Designing Unreinforced Masonry Models

Daniele Panozzo Philippe Block Olga Sorkine-Hornung

ETH Zurich

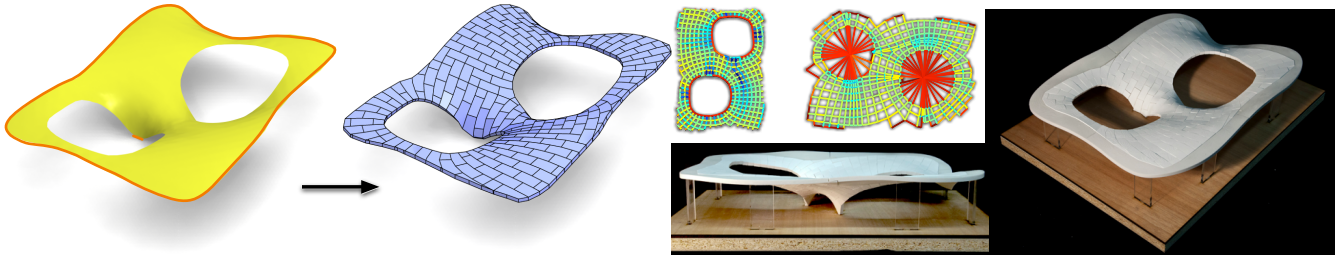

Figure 1: *An input surface is automatically transformed into a masonry 3D model using our algorithm. The equilibrium of the surface is represented by two planar graphs that encode the directions and magnitudes of all forces. The generated blocks are 3D-printed and assembled into a physical model of the surface that stands in compression without using glue or reinforcements.*

Abstract

We present a complete design pipeline that allows non-expert users to design and analyze masonry structures without any structural knowledge. We optimize the force layouts both geometrically and topologically, finding a self-supported structure that is as close as possible to a given target surface. The generated structures are tessellated into hexagonal blocks with a pattern that prevents sliding failure. The models can be used in physically plausible virtual environments or 3D printed and assembled without reinforcements.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling;

Keywords: masonry models, thrust network analysis, self-supporting surfaces, optimization, field alignment, tessellation

Links: DL PDF WEB VIDEO

1 Introduction

Most of the world's architectural heritage consists of buildings in unreinforced masonry. They are made of unsupported bricks, or stone blocks called voussoirs, and they stand thanks to their specific structural form and thickness. No supporting framework is needed, since the entire structure is in a static equilibrium configuration where all the forces are compressing the bricks. Self-supporting structures have applications in architecture and physically plausible virtual environments [Whiting et al. 2009], and their study is intriguing from a discrete differential geometry viewpoint [Vouga et al. 2012].

ACM Reference Format
Panozzo, D., Block, P., Sorkine-Hornung, O. 2013. Designing Unreinforced Masonry Models. ACM Trans. Graph. 32, 4, Article 91 (July 2013), 11 pages. DOI = 10.1145/2461912.2461958 http://doi.acm.org/10.1145/2461912.2461958.

DOI: http://doi.acm.org/10.1145/2461912.2461958

The design of such structures is a challenging task that requires deep structural knowledge. Optimization tools to assist the design have been recently proposed, but they still rely on manual input. The foundation for these methods is the Thrust Network Analysis [Block 2009], a computational framework that allows to understand the static equilibrium of existing masonry structures and be directly used as a design tool [Rippmann et al. 2012]. The core idea is to reduce the dimensionality of the problem by first finding a pair of planar graphs that describe the horizontal equilibrium of the 3D shape, and then optimizing only for the height. The "Safe Theorem" states that if a system of forces that is in static equilibrium with the loads exists, and this system is completely enclosed in the masonry structure, then the structure will stand without the need for additional reinforcements [Heyman 1995].

In this work, we present an algorithm that transforms an input height field, generated with any standard modeling software, into a masonry model consisting of hexagonal blocks. We compute a model whose shape is close to the input surface and which is able to stand in compression, i.e. without requiring any additional reinforcements or "glue". The masonry model can be used in a destructible virtual environment, allowing physically plausible interactions with it, and it can also be 3D-printed and assembled (Figure 1). The design process does not require structural knowledge, since our algorithm is automatic: the user just needs to provide an input shape, to which our method fits a self-supporting surface.

This paper makes the following contributions:

1. We observe that the quality of the discrete force pattern is fundamental to the analysis of the static equilibrium (see Figure 4). We propose a small set of structurally-informed heuristics to estimate the force flow in a masonry structure based on its geometry, and apply these heuristics to guide a field-aligned remeshing [Bommes et al. 2009] to generate high-quality force patterns. Our method handles structures with unsupported edges, sharp creases and negative Gaussian curvature sections.
2. We derive a bottom-up approach to generate a best-fitting self-supporting surface that is as close as possible to an input target shape. We iteratively deform an automatically generated self-supporting surface to approximate the target.
3. We partition the self-supporting surface into blocks that are directly suitable for physical simulation and realization via 3D printing. The tessellation into blocks is aligned with the force flow and has a staggered pattern to avoid sliding failure.

We demonstrate several results of fitting a self-supporting masonry model to challenging freeform shapes with unsupported boundaries. We validate our algorithm by 3D-printing and assembling two models; their construction and destruction sequences are shown in the accompanying video.

2 Related work

Equilibrium analysis of masonry. Unreinforced masonry structures generally fail not due to lack of compressive strength, but due to instability [Heyman 1995]. Understanding the equilibrium of structures in masonry is thus of primary concern. Heyman introduced the limit analysis framework to unreinforced masonry structures. For spatial structures, thrust network analysis can be used to assess the stability of masonry [O'Dwyer 1999; Block 2009; Fraternali 2010]. In computer graphics, an alternative equilibrium approach to masonry based on rigid body mechanics was proposed by [Whiting et al. 2009; Whiting et al. 2012]. The most relevant work for our approach is [Vouga et al. 2012]; we discuss and contrast it with ours in Section 5.3.

Flow of forces. An important issue of the discretized networks is the choice of their topology, as it is clear that the equilibrium solutions heavily depend on it [O'Dwyer 1999; Kilian and Ochsendorf 2005]. For masonry, there has been extensive debate about the structural behavior of masonry vaults (see [Block and Lachauer 2012] for an overview). For the best assessment of the stability of the structure, the chosen connectivity should represent the "flow of forces" in it. Many scholars have assumed that the vault forces flow to the supports in the same manner as water would drain off the vault's upper surface, i.e. following lines of steepest descent [Borgart 2005]. However, this assumption does not explain the equilibrium of vaults with unsupported boundary edges, because such edges cannot transfer thrust, while a curvature analysis of the vaults geometry could result in thrusts hitting these edges. Nonetheless, this approach is popular in state-of-the-art architectural practice, e.g. to produce ribs patterns for shells [Bhooshan and El Sayed 2011], because it is easy to implement, and no alternative exists. In this work, we use a few structural heuristics to estimate the flow of forces in an input geometry, which specifically address unsupported edges.

Heyman [1995] explains that Gothic vaults act as thin shells with stress concentrations along creases, as there is a direct relation between curvature and membrane stresses in shells [Calladine 1983]. But, shell equations are very hard to solve in the general case, and become infeasible to solve for e.g. discontinuities in boundary conditions. This was addressed for the specific case of vaults under vertical loading and supported only at their corners by [Williams 1990]. Others have used linear elastic analysis on flat plates for given boundary conditions to produce force patterns from the principle stress directions [Borgart 2005], or photo-elastic analysis of 3D plastic models [Mark 1982]. The issue with these approaches is that they count on tensile capacity, which significantly influences the resulting force equilibria, and can thus not serve to postulate good force patterns for masonry shells that act in compression only. By contrast, our algorithm relies solely on the structurally-informed geometric analysis of the shape to generate self-supporting surfaces.

Architectural geometry. The topic of architectural geometry is receiving increasing attention in computer graphics and geometry processing. For example, several works consider rationalization of freeform surfaces via PQ meshes (see e.g. [Liu et al. 2006; Bouaziz et al. 2012]); a special type of statics-sensitive PQ meshes is discussed in [Schiftner and Balzer 2010; Vouga et al. 2012]. Rationalization via reducing the number of uniquely-shaped tessellation blocks was presented in [Eigensatz et al. 2010; Fu et al. 2010; Singh and Schaefer 2010]. While rationalization is outside the scope of this paper, the above approaches can be potentially used in combination with our method to create self-supporting models that are cost-effective to produce in practice. A self-supporting structure can also be constructed using folding elements instead of unreinforced masonry [Zimmer et al. 2012].

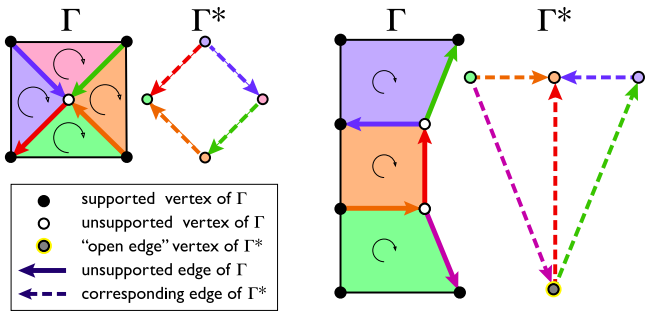

Figure 2: *Each face of the planar form diagram* Γ *is a vertex of the reciprocal* Γ^* *(marked in the same color in the illustration). Each edge of* Γ *has a corresponding, parallel edge in* Γ^* *(depicted in the same color). An additional vertex (bottom right) is added to* Γ^* *for every chain of open edges in* Γ.

