Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2016)
Ladislav Kavan and Chris Wojtan (Editors)

Deployable 3D Linkages with Collision Avoidance

Changxi Zheng[1] Timothy Sun[1] Xiang Chen[2,1]

[1]Columbia University [2]Zhejiang University

Abstract

We present a pipeline that allows ordinary users to create deployable scissor linkages in arbitrary 3D shapes, whose mechanisms are inspired by Hoberman's Sphere. From an arbitrary 3D model and a few user inputs, our method can generate a fabricable scissor linkage resembling that shape that aims to save as much space as possible in its most contracted state. Self-collisions are the primary obstacle in this goal, and these are not addressed in prior work. One key component of our algorithm is a succinct parameterization of these types of linkages. The fast continuous collision detection that arises from this parameterization serves as the foundation for the discontinuous optimization procedure that automatically improves joint placement for avoiding collisions. While linkages are usually composed of straight bars, we consider curved bars as a means of improving the contractibility. To that end, we describe a continuous optimization algorithm for locally deforming the bars.

1 Introduction

Deployable linkages fold and unfold, drastically changing their shapes without introducing any mechanical strain. The design of these linkages has intrigued inventors and engineers for centuries [Ben03], with motivating applications in transportation, architecture, consumer products, toy design, aerospace engineering, and biology. Among them, the most commonly used design is the *uniform scissor linkage*, which collapses uniformly, and is built up from pairs of connected coplanar bars, known as *scissor units*. Examples range from recreational toys such as the Hoberman Sphere [Hob91] (Fig. 2) to scientific endeavors such as the Precision Expandable Radar Calibration Sphere [BST*08], an uniformly expandable satellite launched into low Earth orbit.

The popularity of these linkages can be explained by their many mechanical benefits. A scissor linkage exhibits a single degree of mobility; in theory, only one actuator is needed to drive the motion. The mobility is present even though the various geometric constraints imposed by the linkage, such as the bars' lengths staying constant, outnumber its degrees of freedom. In short, these mechanisms are often *overconstrained*. The redundancy of these constraints offers structural failure tolerance: the linkage remains functional even if some of its components are removed.

The design of overconstrained contractible linkages is difficult because in general, such mechanisms will be rigid (i.e., no nontrivial motions). To obtain a single degree of mobility, we need to carefully construct the linkage's geometry so that the constraints have singularities (see more background in §3). In other words, the design needs to embrace singularities rather than avoid them, distinguishing our problem from recent work in linkage design in computer graphics (e.g., [CTN*13, TCG*14, BCT15]).

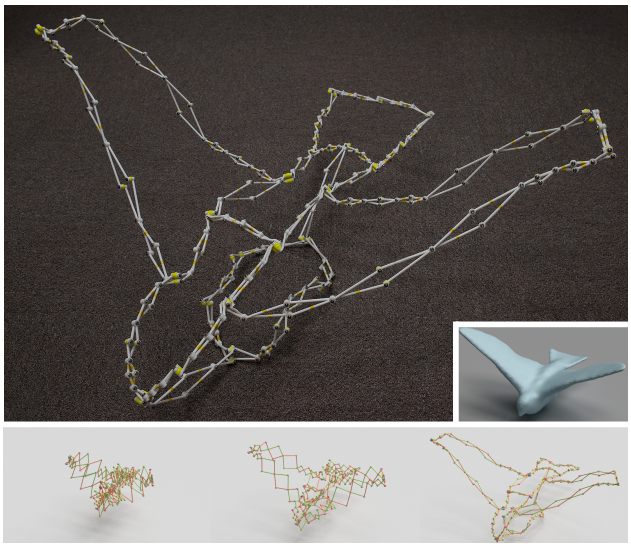


Figure 1: ***Contractible Bird.*** *A computationally generated and fabricated deployable linkage structure (top) resembles a bird shape (inset). It has a single degree of mobility, allowing the bird to uniformly expand and contract (bottom).*

The need for singular mechanical constraints significantly limits current designs of scissor linkages, especially those resembling specific 3D shapes. So far, existing 3D linkages typically have simple and symmetric shapes, such as spheres. Few have explored complex geometries.

In addition, as these linkages contract, bars are folded into a smaller bounding volume. It is unavoidable that they would collide with each other, preventing the linkage from contracting further. The

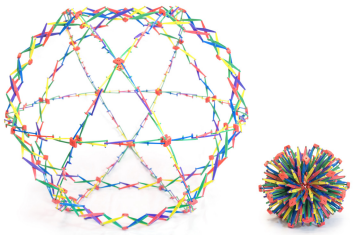


Figure 2: *Hoberman's sphere in expanded and contracted states.*

original intent of these deployable linkages is to save space, but with such collisions, this goal is obstructed. To the best of our knowledge, the goal of maximizing the contractibility remains largely unexplored.

We propose a method that enables non-experts to create 3D deployable linkages (Fig. 1). The linkage resembles a user-provided 3D shape at its most expanded state and is able to uniformly fold as much as possible. The output of our method is the linkage's 3D geometry that can be directly fabricated for physical construction.

Method in brief. Inspired by Hoberman's original design of foldable scissor linkages [Hob90], we characterize geometric properties that a scissor linkage needs to satisfy to form a uniformly deployable structure. Using these properties, we augment Hoberman's method to construct 3D scissor linkages that resemble user-provided 3D shapes. Our method also involves the user in the design loop, allowing interactive editing of the linkage's topology while automatically computing valid geometries.

We consider the problem not simply as one of constructing linkages, but one of constructing *compact* linkages. Exploiting the scissor linkage's geometric properties, we develop a fast continuous collision detection algorithm to detect when a collision occurs during contraction. We rely on this subroutine as a cornerstone for improving the compactness of the linkage, a challenging constrained and discontinuous optimization problem. Our strategy for increasing the linkage's compactness proceeds in two steps: (i) we optimize the positions and lengths of scissor units to avoid collisions using the continuous collision detection in a stochastic optimization algorithm, and (ii) accounting for the fabrication constraints of the linkage, we exploit an elastic rod model to locally deform the linkage's bars, further avoiding collisions without deviating too far from its original shape.

Our method is able to generate complex 3D linkages that are hard to design manually. Regardless of the geometric and structural complexity, the output linkage designs can be directly 3D printed and assembled, as we demonstrate virtually and physically with a variety of examples.

2 Related Work

Linkage design. The origins of linkage design date back to at latest the 18th century, with the four-bar linkages like Watt's linkage that approximated straight-line trajectories. Later work like Burmester's centerpoint theory (see §3.2.2 of [McC06]) enabled the design of complicated trajectories of four-bar linkages, and Ceylan et al. [CLM*13] relied on this theory to generate linkages of animated characters from motion capture sequences.

Beyond four-bar linkages, Kempe's universality theorem [Kin95], shows that any algebraic curve in 2D could be traced out by a point in a linkage, though the resulting linkage could have an intractable number of parts to physically realize. A series of papers [CTN*13, TCG*14] demonstrated how to construct linkages that approximated a desired input trajectory, and Bacher et al. [BCT15] showed how to automatically modify linkages to satisfy further user-specified constraints.

