpolation through a higher number of points results in a higher quality of the anamorphic effect, once reflected on the mirror (Figure 4).



(a) **(b)**

Figure 3: *(a) The incoming rays to the privileged point reflected by the mirror. (b) The output curve generated by interpolating through the intersection points.*

The code is then developed to take images as input, instead of curves. Accordingly, we import an image (Mona Lisa) into Grasshopper®, and divide it into an arbitrary number of cells. Needless to say, a higher number of cells generates a higher resolution result. At this point, the program treats each cell as a curve for



(a) **(b)**

Figure 4: *The output curve generated by interpolating through the intersection points. More points result in a higher resolution output shape. The anamorphic star has been generated through (a) 20 points, and (b) 100 points.*

use with the first part of the code and ultimately generates the corresponding curve on an arbitrary surface, e.g., the world XY plane here. Next, the code extracts the average RGB color values of each cell and assigns the proper colors to the output cells (Figure 5).

Figure 5: *A catoptric anamorphic image of Mona Lisa.*

To generate a three-dimensional model from this result, we use the RGB value of an input cell as an extrusion factor for the corresponding output cell. The RGB values are converted to grayscale ranging from 0 to 255, i.e., black to white, meaning that cells with smaller RGB values receive less extrusion and are, thus, shorter.

Moreover, to avoid failure of the extrusion process, cells with the RGB value of 0, are assigned a small number as their extrusion factor. Figure 6 illustrates this method when the extrusion factor is applied along the Z vector.

(a) **(b)**

Figure 6: *The three-dimensional object corresponding to the Mona Lisa painting. Isometric and top views of (a) the virtual model, and (b) the physical 3D-printed model.*

Likewise, in catoptric anamorphosis, the extrusion is assigned a direction such that the two-dimensional image is displaced in a third dimension that gives it spatial depth. Per the first part of the code, every individual cell of the image correlates to an interpolated cell on the selected surface, i.e., the world XY plane. The extrusion direction of every single cell is a vector extending from the cell center, on the world XY plane, to the corresponding cell center on the mirror. Figure 7 illustrates the extrusion vectors in a conceptual four-cell image, and Figure 8 shows the results of this approach to three-dimensionalizing the Mona Lisa painting.

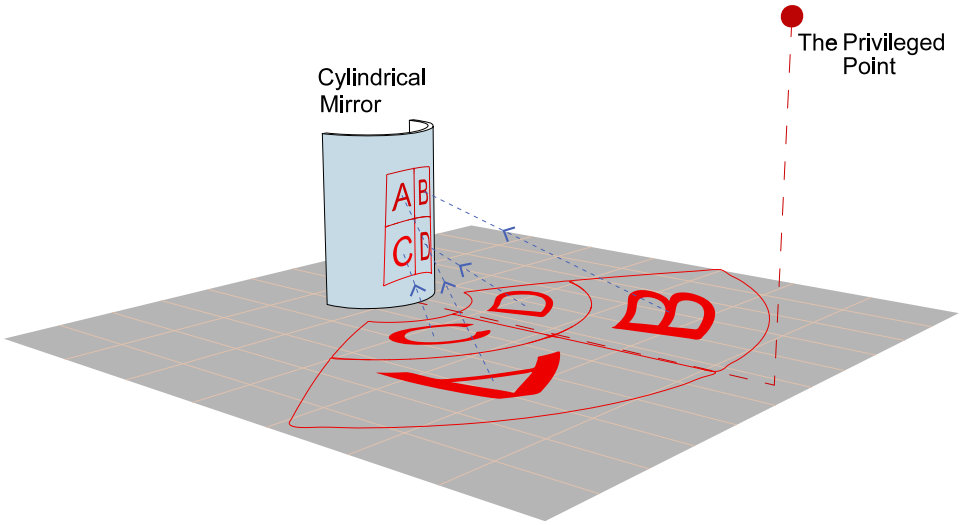


Figure 7: *A sample four-cell image and its extrusion directions.*

Figure 8: *A three-dimensional catoptric anamorphic representation of the Mona Lisa painting. (a) The virtual model. (b) The virtual model with the RGB values assigned to the ends of the extruded cells to enhance the effect. (c) The 3D-printed prototype.*

Results

This data spatialization project uses the method of catoptric anamorphosis and aims to translate widely distributed data into a medium that is easily readable by its audience and represents a spatial relationship between a pavilion and its urban environment. We conclude our project by introducing a data-driven designed pavilion with a mirror-based anamorphic structure (Figure 9).

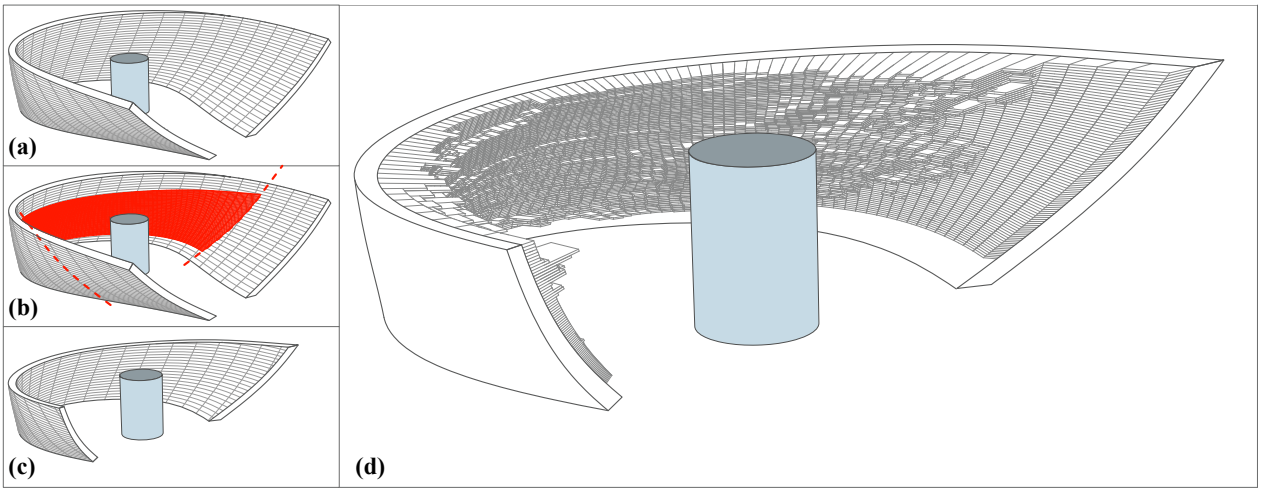


Figure 9: *(a) The host surface, (b) the locus of the anamorphic refection, (c) the modified subsurface cut-off from the host surface, and (d) the pavilion with brick-like elements mounted over the host surface.*

As mentioned earlier, we tend to use environmental datasets since we believe this type of data speaks to a broader range of audiences. In pursuit of this goal, a system is used to retrieve the datasets corresponding to the desired region of interest, together with live updates or changes in the data through a specified period

(https://www.globalgridsystems.com). In our project, the region of interest is North America, and the desired datasets – in the form of 2D images – include elevation, precipitation, population, and temperature data.

Among the retrieved datasets, the elevation data is chosen to build 6400 brick-like elements mounted over the host surface. The reason we choose elevation as the primary source of data to form the structure is that it is relatively consistent through time. These bricks will also be overlaid with updatable datasets of elevation, temperature, precipitation, and population in order to create a dynamic data spatialization. We propose this dynamism in the visualization of data to be supported by illuminating the top ends of each brick using configurable LED lights. This creates a unique projection method that not only supports the dynamic representation of data, but also serves as a lighting design for the anamorphic structure that is capable of rendering a different dataset each time. The results establish a high level of data-legibility in this data-driven pavilion, thanks to the anamorphic projection (Figure 10). Since each brick represents a specific portion of the data, a higher resolution result is achievable by assigning a smaller portion of the data to every single brick, and, therefore, increasing the number of bricks mounted over the shell of the pavilion.

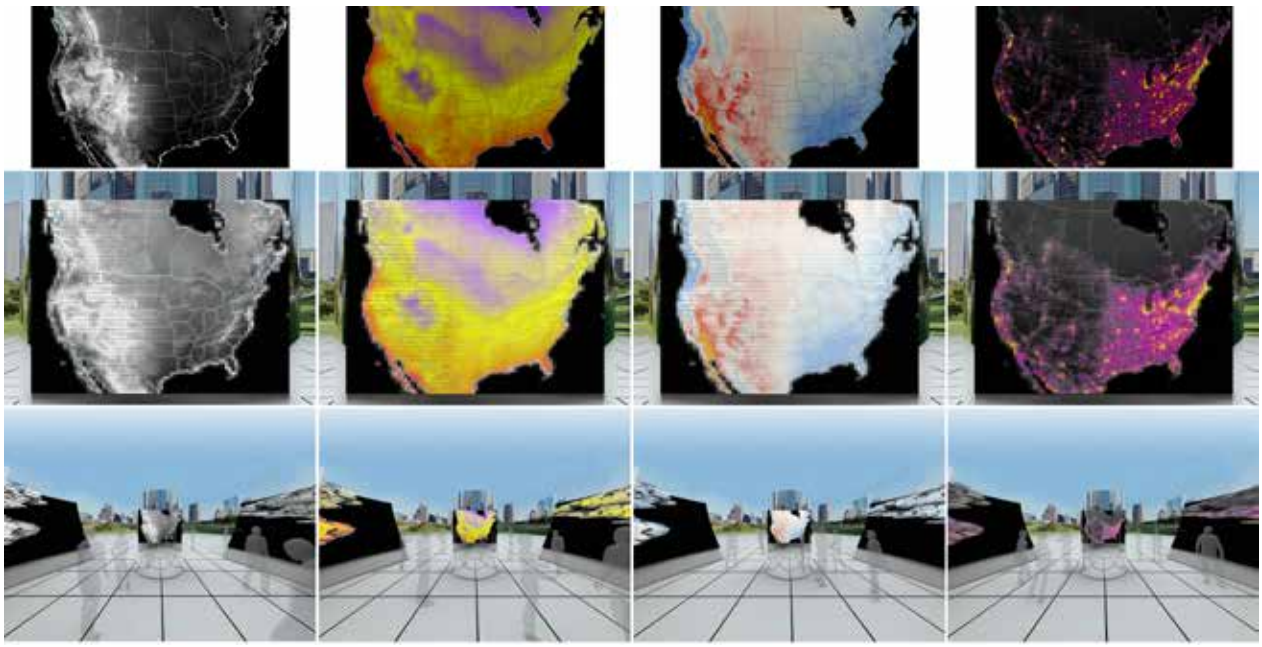


Figure 10: *The anamorphic projection maintains the readability of the input data in the data-spatialized pavilion. Top: the input data. Middle: the representation of the corresponding data in the mirror. Bottom, from left to right: pavilion representing elevation data, temperature data, precipitation data, and population data, respectively.*

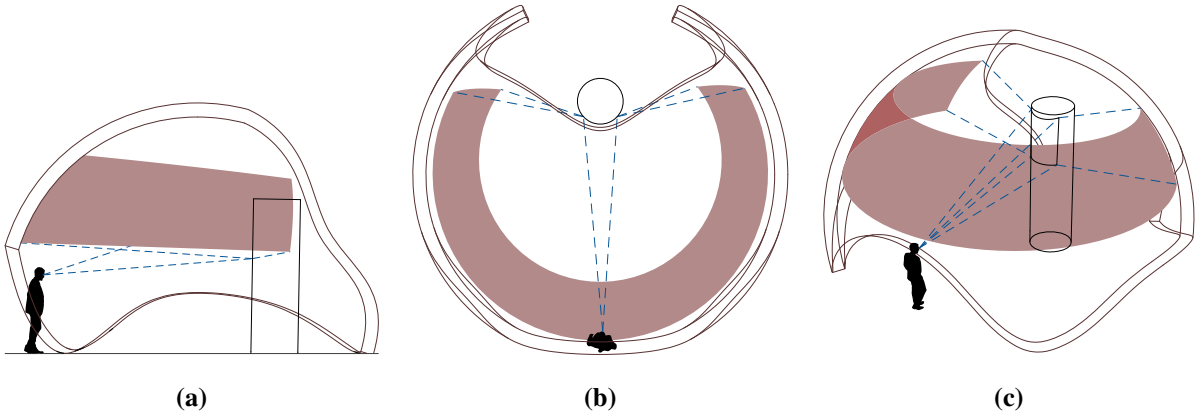
Discussion

The prime objective of designing this anamorphic pavilion is to keep the input data as legible as possible within the built environment while allowing people to interact with the space. While harnessing an optical illusion, it defines a notion of playful urbanism where people feel there are discoveries to be made. In this regard, we shall sacrifice one goal to save a more significant one. For instance, the set-up of the pavilion, including its geometry, radius of the mirror, and the height of the privileged point, pilots the anamorphic effect to remain accessible and, more or less, form within human reach. Indeed, this is the key to framing such a structure as an interactive space. By means of interaction, audiences discover that they can disrupt and/or augment the illusion by the presence of their own body in the space. Nonetheless, partial occlusion becomes inevitable as a consequence of audiences interacting with the space between the structure and the

mirror. Figure 11 illustrates an alternative set-up with an overhead locus of the anamorphic effect to avoid occlusion. However, we intentionally designed the pavilion to let the audience interact and learn about this historical method of projection, at the cost of occlusion. Furthermore, if the only purpose of the design was to inform the public of the input data, a high resolution screen would arguably be the easiest solution.

Fabrication, as another concern associated with this structure, is entirely dependant on materiality, the actual scale of the pavilion, and the accessible tools of fabrication. It is within the future scope of this research to study the pavilion through the lens of tectonics and how the components could possibly mount over the structure once physically fabricated. After finalizing the materiality, the scale, and the tools of fabrication, two critical questions from the digital-fabrication side of the project (as a future work) can be the following.

1. What are the precise dimensions in every individual cross section, where the geometry is changing in support of producing the optical effect?
2. How to dissect the entire structure into small parts to make the pavilion easy to assemble?



(a) **(b)** **(c)**

Figure 11: *Although the overhead alternative eliminates the occlusion, it mostly stays out of reach and does not provide much interaction.*

Conclusion and Future Work

This investigation reaches territories beyond a combination of a simple gimmick, data visualization, and data spatialization. It encourages the audience to discover spatial agencies and orientations through a traditional method of representation, i.e., catoptric anamorphosis.

This pavilion is a spatial structure that represents data in a readable non-abstract format and interacts with its audience and its environment (Figure 12). The proposed methodology allows us to harness data and catoptric anamorphosis to design a publicly-accessible privileged space at an architectural scale.

Despite the fact that the anamorphic effect is designed to be perceived from a unique privileged point in the space, it still seems to be fully graspable within an area adjacent to the privileged point. Defining the threshold of this area where the effect is still recognized could be accomplished through a user study. Moreover, offering control of the illumination to the audience could form the basis of another user study on the concept of democratization of visualizations in data-driven spaces. The aforementioned studies can be conducted via Virtual Reality (VR) as a future work. Users will be asked to virtually navigate within the pavilion and provide feedback on various subjects. For example, locating an optimized privileged point with minor occlusion, gaining knowledge on how the anamorphic effect works, etc.

