Assembling Self-Supporting Structures

Mario Deuss[1] Daniele Panozzo[2] Emily Whiting[2,3] Yang Liu[4] Philippe Block[2]
Olga Sorkine-Hornung[2] Mark Pauly[1]
[1] EPF Lausanne [2] ETH Zurich [3] Dartmouth College [4] Microsoft Research

Figure 1: *We propose a construction method for self-supporting structures that uses chains, instead of a dense formwork, to support the blocks during the intermediate construction stages. Our algorithm finds a work-minimizing sequence that guides the construction of the structure, indicating which chains are necessary to guarantee stability at each step. From left to right: a self-supporting structure, an intermediate construction stage with dense formwork, an intermediate construction stage with our method and the assembled model.*

Abstract

Self-supporting structures are prominent in historical and contemporary architecture due to advantageous structural properties and efficient use of material. Computer graphics research has recently contributed new design tools that allow creating and interactively exploring self-supporting *freeform* designs. However, the physical construction of such freeform structures remains challenging, even on small scales. Current construction processes require extensive formwork during assembly, which quickly leads to prohibitively high construction costs for realizations on a building scale. This greatly limits the practical impact of the existing freeform design tools. We propose to replace the commonly used dense formwork with a sparse set of temporary chains. Our method enables gradual construction of the masonry model in stable sections and drastically reduces the material requirements and construction costs. We analyze the input using a variational method to find stable sections, and devise a computationally tractable divide-and-conquer strategy for the combinatorial problem of finding an optimal construction sequence. We validate our method on 3D printed models, demonstrate an application to the restoration of historical models, and create designs of recreational, collaborative self-supporting puzzles.

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

Keywords: masonry models, static equilibrium analysis, self-supporting surfaces, optimization, sparsity, assembly order

Links: DL PDF WEB

1 Introduction

The majority of man-made objects are composed of multiple interlocking parts, kept together by glue, bolts or other connections. The division into components is often necessary to achieve a certain purpose (computers, cars) or to make the fabrication of large models feasible or cheaper (buildings, furniture, roads, railways, large 3D printed models, etc.).

In this work, we focus on the construction of self-supporting structures that are composed of bricks or stone blocks without any mortar to bind them together. Most of the world's architectural heritage consist of self-supporting masonry structures that require no supporting framework, since the entire structure is in a static equilibrium configuration.

The design of modern, freeform self-supporting structures has recently received a lot of interest in computer graphics [Vouga et al. 2012; Liu et al. 2013; de Goes et al. 2013; Panozzo et al. 2013], but their physical construction has only been addressed for small-scale models. The method proposed in [Panozzo et al. 2013] relies on *dense* formwork (Figure 1) to support all the blocks until the entire construction is completed; after the last piece is put in place, the structure is in equilibrium and the formwork can be carefully removed. This method is difficult to apply to large scale structures, because a dense formwork able to sustain the weight of large stone blocks is too expensive and not practical, especially considering that the formwork has to be dismantled after all the blocks are in place. Also, removing the formwork demands technically complex and expensive solutions: the formwork has to be lowered evenly to avoid failures due to redistribution of forces. Due to the lack of an economically feasible construction strategy, freeform masonry structures are currently rarely built, despite their advantageous structural properties and unique aesthetics. Additionally, the majority of the cost is associated with the foundations necessary to support the formwork.

We propose a different approach, replacing the dense formwork with a sparse set of chains that are connected to fixed anchor points. While chains have been used for construction before, our method specifically aims at finding a work-minimizing assembly sequence, requiring as few chains to be rehung as possible. Our solution leverages the internal force distribution of the partially assembled structure and only provides the minimally required additional supports to keep the structure in static equilibrium at all stages of the assembly. The use of chains has been pioneered in modern construc-

Figure 2: *The Arch-Lock system [Drew 2013] is used to construct a simple arch (left), and a barrel vault (right). [Copyright photographs: Lock-Block Ltd. 2013]*

tion by [Drew 2013], who successfully built simple self-supporting structures using one or more chains per block (Figure 2). Historical methods applied rope supports in arch construction [Fitchen 1961]. Our work extends this idea to general freeform self-supporting structures. We show that designs created with the methods of [Vouga et al. 2012; Liu et al. 2013; de Goes et al. 2013; Panozzo et al. 2013] can all be handled by our algorithm.

The core problem that we address is finding a sequence of block insertions that minimizes a specific cost function defined on the number of chains required during assembly. This search problem is hard, because the space of assembly sequences is exponential in the number of blocks and anchor points. The situation is exacerbated by the fact that even verifying the force equilibrium of a single construction state is already computationally involved (see Section 3). As illustrated in Figure 3, naive solutions lead to an impractically high number of chains and are often not able to complete the structure since it is impossible to find valid configurations of chains that keep the structure in equilibrium for each state (Figures 3, 13, 14, and 15 are the only ones that can be constructed using a trivial z-filling sequence according to the equilibrium model we use). In feasible sequences, the z-ordering approximately doubles the number of times a chain needs to be rehung compared to our solution, which is significant for real construction.

To make this combinatorial optimization problem computationally tractable, we introduce a divide-and-conquer strategy that first decomposes the design model into *stable sections*. This decomposition is computed using an optimization approach that applies sparsity-inducing norms to minimize the number of non-zero forces acting between blocks. Given the decomposition, we apply a greedy optimization to find the assembly sequence of the individual sections. While this strategy is not guaranteed to find the globally optimal solution, it greatly reduces the amount of work and chains compared to non-optimized construction sequences.

While our focus is on the fabrication of self-supporting surfaces, our contributions can be applied to other domains, such as restoration of existing structures and design of *self-supporting puzzles*. These can be printed with a consumer 3D printer and assembled by multiple players using fingers instead of chains.

The contributions of this paper are as follows:

1. We propose an alternative way of constructing masonry structures that requires a negligible amount of formwork compared to traditional techniques.
2. We present an optimization algorithm to analyze the equilibrium of a structure exposing its arches and implying a segmentation into stable sections. We propose an algorithm that leverages sparsity to minimize the number of chains that are necessary to avoid failures in the intermediate construction stages.
3. We validate our algorithm on physical examples of a small-scale 3D printed model and a self-supporting puzzle.

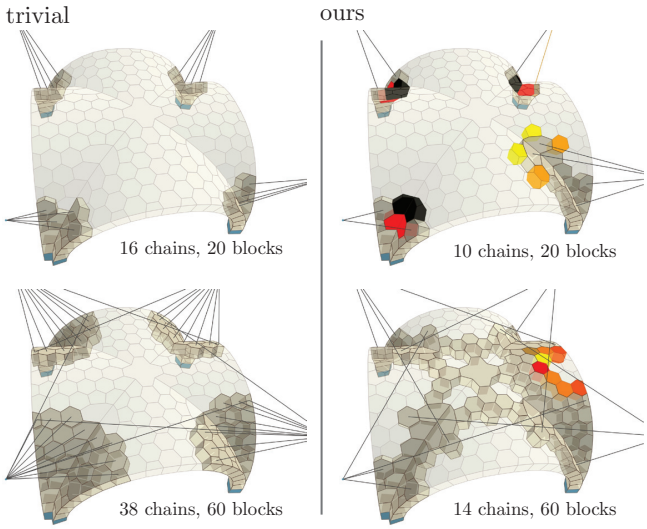


Figure 3: *Two intermediate construction stages of our optimized sequence (right) and a trivial z-ordering (left). Our sequence needs considerably less work (0.62 instead of 1.13 chain changes per state in average), while computing it takes 3.5 times longer than determining the sparse set of chains for the trivial sequence.*

2 Related work

Masonry building fabrication. Historically, construction practices for masonry buildings involved elaborate timber-frame structures guiding the vaulted forms and further sub-structures for intermediate points of support [Fitchen 1961; Fallacara 2012]. In the construction of modern freeform shells, traditional methods are still in use, with wood panels cut according to section curves [Wendland 2009]. Formwork can be reduced in thin tile vault construction [Ramage et al. 2010; Davis et al. 2012], however, these methods rely on significant tensile strength of mortar in comparison to the light weight of the tiles. We address the more general case of heavy masonry blocks where mortar strength must be neglected, and consequently, intermediate stages of construction require dense support structures.

