Position-Based Tensegrity Design

NICO PIETRONI, CNR-ISTI
MARCO TARINI, CNR-ISTI and Univ. Insubria
AMIR VAXMAN, Universiteit Utrecht
DANIELE PANOZZO, New York University
PAOLO CIGNONI, CNR-ISTI

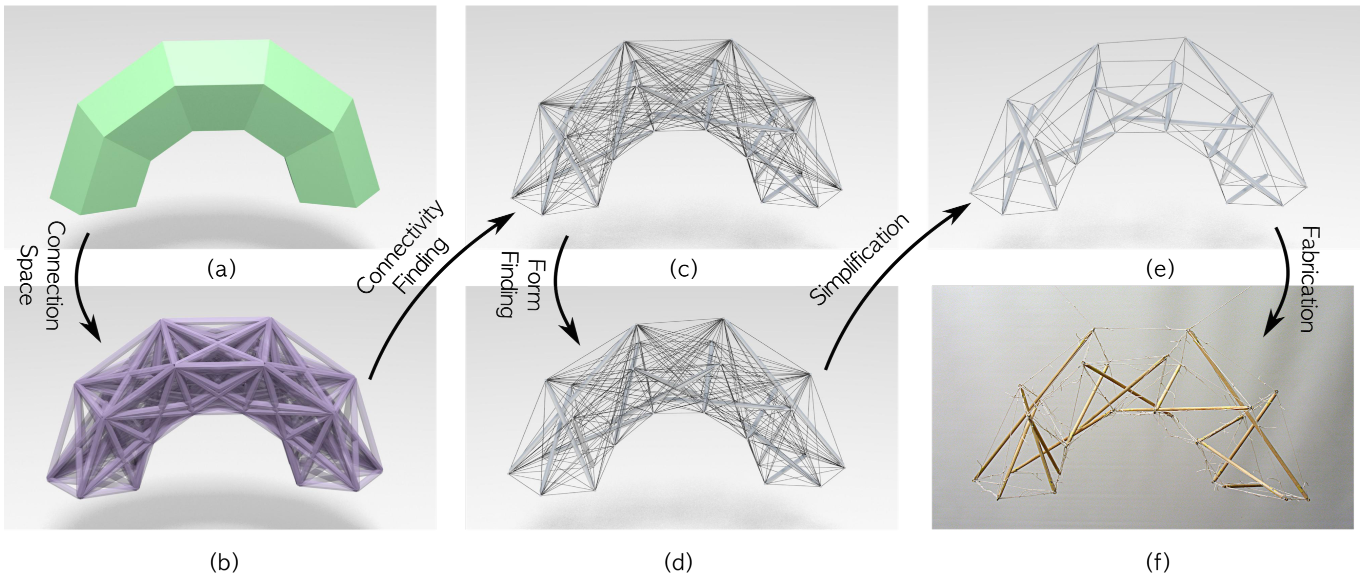


Fig. 1. (a) An input mesh; (b) The derived graph; (c) Edges are divided into struts and cables of a pure tensegrity structure; (d) Stable shape after form-finding; (e) Simplified, and still stable, shape after removing redundant cables; (f) The fabricated model.

We propose a novel framework for the computational design of tensegrity structures, which are constructions made of struts and cables, held rigid by continuous tension between the elements. Tensegrities are known to be difficult to design—existing design methods are often restricted to using symmetric or templated configurations, limiting the design space to simple constructions. We introduce an algorithm to automatically create free-form stable tensegrity designs that satisfy both fabrication and geometric constraints, and faithfully approximate input geometric shapes. Our approach sidesteps the usual force-based approach in favor of a geometric optimization on the positions of the elements. Equipped with this formulation, we provide a design framework to explore the highly constrained space of tensegrity structures. We validate our method with simulations and real-world constructions.

CCS Concepts: • **Computing methodologies** → **Mesh models**;

This work is partially supported by the PRIN project "DSURF" (2015B8TRFM), NSF CAREER award (1652515), and a gift from Adobe Research.

0730-0301/2017/11-ART172 $15.00
https://doi.org/10.1145/3130800.3130809

Additional Key Words and Phrases: Tensegrity, Architectural Geometry

ACM Reference format:
Nico Pietroni, Marco Tarini, Amir Vaxman, Daniele Panozzo, and Paolo Cignoni. 2017. Position-Based Tensegrity Design. *ACM Trans. Graph.* 36, 6, Article 172 (November 2017), 14 pages.
https://doi.org/10.1145/3130800.3130809

1 INTRODUCTION

A tensegrity structure consists of a set of disjoint struts, tied together by cables connecting the endpoints of the struts and nothing more. The term *tensegrity* is a portmanteau of "tensional" and "integrity" and it refers to the integrity of a stable structure balanced by structural elements carrying pre-tension (cables) or pre-compression (struts) forces. These structures are known for their desirable aesthetic and structural qualities, which are explored in architecture and art. Their perceived lightness, where the heavy elements (struts) are held together only by a net of a few cables, is the distinct visual feature of these structures. Tensegrities are lightweight, while enjoying solid mechanical stability. Moreover, no welded joints are required, as struts are only connected through cables, facilitating fabrication. Notable examples are the Kennet Snelson's [Needle Tower 1968] and the [Kurilpa Bridge 2009] in Brisbane (see Figure 2). The structural principle is also known to exist naturally in

Fig. 2. Notable examples of Tensegrity structures. Left: the Kennet Snelson's Needle Tower, Image courtesy of F Delventhal (CC BY 2.0). Right: the Kurilpa Bridge in Brisbane, Image courtesy of Andrew Sutherland (CC BY-SA 2.0).

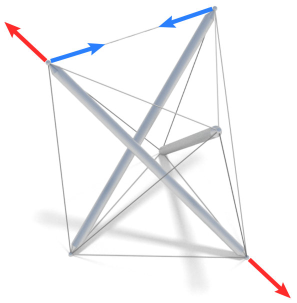
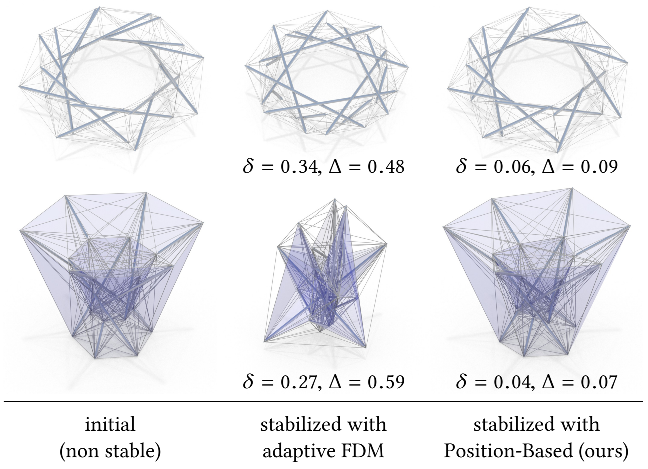


Fig. 3. On the right: a simple example of a prism in tensegrity; struts are represented by thick lines, and cables by thin lines; struts exert compression forces (red arrows), and cable tension forces (blue arrows). These internal forces alone keep the structure in equilibrium, and balance out any external force.



$\delta = 0.34$, $\Delta = 0.48$
$\delta = 0.06$, $\Delta = 0.09$
$\delta = 0.27$, $\Delta = 0.59$
$\delta = 0.04$, $\Delta = 0.07$
initial (non stable)
stabilized with adaptive FDM
stabilized with Position-Based (ours)

Fig. 4. Comparison of form-finding methods on two simple examples; δ and Δ are the average and maximum displacement of vertices (expressed as a fraction of the bounding box of the structure). Even when the adaptive Force-Density method [Zhang and Ohsaki 2006] obtains a stable configurations, the resulting shape may considerably differ from the input.

muscle, tendon, and bone structures, and therefore inspires artists and engineers to creative design by way of biomimicry.

Despite the interest from both art and engineering communities, tensegrity structures are difficult to devise, and are rarely used in novel designs. The space of possible configurations is discrete and highly-constrained, and it is difficult to create original models that are both visually interesting and structurally valid. Existing computational attempts are limited to conservative variations on stock examples, or to assembling such examples together [Gauge et al. 2014]. Known approaches, based on force-densities [Zhang and Ohsaki 2015], can be used for simple models, but produce unsatisfactory results for complex models (Figure 4), due to large differences between the optimized shape and the designer's intent.

We introduce a novel automatic framework to design tensegrities, inspired by a classical formulation for stable structures. Our framework achieves desired structural and user-defined properties, by optimizing directly on the spatial positions of the endpoints of the elements in the tensegrity. As such, we sidestep the need to explicitly estimate forces, or consider equilibrium conditions. This leads to an efficient algorithm, that can be used for different challenging design tasks, including initial connectivity setup of valid tensegrities, shape approximation, and connectivity simplification.

Our system enables designers to explore the space of tensegrity structures, without having to rely on predefined patterns. We demonstrate its practical relevance by designing multiple complex tensegrity structures and assembling two of them. We validate our algorithm by testing our designs with a digital simulation under random loads. Finally, we demonstrate the flexibility of our method, by applying it to the design of scaffolds for practical tensegrity construction, and for the removal of torque and tension from traditional truss structures.

2 BACKGROUND AND OVERVIEW

A pure tensegrity structure (or just "a tensegrity") is a set of disjoint *struts* (sometimes "bars" or "rods" in the literature) that are tied together by a set of *cables*, connected only between their endpoints.

The main structural requirement of a tensegrity structure is *stability*. Intuitively, the structure is stable when it resists attempts to be deformed, behaving as a single *rigid* object. The basic principle of a tensegrity is that the structure does so by relying on only two types of internal forces: *tension* forces from cables, and *compression* forces from struts (Figure 3). Any combination of external forces (such as gravity) must be canceled out by these internal forces.

The structure does not rely on any other internal force to maintain its shape. In particular, a cable will oppose no force if compressed; symmetrically, we do not need to assume that struts offer any resistance to elongation (even if this is the case for many choices of real-world materials). Likewise, there is no reliance on the resistance of struts or cables to bending, torsion, or external forces in any direction but along their length; the joints between struts and cables do not offer any opposing force to any attempt to change their angles. These assumptions are imposed from practical considerations about real-world realizations [Zhang and Ohsaki 2015]. For example, a cable realized with a metallic wire would be able to sustain a large load when tense, but would offer almost no resistance to compression, bending, or twisting.

