Form-finding with Polyhedral Meshes Made Simple

Chengcheng Tang, Xiang Sun, Alexandra Gomes
King Abdullah University of Science and Technology (KAUST)

Johannes Wallner
TU Graz

Helmut Pottmann
KAUST / TU Wien

Abstract

We solve the form-finding problem for polyhedral meshes in a way which combines form, function and fabrication; taking care of user-specified constraints like boundary interpolation, planarity of faces, statics, panel size and shape, enclosed volume, and last, but not least, cost. Our main application is the interactive modeling of meshes for architectural and industrial design. Our approach can be described as guided exploration of the constraint space whose algebraic structure is simplified by introducing auxiliary variables and ensuring that constraints are at most quadratic. Computationally, we perform a projection onto the constraint space which is biased towards low values of an energy which expresses desirable "soft" properties like fairness. We have created a tool which elegantly handles difficult tasks, such as taking boundary-alignment of polyhedral meshes into account, planarization, fairing under planarity side conditions, handling hybrid meshes, and extending the treatment of static equilibrium to shapes which possess overhanging parts.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations;

Keywords: Discrete differential geometry, architectural geometry, form-finding, static equilibrium, self-supporting surfaces, polyhedral surfaces, constraint space, guided projection

Links: DL PDF

1 Introduction

Shape modeling systems often provide the user with little support to satisfy constraints implied by function and fabrication of the designed product. Very recently however we see a trend towards novel design tools combining form, function and fabrication. This new integrated approach is a big challenge and obviously more specific to the intended application than classical geometric modeling, see e.g. [Umetani et al. 2012].

Geometrically complex architecture is one of the areas where an integrated design approach is in high demand. Most of the previous work in this field deals with the combination of form and fabrication; it has already entered a variety of real projects, one of the most recent and prominent ones being the Eiffel Tower Pavilions [Schiftner et al. 2012]. We go one step further and integrate other key aspects like statics, panel sizes, enclosed volume, total weight, and cost. This application scenario however is not the only one where our methods can be applied. The basic problem we solve is *geometric and structural form-finding* with *polyhedral meshes*, i.e., meshes with planar faces, where we particularly emphasize those with mostly quadrilateral faces. This problem may be formulated as follows: Provide a system for interactive design of polyhedral meshes which fit given boundaries and other constraints provided by geometry, statics, manufacturing and user input.

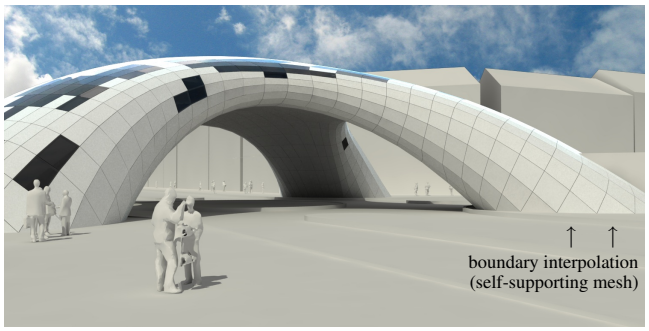


Figure 1: *Form-finding with polyhedral meshes for architectural design. We develop an interactive modeling system for polyhedral meshes which can handle geometric properties like alignment constraints of boundary vertices and planarity of faces, as well as statics constraints like compressive-only equilibrium forces in the edges (overcoming the height field limitation in previous treatments of the self-supporting property).*

An example of the kind of problem solved in this paper, is geometric form-finding with planar quad (PQ) meshes and given boundary. Using available structural form-finding tools to pass a surface S through the boundary and then remeshing it by a geometrically fair PQ mesh, will usually not achieve that vertices are placed exactly on the boundary other than by simple trimming generating a cut-off look. This is because of the close relation between PQ meshes and the surface's curvature behavior. Remeshing a surface by a PQ mesh in static equilibrium has an even stronger relation to curvature and exhibits similar behaviour: there is no freedom in the directions of edges [Vouga et al. 2012]. Also here, remeshing is incapable of producing boundary aligned meshes, in the sense that vertices lie exactly on prescribed boundary curves in a geometrically fair manner such as shown by Fig. 1 (the boundary becoming either a mesh polyline or a sequence of face diagonals).

Related Work. Polyhedral meshes and in particular PQ meshes have a number of favorable properties which make them great candidates for architecture [Glymph et al. 2004]. Research in Architectural Geometry thus had a rather strong focus on PQ meshes and their relatives [Liu et al. 2006; Pottmann et al. 2007] and meanwhile led to effective algorithms for remeshing (rationalization) [Liu et al. 2011; Zadravec et al. 2010] and optimization [Bouaziz et al. 2012; Poranne et al. 2013b]. Spaces of polyhedral meshes and other constrained meshes have been exploited for design and exploration [Yang et al. 2011; Vaxman 2012; Poranne et al. 2013a; Deng et al. 2013]. In particular we point to two contributions which address issues relevant to our paper: an important technical detail has been proposed by Poranne et al. [2013b]. They introduce normal vectors

of faces as extra variables but did not fully exploit the potential of this idea for real-time computations. The latter aim is something we share with [Kaspar and Deng 2013; Deng et al. 2014].

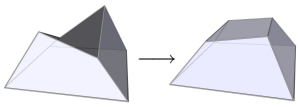
The combination of structural analysis and shape modeling is probably best studied for self-supporting masonry. In particular we here refer to the thrust network method [Block and Ochsendorf 2007; Block 2009], which may be seen as a non-conforming finite element discretization of the continuous theory of stresses in membranes [Fraternali 2010], and which is the basis of recent work on the interactive computational design of self-supporting masonry. Vouga et al. [2012] uses Maxwell's reciprocal force diagram and a discrete Airy potential to embed self-supporting surfaces into discrete differential geometry. The various degrees of freedom inherent in thrust networks were employed in different ways for computation of self-supporting surfaces by de Goes et al. [2013], Liu et al. [2013], and Panozzo et al. [2013]. All these methods treat self-supporting surfaces at least locally as height fields (which is a restriction overcome by our paper).

As to form-finding for self-supporting shapes (unrelated to polyhedral meshes) see the references in [Vouga et al. 2012], in particular hanging chain or membrane models [Heyman 1998; Kotnik and Weinstock 2012]. Force density methods [Linkwitz and Schek 1971] linearize the form-finding problem by solving for static equilibrium with respect to position variables, given prescribed prestresses in the form of axial force densities [Gründig et al. 2000].

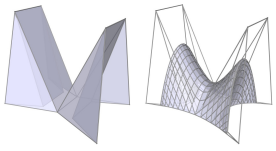
There is very little work on statically sound polyhedral meshes. The statics-sensitive design of planar quad meshes has been addressed by Schiftner and Balzer [2010] in a decoupled form-finding and PQ-remeshing fashion, which is a method with many limitations. Vouga et al. [2012] showed that self-supporting PQ meshes are guided by the coupled curvature of the design surface and the Airy stress surface. However, this approach has not been extended to a solution of the present form-finding problem.

2 Contributions — System Overview

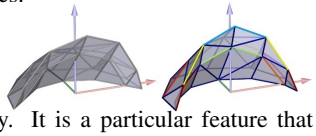
Before discussing algorithms we start with an overview of the features which our modeling system for polyhedral meshes presents to the user. From a higher viewpoint, this system consists of guided exploration of the space of meshes defined by constraints, and its main novelty lies in the unified treatment of side conditions which concern both geometry and statics. The good numerical performance of the system is due to the simple algebraic structure of the constraint manifold which is achieved by making constraints and fairness energies quadratic.