While less common, in medieval vault structures, tensioned ropes were sometimes used to hold arch blocks in place [Fitchen 1961]. Tensioned elements have also been used as an alternative to formwork in the contemporary building industry. For example, the Arch-Lock system [Drew 2013] uses chains in the construction of Roman arch bridges, tunnels and vaults. This chain-based system was an inspiration for our work, however our method greatly expands on the complexity and generality of chain supports, such that it can be applied to *freeform* shell construction.

Self-supporting structures. Optimization of masonry structures is an active area of research in the computer graphics community. The shape of architectural models can be automatically adjusted to guarantee structural stability [Whiting et al. 2009; Whiting et al. 2012]. Our approach uses the same model of statics for verifying structural stability, as formalized by Livesley [1992]. Much effort has been devoted to designing valid self-supporting shapes [Vouga et al. 2012; Liu et al. 2013; de Goes et al. 2013; Panozzo et al. 2013], yet the issue of how to construct such forms from the ground up has been largely ignored. Cable elements were integrated in masonry design in [Whiting et al. 2012] but with predetermined connectivity.

Optimizing construction sequences. Constructability has been investigated in the context of 3D puzzles [Lo et al. 2009; Xin et al. 2011; Song et al. 2012], 3D assembly instructions [Agrawala et al.

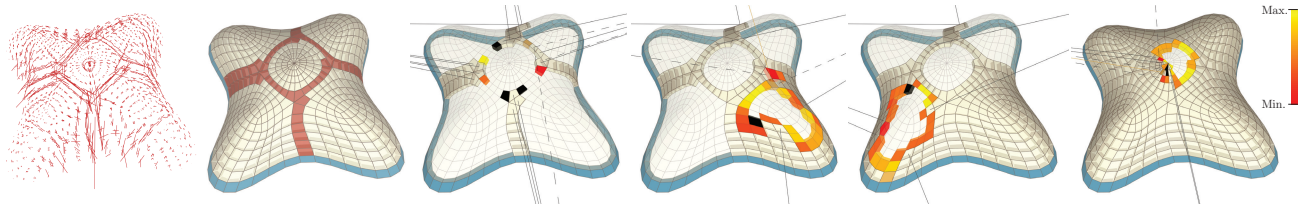


Figure 4: *Our algorithm converts an input masonry model in a work-minimizing construction sequence. From left to right: Forces resulting from our global equilibrium analysis, arch-blocks as extracted from flood-fill, four different states of the construction sequence. In all our figures, the blocks and chains are color-coded as follows: blue for support, light yellow for free blocks, gold for the newly added block and chains, dashed lines for chains that can be removed, and black lines for other active chains. We also color-code the candidate blocks considered by our sequence optimization using the minimum of Equation (5) relative to the other candidates in the color-bar on the right. Candidate blocks leading to an invalid state are shown in black.*

2003], and design with planar interlocking pieces [Hildebrand et al. 2012; Schwartzburg and Pauly 2013; Cignoni et al. 2014]. These methods address geometric constraints that ensure no piece is obstructed by the existing structure during assembly. Some aspects of stability have also been studied in these works, such as the rigidity of joints as a function of slit placement. In contrast, we approach the constructability problem with a focus on equilibrium constraints and optimizations to simplify the assembly process.

3D printing. Support structures are an essential component of the 3D printing process, needed to stabilize a model at all stages of fabrication [Wang et al. 2013]. While 3D printers use an additive method, building models layer by layer, our approach takes advantage of freedom in the construction sequence; blocks can be placed in arbitrary sequences, constrained only by connectivity. Several methods have been developed to determine areas of high stress in the final printed model [Stava et al. 2012; Umetani and Schmidt 2013]. Strut and truss structures have been proposed as a solution to reducing internal stress and preventing breakage of the print material [Stava et al. 2012; Wang et al. 2013]. Our chain-based method is intended for temporary support that can be easily added and removed.

A large body of work exists in optimizing for physical phenomena in fabrication-oriented design. For example, prescribed deformation behavior [Bickel et al. 2010] and kinematic constraints [Coros et al. 2013] have been studied in the context of character design. A sparse set of strings is used in [Skouras et al. 2013] to animate actuated models. We apply fabrication technology for creating physical prototypes, but our goal is rather directed at the assembly, while physical validity of the final shape is assumed at input. New printing technologies supporting large scale 3D printing [Hansmeyer and Dillenburger 2014] could directly be applied to fabricate freeform blocks for masonry structures.

Rationalization. Surface rationalization [Singh and Schaefer 2010; Fu et al. 2010; Eigensatz et al. 2010] and decomposition in freeform sweeps [Barton et al. 2014] can be used to create architectural tessellations that reduce construction costs. These approaches could potentially be used to optimize blocks tessellation, and thus benefit directly from our method that can generate construction sequences for masonry structures composed of arbitrarily shaped blocks.

3 Method

Figure 4 provides an overview of our algorithm to generate a work-minimizing construction sequence for a given self-supporting structure. After introducing some terminology (Section 3.1), we discuss how we verify static equilibrium in a collection of blocks and chains (Section 3.2). This method is an important component of our two-stage optimization algorithm. The first stage analyzes the equilibrium of the surface to find a set of stable regions (Section 3.3). Based on this decomposition, we generate a construction sequence using a greedy algorithm that initially constructs the arches separating the regions, and then fills the stable regions one by one (Section 3.4). We evaluate our algorithm in different scales and applications (see Section 4 for more details).

3.1 Preliminaries

We model a masonry structure as a set of rigid blocks that are represented by closed manifold triangle meshes. The construction site is specified as a collection of *supports* and *anchor points* (see Figure 5). Each block can be in contact with other blocks or any of the supports. We call the contact surfaces between blocks *interfaces*. A block can also be attached to an anchor using a *chain*, represented as an inextensible straight line segment connecting the block and the anchor. Each block has a *hook* that provides an attachment point for the chains. Multiple chains can be attached to the same hook. In our examples, the block's hook is placed at the intersection of the top face and the ray emanating from the center of gravity parallel to the direction of the normal of the top face. An intermediate *construction state* is a spatial arrangement of blocks and chains. We say that a state is *valid* if and only if it is in static equilibrium (Section 3.2).

Our algorithm converts an unordered collection of blocks into an *ordered construction sequence*, composed of valid construction states, each adding one single block to the structure. Of the many construction sequences that exist for a given set of blocks and anchor positions, we strive to find one that minimizes the amount of *work* required in the construction.

