Interactive Modeling of Architectural Freeform Structures – Combining Geometry with Fabrication and Statics

Caigui Jiang,[a] **Chengcheng Tang,**[a] **Marko Tomičić,**[b]
Johannes Wallner,[c] **Helmut Pottmann**[a,d]

[a] KAUST [b] Waagner Biro Stahlbau [c] TU Graz [d] TU Wien

Abstract. *This paper builds on recent progress in computing with geometric constraints, which is particularly relevant to architectural geometry. Not only do various kinds of meshes with additional properties (like planar faces, or with equilibrium forces in their edges) become available for interactive geometric modeling, but so do other arrangements of geometric primitives, like honeycomb structures. The latter constitute an important class of geometric objects, with relations to "Lobel" meshes, and to freeform polyhedral patterns. Such patterns are particularly interesting and pose research problems which go beyond what is known for meshes, e.g. with regard to their computing, their flexibility, and the assessment of their fairness.*

1 Introduction

Architectural projects of high geometric complexity greatly benefit from the incorporation of essential aspects of function, fabrication and statics into the shape modeling process. This integrated view is one of the major goals of Architectural Geometry. One has to avoid detailed physical simulation as this would hardly be compatible with interactive shape manipulation. Instead, we aim at developing shape modeling tools which employ simplified mathematical models in order to respect manufacturing and structural constraints. Still, one needs to develop efficient numerical solvers for the arising systems of constraints. In the present paper, we briefly address a few core ideas on the mathematical formulation and mainly discuss a class of numerical algorithms which are well suited for interactive design.

From a mathematical perspective, the problems to be solved are not typical constrained optimization problems. Constraint-aware design is about the solution of under-determined systems of usually nonlinear, and frequently redundant, equations and inequalities and about ways of guiding the user towards preferred solutions. This guidance is provided by additional targets related to the aesthetics of a design and other goals which depend on the specific application scenario and task.

1.1 Prior work

The methodology presented in our paper is applicable to a wide range of problems in Architectural Geometry, but we confine our survey of prior work to the integration of structural constraints and to interactive constraint-aware design.

Statics-aware design. The incorporation of structural constraints into shape design is probably best accomplished by aiming at force equilibrium. This approach is taken within the *thrust network method* in connection with the design of self-supporting structures [Block and Ochsendorf 2007; Block 2009; Block and Lachauer 2011; Vouga et al. 2012; Panozzo et al. 2013; de Goes et al. 2013; Liu et al. 2013], as well as closely related form-finding methods for compression support structures [Lachauer and Block 2012], fabric formwork for concrete shells [Van Mele and Block 2011] and tension structures [Barnes 2009].

The design of structures which are built from simple elements (such as planar quad panels) and which are structurally sound at the same time, is much less investigated. Schiftner and Balzer [2010] do not simultaneously consider statics and the rationalized geometry. Vouga et al. [2012] show how to rationalize any self-supporting shape by a self-supporting planar quad mesh, but do not directly design it. Form-finding with polyhedral meshes in static equilibrium is the topic of [Tang et al. 2014], of which our paper is a follow-up.

Interactive design tools for freeform architecture. Most early work in architectural geometry was already aimed at fabrication-aware design and realized this goals by means of constrained optimization algorithms. The most successful ones are already incorporated in commercially available software (e.g. "*EvoluteTools*").

A lot of recent progress comes from the research group of M. Pauly at EPFL. Their *shape-up* algorithm [Bouaziz et al. 2012] uses projections onto individual constraints, but has the disadvantage that each constraint requires a separate treatment. Local shape modifications of constrained models using ideas from sparsity have been presented by Deng et al. [2013]. Finally, Deng et al. [2014] extended *shape-up* within an augmented Lagrangian formulation to a framework which aims at real-time manipulation and a careful distinction between hard constraints and soft targets. Poranne et al. [2013] introduced plane coordinates as auxiliary variables for modeling polyhedral surfaces. In this way, the planarity constraint is quadratic. This approach motivated the systematic avoidance of constraints of degree ≥ 3 by Tang et al. [2014] which forms the basis of the present paper. Extensive comparative tests showed that it clearly outperforms previous work in terms of the combination of speed and high accuracy of constraint satisfaction. Moreover, it is well suited for the integration of statics based on the thrust network method.

Remark: Tools based on evolutionary optimization such as *Galapagos* attracted a lot of attention in the architectural community. Their simplicity comes at the cost of low efficiency and accuracy and thus they are not suitable for the interactive modeling tasks we are interested in here.

1.2 Contributions and overview

In § 2, we briefly outline the surprisingly effective method of Tang et al. [2014] which is suitable for interactive design of constrained meshes and more general structures. § 3 discusses form-finding with polyhedral meshes. We show how to design meshes with planar faces subject to further constraints such as alignment with a given boundary, force equilibrium, and others. This method is very well suited for modeling structures with repetitive elements. This is demonstrated in § 4 by means of so-called honeycomb structures and *Lobel frames* which are closely related to them. In § 5 we turn to the wider topic of polyhedral patterns and discuss ways for assessing and controlling the aesthetics of such patterns. Along with a discussion of our findings and insights, we address future research directions in § 6.

2 Computational Setup.

The computational setting we are about to describe is very general, and we illustrate it by means of a few examples which also serve to introduce the geometric objects which are the focus of this paper. Their interrelations are the topic of later sections. Each time we describe a set of variables which define a certain geometric structure, provided certain constraints are fulfilled. We aim at constraints which are linear or quadratic equations, and which involve as few variables as possible.