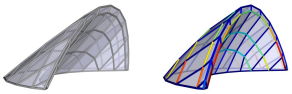
• A basic task within the system is to modify a mesh such that its faces become planar (*planarization*). The constraint of planarity of faces is maintained throughout all computations and during interactive deformations of meshes, e.g. when a user drags a vertex.



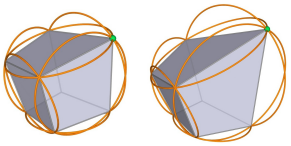
• Our system allows to define meshes by subdivision, followed by a nonlinear step. This defines a coarser layer of handles, or even several layers of handles, which allow for intuitive multiresolution design in the manner which is well known for subdivision surfaces.



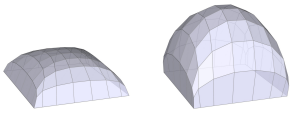
• We can create and interactively edit *thrust networks* of forces, and even of compressive forces useful to design self-supporting masonry. It is a particular feature that we can deal with networks which have overhanging parts. This distinguishes our approach from others.



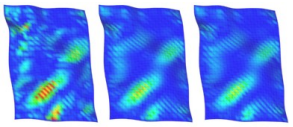
• A new feature is the handling of meshes which have both planar faces, and equilibrium forces in their edges. Again, we handle meshes which are not height fields.



• We can do exploration of meshes with constraints like circular faces or other offset-relevant properties.



• For given combinatorics, we can explore the possible shapes of polyhedral meshes with this combinatorics which fit a given boundary (inset figures show interactive deformation guided by enclosed volume). This exploration extends to meshes in static equilibrium, to meshes with offset properties, and other kinds of constrained meshes.



• The system offers fairing of meshes (plus fairing of Gauss images) under the constraint that faces remain planar. This example concludes this brief system overview. For more details we refer to the individual results presented later.

3 Algorithms and Results

Our approach to the exploration and design of polyhedral surfaces can be described as projection onto a manifold defined by constraints which is biased towards small values of an energy derived from fairness and "soft" constraints. This section describes this procedure in detail: the choice of variables and the constraints they are subject to in §3.1, soft constraints in §3.2, the algorithmic part in §3.3, and initialization of variables in §3.4. In order not to interrupt the flow of the presentation we first present only a simple setup, and describe extensions in §3.5.

3.1 Variables and Constraints

In contrast to triangle meshes, which are encoded in a simple manner by the connectivity and coordinates of vertices, the vertex coordinates of general polyhedral meshes are not independent, and finding an independent set of variables for a polyhedral mesh can be hard. It also turns out that the introduction of additional variables yields to very efficient computations. This section describes our choice of variables for a modeling system for polyhedral meshes, which is guided by the principle that the constraints imposed on the variables should be linear or quadratic.

Vertex coordinates and face normals as variables. For a mesh $\mathcal{M} = (V, E, F)$, we use the vertex coordinates $\mathbf{v}_i$ and also normal vectors $\mathbf{n}_k$ of faces f_k as variables, amounting to a total of $3|F| + 3|V|$ scalar variables. Constraints imposed on them are firstly the conditions $\mathbf{n}_k^T \mathbf{n}_k = 1$, expressing that all normal vectors are unit vectors. Secondly, we use the condition

$$\mathbf{n}_k^T(\mathbf{v}_i - \mathbf{v}_j) = 0$$

to express that the edge $\mathbf{v}_i\mathbf{v}_j$ is contained in the face f_k with normal vector $\mathbf{n}_k$. For each n-gon f_k there are n such conditions. We use them all for reasons of symmetry even if we would need only

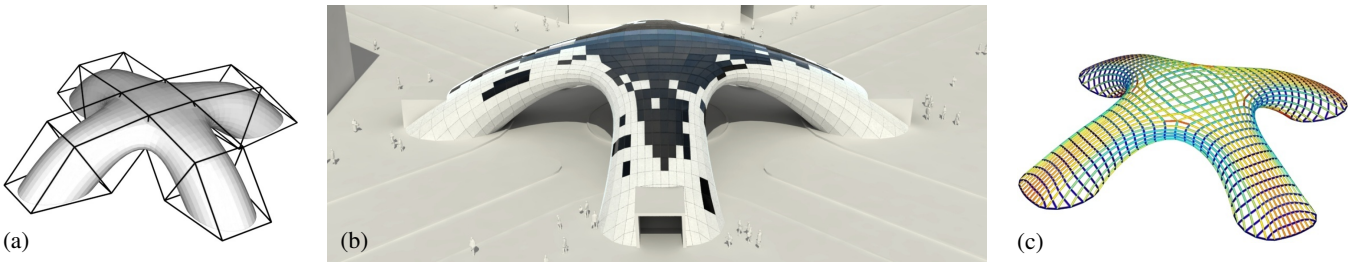


Figure 2: *Self-supporting meshes which are not height fields. (a) The user models a subdivision surface (via control points) to be used as reference shape. From there, we initialize geometry variables directly, and initial forces by calling a bounded least squares algorithm [Liu et al. 2009]. Subsequent projection onto the constraint manifold involves, among others, compressive nature of forces, approximation of the reference shape (a soft nonlinear constraint), gliding of the boundary along the reference shape's boundary (a hard nonlinear constraint), and equally distributed areas of faces. The resulting mesh is shown in (b) and also in Figure 1. (c) Color-coded magnitude of equilibrium forces in the edges of the mesh.*

$n-1$ of them. We emphasize that unlike usual constrained optimization procedures, our computational approach is robust w.r.t. dependencies between constraints.

Remark: We require that normal vectors are consistent with an orientation of the mesh (e.g., they are all pointing outwards).

Edge lengths and forces as variables. Since stability and the self-supporting property of meshes are included in our computations, we introduce the length $l_{ij} \geq 0$ of an edge $\mathbf{v}_i\mathbf{v}_j \in E$ and a force coefficient w_{ij} as new variables: The vector $w_{ij}(\mathbf{v}_i - \mathbf{v}_j)$ represents the force exerted on vertex $\mathbf{v}_i$, while the opposite force $w_{ij}(\mathbf{v}_j - \mathbf{v}_i)$ is exerted on the vertex $\mathbf{v}_j$. Tensile forces (like in cables) have $w_{ij} \leq 0$, while compressive forces have $w_{ij} \geq 0$.

These $2|E|$ new variables introduce new constraints, beginning with the defining equations $l_{ij}^2 = (\mathbf{v}_i - \mathbf{v}_j)^T(\mathbf{v}_i - \mathbf{v}_j)$ of edge lengths. As to forces, our setup is typically applied to a mesh which guides a structure whose beams follow the edges of the mesh (actually, a bar and joint framework whose members follow the mesh). With the cross-section A_{ij} of members and density ρ_{ij} of the material, the edge $\mathbf{v}_i\mathbf{v}_j$ corresponds to a member of total weight $l_{ij}A_{ij}\rho_{ij}$. We require that in each unsupported vertex $\mathbf{v}_i$, the sum of incoming forces counterbalances the weight of members thought to be lumped in that vertex:

$$\sum_{j:\ \mathbf{v}_i\mathbf{v}_j \in E} w_{ij}(\mathbf{v}_j - \mathbf{v}_i) = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \cdot \sum_{j:\ \mathbf{v}_i\mathbf{v}_j \in E} \frac{1}{2}\rho_{ij}A_{ij}l_{ij}.$$

Remark: By introducing the areas of faces as additional variables (see §3.5) we can model other types of loads besides the members' own weight.