We define the work required to assemble a structure as the total

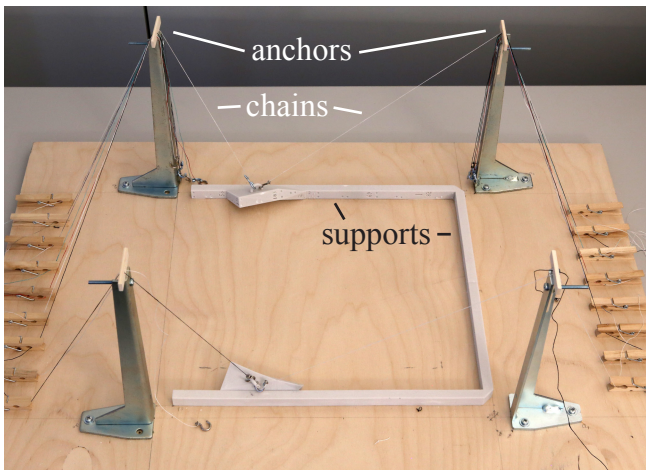


Figure 5: *Construction site mockup.*

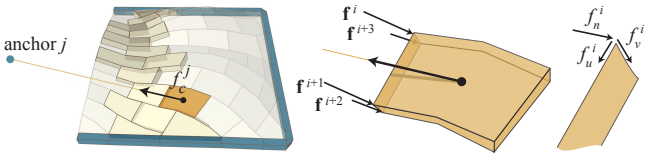


Figure 6: *Model of chain forces and contact forces at interfaces between blocks.*

number of chains added and removed during the construction. This definition is motivated by practical construction concerns: adding or removing a chain is an expensive operation that requires a considerable amount of time and energy, and thus directly relates to the construction cost.

3.2 Construction state validity

An important component of our system is an algorithm to check whether an intermediate construction state is valid, i.e. , the blocks are in static equilibrium. Traditional physical simulation methods are known to be unreliable in a masonry setting due to the extreme stiffness of the system [DeJong 2009]. We thus rely on the alternative model proposed in [Whiting et al. 2009], which we review in the following paragraphs. In this method, a state is valid if there exists a force distribution that satisfies a set of constraints. In general, those constraints admit more than one single distribution and do not fully determine the actual forces acting. We leverage this indeterminacy by picking sparse force distributions to reveal a sparse valid subset of chains on one hand, and an approximately valid subset of blocks on the other hand.

We decompose each force $\mathbf{f}^i$ acting on an interface between two blocks into its axial component f_n^i, perpendicular to the face, and two orthogonal in-plane friction components, f_u^i and f_v^i (Figure 6). We model one force vector per vertex of the contact-polygon between the two blocks. We encode the magnitude of the force introduced by the chain connecting a block $\mathcal{B}$ to an anchor a with a scalar value $f_c^{\mathcal{B},a}$ and its orientation by a unit vector $\mathbf{d}^{\mathcal{B},a}$. Note that we are interested in static equilibrium, so $\mathbf{d}^{\mathcal{B},a}$ is fixed and only depends on the input geometry.

Static equilibrium. Static equilibrium conditions require that net force and net torque acting on each block cancel out. In our model, we assume to have a small set of blocks anchored to the ground and a set of additional forces f_c acting on the blocks due to chain actions. Combining equilibrium constraints for each block yields a linear system of equations [Livesley 1992]. For each block $\mathcal{B}$, the force distribution must satisfy the following equilibrium conditions:

$$\begin{bmatrix} \sum_{i\in V(\mathcal{B})} \mathbf{f}^i & + & \sum_{a\in C(\mathcal{B})} f_c^{\mathcal{B},a}\,\mathbf{d}^{\mathcal{B},a} \\ \sum_{i\in V(\mathcal{B})} \mathbf{A}^{\mathcal{B},i}\mathbf{f}^i & + & \sum_{a\in C(\mathcal{B})} \mathbf{A}_c^{\mathcal{B},a} f_c^{\mathcal{B},a}\mathbf{d}^{\mathcal{B},a} \end{bmatrix} = \begin{bmatrix} -\mathbf{g}_B \\ \mathbf{0} \end{bmatrix} \quad (1)$$

The top row corresponds to force equilibrium, where $\mathbf{g}_B$ is a vector containing the gravity caused by block weight, $V(\mathcal{B})$ is the set of all force indices acting on the interfaces of block $\mathcal{B}$, and $C(\mathcal{B})$ is the set of all chains acting on block $\mathcal{B}$. The bottom row corresponds to torque equilibrium where matrix $\mathbf{A}^{\mathcal{B},i}$ contains coefficients for the torque contribution of force $\mathbf{f}^i$, and coefficient matrix $\mathbf{A}_c^{\mathcal{B},a}$ accounts for the torque contribution of the chain force $f_c^{\mathcal{B},a}$ (see the Appendix in [Whiting et al. 2009]).

Compression and tension constraint. According to the limit analysis of masonry, the material can be assumed to have zero tensile strength. This condition is expressed as a non-negativity constraint on the axial components of the forces:

$$f_n^i \geq 0. \quad (2)$$

Similarly, all chains can only introduce tensile forces:

$$f_c^j \leq 0. \quad (3)$$

Friction constraints. A friction constraint is applied at all block interfaces. For each triplet of forces $\{f_n^i, f_u^i, f_v^i\}$, the two in-plane forces are constrained within the friction cone of the normal force f_n. We linearize the friction constraints with a pyramid approximation:

$$|f_u^i|,\ |f_v^i| \ \leq\ \tfrac{\alpha}{\sqrt{2}} f_n^i\ , \quad \forall\, i \in \text{ interface vertices} \quad (4)$$

where α is the coefficient of static friction. More details are given in Appendix C.

3.3 Global equilibrium analysis

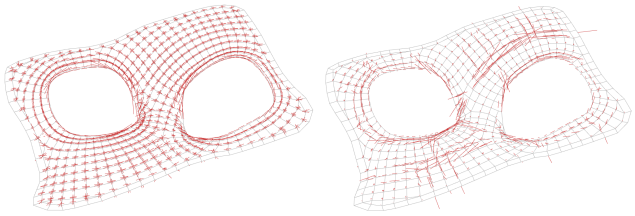
Historical self-supporting structures are highly regular and explicitly composed of *primary arches* that were constructed before the rest of the structure and used as support while building the other parts. Following this construction strategy, the expensive support material is used only for a small fraction of the structure, which then acts as a support for the other parts.

The lack of regularity and symmetry in freeform designs makes the manual identification of arches challenging, even for experts. Furthermore, freeform designs often do not contain any *exact* arches, i.e. , they do not have any subset of blocks that is in static equilibrium without any external support. However, any masonry structure contains one or more of what we term *quasi-arches*: subsets of blocks that require only a small number of chains to stand. Finding quasi-arches is even harder than finding arches, since it is a global problem that requires an understanding of how forces distribute in the entire structure. We propose a segmentation algorithm that analyzes the distribution of forces in a masonry structure and automatically extracts quasi-arches.

Combinatorial problem. Finding quasi-arches is equivalent to finding a maximal, non-trivial subset of blocks that can be removed from a masonry structure without collapsing the remaining part. Determining which blocks can be removed is a hard and ill-posed combinatorial problem, since we want to avoid the trivial solution which contains no blocks. A brute-force approach is infeasible, since testing for the validity of all possible construction sequences has exponential complexity. In the following section, we describe our proposed approach that makes the discovery of quasi-arches a tractable problem.