3 Preliminaries and notation

The input to our algorithm is a target surface, described by a height function $\mathcal{H} = \mathcal{H}(x, y)$ defined over some planar domain $\Omega \subset \mathbb{R}^2$, and a set of supported points $\mathcal{B} \subset \Omega$. We assume that gravity is in the negative z direction, and we wish to find a masonry structure whose shape is as close as possible to $\mathcal{H}$ and which stands under its own weight. The target shape $\mathcal{H}$ can be a piecewise linear surface (a mesh) or a freeform surface, designed with any CAD modeling software. We assume that the structure will be attached to the ground surface or a pillar at every supported point. Our goal is (*i*) to compute a (discrete) surface $\mathcal{S}$ whose shape is as close as possible to $\mathcal{H}$, and which stands in compression while being supported at the specified locations $(x, y, \mathcal{H}(x, y)),\ (x, y) \in \mathcal{B}$; (*ii*) create a tessellation (remeshing) of $\mathcal{S}$, such that all the facets can be turned into bricks with which the masonry structure can be physically realized or simulated in a virtual environment.

Thrust Network Analysis. In order for a surface to stand in compression, all its unsupported vertices must be in static equilibrium with the applied loading. We use Thrust Network Analysis (TNA) [Block 2009] to characterize the equilibrium conditions and we derive a computational technique to find a self-supporting surface that closely fits the input $\mathcal{H}$. We briefly review the basic notions of TNA here for completeness.

The continuous system of forces inside a masonry structure is discretized by a graph $\mathcal{S} = \{\mathcal{V}, \mathcal{E}, \mathcal{F}\}$, whose vertices are embedded in $\mathbb{R}^3$ and whose edges represent directions of compression forces acting on the vertices. If it is possible to find a positive magnitude for all the compression forces such that the resultant acting on every vertex (together with the load) vanishes, then $\mathcal{S}$ represents a state of static equilibrium. If $\mathcal{S}$ fits inside the masonry shell, the Safe Theorem guarantees the stability of the shell for the given loading case, although the actual force distribution might end up being different than the one represented by $\mathcal{S}$. For further discussion, see [Heyman 1995; Block 2009; Vouga et al. 2012].

The loads due to gravity are discretized as vertical forces acting on every vertex $\mathbf{v}_i$, corresponding to the weight of the material in the Voronoi cell of $\mathbf{v}_i$. Assuming a uniform thickness of the masonry

structure (thickness being measured in the surface normal direction), the loads are $\mathbf{p}_i = (0, 0, -\rho A_i)$, where A_i is the Voronoi area on the surface, and ρ is the density of the material multiplied by the uniform thickness value.

Denote by $\mathcal{V}_\mathcal{B}$ the set of supported vertices, that is, all vertices $\mathbf{v}_i = (x_i, y_i, z_i)$ such that $(x_i, y_i) \in \mathcal{B}$; for these vertices we fix $z_i = \mathcal{H}(x_i, y_i)$. Denote by $\mathcal{V}_\mathcal{U} = \mathcal{V} \setminus \mathcal{V}_\mathcal{B}$ the unsupported vertices. In general, we use subscripts $\mathcal{B}, \mathcal{U}$ to refer to supported and unsupported entities (vertices, edges, etc.), respectively. We call an edge supported if both its vertices are in $\mathcal{V}_\mathcal{B}$, and we call edges with at least one vertex in $\mathcal{V}_\mathcal{U}$ unsupported. Unsupported boundary edges are also termed *open* edges.

$\mathcal{S}$ is self-supporting if the forces cancel out at every unsupported vertex and all the forces are in compression. Formally, the following equilibrium equations must be satisfied for each $i \in \mathcal{V}_\mathcal{U}$:

$$\sum_{j \in \mathcal{N}_i} w_{ij}(\mathbf{v}_j - \mathbf{v}_i) = \mathbf{p}_i \tag{1}$$

$$w_{ij} \geq 0 \tag{2}$$

where $\mathcal{N}_i$ is the set of neighbors of vertex i and w_{ij} is a symmetric weight ($w_{ij} = w_{ji}$) that scales the magnitude of the forces associated with every edge. Equation (2) indicates that all forces are in compression.

In TNA, instead of directly enforcing the entire system of equations above, the problem is factored into horizontal and vertical equilibriums, making it simpler to solve and providing a *reciprocal diagram*, that is a graphical representation of the static equilibrium.

Form and reciprocal diagrams. Consider first the horizontal equilibrium of $\mathcal{S}$. Let Γ be the projection of $\mathcal{S}$ onto the horizontal plane. Denote Γ's edge vectors by $\mathbf{e}_{ij} = (x_j - x_i, y_j - y_i, 0)$, where $(i, j) \in \mathcal{E}_\mathcal{U}$ (we do not include any supported edges in Γ). Then from Equation (1) we have:

$$\forall i \in \mathcal{V}_\mathcal{U}, \sum_{j \in \mathcal{N}_i} w_{ij}\mathbf{e}_{ij} = \mathbf{0}, w_{ij} \geq 0. \tag{3}$$

This equation states that the vectors $w_{ij}\mathbf{e}_{ij}$ form a convex, *closed* loop, if laid out as a consecutive chain in the plane. This implies that if Equation (3) is satisfied for every unsupported vertex, then a *reciprocal* planar graph Γ^* exists. Γ^* has the combinatorics of the dual graph of Γ: it has a vertex corresponding to every face of Γ, and it has an edge $\mathbf{e}^*$ corresponding to every (unsupported) edge $\mathbf{e}$ of Γ. For every pair of incident faces in Γ, there is an edge between the corresponding vertices in Γ^*. Additionally, for every consecutive chain of *open* edges in Γ, Γ^* has a corresponding vertex, to which all the corresponding dual edges are connected. Refer to Figure 2 for an illustration.

Equation (3) implies that Γ^* has a valid planar embedding, such that for each edge $\mathbf{e}^* = w_{ij}\mathbf{e}$. And conversely:

Lemma 1. *If two planar graphs Γ and Γ^* (with connectivity correspondence as described above) exist, such that every edge $\mathbf{e} \in \Gamma$ is parallel to the corresponding edge $\mathbf{e}^* \in \Gamma^*$, then Equation (3) is satisfied with $w_{ij} = \|\mathbf{e}^*_{ij}\|/\|\mathbf{e}_{ij}\|$.*

From now on, we will call Γ the form diagram, Γ^* the reciprocal diagram, and w_{ij} the force densities.

Vertical equilibrium. A form diagram Γ and its reciprocal Γ^* characterize a self-supporting surface which can be computed by solving the vertical equilibrium in Equation (1):

$$\forall i \in \mathcal{V}_\mathcal{U}, \sum_{j \in \mathcal{N}_i} \frac{\|\mathbf{e}^*_{ij}\|}{\|\mathbf{e}_{ij}\|}(z_j - z_i) = -\rho A_i \tag{4}$$

$$s.t.\ \forall i \in \mathcal{V}_\mathcal{B},\ \ z_i = \mathcal{H}(x_i, y_i), \tag{5}$$

where z_i are the z-coordinates of the vertices of $\mathcal{S}$. Note that by uniformly scaling the reciprocal diagram, or equivalently, scaling the force densities w_{ij}, we generate an infinite family of self-supporting surfaces (see Figure 3).

Equilibrium equations in matrix form. To simplify the notation, we represent Γ by a pair of matrices $\mathbf{V}$ and $\mathbf{C}$. The matrix $\mathbf{V} = [\mathbf{x}\ \mathbf{y}]$ is a $|\mathcal{V}| \times 2$ matrix, containing the coordinates of Γ's i-th vertex in the i-th row. The matrix $\mathbf{C}$ is a $|\mathcal{E}_\mathcal{U}| \times |\mathcal{V}|$ matrix encoding the connectivity of Γ such that $\mathbf{CV}$ is the matrix of Γ's edge vectors as rows. The e-th row represents the e-th edge of Γ:

$$\mathbf{C}_{e,k} = \begin{cases} 1 & e \in \mathcal{E}_\mathcal{U},\ e = (i, j),\ k = i, \\ -1 & e \in \mathcal{E}_\mathcal{U},\ e = (i, j),\ k = j, \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Note that we assign arbitrary directions to Γ's edges, such that each $(i, j) \in \mathcal{E}_\mathcal{U}$ is an ordered pair (refer to Figure 2). We just need to take care that the reciprocal edges in Γ^* are ordered consistently with this assignment. We assign an orientation to each edge $\mathbf{e}^*$ of Γ^* as follows: if its corresponding edge $(i, j) \in \Gamma$ is incident on faces f_1 and f_2, and w.l.o.g. the planar orientation on f_1 is such that a positive cyclic walk along f_1's faces runs in the same direction as the ordering (i, j), then $\mathbf{e}^*$ will have the ordering (f_1, f_2); otherwise it will be (f_2, f_1) (see Figure 2).