While the prior art deals with the difficult task of designing linkages that follow specified trajectories, these linkages generically have multiple degrees of mobility, and their methods aim to avoid singularities (e.g., [TCG*14, BCT15]). Though we concern ourselves with only simple uniform expansion and contraction, we construct linkages that are highly overconstrained, but are singular enough to be able to flex.

One recent work that deals with overconstrained linkages is that of Zhang et al. [ZWC*15], whose method generates 2D linkages out of scissor units that deform from one polygon to another. While their linkages can deform into a different shape, not just uniform contraction, their method is in 2D and is not directly applicable in 3D. In addition, their method does not aim to avoid collisions.

Transformable objects. In graphics, there has been recent work in the design of actuated characters beyond just linkages, and typically these objects are physically realizable via 3D printing or traditional manufacturing methods. Bächer et al. [BBJP12] and Calì et al. [CCA*12] used ball and socket joints to animate 3D printed characters. Some recent work on transformable puzzles [ZSMS14,SZ15] allows the user to interactively design puzzles that can change shapes by manipulating joints connecting different parts of the object. Our approach shares a similar motivation as the "boxelization" of Zhou et al. [ZSMS14], in that one application of their method is compressing objects into smaller bounding boxes via folding. Another work with similar goals in mind is that of Li et al. [LHAZ15], who modified furniture with extra joints so that they take up less space when folded.

Architectural design. Deployable structures are well-studied in architecture due to their widespread application in structure where retractability is desirable, like fences, awnings, and rooftops (e.g. [Piñ65]). One common approach is to start with a small foldable element and repeat the element over the whole surface (see [YP97]). One of the most important examples of such building blocks is Hoberman's angulated unit, which has been used to make his namesake spheres and retractable dome structures [Hob91]. You and Pellegrino [YP97] generalized the angulated unit in 2D to handle more complicated geometries (so-called "multiangulated" and "generalized angulated" units) while preserving the uniform expansion and contraction properties of Hoberman's original unit.

Recently, Roovers et al. [RMDT13] described a method for converting continuous surfaces to contractible linkages. Their main idea was to place scissor units along principal curvature lines, which are well-behaved for simple continuous surfaces (e.g., quadrics). It is possible to extend their method to discrete surfaces using conical meshes [LPW*06], whose elements form discretized principal curvature lines. Conical meshes find other applications in architecture, in particular the design of glass structures. In the present paper, we describe a more general way of constructing scissor linkages that is not limited by the surface's curvature.

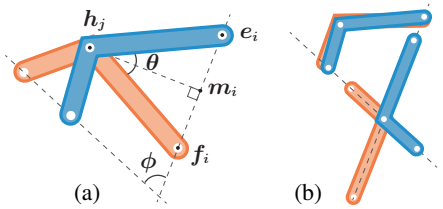
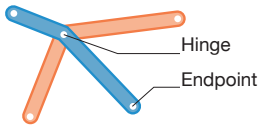


Figure 3: ***Angulated unit*** *used in Hoberman's sphere. The angle* ϕ *formed by the dashed lines is always the same. The right side shows an angulated unit at its theoretically most expanded (top) and contracted (bottom) states.*

Our work is based around Hoberman's angulated unit, but unlike existing literature, we focus on designing 3D linkages that avoid collisions as much as possible, maximizing its contractility. In 2D, the problem is sidestepped by offsetting pieces in a direction perpendicular to the plane, but in 3D, collisions are unavoidable. This unexplored facet of detecting and resolving collisions heavily impacts the physical realizability of these linkages, as some theoretical motions would be otherwise blocked by colliding parts.

3 Background

The goal of this work is to computationally generate uniformly deployable linkages given a 3D model. Its counterpart in 2D has been well studied and provides us useful insights, so we first review the established method for 2D linkage construction and introduce some terminology used throughout this paper.



Mechanical singularities. A 2D scissor linkage is made of a set of "X"-shaped scissor units; each consists of two bars that are coplanar and connected at their kink points by a hinge joint. In general, the geometry of a scissor unit is determined by two pairs of endpoints and the hinge point in 2D, possessing 10 degrees of freedom. Meanwhile, it has six constraints, including four that preserve the lengths between the endpoints and hinge point and two that preserve the kink angles at the hinges. Given a single loop of n scissor units (Figure 4), one can show that it has $6n$ constraints but $6n$-3 geometric degrees of freedom, after the 3 degrees of freedom of 2D rigid motion are eliminated. Thus, even with a single loop of units, the geometry of the linkage is already over-constrained—given arbitrary $6n$ geometric constraints, the linkage might not be physically realizable. With more loops, the linkage has even more constraints than its degrees of freedom.

A scissor linkage can fold only when its geometry is carefully assigned such that the constraints are redundant. However, computing such a geometry is ill-conditioned: to obtain a linkage with a degree of mobility, one necessary (but not sufficient) condition is that the Jacobian matrix of the constraints with respect to the linkage's endpoint and hinge positions must be rank-1 deficient [CDXF14], and finding such a matrix is a difficult task excessively susceptible to numerical imprecisions.

Angulated scissor unit. The aforementioned fundamental difficulty has motivated methods for procedurally constructing special types of linkages. Hoberman [Hob90] devised a special kind of scissor unit, known as his *angulated unit*, where two identical bars are connected by a hinge joint (Fig. 3(a)). One can prove (e.g., see [YP97]) that as the angle formed at the hinge changes, the lines passing through each pair of endpoints always form the same angle. For the remainder of this paper, when we refer to a "scissor unit" we mean Hoberman's angulated unit, unless otherwise specified.

2D linkages resembling polygons. A 2D linkage typically consists of a loop of scissor units. Using the angulated scissor unit, Hoberman [Hob91] developed a procedural algorithm that converts a polygon into a contractible linkage in 2D. Fig. 4 illustrates this algorithm, which has also been extended to simple 3D linkages by essentially joining together multiple copies of a 2D linkage. In the process of contraction, the polygon (dotted lines in Fig. 4) formed by connecting the hinge points is a uniformly scaled version of the user-provided polygon. Throughout this paper, we simply use the term "linkage" to refer to this type of uniformly deployable linkage.

4 Deployable 3D Linkages

In this section, we extend Hoberman's 2D linkage construction algorithm to 3D, creating deployable linkages that resemble specific 3D shapes. The input to this algorithm is a surface mesh, and the output is a set of connected scissor units and the positions of their endpoints and hinges.

At this step, the user is involved through the manipulation of the "linkage graph" (§4.2): the 3D linkage can be edited interactively by changing the edges and node positions of the graph (§4.3). The output of this step will be used in the next section to further optimize the linkage's geometry by taking into account possible collisions.

4.1 Geometric Parameterization

We first analyze Hoberman's 2D linkage and establish a geometric parameterization satisfied by all uniformly deployable scissor linkages. This parameterization provides insights for constructing 3D linkages and further serves as a starting point for us to derive a fast collision detection algorithm in §5. To our knowledge, this particular parameterization is new.