Continuous relaxation. Equations (1), (2), (3) and (4) define a convex constraint set containing all the possible internal force distributions for which the structure is in static equilibrium. By using a variational method, we can explore this constrained space using a sparsity inducing functional to find valid distributions that tend to concentrate the internal force on a small subset of interfaces and thus indicate potential quasi-arches in the model.

To convert the combinatorial problem into a computationally tractable continuous problem, we model only the side effects of removing a block. Each removal implies that the forces on its interfaces should be zero. We continuously relax this condition using a block-sparsity approach that searches for an equilibrium solution that can be explained with the majority of the forces concentrated on a small subset of the interfaces. Note that the trivial solution does not satisfy the equilibrium constraints (1), since they prohibit all interfaces and chains of a block to be zero.



(a) a force solution (b) forces with sparsity objective

Figure 7: *Global analysis on a synthetic example. Force vectors (red lines) indicate the magnitude of forces between blocks.*

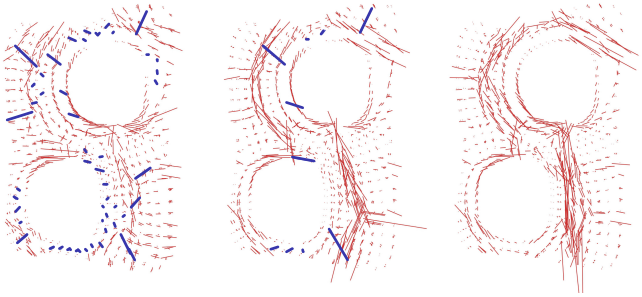


Figure 8: *The parameter λ controls the tradeoff between stable regions and introduced chains. Chain forces are shown as blue lines. From left to right, $\lambda = 0.15, 0.12, 0.06$.*

To induce sparsity on the interface forces, we first introduce a new scalar variable $s_{\mathcal{I}}$ for each interface $\mathcal{I}$ that bounds the magnitude of the resultant of the forces acting on it. We minimize the number of non-zero $s_{\mathcal{I}}$ and f_c solving a L_p-relaxation (Appendix A) of the following optimization problem with an iterative reweighting scheme (Appendix B) :

$$\min_{f_u f_v f_n f_c} (1-\lambda) \sum_{\mathcal{B}} \sum_{a} \mathbb{I}_0(f_c^{\mathcal{B},a}) \; + \; \lambda \sum_{\mathcal{I}} \mathbb{I}_0(s_{\mathcal{I}}) \qquad (5)$$

$$\text{s.t. } (1), (2), (3), (4), \qquad (6)$$

$$s_{\mathcal{I}} \geq \Big(\sum_{i \in \mathcal{I}} \|\mathbf{f}^i\|^2 \Big)^{1/2} \qquad (7)$$

where $\mathbb{I}_0(x) = 1$ for $x = 0$, $\mathbb{I}_0(x) = 0$ otherwise. We use $\lambda = 0.06$ in all our experiments. The functional contains two sparsity-enforcing terms: the first minimizes the number of introduced chains, while the second minimizes the number of used interfaces, see Figure 7. The minimization of the auxiliary variables $s_{\mathcal{I}}$ together with the bound from below imply equality for Equation (7). At a minimum, $s_{\mathcal{I}}$ therefore is equal to right-hand side, which is the resultant of the forces per interfaces. Note that the equilibrium constraints are guaranteed to be satisfied after the minimization since they are enforced as hard constraints. The tradeoff between the number of stable regions and chains used is controlled by a single parameter $\lambda \in [0, 1]$ (Figure 8). For each input model, we normalize the variables by choosing the material density such that the average length of the blocks' gravity forces is one. Our constraints are scale-independent with respect to density.

Quasi-arch extraction. Despite the sparsification, the weight of each block must still be redirected through the assembly down to the boundary or up along the chains to satisfy equilibrium equations. This naturally leads to a concentration of forces along linear subsets of blocks, even if the minimization is not explicitly trying to concentrate the force distribution in connected components.

We extract the quasi-arches from the force distribution using a flood-fill approach restricted to grow across interfaces whose maximum

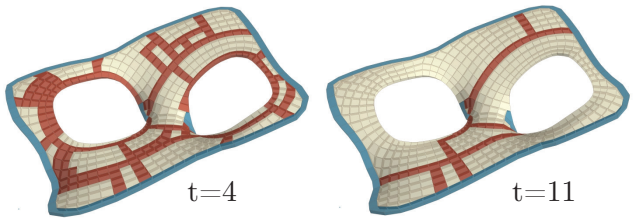


Figure 9: *Extracting quasi-arches. This figure shows the quasi-arches extracted with two thresholds on the maximal normal component of interface forces.*

of the normal component of its forces is above an user-defined threshold t (Figure 9). A quasi-arch is a connected component found by the flood-fill procedure that connects two supported blocks. The following steps extract connected arches and erode dense regions after the flood-fill: We reject a quasi-arch if it contains less than two support blocks, or if the bounding box diagonal of the centroids of support blocks is more than 10 times smaller than that of the whole model including anchor points. We then remove singly-connected blocks which are not supported. After minimizing Equation (5), the extraction can be computed in real-time, allowing the user to manually choose a value for t based on visual feedback.

Remark. The masonry crack prediction of Fraternali [2010] bears some similarity with our quasi-arch extraction: The method applies an iterative scheme to jointly find a discrete stress surface and a compressive stress distribution explaining its equilibrium via a thrust network approach, then predicts cracks in regions of zero or uniaxial compressive stress. In contrast, our quasi-arch extraction explicitly optimizes for sparse forces over all valid equilibrium solutions.

3.4 Construction sequence

We want to find a work-minimizing sequence to assemble each part of the structure using the global equilibrium analysis. To reduce the complexity of this combinatorial problem, we use a greedy approach that inserts a block at a time, taking local decisions. The work required for a certain step depends on the previous and successive states and it is thus impossible to minimize locally. Observing that the maximal work is bounded from above by the number of chains used, we opt for a strategy that directly minimizes the chains used, indirectly minimizing the work.

The detected quasi-arches are constructed first by restricting the greedy approach to only consider the blocks of the quasi-arches. After this stage, each remaining stable region is constructed using the same strategy, one at a time, starting with the region containing the most blocks. If the greedy approach fails to find a valid next block to insert, the next region is added to the candidate blocks. This heuristic drastically reduces the search space and consequently the computation time (5 times faster on Figure 3) and can reduce the work of the resulting sequence.

Next block insertion. Our algorithm attempts to independently insert each block that is connected to a support part or to any other block. For every inserted block, we solve the minimization problem in Equations (5) and (6), with $\lambda = 0$. If no equilibrium state can be found, we discard the configuration. We apply this procedure for all candidate blocks and select the valid configuration with the lowest cost. The cost is given by the L_p-relaxation of Equation (5), see Appendix A. Directly using Equation (5) would potentially assign the same cost to many candidates. In Figure 10, we show the cost for inserting each of the candidate blocks in a specific intermediate configuration.

