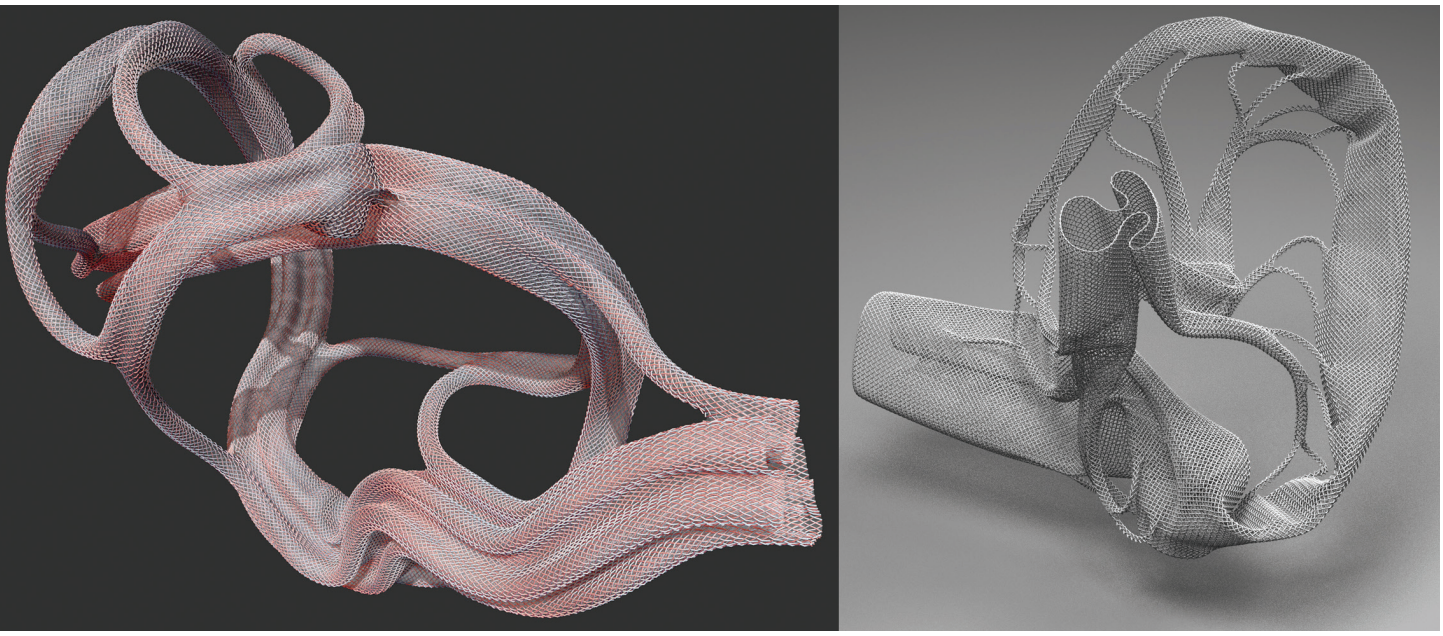
High Resolution Representation and Simulation of Braiding Patterns

Mateusz Zwierzycki
CITA/KADK

Petras Vestartas
CITA/KADK

Mary Katherine Heinrich
CITA/KADK

Phil Ayres
CITA/KADK



1

ABSTRACT

From the hand-crafted to the highly engineered, braided structures have demonstrated broad versatility across scales, materials, and performance types, leading to their use in a plethora of application domains. Despite this prevalence, braided structures have seen little exploration within a contemporary architectural context.

Within the *flora robotica* project, complex braided structures are a core element of the architectural vision, driving a need for generalized braid design modeling tools that can support fabrication. Due to limited availability of existing suitable tools, this interest motivates the development of a digital toolset for design exploration. In this paper, we present our underlying methods of braid topology representation and physics-based simulation for hollow tubular braids. We contextualize our approach in the literature where existing methods for this class of problem are not directly suited to our application, but offer important foundations. Generally, the tile generation method we employ is an already known approach, but we meaningfully extend it to increase the flexibility and scope of topologies able to be modeled. Our methods support design workflows with both predetermined target geometries and generative, adaptive inputs. This provides a high degree of design agency by supporting real-time exploration and modification of topologies. We address some common physical simulation problems, mainly the overshooting problem and collision detection optimization, for which we develop dynamic simulation constraints. This enables unrolling into realistically straight strips, our key fabrication-oriented contribution. We conclude by outlining further work, specifically the design and realization of physical braids, fabricated robotically or by hand.

1 Examples of simulated complex braid morphologies.

INTRODUCTION

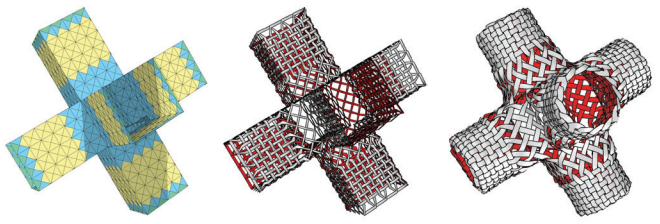
This paper presents a system for the representation and simulation of braid topologies. Here, we briefly describe the motivation for modeling braid in a way that is useful for design and fabrication workflows, and summarize our methods developed to address that motivation.