Furthermore, we assume perfect inextensibility (resp., incompressibility) of cables (resp., struts). That is, we assume that even for negligible elongation from their initial *rest length*, cables would respond with tension forces that are enough to negate any external force (and likewise for compression of struts). In other words, cables and struts are modelled as asymmetric springs with extremely large "spring constants" (the ratio between the magnitude of the response force and the elongation). We note that this assumption is fully justified for a wide range of real-world materials, like steel cables, which exhibit very high Young's moduli. For example, a 1 m steel cable with 4 mm section diameter will elongate of only a few tenths of a millimeter in response to a tension force of 1000 N. Under these assumptions, our framework may safely ignore internal forces or pre-stress, which greatly simplifies the design and simulation of tensegrities. While tension forces should be accounted for during assembly, we experimentally show that for steel cables, like the ones used in the two models in Fig. 5, this is not required: their extension under load is negligible with respect to the actual fabrication errors experienced when preparing the elements.

2.1 Rigidity and Stability

Historically, both rigidity and stability have been formally modeled in many different ways, such as [Connelly 2013; Connelly and Whiteley 1996], emphasizing several subtle differences. The reader is referred to [Whiteley 1984] for an exhaustive treatment of the general case, and [Roth and Whiteley 1981] for a seminal discussion on the specific case of tensegrities. We briefly summarize the concept we use in our paper in the following.

A tensegrity structure is (locally) *rigid* [Recski 2008; Roth and Whiteley 1981; Whiteley 1989] when it cannot be continuously deformed without violating the physical constraints we discuss above. That is, every deformation attempts to either elongate cables or compress struts. This type of rigidity is deemed local as it is characterized by continuous deformations.

A structure is termed *stable* [Roth and Whiteley 1981] when any combination of external forces acting on the vertices is canceled by internal forces, i.e., an equilibrium is held with any load of external forces. This implies that a system is not only in an equilibrium, but also in a strict (local) minimum of a potential energy, rather than just a saddle or a maximum. Much of the recent past literature concentrates of enforcing stability directly (see Section 3).

Although stability is generally a stronger property than (local) rigidity, under our assumptions, the two are equivalent [Roth and Whiteley 1981; Whiteley 1984]. This equivalence is the main premise of our work: we propose a novel and practical framework where only (local) rigidity is explicitly sought, and stability as a consequence. This framework consists in a set of efficient algorithms to test *and enforce* stability, through rigidity. Subsequently, such algorithms allow us to directly optimize for connectivity, and accommodate custom constraints and desiderata, leading to the interactive design of stable tensegrity structures. As such, our entire framework is based on manipulating positions, rather than forces. In computer graphics, methods that purely manipulate positions are often termed "position-based" [Müller et al. 2007], and we thus term our approach *Position-Based Tensegrity Design*.

3 RELATED WORK

Computational design is an active discipline that is transforming architecture, industrial prototyping, and art. It allows a designer to focus on the aesthetics of the design; all other practical considerations and constraints, such as feasibility, stability, cost, assembly sequence or material usage are taken care of by an algorithm in the background. This algorithm must in turn produce a shape that deviates as little as possible from the one carefully prescribed by the designer. It is not surprising that many computational design techniques have been developed as interactive tools in the graphics community, which specializes in modeling and rendering 3D geometry. In particular, dramatic progress has been achieved in the optimal design of architectural models [Whiting et al. 2009, 2012], masonry structures [de Goes et al. 2013; Liu et al. 2013; Panozzo et al. 2013; Vouga et al. 2012], wire assemblies [Miguel et al. 2016], structures composed by interlocking planar pieces [Cignoni et al. 2014; Hildebrand et al. 2012; Schwartzburg and Pauly 2013], modular elements [Skouras et al. 2015], or planar tessellations [Bouaziz et al. 2012; Liu et al. 2011; Pietroni et al. 2015; Tang et al. 2014]. The design of tensegrity structures received little attention so far in the graphics community, with only one work [Gauge et al. 2014] to date. However, they solve a reduced problem, where they interactively design structures by assembling multiple tensegrity components together into one larger structure. On the other hand, the problem received ample coverage in the mechanical engineering and architectural community, as summarized below, which has been motivated by the many perceived structural benefits of tensegrities. We should disclaim that these structural qualities were deemed debatable within this community, e.g., in [Hanaor 2012].

Designing sound tensegrity structures is a challenging problem comprising both discrete and continuous parts: the discrete part, *connectivity design*, determines how struts and cables are joined together, while the continuous part, usually referred as *form-finding*, optimizes the positions of their endpoints in space, to enforce static properties and other constraints. Form-finding and connectivity design for generic and complex tensegrity structures is largely an open problem. In the following, we discuss how the most relevant works in the mechanical engineering community attempted these two challenges. For a more complete review, see [Juan and Tur 2008].

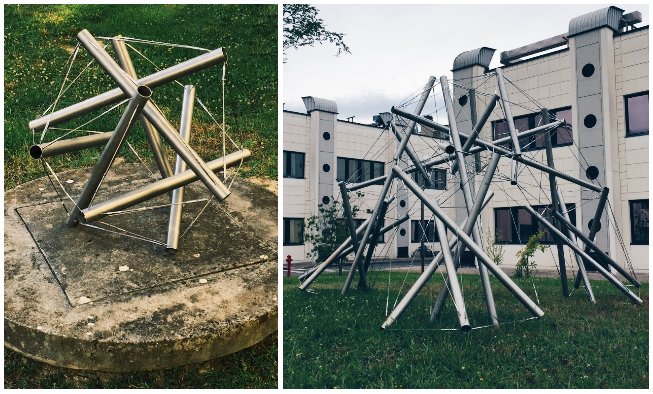


Fig. 5. Two physical tensegrity models constructed from our designs using 60 mm wide stainless steel tubes and 4 mm wide steel cables. Left: a spherical design featuring 7 struts, which is a non-trivial asymmetric variation of the popular icosahedron-based 6 struts tensegrity. Right: realization of the design shown in Figure 7-(b); see also Figure 14 and the attached video.

Working with force densities is the common property of all these works, while our approach favors a purely geometric approach.

Force Density methods. In traditional approaches, cables and struts are modeled as asymmetric springs, and are assumed to exist in a *prestress* equilibrium of tension and compression forces, respectively, regardless of any external forces (like gravity). External forces are assumed to be negligible in magnitude, compared to internal forces. A *force density* (force per length unit) is explicitly assigned to the linear elements of the structure, by solving a system for static equilibrium. Forces are aligned to the linear elements, and expressed as scalars: negative scalar represent tensions, and positive scalars represent compression. As such, the signs of the assigned force densities also determine the identity of the specific element, whether strut or cable. A structure is in static equilibrium when all the internal forces cancel out (ignoring external forces). Stability is then characterized by looking at the potential energy stored in the springs at this equilibrium state. However, a state of equilibrium (zero gradient of the energy) does not imply that the energy is at a strict minimum; it could be at a maximum, an inflection, or a non-strict minimum. By analyzing and imposing conditions on the Hessian of the potential energy, these possibilities are ruled out, and configurations that are the minimum of the energy are then denoted as *super*-stable. Unfortunately, it is computationally intensive and difficult to enforce and test such conditions on the Hessian. In our framework, we instead employ a well-known relation of stability from rigidity, that allows us to work with the positions of the elements alone. See Section 4 for details. Our work is the first computational method that uses these well-known position-based conditions to design tensegrity structures, bypassing these computationally challenging Hessian conditions altogether.

For clarity and brevity, we refer to stability (see Sec. 4.2) in lieu of what traditional force-based methods (e.g., [Tibert and Pellegrino 2011; Zhang and Ohsaki 2006, 2007]) denote as *super-stability*, since tensegrity structures that are only in an unsteady equilibrium are not useful for any practical purpose.

Form Finding. Different form-finding methods for tensegrity structures have been investigated by the mechanical engineering community; a systematic review is in [Tibert and Pellegrino 2011]. One of the conventional approaches for finding an equilibrium configuration is to find a set of force-density values that ensures a valid spatial configuration. Examples are the adaptive force-density methods described in [Estrada et al. 2006; Zhang and Ohsaki 2006]. These solutions are purely algebraic, and lead to effective algorithms that can deform struts and cables from an initial configuration into a stable configuration. However, they can introduce arbitrarily large deformations, deviating significantly from an initial prescribed shape. This makes them difficult to use in the paradigm of computational design. Our formulation sidesteps modeling forces, preferring a direct geometric approach that leads to much smaller displacements and superior user control (Figure 4). Our algorithm offers direct control over the approximation from a given shape, and obtains results that reflect the designer input much better.

Connectivity Design. The problem of finding the connection patterns of a struts and cables so that they could fit a given shape is the central requirement for creatively designing tensegrities. In [Tachi 2013], the authors present an approach that generates a tensegrity of a given polygon mesh surface using a direct face-by-face conversion driven by a predefined pattern; the positions of the nodes of this structure are then optimized to obtain an equilibrium state through a two-step optimization of both node coordinates and force densities. A modular approach was introduced in [Gauge et al. 2014], where the authors present an interactive system to design tensegrities by combining a predefined set of simple basic tensegrity blocks.

In [Paul et al. 2005], a genetic algorithm was used to create the connectivity pattern by evolution. They encoded the initial positions of the vertices, and the connectivity patterns, as the genotype. [Lobo and Vico 2010] followed a similar approach, but explored the use of more compact genotypes that can allow faster evolution. The genetic approach was also used in [Gan et al. 2015], where both the connectivity matrix and the prototype force density vector are generated by an evolutionary approach, starting only from the number of nodes. A different evolutionary approach was presented in [Rieffel et al. 2009] where the authors propose a search technique based on an L-system genotype to grammatically grow a graph representing tensegrity structure. In practice, none of these approaches offer any feasible way to control the shape of the generated structure, and therefore are not well-suited to support the design process like our approach. Moreover, due to the intrinsic exponential nature of genetic exploration, these approaches either have been tested only on very small examples (less than 10 struts for the first three genetic papers) or they generate (in the words of authors) *semi-regular* and repetitive structures [Rieffel et al. 2009].

The approach for connectivity finding that is the most relevant to the one we use in our framework is in [Ehara and Kanno 2010]. Their input is a pin-jointed structure with the specified locations of nodes, and sufficiently many candidate edges to be labelled as either cables or struts. The authors compute a labelling where they maximize the number of struts to adhere to the equilibrium condition of the nodes. In addition, they adhere to the constraint of being a tensegrity, i.e., no two struts share a node. Then, they solve a second ILP problem to remove redundant cables, in a way that does not compromise the stability. While we share some similarities in the objectives and inputs, our approach is considerably different. We introduce a form-finding step, and therefore do not need to solve the tensegrity problem in its full generality merely by connectivity finding. This decoupling allows us the freedom of catering to a wider set of objectives and constraints in the structure, through our ILP minimization process (see Section 5.1), and of including complex criteria in the design specification, such as the avoidance of specified regions of the space, the imposition of a minimal distance between struts, and others.