Recall that in Hoberman's angulated unit, the angle formed by the lines passing through the pairs of endpoints is identical regardless of the *deployment angle* θ (Fig. 3-a), the angle formed by a pair of endpoints and the hinge. Thus, we can assume without loss of generality that for all deployment angles, the orientation of the scissor unit (i.e. the orientation of the two lines passing through the endpoints) stays the same. When the deployment angle θ vanishes, the scissor unit is in its most expanded state where its two bars coincide. At the other extreme (i.e., $\theta = \pi$), pairs of endpoints are collinear with the hinge, forming an "X" shape (Fig. 3-b).

Let $\boldsymbol{e}_i$ and $\boldsymbol{f}_i$ be a pair of endpoints of a scissor unit. Since $\boldsymbol{e}_i$, $\boldsymbol{f}_i$ and the hinge $\boldsymbol{h}_j$ form an isosceles triangle, the line connecting $\boldsymbol{h}_j$ and $\boldsymbol{m}_i = (\boldsymbol{e}_i + \boldsymbol{f}_i)/2$ and the line connecting $\boldsymbol{e}_i$ and $\boldsymbol{f}_i$ are always perpendicular (Fig. 3-a). The length of the segment $\boldsymbol{m}_i\boldsymbol{h}_j$ can be expressed as $\|\boldsymbol{m}_i - \boldsymbol{h}_j\|_2 = L_i \cos(\frac{\theta}{2})$, where L_i is the length between $\boldsymbol{e}_i$ and $\boldsymbol{h}_j$ on the bar, a constant value throughout the linkage's contraction. In other words, the length of $\boldsymbol{m}_i\boldsymbol{h}_j$ changes linearly with respect to $\cos(\frac{\theta}{2})$. In 2D, $\boldsymbol{m}_i\boldsymbol{h}_j$ is a part of the shape polygon (recall the dotted line in Fig. 4), which contracts uniformly. Thus, in 3D, we expect that for any connected collection of scissor units, the distances between the midpoints and hinges also contract linearly

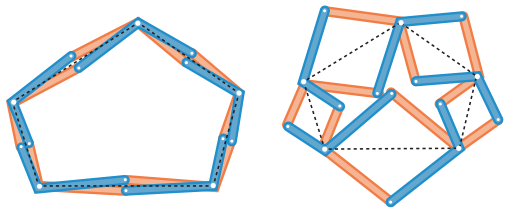
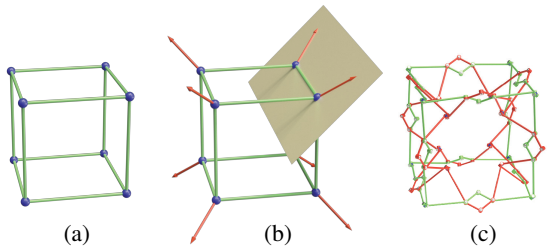


Figure 4: *Hoberman's 2D construction. For each vertex of the polygon (dotted line), there is a scissor unit with the hinge at the vertex's position and bars made from half of the sides incident with the vertex. Note that the polygons formed from the dotted lines in the two configurations are scaled versions of each other.*



(a) (b) (c)

Figure 5: ***Linkage Graph.*** *Starting from a 3D linkage graph* ***(a)****, we first compute normals for each graph node such that every pair of connected nodes has coplanar normals* ***(b)****. The graph is then converted into a 3D deployable linkage* ***(c)****.*

with $\cos(\frac{\theta}{2})$ until all those points collapse at a single point. Without loss of generality, we may assume that this point is the origin, and hence we can write the position of each midpoint and hinge, with respect to the deployment angle θ, simply as $\boldsymbol{m}_i(\theta) = \boldsymbol{m}_i(0)\cos(\frac{\theta}{2})$ and $\boldsymbol{h}_j(\theta) = \boldsymbol{h}_j(0)\cos(\frac{\theta}{2})$.

Now we can parameterize the positions of the units' endpoints. Let $\boldsymbol{n}_i$ denote the direction from $\boldsymbol{e}_i$ to $\boldsymbol{f}_i$, i.e., $\boldsymbol{n}_i = \frac{\boldsymbol{f}_i - \boldsymbol{e}_i}{\|\boldsymbol{f}_i - \boldsymbol{e}_i\|_2}$. Since we fixed the orientation of all the scissor units earlier, $\boldsymbol{n}_i$ stays constant. From basic trigonometry, the endpoints of the linkages can be described using these values:

$$\begin{aligned} \boldsymbol{e}_i(\theta) &= \boldsymbol{p}_i \cos\left(\frac{\theta}{2}\right) - L_i \boldsymbol{n}_i \sin\left(\frac{\theta}{2}\right), \\ \boldsymbol{f}_i(\theta) &= \boldsymbol{p}_i \cos\left(\frac{\theta}{2}\right) + L_i \boldsymbol{n}_i \sin\left(\frac{\theta}{2}\right), \end{aligned} \tag{1}$$

where $\boldsymbol{p}_i$ is a constant, indicating the position of $\boldsymbol{e}_i$ (and also of $\boldsymbol{f}_i$) at $\theta = 0$ (i.e., $\boldsymbol{p}_i = \boldsymbol{e}_i(0) = \boldsymbol{f}_i(0)$).

With this parameterization, the trajectory of a pair of endpoints $\boldsymbol{e}_i$ and $\boldsymbol{f}_i$ can be succinctly specified by a *track* $\boldsymbol{t}_i = (\boldsymbol{p}_i, \boldsymbol{n}_i, L_i)$, and furthermore, the trajectory of a hinge can also be specified by a "degenerate" track $(\boldsymbol{h}_j(0), \boldsymbol{0}, 0)$. **Key insight.** We formalize our intuition using tracks: if two scissor units are connected at two of their endpoints and the two corresponding tracks for those endpoints are identical, then the two units can contract and expand in the normal direction $\boldsymbol{n}$. If this is satisfied for all connected scissor units, then the entire linkage will contract with one degree of mobility.

4.2 Generalization in 3D

The above parameterization and the notion of tracks are independent from the number of dimensions, and thus, we can apply a variation of Hoberman's construction for constructing 3D linkages. Given a 3D surface mesh, we start with a *linkage graph*, whose nodes are points on the given surface (Fig. 5-a). The linkage graph sparsely approximates the given 3D surface and determines the linkage's topology. Here, we assume this graph is given, converting it into a uniformly deployable linkage with each graph node corresponding to a pair of endpoints (i.e., $\boldsymbol{e}_i$ and $\boldsymbol{f}_i$) of some scissor units (Fig. 5-b). In §4.3, we will present our design tool for interactively creating and editing such graphs.