The work has been developed in the context of the *flora robotica* project (http://www.florarobotica.eu; Hamann et al. 2015), which focuses on symbiotic relationships between biological plants and robotic elements in an architectural design context (see Figure 2). The architectural structures created within the project incorporate both living and nonliving material in structural roles, with their organization governed by both robotic controllers and user interaction. This application requires that mechanical scaffolds are not predesigned and prefabricated, but rather self-organize continuously in situ through a distributed construction process. The scaffolds may be collaboratively fabricated by stationary, centrally-controlled braid machines such as industrial braid machines (Smith 1989; Haehnel and Li 1998; Richardson 1993), swarms of distributed mobile braiding robots (Heinrich et al. 2016), or manual braiding by human users. Braid is used as an overall logic for the mechanical system in *flora robotica* due to its properties of robustness, scaleless organization, and flexibility in material and pattern. It is applicable from building-sized structural scaffolds to soft-body robot arms, and can seamlessly incorporate mechanical filaments, living plants, and embedded electronics. In industrial manufacturing contexts (where braid's applications are as varied as rope, bicycle frames, and medical devices), braid is defined as an interlacing pattern of three or more continuous filaments, with all filaments performing functionally equivalent roles and spanning the length of the braid (Wulfhorst et al. 2006; Mitchell 1967); this is differentiated from manufactured weave, which contains distinct functional groups of warp and weft (Kawabata et al. 1973). Using the basic logic of braid, an expansive variety of shapes can be fabricated by changing only the pattern of interlacing filaments (see Figure 3). The resulting structures can include such features as branching, holes, caps, sockets, and flexible joints (flexure bearings).

The focus of this paper is the fabrication-oriented modeling and simulation of hollow tubular braids (see Figure 4), incorporating the above features and properties. We focus on this type because we consider hollow tubular braids to offer structural potential at an architectural scale while maintaining flexibility of topology (see Figure 3). First, we discuss the state-of the-art of relevant representation methods (subdivided into pattern generation and geometrical representation) and simulation methods. Second, we present our braid representation method, subdivided into two workflows. The pattern generation workflow

2

3



4

2 Hand-produced braids acting as scaffolds for plants and robotic elements in the context of an early exploration for the *flora robotica* project

3 Examples of hand-produced complex braid morphologies.

4 A three-step method presented in the paper. From left: pattern generation workflow, geometry represenation workflow, simulation.

uses a designed tile dictionary to assign tiles according to edge colorings, while the geometry representation workflow interprets the generated patterns into a mesh prepared for physical simulation. Third, we present our simulation method, focusing on our dynamic constraints developed to address overshooting and hash-based line-line collision detection. Fourth, we present the results of the representation and simulation methods, assessed according to performance evaluation metrics, including side-by-side comparisons between our modeling results and the manufactured prototypes that we targeted for representation. Finally, we discuss possible directions for future work, including automated fabrication, automated generation of tile order, dynamic remeshing during simulation, and tying the simulation results to structural analysis. These realms of future work are also relevant to our overall motivation for braid modeling, described above.

The primary contribution of the representation and simulation methods we present here is to provide a highly flexible braid modeling toolset that can support a variety of design and fabrication workflows.

STATE-OF-THE-ART

The representation and simulation scopes of this paper cover problems extensively raised in the literature, through modeling of weave, braid, knit and related patterns. Therefore, the state-of-the-art is presented in three parts: pattern generation, geometrical representation of such patterns, and simulation of textiles and other woven or braided materials.

Pattern Generation

The weaving pattern generation algorithm originated in Mercat (2001) and utilized in Kaplan and Cohen (2003) is defined as follows (based on pseudocode presented by Kaplan and Cohen): 1) begin with a planar graph; 2) define the midpoint of each graph edge as a 'crossing'; 3) connect the 'crossings' and define them as 'threads;' 4) expand 'threads;' and 5) offset heights of the 'threads' according to the order in which they overlap. This algorithm, through applications presented in the relevant papers, is able to generate a variety of patterns. Its ability to generate complex geometry such as Celtic knots has been demonstrated. In Akleman et al. (2009), the authors introduce a combinatorial structure called graph rotation systems as a formalization of Mercat's method described above. They analyze the regular, semi-regular and irregular tilings produced with that algorithm. The works described above are further cited in Mallos (2009), which presents a triaxial weaving method for arbitrary orientable meshes using a single tile. The method therefore simplifies the weaving pattern generation problem into a problem of finding a universal tile design. An approach presented in Yuksel et al. (2012) utilizes a predefined set of tiles with two types of edges/directions distinguished (course and wale edge). Those types are introduced so that the resulting knitting pattern follows relevant fabrication constraints. The presented design interface is composed of two stages: low-poly direction assignment (labeling) and yarn-level pattern definition (stitch pattern definition), either with manually introduced modifications or predefined patches.

In summary, the literature presents two general approaches for woven or braided pattern generation, used in a multitude of ways. The two approaches are procedural generation of patterns, as in Mercat (2001) or Kaplan and Cohen (2003) and pattern production with a predefined set of tiles applied over discrete two- or three-dimensional geometry, as in Yuksel et al. (2012) or Mallos (2009).

Geometry Representation

Pattern generation can itself be the end objective, but in some cases the result is then used for physical simulation of textiles or fabrics. In other words, the generated pattern can feed a geometrical model for simulation of a physical object. The end objective of the method (either geometry per se or geometry for simulation) defines the characteristics of that representation and the properties that it should necessarily possess. If the aim is geometry directly, then a necessary property is aesthetic appeal, while if the aim is geometry for simulation, its properties must meet the criteria of the relevant simulation engine. In the system described in Kaplan and Cohen (2003), a two-dimensional Cubic Hermite spline is defined using control points constructed with Mercat's method. After establishing the relationships between curves (the under-over pattern and the terminus points), the curves are then inflated to give the braids the desired thickness. If there are any images introduced in the knot (terminus points), the method has to order the under-over pattern afterwards, which introduces a backtracking procedure into the geometry creation method. A visually appealing three-dimensional geometry is the aim of the Akleman et al. (2009) method. The first step in constructing the geometry is to create a low-poly control mesh/polyline which is then used to construct splines or ribbons. The ribbon/yarn collision avoidance problem is tackled with a set of simple rules, minimizing the chances of collisions to a negligible level. Quadratic B-splines and cubic Bezier surfaces are used for the final stage of geometry creation, resulting in either a better collision avoidance or a more appealing look. The method presented in Yuksel et al. (2012) models fabric with two representations. A low-resolution mesh is used to define directions, and is then utilized by a high-resolution generation routine. The generated high-resolution stitch-mesh is used to define a cubic Catmull-Rom spline representation of the yarns, which becomes the basis for yarn-level relaxation.