Counting degrees of freedom. Summing up, we have introduced a vector $\mathbf{x} \in \mathbb{R}^{3|V|+3|F|+2|E|}$ of variables. For counting constraints, taking redundancies into account, we consider the average number k of edges per face. Note that $|E| \approx \frac{k}{2}|F|$. There are $|F|$ constraints for normalizing normal vectors, $(k-1)|F|$ for planarity, $|E|$ for edge lengths, and $3|V|$ for equilibrium. This yields $|E| - (k-3)|F| \approx (3 - k/2)|F|$ degrees of freedom. For triangle meshes and quad meshes this number is firmly positive.

General form of equality constraints. The above-mentioned N constraints are quadratic, so there are symmetric matrices H_i, vectors $\mathbf{b}_i$ and constants c_i such that the collection of constraints reads

$$\varphi_i(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T H_i \mathbf{x} + \mathbf{b}_i^T\mathbf{x} + c_i = 0, \quad i = 1, \ldots, N. \qquad (1)$$

The fact that the constraints are quadratic is beneficial in several ways: They are easy to linearize, and it is known exactly how well the linearized constraints fit the original ones. This quadratic nature of constraints is due to the introduction of additional variables $\mathbf{n}_i, l_{ij}, \ldots$, which is a well-known trick in scientific computing.

Weighting of constraints. The data $H_i, \mathbf{b}_i, c_i$ defining a constraint are not unique, but can be multiplied by an arbitrary factor (which is used to attach more or less importance to individual constraints).

Inequality constraints. If the members of a structure follow the edges of a mesh, then the material used for a member $\mathbf{v}_i\mathbf{v}_j$ defines lower and upper bounds $\sigma_{ij}^{\max}, \sigma_{ij}^{\min}$ for the axial stress $\frac{\text{force}}{\text{cross-section}} = w_{ij}l_{ij}/A_{ij}$ experienced by this member. These inequalities are turned into equality constraints by introducing dummy variables:

$$\frac{w_{ij}l_{ij}}{A_{ij}} = \sigma_{ij}^{\min} + \lambda_{ij}^2 \quad \frac{w_{ij}l_{ij}}{A_{ij}} = \sigma_{ij}^{\max} - \mu_{ij}^2.$$

The special cases $\sigma_{ij}^{\max} = 0$, resp., $\sigma_{ij}^{\min} = 0$ correspond to the requirement that stresses are tensile, resp. compressive, and lead to the simplified constraints $w_{ij} = -\mu_{ij}^2$, resp., $w_{ij} = \lambda_{ij}^2$.

Remark: For a comparison between this oversimplified model and finite element analysis, see §4.

Nonlinear constraints. Some constraints cannot be made quadratic, e.g. the condition that vertices of a mesh should lie in a prescribed curve, or should not deviate from a reference surface. Such constraints should be called nonlinear or highly nonlinear. They are handled by geometrically meaningful linearization, cf. §3.3.

3.2 Soft Targets and Energies

In contrast to the constraints of §3.1, we here consider desirable properties of meshes like fairness which are expressed by a low value of a certain energy function. Among the many possible fairness energies we choose the following, which is based on 2nd order differences of mesh polylines and the graph Laplacian. For any vertex $\mathbf{v}$ temporarily denote the cycle of its immediate neighbours by $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n$ and let

$$E_{\mathcal{M},\text{fair}}(\mathbf{v}) = \left\|\mathbf{v} - \frac{\mathbf{w}_1 + \mathbf{w}_3}{2}\right\|^2 + \left\|\mathbf{v} - \frac{\mathbf{w}_2 + \mathbf{w}_4}{2}\right\|^2, \quad (n = 4),$$

$$E_{\mathcal{M},\text{fair}}(\mathbf{v}) = \left\|\mathbf{v} - \frac{\mathbf{w}_1 + \cdots + \mathbf{w}_n}{n}\right\|^2, \quad (n \neq 4).$$

A low value of $E_{\mathcal{M},\text{fair}}$ expresses fairness of the mesh $\mathcal{M}$ near $\mathbf{v}$.

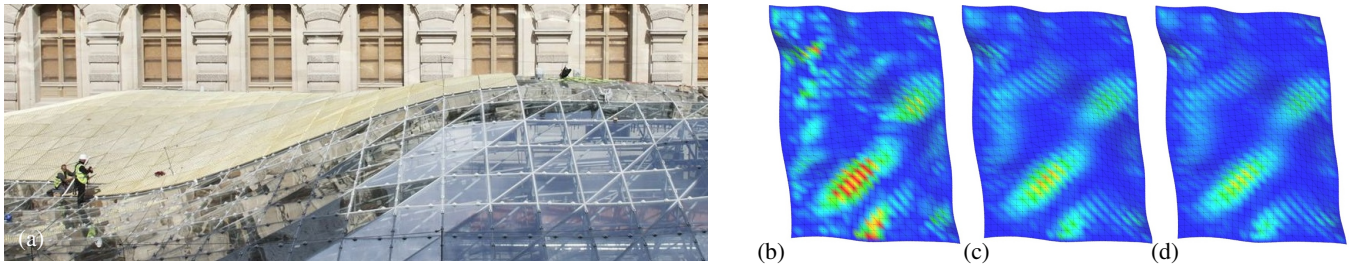


Figure 3: *Fairing under planarity constraints. (a), (b) This hybrid "flying carpet" mesh $\mathcal{M}$ (Louvre, Paris) has high variation of normal vectors (visualized in (b) by color coding the area of the Gauss map of vertex stars). (c), (d) 2, resp. 5 iterations of guided projection onto the constraint space defined by planarity of faces will decreases this variation. We use a fairness energy composed of $E_{\mathcal{M},tang}$ and $E_{\mathcal{M}^*,tang}$.*

Tangential fairness, Gauss image fairness. The curvature of a curve in a surface is decomposed into a normal and a tangential component, where the surface determines most of the normal component and the curve is responsible for the tangential component. This motivates us to consider the *tangential* component $E_{\mathcal{M},\text{tang}}(\mathbf{v})$ of the local fairness energy: It is defined in the same way as $E_{\mathcal{M},\text{fair}}(\mathbf{v})$ but all vertices are first projected onto a tangent plane at the vertex $\mathbf{v}$ (in our computations, that tangent plane is defined by a normal vector at the vertex $\mathbf{v}$ which is estimated and kept fixed during each iteration).

The normal vectors $\mathbf{n}_j$ of faces of the mesh $\mathcal{M}$ are, by definition, the vertices of the Gauss image $\mathcal{M}^*$ of $\mathcal{M}$ (the combinatorics of $\mathcal{M}^*$ are dual to the primal mesh $\mathcal{M}$). Since we have introduced the normal vectors $\mathbf{n}_i$ as variables, expressions $E_{\mathcal{M}^*,\text{tang}}(\mathbf{n}_j)$ are immediately available. By adding them to the total fairness energy, we include fairness of the Gauss image into our computations. This directly translates to a good distribution of curvatures of the primal mesh $\mathcal{M}$. (*Remark:* Meshes $\mathcal{M}, \mathcal{M}^*$ must have comparable scale, otherwise mixing energy terms from both $\mathcal{M}, \mathcal{M}^*$ does not make sense).