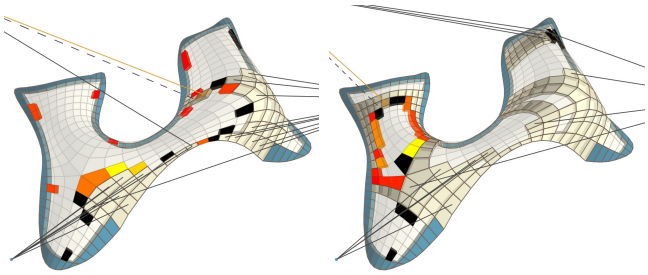


Figure 10: *At each construction step we test the work cost for inserting an additional block and we select the one with minimal energy. We highlight in black that blocks for which the optimization failed to find a force distribution that satisfies the equilibrium constraints.*

Optimization and chain pruning. To simplify the construction process, we only allow a chain to be introduced on newly inserted blocks, that is, whenever a chain is removed from a block, we do not allow our optimization to insert it again. In addition to simplifying construction, this heuristic also reduces the problem size. Note that we disable chain pruning for the comparisons with the trivial z-ordered sequences, since the pruning might prevent the z-ordering strategy to find a valid sequence.

Trivial filling strategy failure. We demonstrate on many examples that our approach is efficient and greatly reduces the work required to assemble the sequence. We show in Figure 3 a comparison between our approach and a height-ordered filling approach: our method requires around 50% less additions and removals of chains.

3.5 Practical constraints

In practice, the anchors cannot support arbitrary forces; the chains will break if the tension is extreme. We therefore add constraints to our algorithm that account for bounds on chain forces. In addition, we introduce geometric and frictional safety factors to account for unavoidable fabrication inaccuracies as described below.

Anchor bound. We conservatively bound the maximal force acting on an anchor a by adding a linear inequality constraint on the sum of the involved forces:

$$\sum_{B} f_c^{\mathcal{B},a} > -f^{max} \tag{8}$$

The condition is linear because we only sum the magnitude of the forces (all negative, see Equation 3), which are exactly the variables that we use in the optimization to represent the chain forces. Note that the bound is strict and our linearization is a conservative estimate that does not take into account possible force cancellations. We show an example of a construction state with and without bounds on the anchors in Figure 11.

Chain bound. Restricting the chain forces is a simple box constraint on the corresponding variables $f_c^{\mathcal{B},a}$, which directly represents its magnitude:

$$f_c^{\mathcal{B},a} < f_c^{max} \tag{9}$$

Manufacturing tolerances and safety factors. To account for manufacturing imprecisions and to incorporate a safety factor in the construction we introduce a geometric tolerance, to prevent torque failure, and a friction tolerance to prevent sliding failures. The former is achieved by uniformly downscaling the interface around the average of its vertices [Heyman 1995], effectively reducing the space of stable equilibrium configurations and forcing our algorithm to introduce more chains. The second tolerance is on the friction parameter, which directly reduces the feasible space by attenuating the magnitude of the friction forces. For all our experiments, we used a 10% geometric tolerance and we conservatively set the friction coefficient α to 0.6, which is approximately 10% lower than the value we experimentally measured on our 3D printed model.

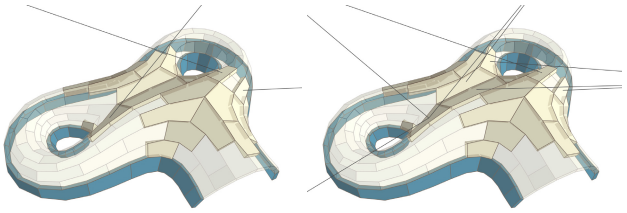


Figure 11: *Adding a bound on the maximal force that an anchor can support generates a sequence that distributes the chain forces more evenly by adding additional chains. Left: Unbounded solution using 3 chains with max. f_c = 7.1 and max. anchor bound 7.1. Note the max. chain is the only one connected to the max. anchor. Right: Solution with 8 chains bounded by 3.5 and anchors bounded by 5. For details please refer to Equations (8) and (9).*

Intersecting chains. To prevent chains from intersecting blocks during the construction sequence, we optimize only over the chains that do not intersect with any of the blocks. This set of chains is determined in a preprocessing step.

4 Results

We tested our algorithm on self-supporting surfaces designed with [Rippmann et al. 2012] (Figures 12, 14), [Panozzo et al. 2013] (Figures 1, 9), [Vouga et al. 2012] (Figures 4, 13, 11), [Liu et al. 2013] (Figure 10) and [de Goes et al. 2013] (Figure 3).We provide full construction sequences for all our results as short movie clips in the supplementary material. All our experiments were performed on a quad-core Intel i7 processor using the multi-threaded conic solver in the MOSEK optimization library [Mosek 2014]. Statistics on the datasets and computation times are provided in Table 1. The optimization requires from a few minutes to a couple of hours, depending on the number of blocks. The computation time of the global analysis, in our largest models below 15 seconds, is negligible w.r.t. the sequence optimization. This comes without surprise, since we only solve a slightly bigger conic problem once, while the sequence needs many of them.

Large scale simulations. We test our algorithm on a complex self-supporting model designed using RhinoVault [Rippmann et al. 2012] (Figure 12) and tessellated with a manually-designed staggered pattern.

To stress test our approach, we create construction sequences for two models tessellated with quads in Figure 4 and 13. These cases are particularly challenging because the lack of interlocking between the blocks makes these structures prone to friction failures.

Self-supporting puzzle. To demonstrate the flexibility of our algorithm, we designed the small self-supporting model in Figure 14 using RhinoVault [Rippmann et al. 2012]. We used our algorithm to find a construction sequence with an upper bound of three blocks supported by chains at any given time. The conditions have been enforced by rejecting all the construction states where more than three blocks were connected to anchors. The construction sequence is the solution to the puzzle, which allows to build it with four hands: three to simulate the chain forces and one to insert the pieces. Our algorithm opens interesting possibilities to design complex 3D

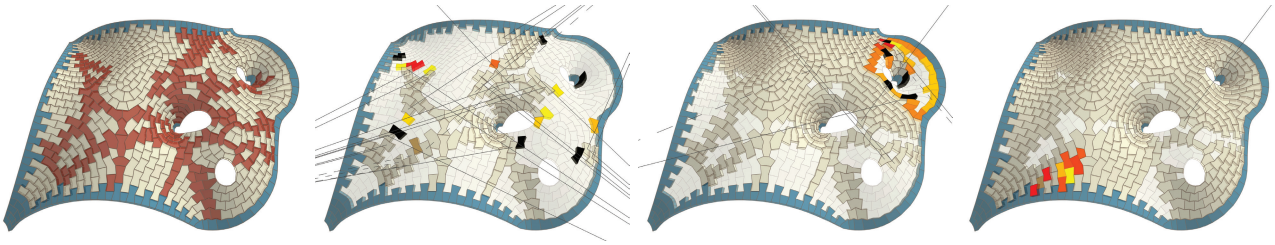


Figure 12: *A large concert hall designed with RhinoVault is constructed using a sequence generated by our algorithm. The entire structure is made of hexagonal blocks placed using an interleaved layout typical of masonry constructions.*

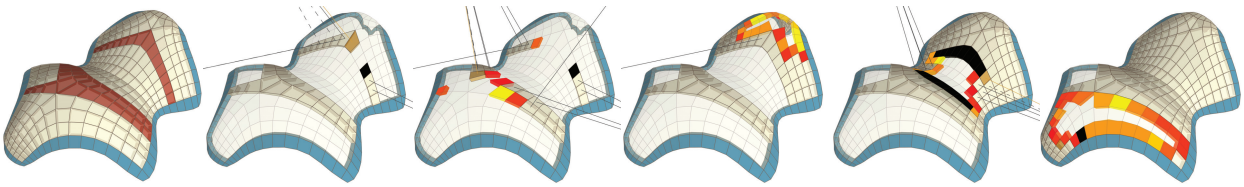


Figure 13: *A freeform self-supporting structure designed with [Vouga et al. 2012]. We tessellated the surface with quadrilateral elements, which have no interlocking and are thus prone to sliding failures, requiring a considerable number of supporting chains to be stable in all intermediate stages.*

self-supporting collaborative puzzles, which we plan to explore in future works.