In 3D, each scissor unit lies on a plane: the two bars are coplanar, and the rotation axis of the hinge is perpendicular to that plane. Unlike in the 2D construction, there may be three or more scissor units that have common endpoints (Fig. 6), ones that correspond to nodes on the linkage graph with valence at least 3. In addition to the analogous 2D linkage constraints, the constraints at each of these "multi-valent" points are that the dihedral angles (ϕ_i in Fig. 6-a) the scissor units make with each other must always be preserved.

Our strategy is to convert each graph node into a pair of endpoints and convert each edge into a chain of two scissor units. To obtain a deployable linkage using the idea of tracks introduced in §4.1, if two scissor units are connected at a pair of endpoints, tracks corresponding to these scissor units must be identical.

Expanding directions. We first compute the expanding directions $\boldsymbol{n}_i$ of the tracks. Because every pair of adjacent vertices in the linkage graph will be connected by scissor units, and every scissor unit lies on a plane, it suffices to have their tracks $\boldsymbol{t}_i$ and $\boldsymbol{t}_j$ be *coplanar*: the line passing through $\boldsymbol{p}_i$ and pointing along $\boldsymbol{n}_i$ must be coplanar with the line passing through $\boldsymbol{p}_j$ and pointing along $\boldsymbol{n}_j$ (Fig. 5-b). We construct a cost function that enforces the coplanarity,

$$E_{\boldsymbol{n}} = \sum_{e_{ij} \in \mathcal{E}} \Big[(\boldsymbol{n}_i \times \boldsymbol{n}_j) \cdot (\boldsymbol{p}_i - \boldsymbol{p}_j) \Big]^2,$$

where e_{ij} is the edge connecting nodes i and j, and the positions $\boldsymbol{p}_i$ of the graph nodes are fixed. Minimizing this cost function over $\boldsymbol{n}_i$ of all graph nodes is a nonlinear least-squares problem, and we use the CERES library [AMO15] to solve it, with all $\boldsymbol{n}_i$ initialized using surface normals of the 3D model at each $\boldsymbol{p}_i$. The output consists of expanding directions that minimize $E_{\boldsymbol{n}}$ while staying close to surface normals, the directions that naturally describe how a linkage should expand and contract.

For a graph of N nodes, this optimization has $3N$ control variables (i.e., x-, y- and z- components of N expanding directions). In all our examples, this number is more than the number of penalty terms in the cost function $E_{\boldsymbol{n}}$ (i.e., the number of edges), as the linkage graph

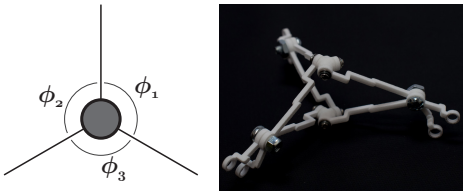


Figure 6: ***Multi-valent nodes.*** *A valence-3 graph node (left) and its corresponding linkage structure (right).*

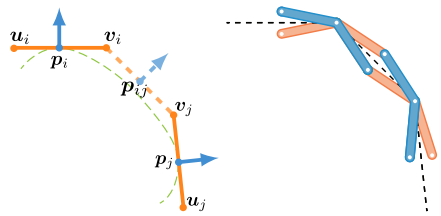


Figure 7: *Tracks and their "dual" polygon sides are represented with blue arrows and orange lines, respectively. The newly added track is depicted with dashed lines. On the right, we add scissor units in the same way as in Hoberman's 2D construction.*

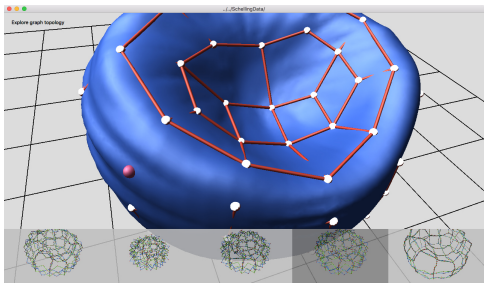


Figure 8: ***Our design interface*** *allows the user to interactively edit a linkage graph and explore different topologies (bottom), with the resulting linkages generated and visualized in real time (see video).*

does not need to be too dense to resemble a provided 3D surface. Consequently, the optimized expanding directions always produce a zero cost function, guaranteeing the coplanarity requirement.

From graph to linkage. We construct the 3D linkage by converting each graph edge into two connected scissor units. Consider two graph nodes i and j connected by an edge with tracks $\boldsymbol{t}_i = (\boldsymbol{p}_i, \boldsymbol{n}_i, L_i)$ and $\boldsymbol{t}_j = (\boldsymbol{p}_j, \boldsymbol{n}_j, L_j)$, respectively. Their normal directions $\boldsymbol{n}_i$ and $\boldsymbol{n}_j$ are those computed in the previous step. Consequently, $\boldsymbol{t}_i$ and $\boldsymbol{t}_j$ are coplanar, and connecting $\boldsymbol{t}_i$ and $\boldsymbol{t}_j$ using scissor units can be considered locally in 2D (see Fig. 7 for an illustration). Using ideas from Hoberman's construction from a polygon, we imagine the tracks $\boldsymbol{t}_i$ and $\boldsymbol{t}_j$ as two sides of a polygon on the plane where $\boldsymbol{t}_i$ and $\boldsymbol{t}_j$ lie. The orientations of both sides are perpendicular to the normal direction $\boldsymbol{n}_i$ and $\boldsymbol{n}_j$, the midpoints lie at $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$, and their lengths are $2L_i$ and $2L_j$, respectively.

To connect two tracks $\boldsymbol{t}_i$ and $\boldsymbol{t}_j$, we "complete" the partial polygon by adding another side between them. Since there are four possible choices for adding a new side (i.e., connecting one of $\boldsymbol{u}_i, \boldsymbol{v}_i$ with one of $\boldsymbol{u}_j, \boldsymbol{v}_j$ in Fig. 7), we choose the pair of vertices that are closest to each other in Euclidean distance. Suppose without loss of generality that the connected vertices are $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$. A new track $\boldsymbol{t}_{ij}$ is then placed at the midpoint $\boldsymbol{p}_{ij}$ of the new side, with its normal direction $\boldsymbol{n}_{ij}$ perpendicular to the new side and its length L_{ij} equal to half the distance between $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$. Following the 2D construction, two new scissor units are added: one with endpoints $\boldsymbol{p}_i$ and $\boldsymbol{p}_{ij}$ and hinge $\boldsymbol{v}_i$ and one with endpoints $\boldsymbol{p}_j$ and $\boldsymbol{p}_{ij}$ and hinge $\boldsymbol{v}_j$. By construction, these scissor units form matching tracks at p_i, p_j, and p_{ij}, thereby guaranteeing the resulting linkage's contractibility.

The above process uniquely constructs a linkage, as long as the track length L_i for every graph node i is given. We start by setting L_i heuristically to be proportional the minimal length of edges incident to node i:

$$L_i = \frac{1}{3} \min_{(i,j)\in\mathcal{E}} \|\boldsymbol{p}_i - \boldsymbol{p}_j\|_2.$$

The simple assignment of L_i produces well-shaped linkage structures for quick user preview. In §5, we will optimize L_i, the graph node positions and the bars' geometries to avoid collisions.