Total Energy. The total fairness energy of the mesh is a weighted sum of local contributions $E_{\text{fair}}(\mathbf{v}_i)$, or $E_{\text{tang}}(\mathbf{v}_i)$, or $E_{\text{tang}}(\mathbf{n}_k)$, in all those places where these kinds of fairness are to be applied. In any case the total fairness energy is the sum of squares of expressions which are linear in the vector $\mathbf{x}$ of variables, and which can therefore be written in the form $\mathbf{k}_i^T\mathbf{x} - s_i$. Each is given a weight ε_i (see Fig. 9 for details). With $\mathbf{s} = (\varepsilon_1 s_1, \varepsilon_2 s_2, \dots)^T$ and the matrix K with rows $\varepsilon_i \mathbf{k}_i^T$, we get

$$E_{\text{total}} = \sum\nolimits_i \varepsilon_i^2 (\mathbf{k}_i^T\mathbf{x} - s_i)^2 = \|K\mathbf{x} - \mathbf{s}\|^2. \qquad (2)$$

3.3 Guided Projection Algorithm

The most basic computational task we solve is to modify a given mesh (together with auxiliary variables, encoded in a vector of variables $\mathbf{x}$) such that the prescribed constraints are obeyed. Among the possible solutions of the constraint equations we seek one which has small energy. I.e., we look for $\mathbf{x}$ which solves (1) such that the energy (2) is small. Assuming we are given an initial instance $\mathbf{x} = \mathbf{x}_0$, we successively compute vectors $\mathbf{x}_1, \mathbf{x}_2$ and so on until $\mathbf{x} = \mathbf{x}_n$ accurately enough represents a mesh with the desired properties. This iterative algorithm stops whenever the desired accuracy is reached or whenever there is no more improvement. The examples in this paper needed at most 10 iterations.

Iterative algorithm. One round of iteration in our guided projection procedure works as follows. We are given an instance $\mathbf{x} = \mathbf{x}_n$ of the variables, and we linearize the constraint equations (1) there. Letting $\mathbf{x} = \mathbf{x}_n + \delta\mathbf{x}$ and deleting terms which are quadratic in the increment $\delta\mathbf{x}$ converts (1) into the linear system

$$\varphi_i(\mathbf{x}) \approx \varphi_i(\mathbf{x}_n) + \nabla\varphi_i(\mathbf{x}_n)^T(\mathbf{x} - \mathbf{x}_n) = 0, \quad i = 1, \dots, N.$$

In the notation of (1), the gradient $\nabla\varphi_i$ of φ_i is given by $H_i\mathbf{x}_n + \mathbf{b}_i$. We rewrite the linearized constraint equations in matrix form:

$$H\mathbf{x} = \mathbf{r}, \quad H = \begin{bmatrix} \nabla\varphi_1(\mathbf{x}_n)^T \\ \vdots \\ \nabla\varphi_N(\mathbf{x}_n)^T \end{bmatrix}, \quad \mathbf{r} = \begin{bmatrix} -\varphi_1(\mathbf{x}_n) + \nabla\varphi_1(\mathbf{x}_n)^T\mathbf{x}_n \\ \vdots \\ -\varphi_N(\mathbf{x}_n) + \nabla\varphi_N(\mathbf{x}_n)^T\mathbf{x}_n \end{bmatrix}.$$

This linear system typically is underdetermined, but contains redundant equations which due to inevitable numerical errors lead to a solution space which is too small, badly positioned, and frequently useless for further computations. We therefore do not solve $H\mathbf{x} = \mathbf{r}$ directly. We use the distance from the previous value $\mathbf{x}^*$ and the energy of (2) as a regularizer, and solve

$$\|H\mathbf{x} - \mathbf{r}\|^2 + \|K\mathbf{x} - \mathbf{s}\|^2 + (\varepsilon\|\mathbf{x} - \mathbf{x}^*\|)^2 \to \min \qquad (3)$$

instead. Note that both K and $\mathbf{s}$ are small. We set $\varepsilon = 0.001$ throughout. The solution of (3) is denoted by $\mathbf{x}_{n+1}$ and represents an almost-solution of the linearized constraint equations biased towards a small energy value. It is computed from the linear system $(H^TH + K^TK + \varepsilon^2 I)\mathbf{x} = H^T\mathbf{r} + K^T\mathbf{s} + \varepsilon\mathbf{x}^*$ by Cholesky factorization.

Interpolation and approximation of reference shapes. A typical constraint imposed on a mesh is that certain vertices $\mathbf{v}_i$ are confined to curves C_i (e.g. for the modeling application in Figures 1, 2) or that the mesh is to approximate a reference surface Φ (e.g. for fairing in Fig. 3). Such nonlinear constraints are linearized as follows: The condition $\mathbf{v}_i \in C_i$ is replaced by $\mathbf{v}_i \in T_i^*$, with T_i^* as the tangent of the curve C_i in the point which is closest to $\mathbf{v}_i$. This amounts to 2 linear constraints, which are recomputed in each round of iteration (similarly for Φ, where T_i^* is a tangent plane, yielding 1 constraint).

We use this manner of linearization because if $\mathbf{v}_i$ is close to C_i, then locally $\text{dist}(\mathbf{x}, C_i)^2 \approx \text{dist}(\mathbf{x}, T_i^*)^2$, in the sense of limits of 2nd order Taylor expansions (similar for Φ, [Pottmann et al. 2006]).

Using guided projection in applications. To employ this iterative algorithm for a concrete application, we must find an initial value $\mathbf{x} = \mathbf{x}_0$, which is discussed in §3.4. The iteration itself is a Newton method with a higher-dimensional common zero set of the equations (1). The energy (2) acts as Tikhonov-type regularizer and decides which point of the zero set to converge to. Interpolation

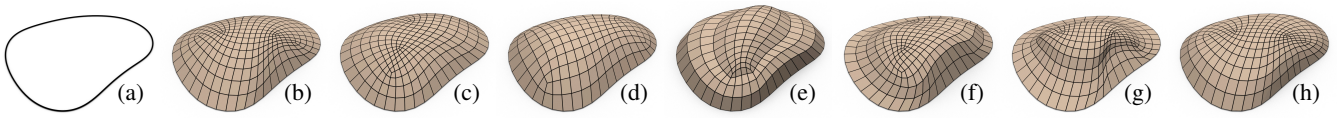


Figure 4: *Exploring the design space and experiencing its limitations: Firstly we observe that polyhedral meshes filling the same boundary (a) and having different combinatorics, generally lead to similar shapes, see (b), (c), (d). User interaction may yield meshes which either have sharp features like (e), or are skewed like (f), or both like (g). Subsequent optimization for equilibrium forces acts like smoothing and reconstructs a dome-like shape apparently largely independent of the combinatorics, see (h).*

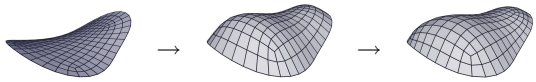
or approximation constraints must be given, otherwise the fairness energy will cause the mesh to shrink.

A particular application is to continually call guided projection during interactive deformation. We describe the details in §4, where we compare our method to [Deng et al. 2014].

3.4 Initialization

Various application scenarios yield different kinds of information on the starting point $\mathbf{x}_0$ employed to initialize the iterative projection algorithm. A simple case is the fairing example of Figure 3, where only vertex coordinates and normal vectors play a role. Here $\mathbf{x}_0$ is known, since the mesh to be faired already is polyhedral. Another typical case, see Fig. 2, is an initial mesh provided by a user. Here, geometry variables are easily initialized (vertices, edge lengths, and estimated face normals). To initialize the force variables w_{ij}, we solve the equilibrium equations for nonnegative values w_{ij}, using the "HLBFGS" procedure of [Liu et al. 2009].