Validation via small-scale models. Since masonry is a problem of stability rather than stresses [Heyman 1995], scaled block models can actually be used as structural models [Zessin et al. 2010; Van Mele et al. 2012]. We validate our algorithm by 3D printing the blocks of a masonry structure designed with [Panozzo et al. 2013] and using metal hooks and sewing string to model the chains. We used our algorithm to generate the construction sequence and then physically constructed the model following all steps (Figure 15). We provide in the additional material the full sequence of photographs for each construction step, validating our simulation results. During the construction, we observed that the chains predicted by our algorithm are always in tension, suggesting that the equilibrium model we use accurately predicts the forces acting on the structure and does not introduce redundant chains.

Restoration of historical buildings. Restoring an existing masonry building is a difficult task, since it is not possible to remove and replace blocks without risking a structural failure. Our method

Model	#B	#C	#A	t	Avg. #C	Avg. ΔC	Time
Figure 1	61	194	4	8	1.4	1.8	2
Figure 3	237	612	4	3.1	9.5	1.2	20
Figure 4	580	1652	4	4.6	4.9	1.2	249
Figure 8	203	1194	6	5	5.7	1.2	15
Figure 10	416	1127	4	0.9	9.4	1.1	95
Figure 11	128	306	4	0.6	2.0	0.5	9
Figure 12	588	2030	4	4.6	10.1	0.8	243
Figure 13	280	666	4	2.7	2.5	0.9	41
Figure 14	12	42	4	1	1.8	2.0	0.2

Table 1: *Table with statistics for our results. From left to right: number of blocks, number of chains in the optimization, number of anchors, threshold parameter for the quasi-arch extraction, average number of used chains, average number of chain changes and computation time in minutes.*

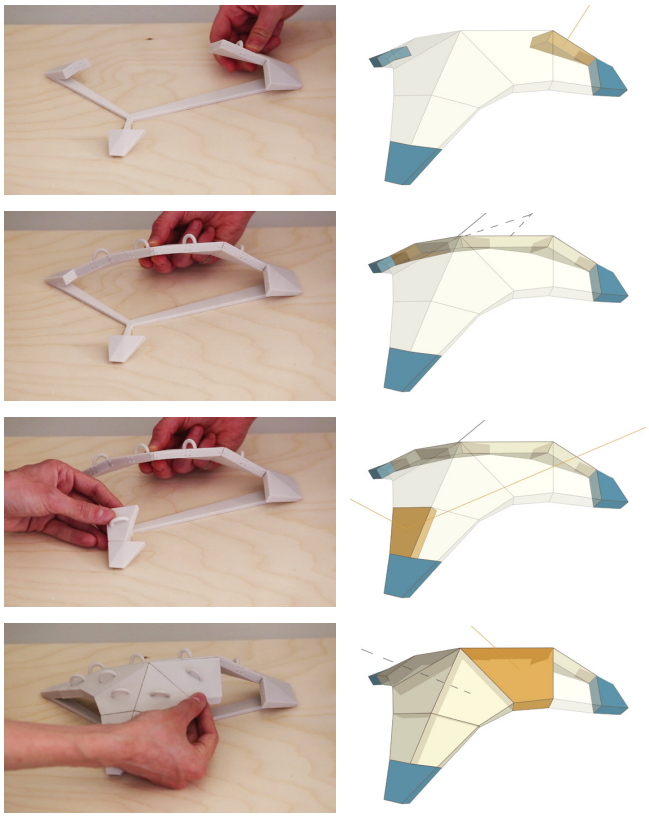


Figure 14: *Our algorithm can be used to design challenging physical puzzles.*

can be used to tackle this difficult problem, as shown in Figure 16. We optimize for a new equilibrium solution that does not contain the blocks we want to replace. The optimization redistributes the internal forces, and decides which chains should be inserted to guarantee the stability of the structure. After the chains are inserted, the blocks can be safely removed and restored. Note that intrusive techniques may be required such as drilling holes in the stones or gluing lightweight hooks/anchors.

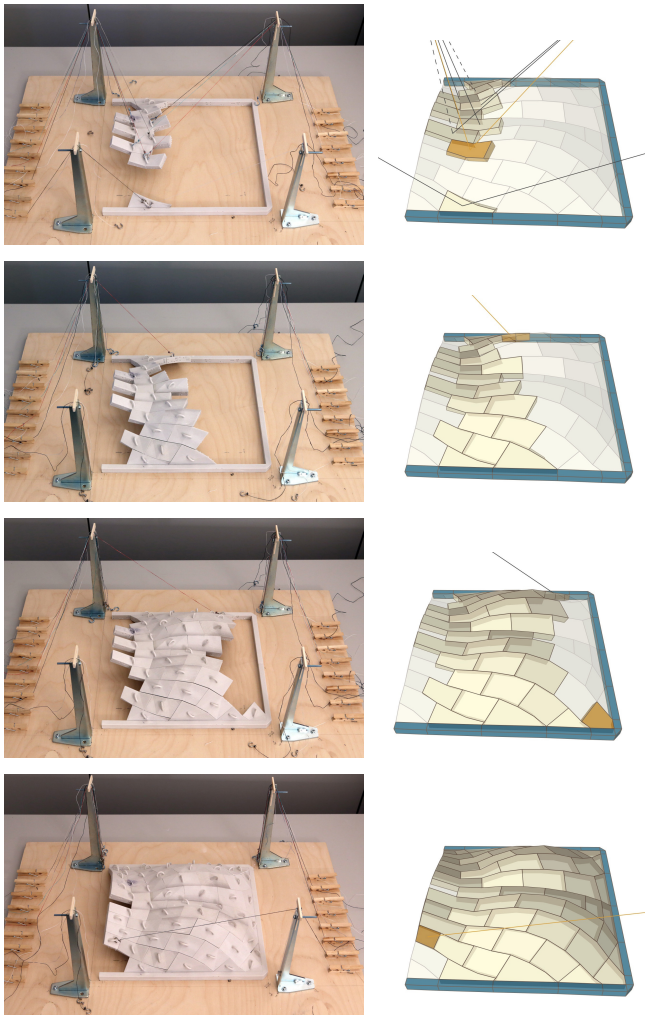


Figure 15: *We validate our algorithm by constructing a masonry structure using our optimized work-minimizing construction sequence.*

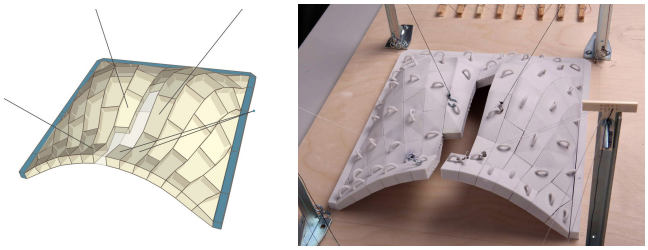


Figure 16: *Our algorithm finds a sparse set of chains that guarantee stability even after the removal of a subset of the blocks. This can be applied to restoration of masonry structures.*

Evaluation of the safety factors. We designed an experiment to evaluate the accuracy of our safety factors and the equilibrium model we use. The global analysis step finds a quasi-arch in the middle of the structure only if we do not introduce a safety factor (Figure 17). Adding a friction and safety factor of 1% is sufficient to make the quasi-arch unstable and forcing our algorithm to introduce a chain. We reproduced this case with our 3D printed model, and verified that the arch is indeed extremely close to be in equilibrium, but cannot stand safely without a chain due to a torque failure.