"Lobel" meshes (cf. Figure 1), are defined by having equilateral triangles as faces. We use the vertices $\mathbf{v}_1, \mathbf{v}_2, \ldots$ as variables, together with the edge length "l". The equilateral property reads $(\mathbf{v}_i - \mathbf{v}_j)^T (\mathbf{v}_i - \mathbf{v}_j) = l^2$, for all edges $\mathbf{v}_i\mathbf{v}_j$. We conclude that the set of constraints imposed on a triangle mesh in order to enforce the Lobel property is one quadratic equation per edge.

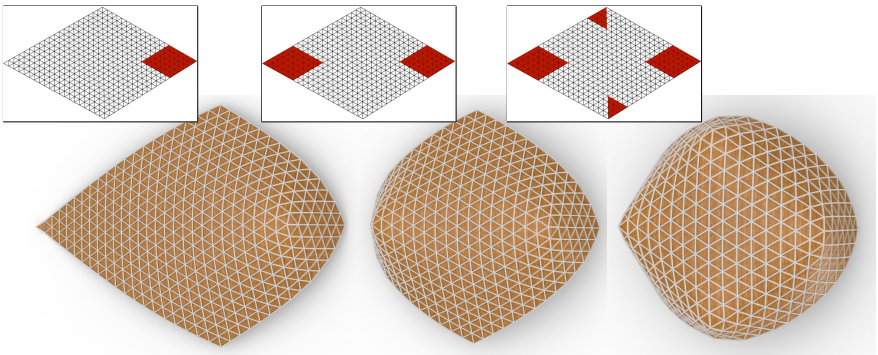


Figure 1: *Lobel Meshes*. This series of images visualizes surfaces composed of equilateral triangles created by cutting out pieces of a diamond (red areas in inset figures) and gluing the newly arising boundaries together. Finding such surfaces is one example of a mesh optimization problem with constraints (here: equal edge lengths in a triangle mesh). Structures made from equilateral triangles are called "Lobel" structures in honor of the French architect Alain Lobel who intensively studied them, see [Lobel 1993].

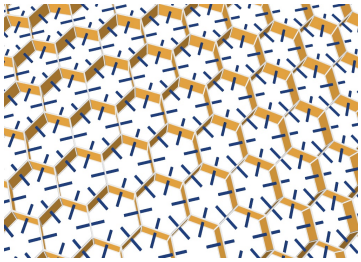


Figure 2: Designing *honeycomb structures* is an instance of modeling with constraints: A collection of quadrilaterals form the walls of a honeycomb structure, if their combinatorial arrangement is along the edges of a hexagonal-dominant mesh, and their intersection angle is 120°. Constraints are planarity of quads and the correct intersection angles. Note that one can find a Lobel mesh which intersects the honeycomb orthogonally, and vice versa.

Polyhedral meshes. A more complex example are meshes whose faces can be arbitrary n-gons, but are required to be planar. In addition to vertices, we here use the normal vectors $\mathbf{n}_1, \mathbf{n}_2, \ldots$ as variables. The condition that face No. k is planar is expressed by $(\mathbf{v}_i - \mathbf{v}_j)^T \mathbf{n}_k = 0$, for all edges $\mathbf{v}_i\mathbf{v}_j$ of that face. It is also convenient to require the normalization $\mathbf{n}_k^T \mathbf{n}_k = 1$. Again, we get a set of quadratic constraints.

Honeycomb structures, as defined by Figure 2 are studied by Jiang et al. [2014]. They are arrangements of quadrilaterals combinatorially different from the arrangement of faces of a mesh; however the constraints describing their planarity are the same. The required intersection angles of 120° are most elegantly expressed by requiring that whenever faces No. i, j, k meet in a common axis, their normal vectors must form an equilateral triangle, which results in the equation $\mathbf{n}_i + \mathbf{n}_j + \mathbf{n}_k = \mathbf{o}$.

Self-supporting meshes. We wish to incorporate *forces* in our computational setting, since meshes with compressive equilibrium forces in their edges (Figure 4) play an important role e.g. in the stability analysis of masonry, see e.g. [Block and Ochsendorf 2007]. The force which vertex No. j exerts on vertex No. i has the form $w_{ij}(\mathbf{v}_i - \mathbf{v}_j)$. Since $w_{ij} = w_{ji}$, information on forces is stored via one force coefficient w_{ij} per edge. This edge experiences compression if $w_{ij} \geq 0$. The inequality $w_{ij} \geq 0$ is made an equality by introducing a dummy variable ω_{ij} and requiring $w_{ij} = \omega_{ij}^2$ (while w_{ij} represents a force per edgelength, ω_{ij} is there only to assist in a mathematical trick and does not have a physical interpretation).

As to constraints, we must formulate what the forces should be in equilibrium with. The simplest case, which e.g. applies to Figure 4b, is discussed by Figure 3.

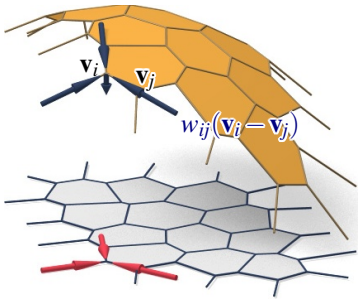


Figure 3: *Force equilibrium.* The simplest case is that forces $w_{ij}(\mathbf{v}_i - \mathbf{v}_j)$ in edges $\mathbf{v}_i\mathbf{v}_j$ counterbalance the weight of edges, which is modelled by the weight "ρ" per unit length. With $(0,0,-1)$ as direction of gravity, force balance at vertex $\mathbf{v}_i$ reads

$$(0,0,-1) \cdot \sum_{j\sim i} \rho l_{ij} = \sum_{j\sim i} w_{ij}(\mathbf{v}_i - \mathbf{v}_j)$$

where summation is over all vertices $\mathbf{v}_j$ connected to $\mathbf{v}_i$ by an edge, and the edge lengths are defined by $l_{ij}^2 = (\mathbf{v}_i - \mathbf{v}_j)^T(\mathbf{v}_i - \mathbf{v}_j)$.