Simulation

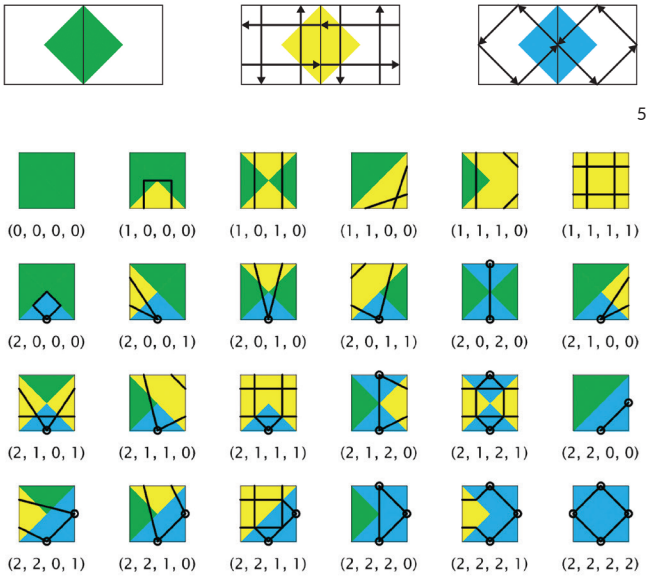
A highly precise yarn-level simulation method is presented in Yuksel et al. (2012). During each timestep, the algorithm looks for intersections of spheres placed along the yarns. If the intersection is found, the same check is performed with smaller spheres either until the intersection is resolved or until the lower limit of the radius is reached. Because of the iterative process for calculating varying radii sphere-sphere collisions, it is relatively slow. The physics engine presented in Jakobsen (2001) is known for its stability and speed of execution. While accuracy is not its main objective, it is visually believable and therefore useful in computer graphics. Its success is built upon the methods incorporated within it, including a Verlet integration scheme, a simple constraint solver using relaxation, modeling of rigid bodies as particles with constraints, and handling of collisions and penetrations using projections. The 'ShapeOp' (see http://shapeop.org) geometry processing and optimization framework has its foundations in projection-based approaches (Bouaziz et al. 2012; Bouaziz et al. 2014). The constraints in this framework define relationships between subsets of points, which are then satisfied through projections. Due to an absence of momentum induced oscillations, the convergence in this method occurs considerably faster. The framework is characterized by its robustness, speed, and generalized definition of constraint. Spatial Hashing applied for collision detection is described in Teschner et al. (2003). The algorithm divides three-dimensional space into fixed-size boxes (cells) and assigns the vertices to them. Checking for intersections is therefore performed on a hash table, with quick search times and the possibility to check for both self-collisions and collisions between distinct objects.

REPRESENTATION METHOD

Our method for braid representation is subdivided into two workflows: braiding pattern generation and geometrical representation of those patterns.

Pattern Generation Workflow

Our pattern generation method takes inspiration from many of the methods in the existing literature. The primary advantages achieved by the presented method are simplicity and generalizability. These provide an increased flexibility in the topologies that can be represented. The implementation is tailored to work with orientable mesh surfaces, therefore the descriptions will be following the particular logic of the braiding patterns. Though the scope of this paper focuses on braiding patterns in 2D (i.e. hollow tubular braids), the approach works conceptually for other problem classes, such as 3D patterns and reciprocal structures. The complexity of braiding patterns presented here (such as those in Figure 3) cannot be simply classified as plain weave, braid or triaxial braiding. Our presented solution is based on



5 Interfaces used in the implementation (from left): Empty, Plain and Braid

6 An example of a complete set of tiles for 3 colors and 4-sided polygons. 24 tiles guarantee all the combinations of colors, given the possibility to rotate them.

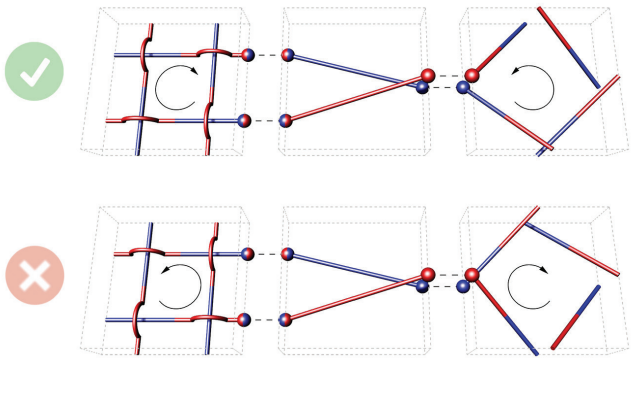
Mercat's method, but it is extended in a way that supports many additional types of braiding and weaving patterns, as well as the mixing of these two pattern categories. The advantage of the presented approach is the implicit nature of under-over relationships, which result from the set of tile generation rules described in subsection *Tile Design*. In general, this method uses a precomputed set of tiles, which, in their underlying logic, combine different approaches seen in the literature. The method works directly on a mesh, and thus whenever used, the terms faces and edges refer to the mesh geometry rather than to a graph.

Edge Coloring

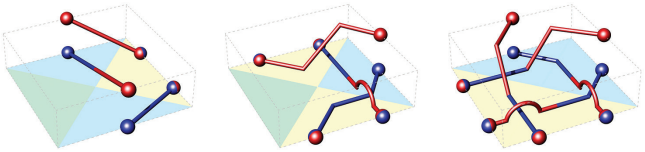
The first step in the pattern generation workflow is to define how many types of interfaces are possible between the tiles (i.e., determining the possible transition cases). Figure 5 shows the three cases in our implementation, which are: no connection between the faces (Empty), connection with two strips separated (Plain), and connection with two strips passing through the middle point (Braid, as in Mercat's method). The interfaces are hereafter interchangeably referred to as colors.