With this notation at hand, we can compactly express the equilibrium equations (1):

$$\mathbf{C}^T_{\mathcal{V}_\mathcal{U}}\mathbf{D}_\mathbf{w}\mathbf{C}\,[\mathbf{x}\ \mathbf{y}\ \mathbf{z}] = [\mathbf{0}\ \mathbf{0}\ \mathbf{p}] \tag{7}$$

$$s.t.\ \ \mathbf{z}_\mathcal{B} = \mathcal{H}_\mathcal{B},$$

where $\mathbf{D}_\mathbf{w}$ is a diagonal matrix with the force densities w_{ij} in the same order as the edges appear in $\mathbf{C}$, $\mathbf{C}_{\mathcal{V}_\mathcal{U}}$ is the matrix $\mathbf{C}$ where all the columns corresponding to the supported vertices $\mathcal{V}_\mathcal{B}$ were removed, and $\mathbf{p}$ is a column vector with the stacked vertical components of the loads $\mathbf{p}_i$.

3.1 Overview

We use the elements of TNA described above, namely the representation of the horizontal equilibrium with the form and reciprocal diagrams, and the vertical equilibrium equations, to guide the optimization of a self-supporting surface to fit a given shape. Our only requirement on the input shape $\mathcal{H}$ is that it must be a height field. Our approach is divided into three consecutive steps. First, in Section 4, we introduce a set of structural heuristics to estimate the flow of forces on the target surface. Using this estimated flow, we generate a structurally informed form diagram Γ by employing a field-aligned

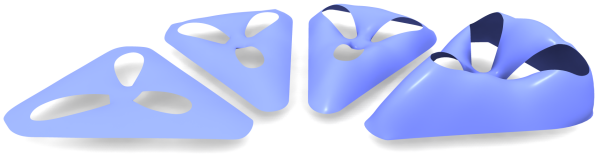

Figure 3: *Different surfaces generated using a fixed Γ and a uniformly scaled Γ^* (scaling values $2, 1, 0.5,$ and 0.02, left to right).*

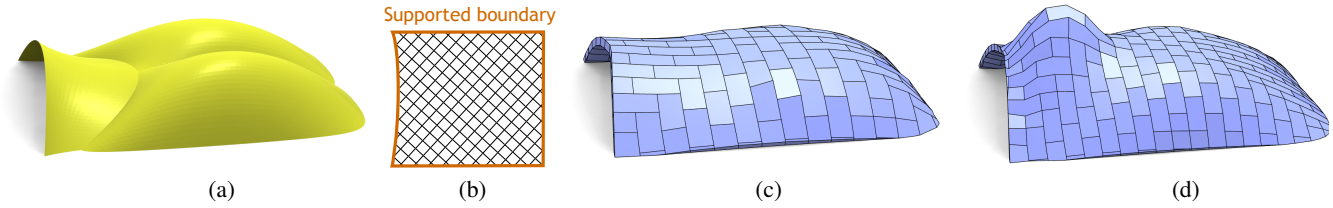

Figure 4: *A non-optimized form diagram (b) cannot describe the equilibrium of the features of this simple shape (a). Our algorithm computes a least-squares solution, removing most of the features (c). The method of [Vouga et al. 2012] does not change the parts of the surface that are locally in equilibrium (the bumps are partially visible on the right), but it introduces an additional feature to make the structure self-supporting (d). In Figure 6, an optimized pattern is used for the same target shape. The Hausdorff distance between the input surface and the computed self-supporting shape is 0.042 for our fit and 0.150 for [Vouga et al. 2012]'s (normalized w.r.t. the bounding box diagonal).*

remeshing technique. This step determines the connectivity of Γ and provides a good initial guess of its geometry. In the second step (Section 5), we synchronously optimize the geometry of Γ and Γ^* to generate a self-supporting surface that is as close as possible to the input shape $\mathcal{H}$. Finally, in Section 6, we describe a structurally informed algorithm to tessellate the generated self-supporting surface $\mathcal{S}$ into hexagonal blocks, such that the compression forces and the friction between the blocks would prevent sliding failure. The extruded blocks can then be put together and the structure stands without using any "glue" between the blocks.

4 Field-Aligned Form Diagram Optimization

The form diagram Γ is a discretization of the force flow of the surface, where forces are aligned with the edges and act on the vertices of Γ. The geometry and connectivity of Γ play a crucial role in the generation of a self-supporting structure. Γ has been manually generated for each model in previous works [Block 2009; Vouga et al. 2012].

Interestingly, increasing the resolution of the form diagram does not directly improve the quality of the discretization; the most important aspect of the form diagram is that its edges represent the directions of the forces acting on a vertex, and thus a good discretization should represent as many directions as possible. This is however impossible for triangle or quad meshes, independently of the density, since the the average valence of the vertices is 6 and 4, respectively. Optimizing the directions of the edges is thus fundamental to generate good discretizations that will lead to high quality best-fitting surfaces. A randomly oriented pattern arbitrary limits the directions of the forces, restricting the expressiveness of the equilibrium model to specific shapes (predominantly with positive Gaussian curvature), as shown in Figure 4.

We propose the first automatic, structurally informed approach to generate the connectivity and geometry of the form diagram by combining a set of structural heuristics with a field-aligned remeshing algorithm [Ray et al. 2008; Bommes et al. 2009]. While we fix the connectivity for the remaining steps of our automatic pipeline, the automatically-generated Γ could also be used as a starting point for existing form-finding tools [Rippmann et al. 2012], where users manually design the form diagram.

4.1 Heuristics for force flow estimation

We analyze the input geometry $\mathcal{H}$ to extract features that require a specific flow of forces to be represented. We observed expert designers use the form finding tool RhinoVault [Rippmann et al. 2012], and with their help we discovered three important rules that are followed to generate good form diagrams. Generally, the form-finding process requires a lot of experience, and the designers cannot always describe why some specific choices are made; however, we found that our three heuristics are sufficient for a wide range of interesting shapes, as shown in Figures 1, 2, 4, 6, 7, 9, 13-15.

(1) Open Edges are unsupported parts of the boundary of the surface; they require special treatment in the form diagram, since the load associated with a vertex on an open edge must be redirected onto the open edge itself, or into the interior of the surface. While for a supported boundary vertex the direction of the incoming forces is irrelevant, for a vertex on an open edge it should be parallel to the direction of the open edge, to allow forces to flow to the support.

(2) Anticlastic Parts (parts with negative Gaussian curvature) are hard to realize in a self-supporting surface because the forces coming from the direction of maximal curvature can only be compensated by bigger forces approximately parallel to the orthogonal direction (see Figure 5). The transition between these regions and the rest of the surface should be smooth, to allow the force to flow until a supporting vertex is reached. An example of an anticlastic part and an appropriate force pattern for it is shown in Figure 6. Note that sections with positive Gaussian curvature do not require any special treatment since all the incident forces can be used to compensate for the weight (Figure 5).

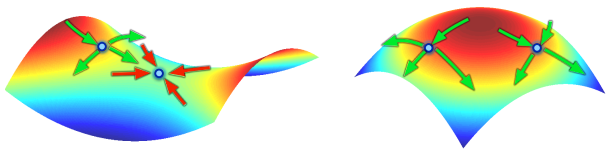

Figure 5: *In anticlastic regions (left), a wrong alignment of the force pattern does not allow to represent the equilibrium of a vertex (red), since all directions of the force flow are pointing downward. This does not happen if the Gaussian curvature is positive (right).*

(3) Sharp Features can appear in self-supporting surfaces (cf. gothic masonry), and they require a chain of edges covering the entire feature in the form diagram in order to properly represent it.

In Figure 6, we show a didactic example where all the heuristics are used to generate the form diagram and the corresponding best-fit surface computed with the algorithm presented in Section 5.

4.2 Cross Field Generation and Quadrangulation

We transform the heuristics above into directional constraints for the generation of a smooth cross field on Ω. The cross field will then guide a remeshing algorithm to generate an initial guess for Γ (both connectivity and geometry). We take this approach because our directional constraints can be naturally represented by a 4-RoSy (cross) field.

We start by densely meshing Ω with a constrained Delaunay triangle mesh $\mathcal{M}$ (we use the Triangle software [Shewchuk 1996] and ask for at least 10K samples). We then use the Mixed-Integer Quadrangulation algorithm (MIQ) [Bommes et al. 2009] to compute a cross field on $\mathcal{M}$ according to our three heuristics, and convert it into a quad mesh.

MIQ computes the cross field by minimizing an energy that asks for a *smooth* field. A discrete cross field is represented in each triangle $\mathbf{t}_i \in \mathcal{M}$ by an angle θ_i with respect to a local frame [Ray et al. 2008]. Smoothness of the field means that the fields of each pair of adjacent triangles $\mathbf{t}_i, \mathbf{t}_j$ should not be too different. However, we cannot just measure the difference $|\theta_i - \theta_j|$ because this angle representation is ambiguous: the angle in every triangle is unique only up to a rotation by multiples of $2\pi/4$, and also the local frames of different triangles might be different. Denote by κ_{ij} the (fixed) rotation angle between the local frames of $\mathbf{t}_i$ and $\mathbf{t}_j$, and let p_{ij} be an integer variable, called the *period jump*. Now the field in triangle $\mathbf{t}_i$ can be expressed relatively to a neighboring triangle $\mathbf{t}_j$ as $\theta_i + \kappa_{ij} + 2\pi p_{ij}/4$.