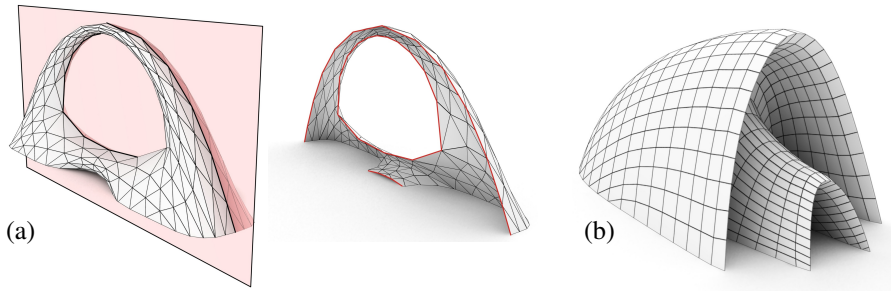


Figure 4: *Self-supporting meshes* represent a bar-and-joint framework with compressive forces in the edges counterbalancing the deadload. They serve as ficticious "thrust networks", proving stability of freeform masonry via Heyman's Safe Theorem (a), or as the basis of structures made of beams and rigid joints (b). *Remark:* These images do not show forces.

Again, all equations are quadratic. For details and other ways to define the weight, e.g. proportional to area as shown by Figure 4a, see [Tang et al. 2014].

Projection onto the constraint space. Tang et al. [2014] describe how to treat constraints of the kind described above (the restriction that constraints should be at most quadratic is not severe, since any equality or inequality expressible in polynomial form can be made quadratic, by introducing products of two existing variables as new variables, until the polynomial degree is down to 2). They describe how to quickly solve the system of constraints, and thus make interactive modeling of constrained geometry possible. A typical application of their method would be to implement a graphical interface where a user can interactively create and modify a given geometry, while the system performs *projection onto the constraint space*, i.e., searching for variables which fulfill the constraints and which are close to the values they had before, and close to those the user wanted.

For their algorithm, Tang et al. [2014] make use of a fairness energy "$E(\mathbf{x})$" which is a quadratic expression in the collection $\mathbf{x}$ of variables. The definition of E varies depending on the application. The method is a Newton iteration; in the i-th iteration the nonlinear system of constraint equations is converted to a linear system of the general form $A_i\mathbf{x} - \mathbf{s}_i = \mathbf{o}$, whose solution $\mathbf{x}_i$ is computed. The iteration stops whenever the desired accuracy is reached or $\mathbf{x}_i$, $\mathbf{x}_{i+1}$ are equal for all practical purposes. Linearization of quadratic equations is a standard procedure; nonlinear constraints like vertices being confined to a curve or surface are linearized by replacing those curved objects by their tangent resp. tangent plane. Since the system of constraints is both underdetermined and redundant, one cannot solve $A_i\mathbf{x} - \mathbf{s}_i = \mathbf{o}$ directly. Instead $\mathbf{x}_i$ is found via regularization, as minimizer of $\|A_i\mathbf{x} - \mathbf{s}_i\|^2 + \varepsilon E(\mathbf{x}_i)$ $+ \varepsilon' \|\mathbf{x}_i - \mathbf{x}_{i-1}\|^2$, with $\varepsilon, \varepsilon' \ll 1$. For details we refer to [Tang et al. 2014].

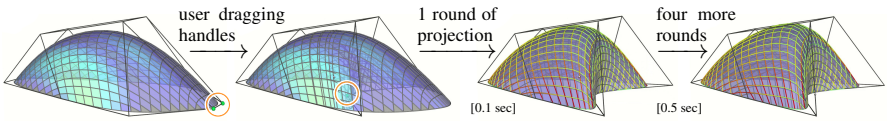


Figure 5: *Interactive modeling of forces in the edges of a quad mesh under planarity constraints*. We use the method of Tang et al. [2014] to combine subdivision and projection onto the constraint space (defined by planarity of faces, the self-supporting property, and the condition that boundary vertices must not leave the boundary curve of the subdivision surface generated by the control points as set by the user). (a) The user works on the control points of a mesh created by subdivision. (b) Each time the user moves the control points and releases them, the system preforms projection onto the constraint space. (c), (d) Numerical information is conveyed by color coding the faces (blue means planar) and edges (color corresponds to magnitude of forces).

3 Interactive form-finding with polyhedral meshes

Loss of design freedom by geometric constraints. Meshes with planar faces are important objects of architectural geometry, since they represent the shapes of steel-glass-constructions. Their modeling is, of course, easiest for triangle meshes, because planarity is automatic there. Quad meshes are more of a challenge. It is known that "fair" quad meshes (whose mesh polylines emulate the isoparameter lines of smooth surfaces) offer little design freedom once the reference surface to be covered is fixed. If edges are required to intersect near-orthogonally, they must already follow the principal directions of the reference surface [Liu et al. 2006], leaving only the density of edges as a design element. This is true also for quad meshes whose edges carry forces counterbalancing vertical loads: The edges must follow *relative* principal curvature lines (with the Airy stress potential assuming the role of unit sphere, cf. [Vouga et al. 2012]). With either side-condition it is usually impossible to perform a fair quad meshing of the reference geometry such that the mesh's boundary is nicely aligned with the original surface's boundary.