Tile Dictionary

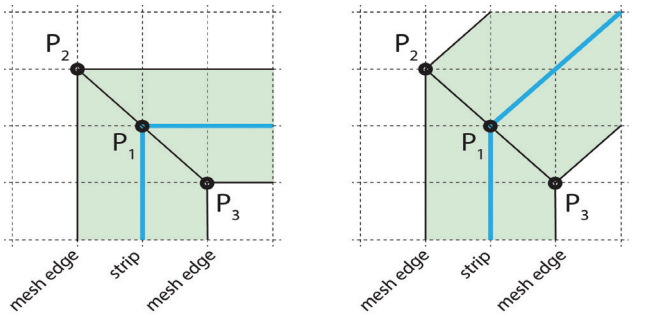
Once the number of colors is defined, the second step in the pattern generation workflow is the utilization of the tile dictionary. It is necessary for this purpose to define the number of polygon types. (Though the dictionary and implementation encompass both triangles and quads, for clarity, the description will refer only to quads.) Given the number of polygon sides to



7



8



9

7 A usage example of the implemented interface schemes. On the left side is the Plain interface, while Braid is shown to the right. The Plain interface is broken by mirroring the leftmost tile (bottom image), making the point coloring logic inconsistent - matched connection points have opposite colors. Mirroring the rightmost tile (bottom image) makes the Braid interface meaningless, as the marking of the strips doesn't follow their order anymore.

8 A set of examples showing some correct tile designs.

9 From left: 90-degree right turn and 45-degree left turn.

be colored and the number of possible colors, a finite set of color combinations becomes evident. The precise number of combinations can be calculated using Polya's Counting Theorem (Polya and Read 1987). For three colors and four-sided polygons, the number of possible color combinations after reducing the rotational symmetries is equal to 24. The implementation uses tuples to identify a particular color combination. By sorting the tuples, the user gets a convenient overview of the complete set, useful for editing the predefined tiles. An example tile set can be seen in Figure 6. This dictionary is therefore a Wang tile set (Wang 1961), with all the possible combinations of colors. However, from the point of view of the user, it is more intuitive to consider the colors of individual edges than particular tiles. It is important to note that contrary to Wang tiles, the ones presented here can be rotated, such that the number of combinations required for a complete set is reduced by rotational symmetry.

Tile Design

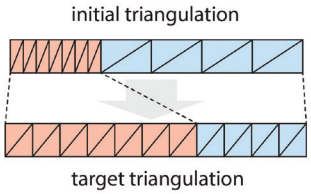
While the edge colors describe the overall pattern, the tiles have to be designed with the interfaces in mind, so that the resulting pattern has a desired under-over order (in this implementation it's a plain weaving order). There is a potentially infinite amount of weaving patterns with a particular color combination, so the rules specified here have to apply only to the strips which are taking part in the tile-to-tile interface (contrary to cycles contained wholly within a tile). As mentioned in the *Edge Coloring* subsection, there are three interface types in the presented implementation, from which two have to be resolved - the Braid and the Weave interface (Empty transition is skipped because it is an obvious case). The blue color hereafter represents the under state and the over strips are marked with red. In the case of the Braid interface the coloring is straightforward—the under point is marked with blue color as seen in Figure 7 and the over point is red. It is important to design tiles in a way following the colors and their meaning. Not crossing over with a red strip means that the blue part will not consequently go under (Fig.7). The meeting points have to be modified in case of the Plain interface. This time each point has 2 colors indicating the order of the strip, which means that the strip changes its order after passing through it. Note the left and right points are marked differently, therefore mirroring the tile will produce a wrong interface as seen in Figure 7. Once all the tiles are solved, it is possible to tile any orientable surface with any combination of colors. A small sample of colorings and resulting patterns is shown in Figure 16.

Geometry Representation Workflow

The second part of our braid representation method—geometry representation—interprets the patterns generated from our edge colorings and tile dictionary. Through construction and discretization, the patterns are represented geometrically such that they can be consistently used as inputs to physical simulation.

Construction and Discretization

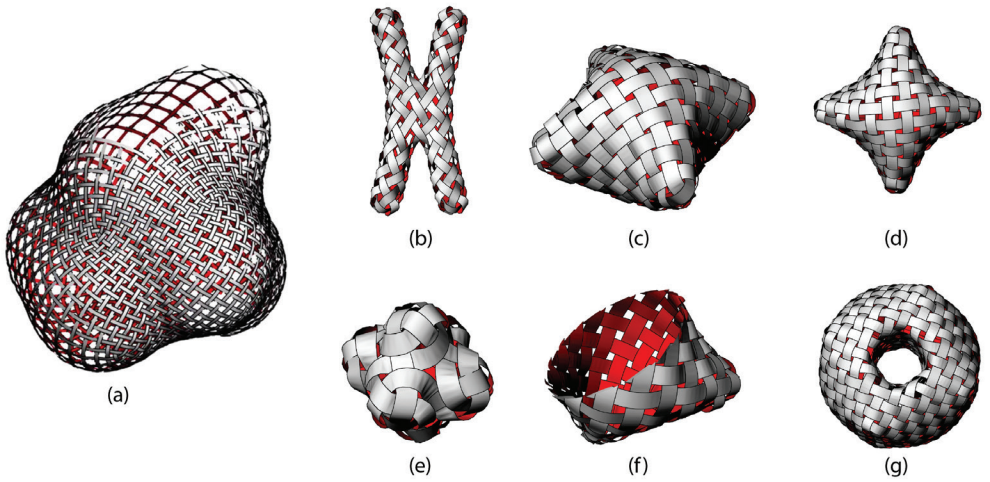
While it is possible to define a continuous parameter space for geometry construction as shown in Akleman et al. (2009), this implementation uses a point grid to ease the tile generation process. This makes the task of strip designing less error-prone. Each strip is declared as a series of grid-based coordinates. The grid with the underlying strips is mapped onto a B-spline surface constructed from the 4 corner points of a particular mesh quad. There is a small chance of self-intersection, particularly when the target face is nonplanar. The under-over order is expressed by



10

10 The geometry solver has an objective to equalize all the naked edges to the same length. The discretization of the initial geometry is one of the input parameters for the solver and determines the final length of each of the strip segments (red and blue).