A more complex application scenario is that only a boundary curve C is known, and the mesh combinatorics is part of the design. Here we project C into the xy plane and use e.g. the method of Peng et al. [2014] or Takayama et al. [2013] to fill its interior by a coarse quad mesh, applying subdivision to refine it. A standard relaxation scheme finds a mesh, having that combinatorics, which fits C (left hand figure):



The user now yields information about the desired shape, e.g. by dragging a vertex. We perform a quick auxiliary guided projection to achieve a mesh with this property (center image, using only fairness, proximity to the previous mesh, and user-imposed constraints) and then proceed with whatever guided projection the user has in mind. See Figures 4 and 13 for examples. We in particular refer to the accompanying VIDEO.

3.5 System Extensions

In order not to disrupt the flow of presentation, §3.1 and §3.2 presented only a basic set of variables, constraints, and energies. More features, requiring additional variables and constraints, are discussed here. It is not difficult to extend our system even further.

Offset-relevant constraints The relevance of the *circular* and *conical* properties of meshes are well known: they are related to shape characteristics like principal curvature lines, and they are intimately connected with the existence of constant-distance offsets and support structures [Liu et al. 2006; Pottmann et al. 2007]. Observe that vertices $\mathbf{v}_1, \ldots, \mathbf{v}_n \in f_k$ lie on a circle $\iff$ there is a point $\mathbf{c}_k$ (on the axis of the circle) such that for all i, $\big(\mathbf{v}_{i+1} - \mathbf{v}_i\big)^T\big(\frac{\mathbf{v}_i+\mathbf{v}_{i+1}}{2} - \mathbf{c}_k\big) = 0$ (indices mod n), meaning that any connection between axis and edge midpoint is orthogonal to that edge. Thus, adding variables $\mathbf{c}_k$ and the above-mentioned equations, extends our system to include circularity. Similarly we add conicality which is nothing but circularity of the Gauss image. [SEE VIDEO]

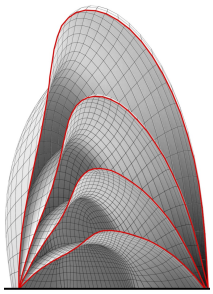


Figure 5: *Vectorial areas can be used to express the signed volume of a mesh over the xy plane (which for closed meshes is the usual volume). The z coordinate of the vectorial area $\mathbf{a}_k$ of an n-gon f_k equals the signed area of its projection onto the xy plane. Multiplying by an average z coordinate of vertices of f_k, and summing over all faces f_k yields an approximate volume. The example at left shows growth of meshes by a user prescribing the total volume.*

Areas and volumes. For various applications a nice distribution of *areas* of faces is important. For that we introduce as new variables the vectorial areas of faces, defined by $\mathbf{a}_k = \frac{1}{2}\sum \mathbf{v}_i \times \mathbf{v}_{i+1}$ (where the sum is over the positive cycle of vertices of f_k). Further new variables are the scalar areas A_k defined by $A_k^2 = \mathbf{a}_k^T\mathbf{a}_k$. In order to set up a fairness energy which favours equidistribution of areas, we introduce the yet undetermined common area of faces as an additional variable A, and add individual contributions $E_{\mathcal{M},\text{area}}(f_k) = (A - A_k)^2$ to the total energy function. See Fig. 6 for a hex mesh produced with and without this term, respectively.

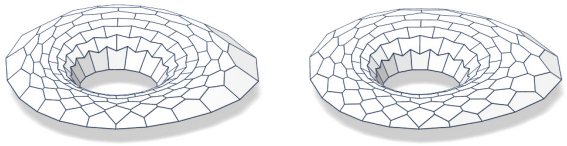


Figure 6: *Effect of computing meshes with "equal area" soft constraints (right) and without (left).*

Since the new relations between variables are quadratic, they fit our framework. Fairing using areas has been used e.g. in the example of Figures 1, 2. Fig. 5 shows how to employ areas to define volumes. Unfortunately global constraints (like total area or volume) lead to equations which involve many variables. This considerably slows down our algorithms, as matrices are no longer sparse.

Force fairness, cost, and further extensions. In engineering, uniform distribution of forces is a desirable property (because then the maximum forces are lower). To achieve this, we define a low-weight constraint expressing equality of forces in opposite edges of every quad (this leads to equations of the type $l_{ij}w_{ij} - l_{kl}w_{kl} = 0$). Interestingly, this "force fairing" has also geometric fairing effects, see Fig. 7a,b.

Many aspects of the *cost* of building a structure can be expressed

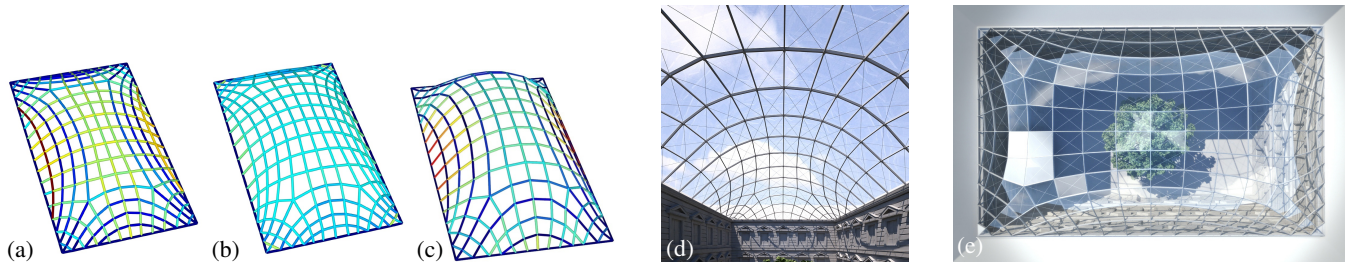


Figure 7: *Architectural design of a roof covering a rectangular courtyard. (a) Optimization of this geometry via guided projection takes planarity of faces into account, as well as fairness and static equilibrium. The magnitude of forces is shown by color coding the edges of the mesh. (b) Similar optimization, with an additional "force fairing" contribution to the total energy. (c) This design encloses more volume than (a) and (b), but with upper bounds on cost, i.e., upper bounds on edge lengths and areas. (d), (e) Different views of an architectural design based on the mesh shown in (a).*

directly in terms of the variables we employ (edge lengths and areas of faces). It is easy to incorporate those aspects of cost into our system, as well as bounds on edge lengths and areas (because reasonably priced panels are available only up to a certain size) and bounds on forces (because otherwise better materials or thicker members have to be used), see Fig. 7c. Since every relation expressible via rational functions can be made quadratic if additional variables are introduced, one can include many more features in our system, see e.g. Fig. 8. This section presented only some of them.

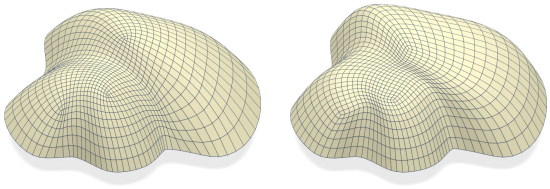


Figure 8: *The right hand mesh differs from the left hand mesh by a contribution to the total energy whose presence tends to make each face a parallelogram (it is a sum of squares of differences of opposite edges in quads). The mesh's combinatorics and a preferred geometry of faces together cause features to emerge.*

3.6 Results Summary

In the course of this paper we have already mentioned some particular applications and results. Fairing under planarity constraints has been tested on a mesh representing freeform architecture which has actually been built (Fig. 3). Our system is generally useful for form-finding (Fig. 4). Geometric modeling with polyhedral meshes and the additional complication of the self-supporting property is the topic of the examples of Figures 1+2, 11 and 12.