We minimize the following specific variant of the cross field energy, proposed by [Panozzo et al. 2012], where directional constraints can be soft or hard:

$$E_{\text{field}}(\theta, p) = (1-\alpha) \sum_{(i,j)\in\mathcal{M}} (\theta_i + \kappa_{ij} + \frac{2\pi}{4} p_{ij} - \theta_j)^2 + \quad (8)$$

$$+ \alpha \sum_{\mathbf{t}_i \in \mathcal{T}_{\text{soft}}} \omega_i (\theta_i - \bar{\theta}_i)^2, \qquad [\textit{soft constraints}] \quad (9)$$

$$s.t.\ \forall \mathbf{t}_k \in \mathcal{T}_{hard},\ \ \theta_k = \bar{\theta}_k. \quad [\textit{hard constraints}] \quad (10)$$

Here, $\mathcal{T}_{\text{soft}}$ and $\mathcal{T}_{\text{hard}}$ denote the sets of triangles where the field is soft- and hard-constrained, respectively, and α, ω_i control the strength of the soft constraints. This is a mixed-integer problem, since the θ_i's are real and the p_{ij}'s are integer. We employ the greedy mixed integer solver of [Bommes et al. 2009]. To make the minimizer unique, the variables p_{ij} should be fixed at a subset of edges [Bommes et al. 2009].

Our heuristics (Section 4.1) are translated into constraints as follows:

1. Open edges are hard constraints that align the field with the boundary edges. Hence all triangles $\mathbf{t}_i$ that have a boundary edge are added to $\mathcal{T}_{\text{hard}}$, with the constraint $\bar{\theta}_i$ being the direction of the boundary edge.
2. Triangles in anticlastic parts are added to $\mathcal{T}_{\text{soft}}$, and the field is soft-constrained to align to the direction of minimal curvature there. The curvature directions are computed with [Cazals and Pouget 2003].
3. The field in triangles containing a sharp feature is hard-constrained to align with the feature. We detect features with a fixed threshold of 10 degrees on the dihedral angle, but a more advanced method like [Hildebrandt et al. 2005] could be used.

Note that for our purposes, we could also use the field formulation of Crane et al. [2010], but we prefer that of Ray et al. [2008], as it allows to impose our directional constraints in a more natural way. In the experiments showed in this paper, we used $\alpha = 0.5$ and a fixed value $\omega_i = 0.05$ for the soft constraints.

Once the field is computed, we follow MIQ [Bommes et al. 2009] to generate a mesh that is as aligned as possible with the cross field. The resolution of the mesh is controlled by the minimal edge length parameter h. We adjust h to obtain meshes with around 500 vertices. MIQ might introduce polygonal faces around singularities of the field if the resolution is too coarse, but this is not a problem for our application, since we need to generate a planar graph, not necessarily a pure quad mesh. In the remeshing phase of MIQ, we force edge chains to snap to open edges and sharp features; in the end we (combinatorially) remove all the supported edges since they should not belong to Γ, as explained in Section 3.

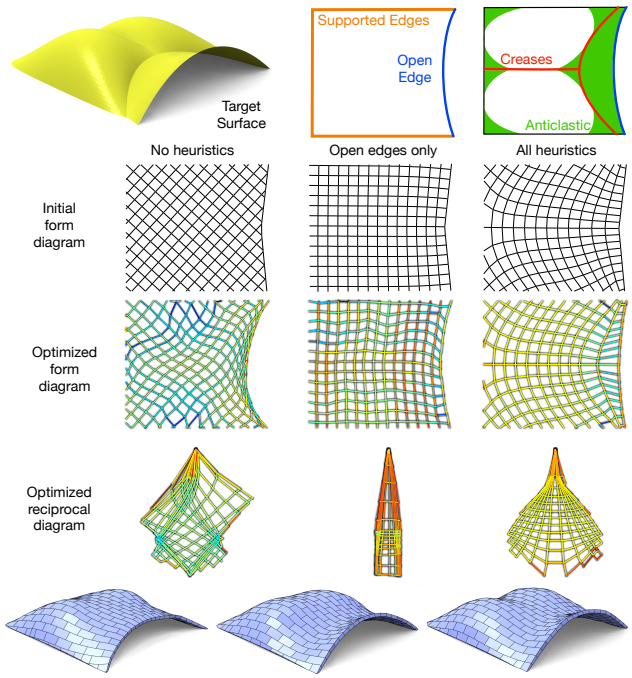

Figure 6: *We demonstrate the effect of our heuristics. We compute the best-fit models (bottom row), starting from the yellow height field $\mathcal{H}$. We use no heuristics on the left column, we add the constraints on the open edges in the middle and all of them on the right. We show the generated form diagram in the second row, the optimized form and reciprocal diagrams after the best-fit optimization in the third and fourth row. The color represents the normalized magnitude of the force densities, i.e., the ratio of the lengths of the edges of the reciprocal and form diagrams.*

The process above provides us with the initial guess for the form diagram, which we term Γ_+. Its computed connectivity will be fixed for the remainder of our algorithm, inducing corresponding connectivity on the reciprocal graph Γ^*. In what follows, we optimize the *geometry* of the form and reciprocal diagrams in order to create a self-supporting surface that fits the input geometry $\mathcal{H}$ well.

5 Fitting self-supporting surfaces

The force diagram Γ alone is not sufficient to describe the equilibrium state of a self-supporting surface, since it only describes the direction of the forces but not their magnitude, and moreover, a Γ resulting from the previous section may not have a planar reciprocal Γ^*. We now describe an algorithm to optimize the geometry of Γ and generate a reciprocal diagram Γ^*, such that the unique self-supporting surface associated with Γ and Γ^* is close, in the least squares sense, to the input geometry $\mathcal{H}$. The "best-fit" algorithm is divided into two steps:

1. Generate an initial pair of diagrams $(\Gamma,\ \Gamma^*)$ such that Γ^* is the reciprocal of Γ (Section 5.1)
2. Find a best-fitting self-supporting mesh $\mathcal{S}$ by iteratively and jointly optimizing Γ and Γ^* (Section 5.2).

5.1 Initial least-squares diagrams

From Lemma 1 we know that a self-supporting surface is uniquely defined by a pair of *reciprocal* graphs Γ and Γ^*. We already have an initial guess Γ_+ for the form diagram resulting from the field-aligned remeshing. We use a geometric construction inspired by [Rippmann et al. 2012], that we call least-square diagrams, to find a pair of reciprocal graphs $(\Gamma,\ \Gamma^*)$ such that Γ is close to our initial Γ_+. The procedure iteratively deforms a pair of planar graphs to make their edges as parallel as possible, in a least squares sense.

The (combinatorial) connectivity of Γ^* is constructed from the connectivity of Γ_+ according to the definition in Section 3. We initialize Γ^*'s vertex positions as the barycenters of the faces of Γ_+; we rotate Γ^* by $\pi/2$ and then iteratively average the corresponding edges of Γ and Γ^* until they become parallel. Specifically, in each iteration we compute a weighted average direction $\mathbf{t}_i$ of each pair of reciprocal edges:

$$\tilde{\mathbf{t}}_i = (1-\beta)\frac{\mathbf{e}_i}{\|\mathbf{e}_i\|} + \beta\frac{\mathbf{e}_i^*}{\|\mathbf{e}_i^*\|};\ \ \mathbf{t}_i = \tilde{\mathbf{t}}_i/\|\tilde{\mathbf{t}}_i\|\,. \tag{11}$$

We then use the $\mathbf{t}_i$'s to compute new diagrams Γ, Γ^* whose edges are aligned with $\mathbf{t}_i$ as much as possible. The weight β allows to control the deformation of the form diagram; we used $\beta = 0.01$ for all the experiments in our paper, since we want to preserve resemblance to the already optimized Γ_+. After computing the averaged directions, Γ is deformed by the following linear optimization:

$$\mathbf{V} = \underset{\mathbf{V}}{\operatorname{argmin}} \sum_i \|\mathbf{e}_i - l_i\mathbf{t}_i\|^2 \tag{12}$$

$$s.t.\ \ \mathbf{v}_1 = \mathbf{0}, \tag{13}$$

where l_i are the edge lengths of Γ in the previous iteration. The deformation for Γ^* is computed equivalently. The constraint (13) is set to eliminate the translational degree of freedom of the energy (12). We then repeat the averaging and the subsequent deformation iteratively until the angles between the edges of Γ and Γ^* become smaller than a tolerance ϵ, which we set at 0.01 degrees in our experiments. The effect of this procedure is shown in Figure 7, where we compute the initial (Γ, Γ^*).

The generated pair of reciprocal graphs can be used to extract a self-supporting surface using Equation 4. Rewritten in matrix form (7), it reduces to solving the following system:

$$\begin{aligned} \mathbf{C}_{\mathcal{V}_\mathcal{U}}^T\mathbf{D}_\mathbf{w}\mathbf{C}\,\mathbf{z} &= \mathbf{p} \\ s.t.\ \ \mathbf{z}_\mathcal{B} &= \mathcal{H}_\mathcal{B} \end{aligned} \tag{14}$$

where $\mathbf{D}_\mathbf{w}$ is the diagonal matrix with the force densities, computed as ratios between the edge lengths of Γ^* and Γ (see Lemma 1).