11 A collection of exemplary geometries generated with the presented method.



11

offsetting strip points by an interpolation of faces' vertex normals. The grid points are also used in the process of mesh drawing. As shown in Figure 9, the point P1 is the point from the predefined strip. The angle of the strip turn determines the choice of the points P2 and the P3. The resulting variance of the mesh width is later unified by the geometry solver. The last step is to divide each of the strip segments into triangles, having in mind the final length is dependent on the number of divisions as seen in Figure 10. While it is not the most intuitive way to set the final length for each strip, it does provide a uniform and simple triangulation method.

SIMULATION METHOD

Once a braid pattern has been generated and has been represented geometrically to prepare it for simulation, our simulation method relaxes that pattern into a tightened braid that is both visually realistic and able to be fabricated.

Overview of Geometry Solving and Simulation

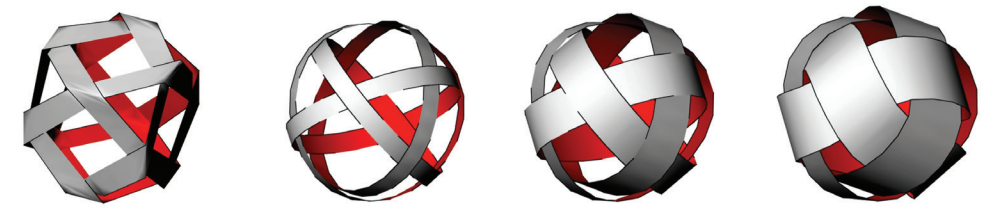
In order to evaluate physical properties of the digital model such as yarn-yarn interaction and self-weight, stripes cannot be simplified to a gridshell-like model, because of lack of the torsion induced by the flat strip geometry. As a result, mesh topology is constructed from triangle meshes with varying density to properly model the physical properties. The constraint-based geometry solver Kangaroo2 (see http://kangaroo3d.com) plays a critical role for the simulation and fabrication purposes. It both enables prediction of the anticipated physical appearance, and also forces the generated mesh strips into straight shape using custom-coded constraints. By unrolling the mesh strip geometry it is possible to evaluate how close the digital geometry is to the physical prototypes - the straighter the unrolled strip, the closer it is to the straight band used in the fabrication process. Collision detection is a great impediment in the case of two-dimensional manifolds. Because of the discrete simulation timestep and geometry with no thickness, rapid checks of the side of collision can be rendered impossible (depending on the approach). The underconstrained nature of fabric simulations makes this problem even harder, as a variety of types of collisions occur in such simulations such as self-collisions, low and high speed collisions etc. (Bridson et al. 2002).

Hash-based Line-Line Collision Detection

The spatial grid method (Teschner et al. 2003) is chosen for collision detection as it is appropriate for range searches in the dynamic environment of the simulation and it is not necessary to rebuild the spatial grid for each frame. The initial mesh model as established in the subsection "Construction and Discretization" defines the relationships between the particles and utilizes the naked edges of the mesh for collisions. The middle points of those edges are stored in the hash table and used to efficiently find nearby points, thereby greatly reducing the necessity to perform time-consuming line-line collisions. The 3D bucket size is set to half of the target length of the naked edge. While in the initial geometry there are edges of various lengths, the solver aims to equalize them to the target size and therefore the collision detection gets more robust over the time of the simulation (Note, the collisions are getting more frequent over time as well).

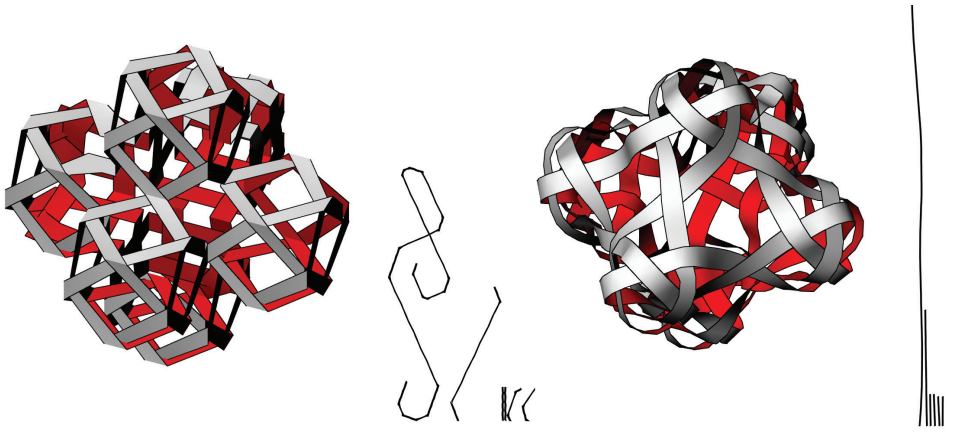
Overshooting

Overshooting may be caused when the triangles of the mesh are undesirably stretched or compressed by large percentages.



12 Overshooting prevention. From left: Initial mesh, first step in relaxation and incremental strip width increase.

13 Side-by-side comparisons of the fabricated T-junction and the simulation of the same topology (left), and a fabricated 'foot' detail, where a column made with Braid pattern changes into Plain pattern.