A simpler example which nevertheless represents a typical and important application is shown by Fig. 7: fitting a dome-like polyhedral mesh to a boundary with corners (taking statics into account). The side condition of a straight line being contained in a smooth mesh with planar faces rather restricts the possible shapes of that mesh, cf. the discussion in the "limitations" section.

4 Discussion

Implementation Details. Figure 9 gives details on the computation for several of our examples. The timings refer to a Thinkpad T530 with a two-cores Intel® Core™ i5-3320M CPU [2.60GHz and 8G RAM]. Note that guided projection onto the constraint space is fast enough for real-time geometric modeling, SEE VIDEO.

Comparison with related work. The following paragraphs compare our paper with several recent contributions dealing with the geometric modeling of constrained meshes, namely [Deng et al. 2014], [Bouaziz et al. 2012], and [Poranne et al. 2013b].

Comparison of performance. We compare only with these three papers, since they themselves compare with previous work and show that they outperform methods which do not use the trick of introducing auxiliary variables. Fig. 10 gives an account of the convergence behaviour of our method and of both [Deng et al. 2014] and [Bouaziz et al. 2012] when used for "planarization" (i.e, optimizing a mesh with nearly-planar faces such that faces become planar but stay close to the reference geometry).

We performed several other comparisons, including on circular meshes, which yielded similar results. We also implemented [Poranne et al. 2013b], which inspired part of our work. Their alternating approach yielded a much larger deviation (1–2 orders of magnitude) from the original mesh, so no data are included in Fig. 10.

Comparison of interactive modeling capabilities. The accompanying VIDEO shows a side-by-side comparison of our method and the method of [Deng et al. 2014] when both are used for interactive design of a larger mesh with planar faces ($|V| = 3504$).

Fig.	$\lvert V\rvert$	statics?	w_{planar}	w_{force}	$w_{\text{compr.}}$	w_{bdry}	w_{prox}	w_{area}	w_{para}	$\varepsilon_{\text{fair}}$	$\varepsilon_{\text{tang}}$	$\varepsilon_{\text{Gauss}}$	#Iter	T[sec]
2	1726	yes	1	1	1	1	.1	.001		$.9^n$			5	2.8
3c	1766	no	1			1	.1				.002	.002	2	.45
3d	"	"	"			"	"				"	"	5	1.1
8b	1121	no	1			1			.01	.01			5	.4
11	481	yes	1	1	1	1	.1			.01	.05		5	.6
12	521	yes		1	1	1				$.05 \cdot .9^n$	$.05 \cdot .9^n$		10	1.5
13	1056	yes	1	1	1	1					$.05 \cdot .95^n$		10	3.3†

† computation preceded by 0.8 sec "lifting" phase w/o planarity constraints, with boundary interpolation and the constraint that the mesh projects onto the flat mesh it is initialized from [SEE VIDEO].

Figure 9: *Statistics. We give the following data: • mesh size • weights of constraints, namely w_{planar} for planarity of faces, w_{force} for force balance, $w_{\text{compr.}}$ for compressive forces, w_{bdry} for boundary interpolation, w_{prox} for proximity to reference geometry, w_{area} for equal areas, w_{para} for the "parallelogram" constraint of Fig. 8 • weights $\varepsilon_{\text{fair}}$, $\varepsilon_{\text{tang}}$, $\varepsilon_{\text{Gauss}}$ associated with fairness, cf. Equ. (2) • number of iterations • time.*

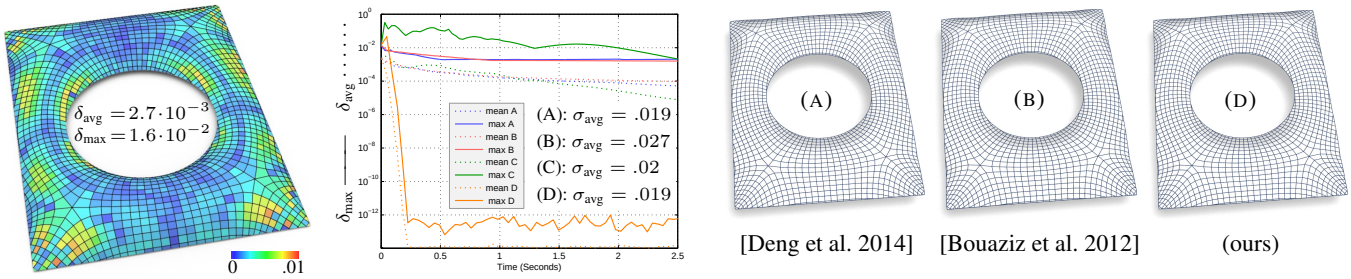


Figure 10: *Different "planarization" methods which optimize the mesh "$\mathcal{M}$" (left) from* [Zadravec et al. 2010] *such that faces become planar. We compare the following methods:* (A) [Deng et al. 2014], (B) [Bouaziz et al. 2012], (C) *our method with cubic constraints, without auxiliary variables;* (D) *our method. The chart illustrates convergence by means of the planarity measure δ and the deviation σ from the reference mesh $\mathcal{M}$: $\delta = \frac{\text{distance of diagonals}}{\text{avg. edge length}}$, $\sigma = \frac{\text{deviation from } \mathcal{M}}{\text{bounding box diameter}}$. The planarity measure is the only significant difference between the respective results, which are almost indistinguishable (see right hand figures).*

Our interactive modeling procedure works as follows: Let the current mesh be described by variables $\mathbf{x}_0$. When a user selects a handle, we linearize all constraints, defining a tangent space T of the constraint manifold $\mathscr{C}$. While the user drags the handle, we apply the smallest deformation in T which lets the mesh follow the handle. When the handle is released, we apply 10 iterations of our projection onto $\mathscr{C}$: weights associated with hard constraints are set to 1, and for regularization we use the following rule of thumb: In the n-th iteration, $1 \leq n \leq 5$, we use mesh fairness with weight $0.2 \cdot 0.9^n$, while for $5 < n \leq 10$ no fairness is employed.

On the other hand, Deng et al. [2014] use a fixed soft energy for fairness of the deformation, and another one for achieving the user's design intent. This causes behaviour different from our method: (i) Switching off fairness will result in meshes which satisfy the constraints but which are not fair. (ii) Starting from a mesh which is not fair usually does not produce fair meshes. The main difference, however, is that we do not try to find a minimum of the fairness energy but use it only for regularization.

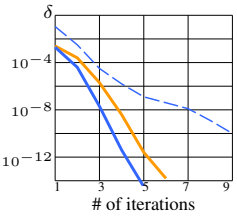
Summing up, the experiments yield an impression similar to Fig. 10: In contrast to related work, our method is capable of satisfying constraints (up to machine precision) quickly.

Work on self-supporting meshes. Self-supporting surfaces which are not height fields could be handled by [Vouga et al. 2012], but require special treatment of vertical face planes. Our method achieves greater accuracy and is faster, owing to their alternating approach and highly nonlinear planarity formulation.