Based on the results of these VR studies, a refined design for the pavilion can undergo a fabrication process at its actual scale, which provides the chance to consider this project as a behavioral-design objective. In

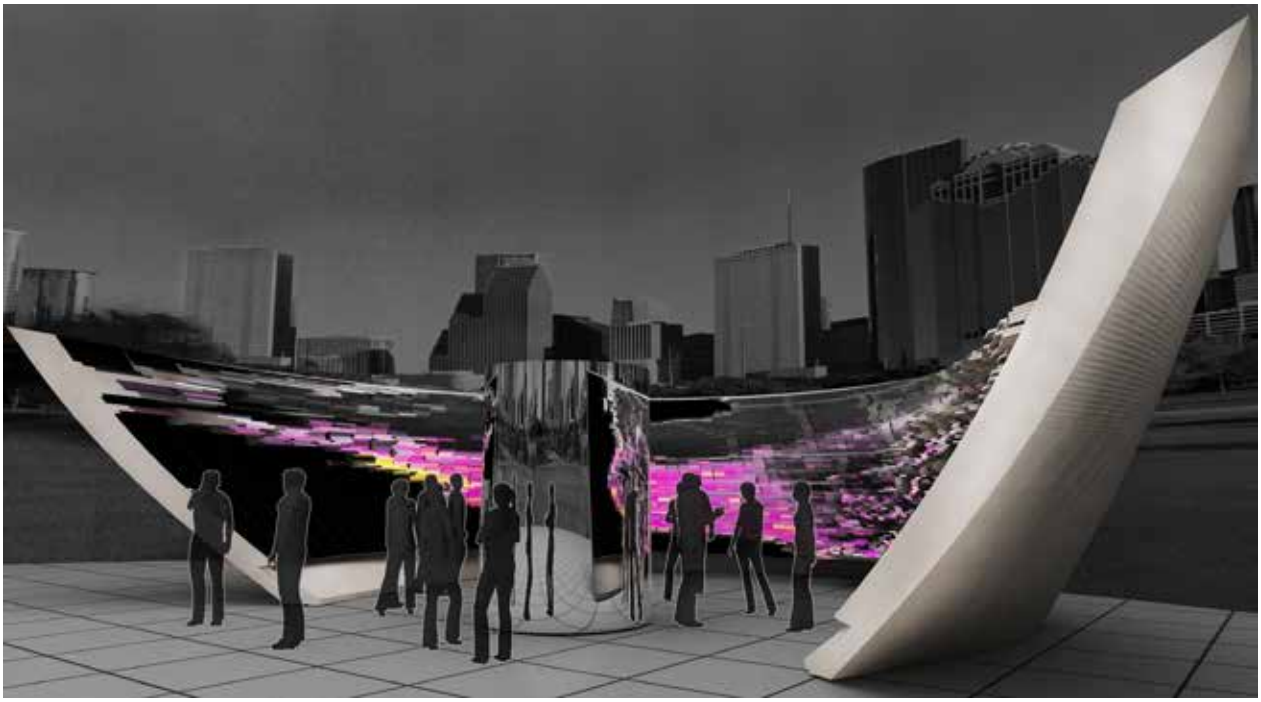


Figure 12: *The catoptric-anamorphic pavilion.*

this regard, research will be conducted to evaluate the potential to foster sustained curiosity, awareness, and discovery within the built environment that may otherwise lack interest. Such an approach creates interaction possibilities for users by encouraging them to engage with the design and become part of it while reading and understanding the data.

Image Credits

- Figure 1(a): Cloud Gate, by Craig Sinclair, https://www.flickr.com/photos/craig-sinclair/2626662070. https://www.flickr.com/photos/craig-sinclair/, CC BY 2.0, via Flickr.com.
- Figure 1(b): Living Light, photo courtesy of The Living http://www.thelivingnewyork.com/.
- Figure 1(c): Centenial Chromograph, photo courtesy of Variable Projects http://www.variableprojects.com/.
- Figure 1(d): Data Moiré, photo courtesy of Synthesis Design + Architecture, https://synthesis-dna.app.box.com/v/IBMSF.
- Figure 1(e): Weather Report, MINN LAB Design Collective, by Krista McCullough (used with permission).

  All other drawings and images by the authors.

References

[1] (16-May-2019). Curvethroughpt. https://docs.mcneel.com/rhino/6/help/en-us/index.htm#commands/curvethroughpt.htm?Highlight=interpolate. Accessed: 2019-05-20.

[2] (2007). Living light. https://www.livinglightseoul.net. Accessed: 2017-11-01.

[3] Brown, N. and Mueller, C. (2017). Designing with data: Moving beyond the design space catalog. In *"Disciplines and Disruption" Proceedings of the 37th Annual Conference of the Association for Computer Aided Design in Architecture*, pages 154–163. ACADIA.

[4] De Comite, F. and Grisoni, L. (2015). Numerical anamorphosis: an artistic exploration. In *SIGGRAPH ASIA 2015 Art Papers*, page 1. ACM.

[5] Di Paola, F., Pedone, P., Inzerillo, L., and Santagati, C. (2015). Anamorphic projection: analogical/digital algorithms. *Nexus network journal*, 17(1):253–285.

[6] Gonzalez Rojas, P. (2017). Space and motion: Data-driven model of 4d pedestrian behavior. In *"Disciplines and Disruption" Proceedings of the 37th Annual Conference of the Association for Computer Aided Design in Architecture*, pages 266–273. ACADIA.

[7] Hansford, D. and Collins, D. (2007). Anamorphic 3d geometry. *Computing*, 79(2-4):211–223.

[8] Hickin, P. (1992). Anamorphosis. *The Mathematical Gazette*, 76(476):208–221.

[9] Hosseini, S. V., Taron, J. M., and Alim, U. R. (2017). "optically illusive architecture: Producing depthless objects using principles of linear perspective". In *"Disciplines and Disruption" Proceedings of the 37th Annual Conference of the Association for Computer Aided Design in Architecture*, pages 274–283. ACADIA.

[10] Huang, A. and Chaney, M. (2017). Data moirÃľ: Optical patterns as data-driven design narratives. In *"Disciplines and Disruption" Proceedings of the 37th Annual Conference of the Association for Computer Aided Design in Architecture*, pages 162–167. ACADIA.

[11] Huron, S., Carpendale, S., Thudt, A., Tang, A., and Mauerer, M. (2014a). Constructive visualization. In *Proceedings of the 2014 conference on Designing interactive systems*, pages 433–442. ACM.

[12] Huron, S., Jansen, Y., and Carpendale, S. (2014b). Constructing visual representations: Investigating the use of tangible tokens. *IEEE transactions on visualization and computer graphics*, 20(12):2102–2111.

[13] Jansen, Y. (2014). *Physical and tangible information visualization*. PhD thesis, Université Paris Sud-Paris XI.

[14] Jansen, Y., Dragicevic, P., and Fekete, J.-D. (2013). Evaluating the efficiency of physical visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2593–2602. ACM.

[15] Jansen, Y., Dragicevic, P., Isenberg, P., Alexander, J., Karnik, A., Kildal, J., Subramanian, S., and Hornbæk, K. (2015). Opportunities and challenges for data physicalization. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3227–3236. ACM.

[16] Jovanovic, M., Stojakovic, V., Tepavcevic, B., Mitov, D., and Bajsanski, I. (2016). Generating an anamorphic image on a curved surface utilizing robotic fabrication process. In *"Complexity and Simplicity" Proceedings of the 34th eCAADe Conference*, pages 185–191. eCAADe.

[17] Keim, D. A., Mansmann, F., Schneidewind, J., and Ziegler, H. (2006). Challenges in visual data analysis. In *Tenth International Conference on Information Visualisation (IV'06)*, pages 9–16. IEEE.

[18] Liu, S., Cui, W., Wu, Y., and Liu, M. (2014). A survey on information visualization: recent advances and challenges. *The Visual Computer*, 30(12):1373–1393.

[19] Marcus, A. (2014). Centennial chromagraph: Data spatialization and computational craft. In *"Design Agency" Proceedings of the 34th Annual Conference of the Association for Computer Aided Design in Architecture*, pages 167–176. ACADIA.

[20] Moere, A. V. and Patel, S. (2009). The physical visualization of information: designing data sculptures in an educational context. In *Visual information communication*, pages 1–23. Springer.

[21] Perez-Gomes, A. and Pelletier, L. (2000). Relocating anamorphosis. In *Architectural representation and the perspective hinge*, chapter 1, pages 138–149. MIT Press, Cambridge.

[22] Rohan, T. M. (2000). Rendering the surface: Paul rudolph's art and architecture building at yale. *Grey Room*, pages 84–107.

[23] Stusak, S., Schwarz, J., and Butz, A. (2015). Evaluating the memorability of physical visualizations. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3247–3250. ACM.

[24] Swackhamer, M., Johnson, A. J., Keefe, D., Johnson, S., Altheimer, R., and Wittkamper, A. (2017). Weather report: Structuring data experience in the built environment. *Proceedings of Architectural Research Centers Consortium*, pages 102–111.

[25] Taher, F., Hardy, J., Karnik, A., Weichel, C., Jansen, Y., Hornbæk, K., and Alexander, J. (2015). Exploring interactions with physically dynamic bar charts. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3237–3246. ACM.

[26] Viégas, F. B. and Wattenberg, M. (2007). Artistic data visualization: Beyond visual analytics. In *International Conference on Online Communities and Social Computing*, pages 182–191. Springer.

[27] Ware, C. (2012a). Interacting with visualizations. In *Information visualization: perception for design*, pages 345–346. Elsevier.

[28] Ware, C. (2012b). The perceptual evaluation of visualization techniques and systems. In *Information visualization: perception for design*, pages 431–443. Elsevier.

[29] Wright, L. (1983a). Grand illusions. In *Perspective in perspective*, pages 139–156. Routledge.

[30] Wright, L. (1983b). The object, the eye and the picture. In *Perspective in perspective*, pages 1–33. Routledge.

[31] Zhao, J. and Moere, A. V. (2008). Embodiment in data sculpture: a model of the physical visualization of information. In *Proceedings of the 3rd international conference on Digital Interactive Media in Entertainment and Arts*, pages 343–350. ACM.

Decorative Knots in 3D Artwork: Fabricating Models with Successive Knotting

Haruka Ikeda[1] Leo Miyashita Masahiro Hirano Masatoshi Ishikawa

The University of Tokyo, Japan
[1]Haruka_Ikeda@ipc.i.u-tokyo.ac.jp

Abstract

We propose a novel method for fabricating 3D shapes with decorative knots, which are unique patterns created by overlapping and tying thick cords and are an art form passed down since ancient times. Our proposed method automatically computes and displays a route for connecting decorative knots successively in the discretized input model. We introduce a new technique for converting the knotting process into a knitting-like procedure, which removes restrictions on the actual manufacturing steps for a number of knots. After the route is generated, appropriate types of decorative knots are selected that most closely imitate the detailed surface of the input model. Our rendering results show that a user can create large-scale and complex 3D artwork that makes good use of the rich geometric patterns of decorative knots, which will help and extend the fabrication possibilities in handmade contexts.

1 Introduction

Digital fabrication is the technology of shaping various structures of 2D or 3D objects based on digital data, such as polygon mesh models, and the application of algorithms to design and control the structures through computational methods. As a new application of digital design, this paper focuses on decorative knots [1], which are a type of artwork comprising a series of unique patterns and have been passed down since ancient times. Examples of decorative knots are shown in Figure 1. These knots are created by tying and crossing thick cords of about 5–10 mm diameter to express various geometric patterns including motifs of flowers as well as simple and complex patterns, mainly for display and ornamental use.

However, structures of decorative knots have thus far not been extended to construct 3D shapes. This is because knotting (i.e., the process of tying knots) includes procedures to pass the cord through loops made of other sections of the same cord such that a large structure with numerous knots requires a considerable amount of time in the manufacturing step, where the remainder of the cord must be passed through every loop in the knot. Because of this difficulty, there was no choice but to use very thin string, which is easier to maneuver through a loop. For example, in lace tatting, we wind the thread of about 0.3–1 mm diameter into a small lump. In knitting, on the other hand, string of about 1–5 mm diameter is used to create models by continuously connecting stitches, which are the basic or unit elements of the models and are made simply by hooking a part of the string (unlike a knot). Hence, the knitted fabric cannot express complex structures, which require more than just consecutive rows of the simple stitch.

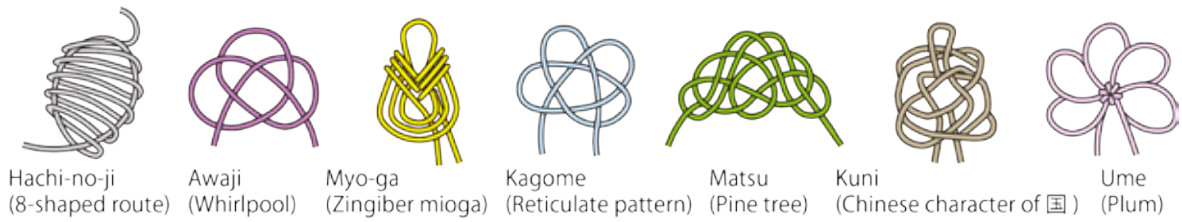


Figure 1 : *Examples of decorative knots with various shapes.*

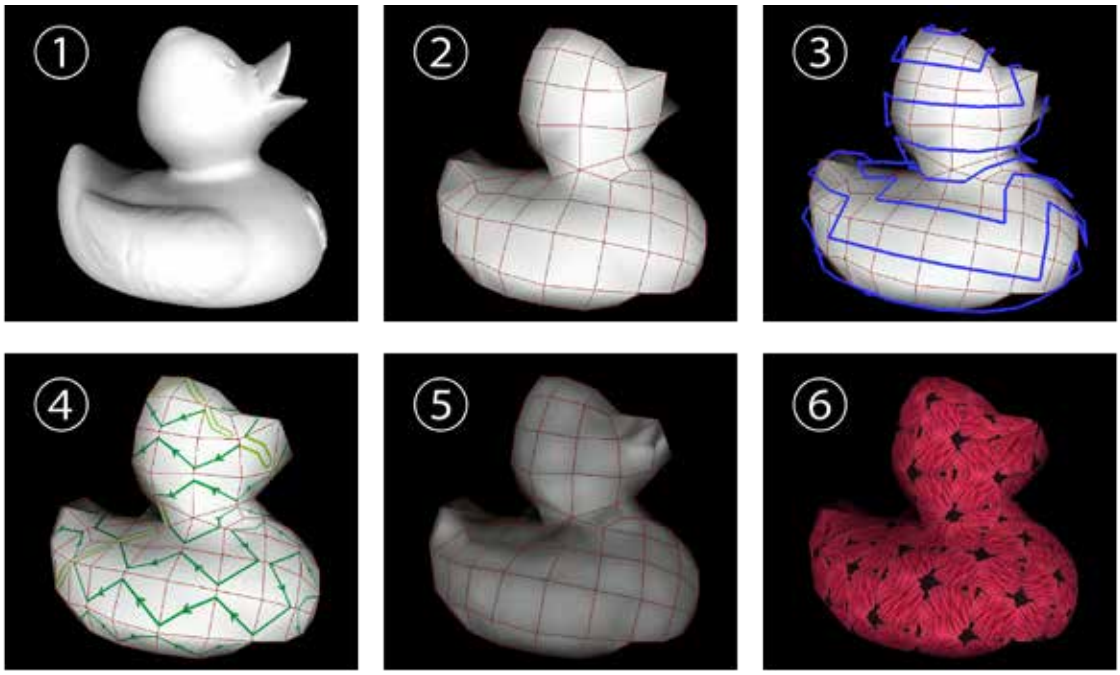


Figure 2 : *Pipeline of our method: ① inputting a model, ② generating a quad mesh, ③ finding a polygonal path, ④ finding a knot route, ⑤ calculating displacement maps, and ⑥ applying decorative knots.*

Therefore, we propose a fully automatic pipeline for constructing large-scale 3D objects, as shown in Figure 2. We first discretize the input model into mesh structures, then generate a path along which the knots can be successively connected, and finally apply the rich structures of decorative knots. This is an innovative approach to combining digital design and traditional handicraft to extend range of shapes possible.