Interactive modeling of polyhedral meshes. If one can solve the constraint equations quickly enough, however, then interactive geometric modeling becomes available. It is no longer necessary to perform surface analysis and subsequent remeshing as proposed e.g. by [Liu et al. 2006; Vouga et al. 2012]. Figure 4b shows a self-sup-

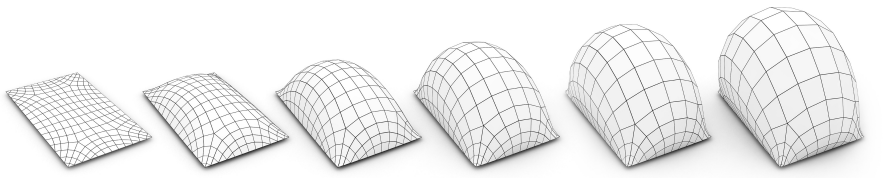


Figure 6: Interactively modifying a polyhedral mesh by increasing the enclosed volume.

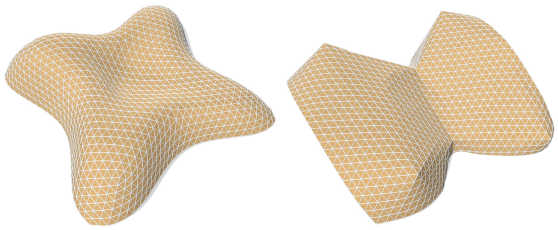


Figure 7: Projecting a triangle mesh onto the constraint space defined by the "Lobel" property makes all faces equilateral and the mesh developable, apart from cuts necessary to accommodate valence 5 vertices.

porting quad mesh created in this way. For details see Figure 5.

The computational setup for this projection is as described in § 2. As fairness energy we use the sum of terms of the form $\|\mathbf{v}_i - 2\mathbf{v}_j + \mathbf{v}_k\|^2$, for each possible choice of consecutive vertices $\mathbf{v}_i, \mathbf{v}_j\mathbf{v}_k$ on a mesh polyline.

We do not pursue this topic further, since it is anyway treated by Tang et al. [2014]. We mention only a few more constraints easily incorporated into this setup, such as symmetry w.r.t. to a plane, or w.r.t. to rotation about an axis, or inequalities involving panel sizes. Another constraint is a prescribed total area of the mesh, or a prescribed enclosed volume, see Figure 6. Constraints like the latter however involve many more variables than the ones expressing local properties like planarity. They will slow down computations, since the matrices A_i will no longer be sparse.

4 Element repetition: Honeycomb structures and Lobel frames

For fabrication of freeform skins in architecture, it is very relevant if a substantial number of parts is the same. E.g. Sing and Schaefer [2010] studied meshes where only a small number of shapes of triangles occur. Here we are very much interested in two kinds of geometry: meshes where all faces are the same, and support structures where all nodes are the same. These two questions lead us to the concept of Lobel mesh on the one hand, and honeycomb structure on the other hand.

Lobel meshes. Frameworks built from equilateral triangles are sometimes called "Lobel frames", after the French architect Alain Lobel who intensively studied them. We are here interested in triangle meshes all of whose faces are equilateral triangles and which we would like to call *Lobel* meshes. An equivalent definition of such a Lobel mesh is that all edges have the same length. Yet another definition is that all angles in the mesh are 60 degrees. Examples are shown by Figures 1, and 7.

The duality Honeycomb — Lobel mesh. Objects dual to Lobel meshes are honeycomb structures, defined as an arrangement of open hexagonal cells bounded by quadrilateral walls, such that the cells and walls follow the faces and edges of a hexagonal mesh, respectively. In addition we require that walls intersect at 120 degrees — see Figure 2. This leads to the important property that any structure made of beams which follow the walls of the honeycomb has *congruent nodes*. For modeling and properties of honeycombs see [Jiang et al. 2014].

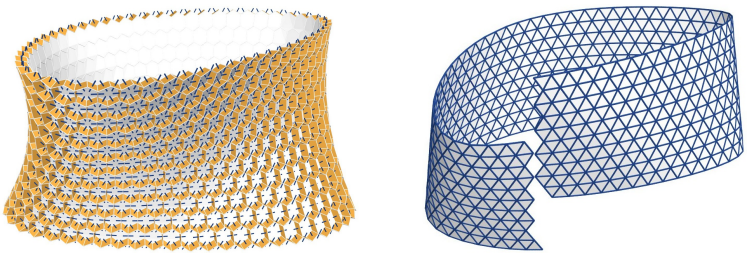


Figure 8: *Lobel meshes and Honeycombs*. A combinatorial dual to the walls of a honeycomb (left) is a triangle mesh where edges intersect at 60 degrees (right), implying the Lobel property. Step-by-step construction of the Lobel mesh orthgonal to the honeycomb is possible only for simply connected honeycombs; otherwise the construction does not globally close.

Given a simply connected honeycomb, one can step by step construct a Lobel mesh whose edges are orthogonal to the walls of the honeycomb, see Figure 8. Conversely, given a Lobel mesh, one may construct the corresponding honeycomb. The honeycomb, however, carries more geometric information than the Lobel mesh: the latter is uniquely determined, up to scale, by the planes carrying the walls of the honeycomb. The information on the location of vertices of the honeycomb is lost. This duality between honeycomb structures and Lobel meshes is easy to see, but it does not seem to have been mentioned in earlier publications. We also mention that Lobel meshes are a special case of the *circle packing meshes* studied by Schiftner et al. [2009], since placing spheres at the vertices (whose diameter is the mesh's edge length) yields a packing.