Global Rigidity. Our work adopts the characterization of stability from local rigidity. A much stronger structural requirement, which is sometimes addressed in related work, is *global*, rather than local, rigidity. In its general definition, a graph or a structure is globally rigid when it admits only one embedding that satisfies the constraints defined on the lengths of the elements, i.e., if it is not possible to disassemble it and then reassemble it, connecting the elements in the same topological way, in any different spatial configuration. Recently, new conceptual tools have been introduced to study and test global rigidity [Gortler et al. 2010], including ones specifically for tensegrities [Connelly 2013], where the constraints are defined as inequalities rather than equalities. Although it has important applications, global rigidity is not a strict necessity for structural design, and it is much more difficult to test or enforce; it might even needlessly limit the design space. Therefore, we concentrate on the weaker, and more feasible, property of local rigidity.

4 DESIGNING TENSEGRITY STRUCTURES

We present a flexible framework for the design of tensegrity structures. Given an input shape, and an auxiliary set of practical geometric constraints, both prescribed by a designer, our pipeline computes a *stable* tensegrity structure that approximates the intended shape as much as possible, while adhering to the constraints.

Input shape. The designer can prescribe the input shape in several ways: (1) provide a triangle mesh, that should contain the tensegrity structure in its interior, or (2) provide a point cloud, and an accompanying *collision mesh*, to signal the algorithm to keep certain areas free from struts and cables (Figure 12.a). The vertices of the mesh, or the points of the cloud, are used as the initial endpoints to the struts, and fed to our pipeline as *seed vertices.*

Geometric constraints. In many design scenarios, a tensegrity must adhere to auxiliary geometric constraints other than stability. These constraints can be quite general, like minimal distances between elements, coplanarity of a subset of the elements, and more. We describe them in detail in Section 5.4.

Pipeline. Given the input and the constraints, our pipeline proceeds as follows (Figure 1):

(1) Use **Connectivity Finding** to initialize a tensegrity structure with a set of struts and redundant cables connecting the given seed vertices (Section 5.1).
(2) Use **Constrained Form Finding** to enforce stability, while simultaneously adhering to the auxiliary geometric constraints (sections 5.2 to 5.4).
(3) Perform **Structure Simplification** to remove redundant cables (Section 5.5).

4.1 Internal Representation of the Structure

We model a tensegrity structure as a set of n nodes $P = \{p_1 \dots p_n\}$, representing the endpoints of the struts $T = \{t_1, t_1, t_3 \dots\}$, and a set of cables $C = \{c_1, c_1, c_3 \dots\}$ connecting them. Each node p_i is represented by its position $p_i = (x_i, y_i, z_i)$. Each strut in T and cable in C is denoted by an unordered pair of node indices (i, j). Node positions P implicitly determine the physical rest length of the elements (struts and cables): this length (i, j) is defined by the Euclidean distance between the two corresponding nodes p_i,$p_j \in P$.

We call $P = \{p_1 \dots p_n\}$ the *shape* of the structure. The space of possible shapes is a $3n$ dimensional vector space. The distance between two shapes Q and P is measured as the squared L2 norm between the endpoint positions $|Q - P|_2^2$.

Connectivity Constraints. For a triplet (P, T, C) to represent a pure tensegrity structure, all struts need be isolated from each other; therefore, each node in P must belong to exactly one strut in T (implying n to be even). There is a single element at most between any pair of nodes, meaning that the sets T and C must be disjoint (i.e. we disallow the superposition of a cable and a strut).

4.2 Stability from Rigidity

We use the well-known characterization of local rigidity (e.g. [Connelly and Whiteley 1996; Roth and Whiteley 1981; Whiteley 1984]), which is purely geometric, defined solely with *the positions of the endpoints* and the connectivity, and without explicitly accounting for forces or energy. Specifically, this bypasses the need to associate sign-constrained, scalar force-density values to each strut and cable.

Given two sets of cables and struts T and C, we say that shape Q is *dominated* by P, denoted as $Q \dot{\subseteq} P$, if and only if the positions in Q do not prescribe any cable to be longer, nor any strut to be shorter, than it is in P:

$$\begin{aligned} |q_i - q_j|_2 &\leqslant |p_i - p_j|_2 \quad \forall (i,j) \in C \\ |q_i - q_j|_2 &\geqslant |p_i - p_j|_2 \quad \forall (i,j) \in T \end{aligned} \tag{1}$$

intuitively, $Q \dot{\subseteq} P$ means that we could physically construct structure (Q, T, C), with nodes in positions Q, but with cables C and struts T having rest-lengths as dictated by P, and that would abide to the principle of incompressibility of struts and inextensibility of cables.

For (P, T, C) to be *locally* rigid, we need ensure that no Q with $Q \dot{\subseteq} P$ exists in a neighborhood of P. More formally, a structure (P, T, C) is stable if and only if any shape Q in an open neighborhood of P, such that $Q \dot{\subseteq} P$, is congruent (i.e., isometric) to P; note that the inequalities are then strict equalities.

An important observation is that this geometric condition is much stronger than just requiring that no single node can be relocated continuously around its current position, while the rest of the nodes are kept fixed; we also require that no subset of nodes can be simultaneously relocated into new positions near their current positions (unless this amounts to a global isometry). Examples of such movements include the rigid movement of just one strut (without elongation or shortening), or a situation where multiple, but not all, struts perform a coordinated movement (i.e. a *mechanism*). In our setting, we need to forbid any such movement as well.

5 PIPELINE STAGES

The stages of our pipeline alternate between discrete and continuous optimizations, to handle different aspects of the highly-constrained tensegrity design problem. Given an input shape from the designer, we generate a initial, non-stable guess for the connectivity of the tensegrity (Section 5.1). We define a stability testing algorithm (Section 5.2) and subsequently use it to guide the form-finding process (Section 5.3), which optimizes the positions of struts and cables without changing the connectivity. Auxiliary geometric constraints (Section 5.4) are used in the form-finding step to provide the designer with an extended control over the result. Finally, the structure is simplified, removing all redundant cables that do not contribute to the stability, making the structure as lightweight as possible (Section 5.5).

5.1 Connectivity Finding

Our first step creates a graph $G = (P, E)$, encoding the initial connectivity of the tensegrity, and devises the initial tensegrity structure (P, T, C) from it, by labelling edges E as either cables or struts.

Initialization. Recall that we admit two types of inputs: point clouds (with collision meshes to describe forbidden zones), and triangle meshes (which restrict the tensegrity to their interior). We assign the set P to be either the points of the cloud, or the vertices of the mesh. In the former case, we assign E according to the k-nearest neighbors of every vertex ($k = 16$ for the presented examples). In the latter, we connect two vertices if there is a path of up to h polygonal edges connecting them ($h = 4$ for the presented examples).

Pruning forbidden edges. The initialization can create edges that pass through the prescribed forbidden zones. For triangle meshes, we prune edges that pass outside of the mesh. For point clouds, we prune those that intersect the provided collision mesh.

Assignment of struts and cables. Our next challenge is to partition the edges E into the disjoint sets of struts T and cables C. We need to adhere to the connectivity constraints of Section 4.1, therefore our search space is that of all *maximal matchings*: for n vertices there are exactly $n/2$ disjoint edges that are chosen as T, and all others can be conservatively labeled as cables C. Note that this introduces many cables, most of which might not be necessary for stability: these cables are removed at the end of the pipeline (Section 5.5).

In addition to the connectivity constraints, we need to chose an assignment of T and C that favors stability. In general, no admissible choice results directly in a stable (P, T, C) structure (as we empirically verified by exhaustively testing all possibilities for 100 random graphs of 8 nodes). Even when one such choice exists, finding it would be a difficult combinatorial problem. Fortunately, at this stage we are only looking for an initial graph that can likely be morphed into a stable structure in next stages (Section 5.3). We thus decouple the problem of finding P from this step, where we only focus on T and C.

Selecting T. We use an Integer Linear Programming (ILP) formulation to address the matching problem. We associate a binary variable v_{ij} in $\{0, 1\}$ to each edge (i, j), where $v_{ij} = 1$ if the edge is a strut, and $v_{ij} = 0$ if it is a cable.

Connectivity constraints (Section 4.1) are enforced as n hard linear constraints for each node:

$$\forall i : \sum_{j \neq i} v_{ij} = 1. \tag{2}$$

The following linear constraints are added, to favor stability and to follow aesthetic or structural requirements:

(1) **Cutting planes.** We observe that, in any stable structure, any plane that bisects the seed vertices into two non-empty sets must be crossed by at least one strut. Otherwise, it would impossible to prevent "squeezing" of the structure perpendicularly to the plane, and thus the structure cannot be made stable (see Figure 6a–b). We sample a number of random planes p_k, and for each plane that partitions the seeds into two non empty sets, we find the set of edges E_p crossing that plane. We add an additional constraint to the ILP: $\forall (i, j) \in E_{p_k}, \sum v_{ij} \geq 1$.

(2) **Avoiding struts on boundary.** Another practical observation is that struts are rarely helpful for stability if they are tangential to the external surface, because they can only oppose compression forces. We enforce this by disallowing edges on the convex-hull to be labeled as struts.

(3) **Minimal strut-to-strut distance**. To avoid collisions, and for aesthetic purposes, we ensure a minimal distance between struts (Figure 6b–c). This will also be enforced in geometry optimization stages later, but we impose this in the initial configuration (so that smaller geometric displacements will be necessary later). Given two edges (i_0, j_0) and (i_1, j_1) whose distance is below a fixed threshold (we use 1/100 of diagonal of bounding box), we impose that at most one of them can be a strut: $v_{i_0,j_0} + v_{i_1,j_1} \leq 1$.

(4) **Symmetries.** Optionally, we can also add *symmetry constraint* to impose the struts to be selected coherently with the class of equivalence of the chosen symmetry (see Figure 7). For example, if edge (i_0, j_0) is in a reflectional symmetry with (i_1, j_1) , we impose $v_{i_0,j_0} = v_{i_1,j_1}$.

Next, we associate a coefficient $s_{ij} \in \mathbb{R}^*$ to each edge (i, j), denoting the likelihood of (i, j) to be a strut (again, after either structural of aesthetic considerations), and then solve:

$$v_{ij} = argmax \left(\sum_i v_{ij} s_{ij} \right) \tag{3}$$

Every choice of likelihood coefficient s_{ij} leads to visually different structures, and this is the main aesthetic parameter of our method (Figure 6). We propose four such choices:

(1) **Minimal length.** Minimizing length of struts with $s_{ij} = -\left|p_i - p_j\right|$ leads to lighter designs (Figure 6e).

(2) **Maximal length.** By choosing $s_{ij} = \left|p_i - p_j\right|$, we promote longer struts. As such, structures tend to appear more heavyset (Figure 6d).