As discussed in Section 3, by scaling the reciprocal diagram we generate a family of self-supporting surfaces; we can thus pick the surface that is the closest, in the least squares sense, to the target geometry $\mathcal{H}$. The solution is unique, and can be computed by solving the following quadratic program:

$$\underset{\mathbf{z},r}{\operatorname{argmin}}\ \|\mathbf{z}-\mathbf{h}\|^2 \tag{15}$$

$$s.t.\ \ \mathbf{C}_{\mathcal{V}_\mathcal{U}}^T\mathbf{D}_\mathbf{w}\mathbf{C}\,\mathbf{z} - r\mathbf{p} = \mathbf{0}, \tag{16}$$

$$\mathbf{z}_\mathcal{B} = \mathcal{H}_\mathcal{B},\ \ r > 0, \tag{17}$$

where $\mathbf{h}$ is a vector containing the target height $\mathcal{H}(x_i, y_i)$ for vertex i of Γ, and r is the scaling factor for the force densities. Note that the equilibrium constraint (16) is linear in $\mathbf{z}$ and r. After solving Equation (15), we update the vertex coordinates of the reciprocal diagram Γ^* by scaling all of its vertices by $1/r$.

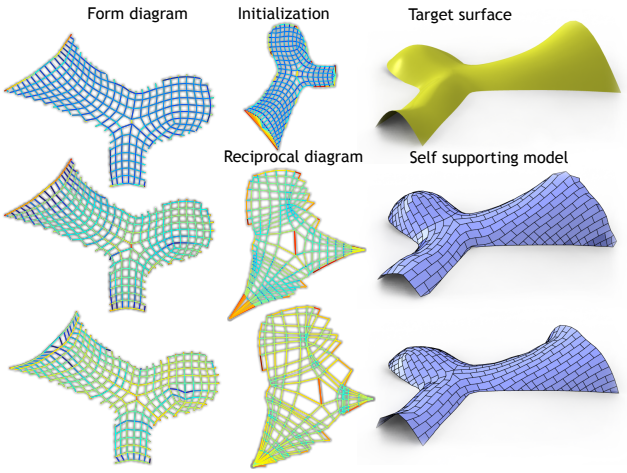

Figure 7: *From top to bottom: the initial diagrams, the same diagrams optimized to be reciprocal pairs, and the diagrams after the best-fit optimization. On the right, we show the target (top) and the self-supporting surfaces generated from the diagrams on the left.*

5.2 Best-fit gradient descent

The iterative averaging process in Section 5.1 did not take into account the target geometry $\mathcal{H}$ when producing (Γ, Γ^*), it was only concerned with computing a Γ^* that is indeed a reciprocal of Γ. Although the subsequent optimization of the scaling factor r brings us closer to fitting the target shape, the self-supporting mesh we computed can be still quite far from $\mathcal{H}$ (Figure 7). We now deform the pair of diagrams to achieve a better fit.

We measure the distance to the target shape $\mathcal{H}$ using the same objective as in Equation (15), but this time we optimize by varying the positions of the vertices of Γ and Γ^*, $\mathbf{V}$ and $\mathbf{V}^*$, while constraining them to remain a reciprocal pair. Since they are a reciprocal pair, the geometry of Γ and Γ^* can be parameterized by the force densities $\mathbf{w}$: $\mathbf{V} = \mathbf{V}(\mathbf{w})$, $\mathbf{V}^* = \mathbf{V}^*(\mathbf{w})$. Equation (14) enables us to express the z-coordinates of the self-supporting surface as a function of $\mathbf{w}$ as well (see the Appendix). Hence our objective becomes

$$f(\mathbf{w}) = \|\mathbf{z}(\mathbf{w}) - \mathcal{H}\left(\mathbf{V}(\mathbf{w})\right)\|^2 \tag{18}$$

and the optimization of $\mathbf{w}$ is formulated as

$$\min f(\mathbf{w})\quad s.t.\ \Gamma^*(\mathbf{w}) \text{ is the reciprocal of } \Gamma(\mathbf{w}). \tag{19}$$

Our strategy to minimize (19) is to perform gradient descent steps, and enforce the reciprocity constraint in each iteration. Similarly to [Vouga et al. 2012], we assume the loads $\mathbf{p}$ are fixed for one optimization step, as well as the target height values $\mathbf{s} := \mathcal{H}\left(\mathbf{V}(\mathbf{w})\right)$.

The gradient descent step $(t+1)$ updates the current force densities:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \lambda\nabla f(\mathbf{w}^{(t)}). \tag{20}$$

λ is selected with a backtracking line search algorithm, and $\nabla f(\mathbf{w}^{(t)})$ can be derived analytically, as shown in the Appendix.

After obtaining the updated force densities $\mathbf{w}^{(t+1)}$, we deform Γ^* and Γ so that the ratios between their respective edge lengths are the $\mathbf{w}^{(t+1)}$'s and they stay reciprocal. To do this, the positions of the vertices of Γ and Γ^* are updated using the same least-squares diagram algorithm presented in Section 5.1, with the only difference that instead of trying to preserve the original edge lengths $\|\mathbf{e}\|$, $\|\mathbf{e}^*\|$,

we want to have the ratio $\|\mathbf{e}^*_{ij}\| = w_{ij}^{(t+1)}\|\mathbf{e}_{ij}\|$. These slightly different iteration formulas are given in the Appendix.

Similarly to [Vouga et al. 2012], we are not guaranteed to decrease the best-fit energy at every step, since the energy might increase when we enforce the reciprocal constraints. However, in our experiments the algorithm always converges to a local minimum, which of course might not be the global optimum (Figure 9). To speed up the gradient descent, we do not run the least-squares diagrams algorithm until (Γ, Γ^*) are perfectly reciprocal: a single iteration is sufficient to approximately enforce the constraints, and we iterate until they are precisely reciprocal only after the last step of gradient descent.

5.3 Discussion

Our best-fit approach generates a sequence of self-supporting surfaces that approximate the input shape $\mathcal{H}$. This is different from the previous method of [Vouga et al. 2012], which starts from the target shape $\mathcal{H}$ and gradually deforms it into a self-supporting surface. The energy minimized by the two methods is also different: [Vouga et al. 2012] minimizes the residuals of Equation (1), using penalty terms in order to stay close to $\mathcal{H}$. In contrast, we minimize directly the squared difference between $\mathcal{H}$ and the generated surface. The fit is thus different, as shown in the extreme example of Figure 4. The use of a non-optimal pattern clearly shows that our method is more global, but loses local features like the shape of the bumps, while [Vouga et al. 2012] is local, but must introduce additional features to make the structure self-supporting. The two algorithms perform similarly on the example in Figure 1, producing two indistinguishable fits with a Hausdorff distance of 0.008 to the input height field (ours) and 0.003 ([Vouga et al. 2012]).

Similarly to [Vouga et al. 2012], we are also not guaranteed to find the global best-fitting solution, due to the nonlinear nature of the problem and the dependence on the connectivity of Γ, which in our case is obtained using structural heuristics. In fact, both algorithms may get stuck in a local minimum or fail to converge if the force patterns are not sufficiently supported. In Figure 8, we show that our method is quite robust to bad input surfaces, and works even if they are far from being self-supporting, while on the same example the method of [Vouga et al. 2012] does not converge. On the other hand, in Figure 9, our method finds a local minimum that is not as good as the one found by [Vouga et al. 2012]. The difference could also be a consequence of the different discretization of the loads: we use the Voronoi areas on the *surface*, whereas [Vouga et al. 2012] uses the Voronoi areas in the horizontal projection. In flat regions they nearly coincide, but on steep regions (like the borders of Figure 9) the projected approximation is less accurate.

In the method of Vouga et al. [2012], the scaling of the forces (called ρ in their paper), that in theory should not affect the shape of the final best-fitting surface, actually does change the computed surface, since it scales only the vertical part of the equilibrium equation (Equation (7) of [Vouga et al. 2012]) and of the corresponding energy, changing the minimum. An improper choice of this weighting term

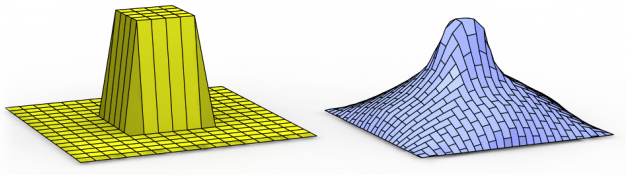

Figure 8: *Our method can find a self-supporting surface (right) even if the input heighfield is far from being self-supporting (left).*

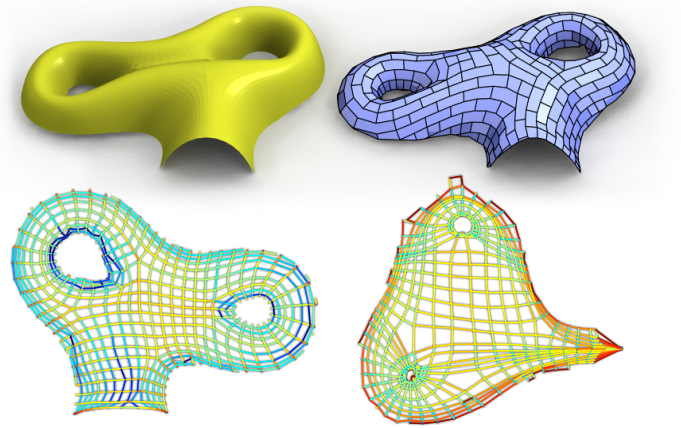

Figure 9: *Our method is not guaranteed to find the global minimum; in this example the self-supporting surface we compute (top right), is not as close to the input height field (top left) as the result in [Vouga et al. 2012]. The Hausdorff distance (normalized w.r.t. the bounding box diagonal) is 0.071 for our fit and 0.059 for [Vouga et al. 2012]'s.*

may cause the algorithm to diverge. In our method, this parameter (r) is actually a *variable* that participates in the best-fit: we optimize it in Equation (15).