Discrete developable surfaces. Lobel meshes are *developable*, meaning that every 1-ring neighbourhood of a valence 6 vertex can be mapped isometrically into the plane without stretching or tearing. This follows immediately from the fact that all angles in the mesh are 60 degrees. Depending on the global topology and geometry of the mesh, one can grow this developable neighbourhood by adding more and

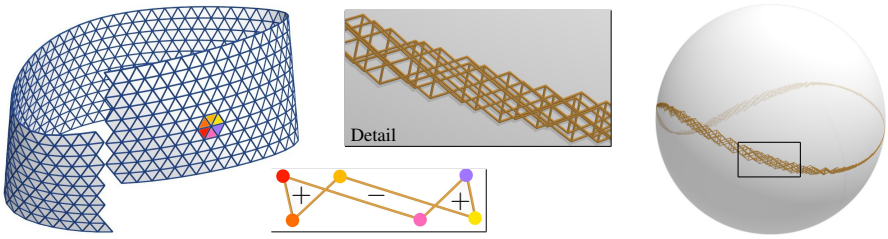


Figure 9: *Local and global properties of Lobel meshes.* The image at right illustrates the curve-like *spherical image* of the Lobel mesh at left. It follows from the discussion in [Jiang et al. 2014] that the normal vectors of a 1-ring neighbourhood of any vertex typically form a zero area hexagon on the unit sphere.

more triangles. If a vertex has valence different from 6, we must cut the Lobel mesh near that vertex to develop it into the plane, cf. Figures 1 and 10b.

A Lobel mesh which approximates a *smooth* surface should therefore have properties similar a developable surface. One such property is that developables have only a 1-dimensional variety of normal vectors in contrast to the 2-parameter set which non-developable surfaces have. Lobel meshes exhibit a discretized version of this behaviour, cf. Figure 9: When constructing a Lobel mesh from a honeycomb, the intersection lines of walls become the normals of faces in the Lobel mesh. We visualize those normals by the *spherical image* of the Lobel mesh. As Jiang et al. [2014] remark in their discussion of honeycombs, the spherical image of the Lobel mesh consists of zero-area spherical hexagons (which follows from the Gauss-Bonnet theorem), and a certain fairness/regularity implies that these hexagons fill a curve-like stripe on the unit sphere. For details we refer to that paper.

5 Novel forms of fairness: Polyhedral patterns

Motivation. We already discussed the problematic task of representing a given reference shape by a mesh with planar quadrilateral faces. If the mesh polylines are to mimick the isoparameter lines of a smooth surface parameterization e.g. in the manner of Fig. 6, we have very little design freedom, because such a quad mesh is never far away from the network of principal curvature lines, cf. [Liu et al. 2006; Zadravec et al. 2010]. If the reference surface is not convex, the resulting meshes might easily be unacceptable (besides the fact that lack of design freedom is frequently unacceptable in itself). There are several ways out of this situation:

1. One might give up the condition that all faces are quadrilaterals and thus gain design freedom;
2. one might give up the condition that the mesh strictly follows the reference geometry (making use of interactive modeling tools);
3. one might give up the condition of smoothness/regularity.

Solution No. 1 has been sought for the “flying carpet” roof in the Cour Visconti in the Louvre, Paris, by R. Ricciotti and M. Bellini. Even if not visible through the outer skin of triangular shading elements, many of the glass panels below are quadrilaterals, making the structure lighter and have fewer parts.

Polyhedral patterns from honeycombs. For the third solution proposed above, it is obviously important that regularity is given up in an aesthetic and regular (if it may be called that) way. What we mean by this is demonstrated by means of the example shown by Figure 10: Jiang et al. [2014] have proposed to derive polyhedral patterns from freeform honeycomb structures. The honeycomb contains a “top layer” hexagonal mesh, and in this mesh we split hexagons in half by introducing additional edges. The resulting quad mesh is being planarized by means of the computational framework of Tang et al. [2014]. We cannot however employ this