4.3 User-Guided Graph Design

With the linkage construction algorithm depicted above, we address the problem of creating the linkage graph. Provided a 3D surface shape, our first attempt was to generate the linkage graph fully automatically. The most straightforward approach is to use the dual of the surface mesh as the linkage graph. Unfortunately, we found that it is hard to produce a reasonably structured linkage from a surface mesh, even with various remeshing algorithms we experimented with. We made the following observation: unless largely simplified, using a mesh as the linkage graph produces a large number of scissor units (see video), thus significantly increasing the manufacture and assembling cost. However, when a mesh is heavily simplified, it becomes hard to preserve geometric features, and the resulting linkage would fail to resemble the given 3D shape. Moreover, the process of manufacturing and assembling linkages imposes additional limitations—for example, the valency of a graph node needs to be small, and the dihedral angle between two scissor units needs to be sufficiently large. Using a surface mesh is unable to account for all these criteria.

User interface. Inspired by the recent work on linkage editing [TCG*14, BCT15], we develop a tool for interactively editing linkage graphs (Fig. 8). Using this tool, the user can edit graph node positions by simply clicking and dragging on the surface of the 3D shape. One can also change the graph's connectivity by adding or removing edges. In real time, our method generates the linkage structure using the algorithm in §4.2 and allows the user to interactively fold and unfold the resulting linkage. To help the user edit the graph's topology intuitively, whenever the user selects a graph node, our tool samples different sets of edges that connect the selected node to its nearby graph nodes, while ensuring the valencies of the nodes are smaller than a user-defined number (typically 4). Meanwhile, it computes the resulting linkage for each sample configuration and presents them to the user. The user then chooses the most proper graph topology and moves on to the next editing operation (see video). As an extension, when the shape is symmetric with respect to a plane, our tool can optionally respect the symmetry: all the user editing operations are symmetrically duplicated to ensure that the graph and in turn the linkage structure stays symmetric.

Our editing tool serves an important purpose: the user needs to preserve the important geometric features of the 3D shape while discarding others to produce an aesthetically-pleasing and expressive linkage—a creative process that is hard to codify algorithmically.

5 Avoiding Collisions

In the process of folding, a linkage collides with itself unavoidably. In this section, we fix the linkage's topology but adjust its geometry to avoid collisions and thereby increase its compactness. Our intuition is illustrated using a 2D linkage in Fig. 9: many colli-

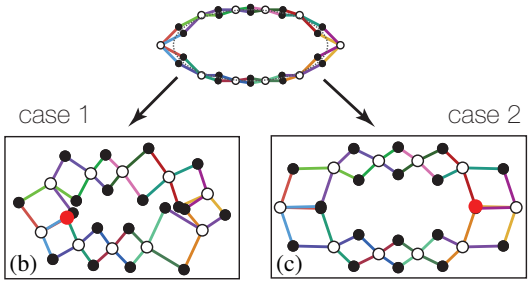


Figure 9: *(top) A 2D linkage resembling an ellipse contracts (left) until it collides, where the collision point is indicated by the red dot. Even a small perturbation of the linkage nodes leads to a very different collision state (right).*

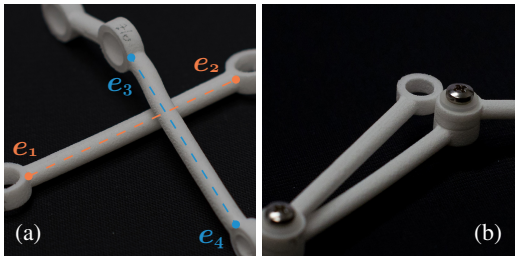


Figure 10: ***Collisions.*** *(a) A collision occurs when two scissor unit bars become coplanar and intersect.* ***(b)*** *The end joints and bolts can also cause collisions.*

sions can be avoided by perturbing the scissor lengths of tracks and node positions of the linkage graph on a 3D surface.

Challenging discontinuous optimization. Optimizing a linkage's geometry to avoid collisions is challenging, as the collision states change *discontinuously* with respect to the linkage's geometric parameters. Even an infinitesimal change of a node position can introduce or eliminate collisions in the process of linkage contraction. Thus, it is fundamentally hard to express how much a linkage can fold with respect to its geometric parameters in any analytical form. In addition, the optimization process needs to account for other geometric and mechanical constraints, such as the limitation of dihedral angles between scissor units and the preservation of geometric features of the original 3D shapes, many of which cannot be analytically expressed. Consequently, the widely used gradient-based optimization methods become inapplicable for solving this problem.

An alternative approach is to use stochastic optimization, one that samples geometric parameters probabilistically and evaluates a predefined objective function to accept or reject the samples. With sufficient samples, this method converges to find the optimum of the objective function without resorting to gradient computation. Since stochastic optimization requires many samples, the key to this approach is a *fast* evaluation of the objective function for each sample. In our problem, this demands a fast collision detection algorithm.

Algorithm overview. Our method for avoiding collisions has two steps: first, we use a modified simulated annealing method to optimize the graph node positions and the lengths of scissor units (§5.2). To make this approach efficient, a critical component is a fast algorithm for detecting the earliest collisions continuously (§5.1) and evaluating the objective function in the process of sampling. Furthermore, our sampling algorithm considers the mechanical and geometric constraints that the linkage needs to satisfy. This step aims to avoid collisions by adjusting the linkage geometry globally. After the stochastic optimization step, we adjust the linkage locally, taking into account physical parameters necessary for fabrication, such as the bars' thickness and joint sizes. To this end, we fix scissor units' endpoints and hinge positions, but locally deform the shape of the bars to avoid more collisions (§5.3).

5.1 Continuous Collision Detection

We derive a continuous collision detection algorithm with the assumption that the bars are infinitely thin. In fact, the bars' thickness depends on many factors, such as the fabrication material and the overall size of the linkage. Since our collision detection algorithm is to support the optimization of a linkage's intrinsic geometry (i.e., the endpoints, hinges and scissor lengths), we ignore the bars' thickness here in exchange for computational performance, but will consider it later in §5.3.

With this assumption, a scissor linkage is a set of straight line segments connecting endpoints with hinges (as illustrated in Fig. 9). Recall the linkage parameterization introduced in §4.1. According to Eq. (1), the positions of endpoints and hinges can be expressed in the form $\boldsymbol{e}_i(\alpha) = \alpha \boldsymbol{x}_i + \sqrt{1-\alpha^2}\boldsymbol{y}_i$, where $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ are $\boldsymbol{p}_i$ and $L_i\boldsymbol{n}_i$ in Eq. (1), respectively, and $\alpha = \cos(\frac{\theta}{2})$. For hinges, $\boldsymbol{y}_i$ vanishes. Since we start from the most expanded state and try to collapse the linkage as much as possible, α decreases from 1 (corresponding to $\theta = 0$) to 0 (corresponding to $\theta = \pi$). Our goal is to search for the largest $\alpha \in [0,1]$ that produces an intersection between the line segments.