2 Related Work

Various methods have been proposed in literature to support the fabrication of 3D objects in handmade contexts. Beady [5] is a system that assists the design and construction of 3D beadwork, which is the art of connecting beads together by wires, based on polygonal mesh structures that represent the whole model. The structure of the problem is similar to that of ours with knots in that the user connects the elements (i.e., a bead and knot, respectively) to create the whole model. However, they cannot be treated in the same manner because knots should be arranged along the cord that they are made from.

In particular, fabrication using strings is drawing increasing attention. An interactive design tool for authoring, simulating, and adjusting yarn-level patterns for knitted and woven cloths are proposed [9]. Stitch meshing [12] is a proposal to construct arbitrary 3D shapes by knitting. It converts input models into quad-dominant polygon mesh structures, each quad of which correspond to the smallest units of knitting. An extension called knittable stitch meshes [13] guarantees that the models designed with this method can actually be produced by manual knitting. Similarly, our problem deals with the given models as meshed structures to construct surfaces with decorative knots. However, these methods are inapplicable to our knotting problem because the decorative knots are not always connected in the same direction as the knitting. Further, decorative knots express their patterns as lumped units with definite sizes, whereas yarn-level knitting methods treat the whole structure as a set of small and simple stitches.

3 Problem Settings

Representation of 3D Models: Quad Mesh

In order to construct surface shapes with finite collections of decorative knots, we deal with them as structures made of discretized units of polygons [3]. Polygon meshes are suitable because we can use the topological information of the mesh structure to reflect the relationships between adjacent pairs of polygons. Of the available choices of polygons to subdivide the surface, we use quad faces so that we can easily connect them from one to the next by passing the cord along the diagonal of each quad. In this paper, we call the path for connecting quads as the *polygon path*. Further, we call the successive diagonal directions based on the polygon path as the *knot route* along which decorative knots are appropriately arranged.

Structure of the Problem on Mesh: Graph Theory

Once the meshes are generated on the model surface, we can define an undirected graph in which a node corresponds to each quad and an edge represents an adjacent relationship between the jointed nodes. In graph theory, a graph in which any two vertices are connected by exactly one path is called a tree. In this paper, if there are branching points in a tree and multiple routes extending from one node, we call each extended route as a *branch*. If a branch consists of only one polygon, we call it a *leaf*. In our problems, series of decorative knots must be connected by a tree that covers all of the nodes. The problem of finding a path that visits all of the nodes exactly once is equivalent to the Hamiltonian path problem [2]. However, determining whether such paths exist is known to be NP-complete [7]. Hence, we must define an approach to finding the appropriate path without relying on the conventional path planning problem.

Technique for Successive Knotting: Double Knotting

As shown in Figure 3, there are two types among structures of decorative knots: a closed path, where both the start and end point of the knot come at the same point, and structures whose end points extend in different directions. For the latter case, knots cannot be always created continuously along one-stroke paths because the direction of the cord depends on the type of the knot, which makes it difficult to connect knots according to a certain rule.

Therefore, we propose a preprocessing step that we call *double knotting*. First, we select twice the length of the cord that is required to make a knot with conventional techniques. Then, we fold the selected part at its midpoint and treat it as a single cord when making a knot. This generates a virtual double-tracked knot whose end point always comes back to the beginning point of the knot. In Figure 4, structures of the *Hachi-no-ji* knot made according to both (a) conventional method and (b) double knotting are shown. In the double-knotted structure, the exit side of the knot always comes to the entry side. Further, the processes of passing the remaining length of a cord through a loop every time to make a new knot can be removed, which used to be a problem when constructing large structures with thick cords.

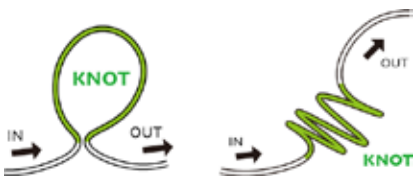


Figure 3: *End points of knots are sometimes at the same point and sometimes extend in different directions.*

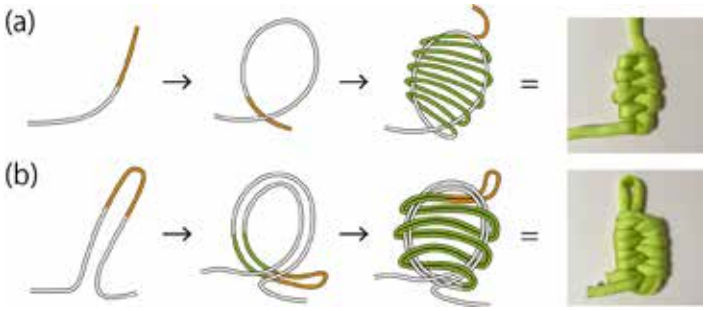


Figure 4: *Structures of the (a) conventional and (b) double knotted Hachi-no-ji knot.*

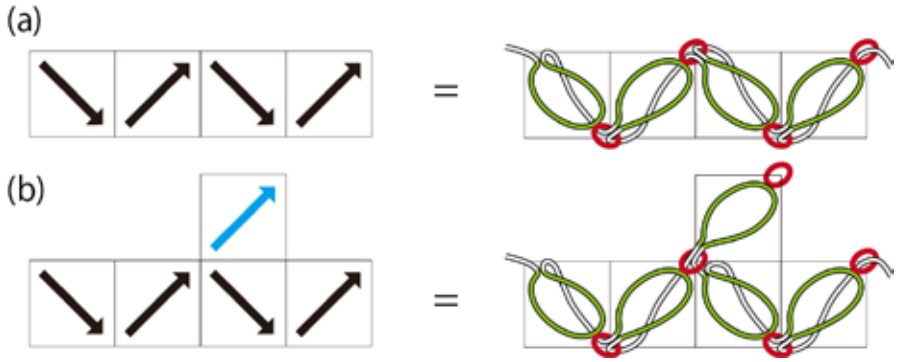


Figure 5: *Knot routes and corresponding decorative knots. The green areas are the bodies of the knots, and the red areas are the loops generated during the double knotting. (a) All knots are connected by the one-stroke path of the cord. (b) Even when a leaf (shown in blue) is inserted, the additional knot does not affect the original route.*

During the double-knotting process, a loop is generated at the points corresponding to the end of the knot with conventional methods, as highlighted in orange in Figure 4. In our method, we use these loops as the beginning points of the successive knot when connecting it to the already-generated one. As shown in Figure 5(a), by passing the cord to generate the next knot inside the loop, we can successively generate a new knot which is connected to the diagonal direction of each quad. At the same time, additional tension of pulling the cord toward the outside of the knot is caused, which prevents the generated loop from coming back towards the starting point of the knot and getting untied.

Constraints of a Route for Connecting Knots

Of the routes available to connect decorative knots, one-stroke paths with no branching points are the most desirable because we can generate knots without deviating from the route. However, such paths are not ensured to exist in arbitrary input models. Hence, we relaxed this restriction; as shown in Figure 5(b), even if a polygon path has branches constructed of one polygon (i.e., a leaf), other parts of the route are not affected by the existence of the additional suspended knot. Therefore, leaves are viable components. However, branches of two or more polygons are not acceptable because we cannot return to the starting point after generating the series of knots without influencing the connection to the next polygon.

4 Methods for Creating Models with Decorative Knots

Generating Quad Meshes

In order to create a model with decorative knots, our method converts the input model into quad meshes by applying instant meshes [6], which is a robust and efficient method that converts a surface into a naturally aligned mesh with high isotropy. The number of quads is determined based on the level of detail needed. After the mesh is generated, we regard it as an undirected graph in which each node corresponds to a quad in the mesh. We assign the structure of the decorative knot to each quad and solve the problem of appropriately connecting the knots where the corners of the quads are shared.

Finding Polygon Paths

On the generated graph, in order to find a path that passes through all of the nodes exactly once, we apply a type of greedy algorithm [4] instead of solving the Hamilton path problem. These algorithms choose a path based on minimizing the local cost at each stage without worrying about the effects that these decisions

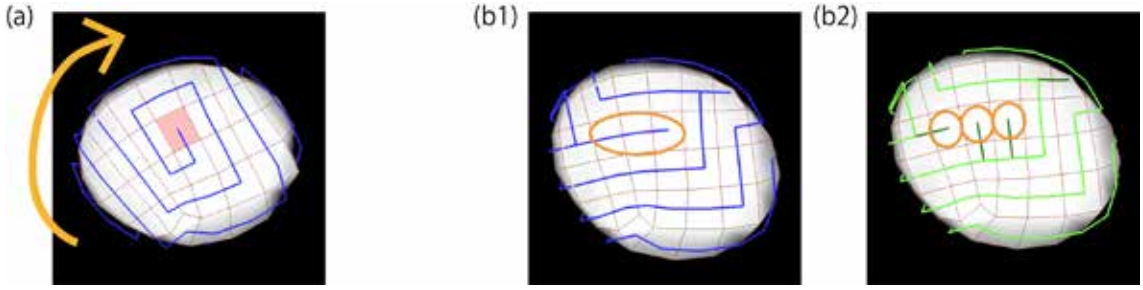


Figure 6: *(a) Spiral polygon path. (b1) Polygon path with a branch made of three polygons (circled in orange). (b2) Corrected path: the branch is disassembled into three individual leaves.*

may have in the future, which can be solved in polynomial time in many problems. Kruskal's algorithm [8] is one of the algorithms derived on the basis of this concept. It is a minimum spanning tree algorithm on a weighted graph, which regards all nodes as a tree and repeatedly combines the trees with the smallest-weight available edge up to a single spanning tree. In our problem, however, we cannot determine the order of the selected edges of the graph solely from the information of local points because there are no weights on them. Hence, we determine the order of connecting knots according to the following steps (i)–(iv).

(i) **Picking a Starting Point and Generating a Spiral Polygon Path**

To generate a polygon path, we pick one of the quads as the starting point of the search and then repeatedly select a polygon on the right from among the available polygons as the direction of extension of the path. This process connects the whole structure in a spiral path, as shown in Figure 6(a).

(ii) **Generating Branches to Deal with Dead Ends**

On drawing the spiral paths, if all the accessible polygons have already been passed, it is impossible for the path to move any further because of the rule that no polygon must be traversed more than once. In such cases, we leave the dead end as a branch or a leaf and search for the next available path according to the spiral rule (i) from the point where the branch begins.

(iii) **Disassembling Branches into Leaves**

If the generated branch contains more than one polygon, it disturbs the connecting knots continuously. Hence, we convert the branch into a structure made of multiple leaves. Figure 6(b1) is an example of a polygon path, where there is a branch made of three polygons (circled in orange). In such cases, we generate new leaves at the adjacent nodes and replace the branch with these leaves, as shown in Figure 6(b2). We continue disassembling all the branches on a path, as long as this is possible.

(iv) **Searching for the Best Path**

In case the generated path has too many leaves, we pick up another polygon as the starting point and repeat the processes of (i)–(iii). These processes can be calculated in a very short time because they are based on the fast greedy algorithm. After paths from all of the polygons are searched, we choose the best path with the fewest leaves.

Converting the Polygon Path into the Knot Route

Based on the generated polygon path, we make a knot route, which shows the order and direction for decorative knots to traverse in the fabrication process. We begin with the diagonal of the first polygon and continuously extend the route so that the knots are successively connected; diagonal directions are selected for each polygon, keeping the order of the polygon in the polygon path. Figure 2③ and ④ respectively show the polygon path and the corresponding knot route.

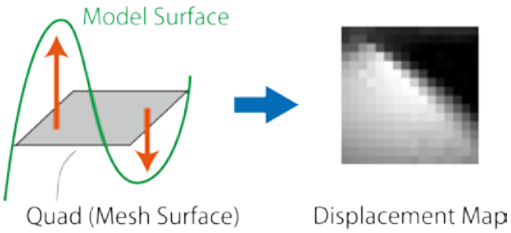


Figure 7 : *Process of calculating a displacement map. The brightness value is proportional to the distance from the mesh surface to the model surface.*

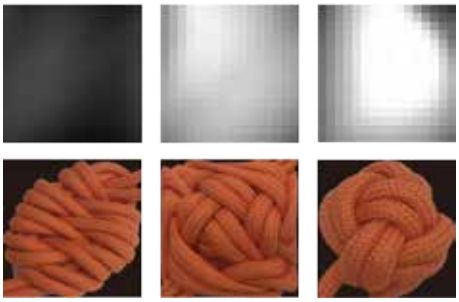


Figure 8 : *Displacement maps and corresponding knots. (L to R)* Hachi-no-ji *knot,* Kuni *knot, and* Awaji *knot. Standard deviations of the brightness are 12, 30, and 52.*

Finding Appropriate Decorative Knots

Finally, for each quad, we assign a decorative knot which reflects the patterns of the surface as much as possible. For this purpose, we compute a displacement map, which is an image acquired by calculating the brightness values at each point in proportion to the distances from the mesh surface to the surface of the input model, as shown in Figure 7. Then, we choose the best decorative knot whose surface has a texture closest to the input shape based on it. We select candidates for the assigned decorative knot from [11] and then choose one among them for each quad; we set thresholds of the standard deviations of brightness in the displacement map to classify images into three types according to the geometrical patterns on quads. We basically use the *Hachi-no-ji* knot because it is plain and can fit into a wide range of patterns. For quads whose standard deviations are a bit high, the *Kuni* knot, which has more uneven patterns, is selected. If the standard deviation is especially high, the *Awaji* knot is selected, whose shape is the most convex. Figure 8 shows examples of the displacement maps and decorative knots that are assigned to each type.