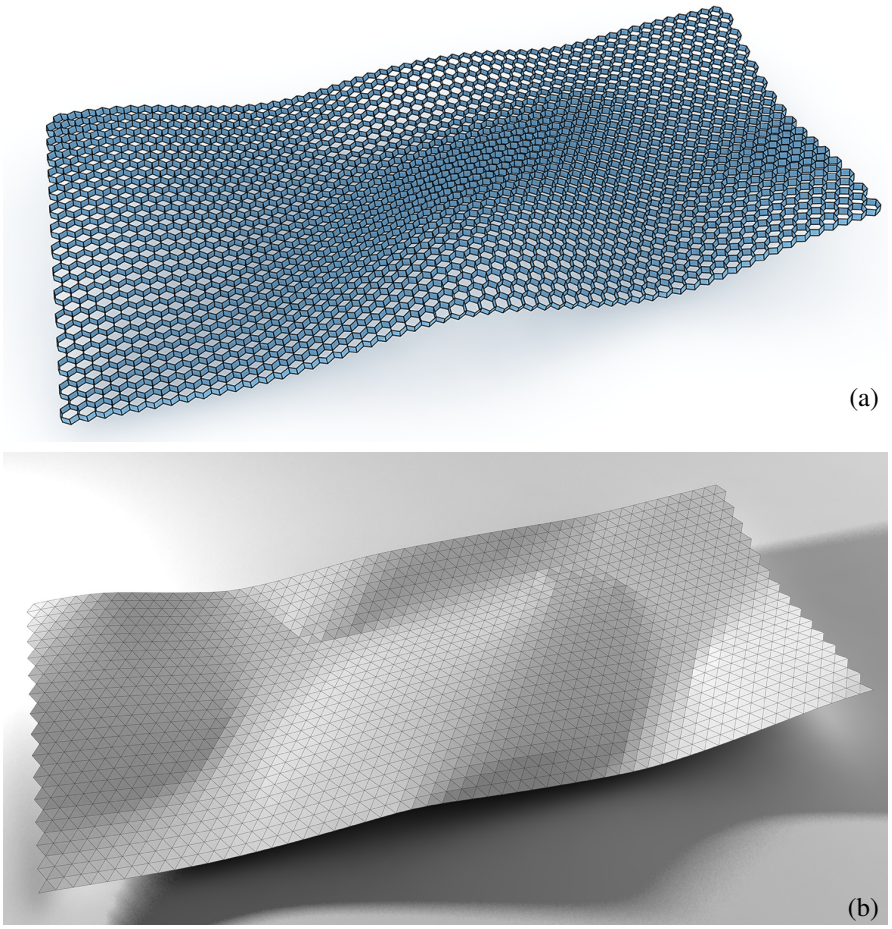
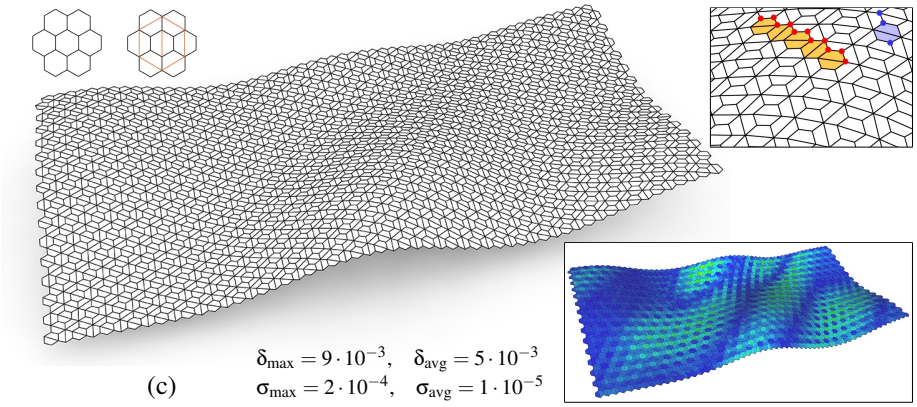


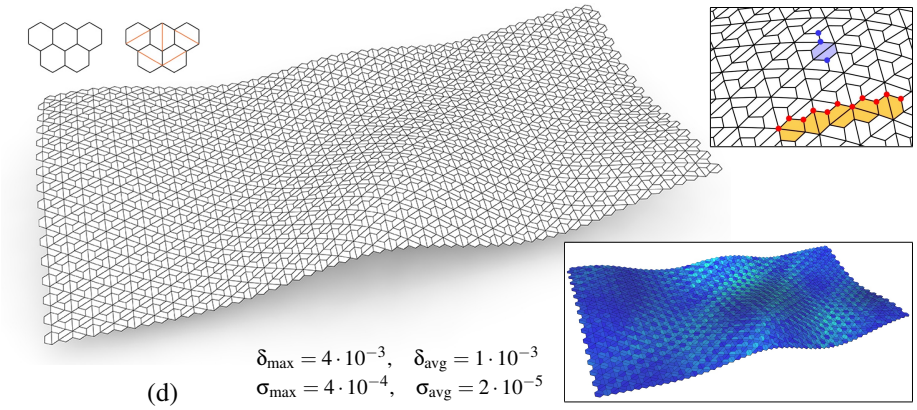
Figure 10: *Honeycombs and polyhedral patterns.* (a) A honeycomb following the "flying carpet" roof in the Cour Visconti in the Louvre, Paris. Its walls are near-orthogonal to the reference geometry in so far as they have been initialized this way, before projection onto the constraint manifold has been applied.

(b) The Lobel mesh dual to this honeycomb. Developability of this surface is visualized by a rendering with tangential light, recalling crumpled paper.

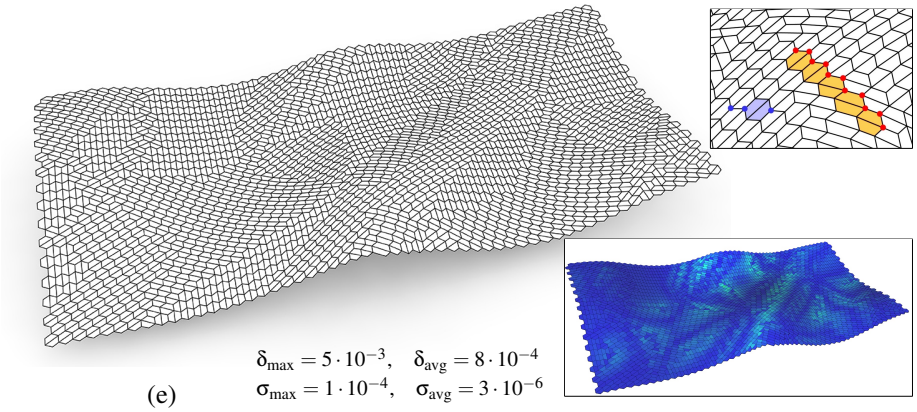
(c–e) (*opposite page*) Consider the top layer hex mesh of the honeycomb in (a), introduce new edges (colored strokes in left hand insets) and seek a nearby polyhedral mesh, guided by an alternate smoothness energy. While the patterns in (c) and (d) follow a single rule, the one in (e) is irregular. It is created by a greedy rule: among the three possibilities to split a hexagon in half we take the one which makes the resulting quads most planar. The inset figures are right show a detail and the quality of planarity by color coding faces according to the value of δ, where δ is defined as distance of diagonals, divided by average edge length. We also give the value σ which is the distance of vertices to the reference geometry, divided by average edge length of the mesh.