5 Limitations and concluding remarks

We presented a method to assemble self-supporting structures using a sparse set of chains instead of a dense formwork. Our algorithm can process models generated with any of the existing design methods and can incorporate practical construction constraints in the optimization. We assume the construction site, and the masonry structure are given as input, and we focus on optimizing a valid construction sequence: An interesting venue for future work is a relaxation of this problem, where the algorithm is allowed to alter the anchor and hook placements. This would greatly increase the solution space and can further reduce the construction cost, but is extremely challenging since moving hooks and anchors has a global effect on all the construction states. Additionally, surface shape and/or tessellation could also be optimized.

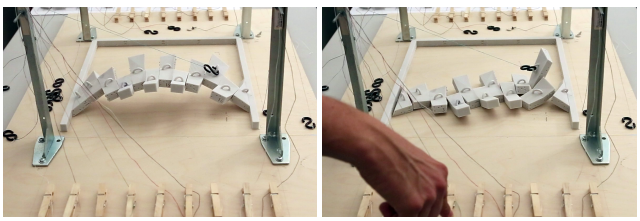
Our greedy sequence optimization may fail when none of the candidate blocks can be inserted. We handled this case by manually perturbing the anchor positions to seek a possible solution. In particular, we moved the anchors to reduce the intersections between candidate chains and blocks, expanding the solution space. Relocating or adding anchor points automatically will be an interesting venue for future work. Alternatively, a backtracking strategy could be employed to further explore the space of valid sequences.

Explicitly maximizing temporal smoothness on the (in)active chains could theoretically help our method to reduce work. This maximization is hard to implement in our framework since the chain force magnitudes can vary substantially between states, while the corresponding chain stays active. A simple smoothness term on the magnitudes could therefore lead to denser solutions, producing sequences needing more work.

The static equilibrium analysis we use is based on the lower bound theorem [Heyman 1995] which assumes hinging failure between blocks. While sliding failure is not accounted for, we guarantee existence of a feasible static equilibrium solution which satisfies the friction cone constraint. Further, we use a conservative coefficient of friction. Stability of masonry structures under sliding is an active area of research. Hinging is predominantly considered the limiting constraint in masonry analysis, particularly for arched and domed structures [Livesley 1992].

More research will be required to address the additional constraints of large-scale constructions sites, but we did validate our algorithm on a small scale, 3D printed models. We demonstrated that our algorithm can be used to design interesting physical puzzles, using hands instead of chains.

In our 3D printed models, we currently use small registration spheres (hemispheres added to the interface, respectively carved out from the interface in contact) to ensure that the construction is precise. They are not necessary but they greatly simplify construction since exact block alignment is difficult at a small scale. However, we found that small errors in chain length and fabrication technology



(a) stable with single chain (b) collapse with chain released

Figure 17: *This quasi-arch needs a chain to be stable. When the chain is loosened, the arch collapses due to a torque failure.*

(i.e. imbuing with glue powder-printed 3D blocks) can quickly sum up in an error of a few millimeters, making the construction difficult. We believe that a better locking mechanism between the blocks could be introduced to ameliorate this problem, or a computer vision system could be developed to help the exact placement of each block. Errors with fabrication tolerance would be negligible at full scale. The chain stretching is proportional to the load and dependent on the cables' properties (cross-sectional area and axial stiffness), and can be computed precisely. In a real scenario, the cables should be adjusted after each construction step to compensate for the changing load. In our 3D printed models, we assumed chains do not stretch and adjusted the chain length manually when first introduced.

The quasi-arches have an unexplored advantage: They might allow using full, curvilinear formwork (from below) to fix only the quasi-arches, decoupling the equilibrium of each stable region. The stable regions could then be constructed in parallel, drastically speeding up the construction time.

The C++ source code of our implementation is available for download at `http://lgg.epfl.ch/selfassembly`.

Acknowledgements

The authors wish to thank: Hao Pan and Xiaoming Fu for providing their source code and support. Etienne Vouga, Fernando de Goes, Ramon Weber and Matthias Rippmann for providing datasets. Bailin Deng, Andrea Tagliasacchi and Sofien Bouaziz for inspiring discussions. This research was supported in part by the SNF Grant (200021-137626) and received funding from the European Research Council under the European Union's 7th Framework Programme (FP/2007-2013)/ERC Grant Agreement 257453, ERC Starting Grant COSYM and ERC Starting Grant iModel (StG-2012-306877).

References

AGRAWALA, M., PHAN, D., HEISER, J., HAYMAKER, J., KLINGNER, J., HANRAHAN, P., AND TVERSKY, B. 2003. Designing effective step-by-step assembly instructions. *ACM Trans. Graph. 22*, 3, 828–837.

BARTON, M., POTTMANN, H., AND WALLNER., J. 2014. Detection and reconstruction of freeform sweeps. *Comput. Graph. Forum 33*, 2, 23–32.

BICKEL, B., BÄCHER, M., OTADUY, M. A., LEE, H. R., PFISTER, H., GROSS, M., AND MATUSIK, W. 2010. Design and fabrication of materials with desired deformation behavior. *ACM Trans. Graph. 29*, 4, 63:1–63:10.

CIGNONI, P., PIETRONI, N., MALOMO, L., AND SCOPIGNO, R. 2014. Field-aligned mesh joinery. *ACM Trans. Graph. 33*, 1, 11:1–11:12.

COROS, S., THOMASZEWSKI, B., NORIS, G., SUEDA, S., FORBERG, M., SUMNER, R. W., MATUSIK, W., AND BICKEL, B. 2013. Computational design of mechanical characters. *ACM Trans. Graph. 32*, 4, 83:1–83:12.

DAVIS, L., RIPPMANN, M., PAWLOFSKY, T., AND BLOCK, P. 2012. Innovative funicular tile vaulting: A prototype in switzerland. *The Structural Engineer 90*, 11, 46–56.

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

DEJONG, M. J. 2009. *Seismic Assessment Strategies for Masonry Structures*. PhD thesis, MIT.

DREW, J., 2013. United Lock-Block Ltd. `http://www.lockblock.com/`.

EIGENSATZ, M., KILIAN, M., SCHIFTNER, A., MITRA, N. J., POTTMANN, H., AND PAULY, M. 2010. Paneling architectural freeform surfaces. *ACM Trans. Graph. 29*, 4, 45:1–45:10.

FALLACARA, G. 2012. *Stereotomy: Stone Architecture and New Research*. Presses Ponts et Chaussées.

FITCHEN, J. 1961. *The Construction of Gothic Cathedrals: A Study of Medieval Vault Erection*. University of Chicago Press.

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

FU, C.-W., LAI, C.-F., HE, Y., AND COHEN-OR, D. 2010. K-set tilable surfaces. *ACM Trans. Graph. 29*, 4, 44:1–44:6.

HANSMEYER, M., AND DILLENBURGER, B., 2014. Digital grotesque. http://www.digital-grotesque.com/.