5 Results of Rendered Artwork Models

Representative results of artwork models where the quads are replaced by the handmade decorative knot images using computer renderings (Xeon Gold 5122 CPU @3.60GHz) are shown in Figure 9, as well as in Figure 2⑥. Various 3D shapes including even topologically different ones can be produced. Based on our method, a user can create a model both with high resolutions and with low polygon meshes. In

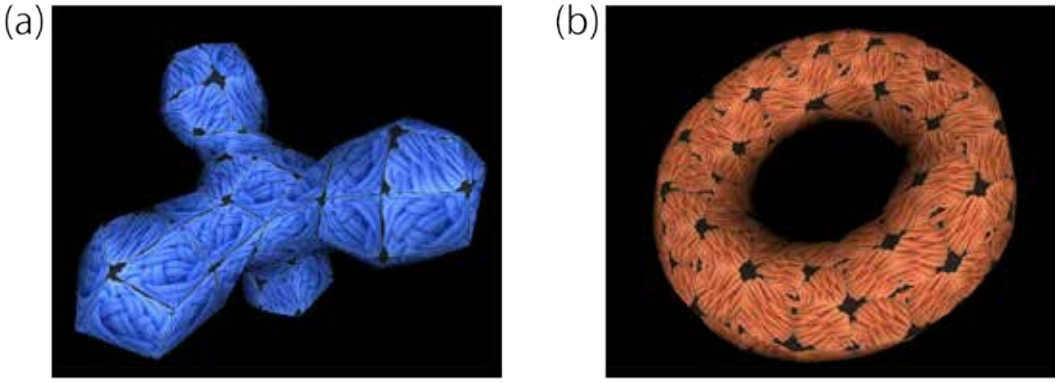


Figure 9 : *Rendered results of (a) a trahedral block, and (b) a torus. Appropriate decorative knots are assigned to each quad and detailed textures are expressed.*

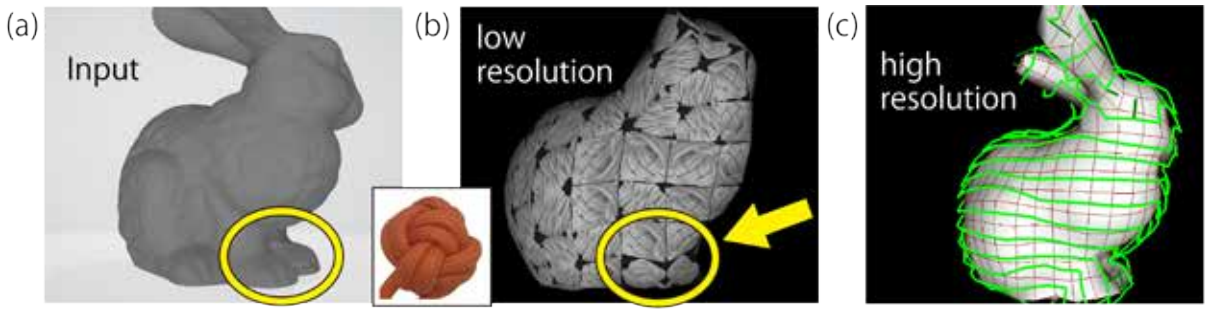


Figure 10: *(a) An input bunny model, (b) a rendered model with a low resolution, and (c) a polygon path with a high resolution.*

the latter case, the rich structures of decorative knots are especially important. For example, a model of a bunny in Figure 10(a) is discretized into a low resolution model with about 150 quads and is rendered into Figure 10(b), in which the front legs (circled in yellow) are appropriately expressed by the *Awaji* knot, whose spherical shape highly imitates the details of the corresponding parts in the input model. Further, our method can design the model with high resolution of more than 500 quads in a very short time (less than 10 seconds), which is very helpful for both skilled creators and beginners because previously creating such large models with complex structures was difficult to be considered.

6 Limitations and Future Work

There are some cases where our method cannot be smoothly applied. For example, in the process of discretizing input models into quad meshes, structures of narrow areas, like the ears of the bunny, are easily lost especially when the resolution is low. Also, even if such structures are expressed with high resolution models, it is not always ensured that branches can be disassembled into leaves on the generated polygon path. Actually, in the narrow region of the ears in Figure 10(c), no suitable paths were found without branches of more than two polygons. This is because once the path extends from the body to the top end of the ears of the bunny according to the spiral rule, there are no more available polygons remaining for the path to return to the body area. However, such problems are not unique to our method. For example, the traditional process of knitting a sweater is not all done continuously; the sleeves are usually knitted separately and then attached to the torso to construct the whole shape. Similarly, we can work around the problem by dividing the input model into parts in advance, applying our method to each divided part, and then assembling the parts. Hence, this limitation is not a serious disadvantage.

In the future, it may be possible to generate a polygon mesh that is optimized to produce a knot route. If the grid patterns of meshes are controlled interactively using the method like Boundary First Flattening [10], we will be able to freely design the surface and selectively assign polygons to the area we want to express with a specific decorative knot. Further, because we can easily pull the cord and stretch its shape to increase the variety of geometric patterns, if the deformability of knots are taken into consideration when an optimized route is generated, we will be able to fabricate models with more detailed expressions. In our classification algorithm, we currently use only three types of decorative knots. However, it is possible to use other types of knots to express more unique shapes of models. Such further extensions will lead to richer artwork based on the combination of both experimental techniques and computational strategies.

7 Conclusion

In this paper, we proposed a new pipeline to automatically design and construct 3D shapes with successive decorative knots. We first discretized input models into quad mesh structures and then determined routes of connecting knots. In order to construct model surfaces with a single-stroke cord, we proposed the double knotting technique, which converts the process of knotting into a knitting-like process to remove the difficulty of connecting a large number of knots in the actual manufacturing steps. Also, we proposed a practical algorithm of finding routes to smoothly tie knots, which makes it possible to connect the overall structure in a viable way in polynomial time. We showed rendered results where knots with appropriate geometric patterns were assigned to each quad, which demonstrate the possibilities of creating both a large-scale 3D artwork and a model with detailed expression. This type of artwork is first realized by our method of applying decorative knots to create 3D models with rich patterns on the surface and complex structures, which is important in the sense that we combined both computational methods and traditional techniques to expand the available range and possibilities of handmade artwork.

References

[1] K. Bien. *Hajimete-no kazari musubi*. Suiyosha Publishing, 2009.

[2] J. A. Bondy, U. S. R. Murty, et al. *Graph theory with applications*, volume 290. Citeseer, 1976.

[3] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Lévy. *Polygon mesh processing*. AK Peters/CRC Press, 2010.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT press, 2009.

[5] Y. Igarashi, T. Igarashi, and J. Mitani. Beady: interactive beadwork design and construction. *ACM Transactions on Graphics (TOG)*, 31(4):49, 2012.

[6] W. Jakob, M. Tarini, D. Panozzo, and O. Sorkine-Hornung. Instant field-aligned meshes. *ACM Transactions on Graphics (TOG)*, 34(6):189:1–15, 2015.

[7] D. S. Johnson. The NP-completeness column: an ongoing guide. *Journal of Algorithms*, 6(3):434–451, 1985.

[8] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.

[9] J. Leaf, R. Wu, E. Schweickart, D. L. James, and S. Marschner. Interactive design of periodic yarn-level cloth patterns. In *SIGGRAPH Asia 2018 Technical Papers*, page 202. ACM, 2018.

[10] R. Sawhney and K. Crane. Boundary first flattening. *ACM Transactions on Graphics (TOG)*, 37(1):5, 2018.

[11] H. Takahashi. *Musubi dai-hyakka*. Boutique-sha, 2007.

[12] K. Wu, X. Gao, Z. Ferguson, D. Panozzo, and C. Yuksel. Stitch meshing. *ACM Transactions on Graphics (TOG)*, 37(4), 2018.

[13] K. Wu, H. Swan, and C. Yuksel. Knittable stitch meshes. *ACM Transactions on Graphics (TOG)*, 38(1):10, 2019.

Modular Construction of Symmetrical Knots

C. H. Séquin, W. Brandon, J. Liu
CS Division, University of California, Berkeley

Abstract

The goal is to construct highly symmetrical models of mathematical knots from a single modular component that can be fabricated easily on inexpensive 3D printers or by injection molding. The "elbow" module, or *Λ-module,* is a piece of tubing that bends through a fixed given angle. To keep control over the torsional angle between subsequent modules, the tube cross section is not a circle but a regular polygon. We discuss possible optimization algorithms that allow the design of such knots with minimal amounts of deviation from the geometrical parameters imposed by the given modular component.

1. Mathematical Knots

Mathematical knots are fascinating objects. They are closed loops that are defined only at the topological level – by the crossing of different branches of the overall loop in 3D-space (Fig.1a), or by the connectivity of the complement space. A particular knot can take on many different geometrical shapes, and there is still no computer program that can unambiguously prove that two different looking versions of the same mathematical knot are indeed the same.

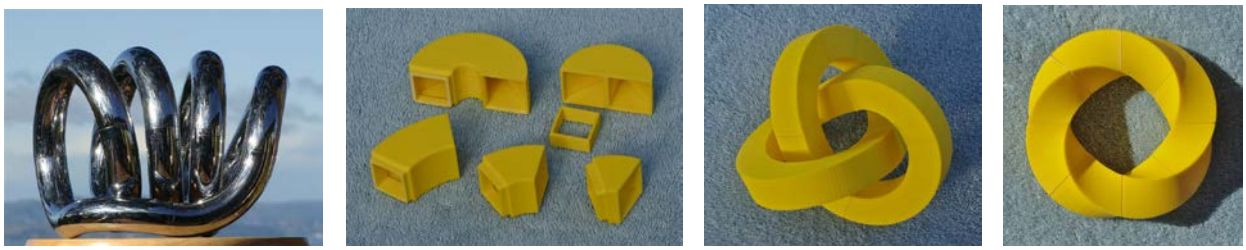
(a) (b) (c) (d) (e)

Figure 1: *Knot 10_{19}; (b) knot construction from tubular elements [9]; (c) Zawitz sculpture [11]; (d) bronze cast of "Torus knot 5,3" [4]; (e) "Music of the Sphere" (Trefoil or knot 3_1) [1].*

Physical realizations of mathematical knots, assembled from tubular elements (Fig.1b,c) or cast in metal (Fig.1d,e) can also result in artistically pleasing objects. Some large-scale knot sculptures cast in bronze (Fig.1e) [1] are expensive and take a lead-time of many months from design to the time when one can enjoy them. Thus, for quick experimentation and exploration of mathematical knots, it is desirable to have a set of snap-together parts, with which one can quickly construct physical models of arbitrary mathematical knots. Project LEGO-Knots [5] used some generic "snap-together" parts [6], supplemented with a few custom-made parts. A small set of different components (Fig.2b) – inspired by the LEGO system – has been designed to form most of the geometrical features found in sculptural knot models (Fig.2c). Different modules have different lengths and bend through different angles. The cross section of these tubular elements is not just a circle, but is a square or an equilateral triangle; this allows more interesting sculptural models to be constructed. These prismatic cross sections make it possible to form twisted structures (Fig.2d); for this reason, different modules come with different amounts of twist. However, often an additional custom-part had to be designed to guarantee a smooth closure of a particular knot.

This paper focusses on a different approach that relies on just a single generic module from which all possible knots are supposed to be constructed. One such system is represented by the "Museum Tangles" by Richard Zawitz (Fig.2a) [12]. They are composed of 18 tubular elements that bend through 90 degrees.

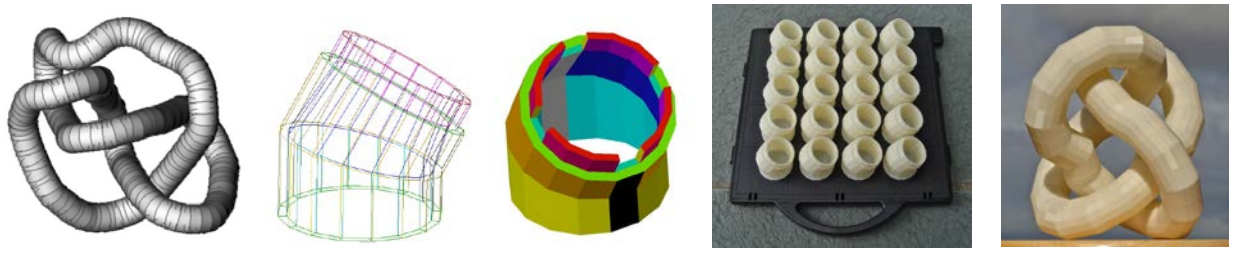
As packaged, the *Tangles* form a single un-knotted loop, which mathematicians call the *Un-knot* or the *Trivial knot*. Some of these tangles can be pried apart, and several of them can then be snapped together to form more complex, truly knotted structures. Another system has been proposed by Zawidzki et al. [10]. Their basic module is a rather short "wedge" that bends only a little bit. This makes it easy to form "organic" looking worms of possibly high knottedness (Fig.3a). Their proposal does not discuss how one might guarantee perfect closure of the loop or how to achieve some desired overall symmetry.



(a) (b) (c) (d)

Figure 2: *(a) Zawitz "Museum Tangle" [12]; (b) LEGO Knot modules; (c) trefoil; (d) twisted loop.*

This paper focuses on knot model construction from just a single module that could readily be mass-produced by injection molding, but which can also be easily be fabricated on a low-end 3D-printer (Fig.3d), since it needs no supporting scaffolding, which then must be removed manually. In our current project, we want to form knots of high symmetry with far fewer modules than what would be needed by the *Pipe-Z System* [10] (Fig.3a). After some experiments, we have focused on an "elbow" module, which we also call the *Λ–Module*, consisting of two short tubular segments joined at an angle of 30° (Fig.3b,c). With a bending angle, β, of only 30°, rather than 90°, we are able to construct smoothly bending knot models without excessive meandering wiggles. With only this single *Λ–Module* available, the resulting, closed, possibly knotted, tubular loop is completely specified by the torsional angles (the rotation around the cylinder axis) by which two adjoining modules fit together. To allow an artist to follow precisely the given design information, we have designed our *Λ–Module*, with a cross section in the shape of a regular 16-gon, rather than a circle, which then offers 16 discrete torsion angles, separated by 22.5°. The value 16 is large enough to offer sufficient flexibility in constructing pleasing looking knots, yet coarse enough, so that a designer can readily chose the desired torsion angle from the 16 "clickable", "legal" rotational positions. The resulting component is shown in Figure 3c. Its outer tube diameter is 34mm and the axis length of each leg is 15mm. The tightest toroidal loop, composed of 12 modules, has an inner tunnel with a diameter of 70mm.



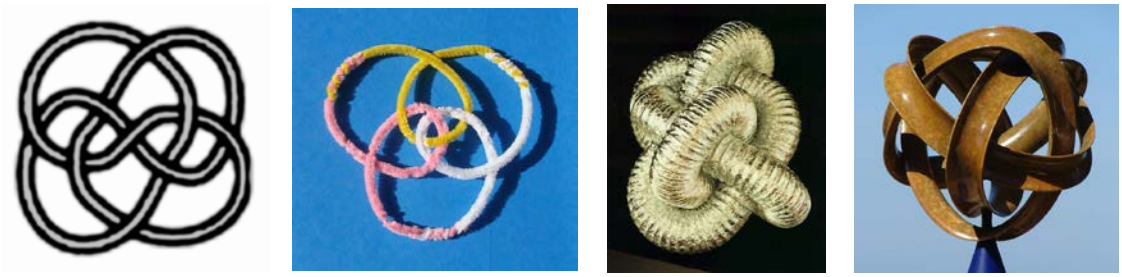
(a) (b) (c) (d) (e)

Figure 3: *(a) Figure-8 knot formed by the "Pipe-Z" system [10]; (b,c) our new "Λ-Module"; (d) 3D-prints of 20 Λ-Modules; (e) Figure-8 knot formed from 40 Λ-Modules.*

2. High-Level Design Issues

A knot specification like *knot* 3_1 (a trefoil) or *knot* 4_1 (the figure-8 knot) just defines the knot topology, but says nothing about the geometry of a particular knot realization. The users must decide what kind of knot model they would like to obtain; whether they want a relatively flat, "2.5-dimensional" configuration that resembles the diagrams (Fig.4a,b) in the knot tables [3] or whether they prefer a more space-filling,

"spherical" realization, which might form the basis for a design of a constructivist 3D sculpture (Fig.4c,d). The user must also specify explicitly the degree of geometrical symmetry they would like to obtain. The diagrams in the knot tables often do not show the highest degree of symmetry that is inherent in a particular knot. As an example, the *Chinese Button Knot* (*knot* 9_{40}) can be realized with dihedral (D_3)-symmetry of order 6 (Fig.4b,c,d), but the standard depiction (Fig.4a) gives no hint of this. The user must also indicate how loose or how tight a representation is desired.