To summarize, both methods have their advantages and disadvantages. We suggest to use our method if a least-squares fitting is desired, and use [Vouga et al. 2012] if it is important to preserve the parts of the input shape that are locally self-supporting, at the price of potentially introducing additional features in the non-self-supporting parts of $\mathcal{H}$. Combining the two methods by deforming the target surface using our least-squares diagrams to impose the horizontal equilibrium is an interesting venue for future work.

6 Optimization of the block pattern

A self-supporting model can be constructed by extruding the self-supporting mesh computed in Section 5 equally in the positive and negative vertex normal directions, and then remeshing it into blocks. The structure will stand without any support other than the defined one, provided that there is sufficient friction between the blocks. The force pattern optimized in Sections 4-5 is the optimal explanation of the equilibrium for the model, since we generate the model by extruding it: the form diagram is exactly in the middle by construction, and it is the safest equilibrium description, since a structure with this specific shape and an infinitesimally small thickness will stand. As this shape is the direct result of the chosen combination of form- and reciprocal diagrams, we thus align our blocks to the resulting force pattern to avoid sliding failure [Rippmann and Block 2013].

The most obvious tessellation can be computed by projecting the form diagram onto the self-supporting surface, obtaining a quad mesh tessellation aligned with the flow of forces. However, this choice is not optimal, since chains of consecutive quads (a quad strip between two edge loops) can freely slide and fall off, as shown in Figure 10. To prevent this problem, masonry structures are usually built using staggered patterns [Rippmann and Block 2013]. We generate such patterns using a greedy algorithm that converts an existing quad mesh into an hexagon-dominant tessellation. Each hexagon is then transformed into a block by extruding its vertices both in the positive and negative vertex normal directions, and the generated non-planar faces are arbitrarily triangulated.

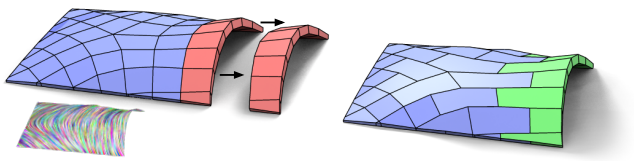

Figure 10: *A staggered block pattern (right) prevents the formation of chains of blocks that can easily slide off the surface.*

Initial quad mesh. To keep the resolution of the block tessellation decoupled from the resolution of the form diagram, we perform an additional remeshing step: the best-fit surface $\mathcal{S}$, computed in Section 5, is uniformly triangulated and then remeshed into a quad mesh $\mathcal{Q} = \{\mathcal{V}_\mathcal{Q}, \mathcal{E}_\mathcal{Q}, \mathcal{F}_\mathcal{Q}\}$, employing the same heuristics used to generate the force diagram in Section 4.

Staggered pattern. A regular grid can be transformed into an hexagonal, staggered pattern by removing "every second edge". In Figure 11, we apply this procedure to a regular grid, and as can be seen, the generated pattern has an orientation, i.e., all the edges that we removed are parallel. A different tessellation could be obtained by rotating this image by 90 degrees, i.e., by removing horizontal instead of vertical edges. The orientation of the pattern is important in order to remove all the chains of quads that can slide off from open edges (Figure 10). In the regions that are far from the open edges, the orientation of the pattern is not important, but we would like it to change smoothly over the surface for aesthetics reasons.

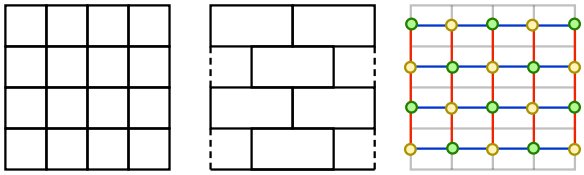

Figure 11: *A quad mesh (left) is transformed in a block pattern made of hexagonal pieces (middle) by removing the edges that correspond to the yellow nodes in the dual graph (right).*

A regular staggered pattern can be constructed only starting from a quad mesh that does not contain singularities, and since it is usually not the case for complex shapes, we must robustly introduce variations in the pattern to handle the singularities. The orientation of the block pattern is computed as a smooth 2-RoSy field $\mathbf{F}$ on the self-supporting surface $\mathcal{S}$. The orientation is used to decide whether an edge is "horizontal" or "vertical": only the vertical edges will be candidates for removal (Figure 11). We constrain the field to be parallel to open edges, obtaining a correct alignment of the pattern on them and smoothly varying it everywhere else. We denote by $\mathbf{s}(e)$ the alignment score of an edge $e \in \mathcal{E}_\mathcal{Q}$, computed as:

$$s(\mathbf{e}) = \|\mathbf{e}^T \mathbf{F}(\mathbf{e})\| \tag{21}$$

where $\mathbf{F}(\mathbf{e})$ is the direction of the 2-RoSy field on the barycenter of the edge $\mathbf{e}$. A subset of the edges of $\mathcal{Q}$ is removed to create the block pattern; we select the edges to remove by solving a 2-coloring problem on a graph $\mathcal{G} = \{\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G}\}$, where $\mathcal{V}_\mathcal{G} = \mathcal{E}_\mathcal{Q}$. $\mathcal{E}_\mathcal{G}$ contains an edge for every face $f \in \mathcal{F}_\mathcal{Q}$ that connects the two edges of f with lower alignment score $\mathbf{s}$ (these are the blue edges in Figure 11). Similarly, $\mathcal{E}_\mathcal{G}$ contains an edge for every vertex $v \in \mathcal{V}_\mathcal{Q}$, that connects the two edges incident on v with lower alignment score $\mathbf{s}$ (red edges). After the 2-coloring problem is solved, all edges of the first color are removed. To solve the 2-coloring problem, we start from a random vertex and we propagate the color to the neighbors with a breadth-first search.

7 Results

We ran our experiments on an Intel i7 quad-core processor clocked at 3.4 GHz. Our implementation is written in MATLAB, with the exception of the field generation and quadrangulation algorithms, which have been implemented in C++.

For all the examples in the paper, the user input is a surface designed with Rhino and a set of supporting vertices $\mathcal{B}$. We show statistics for our examples in Table 1.

We 3D-printed and constructed two models: a freeform staircase inspired by the famous Guastavino's stairs (Figure 12) and a freeform vault (Figure 1). The blocks have been 3D-printed using a Z Corp ZPrinter 650 with a fixed thickness of 8 mm. In addition to the blocks, a wooden base has been manually designed and laser-cut to hold the supported blocks in place. The construction of the models also requires formwork to hold the pieces until they are all placed. The formwork has been milled and cut up into pieces, so that it can be taken out at the end of the assembly (Figure 13). The accompanying video shows the construction of the model. Interestingly, since masonry is a problem of stability rather than stresses [Heyman 1995], it is independent of scale. As a result, scaled block models can actually be used as structural models [Zessin et al. 2010; Van Mele et al. 2012].