(3) **Normal orientation.** We estimate the "normal" to a node by averaging the normalized directions of its incident edges. We give higher s_{ij} values to edges that are more oriented with the normals at their nodes. Hence, $s_{ij} = |d_e \cdot \overline{n}_i| + \left|d_e \cdot \overline{n}_j\right|$ where d_e is the normalized direction the edge, and $\overline{n}_i, \overline{n}_j$ are the normals at incident vertices. The effect is that struts tend to cross through the volume more directly, rather than being tangent to the structure (Figure 6c).

(4) **Random.** Assigning random s_{ij} values is another interesting option, which allows to explore the large solution space, as well as breaking undesirable symmetries (Figure 12a–c).

5.2 Stability Testing

We next introduce a procedure to determine if a given structure (P, T, C) is stable (Section 4.2). Assume P is unstable, i.e. not locally rigid [Roth and Whiteley 1981]. This means that there exists an infinitesimal displacement of nodes of P toward state Q, such that $Q \subseteq P$, and Q is not congruent to P. In our approach, we explicitly search for such a displacement. Our search is guaranteed to find one solution, if any exists. Therefore, we can declare our structure as stable if the search fails.

This is, strictly speaking, too strong a condition: a locally rigid structure could still permit an *infinitesimal* displacement, as can

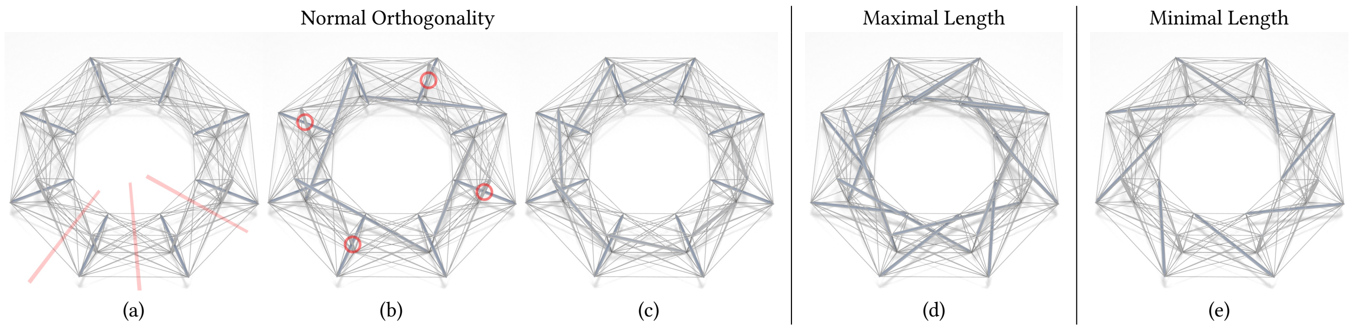


Fig. 6. Examples of Connectivity Finding results, for three different choices of likelihood criterion, as reported on top. (a) only *Connectivity* constraints are enforced; (b) *Cutting-Planes* constraints are added—three planes violating this constraints are shown in (a); (c-d-e) *Minimal Strut-to-Strut Distance* constraint added—violations of this constraint are circled in (b).

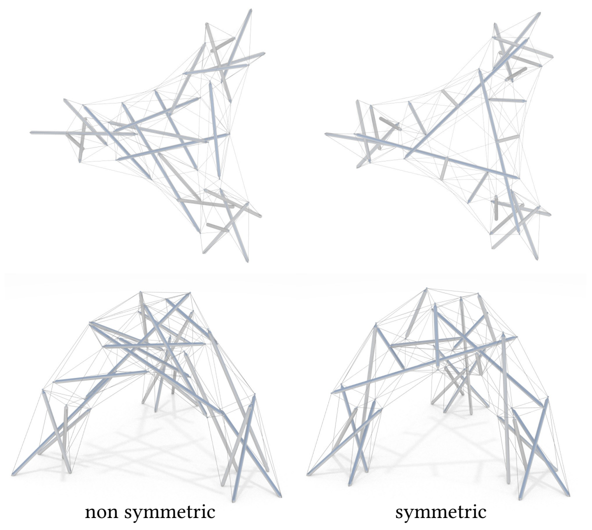


Fig. 7. Stable structures obtained with and without rotational symmetry constraint during the initialization (above: top view; below: side view).

be easily verified with specifically designed examples[1]; however, such examples are rare in practice, and we found that this does not impact the usability of our method.

We model this *displacement* as a set of per-node displacement vectors $D = \{\delta_1 \dots \delta_n\}$, describing the deformation $q_i = p_i + t \cdot \delta_i$, which is infinitesimal when $t \to 0$.

Factoring out isometries. Global isometries are trivially permitted transformations, so we need to first factor them out. For that, we limit the search to displacements that preserve the center of mass and the angular momentum of the structure. Any arbitrary distribution of masses can be assumed for this purpose: we just assume a constant mass to be concentrated at each node. In mathematical terms, we only look for displacements D fulfilling:

$$\begin{aligned} \textstyle\sum_i \delta_i &= (0,0,0)^\mathsf{T} \\ \textstyle\sum_i \delta_i \times (p_i - b) &= (0,0,0)^\mathsf{T}, \end{aligned} \tag{4}$$

where b is the averaged positions of all nodes in P. Similar formulations appear in, e.g., [Kilian et al. 2007]. Any arbitrary displacement is composed of a displacement fulfilling (4), a global translation, and a displacement field inducing a rotation around some axis passing through b. The only *isometric* displacement fulfilling (Equation 4) is the null displacement.

Searching for infinitesimal displacements. We are looking for non-zero displacements D that neither elongate cables, nor shorten struts, as per Equation 1. If we find one, then the structure is not stable. For this, the following well-known conditions (e.g.[Connelly and Whiteley 1996; Roth and Whiteley 1981; Whiteley 1984]) on D must hold:

$$\begin{aligned} (p_i - p_j)^\mathsf{T}(\delta_i - \delta_j) &\leqslant 0 \quad \forall (i,j) \in C \\ (p_i - p_j)^\mathsf{T}(\delta_i - \delta_j) &\geqslant 0 \quad \forall (i,j) \in T. \end{aligned} \tag{5}$$

Putting together equations (4) and (5), we get a system of linear inequalities, concisely represented as:

$$\mathbf{M}\,\mathbf{x} \geqslant 0, \tag{6}$$

where $\mathbf{x}$ is a column vector of $3n$ unknowns $\mathbf{x} = \{\delta_1, \delta_2 \ \dots\}$, and $\mathbf{M}$ is a sparse matrix with $|T| + |C|$ (for Eq. 5) +4 (for Eq. 4) rows.

The space of solutions of Equation 6 is either only the origin, or an unbounded hyperpyramid-shaped convex polytope centered at the origin. We seek a non-zero solution, using an adaptation of LP, as detailed in the following paragraph; if no such solution exists, then the tensegrity is found to be stable. As a side effect to detecting non-stabilty, we produce a *permitted* displacement D, which is a crucial component in the following form-finding step.

Solving for Equation 6. Linear Programming (LP) is a well established method for solving Equations of the type 6; it is known to be extremely efficient in practice [Gamrath et al. 2016]. Unfortunately, we cannot use LP in a straightforward manner, as we do not possess a linear objective function, which is a requirement. Clearly, just asking the distance of $\mathbf{x}$ from the origin to be maximized (in order to produce a non-zero solution, if it exists), cannot be expressed

[1] Consider the following structure: only four nodes A,B,C,D that are perfectly aligned on a line; cables connect AB, BC, and CD; struts connect AD and BC (for the sake of this example, we allow BC to be connected by both, against our constraints); the configuration is actually rigid, yet an isolated rotation of BC around its center corresponds to an admissible infinitesimal displacement, as B and C initial movement is orthogonal to any cable or strut.

as a linear, or even convex, objective function. Instead, we adopt a simple but effective randomized algorithm which works robustly in practice.

We produce a random n-sized objective vector $\mathbf{g}$ (for goal), and we solve a pair of LP problems, maximizing $(\mathbf{g}^\intercal\mathbf{x})$ and $(-\mathbf{g}^\intercal\mathbf{x})$ independently. If either problem returns an unbounded growth direction $\mathbf{d}$, this $\mathbf{d}$ is our non-zero solution for (6); otherwise, we know there is no such solution.

The rationale is as follows: it is obvious that, if the origin is the only solution of (6), that both problems should produce the origin as the optimum. Suppose that, conversely, some non-zero solution $\mathbf{s}$ does exist. If $(\mathbf{g}^\intercal\mathbf{s}) > 0$, then the first objective function is unbounded from above, under (6), and LP returns an unbounded growth direction. If $(\mathbf{g}^\intercal\mathbf{s}) < 0$, then the second objective function will be unbounded from above. The only other case is when $(\mathbf{g}^\intercal\mathbf{s}) = 0$ exactly, which, in any dimension, has a vanishing probability for a randomly chosen vector $\mathbf{g}$. This virtually never occurs in practice. Note that we do not assume either $\mathbf{g}$ or $-\mathbf{g}$ to be direct solutions of (6) (in most cases, neither is).

5.3 Form-Finding

Given an initial, and unstable structure (P_0, T, C), we wish to find a stable structure (P, T, C) that is close to the original solution. That is, $d(P, P_0) \leqslant \tau$ for some user defined *tolerance* τ.

Algorithm. The algorithm is iterative, producing a sequence of shapes P_i, starting from P_0. Each iteration i comprises the following steps:

(1) **Test**: is P_i stable (Sec. 5.2):
 1.a: *if so*: end process with success, return P_i.
 1.b: *else*: a morphing D_i is produced from the stability test (Sec. 5.2).
(2) **Deform**: $P_{i+1} = P_i + sD_i$, for some scalar $s > 0$ (see below).
(3) **Test**: is $|P_i - P_0|^2_\infty \leqslant \tau$
 3.a *if so*: $i \leftarrow i + 1$, goto 1.
 3.b *else*: end process with Failure.

Depending on the input, form-finding can either terminate with a success, when the sought stable configuration is found, or a failure, which is the only possible outcome when no stable configuration exists within the required tolerance τ. Although we cannot guarantee this algorithm to always succeed if there is a solution, it works well in practice (see Sec. 6).

We offer the following intuitive explanation for this result: usually, $P_{i+1} \dot{\subseteq} P_i$. While this is guaranteed only for a sufficiently small s (in rare cases, for no strictly positive value of s), it is approximately true (i.e. most of the Equations 1 are fulfilled) for a much larger set of values. The relation $Q \dot{\subseteq} P$ can be understood as "Q is tighter than P", because cables are not longer, and struts are not shorter in Q than in P, making Q more restrictive than P. The iterative process, then, tends to be a progressive *tightening* of the structure, which is carried on until the tolerance is exceeded (3.b), or no further tightening is possible (1.a). The latter condition is equivalent to stability.