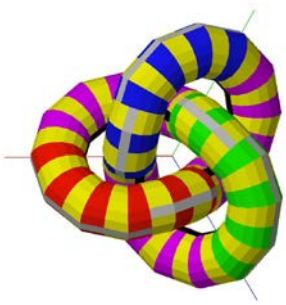


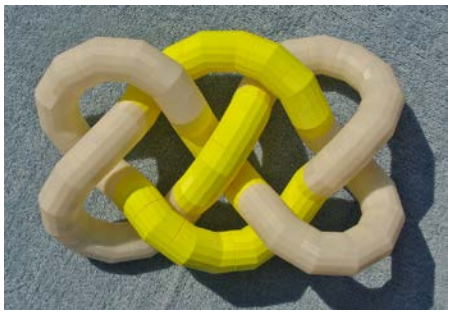
(a) (b) (c) (d)

Figure 4: *Chinese Button Knot: (a) diagram for* 9_{40} *in Knot table; (b) another layout of this knot; (c) a tight 3D configuration; (d) an artistic realization of this knot. (Artwork by C. H. Séquin).*

We have created a couple of experimental CAD tools to assist a designer in realizing a desired knot based on a single modular component. In our CAD systems, the *Λ-module* is specified by its *bending angle*, β, (deviation from a straight line, in degrees), the *leg-length*, *L*, of the two tubular segments emerging from the central node of the module, the *tube-diameter*, *TD*, of the two legs, and the *number of discrete torsional angles* allowed, (e.g., defined by the 16-gonal cross section at the tube joints).

Our first CAD tool to design such modular knots was an interactive graphics tool implemented with Berkeley SLIDE [8], a graphic modeling system that allows parametric, procedural specifications. The designer sets all independent torsion angle values with a corresponding number of interactive sliders (grey field on the left). The modeling system then constructs a corresponding branch by assembling consecutive modules with the specified torsion angles. If the overall symmetry is of type D_n, then the constructed branch will already have C_2 symmetry. A set of *n* copies of this branch are then placed on *n* appropriately spaced, radial C_2 symmetry axes, where the designer can move these branches radially along those axes and rotate the branches around them in order to obtain the best match at the joints where two of these branches come together. Knots with only five or six independent torsion angle values can typically be found in a couple of hours of trial and error experimentation [6]. Examples of knots that have been constructed with this interactive graphics tool are shown in Figure 5.





(a) Trefoil, 33 modules (b) Cinquefoil, 50 modules (c) Knot 7_4, 64 modules

Figure 5*: Successful modular knot constructions.*

More complex knots, where a larger number of torsion angles need to be set, require more assistance from the CAD tool. The focus of this paper is the development of an optimization system that assists the user in finding a good set of torsion angles for all the joints to produce a knot model (Fig.3e) close to what the user may have had in mind, when imagining the knot being formed from a flexible hose or drier duct (Fig.4c).

3. Looking for Optimal Solutions

The users describe the desired knot they aim for with a polyline or a B-spline embedded in 3D space. The number of vertices on the poly-line, or the number of sampling points on the B-spline, indicate the number of modules the user plans to use for the knot construction. A first adjustment is a uniform scaling operation that assures that all the distances at skewed branch crossings exceed the tube diameter, (*TD=34mm)*, of the available module. At the proper scale, the total length of the knot curve is then evaluated. Dividing this length by ($2L$=30mm) yields a lower bound for the number of modules that could realize this particular knot geometry. Since we want the final knot to exhibit maximal symmetry, we choose the number of modules to be a multiple of n for a knot with C_n or D_n symmetry. The n C_2-symmetry axes must then pass through some module centers or through the joints between two modules.

It is useful to do one more test in this initial planning stage: All the bending angles between pairs of adjacent segments of the sampled B-spline curve are checked. If any of these bending angles exceed some upper bound, e.g., 2*30°, then the designer will be advised to make the starting curve "rounder."

We now let the vertices of the polyline represent the center points of our given modules. In a physically realizable solution, these vertices must be spaced from their neighbors by a distance of $2L$, given by the length, (*L=15mm)*, of the legs of the tubular module. Using greedy optimization, based on gradient descent, the distances between neighboring vertices in the poly-line can be pushed towards $2L$, and the bending angle, β, (the deviation from a straight-line tube) at each vertex is forced to approach 30°. If the desired knot curve has some relatively straight branches, then in these branches, the solution will have to introduce some zig-zag or helical sweep paths, so that the angles between subsequent segments in the poly-line are always 150°.

If our *Λ-Module* had a circular cross section then a greedy optimization with only two types of error terms can often deliver a solution with no local deviations from the ideal geometrical characteristics. One of the two terms is a *total length error, TLE,* obtained by summing the squares of the differences of any individual segment length from the ideal length of $2L=30mm$:

Total Length Error: $TLE = \sum (length_i - 2L)^2$.

The other error term is the deviations of the actual bending angles at each vertex (the angle formed by three consecutive points on the polyline) from the specified value β (β=30° in our examples):

Total Bending Error: $TBE = \sum (bend\text{-}angle_i - \beta)^2$.

There is some ambiguity as to how these two error terms, *TLE* and *TBE*, should be combined into a single overall penalty value that then can be minimized. What are equally "bad" (noticeable) deviations in these two categories? By physically manipulating assemblies of our *Λ-Module*, we determined that a practical error threshold for visibility of any geometrical deviation from a perfect assembly is about one degree for the bending angles and half a millimeter for module separations. Such deviations would be barely noticeable for our *Λ-Modules* fabricated on inexpensive 3D-printers. We assign both of these error thresholds a penalty value of 1.0. The assigned penalties are increased quadratically with increasing deviation. All the local penalties are summed into a global penalty value, which is then divided by the number of error terms being evaluated; in the optimization above, this is equal to twice the number of modules. This results in an averaged penalty (error) measure that is independent of knot complexity.

Given that we want to find a perfectly symmetrical solution with C_n or D_n symmetry, the number of individual torsion angles that need to be determined is reduced by a factor of n, or by $2n$, respectively. However, this also adds more constraints and reduces the number of possible solutions. These symmetry

constraints are captured in a hierarchical description of the knot poly-line, given by n or $2n$ identical pieces; therefore, the computational effort of finding the optimal vertex positions is also reduced by the same factor.

But, a physically realizable solution may not always exist. If we wanted to make a toroidal loop with only eleven components, then this cannot be realized with a bending angle β of 30°. In this case the optimization would result in the same slanted gap between all 11 modules, and the final penalty would be:

Penalty of 11-module torus = $11 * (30/11)^2 / 22 = 3.719$ (penalty units).

We also need to check, whether the final "wiggly" knot axis has no skewed branch crossings with a separation of less than the tube diameter, *TD*, of the module. It is even possible that the resulting polyline now has a different knotted-ness, because the optimization process may have caused some inadvertent branch crossings that could have changed the knot type. Both these concerns can be addressed during the optimization process itself by introducing an additional error term that increases dramatically as two branch elements approach each other to a distance of *TD*.

4. Discrete Torsion Angles

Any given solution coming from the above optimization process, will be presented in the form of a list of torsion angles. These angles may take on arbitrary values. But, how would a user realize a torsion angle of, say, 142.7° between two subsequent nodules? This is why we have given our module a cross section of a regular 16-gon, with 16 precisely defined torsion angles to choose from. A 16-gonal cross section can be robustly realized on inexpensive printers, and it also is reasonably convenient when assembling a particular knot model, giving just the list of torsion angles. To make it even easier to apply the correct torsion angle when joining together two *Λ–Modules*, the flanges needed to make the tubular joints are notched to give a ready indication of some particular torsion angles, such as 0°, ±90°, or 180° (Fig.3b).

Suppose we want to construct a trefoil knot with perfect D_3-symmetry from 30 *Λ-Modules* with 16 "legal" torsion angles. We may first form a branch of 10 modules by adding 5 modules on either side of a central joint, maintaining perfect C_2-symmetry. This symmetrical branch is entirely specified by 5 discrete torsion angles, 4 of which are applied pairwise symmetrically on either side of the central joint. The question then is, what set of 5 angles will define a *branch*, where three copies of it can be assembled into a trefoil knot with a low enough error value at the three joints between those *branches*; the penalties at all other joints are zero by construction. This now changes the optimization domain from a smooth, contiguous solution space into a discrete set of grid points, where gradient descent can no longer be used.

However, this approach, which was inherent in our first graphical modeling tool [6], is not optimal! In an actual, physical realization, errors will naturally distribute themselves over all the 30 joints in the knot model. At every joint, there could be a small axial misalignment and a little bit of torsional deviation from one of the 16 admissible angles – hopefully rendering all of these deviations almost invisible.

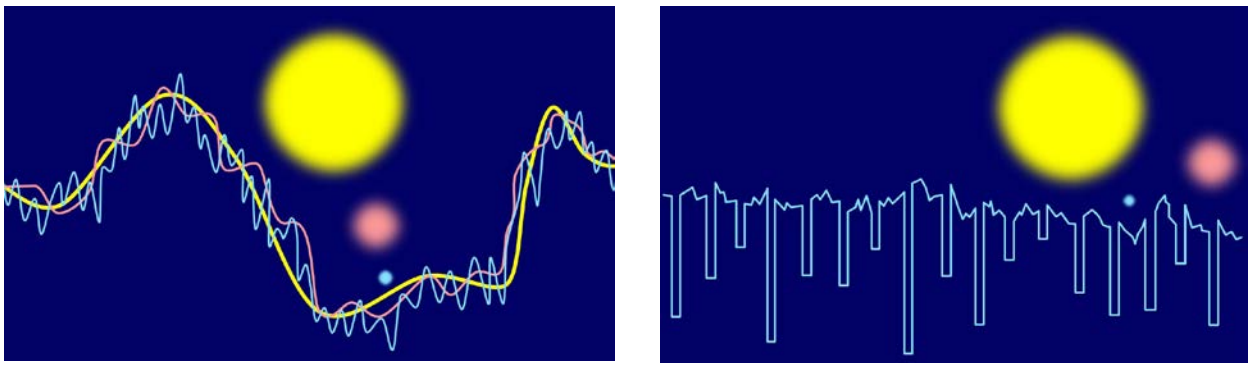
Thus, to evaluate the true quality of any discrete choice of torsion angles, we need to perform an additional gradient descent optimization. In this process, the actual torsion angles will deviate from their ideal discrete values, and additional small errors will build up in the bending angles and module separations in order to reduce the errors at the three branch junctions. In the overall penalty function, we equate a 1° deviation in a torsion angle with a 1° deviation in bending angle. It is then among these optimized, "nearly discrete" solutions that we need to find the best possible candidates.

5. (Unsuccessful) Optimization in a Discrete Solution Space

Two often-used techniques in the domain of discrete optimization are *Genetic Algorithms* [2] and *Simulated Annealing* [7] Genetic Algorithms are particularly useful when the entity to be optimized has a limited set of discrete characteristics that can be optimized more or less independently of one another. Examples might be the eyes, ears, and noses in an animal, which together contribute to an animal's survivability. In our knots, all torsion angles are equally important, and they all work "in concert" with one another. Except for random "mutations," Genetic Algorithms do not offer a useful approach for this particular design problem.

Simulated Annealing is a technique that is widely used in the placement and routing of integrated circuits (IC). A given set of components, and all the required wire connections between them, need to be placed as tightly as possible on a small IC chip. The algorithm tries to improve the current solution, by re-locating and re-orienting a few components at a time. We cannot expect that there always exists a possible swap of a few components that will improve the current solution. Therefore, the algorithm, particularly in the early phases of the optimization, will accept moves that lead to slightly worse results, hoping that from this new position, there are other moves that will result in an overall improvement. The accepted amount of additional error in a particular move is equated with a "temperature" of the system, where higher temperatures are synonymous with larger uncertainties of what constitutes a "good" move. This generic optimization paradigm seemed most suited for our problem domain, where we try to make the best possible changes to our discrete "legal" torsion angles. However, almost all "small" digital moves, i.e., changing one torsion angle by one discrete "click," resulted in a large increase in the overall error value, and we have not been able to find a strategy that would guide us reliably and efficiently towards "the best" solution.

For many types of problems, such as placement and routing of components on an IC chip, Simulated Annealing [7] works well, because there is some correlation between similar moves and the benefit they yield with respect to the overall error function. This corresponds to a solution space (depicted for just one dimension) that looks like Figure 6a. Rolling around a "big fuzzy ball" (corresponding to a high temperature) in this landscape would let it find the deepest large (yellow) valley. Then, as the temperature is reduced gradually, the diameter of the ball shrinks, and it will start to explore the finer (pink) structure found in this large valley. Eventually the ball will be small enough (blue) to fall into one of the little crannies at the bottom of the big yellow basin.



(a) (b)

Figure 6: *Solution space profiles: (a) a well-behaved function; (b) difficult "golf-course" profile.*

To understand whether the solution space of our design problem has the appropriate structure for Simulated Annealing to work, we have studied the solution spaces of trefoil knots assembled from 30 or 33 *Λ-Modules.* For this simple, highly symmetrical knot with only five independently selectable torsion angles, we could perform an exhaustive search. For each combination of discrete torsion angles, we performed a quick analysis to see whether it would have a chance to produce the desired trefoil knot; this eliminated most of the one million possible combinations. The optimization process was then run only for these promising, "constructible" combinations (about 4300). This allowed us to study the relationships between "digital neighbors", i.e., knots that differ by only one or two small changes in the discrete torsion angles.

The results were rather surprising and illuminating: There are only very few digital neighbor pairs where both of them have low error values. This indicates that the structure of the solution space looks more like Figure 6b, which could be characterizes as a "devil's golf course" – a rugged surface with isolated pockets of different depth. In such a solution space, there is no good strategy to move from one good pocket to another one. One good solution offers no indication that there might be other good solutions nearby (in the digital sense). Thus, searching for the best solution in the purely digital domain is <u>not</u> a good approach.

6. A More Promising Approach

Recently we have started to pursue a different approach, in which we retain more information from the contiguous solution space of the initial polyline with arbitrary torsion angles. We now investigate how this information can guide us to digital combinations that are more likely to deliver good solutions.

One approach might be to let the designer make small changes to the initial knot curve, perhaps by changing some parameters of the defining B-spline. Perhaps, the initial B-spline representations for the knot was too smooth. The final polyline of any realized modular knot has many wiggles and undulations, since it must bend through 30° at every module center. In a lucky situation, these undulations may be exploited to route one branch nicely around an adjacent branch. But, to find such opportunities, the initial knot curve would have to rely on several more control points, and it would be difficult for the designers to add such details into their initial knot geometry specification. Thus, we found no good strategy to guide the designers in how to make small random changes to the initial knot curve, hoping to stumble across the right kind of undulations that would then result in a modular knot with a low overall error value.

Another option might be to shift the initial sampling points along the B-spline. But, an arbitrary shift in the sampling positions would break the strict symmetry of the desired knot geometry, and we disallow this operation. However, there is a second sampling option that maintains the chosen symmetry: Any possible C_2-symmetry axis must either pass through the center of a module or through the junction between two of them. Thus, we should evaluate the original knot curve with twice as many sample points as the number of modules we plan to use and then separately evaluate and optimize the polylines resulting from the even samples as well as the from the odd numbered ones.