(c) $\delta_{max} = 9 \cdot 10^{-3}$, $\delta_{avg} = 5 \cdot 10^{-3}$
$\sigma_{max} = 2 \cdot 10^{-4}$, $\sigma_{avg} = 1 \cdot 10^{-5}$



(d) $\delta_{max} = 4 \cdot 10^{-3}$, $\delta_{avg} = 1 \cdot 10^{-3}$
$\sigma_{max} = 4 \cdot 10^{-4}$, $\sigma_{avg} = 2 \cdot 10^{-5}$



(e) $\delta_{max} = 5 \cdot 10^{-3}$, $\delta_{avg} = 8 \cdot 10^{-4}$
$\sigma_{max} = 1 \cdot 10^{-4}$, $\sigma_{avg} = 3 \cdot 10^{-6}$

algorithm in the way in its original form, since it is guided by a smoothness energy which is not appropriate here.

In order to derive an alternative smoothness energy, we have a closer look a the patterns under consideration. Obviously patterns 10c,d are quite regular, but the edges do not form "straight" mesh polylines. This is even less so for pattern 10e. It is still possible, however, to select sequences of vertices on a succession of hexagons which form a periodic zigzag sequence, such as those highlighted in red in Figure 10 (right hand insets).

A zigzag sequence is considered fair, if $(\mathbf{v}_i - \mathbf{v}_j) - (\mathbf{v}_k - \mathbf{v}_l)$ is small, for any choice of consecutive vertices $\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k, \mathbf{v}_l$. We therefore take as a fairness energy the sum of squares of such expressions. Similarly we consider triples $\mathbf{v}_i$, $\mathbf{v}_j$, $\mathbf{v}_k$ of vertices analogous to those highlighted in blue – there are six of them for every hexagon. We would like the kink in the short polyline $\mathbf{v}_i\mathbf{v}_j\mathbf{v}_k$ to be small, so we add as a constraint $(\mathbf{v}_i - \mathbf{v}_j) \times (\mathbf{v}_j - \mathbf{v}_k) = 0$, but mulitplied with a small factor to make it a "soft" constraint. With these modifications, the method of Tang et al. [2014] works fine, as demonstrated by Figure 10.

6 Conclusion

We have shown how the algorithmic concept of Tang et al. [2014] can be applied to several classes of geometric objects which are relevant to architectural geometry for various reasons (flatness of panels, or repetition in elements). These objects include:

- Polyhedral meshes in general, and polyhedral meshes with equilibrium forces in their edges in particular.
- More general than meshes are honeycomb structures with their interesting relations to Lobel meshes and to developable surfaces.
- An interesting topic are polyhedral patterns. We have in particular demonstrated patterns derived from honeycomb structures.

While part of this paper is merely an exposition of the capabilities of [Tang et al. 2014] we have also presented new geometry, namely the relations beween honeycomb structures, Lobel frames, and developable surfaces, thereby extending recent work on honeycombs, cf. [Jiang et al. 2014].

Limitations. The algorithmic concept for solving constraint equations we used in this paper has turned out to be efficient and fast in such cases where the constraints can be formulated by linear or quadratic equations each of which involve only few variables. Already Tang et al. [2014] observed that higher-order polynomial constraints cause a dramatic drop in performance. So do constraints which involve many variables (such as total volume of a mesh).

A different kind of limitation is posed by geometric problems which contain a discrete optimization component, such as a change in combinatorics. An example of this is the paneling algorithm of Eigensatz et al. [2010] which contains an

assignment problem. Relevant to our work, it would be a challenge to incorporate automatic changes in combinatorics into our modeling system for polyhedral meshes.

Future Work. It should not be difficult to extend the methods discussed in this paper to situations where the variables and constraints do not act on "visible" geometry variables directly, but indirectly, e.g. on control points of NURBS surfaces. In this way e.g. developability of NURBS can be incorporated into our computational framework. An interactive system which combines aspects of shape, function, and fabrication is a long term goal of our work. For all types of applications (developables, crumpled paper, ...), comparison with existing methods is needed.

Acknowledgments

This research was supported by KAUST base funding, NAWI Graz funding, by the DFG-Collaborative Research Center, TRR 109 *Discretization in Geometry and Dynamics*, through grants I 705 N-26 and I 706-N26 of the Austrian Science Fund (FWF), by FWF project P 23735-N13 and by the European Community's 7th Framework Programme under grant agreement 286426 (GEMS).

References

BARNES, M. R. 2009. Form finding and analysis of tension structures by dynamic relaxation. *Int. J. Space Struct. 14*, 2, 89–104.

BLOCK, P., AND LACHAUER, L. 2011. Closest-fit, compression-only solutions for free form shells. In *Proc. IABSE — IASS London Symposium*. 8 pp.

BLOCK, P., AND OCHSENDORF, J. 2007. Thrust network analysis: A new methodology for three-dimensional equilibrium. *J. Int. Assoc. Shell and Spatial Structures 48*, 3, 167–173.