Quadratic equation for collision detection. To check if two line segments are intersecting, we first check if the line segments are coplanar. Given two pairs of segment endpoints $(\boldsymbol{e}_1(\alpha), \boldsymbol{e}_2(\alpha))$ and $(\boldsymbol{e}_3(\alpha), \boldsymbol{e}_4(\alpha))$ (Fig. 10(a)), they are coplanar if and only if the following determinant vanishes:

$$\det(\boldsymbol{u}_2(\alpha), \boldsymbol{u}_3(\alpha), \boldsymbol{u}_4(\alpha)) = (\boldsymbol{u}_2(\alpha) \times \boldsymbol{u}_3(\alpha)) \cdot \boldsymbol{u}_4(\alpha) = 0, \quad (2)$$

where $\boldsymbol{u}_i(\alpha) = \boldsymbol{e}_i(\alpha) - \boldsymbol{e}_1(\alpha)$. Each $\boldsymbol{u}_i$ can also be expressed in the form of $\boldsymbol{u}_i(\alpha) = \alpha \boldsymbol{x}'_i + \sqrt{1-\alpha^2}\boldsymbol{y}'_i$ with $\boldsymbol{x}'_i = \boldsymbol{x}_i - \boldsymbol{x}_1$ and $\boldsymbol{y}'_i = \boldsymbol{y}_i - \boldsymbol{y}_1$. Expanding the determinant (2) yields the equation

$$0 = b_3\alpha^3 + b_2\alpha^2\sqrt{1-\alpha^2} + b_1\alpha(1-\alpha^2) + b_0(1-\alpha^2)\sqrt{1-\alpha^2}, \quad (3)$$

where

$$\begin{aligned} b_3 &= \det(\boldsymbol{x}'_2, \boldsymbol{x}'_3, \boldsymbol{x}'_4) \\ b_2 &= \det(\boldsymbol{y}'_2, \boldsymbol{x}'_3, \boldsymbol{x}'_4) + \det(\boldsymbol{x}'_2, \boldsymbol{y}'_3, \boldsymbol{x}'_4) + \det(\boldsymbol{x}'_2, \boldsymbol{x}'_3, \boldsymbol{y}'_4) \\ b_1 &= \det(\boldsymbol{y}'_2, \boldsymbol{y}'_3, \boldsymbol{x}'_4) + \det(\boldsymbol{y}'_2, \boldsymbol{x}'_3, \boldsymbol{y}'_4) + \det(\boldsymbol{x}'_2, \boldsymbol{y}'_3, \boldsymbol{y}'_4) \\ b_0 &= \det(\boldsymbol{y}'_2, \boldsymbol{y}'_3, \boldsymbol{y}'_4). \end{aligned}$$

Notice that each line segment represents a bar connecting one of a scissor unit's endpoints with its hinge. Without loss of generality, assume $\boldsymbol{e}_1$ and $\boldsymbol{e}_3$ are the hinges. Thus, $\boldsymbol{y}'_3$ vanishes, so $b_0 = 0$. Squaring and rearranging Eq. (3) yields a degree-6 equation in α,

$$\left(2b_1b_3 - b_1^2 - b_2^2 - b_3^2\right)\alpha^6 + \left(2b_1^2 - 2b_1b_3 + b_2^2\right)\alpha^4 - b_1^2\alpha^2 = 0.$$

Algorithm 1 Linkage Geometry Optimization

1: **while** not terminate **do**
2: sample graph node positions $\{\boldsymbol{p}_i\}$
3: project the sampled positions $\{\boldsymbol{p}_i\}$ on 3D surface
4: **if** fail_graph_check($\{\boldsymbol{p}_i\}$) **then continue**
5: **for** $k \leftarrow 1, N$ **do**
6: sample scissor lengths $\{L_i\}$
7: **if** fail_graph_check($\{\boldsymbol{p}_i\}, \{L_i\}$) **then continue**
8: $l_n \leftarrow$construct_linkage($\{\boldsymbol{p}_i\}, \{L_i\}$) ▷ §4.2
9: **if** fail_linkage_check(l_n) **then continue**
10: $E_n \leftarrow$objective_function(l_n) ▷ §5.1 and §5.2
11: **if** rand(0,1)<acceptance_probability(E_0, E_n,T) **then**
12: $E_0 \leftarrow E_n$; $l_0 \leftarrow l_n$ ▷ accept the sample
13: **end if**
14: **end for**
15: decrease_temperature(T)
16: **end while**

After factoring out α^2, we are left with solving just a quadratic equation in α^2. After finding candidate solutions, we check if the bars are actually coplanar at these α values.

Once we have α values that result in coplanar line segments, we find the intersection point between the lines containing the segments and check if the intersection is contained in both segments. In particular, we parameterize both line segments as $\boldsymbol{s}_a(t_a) = \boldsymbol{e}_1 + t_a(\boldsymbol{e}_2 - \boldsymbol{e}_1)$ and $\boldsymbol{s}_b(t_b) = \boldsymbol{e}_3 + t_b(\boldsymbol{e}_4 - \boldsymbol{e}_3)$. Equating the two parameterizations yields a linear system about t_a and t_b. If they are in the range $[0,1]$, then the two line segments collide at α.

In summary, our continuous collision detection algorithm iterates over all pairs of line segments that connect endpoints of scissor units and their hinges. For each pair of segments, we solve the aforementioned quadratic equation to find the earliest collision when the linkage collides. The entire process is fast to compute (typically taking less than 200 milliseconds for all our examples).

For fabricated linkages, the deployment angle cannot increase to π. This is because the revolute joints that connect two scissor units can block the bars from increasing the deployment angle (Fig. 10-b). In this case, we simply compute the maximum possible deployment angle for each scissor unit and use the corresponding α (rather than 0 as used above) to bound the collision check.

We note that the simplification to quadratic equations and small linear systems is made possible by assuming that the bars have no thickness. By adding thickness, we would have to check if the bars are within a distance threshold from each other, and the equation that describes such collisions has no closed-form solution (i.e. it is a higher-degree polynomial). We chose to ignore thickness because doing so significantly accelerates the stochastic optimization process, whose bottleneck is exactly the collision detection problem. We will consider the thickness and other geometric and mechanical constraints in the subsequent step.

5.2 Optimizing Linkage Geometry

Given a user-edited linkage graph, we now improve the compactness of the resulting linkage. We formulate this step as an optimization problem, whose variables are the positions of the graph nodes $\boldsymbol{p}_i$ and their associated scissor lengths L_i. The graph topology stays unchanged. Following the linkage construction algorithm in §4.2, these variables uniquely determine a linkage structure.

Objective function. There are many choices for measuring the compactness of a linkage. For example, one choice uses the deployment angle when the earliest linkage self-collision occurs. We propose to use the volume of the bounding box at the earliest collision, as it best represents how much space the linkage saves in its most contracted state. Evaluating this objective function first requires performing the continuous collision detection (§5.1). Then, using the deployment angle at which the collision occurs, we compute the corresponding state of the linkage following the parameterization (1) and evaluate the bounding box volume. We also note that the optimization algorithm we propose here does not depends on any particular form of objective function.