Ideally, in phase 1.b, we prefer to use solutions for Equation (6) that are in the interior of the feasible space, rather than close on its boundary. Unfortunately, our LP-based solver (Sec. 5.2) will produce solutions that are exactly on the boundary. A simple countermeasure is to repeat step 1 for a small number (in our experiments we used 4) of randomly generated $\mathbf{g}_{1..4}$. For each, we solve the LP problem (Section 5.2), and average the solutions (because the feasible space is convex, this yields a valid solution). In this way, the produced solution D_i tends to be in the interior of the feasible space.

The two parameters of this algorithm are the tolerance τ and step size s. Using a smaller s requires more steps, but produces a smaller overall displacement with respect to P_0. For all our experiments, we used $\tau = 0.2l$ and $s = 0.002l$, where l is the length of the diagonal of the axis-aligned bounding box. The Form-Finding procedure requires an average of 10–20 iterations in our experiments.

Connectivity Revision. When the form-finding procedure returns a failure, it is a strong indication that the provided connectivity cannot be converted into a stable tensegrity without significant changes in its shape. Our strategy in this case is to revert to the connectivity-finding step, and bias it towards finding a better starting point, and subsequently redo the form-finding. We bias it by adding a set of auxiliary constraints $c = \{c_1, c_2, \ldots\}$, and alternating macro-iterations between connectivity finding and form-finding, as follows.

For a macro iteration m, and given a selection of struts T that failed to reach stability in the previous macro iteration $m-1$, we begin with a solution where all cables of the graph are equipped with dummy coincident struts (temporarily violating connectivity constraints). Let T_D be the set of dummy struts added for the configuration T. This produces a structure that is trivially stable, due to the excess of struts.

Then, we start removing the dummy struts one by one, and collecting them in an initially empty set T_R. At every removal, we do a form-finding step, until one fails to reach stability (which is bound to happen before all struts in T_D are removed, otherwise T would not have failed). When we obtain the first failing configuration, we know that at least one of the removed struts in T_R is crucial for the form-finding to reach stability. Thus, we add an additional constraint to the ILP solver in the connectivity finding step, that requires that at least one of T_R must be a strut:

$$c_m := \sum_{(i,j)\in T_R} v_{ij} \geqslant 1 \tag{7}$$

Macro iterations are repeated until form-finding returns a success. We exemplify this in Figure 8.

Final affine correction. The stability of tensegrity structures is invariant under affine transformations [Roth and Whiteley 1981; Zhang and Ohsaki 2015]. We exploit this to make the final result P more similar to P_0: when form-finding succeeds in finding P, we compute the best-fitting affine transformation $A_{3\times3}$ by solving the following quadratic least-squares problem:

$$\arg\min_A |P_0 - A(P)|^2_2 , \tag{8}$$

and substitute P with $A(P)$. Note that no translational component needs be included in $A_{3\times3}$, since our algorithm factors out rigid motions and preserves the center of mass. An example of the effect of the affine correction can be observed in Figure 9.

Discussion. Form-finding is the central step of our pipeline, and it dramatically simplifies the design of stable tensegrities. Equipped

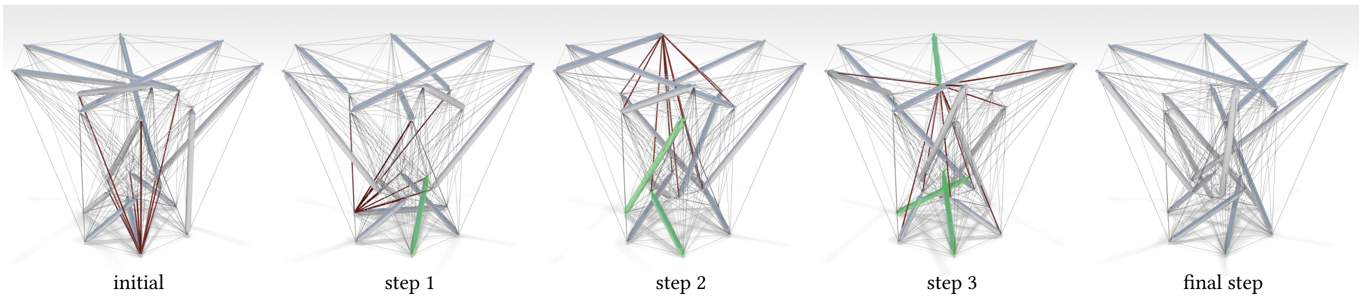


Fig. 8. Exploring the space of possible connectivities that leads to a stable configuration after 11 steps. For each step, the set c of additional hard constraints is shown in Red, while the green strut is the one inserted in the step $m + 1$ to accommodate the constraint c_m defined on a previous step.

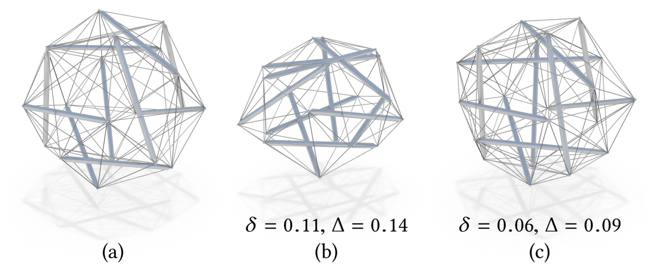

$\delta = 0.11, \Delta = 0.14$
$\delta = 0.06, \Delta = 0.09$
(a)
(b)
(c)

Fig. 9. The effect of affine correction: (a) The tensegrity before form-finding; (b) Stable tensegrity without affine correction step; (c) Stable tensegrity with affine correction step; refer to Fig. 4 for the meanings of δ, Δ.

with it, the other steps are exempt from providing exactly stable configurations. Empirically, we found that form-finding succeeds for a wide variety of initial structures of many shapes (e.g. Fig. 11 and many other thought this article), with little visual difference between the original configuration P_0 and the final result P. That in itself brings out an important insight: stable and unstable configurations can appear visually similar and can be difficult to discriminate by visual inspection alone (see for example Figures 1, 4, 9, or 13); this is one reason why tensegrity structures are especially difficult to design manually.

5.4 Geometric Constraints

This step complements the form-finding, by making the tensegrity structure adhere to auxiliary geometric constraints on a given stable configuration P, while maintaining stability. Our position-based framework can easily accommodate any constraint that can be expressed as a geometric condition on the position of the nodes. This is intended mainly to recover geometric conditions that are lost by the form-finding. We therefore assume that the sought conditions are approximately satisfied in the initial structure.

Algorithm. We employ an alternating algorithm: constraint projection, where we directly enforce the constraints on P, while minimizing the total squared amount of applied displacement. This is followed by a form-finding step (Section 5.3) to recover stability. The process is repeated until convergence, i.e. until the constraints-projected structure is stable prior to the form-finding step. While this process is not guaranteed to converge, we empirically found it to do so in less than 5 iterations in all our experiments (Figure 10). We can accommodate the following classes of constraints:

(1) **Minimal distance**, enforced between pairs of struts.
(2) **Coplanarity of a subset of nodes**, which can be useful for mounting a flat panel on top of them (Figure 10e).
(3) **Original face-shape preservation**: this is achieved by finding the best rigid alignment of the original face geometry to the current one (Figure 10f).
(4) **Forbidden zones**: the struts and cables are moved out of the forbidden zones prescribed by the designer in Section 5.1 (Figure 12a).

These constraints are imposed using a local-global approach, in the manner of [Bouaziz et al. 2012].

5.5 Structure Simplification

In this step, we remove all redundant cables from a stable structure, while keeping it stable.

Algorithm. Starting from a stable structure, we tentatively remove cables one after the other. After each removal, we test the structure for stability. If the structure is then unstable, the cable is labeled as indispensable and returned to the structure, and never tested again during the simplification. This means that each cable needs be tested for removal only once. Note that the removal of cables cannot violate any connectivity constraint (Section 4.1).

The removal order may significantly affect the final result, as the removal of one cable can make another cable indispensable. We adopt a simple strategy, based on valency: at each step, we test for removal the unlabelled cable with the largest number of surviving adjacent cables. Ties are resolved in favor of the longer cable. In our experiments, this leads to balanced structures with little clutter.

Discussion. Compare Figure 1 (d) and (e) for an example of a simplification process. This step allows us to use structures with an excess of cables in all previous steps; this eases the search for valid connectivites, and increases the robustness of the form-finding, since it increases the solution space. In this way, we are exempt from the otherwise difficult task of identifying the exact cables required to make the structure stable before the form-finding. On the other hand, the simplification step is time consuming. To ameliorate performances, we adopt the following two improvements.

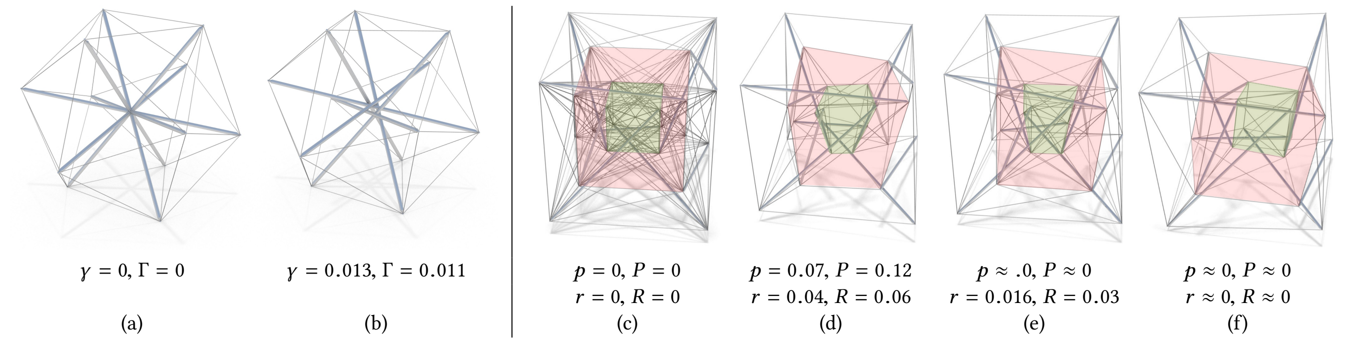


Fig. 10. Effects of geometric constraints on the form-finding procedure. A simple (and already stable) sphere model (a) is stabilized again (b) enforcing minimal distance between struts. An initial “hypercube” model (c) is made stable: without any constraint (d); enforcing face planarity (e); additionally enforcing face-shape preservation (f). The structures (d), (e) and (f) are also simplified; γ and Γ are the average and minimum distance between any strut and its closest one; p and P are the average and worst planarity value of constrained faces, computed as the distance from the best fitting plane; r and R are the average and maximum distance to the rigidly aligned original polygon. Distances are expressed as fractions of the bounding box of the structure.