BLOCK, P. 2009. *Thrust Network Analysis: Exploring Three-dimensional Equilibrium*. PhD thesis, MIT.

BOUAZIZ, S., SCHWARTZBURG, Y., WEISE, T., AND PAULY, M. 2012. Shaping discrete geometry with projections. *Computer Graphics Forum 31*, 1657–1667.

DE GOES, F., ALLIEZ, P., OWHADI, H., AND DESBRUN, M. 2013. On the equilibrium of simplicial masonry structures. *ACM Trans. Graph. 32*, #93, 1–10.

DENG, B., BOUAZIZ, S., DEUSS, M., ZHANG, J., SCHWARTZBURG, Y., AND PAULY, M. 2013. Exploring local modifications for constrained meshes. *Computer Graphics Forum 32*, 11–20.

DENG, B., BOUAZIZ, S., DEUSS, M., KASPAR, A., SCHWARTZBURG, Y., AND PAULY, M. 2014. Interactive design exploration for constrained meshes. *Computer-Aided Design*.

EIGENSATZ, M., KILIAN, M., SCHIFTNER, A., MITRA, N., POTTMANN, H., AND PAULY, M. 2010. Paneling architectural freeform surfaces. *ACM Trans. Graph. 29*, #45, 1–10.

JIANG, C., WANG, J., WALLNER, J., AND POTTMANN, H. 2014. Freeform honeycomb structures. *Computer Graph. Forum 33*, 5. Proc. SGP.

LACHAUER, L., AND BLOCK, P. 2012. Compression support structures for slabs. In *Advances in Architectural Geometry 2012*, L. Hesselgren et al., Eds. Springer, 135–146.

LIU, Y., POTTMANN, H., WALLNER, J., YANG, Y.-L., AND WANG, W. 2006. Geometric modeling with conical meshes and developable surfaces. *ACM Trans. Graph. 25*, 3, 681–689.

LIU, Y., PAN, H., SNYDER, J., WANG, W., AND GUO, B. 2013. Computing self-supporting surfaces by regular triangulation. *ACM Trans. Graph. 32*, #92, 1–10.

LOBEL, A. 1993. *Formes et structures engendrées par des éléments identiques*, vol. 1–6. A. Lobel, Bourg-la-Reine. ISBN 2-9514254-0-6.

PANOZZO, D., BLOCK, P., AND SORKINE-HORNUNG, O. 2013. Designing unreinforced masonry models. *ACM Trans. Graph. 32*, #91, 1–12.

PORANNE, R., OVREIU, E., AND GOTSMAN, C. 2013. Interactive planarization and optimization of 3D meshes. *Computer Graphics Forum 32*, 1, 152–163.

SCHIFTNER, A., AND BALZER, J. 2010. Statics-sensitive layout of planar quadrilateral meshes. In *Advances in Architectural Geometry 2010*, C. Ceccato et al., Eds. Springer, 221–236.

SCHIFTNER, A., HÖBINGER, M., WALLNER, J., AND POTTMANN, H. 2009. Packing circles and spheres on surfaces. *ACM Trans. Graph. 28*, #139, 1–8.

SINGH, M., AND SCHAEFER, S. 2010. Triangle surfaces with discrete equivalence classes. *ACM Trans. Graph. 29*, #46, 1–7.

TANG, C., SUN, X., GOMES, A., WALLNER, J., AND POTTMANN, H. 2014. Form-finding with polyhedral meshes made simple. *ACM Trans. Graph. 33*. Proc. SIGGRAPH.

VAN MELE, T., AND BLOCK, P. 2011. A novel form finding method for fabric formwork for concrete shells. *J. Int. Assoc. Shell Spatial Struct. 52*, 217–224.

VOUGA, E., HÖBINGER, M., WALLNER, J., AND POTTMANN, H. 2012. Design of self-supporting surfaces. *ACM Trans. Graph. 31*, #87, 1–11.

ZADRAVEC, M., SCHIFTNER, A., AND WALLNER, J. 2010. Designing quad-dominant meshes with planar faces. *Computer Graphics Forum 29*, 1671–1679.

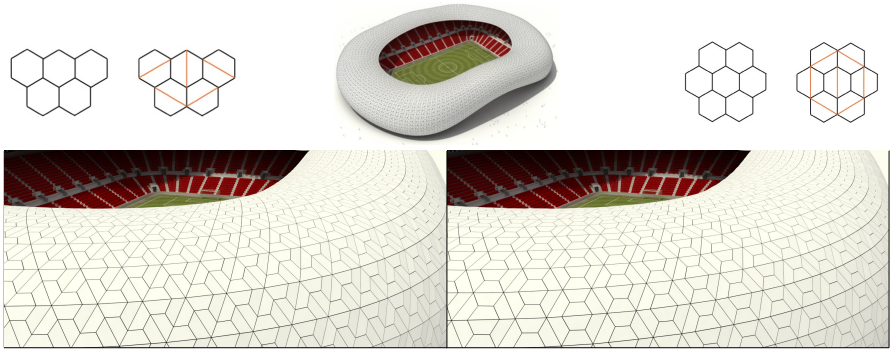


Figure 11: Different kinds of polyhedral patterns resolve this freeform skin into planar quads. Like for the example of Figure 10, the pattern has been found via a two-step procedure: (i) compute a honeycomb structure following the given reference surface, and (ii) split hexagons in half and planarize the resulting quad mesh. Both steps use the method of [Tang et al. 2014]; the 2nd step being guided by the "zigzag" fairness energy mentioned in § 5.