14 Unrolled strips before and after geometry solving.

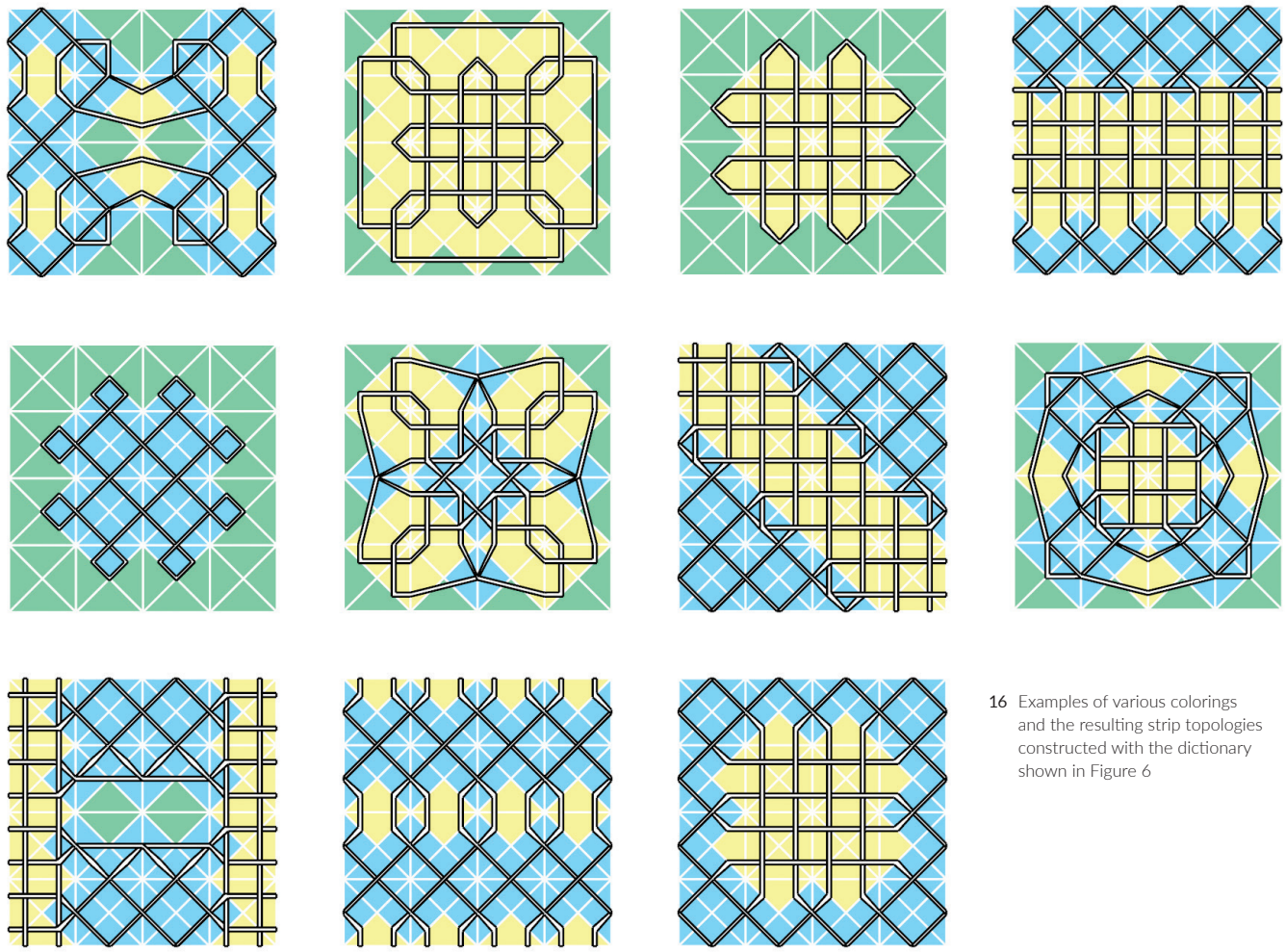
15 The low-resolution of the mesh strip is what prevents it to form in the expected shape. With a proper remeshing technique the tight corner would likely be fully expressed.

A rule of thumb in computational mechanics is that to prevent overshooting, a triangle edge should not change length by more than 10% in a single time step (Caramana et al. 1998). This can be enforced by either adaptively decreasing the time step or decreasing the strain rate (the latter is used in this implementation). In usual cases, the projection solver would change the length of each edge to the target value as quickly as possible resulting in a loss of the under-over order within a few iterations. A dynamic constraint is introduced as a solution for this overshooting problem. It starts with the initial edge lengths and incrementally adds up to reach the target. This results in longer calculation time but helps to achieve extremely tight braids with strips composed of well shaped isosceles triangles (see Figures 11 and 12).

RESULTS

The geometry obtained with the presented method closely approximates the manufactured prototypes, as seen in the comparison of isolated braid conditions of Figure 13. As the problem of cloth and fabric simulation is by definition an under-constrained problem, it is unreasonable to expect exactly the same results. For the purpose of design exploration and macro-scale behavior prediction, the presented approach seems to be accurate enough. The side-by-side comparisons reveal the same kinds of artifacts emerging in the real-world pieces and their virtual models. Furthermore, the strips remain straight after unrolling thereby satisfying a fabrication constraint (see Figure 14).

The simulation method was tested on a series of different models and topologies, with varying complexity and triangle count. When it comes to performance (tested on a mid-range professional PC), thanks to the projection-based geometry solver a simulation with a triangle count between 1,000-10,000 converges after 15-120 seconds depending on the case (Figure 11 b,c,d,g). Smaller forms like example (e) with few hundred triangles, reach their final shape in a couple of seconds. The amount of time required to reach equilibrium is also dependent on the global shape, and with simple forms like example (a) (around 40,000 triangles) it took only 120 second to converge as well.



16 Examples of various colorings and the resulting strip topologies constructed with the dictionary shown in Figure 6

16

LIMITATIONS AND FUTURE WORK

The next step in the implementation of the methods shown in this paper is to fully explore the relationship between design and fabrication, including extension to automated fabrication methods. The braided physical objects shown in this paper (such as those in Figure 3) are manufactured by hand, and thus the translation of our braid topology representations into machine-readable code remains to be addressed.

Another future research direction is automated generation of tile order in complex braided structures. The rules described in the "Tile Design" subsection provide a complete framework to build upon, such that approaches with various priorities could utilize this framework to generate structures tailored to their requirements. This paper's implementation of the presented pattern generation system focuses on weaving and braiding, but a very similar logic could be used to generate many other types of patterns. While tile-based texture generation is exhaustively explored in the literature, the presented system adds another layer of logic to it, making it possible to generate more complex types of dependencies between the tiles. This can potentially lead to research on structures such as reciprocal frames and tensegrity, which stand out because of the fundamental importance of element-to-element relations.