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

HILDEBRAND, K., BICKEL, B., AND ALEXA, M. 2012. Crdbrd: Shape fabrication by sliding planar slices. *Comput. Graph. Forum 31*, 2, 583–592.

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

LIVESLEY, R. 1992. A computational model for the limit analysis of three-dimensional masonry structures. *Meccanica 27*, 3, 161–172.

LO, K.-Y., FU, C.-W., AND LI, H. 2009. 3D Polyomino puzzle. *ACM Trans. Graph. 28*, 5, 157:1–157:8.

MOSEK, 2014. Mosek. `http://www.mosek.com`.

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

RAMAGE, M. H., OCHSENDORF, J., RICH, P., BELLAMY, J. K., AND BLOCK, P. 2010. Design and construction of the Mapungubwe national park interpretive centre, South Africa. *African Technology Development Forum 7*, 1, 14–23.

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

SCHWARTZBURG, Y., AND PAULY, M. 2013. Fabrication-aware design with intersecting planar pieces. *Comput. Graph. Forum 32*, 2, 317–326.

SINGH, M., AND SCHAEFER, S. 2010. Triangle surfaces with discrete equivalence classes. *ACM Trans. Graph. 29*, 4, 46:1–46:7.

SKOURAS, M., THOMASZEWSKI, B., COROS, S., BICKEL, B., AND GROSS, M. 2013. Computational design of actuated deformable characters. *ACM Trans. Graph. 32*, 4, 82:1–82:10.

SONG, P., FU, C.-W., AND COHEN-OR, D. 2012. Recursive interlocking puzzles. *ACM Trans. Graph. 31*, 6, 128:1–128:10.

STAVA, O., VANEK, J., BENES, B., CARR, N., AND MĚCH, R. 2012. Stress relief: Improving structural strength of 3D printable objects. *ACM Trans. Graph. 31*, 4, 48:1–48:11.

UMETANI, N., AND SCHMIDT, R. 2013. Cross-sectional structural analysis for 3D printing optimization. In *SIGGRAPH Asia 2013 Technical Briefs*.

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, 87:1–87:11.

WANG, W., WANG, T. Y., YANG, Z., LIU, L., TONG, X., TONG, W., DENG, J., CHEN, F., AND LIU, X. 2013. Cost-effective printing of 3D objects with skin-frame structures. *ACM Trans. Graph. 32*, 5, 177:1–177:10.

WENDLAND, D. 2009. Experimental construction of a free-form shell structure in masonry. *International Journal of Space Structures 24*, 1, 1–11.

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

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.

XIN, S., LAI, C.-F., FU, C.-W., WONG, T.-T., HE, Y., AND COHEN-OR, D. 2011. Making burr puzzles from 3D models. *ACM Trans. Graph. 30*, 4, 97:1–97:8.

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

A L_p-Relaxation

Minimizing Equation (5) is a hard, discrete problem. We therefore use a continuous L_p-relaxation ($0 < p < 1$) to approximate the minimization by

$$\min_{x_i} \sum_i w_i (r_i |x_i|^p) \quad \text{s.t. } (6), (7), \tag{10}$$

where r_i are additional, dynamic weights used for iterative reweighting as describe in the next section. w_i are weights of the discrete problem, e.g. $1-\lambda$ and λ in Equation (5). We experimentally found that $p = 0.05$ produces the least work, see Table 2.

p	0.001	0.01	0.025	0.05	0.075	0.1	0.25	0.5	0.75	1
#C	7.36	6.51	3.51	1.28	1.16	1.31	2.05	2.84	8.03	18.33
ΔC	1.41	1.34	1.25	1.02	1.15	1.21	1.31	1.41	1.41	3.25

Table 2: *Average number of chains and chain changes for sequences of the model in Figure 1 computed with different parameter p. The sequences were computed without quasi-arches to isolate the influence of the parameter. Note that $p = 1$ requires no reweighting.*

B Iterative reweighting

Iterative reweighting strategies convert non-linear L_p-minimization problems into a series of linear or quadratic problems. We choose an *iterative reweighted L_1-minimization (IRL1)* to minimize Equation (10), in particular because each subproblem is a conic program that can be solved globally optimal. Suppose $\hat{x}_i$ is a minimum of Equation (10) with $r_i = 1$. Then $\hat{x}_i$ is also a minimum of the conic program in Equation (10) with $p = 1$ and $r_i = \hat{r}_i(\hat{x}_i)$:

$$\hat{r}_i(x_i) = \frac{1}{|x_i|^{1-p} + \epsilon_r} \tag{11}$$

for $\epsilon_r = 0$ and $\hat{r}_i(0) = \infty$. In practice we use $\epsilon_r = 10^{-5}$ to avoid division by zero. However, we do not know $\hat{x}_i$. According to *IRL1* we iteratively estimate $\hat{r}_i^t$ from the previous solution $\hat{r}_i^t := \hat{r}_i(\hat{x}_i^{t-1})$, leading to the following series of conic problems:

$$\hat{x}_i^t := \operatorname*{argmin}_{x_i} \sum_i w_i (\hat{r}_i^t |x_i|) \quad \text{s.t. } (6), (7), \tag{12}$$

starting with $\hat{r}_i^0 := 1$. Note that the modulus can be replaced with the proper sign since all the variables subject to minimization are constrained to be either non-negative or non-positive according to Equation (2) and (3). We perform 5 iterations in our experiments. The resulting chain forces however are rarely precisely zero due to the relaxation and numerical precision. We therefore choose a parameter $\epsilon_c = 10^{-8}$ below which we consider a chain to inactive. After minimizing Equation (12) we solve the first iteration again, this time setting variables below ϵ_c to zero: In case we cannot find a solution, we reject the candidate block. Otherwise our construction sequence optimization picks the candidate minimizing Equation (10) with $r_i = 1$.

C Friction cones

Cone constraints of a conic program would be ideal to model the friction cone. In Equation (5), with $\lambda > 0$, some variables would then be part of multiple cone constraints, which is not allowed in the standard conic program formulation. We therefore use the conservative pyramidal approximation of Equation (4) for quasi-arch extraction, but use cone constraints for friction during the sequence optimization. The standard conic program does only allow cones of the form of Equation (7), but the friction cone needs a scalar coefficient s_K on the variable on the left-hand side. We therefore transform the conic program with scaled cones:

$$\begin{aligned} &\min_x w^T x \quad \text{s.t. } b_l \le x \le b_u, Ax \le b, Cx = d, \\ &\forall K \in \text{Cones}: \quad s_K x_{K_0} \ge \Big(\sum_i \|x_{K_i}\|^2 \Big)^{1/2} \end{aligned}$$

where x_{K_0} is the left-hand side variable in cone K, and x_{K_i} summands of the right-hand side, into a standard conic program with the same minima:

$$\begin{aligned} &\min_y (w^T D)^T y \quad \text{s.t. } Db_l \le y \le Db_u, \\ &(AD^{-1})\, y \le b, \quad (CD^{-1})\, y = d, \\ &\forall K \in \text{Cones}: \quad x_{K_0} \ge \Big(\sum_i \|x_{K_i}\|^2 \Big)^{1/2} \end{aligned}$$

where D is a diagonal matrix where $D_{ii} = s_K$ if x_i appears in the left-hand side of cone K, $D_{ii} = 1$ otherwise. Note that this transformation is only meaningful if each variable only appears in the left-hand side of cones with the same s_K.