Adaptively delayed stability check. To reduce the computational cost, we perform a stability test only once every k consecutive tentative removals. If stability is maintained, we finalize all of them; otherwise, we reverse them, and try again with a smaller k (until $k = 1$, then we label the tested cable as indispensable and move on, as before). Specifically, we start with $k = 1$, quadruple it every time a removal attempt is successful, and we divide it by 4 otherwise.

Fast single-cable test. When a single cable (i, j) is tested for removal, the stability test can be made more efficient; only one LP problem needs to be solved instead of two, using $(\delta_j - \delta_i)$ as the coefficients of the linear objective function. In other words, if the system, once cable (i, j) is removed, is still unable to pull nodes p_i and p_j further apart, then cable (i, j) is redundant.

6 EXPERIMENTAL RESULTS

We test our framework by designing a variety of stable tensegrities, starting from both polygonal surfaces and point clouds. We show a few models that we used to test the ability of the framework to generate coherent results for different variations of initial shapes in Figure 11, as a typical part of the artistic design process. Figure 17 (right) illustrates how our framework readily handles complex inputs both in terms of node numbers (> 100), and of initial geometry (not homeomorphic to a sphere, and with thin parts). Numerical details about the presented structure are found in Table 1.

6.1 Empirical validation

We empirically validate the generated structures in three ways as listed below.

Accordance with pre-stress based methods. We empirically verify that all the stable structures built with our framework are also “super-stable” according to the definition of [Zhang and Ohsaki 2007]. This requires solving for a valid set of per-element pre-stresses (which are not needed in our pipeline); we successfully did this for each of our results, using a variation the method proposed in [Zhang and Ohsaki 2015]; more details are provided in Section 6.4. Note that finding a pre-stress *a-posteriori*, over an already valid graph

Table 1. Tensegrity design statistics: model name, number of nodes, type of edge initialization (↷ is h-path connected, and ⋆ uses k-closest, see Section 5.1), resulting number of edges, presence of a collision mesh (see Section 4), enforcement of symmetry constraints (see Section 4.1), final number of struts and cables, illustrating figure(s), and total computation time.

model	*nodes* $\lvert P\rvert$	*edge init.*	*edges* $\lvert E\rvert$	*coll.*	*sym.*	*struts* $\lvert S\rvert$	*cables* $\lvert C\rvert$	*figure*	*time*
arc A	30	↷	266			15	78	1	30 s
arc B	84	↷	1150			42	245	11.a	124 s
arc C	56	↷	569			28	144	11.b	58 s
arc D	84	↷	1075			42	281	11.c	57 s
arc E	56	↷	534			28	147	11.d	68 s
	24	⋆	228			12	76	10.d	30 s
hyper-cube	24	⋆	228			12	103	10.e	42 s
	24	⋆	228			12	73	10.f	45 s
dolphin	52	↷				26	354	13.a(r)	2 s
torus	32	↷	200			16	105	13.b(r)	10 s
sphere-7	14	⋆	91			7	30	5(l)	21 s
triple-arc A	48	↷	523			24	168	7(l)	72 s
triple-arc B	48	↷	523		✓	24	146	5(r),7(r)	64 s
chair A	28	↷	256	✓		14	84	12 (r)	5 m
chair B	28	↷	256	✓		14	87	12 (l)	3 m
chair C	28	↷	256	✓		14	72	12 (c)	3 m
bridge	80	⋆	2432	✓		420	226	15 (r)	25 m
dogfight	50	⋆	672			5	69	16	10 m
twirl	124	↷	1980			62	611	17 (r)	12 m
knot	240	↷	2983			120	728	17 (l)	82 m

is much easier, as the position of the nodes and the verse of the forces (inward for cables, outward for struts) are already known. The computed forces can also be used in any scenario requiring a pre-stress over each element (e.g. physical realization). However there is no formal proof that infinitesimal rigidity is equivalent to super-stability as defined by Zhang and Ohsaki [2007].

Physical simulation. We test the stability of our structures by using a physically-based simulation. The simulation is based on Position-Based Dynamics [Müller et al. 2007], in which reaction

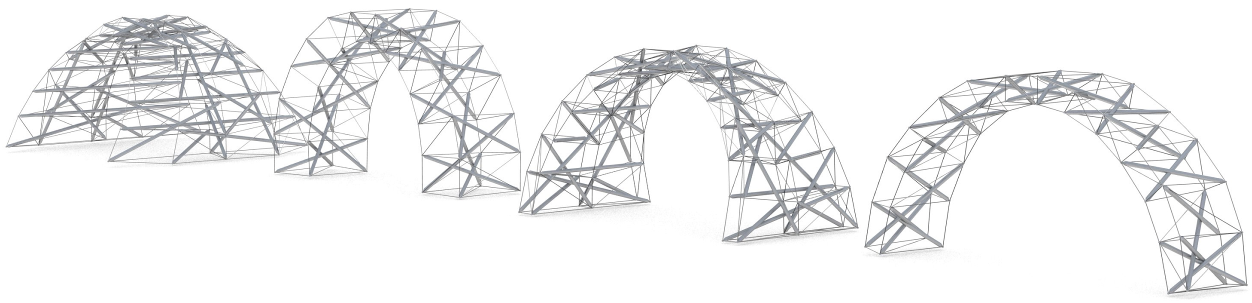


Fig. 11. An array of different arches designed with our pipeline, initiated with polygonal meshes of various shapes and dimensions. Our pipeline preserves well the input arch shapes, while enforcing stability.

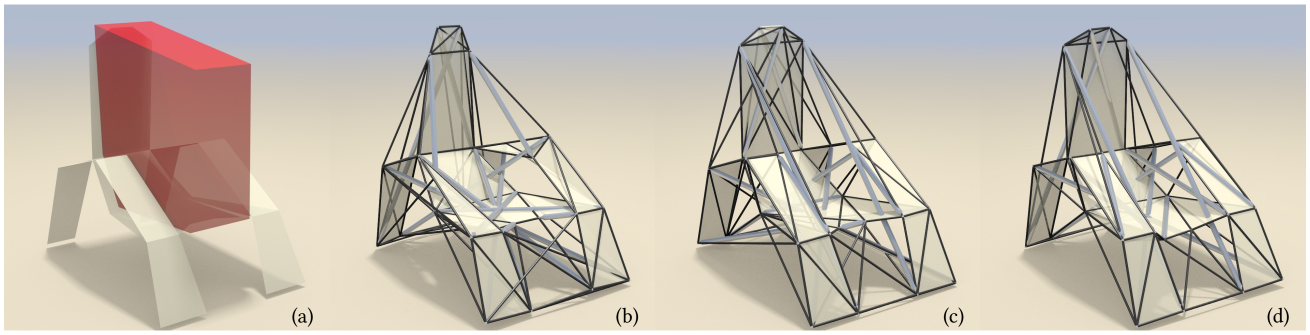


Fig. 12. Three variations on a "tensegrity chair", obtained from the same input mesh (on the left) and collision mesh (in red) to keep the seating area free. The variations are obtained for various choices of connectivity-finding parameters: normal orientation, random, and maximal total length (Section 5.1).

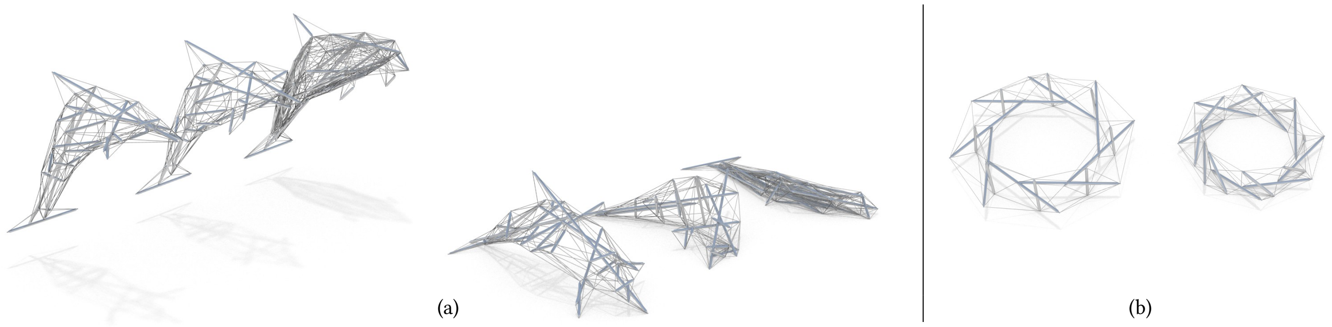


Fig. 13. Two frames from the simulation shown in the video: (a) The dolphin on the left is the result of our entire pipeline, the one on the center has not been optimized for stability (skipping the form-finding phase), and the last one has been initialized without the separating plane constraint (and consequently not optimized for stability). Even though the last model has many more cables than the others, it is the weaker one. (b) A comparison showing an optimized torus (left) versus a non-optimized one (right), which partly collapses.

forces from rigid objects are not explicitly modeled; instead, node positions are reprojected so to adhere to hard positional constraints (e.g. non intersection with the ground); the updates of positions imply velocity changes. In our simulation, the positional constraints for *internal* reaction forces abide exclusively to the principles of Tensegrity: we impose that the length of all cables/struts are bounded above/below by their rest (i.e. initial) length (Eq. 1). Every other deformation is allowed. Note that we allow the extension of struts, even if this is counterintuitive; this is done to factor out the resistance of the strut to extension, which we do not rely on (Section 2).

To test a given structure, we simulate two scenarios: we apply a random load to its nodes; we drop it from an height and let it hit the ground at random angles. We verified that, as expected, all structures produced by our framework behave rigidly, while structures that are not optimized for stability deform visibly, or even collapse (see attached video and Figure 13).