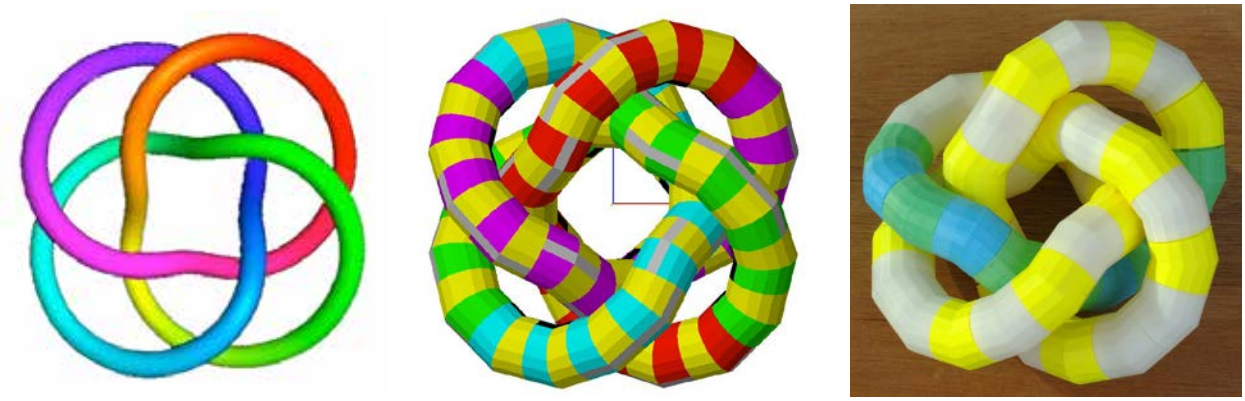
In the purely digital realm, a knot design with A individually choosable torsion angles, that maintain the specified overall symmetry, offers 3^A digital neighbors where individual angles differ by no more than ±1 "clicks" from the initial digital combination. For A=10 this gives us 59,049 options to explore, – with no indication which ones of these may be more promising to offer a low overall penalty.

If we remember how any particular discrete torsion angle was chosen by rounding the initial analog angle, we only have to focus on two options. Each analog angle has two closest legal values, which can be reached by rounding either up or down. Thus, we may limit ourselves to evaluate only 2^A combinations. For A=10 this is 1024 and thus equivalent to a speed-up of more than 50 times!

But, we can be even more selective, since we have some indication as to which options may be more promising. If one of the torsion angles of the initial polyline is close to half-way between its two nearest "legal" angles, might it not be likely that rounding to the other, slightly farther away legal torsion angle may yield a solution just as good, or perhaps even better? Thus, we evaluate every analog torsion angle for its potential to be rounded in the alternate direction. Any rounding operation is associated with a local penalty equal to the square of the angle change needed to reach a "legal" value. With A independent torsion angles to be set, we might evaluate all 2^A combinations of different rounding directions for their initial costs associated with rounding in one or the other direction. We may then choose the 100 "least expensive" combinations and perform gradient descent into the corresponding local optimum. Among the solutions with low enough overall penalties, the designer may then choose one, either because it has the lowest penalty value or because it is most aesthetically pleasing, – perhaps because it has the fewest gratuitous undulations, or the most uniform clearances between skewed crossings of adjacent branches.

Based on some limited early results, it seems that small, tight knots offer the most challenges. Larger, more complex knots, while taking more computation time for each optimization run, actually are more often giving good results with low average penalty. A larger number of modules offers more junctions, where a little deviation from the ideal geometry can be introduced to achieve the overall proper closure of the knot loop. Physical implementations with more modules result in more flexible assemblies, which can more easily be deformed to get the last junction to match up properly.

Figure 7 shows the example of a (4,3) torus knot with D_4 symmetry. It is composed of 48 *Λ-Modules*. The six torsion angles in one of the replicated 6-module branches are: 0°, 90°, -67.5°, 0°, -45°, 22.5°, respectively.



(a) (b) (c)

Figure 7: *(4,3)-torus-knot: (a) knot curve; (b) modular CAD model; (c) physical realization.*

7. Conclusions

At the moment, this is work in progress, and the three authors are the sole users of this emerging system. A few different optimization approaches have been tried and rejected. We are now focusing on a more promising approach that seems to yield more readily solutions with low error values. A broader evaluation of its performance statistics has been started. Once all algorithmic issues have been fully resolved, our next goal is to provide this system with a decent user interface, so that designers without any computer-science training can easily design modular mathematical links and knots with high complexity and high symmetry.

References

[1] B. Collins, D. Lynn, S. Reinmuth, C. Séquin. "Realization of Two New Large-Scale Sculptures." FASE 2012, – http://people.eecs.berkeley.edu/~sequin/PAPERS/2012_FASE_NewSculptures.pdf

[2] Genetic Algorithms. – https://en.wikipedia.org/wiki/Genetic_algorithm

[3] Rolfsen Knot Table. – http://katlas.org/wiki/The_Rolfsen_Knot_Table

[4] C. H. Séquin. "Torus-Knot_5,3." -- http://people.eecs.berkeley.edu/~sequin/ART/Science_Sculpture/

[5] C. H. Séquin. "Lego Knots." *Bridges Conf. Proc*. pp 261–270 (2014) – http://archive.bridgesmathart.org/2014/bridges2014-261.html

[6] C. H. Séquin. "Reconfigurable Snap-Together Sculpting." Computer-Aided Design and Applications, (2015). – http://people.eecs.berkeley.edu/~sequin/PAPERS/2015_CAD-A_SnapSculpt.pdf

[7] Simulated Annealing. – https://en.wikipedia.org/wiki/Simulated_annealing

[8] J. Smith. "SLIDE design environment." (2003). – http://www.cs.berkeley.edu/~ug/slide/

[9] F. Smullin. "Analytic Constructivism: Computer-Aided Design and Construction of Tubular Sculptures." Luncheon Presentation, 18th Desigri Automation Conference, Nashville, TN, (June 30, 1981).

[10] M. Zawidzki, K. Nishinari. "Modular Pipe-Z System for 3D Knots." Jour. Geometry and Graphics, pp 81-87, Vol.17 (2013).

[11] R. X. Zawitz. "Tangle Creations." – https://www.tanglecreations.com/pages/richard-x-zawitz

[12] R. X. Zawitz. "Museum Tangle." – https://www.tanglecreations.com/search?type=product&q=museum+tangle

The perception of fluid properties and how it influences an artistic direction in 3D printing

Dominique Claire Matthew and Oleg Fryazinov

The National Centre for Computer Animation, Bournemouth University, UK

Abstract

In this work we explore this fusion of art and technology through the 3D printing of fluid simulations and what limits the design of such simulations before perception is abstracted. The challenge of 3D printing complex forms presents itself through the comparison of the artefacts and what was rendered online. The properties of fluid are controlled using viscosity, speed and collision objects and a static frame of the simulation is 3D printed. This paper presents the potential pipeline to 3D printing of fluids and the process undergone to achieve aesthetically pleasing results.

Introduction

3D printing is a subject that is now entwined in the extensive areas of life. From medicine to industrial applications, this form of manufacturing has aided design and advancement from the stages of prototypes to the final output. This method became embedded within the art community as a new way to creating forms that would otherwise be extremely difficult or almost impossible. Artists can now create physical forms without having to restart processes from scratch when they wish to go back a step or edit a piece as the 3D model is saved on 3D software; thus, providing the creative freedom of unlimited iterations.

In this work we explore the artistic side of fluids, engaging in the 3D printing of computer-generated fluid, and the limitations that affect our perception of what makes a 'fluid form'. We present a range of sculptures that demonstrate fluid motion in abstract ways whilst also retaining the visual properties that make the fluid recognisable as one. The reason behind the interest of printing fluid simulations from 3D software came from wanting to push the boundaries of printing thin meshes before the printer found it difficult to proceed with the formation of them. This interest echoes from 3D printing being described as having an unlimited range of possibility of what can be produced, even with complex designs.

This paper presents the potential pipeline to 3D printing of fluids and the process undergone to achieve aesthetically pleasing results. This pipeline can be used by creators with 3D software and 3D printing hardware to create intricate freeze-frame forms of fluid with the only restriction being the (lack of) thickness of the mesh.

Related works

Fluid-shaped or fluid-inspired artworks existed for some time yet being quite rare because of the way of producing it. For example, the artist Aylin Bilgiç creates porcelain bowls [1] with gold-accented droplets (Regus, 2017). Annaluigia (Annalù) Boeretto uses resin-glass to create her 2012 sculpture Un Salto Nel Blu [2] of which takes on average 24 hours to reach 95% of its full cure. Artists such as Boeretto and Bilgiç have more control with the overall look of their sculptures as they are handmade, unlike Jack Long, a fluid photography artist, who creates intricate fountains containing various colours of liquids [3]. Eyal Gever, a contemporary artist with a passion for tech-art, has created a two-metre sized Waterfall 3D print [4], which captures the essence of a destructive force of nature. The main inspiration for this work came from Fung Kwok Pan's Fluid Vase [5], an interactive 3D printed ceramic piece where the customer can

alter the height of the fluid being poured, the container shape as well as when they want to pause the simulation, and hereby this artist avoids the act of randomness.

Method overview

Sculpting of fluid-like shapes is not a trivial process which in our method was separated into several steps. On the first step the fluid was simulated in specialised software such that the overall shape follows the artist's aesthetics. On the second step the simulation is converted into the shape representing 3D solid model and on the last one the model was fabricated by using 3D printing to assess the resulting sculpture. Artistic direction was important on every step of the process as we are discussing below.

In this work the RealFlow software was used to simulate the fluid. The software allows changing a wide range of parameters in order to get the desired shape. The artistic control was implemented with taking parameters such as viscosity, pressure/velocity fields and simulation ratio into an account.

The main parameter which controls the shape of the sculpture is viscosity. As it was noted in [6], observers ordered their judgements of liquids based on their variation in viscosity and when the liquid behaved differently from being poured, viscosity deviation was still obvious.

Figure 1: *D-Spline meshes with viscosities/speed: 100 Pa.s / 10m/s, 50 Pa.s / 1m/s and 3 Pa.s / 1m/s.*

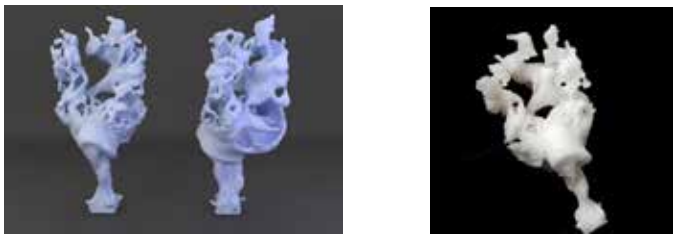
The early experiments show that constant viscosity, especially of high value, turned out to be of a little artist control and resulted in undesirable shape. It was noted that as the viscosity increased, there was a significant decrease of the volume in the main component of the shape, which was noticeable when the extra components were removed (Figure 1). In our work the viscosity was further controlled by splines that follow the flow of the simulated fluid and allow to be modified by the artist. In the RealFlow software this was implemented with so-called D-Splines. The decision to use splines was inspired by the works of David Lund, in particular "Liquid Gold" [7], of which the fluid appeared to be controlled by other forces rather than gravity. Unlike simulating fluid through pouring, specifically those with high viscosities, using splines allows more creative flow and unique contours due to the controllers of the vortex strength, axel strength and the radial strength. Variations in the speed of the D-Spline which allowed more particles to flow through at a greater rate, despite the increase of viscosity of which increases the difficulty when determining the velocity distribution. As a result, we were able to find the balance between lower viscosity areas that allowed to achieve a more visually aesthetic form that has depth and tone due to the holes in the mesh due to the free-flowing particles and higher viscosity areas that resulted in thinner shapes with high number of disjoined components.

Variation of parameters on the splines allowed also to control the shape by using modifications in the velocity fields. D-Splines also contained some vortex parameters that affected the rotational force of the fluid (Figure 3c). When viscosity was introduced into velocity field simulations, the form began to abstract and take on coral-like formations (Figure 3c), similar to Noiterksa [8].

The shape of the fluid simulation was controlled by using collision objects and varying the number of simulated frames. In the early experiments the fluid simulation of a box-shape object was interrupted

after 3-5 frames and the collision object was added after that as an extra boundary condition that allowed to get the fluid-like appearance due to the ripples and ridges created that acted similar to a heightfield to represent a 2D shallow water equation [10]. Further control on the shape of the fluid was achieved by using additional external and internal pressure fields. External pressure fields allowed to dynamically change the form of the fluids, adding the visual cues of heightfields as well as overhangs and droplets on the surface. Internal pressure field was more difficult to implement, but with extra particles emitting from the fluid some variations were allowed.

For some shapes the result of the fluid simulation itself was not sufficient enough and further modification of the shape was required. The resulting "amalgamation" process, which was inspired by Kevin Mack's [9] and Alberto Seveso's works [11], was achieved by further sculpting based on simulated shape. In our work (Figure 2) the shapes from the fluid simulation were converted into the mesh object and then merged and combined in ZBrush software. The result still captures the perception of fluid and its properties yet having its own unique free-flowing final form.



(a) (b)

Figure 2: *Amalgamation: (a) render, formed by combining two previously created fluid meshes, (b) The 3D print, some parts of the resulting object were too thin and broke due to the fragility.*

One drawback of the fluid simulation is that the resulting shape can have several disjoined components which cannot be 3D-printed as a single object. In this work we had to remove extra components leaving only the main one. Occasionally this would have an impact on the aesthetic of the piece, although, to combat this, the entirety of the mesh could then be taken into the sculpting software and the simulation could be conjoined manually which would alter the perception of said fluid.

Majority of the 3D prints were fabricated as solids with translucent plastic, i.e. the infill was set to 100% providing a smoother and more visually appealing finish. Translucent material made it look more like a fluid whereas non-translucent materials felt and looked artificial even if the colours of water blue-green palette was used. As majority of the prints required support structure, the support was added either of the same material as the mesh to be printed or soluble support material which helped with the construction of the shapes.

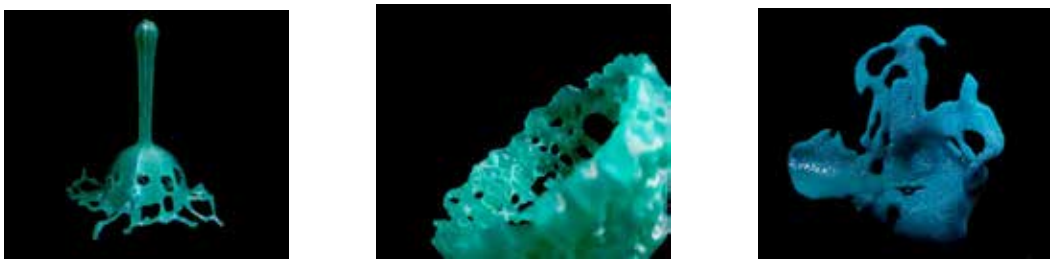
Results

The process of getting the final sculpture in our work was formalised into the following pipeline:

1) Set up parameters for the simulation and simulate fluid;
2) Export the mesh object from fluid simulation;
3) If needed, adjust the resulting mesh in the sculpting software;
4) Run component analysis on the mesh object and remove all the components except of the main one, i.e. the one with the largest volume;
5) Send the one-component mesh object into the slicing software and 3D print the object.
6) Clean up the resulting 3D printed object from the support material.

The results show that the pipeline proved to be a successful way to create fluid-like shapes. The pipeline worked in sync with the software used with mathematical calculations of fluid happening at the start of the process and visualisation happening at the end. The Ultimaker was successful with printing small yet complex designs, particularly those that had features with a high chance of breaking within the

structure during creation such as the Amalgamation piece (Figure 2). Some of these prints required to be printed multiple times specifically the external-force test that was printed with and without support (Figure 3b). At the same time the drawbacks of low-budget 3D printers were apparent during digital fabrication process. Although soluble support material was used, the resulting object sometimes was too fragile which resulted in unsuccessful 3D printed object. Moreover, for some objects, soluble support material was even more destructive than helpful due to the weight of the support material as it was dissolving on the fragile parts of the tendrils.



(a) (b) (c)

Figure 3: *Results: a) The Initial-Fluid print, b) External-Force print. c) High-viscosity Vortex print*

Summary and Conclusions

This paper presents an insight into the process of using 3D printing to create a fluid-like sculptures. This project aimed to establish the pipeline and justify the idea rather than create an artefact for exhibition.