In terms of simulation, future research directions include the use of a dynamic remeshing method for the strips, which could prevent undesired artifacts (as shown in Figure 15). Implementing a remeshing technique similar to the one presented in Narain et al. (2013) could resolve this issue and add to the overall geometry relaxation quality by making the unrolled strips even more similar to the desired straight shape (as described in the "Simulation Method" section). Currently, the line-line collisions are detected by measuring distances between their middle points, as described in the "Hash-based Line-Line Collision Detection" subsection. The authors plan to investigate rasterization to fully allocate all the buckets containing the lines.

A broader direction for future research is the integration of structural analysis. The nonlinear relaxation methods this

paper employs could be investigated for integration with Finite Element Analysis. If the focus remains on hollow tubular braids, this paper's understanding of braid as a configuration of individual strips and their collisions could be usefully coupled with a macro-scale representation relating to its shell-like behavior. This combination may allow the exploration of relationships between braid topology, braid material, and structural performance. The stiffness of material used will impact shape, deformation, and overall possibilities for architectural application.

As the implementation described here is design-oriented, it is evident that it currently lacks a suitable user interface. With the possibility to show both colorings and simulation results simultaneously, it may be challenging to make it as user-friendly as the interface in Yuksel et al. (2012). The authors have begun to consider this problem in the context of an overall design workflow (Vestartas et al. 2017).

CONCLUSION

This paper has presented a braid representation method through pattern generation and geometric representation of those patterns, as well as a simulation method for relaxing the geometry into tight and visually realistic braids. Generation of patterns is greatly simplified with the proposed approach and guarantees proper, plain order weaving/braiding patterns. This approach meaningfully extends existing tile-based approaches. It provides higher flexibility, noticeably expanding the range of braid topologies that are able to be represented. The geometrical representation of the patterns is based on mesh geometry, built upon an underlying point grid. This reduces self-intersections in the initial stage. The discretized geometry undergoes the simulation of its physical behavior. The projection-based solver uses dynamic constraints to achieve the final woven/braided shape and straighten the strips. The achieved straightness of individual unrolled strips is the key fabrication constraint that makes this simulation approach a useful contribution, compared with other publications and previous approaches.

Overall, the high flexibility of our braid representation and simulation method is this paper's primary contribution, as it enables open-ended exploration of design and fabrication workflows for tubular braid. This has two impacts. First, it addresses our motivation by laying a foundation for braided structures to be applied to the architectural domain. Second, the general modeling approach followed here could be usefully explored for flexible architectural design of other pattern-based nonlinear structures.

ACKNOWLEDGMENTS

Project *flora robotica* has received funding from the European Union's Horizon 2020 research and innovation program under the FET grant agreement, no. 640959.

The authors gratefully acknowledge the assistance of David Andres Leon and Ashkan Cheheltan in the hand-production of the braided structures.

REFERENCES

Akleman, Ergun, Jianer Chen, Qing Xing, and Jonathan L. Gross. 2009. "Cyclic Plain-Weaving on Polygonal Mesh Surfaces With Graph Rotation Systems." *ACM Transactions on Graphics* 28 (3): 78.

Akleman, Ergun, Jianer Chen, and Jonathan L. Gross. 2015. "Extended Graph Rotation Systems as a Model For Cyclic Weaving on Orientable Surfaces." *Discrete Applied Mathematics* 193 (C): 61–79.

Bailey, Christopher. 2008. "Creating Celtic Knot Work From User Parameters." Ph.D. thesis, The University of Bath.

Bouaziz, Sofien, Mario Deuss, Yuliy Schwartzburg, Thibaut Weise, and Mark Pauly. 2012. "Shape-Up: Shaping Discrete Geometry With Projections." *Computer Graphics Forum* 31 (5): 1657–1667.

Bouaziz, Sofien, Sebatian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. "Projective Dynamics: Fusing Constraint Projections for Fast Simulation." *ACM Transactions on Graphics* 33 (4): 154.

.Branscomb, David, David Beale, and Royall Broughton. 2013. "New Directions in Braiding." *Journal of Engineered Fibres and Fabrics* 8 (2): 11–24.

Bridson, Robert, Ronald Fedkiw, and John Anderson. 2002. "Robust Treatment of Collisions, Contact and Friction For Cloth Animation." *ACM Transactions on Graphics* 21 (3): 594–603.

Caramana, E. J., D. E. Burton, M. J. Shashkov, and P. P. Whalen. 1998. "The Construction of Compatible Hydrodynamics Algorithms Utilizing Conservation of Total Energy." *Journal of Computational Physics* 146 (1): 227–262.

Durupinar, Funda, and Uğur Güdükbay. 2007. "Procedural Visualization of Knitwear and Woven Cloth." *Computers and Graphics* 31 (5): 778–783.

Haehnel, Rudolf, and Xing Li. 1998. Rotary braider machine. US Patent 5,775,195, filed January 14, 1997, and issued July 7, 1998.

Hamann, H., M. Wahby, T. Schmickl, P. Zahadat, D. Hofstadler, K. Stoy, S. Risi, A. Faina, F. Veenstra, S. Kernbach, I. Kuksin, O. Kernback, P. Ayres, and P. Wojtaszek. 2015. "Flora Robotica – Mixed Societies Of Symbiotic Robot-Plant Bio-Hybrids." In *IEEE Symposium Series on Computational Intelligence*, 1102–09. Cape Town, South Africa: SSCI.

Heinrich, Mary K., Mostafa Wahby, Mohammad D. Soorati, Daniel N. Hofstadler, Payam Zahadat, Phil Ayres, Kasper Stoy, and Heiko Hamann. 2016. "Self-Organized Construction with Continuous Building Material: Higher Flexibility Based on Braided Structures." In *IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, 154–59. Augsburg, Germany: FAS*W.