Physical Validation. We also manually assembled physical realizations of three of our designs (see Figures 1 and 5), and verified their ability to stay rigid in presence of external stress, confirming the validity of our framework. In particular, the two models shown in Figure 5 are made of stainless steel tubes (diameter 60 mm) and

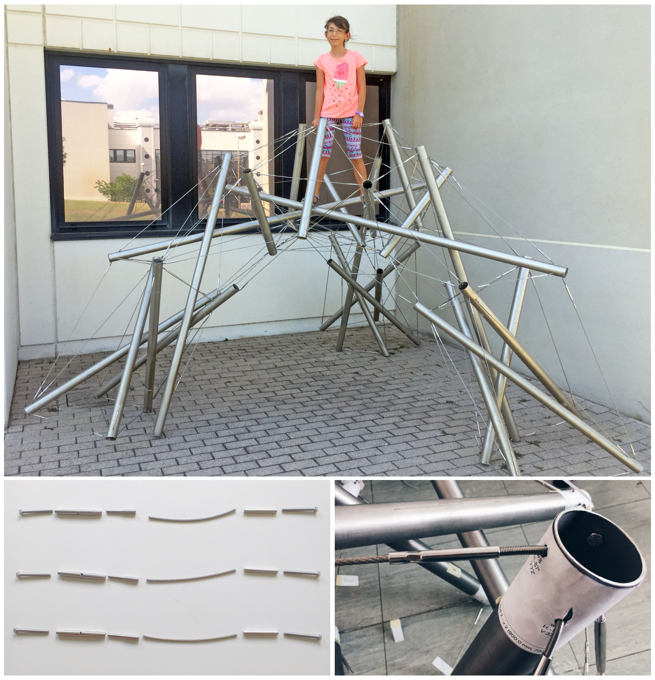


Fig. 14. Top: The rigidity of our structures allows a person to climb on them without introducing deformations. Bottom: The metallic components and labelling used for assembly.

steel cables (diameter 4 mm). As expected, these structures are not only rigid but also extremely sturdy (Figure 14, top). The bigger one (24 tubes, 145 cables) weights about 130 kg and is 2.3 m tall; its construction required approximately one week. Detail are as follows (Figure 14, bottom). We drilled one hole at a tube extremity for each incident cable; the hole position was computed by carefully taking into account the insertion angle and the tube width. Bent bolts were inserted in the holes, and cable ends were secured to them with threaded cable terminals. Pre-stress was added by using tensioners in each cable. Because of the difficulties of accurately reproducing forces, we did not use the values we computed using [Zhang and Ohsaki 2015]; instead, we applied a simple and intuitive procedure: we progressively tightened the tensioners manually, until the structure reached full rigidity.

6.2 Tweaking the Tensegrity Pipeline

Our main pipeline (Section 4) can also be modified to handle variations of the tensegrity design problem, as follows.

Removing tension from a truss structure. Given a truss, i.e., a structure made of welded struts which relies on both compression and tension forces from the struts for stability, we want to secure trusses to a small number of cables, so that the stability of the structure can be guaranteed when only the cables sustain the tension. That is, not considering the resistance of the struts to tension, as per our assumptions. To tackle this problem, it is sufficient to initialize a graph by adding cables between the endpoint of the struts (Section 5.1), and then perform a step of structure simplification (Section 5.5).

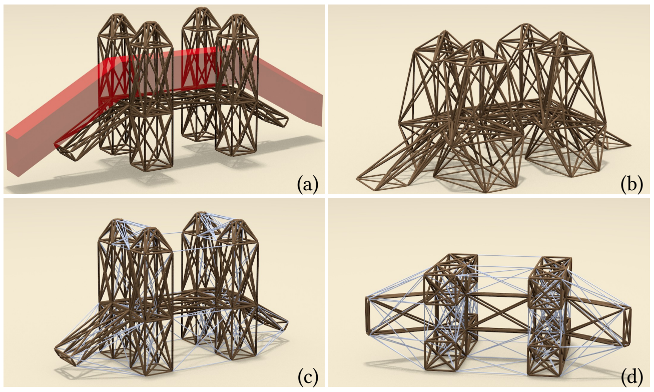


Fig. 15. (a) An input welded truss bridge, held stable by both the compression and tension forces of its 420 struts (b) Removing tension forces, the bridge is no longer stable. (c) A set of 226 cables, identified by our system, is added, obtaining a “compression-only truss”, where all tension forces are delegated to cables (no cable is allowed in the red volume visible in (a), which is provided as an input to prescribe a free passage through the bridge); (d) As a stress test, the resulting object is dropped on the ground in a simulation, but it sustains the impact without deforming (see attached video).

In this setup, the tensegrity connectivity constraints are ignored, as struts must be connected to each other. This is demonstrated in Figure 15, and in the video.

Designing a supporting tensegrity structure for an installation. Given a set of rigid, sturdy objects, each coming with a set of “attachment points” where cables can be secured, we want to design a supporting tensegrity structure. This structure, consisting of a few struts connected by cables, should be capable of keeping the installation in mid-air, in a prescribed 3D spatial arrangement. We want the objects to be held in their place exclusively by cables connecting pairs of attachment points on different models, and/or endpoints of the struts. To address this task, we construct a graph with a node for each attachment point of each object at its designated place, and we add five extra struts connecting random points of an enclosing bounding sphere of the scene. To recreate the sturdiness of the objects, we add both a “virtual” cable and a “virtual” strut between each pair of attachment points of the same model, thus making it fully rigid (in this setup, the normal connectivity constraints are disregarded). Then, we proceed as usual: we populate the scene with cables, apply form-finding to achieve stability, and finally use structure simplification to remove redundant cables. During this process, we treat the 3D models of the input objects as collision objects (to avoid cables or struts to traverse them), and we enforce the shapes and sizes of the original models as auxiliary geometric constraints, to ensure that they are not deformed by the form-finding process. A result is shown in Figure 16.

6.3 Timings

The processing times vary considerably between different models, but are well within a reasonable range for interactive processing with medium-sized structures (six to a few hundreds nodes). The most time-consuming step is *structure simplification*, as the number

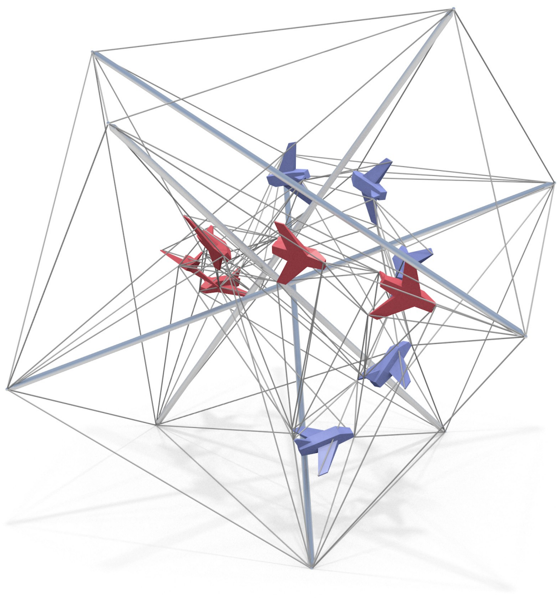


Fig. 16. A "tensegrity support structure" automatically designed by our system keeping ten rigid toy planes in a prescribed spatial configuration.

of stability tests is proportional to the number of cables. Nevertheless, with the batch removal improvement, this phase takes only a few minutes for most models (Table 1). For the most complex structure, the knot of Fig. 17, the overall process took 82 minutes; tests were performed on a 2.3 GHz Intel Core i7 with 8Gb and we used SCIP [Gamrath et al. 2016] for the ILP problem.

6.4 Direct comparisons with alternatives

With Adaptive Force-Density Method. As mentioned in Section 3, the task of making existing structures stable has a rather established solution based on the adaptive force-density method [Zhang and Ohsaki 2006]. This process of form-finding deforms (if needed) the input structure so that they become *super-stable* (according to the definition in [Zhang and Ohsaki 2006]). This requires an initialization of "prestress forces" t_{ij} to each element, which is hard to do, yet essential to the task. The direction of these forces, relative to the respective element, is determined from a preprocessed labelling of the graph edges as cables or struts. This preprocess is not directly addressed in [Zhang and Ohsaki 2006], and so, for comparison purposes, we used our connectivity-finding procedure (Section 5.1). We compute the magnitudes of the forces by solving the force system for equilibrium, under the condition that $t_{ij} \leqslant -\alpha$ for each strut, and $t_{ij} \geqslant \alpha$ for each cable, for a given $\alpha > 0$. This avoids the convergence to the degenerate solution, where all prestresses are zero. This form-finding process, when applied to non-stable structures, deforms the input shape in an unpredictable way. See the comparison in Figure 4.

With [Tachi 2013]. We provide a direct comparison with the design system proposed by [Tachi 2013] in Figure 18. For a fair comparison, we generated both results automatically, without exploiting any interactive feature. Our framework is able to adhere to the input shape and graph and without increasing its complexity (resulting in

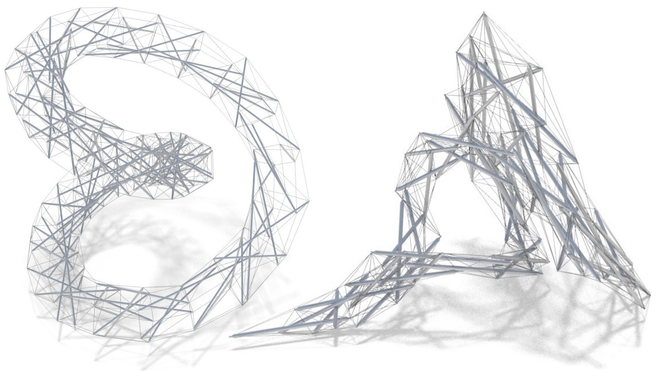


Fig. 17. Two tensegrities constructed from initial complex geometric models.

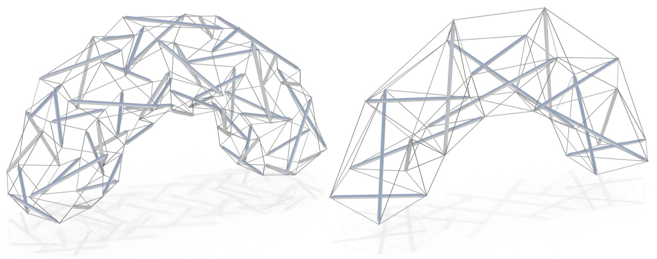


Fig. 18. The same input model is converted into a tensegrity using [Tachi 2013] (left) and with our algorithm (right).

fewer than one third of the struts being used: 15 vs. 55). Moreover, [Tachi 2013] is not guaranteed to produce stable configurations; indeed, the shown example by [Tachi 2013] is not stable, and cannot be made stable using form-finding (we tried using both our algorithm and [Zhang and Ohsaki 2006]).

7 CONCLUSIONS

We proposed a novel and flexible framework for the computational design of tensegrity structures, which includes a fully automatic pipeline ranging from connectivity optimization, through form finding, to enforcing auxiliary geometric constraints. This allows the exploration of the highly-constrained space of tensegrity structures. In the following we address a few limitations and sketch directions for future research.

Stronger rigidity conditions. In our work we optimize for *local* (*infinitesimal*) rigidity only. This guarantees stability, i.e. that the object cannot deform continuously. This is guaranteed as the struts and the cables are perfectly incompressible/inextensible, and exert internal forces that always negate external forces.

Nevertheless, the continuous deformation assumption is only an approximation in real-world settings: small elastic deformations occur in response to sudden external forces, and the physical object briefly goes through a non-admissible state, potentially ending up in an different admissible state; we have no way to predict if the resulting state would also be locally rigid. Because of this, an object that is deemed stable in our system may not be robustly rigid in practice. Our physical experiments and empirical simulations, however, strongly indicate that this infinitesimal approximation works well in practice.