Modified Simulated Annealing. Because of the discontinuous nature of the collision state with respect to the graph configuration, we opt for a gradient-free optimization method. We choose to use a modified simulated annealing method [VLA87] (Algorithm 1). The only difference from the standard version is that we sample the scissor lengths L_i more frequently than node positions $\boldsymbol{p}_i$ (typically $N = 10$ in Line 5 of Algorithm 1), because sampling graph nodes involves projecting the sampled positions back onto the provided 3D surface (Line 3 of Algorithm 1), which is more expensive than sampling the scissor lengths. Thus, this sampling scheme leads to faster performance than the standard approach.

Constraints. For every sampled linkage generated during the optimization, we check it against the following mechanical and geometric constraints and reject if it fails any of the checks. To preserve the geometric features, we check if the sampled graph nodes is within a distance threshold away from the user-edited graph node (Line 4 of Algorithm 1). The distance threshold is proportional to the "geometric saliency" of the graph nodes, computed using the method of Chen et al. [CSPF12]. While the graph nodes are sampled on the 3D surface, the hinges and the endpoints that connects two graph nodes may be off the surface (see Fig. 7(a), where the dotted line indicates an object's surface). We therefore check if those points are within a threshold away from the surface (Line 9 of Algorithm 1).

In addition, we also ensure that the distance between each pair of graph nodes is larger than a threshold (Line 4 of Algorithm 1), if each sampled scissor length L_i is bounded (Line 7 of Algorithm 1), and if the dihedral angle between each pair of connected scissor units is larger than a threshold. All these checks guarantee that the optimized linkage can be easily fabricated and assembled.

5.3 Deforming the Bars

We now consider bars' thickness and volumes of the revolute joints, as they can cause collisions that were not predicted in the previous collision detection step, or even worse render the linkage mechanically impossible to fabricate. To this end, our key observation is that even though almost all existing linkage construction methods, including our previous step, assume straight bars, the straightness is in fact unnecessary. We demonstrate that by bending them, we can avoid more collisions and contract the linkages further,with about 10% improvement in terms of bounding box volume.

Briefly, after optimizing the linkage graph with simulated an-

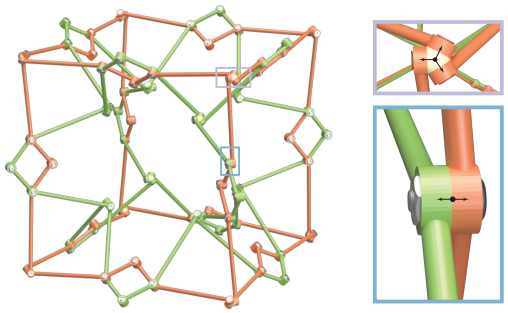


Figure 11: *To construct a mechanically sound linkage, the positions of the bars' endpoints are offset from the joint centers (inset). These offsets need to be considered to avoid collisions.*

nealing, we use it to generate the initial geometry, in which scissor units are made from straight bars with nonzero thickness, and all the pieces are offset from each other so that they do not intersect (Fig. 11). We then contract the linkage. To avoid collisions, we try to deform the colliding bars, terminating when a bar is too deformed or a collision cannot be avoided. Conceptually, we treat each bar as an elastic body with fixed endpoints, and while the linkage contracts, the collisions deform the bars plastically.

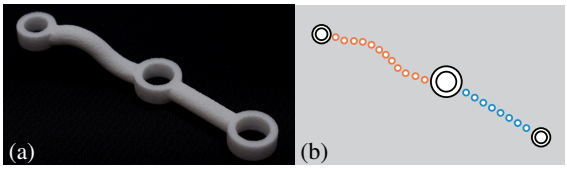


Figure 12: *When deforming a linkage's bar **(a)**, we represent it using two chains of particles connecting from its endpoints to its hinge (red and blue chains in **(b)**).*

Optimization problem. We formulate our strategy as a continuous optimization problem. We first discretize every bar into two chains of particles, each connecting one of its endpoints to its hinge (Fig. 12). The radius of each particle equals the bar thickness used in fabrication. Thus, the optimized variables are the positions of these particles in each scissor unit's local frame of reference, except the beginning and ending particles of each chain, which are fixed. In addition, we place a particle at the position of each endpoint and hinge point. The radii of those particles is as large as the radii of the bounding spheres of the mechanical joints at those points. These particles are all fixed, serving only for collision detection.

Elastic energy. Borrowing concepts from continuum mechanics [Ies08, BWR*08], we define two energy terms for each chain of particles to resist deformation, namely stretching and bending. Fig. 13 summarizes our indexing and notation. The stretching energy of a single chain of particles (denoted as $\mathcal{C}$) simply penalizes

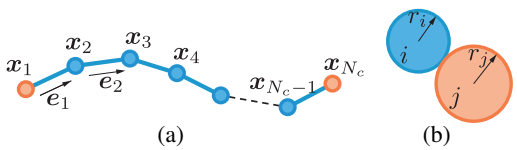


Figure 13: *Notation used for elastic energy formulation **(a)**, and for collision energy formulation **(b)**. Both $\boldsymbol{x}_1$ and $\boldsymbol{x}_{N_c}$ are fixed particles.*

the distance between two consecutive particles:

$$E_s(\mathcal{C}) = \sum_{i=1}^{N_\mathcal{C}-1} \|\bar{\boldsymbol{e}}^i\|_2 \|\boldsymbol{x}_i - \boldsymbol{x}_{i+1}\|_2^2,$$

where $\|\bar{\boldsymbol{e}}^i\|_2$ is the length of the undeformed particle segment i. Because the starting and ending particles are fixed, minimizing this energy tends to produce uniformly distributed particles along the chain. Following the model proposed in [BWR*08], the bending energy involves the discretized curvature, defined as

$$E_b(\mathcal{C}) = \sum_{i=1}^{N_\mathcal{C}} \frac{\|\kappa \boldsymbol{b}_i\|_2^2}{\|\bar{\boldsymbol{e}}^{i-1}\|_2 + \|\bar{\boldsymbol{e}}^i\|_2},$$

where $\kappa \boldsymbol{b}_i$ is the discretized curvature binormal

$$\kappa \boldsymbol{b}_i = \frac{2\boldsymbol{e}^{i-1} \times \boldsymbol{e}^i}{\|\bar{\boldsymbol{e}}^{i-1}\|_2 \|\bar{\boldsymbol{e}}^i\|_2 + \boldsymbol{e}^{i-1} \cdot \boldsymbol{e}^i}$$

at particle i. The total elastic energy of a single chain $\mathcal{C}$ is $E_s(\mathcal{C}) + \alpha E_b(\mathcal{C})$, where α is a weight to balance both terms (typically $\alpha = 0.8$). When there are no collisions, minimizing this energy results in a straight chain of particles.