This project has many aspects which were not properly explored in detail. For example, the choice of materials was limited to types of plastics supported by 3D printer we used. For more detailed investigation it would be useful to take a look at other materials such as metals, ceramics and resin. Also the scale of the resulting prints in this work was relatively small, with the average size being 60mm x 45mm x 50mm, while sculptures of professional artists tend to be a lot larger. In order to get a larger scale, the design should be planned more carefully.

The artistic direction of printing fluids is still limited due to the issues there are with printing thin meshes as well as the difficulty of having complex and entwining structures. We hope one day 3D printing will offer an unlimited choice of what can be printed. Ultimately, the scope as to what can be perceived as a fluid is unlimited as, even when abstracted, there are properties that present visual cues of that help to determine the sculpture as being a fluid.

References

[1] K. Regus, 2017. *Une fluide céramique par Aylin Bilgiç*. http://www.designmaroc.com/blog/2017/12/15/fluide-ceramique-aylin-bilgic/.

[2] A. Boeretto, *LIQUIDITY*. http://www.annalu.it/en/liquidity/.

[3] J. Long, *Liquid Art Photography*, 2016. https://www.jacklongphoto.com/f437571659.

[4] E. Gever, *Waterfall*, 2017. http://www.eyalgever.com/waterfall/.

[5] F. Kwok Pan, *Fluid Vase,* 2010. https://www.designboom.com/technology/fung-kwok-pan-fluid-vase/

[6] J. J. R. van Assen, J., P. Barla and R. W. Fleming. "Visual Features in the Perception of Liquids." *Current Biology*, vol. 28, no 3, 2018, pp. 452-458.

[7] D. Lund. *Liquid Photography.* http://www.davidlund.co.uk/david-lund-liquid-photography/

[8] N. Ervinck. *Noiterksa.* http://nickervinck.com/en/works/detail-2/noiterksa

[9] K. Mack. *Amalgamations*. http://www.kevinmackart.com/amalgamations.html

[10] N. Foster, D. Metaxas. "Controlling Fluid Animation". *Proceedings of CGI'97*, pp 178

[11] A. Seveso. *Illustrations & Photography*. http://www.burdu976.com/phs/

Bi-Scale Porous Structures

Cong Rao[1], Fan Xu[1], Lihao Tian[1] and Lin Lu[1,2]

[1]School of Computer Science and Technology, Shandong University, China; [2]llu@sdu.edu.cn

Abstract

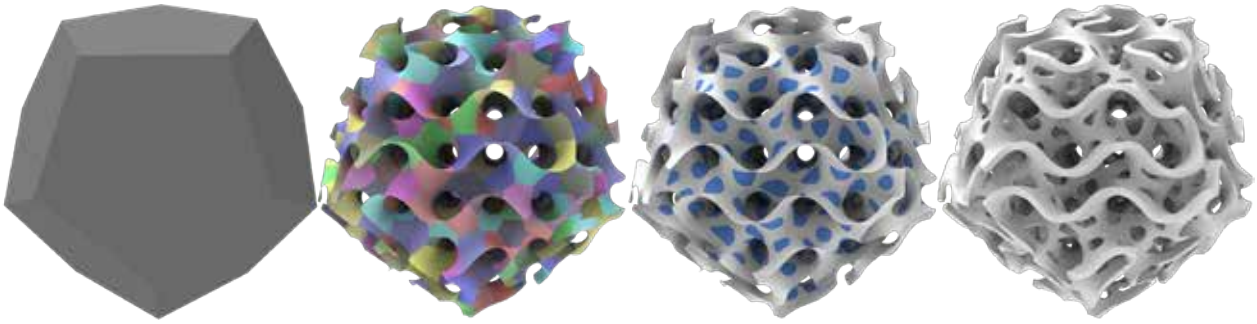
Porous structures are ubiquitous in nature and widely used for many applications. Besides the functionalities, the porous structures have considerable aesthetic appeal. In this paper, we propose a modeling framework to automatically generate bi-scale porous structures. We employ the minimal surfaces to control the large scale of the porous structures. Smooth B-Spline curves are then sculpted on the surface, such that the fine scale pores are achieved. Both the large and fine scale structures are controlled by parameters. Results can be directly fabricated and of aesthetics.

Introduction

Porous structures are of both functionality and aesthetics, which have drawn lots of attention in recent years. In order to achieve specific physical properties such as permeability, dispersion, resistivity, etc., researches try to control geometric characteristics of porous structures, like locations, patterns, and density of pores. However, it is still a tough task for designing irregular porous structures.

Minimal surfaces are defined as surfaces with zero mean curvature. I.e, at every point, the curvatures of the two maximal normal sections are equal but have opposite signs. This equilibrium of the curvatures is the basis for the aesthetic charm of minimal surfaces. Triply periodic minimal surface (TPMS) is a subset of minimal surfaces that extend periodically and indefinitely in space [1].

Our work draws inspiration from the Voronoi structure and TPMS. We aim at combining the merits of minimal surfaces and surface sculpting to generate the bi-scale porous structures.



(a) (b) (c) (d)

Figure 1: *Pipeline for generating the bi-scale porous structure. (a) A given dodecahedron shape; (b) the corresponding centroidal Voronoi tessellation on TPMS bounded by dodecahedron; (c) smoothly hollowed surface; (d) the fabricable, solid cast with uniform thickness.*

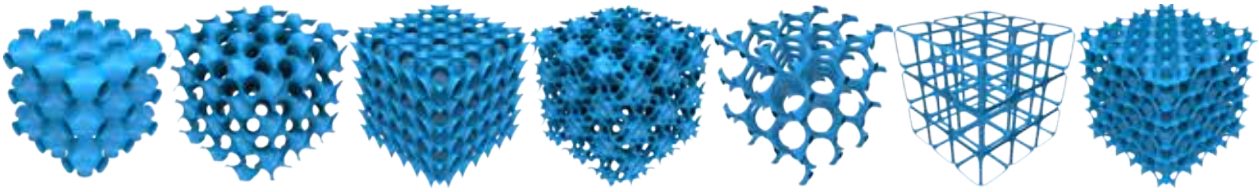
Overview

We propose a modeling framework to generate a bi-scale structure that is aesthetic and light-weighted. The TPMS is taken as the basic porous structure in a large scale, hollowed by B-Spline controlled smooth pores

in a fine scale. The algorithm pipeline is illustrated in Fig. 1. We first take a closed manifold mesh as the input boundary and generate the TPMS defined by an implicit function in the shape. Then we compute the centroidal Voronoi tessellation (CVT) on the TPMS. Taking the Voronoi vertices as control points of a closed B-Spline curve, we create a smooth hole in each Voronoi cell. After uniform extrusion, the bi-scale porous structure can be fabricated by standard 3D printers.

Technical Details

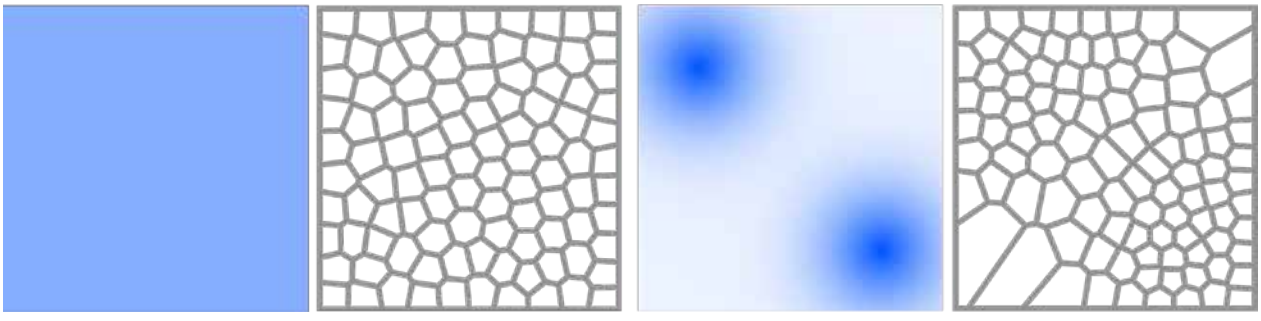
TPMS construction. A TPMS is a periodic implicit surface defined independently in three orthogonal directions. It defines a large family of surfaces which have a closed form implicit representation that results from convolving sin and cos terms along x, y, and z-axes in Euclidean space. The implicit function $\mathcal{F}(\mathbf{x}) = 0$ of TPMS can be generally expressed as $\mathcal{F}(\mathbf{x}) = \sum_{k=1}^{K} A_k \cos[2\pi(h_k \cdot \mathbf{x})/\lambda_k + p_k] - C$, where λ_k is the periodic wavelength, p_k is the phase offset, A_k is the amplitude factor, and C is a constant. The TPMS is mainly controlled by the four parameters A_k, h_k,λ_k, and p_k. Fig. 2 shows different types of TPMS.



(a) (b) (c) (d) (e) (f) (g)

Figure 2: *TPMS family. (a) P, (b) G, (c) D, (d) L, (e) Tubular-G, (f) Tubular-P, (g) FDR.*

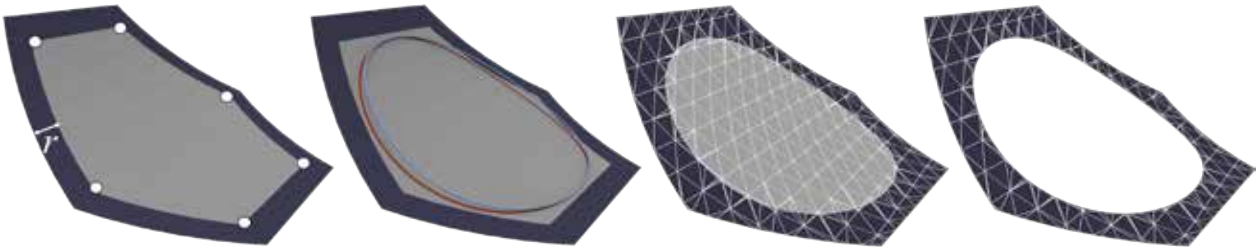
CVT-governed distribution. We adopt a hollowing operation on the TPMS to generate the bi-scale porous structure. Denote the bounded TPMS by S, we compute the centroidal Voronoi tessellation (CVT) restricted on S [2]. Let $X = (\mathbf{x}_i)_{i=1}^{n}$ be an ordered set of n sites on S. The Voronoi cell for $\mathbf{x}_i$ is defined by $\psi_i = \{\mathbf{x} \in S | d(\mathbf{x}, \mathbf{x}_i) < d(\mathbf{x}, \mathbf{x}_j), j \neq i\}$, where $d(\cdot)$ denotes the Euclidean distance. The CVT energy function is $E(X) = \sum_{i=1}^{n} \int_{\psi_i} \rho(\mathbf{x}) d(\mathbf{x}, \mathbf{x}_i) d\mathbf{x}$, where $\rho(\mathbf{x})$ is the density map that can be user designed. To minimize $E(X)$, we employ Lloyd's method [3] by iteratively updating the sites to the centers of their corresponding Voronoi cell. In each iteration the center of each cell is projected back on S. Fig. 3 shows two 2D examples under uniform and non-uniform distributions.



(a) (b)

Figure 3: *Illustration of 2D CVTs governed density map (deep color indicates high density). Two 2D examples are respectively under uniform (a) and non-uniform (b) distributions.*

B-Spline controlled hollowing. To generate the smooth pore in fine scale, we generate the restricted B-Spline curve in each Voronoi cell, similar to [4]. We first re-scale the cell ψ_i via width r, which implies the hollowing ratio. Then, we take the new cell vertices as the control points and use them to construct a closed B-Spline curve of degree three. We sample the closed curve and project the sampled points on S. We split and remesh the original triangular mesh in the intersection area between the projected curve and S, then we could get rid of the intersection to finally obtain a smooth, hollowed Voronoi cell. Once getting the final mesh for all cells, we extrude it along the surface normal direction to obtain a solid fabricable model. The hollowing process for each Voronoi cell is illustrated in Fig. 4.



(a) (b) (c) (d)

Figure 4: *Hollowing process in a Voronoi cell. (a) Scale down the cell by r and get new vertices (white dots). (b) Generate a close B-Spline curve (blue line), and project it onto the original mesh (red line). (c) Remesh the mesh considering the B-Spline curve. (d) Hollow the cell.*

Results

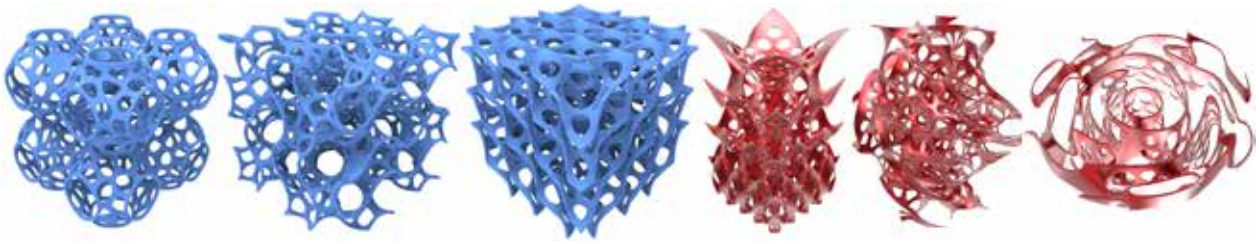
We show results based on different types of minimal surfaces and the two parameters, width r and the number of sites n. Fig. 5 shows various scale and density of Voronoi cells with width r from 1mm to 4mm and n from 200 to 600 on a $100mm^3$ P-type TPMS unit cell. Fig. 6 shows results on regular TPMS like P, G and D types and other distorted TPMS. More results can be found in Fig. 7.

(a) (b) (c) (d)

Figure 5: *Various width r(mm) and number n to control the scale and density of Voronoi cells on a P-type TPMS unit cell, (a) $r = 1, n = 400$, (b) $r = 4, n = 400$, (c) $r = 2.5, n = 200$, (d) $r = 2.5, n = 600$.*

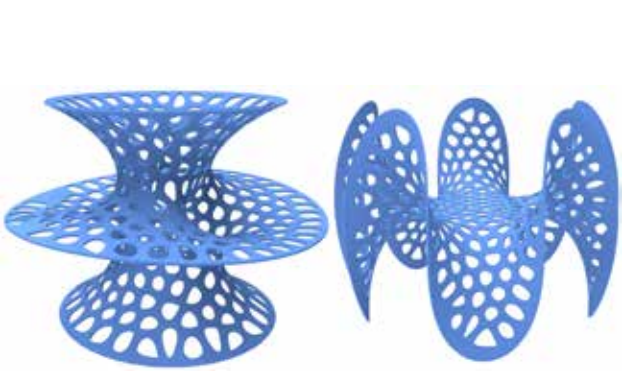
Conclusion

We have proposed a novel framework for modeling porous structure in a given boundary. Our bi-scale porous structures possess the properties like aesthetics, lightweight, boundary smoothness and ventilation. The large scale structure is created by triply periodic minimal surface bounded by the given shape. The fine scale pores

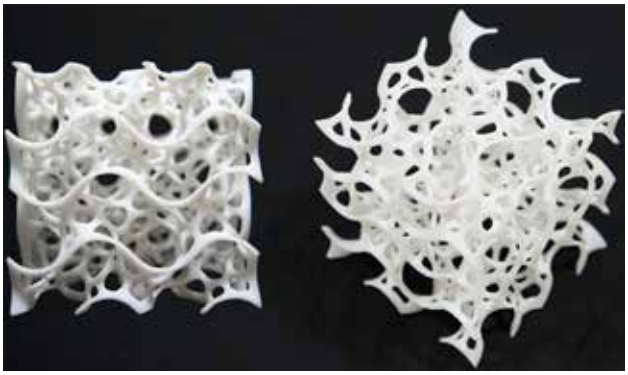


(a) (b) (c) (d) (e) (f)

Figure 6: *The bi-scale porous structures on regular TPMS: (a) P surface, (b) G surface, (c) D surface, and porous structures on distorted TPMS: (d-f).*



(a) *Bi-scale porous structures on other minimal surfaces. Costa surface (Left); Enneper surface (Right).*



(b) *The bi-scale porous structure fabricated by an SLA-3D printer, photographed from two different views.*

Figure 7: *More results.*

are distributed based on centroidal Voronoi tessellation and controlled by B-Spline curve in each cell. To our knowledge, this is the first attempt to combine these techniques for generating porous structures.