A systematic way to handle the problem, in the context of our work, is to adopt a stronger condition for rigidity, experimenting with deformations that are small but are no longer assumed to be infinitesimal. While still a weaker condition than global rigidity, this problem appears, mathematically and algorithmically, more challenging. We plan to address it in the near future.

Physical construction. In our experience (see Section 6.1), the complexity of designed structure makes their assembly a daunting task. With many pieces, we encountered a number of practical problems, pertaining to both maintaining a partial structure in correct shape, and understanding the correct order of progression. Previous works about tensegrity assembly are limited to the actual construction of very simple designs, or exploit regular and modular structures, which allow independent construction of their parts. In the future, we plan to computationally assist the assembly process.

ACKNOWLEDGMENTS

We wish to thank the anonymous reviewers for the many insights and suggestions. We thank the many colleagues and friends who helped us during the fabrication process: Luigi Malomo, Marco Callieri, Gaia Pavoni, Federico Ponchio, Manuele Sabbadin, Gianpaolo Palma, Matteo Dellepiane, Massimiliano Corsini, Marco Potenziani, Francesco Banterle, Lorenzo Della Rovere, Pietro Cignoni (for their hard work in assembling the structure), Mauro Voliani, Federico Pardini and Fabio Micheletti (for cutting and drilling), Roberto Scopigno, Claudio Montani and Domenico Laforenza (for allowing to place the structure in the main garden), and finally Margherita Cignoni (for fearlessly testing the structure).

REFERENCES

Sofien Bouaziz, Mario Deuss, Yuliy Schwartzburg, Thibaut Weise, and Mark Pauly. 2012. Shape-Up: Shaping Discrete Geometry with Projections. *Comput. Graph. Forum* 31, 5 (Aug. 2012), 1657–1667. DOI: http://dx.doi.org/10.1111/j.1467-8659.2012.03171.x

Paolo Cignoni, Nico Pietroni, Luigi Malomo, and Roberto Scopigno. 2014. Field-aligned Mesh Joinery. *ACM Trans. Graph.* 33, 1, Article 11 (Feb. 2014), 12 pages. DOI: http://dx.doi.org/10.1145/2537852

Robert Connelly. 2013. *Tensegrities and Global Rigidity.* Springer New York, New York, NY, 267–278. DOI: http://dx.doi.org/10.1007/978-0-387-92714-5_21

Robert Connelly and Walter Whiteley. 1996. Second-order rigidity and prestress stability for tensegrity frameworks. *SIAM J. on Discrete Mathematics* 9, 3 (1996), 453–491.

Fernando de Goes, Pierre Alliez, Houman Owhadi, and Mathieu Desbrun. 2013. On the Equilibrium of Simplicial Masonry Structures. *ACM Trans. Graph.* 32, 4, Article 93 (July 2013), 10 pages. DOI: http://dx.doi.org/10.1145/2461912.2461932

Shintaro Ehara and Yoshihiro Kanno. 2010. Topology design of tensegrity structures via mixed integer programming. *Int. J. of Solids and Structures* 47, 5 (2010), 571–579.

G Gomez Estrada, H-J Bungartz, and Camilla Mohrdieck. 2006. Numerical form-finding of tensegrity structures. *Int. J. of Solids and Structures* 43, 22 (2006), 6855–6868.

Gerald Gamrath, Tobias Fischer, and Tristan Gally et al. 2016. *The SCIP Optimization Suite 3.2.* Technical Report 15-60. Zuse Institute, Berlin.

Buntara Sthenly Gan, Jingyao Zhang, Dinh-Kien Nguyen, and Eiji Nouchi. 2015. Node-based genetic form-finding of irregular tensegrity structures. *Computers & Structures* 159 (2015), 61–73.

Damien Gauge, Stelian Coros, Sandro Mani, and Bernhard Thomaszewski. 2014. Interactive Design of Modular Tensegrity Characters. In *Proc. of the ACM Symp. on Computer Animation (SCA '14).* ACM, Eurographics Association, 8. http://dl.acm.org/citation.cfm?id=2849517.2849539

Steven J Gortler, Alexander D Healy, and Dylan P Thurston. 2010. Characterizing generic global rigidity. *American J. of Mathematics* 132, 4 (2010), 897–939.

Ariel Hanaor. 2012. Debunking Tensegrity - A Personal Perspective. *Int. J. of Space Structures* 27, 2-3 (2012), 179–183. DOI: http://dx.doi.org/10.1260/0266-3511.27.2-3.179 arXiv:http://dx.doi.org/10.1260/0266-3511.27.2-3.179

Kristian Hildebrand, Bernd Bickel, and Marc Alexa. 2012. Crdbrd: Shape Fabrication by Sliding Planar Slices. *Comput. Graph. Forum* 31, 2pt3 (May 2012), 583–592. DOI: http://dx.doi.org/10.1111/j.1467-8659.2012.03037.x

Sergi Hernandez Juan and Josep M Mirats Tur. 2008. Tensegrity frameworks: static analysis review. *Mechanism and Machine Theory* 43, 7 (2008), 859–881.

Martin Kilian, Niloy J. Mitra, and Helmut Pottmann. 2007. Geometric Modeling in Shape Space. *ACM Trans. Graph.* 26, 3, Article 64 (July 2007). DOI: http://dx.doi.org/10.1145/1276377.1276457

Kurilpa Bridge. 2009. (2009). http://www.arup.com/projects/kurilpa_bridge.

Yang Liu, Hao Pan, John Snyder, Wenping Wang, and Baining Guo. 2013. Computing Self-supporting Surfaces by Regular Triangulation. *ACM Trans. Graph.* 32, 4, Article 92 (July 2013), 10 pages. DOI: http://dx.doi.org/10.1145/2461912.2461927

Yang Liu, Weiwei Xu, Jun Wang, Lifeng Zhu, Baining Guo, Falai Chen, and Guoping Wang. 2011. General Planar Quadrilateral Mesh Design Using Conjugate Direction Field. *ACM Trans. Graph.* 30, 6 (Dec. 2011), 10. DOI: http://dx.doi.org/10.1145/2070781.2024174

Daniel Lobo and Francisco J Vico. 2010. Evolutionary development of tensegrity structures. *Biosystems* 101, 3 (2010), 167–176.

Eder Miguel, Mathias Lepoutre, and Bernd Bickel. 2016. Computational Design of Stable Planar-rod Structures. *ACM Trans. Graph.* 35, 4, Article 86 (July 2016), 11 pages. DOI: http://dx.doi.org/10.1145/2897824.2925978

Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. 2007. Position based dynamics. *J. of Visual Communication and Image Representation* 18, 2 (2007), 109–118.

Needle Tower. 1968. (1968). http://www.kennethsnelson.net/.

Daniele Panozzo, Philippe Block, and Olga Sorkine-Hornung. 2013. Designing Unreinforced Masonry Models. *ACM Transactions on Graphics (proc. of ACM SIGGRAPH)* 32, 4 (2013), 91:1–91:12.

Chandana Paul, Hod Lipson, and Francisco J Valero Cuevas. 2005. Evolutionary form-finding of tensegrity structures. In *Proc. of the 7th annual conf. on Genetic and evolutionary computation.* ACM, 3–10.

Nico Pietroni, Davide Tonelli, Enrico Puppo, Maurizio Froli, Roberto Scopigno, and Paolo Cignoni. 2015. Statics Aware Grid Shells. *Computer Graphics Forum* 34, 2 (2015), 627–641.

András Recski. 2008. *Combinatorial Conditions for the Rigidity of Tensegrity Frameworks.* Springer Berlin Heidelberg, Berlin, Heidelberg, 163–177. DOI: http://dx.doi.org/10.1007/978-3-540-77200-2_8

John Rieffel, Francisco Valero-Cuevas, and Hod Lipson. 2009. Automated discovery and optimization of large irregular tensegrity structures. *Computers & Structures* 87, 5 (2009), 368–379.

Ben Roth and Walter Whiteley. 1981. Tensegrity frameworks. *Trans. of the American Mathematical Society* 265, 2 (1981), 419–446.

Yuliy Schwartzburg and Mark Pauly. 2013. Fabrication-aware Design with Intersecting Planar Pieces. *Computer Graphics Forum (proc. of Eurographics 2013)* 32, 2 (2013), 317–326.

Mélina Skouras, Stelian Coros, Eitan Grinspun, and Bernhard Thomaszewski. 2015. Interactive Surface Design with Interlocking Elements. *ACM Trans. Graph.* 34, 6, Article 224 (Oct. 2015), 7 pages. DOI: http://dx.doi.org/10.1145/2816795.2818128

Tomohiro Tachi. 2013. Interactive Freeform Design of Tensegrity. In *Advances in Architectural Geometry 2012.* Springer, 259–268.

Chengcheng Tang, Xiang Sun, Alexandra Gomes, Johannes Wallner, and Helmut Pottmann. 2014. Form-finding with Polyhedral Meshes Made Simple. *ACM Trans. Graph.* 33, 4, Article 70 (July 2014), 9 pages. DOI: http://dx.doi.org/10.1145/2601097.2601213

AG Tibert and Sergio Pellegrino. 2011. Review of form-finding methods for tensegrity structures. *Int. J. of Space Structures* 26, 3 (2011), 241–255.

Etienne Vouga, Mathias Höbinger, Johannes Wallner, and Helmut Pottmann. 2012. Design of Self-supporting Surfaces. *ACM Trans. Graph.* 31, 4, Article 87 (July 2012), 11 pages. DOI: http://dx.doi.org/10.1145/2185520.2185583

Walter Whiteley. 1984. Infinitesimally rigid polyhedra. I. Statics of frameworks. *Trans. of the American Mathematical Society* 285, 2 (1984), 431–465.

Walter Whiteley. 1989. Rigidity and Polarity. 2. Weaving lines and Tensegrity Frameworks. *Geometriae Dedicata* 30, 3 (1989), 255–279.

Emily Whiting, John Ochsendorf, and Frédo Durand. 2009. Procedural Modeling of Structurally-sound Masonry Buildings. *ACM Trans. Graph.* 28, 5, Article 112 (Dec. 2009), 9 pages. DOI: http://dx.doi.org/10.1145/1618452.1618458

Emily Whiting, Hijung Shin, Robert Wang, John Ochsendorf, and Frédo Durand. 2012. Structural Optimization of 3D Masonry Buildings. *ACM Trans. Graph.* 31, 6 (2012), 159:1–159:11.

JY Zhang and M Ohsaki. 2006. Adaptive force density method for form-finding problem of tensegrity structures. *Int. J. of Solids and Structures* 43, 18 (2006), 5658–5673.

JY Zhang and M Ohsaki. 2007. Stability conditions for tensegrity structures. *Int. J. of Solids and Structures* 44, 11 (2007), 3875–3886.

Jing Yao Zhang and Makoto Ohsaki. 2015. *Tensegrity Structures.* Vol. 6. Springer Japan.