Igarashi, Yuki, Takeo Igarashi, and Hiromasa Suzuki. 2008. "Knitting a 3D Model." *Computer Graphics Forum* 27 (7): 1737–43.

Jakobsen, Thomas. 2001. "Advanced Character Physics." In *Proceedings of the Game Developers Conference*, 1–16. GDC.

Jung, K., S. J. Kim, T. J., Kang, K. Chung, and J. R. Youn. 2003. "Optimum Modeling of 3-D Circular Braided Composites." In *Proceedings of the 14th International Committee on Composite Materials Conference*, 14–18. San Diego, CA: ICCM.

Kaldor, Jonathan M., Doug L. James, and Steve Marschner. 2008. "Simulating Knitted Cloth at the Yarn Level." *ACM Transactions on Graphics* 27 (3): 65.

Kaplan, Matthew, and Elaine Cohen. 2003. "Computer Generated Celtic Design." In *Proceedings of the 14th Eurographics Workshop on Rendering*, 9–19. Leuven, Belgium: EGRW.

Kawabata, S., Masako Niwa, and H. Kawai. 1973. "The Finite Deformation Theory Of Plain-Weave Fabrics Part I: The Biaxial Deformation Theory." *Journal of the Textile Institute* 64 (1): 21–46.

Mallos, James. 2009. "How to Weave a Basket of Arbitrary Shape." In *Proceedings of the 8th Interdisciplinary Conference of the International Society of the Arts, Mathematics, and Architecture*, 13–19. Albany, NY: ISAMA.

McCann, J., L. Albaugh, V. Narayanan, A. Grow, W. Matusik, J. Mankoff, and J. Hodgins. 2016. "A Compiler For 3D Machine Knitting." *ACM Transactions on Graphics* 35 (4): 49.

Mercat, C. 2001. "Les entrelacs des enluminure celtes." *Dossier Pour La Science* 15.

Mitchell, R., 1967. Braid and method of making it. US Patent 3,323,406. Filed April 7, 1964, and issued June 6, 1967.

Narain, Rahul, Tobias Pfaff, and James F. O'Brien. 2013. "Folding and Crumpling Adaptive Sheets." *ACM Transactions on Graphics* 32 (4): 51.

Pólya, G., and R. C. Read. 1987. *Combinatorial Enumeration Of Groups, Graphs, And Chemical Compounds*. New York: Springer-Verlag..

Qing, Xing, Gabriel Esquivel, Ryan Collier, Michael Tomaso, and Ergun Akleman. 2011. "Weaving Methods in Architectural Design." In *ACADIA 2011 Regional: Parametricism*, 59–66. Lincoln, NE: ACADIA.

Richardson, Donald. 1993. Maypole Braider Having a Three Under and Three Over Braiding Path. US Patent 5,257,571. Filed January 28, 1993, and issued November 2, 1993.

Smith, Michael F. 1989. Apparatus and method for automated braiding of square rope and rope product produced thereby. US Patent 4,803,909. Filed April 13, 1987, and issued February 14, 1989.

Teschner, M., B. Hiedelberger, M. Müller, D. Pomeranets, and M. Gross. 2003. "Optimized Spatial Hashing for Collision Detection of Deformable Objects." In *Proceedings of the 8th Workshop on Vision, Modeling, and Visualization*, 47–54. Munich, Germany: VMV.

Vestartas, P., M. K. Heinrich, M. Zwierzycki, and P. Ayres. 2017. "Design Tools and Workflows for Braided Structures." In *Design Modelling Symposium 2017* (accepted paper, not yet published)

Wang, Hao. 1961. "Proving Thorems By Pattern Recognition II." *Bell Systems Technical Journal* 40 1–41.

Wulfhorst, Burkhard, Oliver Maetschke, Markus Osterloh, Alexander Büsgen, and Klaus-Peter Weber. 2006. *Textile Technology*. Wiley Online Library.

Xing, Qing, Ergun Akleman, Jianer Chen, and Jonathan L. Gross. 2010. "Single-Cycle Plain-Woven Objects." In *Proceedings of the Shape Modeling International Conference*, 90–99. Aix en Provence, France: SMI.

Yuksel, Cem, Jonathan M. Kaldor, Doug L. James, and Steve Marschner. 2012. "Stitch Meshes For Modeling Knitted Clothing With Yarn Level Detail." *ACM Transactions on Graphics* 31 (4): 37.

Zhou, Kun, Xin Huang, Xi Wang, Yiying Tong, Mathieu Desbrun, Baining Guo, and Heung Yeung Shum. 2006. "Mesh Quilting For Geometric Texture Synthesis." *ACM Transactions on Graphics* 25 (3): 690–97.

IMAGE CREDITS

Figure 2: Anders Ingvartsen, 2016

All other drawings and images by the authors.

Mateusz Zwierzycki is an architect, developer and Grasshopper user. He is author or co-author of such libraries and plugins as Anemone, Volvox, Starling, and most recently Owl. He is also the founder of Object (theObject.co), a long time workshop tutor, teacher and a parametric design popularizer.

Petras Vestartas is a PhD student at IBOIS, EPFL. He has previously been a research assistant at CITA, KADK, where he was involved in several research projects such as CM5 – Inflated Restraint, led by Associate Professor Phil Ayres, and EU FET project *flora robotica*. Petras holds a master's degree in architecture from the Vilnius Academy of Arts (VAA) and worked in different international offices such as DMAA, Austria and CEBRA, Denmark.

Mary Katherine Heinrich is a PhD Fellow at CITA, KADK, funded by the EU FET project *flora robotica*.

Phil Ayres is an architect, researcher and educator. He joined CITA in 2009 after a decade of teaching and research at the Bartlett, UCL, and completing his PhD in Denmark at the Aarhus School of Architecture. Phil is the editor of the title *Persistent Modelling – extending the role of architectural representation* published by Routledge (2012), and a principle Investigator on the EU FET project *flora robotica*.