Comparison with local exploration methods. An alternative approach to constrained meshes are methods which locally explore the constraint manifold $\mathscr{C}$ by Taylor expansion [Yang et al. 2011]. They can be extended to include force variables as well, but they solve a different kind of problem. They have a particular limitation, namely being unable to start exploring $\mathscr{C}$ from most flat meshes (which is a feature an interactive tool should provide). The reason for this lies in the fact that the flat meshes constitute a "trivial" linear component of $\mathscr{C}$ [Poranne et al. 2013a]. It is known that a flat mesh is connected to the nontrivial components of $\mathscr{C}$ if and only if it possesses an equilibrium system of forces.

Robustness. The algorithm is not sensitive to parameters (weights of fairness terms). It was not necessary to tune them for better computational performance. The hard constraints form an underdetermined system and can be enforced with high precision, even in the presence of regularizing energies. Unlike e.g. [Deng et al. 2014] we are not aiming at minimizing those energies.

Convergence. We are attributing the good behaviour of our guided projection algorithm to the fact that it is essentially a regularized Newton method (with a higher-dimensional zero set) applied to equations which enjoy only the simplest kind of nonlinearity. Using fairness energies as a regularizer is enough for our purposes. Our experiments exhibit a linear convergence rate: In the inset figure, the orange line is a logarithmic plot of "planarity" δ over the iterations, for the example of Fig. 10. The blue lines show the progress of planarity and circularity (dashed) for another mesh with 384 faces.



Limitations. The limitations of our method are essentially those of a classical Newton method for solving nonlinear equations, which is in widespread and successful use but of course can be made to fail despite the additional regularization we employ. Below we describe limitations of a quite different kind, which are geometric and which would apply to any method used for exploring possible shapes of polyhedral meshes.

- Often architectural designs require straight boundaries. For quad meshes, if that boundary is to be a mesh polyline, then all faces incident with it lie in a common plane. This restriction is apparent in Figure 7.
- Often a designer intends a quad mesh with a mesh polyline in the xy plane, such that transverse edges meet that polyline under right angles (see e.g. Fig. 13). For smooth surfaces, an analogous property would be that its intersection with the xy plane is a principal curvature line (the edges of a polyhedral quad mesh follow a conjugate curve network, which together with orthogonality implies the principal property, cf. [Liu et al. 2006]). Joachimsthal's theorem then says that the surface meets the xy plane under constant slope. This effect can be observed in Fig. 13.

These limitations are consistent with our experiences with local shape space exploration methods (e.g. by Yang et al. [2011]: many variations will be either tangential, or reduce smoothness).

Verification by FE analysis. Our computations do not model an actual structure, but a bar and joint framework with flexible joints. The magnitude of forces in that framework does not allow us to draw conclusions as to the magnitude of stresses in an actual structure (which arises from that framework e.g. by making joints rigid and modeling the edges with beams undergoing bending and torsion). There is the following heuristic, however: equilibrium in the hypothetical framework means that an actual structure, under

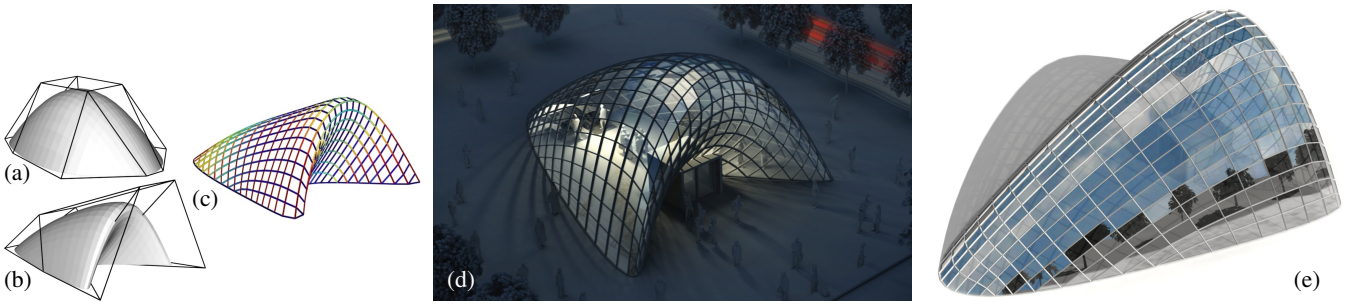


Figure 11: *Like Figure 2, this is an example of a self-supporting polyhedral mesh created by interactive subdivision modeling, coupled with guided projection onto the constraint manifold* [SEE VIDEO]. *(a) initial subdivision surface with control points. (b) final subdivision surface, initializing projection. (c) projected mesh with color-coded magnitude of equilibrium forces in its edges, which is the basis of the architectural design shown in (d) and (e). Note that our method provides perfect user-controlled boundary interpolation: Partly the boundary is a mesh polyline, partly the boundary consists of diagonals.*

its own deadload, is in equilibrium without any moments in the nodes. In order to find out to what extent this heuristic is justified, several of the meshes we produced were submitted to linear static FE analysis, using *Ansys Mechanical* (R14.5). For the example of Figures 1, 2, edges are modeled as beams with thin-walled square cross-sections from S235 structural steel, while the example of Figure 11 was modeled with 20cm circular "Portland cement" concrete beams. Joints were modeled with infinite stiffness. The structure was subsequently subject to its own deadload with fixed boundary. We observe:

- The axial load is of the same order of magnitude as the respective values occurring in our algorithm. The constraint $w_{ij} \geq 0$ is not quite sufficient to ensure compressive forces, since usually some beams carry negligible tension forces.
- Stresses due to bending are of the same magnitude and are superimposed on the axial stresses. The occurrence of tensile stresses is not negligible here.
- Stresses are 2 orders of magnitude less than the recommended maximum, except in the case of tension in concrete, where they are one order of magnitude below. There was no flexural buckling.

We conclude that including equilibrium forces in our algorithm prepares well for subsequent structural analysis, but cannot replace it. This is in line with the goal of statics-aware geometric modeling: avoid infeasible designs, but do not replace structural analysis.

5 Conclusion and Future Work

We have presented a framework for the interactive modeling of polyhedral surfaces or – more generally – meshes constrained by equalities and inequalities. In particular we handle self-supporting meshes and give the user control over both combinatorics and geometry. The framework goes well beyond meshes for architecture (which we used as our main motivation).

On the technical level, we have solved constraint equations for meshes by a two-step process: Preprocessing (i.e., the choice of variables) makes the constraint equations quadratic as far as possible. The actual computation is done by a Newton-type method, numerical problems due to redundant constraints being circumvented by using a fairness energy for regularization.

The main successes of our approach are its speed (it is fast enough to allow for interactive modeling of constrained meshes without GPU implementation), and also the easy way to deal with problems which so far have been considered difficult: handling force equilibrium and force constraints in meshes which are not locally height fields, and manipulating both combinatorics and shape. The authors are not aware of previous work in this area which for instance is able to combine statics with planarity constraints.

Future work. The combination of form, function and fabrication is a wide and open field of research which is not confined to architecture. We are confident that the method of guided projection onto constraint manifolds presented in this paper will find applications completely unrelated to architecture, and even unrelated to meshes.

Acknowledgments. We are grateful to the anonymous referees for their comments. We would particularly like to thank Bailin Deng and Alexandre Kaspar (EPFL) for their extensive help and support concerning comparisons, and Wito Engelke and Marko Tomičić for help with illustrations. This research was supported by the Austrian Science Fund (FWF) via grants P23735-N13 and I706-N26 (DFG-Collaborative Research Center, TRR 109, *Discretization in Geometry and Dynamics*). Chengcheng Tang and Xiang Sun were supported by KAUST base funding, Alexandra Gomes by the Visual Computing Center at KAUST.