Acknowledgements

This work is supported by grants from NSFC (61572291) and Young Scholars Program of Shandong University (YSPSDU).

References

[1] A. Schoen, *Infinite Periodic Minimal Surfaces Without Self-intersections*, ser. NASA TN D. National Aeronautics and Space Administration, 1970. [Online]. Available: https://books.google.co.il/books?id=LxOb_NlMLtMC

[2] D.-M. Yan, B. Lévy, Y. Liu, F. Sun, and W. Wang, "Isotropic remeshing with fast and exact computation of restricted voronoi diagram," *Computer Graphics Forum*, vol. 28, no. 5, pp. 1445–1454, jul 2009.

[3] S. P. Lloyd, "Least squares quantization in PCM." *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–136, 1982.

[4] C. Rao, L. Tian, D.-M. Yan, S. Liao, O. Deussen, and L. Lu, "Consistently fitting orthopedic casts," *Computer Aided Geometric Design*, vol. 71, pp. 130–141, may 2019.

Space Filling Delaunay Loft Sculptures

SAI GANESH SUBRAMANIAN, MATTHEW ENG,
VINAYAK KRISHNAMURTHY AND ERGUN AKLEMAN
Texas A&M University

Abstract

This is a poster for a sculptural exhibition of a class of our new space filling modular shapes. This class of shapes is inspired by "scutoids" — shapes that were recently reported to occur in epithelial cells due to topological changes between the extremal (apical and basal) surfaces of epithelia. Inspired from this discovery, we develop a generalized procedure for generating space filling shapes, which we call *Delaunay Lofts* — a new class of scutoid-like shapes. Delaunay Lofts are produced as an interpolation of a stack of tilings that are defined by Delaunay diagrams. Let a stack of planar surfaces with Delaunay diagrams be given, Delaunay Lofts are the shapes that result from Voronoi tessellation of all intermediate surfaces along the curves joining the vertices of Delaunay diagrams that define the tessellations. Combined with the use of wallpaper symmetries, this process allows for an intuitive design of complex space filling shapes in 3-space.

(a) *One tile.* (b) *Two tiles.* (c) *Three tiles.* (d) *Four tiles.* (e) *Seven tiles.*

Figure 1: An example of a single Delaunay Loft tile that can fill both 2.5D and 3D space. This tile is created as an interpolation of two layers of tilings, namely (1) a square tiling; and; (2) another square tiling, which is a translation of the first square tiling. The interpolating control curves are straight lines.

1 Introduction

A space filling shape is a cellular structure whose replicas together can fill all of space watertight, i.e. without having any voids between them [11], or equivalently, it is a cellular structure that can be used to generate a tessellation of space [8]. 2D tessellations and 2D space filling shapes are relatively well-understood. However, problems related to 3D tessellations and space filling shapes are still interesting and have applications in a wide range of areas from chemistry and biology to engineering and architecture [11].

A well-known anecdote demonstrates the difficulty of 3D tessellations is that Aristotle claimed that the tetrahedron can fill space and many people tried to prove his claim [13] despite the fact that the cube is the only space filling Platonic solid [5]. Goldberg exhaustively cataloged many of known space-filling polyhedra with a series of papers from 1972 to 1982 such as [6]. There are only eight space-filling convex polyhedra and only five of them have regular faces, namely the triangular prism, hexagonal prism, cube, truncated octahedron [16, 15], and Johnson solid gyrobifastigium [10, 1]. It is also interesting that five of these eight space filling shapes are "primary" parallelohedra [3], namely cube, hexagonal prism, rhombic dodecahedron, elongated dodecahedron, and truncated octahedron.

We have recently developed an approach to construct and eventually design a new class of tilings in 3-space. Our approach is based on interpolation a stack of planar tiles whose dual tilings are Delaunay diagrams. We construct control curves that interpolate one Delaunay vertex of each planar tile. Voronoi

decomposition of the volume using these control curves as Voronoi sites gives us lofted interpolation of original faces. This, combined with the use of wallpaper symmetries allows for the design of the new class of space filling shapes in 3-space. In the poster exhibition, we will demonstrate 3D printed examples of this new class of shapes (See Figures 1, 2 and, 4).

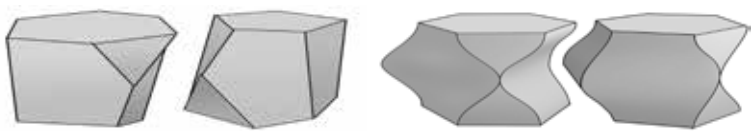
(a) *A Delaunay Loft.* (b) *Two tiles.* (c) *Three tiles.* (d) *Ten tiles.* (e) *Ten tiles.*

Figure 2: Another example of single space filling Delaunay Loft tile. This tile is created as an interpolation of three layers of tilings, namely (1) a regular hexagonal; (2) a square and; (3) another regular hexagonal tilings, which is a translation of the first hexagonal tiling. It is interesting to note that the interpolating control curves are straight lines. Regular rectangular tiling in the middle is just an automatically produced byproduct.

2 Related Work

Our approach, which can be considered as a generalization of parallelohedra, is inspired by a recent discovery by Gómez-Gálvez et al. who observed a simple polyhedral form, which they call *"scutoids"*, commonly exist in epithelial cells in the formation of thin skin layers [7]. They demonstrated that having this polyhedral form in addition to prisms provides a natural solution to three-dimensional packing of epithelial cells. In skin cells, the top (apical) and bottom (basal) surfaces of the cellular structure are Voronoi patterns (as these occur frequently due to physical constraints) [9]. Gómez-Gálvez et al. observed that the fundamental problem of packing occurs when the polygonal shapes at apical and basal surfaces do not match (e.g. pentagonal top and hexagonal bottom) leading to topological shift and resulting in scutoids.

The literature on this discovery shows the occurrence of scutoids and provides some statistical information of when and how they form [7, 2] (See 3). The reason why these shapes occur in nature is that they are the sole enablers for a space-filling packing on the skin cells. The scutoidal shapes can be considered as interpolation of 2D tiling patterns, that usually consists of hexagons and pentagons that appear on many natural structures. Interpolation is obtained by edge-collapse and vertex-split operations.



(a) *Two views of a scutoid.* (b) *Two views of a Delaunay loft.*

Figure 3: The comparison of scutoids with our Delaunay Lofts. The original scutoids usually depicted using straight edges as shown in this visual representation. Delaunay Lofts, on the other hand, (1) have curved edges and (2) can fill space.

The Figure 3a demonstrates a usual depiction of the originally discovered scutoid structures obtained

by edge-collapse or vertex-split operations between pentagons and hexagonal faces. This view results in non-planar pentagons or hexagons with straight boundaries as shown in the Figure 3a, but it does not provide any well-defined process to fill inside of these non-planar faces. Our approach to obtaining scutoid-like structures is to use 3D Voronoi decomposition using a set of curves as Voronoi sites. If this set of curves are closed under symmetry operations, the resulted Voronoi shapes are guaranteed to be space filling. It is interesting to note that this approach is also in sync with Delaunay's original intention for the use of Delaunay diagrams. Delaunay was, in fact, the first to use symmetry operations on points (instead of curves) and Voronoi diagrams to produce space filling polyhedra, which he called Stereohedra [4, 12]. Our approach can be viewed as an extension of his idea to curves. We, therefore, called our approach Delaunay Lofting.

3 Methodology and Implementation

When using points, construction of 3D Voronoi decomposition is relatively simple since distances to points guarantee to produce planar faces. On the other hands, when we use curves or even straight lines Voronoi decomposition can produce curved faces, which, in fact, makes this method interesting. However, having curved faces significantly complicates the algorithms to construct 3D Voronoi decomposition in high resolution. We, therefore, choose to deal with a subset of this general problem.

We decompose thin rectangular structures that consist of a discrete set of z-constant planar layers. We also choose the control curves in the form of $(x_i = f_{i,x}(z), y_i = f_{i,y}(z))$, where $i = 0, 1, \ldots, n$. This constraint guarantees that each curve intersects with each layer only once. We also use a specific distance function to further simplify the process into a set of 2D Voronoi decomposition. Based on these simplifications, the general process that consists of the following steps: (1) Discretize the rectangular prism with N number of constant z planes, which we call layers. (2) Design M number of curves inside of the rectangular domain. (3) Find the intersection of curves with intermediate layers. (4) For each layer, compute its Voronoi partitioning by using intersection points with that particular layer as Voronoi sites[1]. (5) Offset each Voronoi polygon the same amount using Minkowski difference [2]. (6) Treat each vertex as a single manifold and insert edges between consecutive vertices [3]. (7) Insert edges between closest vertices in consecutive layers based on face normal.

This process automatically creates the Delaunay Lofts and the resulting structures resemble scutoids with curved edges and faces. To produce space filling tiles, control curves must be closed under symmetry operations and each rectangle must be a regular domain, which topologically forms a 2-toroid. To easily produce control curves that are closed under symmetry operations, we interpolate 2D Delaunay diagrams with wallpaper symmetries. If the top and bottom tilings are rigid transformations of each other, this process produces 3D space filling shapes. In the 3D printed examples shown in Figures 1, 2 and, 4, control curves are just straight lines that interpolates top and bottom Delaunay vertices. We also have examples that interpolate more than two tiles, one such example is shown in Figure 3b, but we have not printed those yet.

References

[1] Santiago Alvarez. The gyrobifastigium, not an uncommon shape in chemistry. *Coordination Chemistry Reviews*, 350:3–13, 2017.

[2] Guy Blanchard. A 3d cell shape that enables tube formation, 2018.

[1] Since space is bounded, the boundaries of the prism become part of Voronoi polygons. For regular domain, we compute Voronoi decomposition in a 2-toroidal domain.

[2] Note that this offsetting process can also change the topology of the polygons.

[3] This process turns each original face into a 2-sided face [14], which is actually a 2-manifold.

(a) *Two views of a two 3-Space Filling 2.5 Tiles.* (b) *Three tiles.* (c) *Three tiles.* (d) *Four tiles.* (e) *Four tiles separated.*

Figure 4: Another example of single space filling Delaunay Loft tile This tile is created as an interpolation of three layers of tilings, namely (1) a semi-regular pentagonal; (2) a regular rectangular and; (3) another semi-regular pentagonal tiling, which is rotated version of the first pentagonal tiling. In this case, the interpolating control curves are also straight lines. Regular rectangular tiling in the middle is again an automatically produced byproduct.

[3] Harold Scott Macdonald Coxeter. *Regular polytopes*. Courier Corporation, 1973.

[4] Boris Nikolaevich Delaunay and Nina Nikolaevna Sandakova. Theory of stereohedra. *Trudy Matematicheskogo Instituta imeni VA Steklova*, 64:28–51, 1961.

[5] Martin Gardner. *Sixth book of mathematical games from Scientific American*. WH Freeman San Francisco, 1971.

[6] Michael Goldberg. On the space-filling enneahedra. *Geometriae Dedicata*, 12(3):297–306, 1982.

[7] Pedro Gómez-Gálvez, Pablo Vicente-Munuera, Antonio Tagua, Cristina Forja, Ana M Castro, Marta Letrán, Andrea Valencia-Expósito, Clara Grima, Marina Bermúdez-Gallardo, Óscar Serrano-Pérez-Higueras, et al. Scutoids are a geometrical solution to three-dimensional packing of epithelia. *Nature communications*, 9(1):2960, 2018.

[8] Branko Grünbaum and Geoffrey C Shephard. Tilings with congruent tiles. *Bulletin of the American Mathematical Society*, 3(3):951–973, 1980.

[9] Hisao Honda. Description of cellular patterns by dirichlet domains: The two-dimensional case. *Journal of Theoretical Biology*, 72(3):523 – 543, 1978.

[10] Norman W Johnson. Convex polyhedra with regular faces. *Canadian Journal of Mathematics*, 18:169–200, 1966.

[11] Arthur L Loeb. Space-filling polyhedra. In *Space Structures*, pages 127–132. Springer, 1991.

[12] Moritz W Schmitt. *On Space Groups and Dirichlet–Voronoi Stereohedra*. PhD thesis, Berlin: Freien Universitt Berlin, 2016.

[13] Marjorie Senechal. Which tetrahedra fill space? *Mathematics Magazine*, 54(5):227–243, 1981.

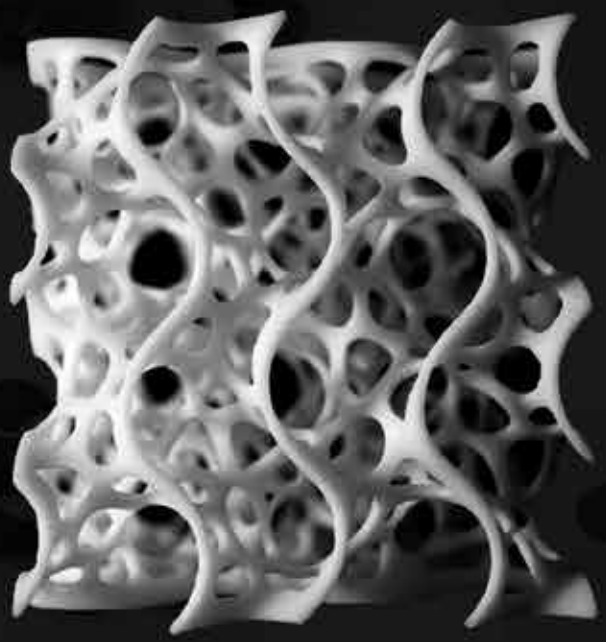
[14] Vinod Srinivasan, Ergun Akleman, and Jianer Chen. Interactive construction of multi-segment curved handles. In *Proceedings of 10th Pacific Conference on Computer Graphics and Applications, 2002.*, pages 429–430, New Jersey, 2002. IEEE.

[15] Robert Williams. *The geometrical foundation of natural structure: A source book of design*. Dover New York, 1979.

[16] Robert Edward Williams. Space-filling polyhedron: its relation to aggregates of soap bubbles, plant cells, and metal crystallites. *Science*, 161(3838):276–277, 1968.

The Proceedings of the SMI 2019 Fabrication & Sculpting Event (FASE)

In Cooperation with ISAMA



FASE Paper Chairs:

Negar Kalantar
Vinayak Raman Krishnamurthy
Konrad Polthier

Conference Chairs:
Ergun Akleman
Karina Rodriguez Echavarria

Cover Sculptures:

Front Page:

Carlo H. Séquin and Toby Chen

Back Page:

Cong Rao, Fan Xu, Lihao Tian and Lin Lu

Background:

Bih-Yaw Jin and Chia-Chin Tsoo

Cover & Proceedings Design:
Ergun Akleman

ISBN 978-0-359-46970-3
90000
9 780359 469703