Guastavino's stairs. In Figure 12 we show two unreinforced shapes inspired by Guastavino's stairs. On the top, we used straight open edges and a more geometrical design, while in the bottom the same idea is realized with smooth edges and supports. Our method automatically finds best-fit surfaces that are very close to the sketches, with the exception of the top part that needs to be modified to make the structure self-supporting. The fits computed for both surfaces suggest that both are self-supporting, but by looking at the form and reciprocal diagrams, we can get an insight on the stability of the model (the magnitudes of the forces are pseudo-colored on a logarithmic scale, dark red indicates large magnitude). We clearly see that the structure in the top, while self-supporting, will not be stable when 3D-printed. The reciprocal diagram of the top result, when drawn to scale, is approximately 2 times larger than the one for the bottom result; this means that the forces are overall much

Model	$\lvert\mathcal{E}_\mathcal{U}\rvert$	T_{Γ_+}	T_{bestfit}	T_{tess}	#B	Err.	d_H
Teaser	1436	6.8	24.0	10.9	248	$1.2 \cdot 10^{-06}$	0.01
Stairs (t)	518	2.6	10.1	5.0	131	$5.6 \cdot 10^{-06}$	0.05
Stairs (b)	871	2.9	13.4	5.7	148	$1.4 \cdot 10^{-08}$	0.04
Bumps	418	6.3	14.3	7.3	49	$1.1 \cdot 10^{-04}$	0.02
Tristar	444	3.1	17.0	11.6	358	$1.3 \cdot 10^{-09}$	0.02
Db. torus	1151	3.6	25	11.9	334	$4.7 \cdot 10^{-10}$	0.03
Spiral	249	3.1	6.3	8.7	222	$5.8 \cdot 10^{-12}$	0.01
Jerónimos	1004	6.8	20.3	69.2	1164	$5.6 \cdot 10^{-04}$	0.02

Table 1: *Statistics for our experiments. From left to right: the number of edges in Γ; the optimization time (in seconds) to generate the initial guess Γ_+ (Sec. 4), best-fit (Sec. 5), tessellation (Sec. 6); the number of blocks; the maximum relative error (measured as the maximal per-vertex residual in Equation (1), normalized by the Voronoi area); the Hausdorff distance (normalized w.r.t. the bounding box diagonal) between our best-fit solution and the user-provided height field.*

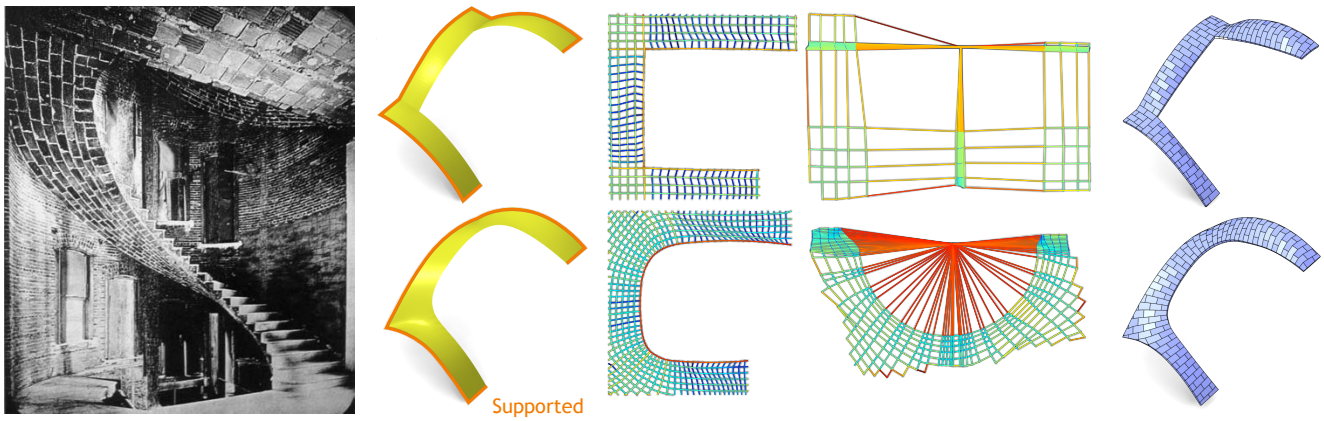

Figure 12: *Two designs inspired by the Guastavino's stairs. On the top, the sharp edges of the input model generate a surface with an uneven force distribution. A smoother design allows forces to be evenly distributed, as can be seen from the form and reciprocal diagrams. [Copyright photograph: Guastavino/Collins Collection, Avery Architectural and Fine Arts Library, Columbia University].*

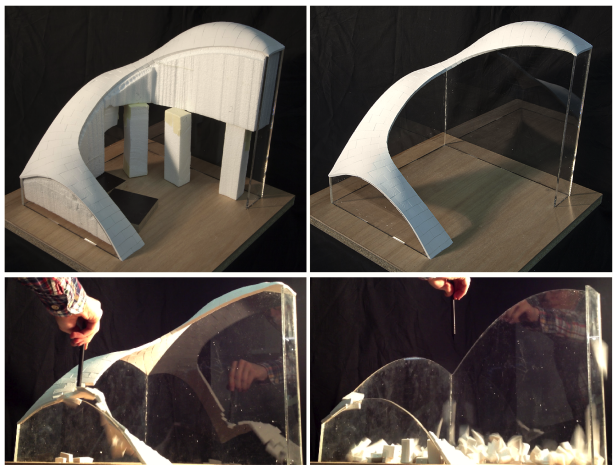

Figure 13: *We use a milled formwork to assemble the model. When all the pieces are in place, the model becomes self-supporting, and the formwork can be removed. Additional loading lead to collapse.*

bigger and can potentially break the bricks in a real structure. Also, the distribution of the forces is different. In the top, the forces are concentrated in a few edges, meaning that the forces acting on most of the blocks are very small, and they could thus easily collapse when a small point load is applied or if the 3D printing is not perfectly accurate. In the bottom, the distribution is more even, with bigger forces on the open edge. We printed this model at the size of 45cm $\times$ 45cm $\times$ 45cm, and we show a collapse sequence generated by applying external point loads in the accompanying video and in Figure 13.

Freeform vault. Our second 3D-printed result is a freeform vault, supported on the boundary and on two small edges in the middle (Figure 1). The model has two wide unsupported holes and it also contains parts that are almost horizontal, but surprisingly it can stand in compression only. The model is very stiff and robust to external point loads, as shown in the accompanying video. Its size is 55cm $\times$ 40cm, and the height is just 10cm. The boundary support has been 3D-printed, and the plexiglas columns laser cut.

Spiral stairs. The model shown in Figure 14 is supported all along the outside edge and the two short edges at the top and at the bottom.

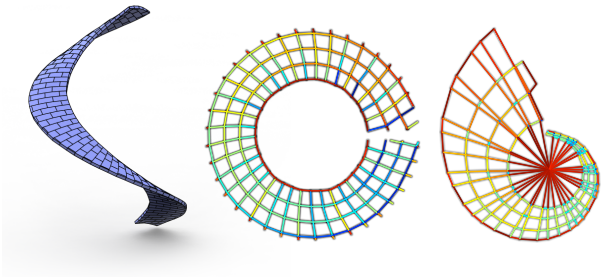

Figure 14: *A self-supporting spiral structure and the corresponding form and reciprocal diagrams. This model is supported only on the outer edge and on the end lines.*

The target surface is a half-parabolic section swept along a helix with a constant "climbing angle". The spiraling vault would want to be synclastic, meaning that the sweep curve would also want to be e.g. a parabolic section, rather than a straight line. To compensate for this, as shown in the reciprocal diagram in Figure 14, the forces need to increase towards one of the short supported boundaries, to approximate the straight section (zero curvature) in the spiraling "direction", resulting in the beautiful shell shape.

Jerónimos vault. We sketched a section of the Jerónimos vault and used it to generate a physically plausible 3D model (Figure 15). The model is supported on the two central pillars and at the boundary, since this is not the full vault but just a section. For this example, we constrained the form diagram generation to have edges perpendicular to the boundary. This simple change to our constraints is a useful tool, since it allows to control the boundary conditions of the form diagram. This can be used to divide a complex but symmetric vault into multiple sections and compute the best-fit separately for each.

Limitations. We do not have a guarantee that the least-square diagrams algorithm will always generate a pair of reciprocal diagrams, although in all our experiments this was indeed the case. Further, our algorithm may fail to produce a self-supporting surface when the input is not properly supported. While we do not have a formal proof, we experimentally found that our best-fit algorithm is robust, and it always generates a self-supporting surface given a properly supported form diagram, possibly generating a surface that is very far from the input surface if it contains impossible features (Figure 8).

We are not always able to find the globally optimal best-fit surface, since this would require to find the connectivity and geometry of the optimal form diagram, which is a very hard combinatorial problem.

8 Conclusions and future work

We presented an automatic algorithm to generate self-supporting models, starting from a user-provided height field. The algorithm is automatic and able to generate models for a wide range of shapes. The 3D-printed models validate our approach, showing that it is possible to completely automatize the generation of masonry models.

Our formulation can currently handle height fields, but it should be possible to extend it to support arbitrary surfaces; we leave this extension as a future work. Another interesting research problem is the automatic segmentation of the input into parts that can be constructed independently, thus greatly reducing the formwork needed for their realization. This is a combinatorial problem on top of an already hard optimization problem that is also combinatorial. Finally, machine learning techniques could be used to obtain more refined heuristics and further improve the quality of the diagrams and the block tessellations.

We believe that our contribution will have an impact both in the computer graphics and the structural engineering community. In the former, the automatic generation of physically plausible masonry models will allow to generate complex virtual environments where the characters can realistically interact with the structures. In the latter, the use of the proposed algorithms, combined with additional user input, will enable designers to freely explore self-supporting forms without requiring structural knowledge.

Acknowledgements

We thank Etienne Vouga for consulting us on the reimplementation of [Vouga et al. 2012]. We are grateful to Matthias Rippmann, Ramon Elias Weber and Paul-Emmanuel Sornette for helping with the fabrication and construction of the physical masonry models, to Masoud Akbarzadeh for designing the models in Figures 3 and 9 and to Emily Whiting for narrating the accompanying video. This work was supported in part by an SNF award 200021_137879, ERC grant iModel (StG-2012-306877) and a gift from Adobe Research.