References

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, M.I.T.

BOUAZIZ, S., SCHWARTZBURG, Y., WEISE, T., AND PAULY, M. 2012. Shape-up: Shaping discrete geometry with projections. *Comp. Graph. Forum 31*, 1657–1667. Proc. SGP.

DE GOES, F., ALLIEZ, P., OWHADI, H., AND DESBRUN, M. 2013. On the equilibrium of simplicial masonry structures. *ACM Trans. Graph. 32*, 4, #93, 1–10. Proc. SIGGRAPH.

DENG, B., BOUAZIZ, S., DEUSS, M., ZHANG, J., SCHWARTZBURG, Y., AND PAULY, M. 2013. Exploring local modifications for constrained meshes. *Comp. Graph. Forum 32*, 2, 11–20. Proc. Eurographics.

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

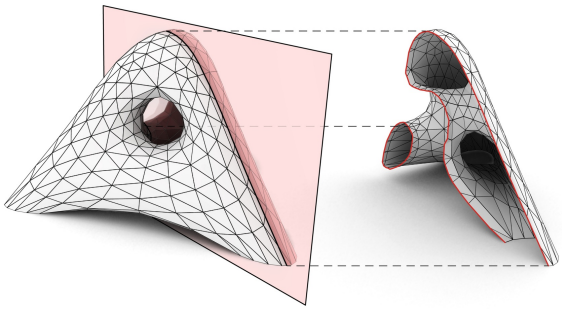


Figure 12: *A triangle mesh which is not a height field and which is self-supporting for a load proportional to the area of faces.*

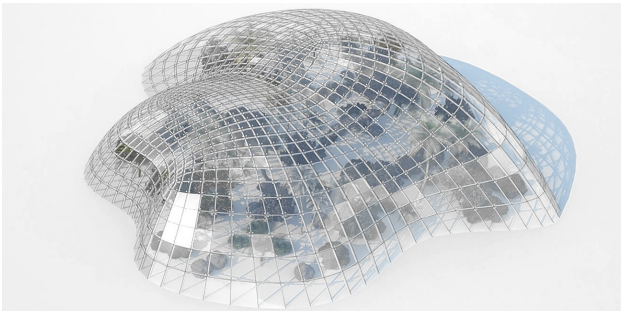


Figure 13: *Architectural freeform design in static equilibrium with planar faces ("greenhouse") by lifting a flat mesh filling out a given boundary.*

FRATERNALI, F. 2010. A thrust network approach to the equilibrium problem of unreinforced masonry vaults via polyhedral stress functions. *Mechanics Res. Comm. 37*, 2, 198 – 204.

GLYMPH, J., SHELDEN, D., CECCATO, C., MUSSEL, J., AND SCHOBER, H. 2004. A parametric strategy for free-form glass structures using quadrilateral planar facets. *Automation in Construction 13*, 2, 187 – 202.

GRÜNDIG, L., MONCRIEFF, E., SINGER, P., AND STRÖBEL, D. 2000. A history of the principal developments and applications of the force density method in Germany 1970–1999. In *4th Int. Coll. Computation of Shell & Spatial Structures*.

HEYMAN, J. 1998. *Structural Analysis: A Historical Approach*. Cambridge University Press.

KASPAR, A., AND DENG, B. 2013. Realtime deformation of constrained meshes using GPU. In *Symposium on GPU Computing and Applications*. to appear.

KOTNIK, T., AND WEINSTOCK, M. 2012. Material, form and force. *Architectural Design 82*, 104–111.

LINKWITZ, K., AND SCHEK, H.-J. 1971. Einige Bemerkungen zur Berechnung von vorgespannten Seilnetzkonstruktionen. *Ingenieur-Archiv 40*, 145–158.

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. Proc. SIGGRAPH.

LIU, Y., WANG, W., LÉVY, B., SUN, F., YAN, D.-M., LU, L., AND YANG, C. 2009. On centroidal Voronoi tessellation – energy smoothness and fast computation. *ACM Trans. Graph. 28*, 4, #101, 1–17.

LIU, Y., XU, W., WANG, J., ZHU, L., GUO, B., CHEN, F., AND WANG, G. 2011. General planar quadrilateral mesh design using conjugate direction field. *ACM Trans. Graph. 30*, 6, #140, 1–10. Proc. SIGGRAPH Asia.

LIU, Y., PAN, H., SNYDER, J., WANG, W., AND GUO, B. 2013. Computing self-supporting surfaces by regular triangulation. *ACM Trans. Graph. 32*, 4, #92, 1–10. Proc. SIGGRAPH.

PANOZZO, D., BLOCK, P., AND SORKINE-HORNUNG, O. 2013. Designing unreinforced masonry models. *ACM Trans. Graph. 32*, 4, #91, 1–12. Proc. SIGGRAPH.

PENG, C.-H., BARTON, M., JIANG, C., AND WONKA, P. 2014. Exploring quadrangulations. *ACM Trans. Graph. 33*, 1, #12,1–12.

PORANNE, R., CHEN, R., AND GOTSMAN, C., 2013. On linear spaces of polyhedral meshes. ArXiv:1303.4110.

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

POTTMANN, H., HUANG, Q.-X., YANG, Y.-L., AND HU, S.-M. 2006. Geometry and convergence analysis of algorithms for registration of 3D shapes. *Int. J. Computer Vision 67*, 3, 277–296.

POTTMANN, H., LIU, Y., WALLNER, J., BOBENKO, A., AND WANG, W. 2007. Geometry of multi-layer freeform structures for architecture. *ACM Trans. Graph. 26*, 3, #65,1–11. Proc. SIGGRAPH.

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

SCHIFTNER, A., LEDUC, N., BOMPAS, P., BALDASSINI, N., AND EIGENSATZ, M. 2012. Architectural geometry from research to practice — the Eiffel Tower Pavilions. In *Advances in Architectural Geometry 2012*. Springer, 213–228.

TAKAYAMA, K., PANOZZO, D., SORKINE-HORNUNG, A., AND SORKINE-HORNUNG, O. 2013. Sketch-based generation and editing of quad meshes. *ACM Trans. Graph. 32*, 4, # 97,1–8.

UMETANI, N., IGARASHI, T., AND MITRA, N. J. 2012. Guided exploration of physically valid shapes for furniture design. *ACM Trans. Graph. 31*, 4, #86, 1–11. Proc. SIGGRAPH.

VAXMAN, A. 2012. Modeling polyhedral meshes with affine maps. *Comp. Graph.. Forum 31*, 1647–1656. Proc. SGP.

VOUGA, E., HÖBINGER, M., WALLNER, J., AND POTTMANN, H. 2012. Design of self-supporting surfaces. *ACM Trans. Graph. 31*, 4, #87, 1–11. Proc. SIGGRAPH.

YANG, Y., YANG, Y., POTTMANN, H., AND MITRA, N. 2011. Shape space exploration of constrained meshes. *ACM Trans. Graph. 30*, 6, #124,1–11. Proc. SIGGRAPH Asia.

ZADRAVEC, M., SCHIFTNER, A., AND WALLNER, J. 2010. Designing quad-dominant meshes with planar faces. *Comp. Graph. Forum 29*, 5, 1671–1679. Proc. SGP.