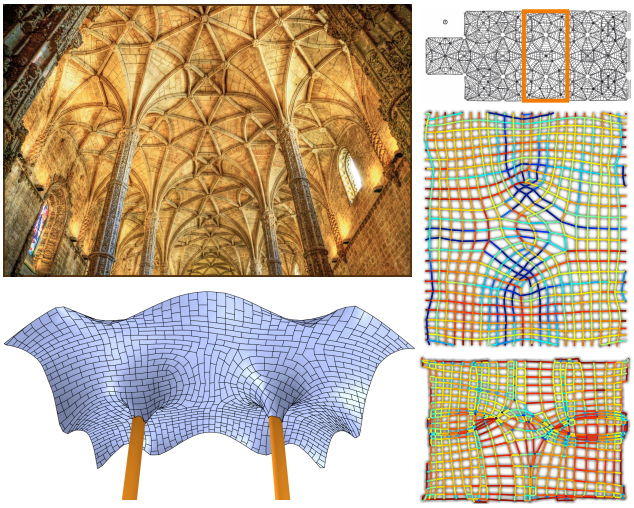

Figure 15: *A physically valid model of a section of the vault in the Jerónimos monastery, Lisbon, has been generated using our algorithm. [Copyright photograph: Bert Kaufmann]*

References

BHOOSHAN, S., AND EL SAYED, M. 2011. Use of subdivision surfaces in architectural form-finding and procedural modelling. In *Proc. Symp. Simulation for Architecture and Urban Design*.

BLOCK, P., AND LACHAUER, L. 2012. Three-dimensional equilibrium analysis of gothic masonry vaults. In *Proc. VIII Int. Conference on Structural Analysis of Historic Construction*.

BLOCK, P. 2009. *Thrust Network Analysis: Exploring Three-dimensional Equilibrium*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA.

BOMMES, D., ZIMMER, H., AND KOBBELT, L. 2009. Mixed-integer quadrangulation. *ACM Trans. Graph. 28*, 3, 77.

BORGART, A. 2005. The relationship of form and force in (irregular) curved surfaces. In *Proc. Int. Conf. Computation of Shell and Spatial Structures*.

BOUAZIZ, S., DEUSS, M., SCHWARTZBURG, Y., WEISE, T., AND PAULY, M. 2012. Shape-up: Shaping discrete geometry with projections. *Comput. Graph. Forum 31*, 5, 1657–1667.

CALLADINE, C. R. 1983. *Theory of Shell Structures*. Cambridge University Press, Cambridge.

CAZALS, F., AND POUGET, M. 2003. Estimating differential quantities using polynomial fitting of osculating jets. In *Proc. SGP*.

CRANE, K., DESBRUN, M., AND SCHRÖDER, P. 2010. Trivial connections on discrete surfaces. *Comput. Graph. Forum 29*, 5.

EIGENSATZ, M., KILIAN, M., SCHIFTNER, A., MITRA, N. J., POTTMANN, H., AND PAULY, M. 2010. Paneling architectural freeform surfaces. *ACM Trans. Graph. 29*, 4.

FRATERNALI, F. 2010. A thrust network approach to the equilibrium problem of unreinforced masonry vaults via polyhedral stress functions. *Mechanics Research Communications 37*, 2.

FU, C.-W., LAI, C.-F., HE, Y., AND COHEN-OR, D. 2010. K-set tilable surfaces. *ACM Trans. Graph. 29*, 4.

HEYMAN, J. 1995. *The Stone Skeleton: Structural engineering of masonry architecture*. Cambridge University Press.

HILDEBRANDT, K., POLTHIER, K., AND WARDETZKY, M. 2005. Smooth feature lines on surface meshes. In *Proc. SGP*.

KILIAN, A., AND OCHSENDORF, J. 2005. Particle-spring systems for structural form finding. *J. IASS 148*, 77.

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.

MARK, R. 1982. *Experiments in Gothic structure*. The MIT Press, Cambridge.

O'DWYER, D. W. 1999. Funicular analysis of masonry vaults. *Computers and Structures 73*, 1-5.

PANOZZO, D., LIPMAN, Y., PUPPO, E., AND ZORIN, D. 2012. Fields on symmetric surfaces. *ACM Trans. Graph. 31*, 4.

RAY, N., VALLET, B., LI, W., AND LÉVY, B. 2008. N-Symmetry Direction Field Design. *ACM Trans. Graph. 27*, 2.

RIPPMANN, M., AND BLOCK, P. 2013. Rethinking structural masonry: unreinforced, stone-cut shells. In *Proc. ICE - Construction Materials*.

RIPPMANN, M., LACHAUER, L., AND BLOCK, P. 2012. Interactive vault design. *International Journal of Space Structures 27*, 4, 219–230.

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

SHEWCHUK, J. R. 1996. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In *Applied Computational Geometry: Towards Geometric Engineering*, vol. 1148 of *Lecture Notes in Computer Science*. Springer-Verlag, 203–222.

SINGH, M., AND SCHAEFER, S. 2010. Triangle surfaces with discrete equivalence classes. *ACM Trans. Graph. 29*, 4.

VAN MELE, T., MCINERNEY, J., DEJONG, M., AND BLOCK, P. 2012. Physical and computational discrete modeling of masonry vault collapse. In *Proc. Int. Conf. Structural Analysis of Historical Constructions*.

VOUGA, E., HÖBINGER, M., WALLNER, J., AND POTTMANN, H. 2012. Design of self-supporting surfaces. *ACM Trans. Graph. 31*, 4.

WHITING, E., OCHSENDORF, J., AND DURAND, F. 2009. Procedural modeling of structurally-sound masonry buildings. *ACM Trans. Graph. 28*, 5, 112:1–112:9.

WHITING, E., SHIN, H., WANG, R., OCHSENDORF, J., AND DURAND, F. 2012. Structural optimization of 3D masonry buildings. *ACM Trans. Graph. 31*, 6, 159:1–159:11.

WILLIAMS, C. J. K. 1990. The generation of a class of structural forms for vaults and sails. *The Structural Engineer 68*, 12.

ZESSIN, J., LAU, W., AND OCHSENDORF, J. 2010. Equilibrium of cracked masonry domes. *Proc. ICE-Engineering and Computational Mechanics 163*, 3, 135–145.

ZIMMER, H., CAMPEN, M., BOMMES, D., AND KOBBELT, L. 2012. Rationalization of triangle-based point-folding structures. *Comput. Graph. Forum 31*, 2, 611–620.

Appendix

Gradient of the best-fit energy. Equation (14) allows to express the z-coordinates of the unsupported vertices of the self-supporting surface as a function of $\mathbf{w}$:

$$\mathbf{z}_{\mathcal{V}_\mathcal{U}}(\mathbf{w}) = \left(\mathbf{C}_{\mathcal{V}_\mathcal{U}}^T \mathbf{D}_\mathbf{w} \mathbf{C}_{\mathcal{V}_\mathcal{U}}\right)^{-1} \left(\mathbf{p}_{\mathcal{V}_\mathcal{U}}(\mathbf{w}) - \mathbf{C}_{\mathcal{V}_\mathcal{U}}^T \mathbf{D}_\mathbf{w} \mathbf{C}_{\mathcal{V}_\mathcal{B}} \mathbf{z}_{\mathcal{V}_\mathcal{B}}\right).$$

The gradient of Equation (18) can be derived analytically:

$$\begin{aligned}\nabla f(\mathbf{w}) = &-2(\mathbf{z}_{\mathcal{V}_\mathcal{U}}(\mathbf{w}) - \mathcal{H}\left(\mathbf{V}_{\mathcal{V}_\mathcal{U}}(\mathbf{w})\right))^T \\ &(\mathbf{C}_{\mathcal{V}_\mathcal{U}}^T \mathbf{D}_\mathbf{w} \mathbf{C}_{\mathcal{V}_\mathcal{U}})^{-1} \mathbf{C}_{\mathcal{V}_\mathcal{U}}^T \mathbf{D}_{(\mathbf{C}z(\mathbf{w}))}.\end{aligned}$$

Least-squares diagrams for the best-fit iteration inside the gradient descent iterations of best-fit (Section 5.2) look as follows: (we count the least-squares diagram iterations by index s, and the outer loop of the gradient descent is indexed by t)

$$\begin{aligned}\tilde{\mathbf{t}}^{(s+1)} &= (1-\beta)\frac{\mathbf{e}_i^{(s)}}{\|\mathbf{e}_i^{(s)}\|} + \beta\frac{\mathbf{e}_i^{*(s)}}{\|\mathbf{e}_i^{*(s)}\|} \\ \mathbf{t}_i^{(s+1)} &= \tilde{\mathbf{t}}_i^{(s+1)}/\|\tilde{\mathbf{t}}_i^{(s+1)}\| \\ \mathbf{V}^{(s+1)} &= \operatorname*{argmin}_{\mathbf{V}} \sum_i \left\| \mathbf{e}_i^{(s+1)} - \|\mathbf{e}_i^{(s)}\| \cdot \mathbf{t}_i^{(s+1)} \right\|^2 \\ \mathbf{V}^{*(s+1)} &= \operatorname*{argmin}_{\mathbf{V}^*} \sum_i \left\| \mathbf{e}_i^{*(s+1)} - \|\mathbf{e}_i^{(s)}\| \cdot w_i^{(t+1)} \cdot \mathbf{t}_i^{(s+1)} \right\|^2.\end{aligned} \tag{22}$$