Collision energy. Next, we discretize the deployment angle $\theta \in [0, \pi]$ into N intervals, each corresponding to a discretized timestep in contracting the linkage. At each step s, we consider each pair of particles (i, j) and define the collision energy

$$E_c(i, j, s) = f\left(\frac{\|\boldsymbol{x}_i(s) - \boldsymbol{x}_j(s)\|_2}{r_i + r_j}\right)^2,$$

where r_i and r_j are values slightly larger (typically $1.1\times$ larger) than the radius of particle i and j. One of the two particles can be the particle at an endpoint or hinges to account for possible collisions between bars and joints. $\boldsymbol{x}_i(s)$ is the position of particle i at the contraction step s in the world frame of reference, and f is a function that vanishes when the two particles are separated:

$$f(d) = \begin{cases} \frac{1}{d} + d - 2, & d < 1 \\ 0, & \text{otherwise.} \end{cases}$$

We note that this collision energy slightly differs from one typically used in elastic rod simulations (e.g., [KJM08]), because we prefer to define all energy terms as a sum of squared functions, so that minimizing this energy can leverage full-fledged nonlinear least-squares methods (e.g., [AMO15]).

To solve the optimization problem, we increase the contraction step s iteratively, starting from 0. At each step s, we minimize:

$$E = \sum_\mathcal{C} (E_s(\mathcal{C}) + \alpha E_b(\mathcal{C})) + \frac{\beta}{s} \sum_{k=0}^{s} \sum_{(i,j)} E_c(i, j, k),$$

with β being a scalar to balance elastic and collision energies (typically $\beta = 0.2$ in our examples). Minimizing this energy amounts to avoiding collisions up to timestep s while keeping the bars as straight as possible. We iteratively increase s until a step where the elastic energy is so strong that the resulting particle shapes fail to avoid collisions. At each step, we solve this nonlinear least-squares problem using a quasi-Newton method with a warn-start initialization resulting from the previous solve.

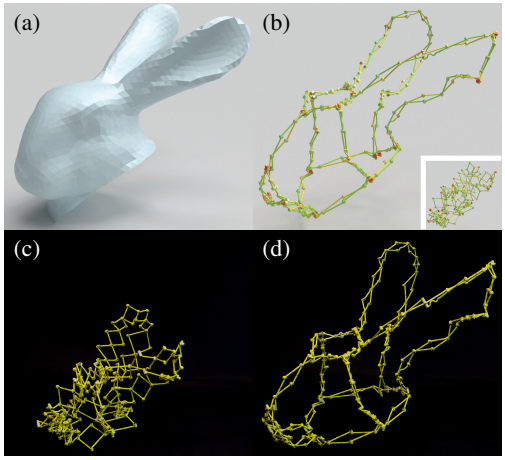


Figure 14: BUNNY. *We fabricated and assembled the* BUNNY*'s head* ***(a)*** *into a functional linkage* ***(b-d)****.*

Finally, we extrude the curves defined by particles into bars. Joining these bars with standard revolute joints, we produce the final geometry of bars that are ready for fabrication.

6 Results

We applied our algorithm to design eight examples, starting from an input mesh and ending with an optimized geometry of a linkage. We fabricated a simple cube example (Fig. 5 and Fig. 15) to validate our linkage construction algorithm and two more complex examples (Fig. 1 and Fig. 14) to validate the entire pipeline. The major bottleneck of the design and fabrication process is in fact the fabrication and assembly, as both examples consist of hundreds of bars and revolute joints, and assembling them is time-consuming. In the following, we briefly discuss our examples (§6.1), followed by a comparison to validate our compactness optimization (§6.2).

6.1 Examples

Our examples have a wide range of geometric and topological details (see statistics in Table 1). In our animal zoo examples, including the BUNNY (Fig. 14), WHALE (Fig. 16(a)), BIRD (Fig. 1), and OCTOPUS (Fig. 16(d)), small features like the bunny's ears and the bird's wings are captured with valence 2 vertices in the linkage graphs. The symmetry-preserving option in our interface enabled the symmetric linkages for the bird and whale.

Our examples resemble shapes with different topologies. BOB, the duck-shaped lifesaver (Fig. 16(c)), demonstrates that our approach easily generalizes to surfaces of nonzero genus. The OCTOPUS example shows that our method can also handle open chains of

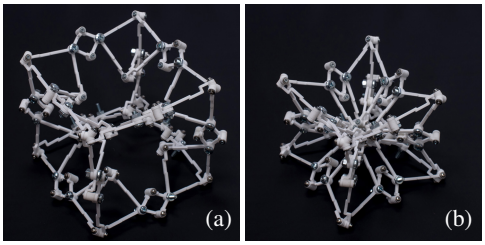


Figure 15: CUBE. *A fabricated cubic linkage in expanded* ***(a)*** *and contracted* ***(b)*** *states.*

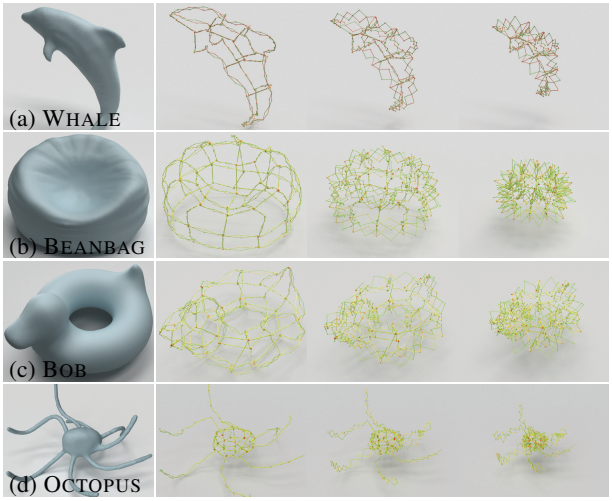


Figure 16: *Contracting Shapes.* *Four additional deployable linkages that resemble different shapes (see video).*

scissor units for depicting the OCTOPUS's tentacles. The BEANBAG example (Fig. 16(b)) is inspired by foldable furnitures [LHAZ15], demonstrating a beanbag-shaped contractible chair.

6.2 Comparison

The improvements in the contractibility is summarized in Table 1 with the linkages' complexity and the contraction ratio of bounding volumes after each step of our method. The contraction ratio is defined as the ratio of bounding volume of the linkage at its most contracted state to the volume at its most expanded state. The "original" ratio is computed after the user design step; the "repositioned" ratio is computed after the geometry optimization step (§5.2); and the "deformed" ratio is computed after the final shape deformation step (§5.3). We also animate these comparisons in the video.

7 Conclusion

We presented a method for constructing deployable scissor linkages in 3D to resemble user-provided 3D shapes. By optimizing the linkages' joint placement and the pieces' geometries, we were able to improve its contractibility. We demonstrated the correctness of our method by fabricating the output geometries, manually assembling the final linkage, and verifying its compactness.

Limitations and future work. However, we noted that while the fabricated results were able to contract and expand as predicted, it was sometimes difficult to actually move the linkage along its degree of freedom. Our optimization procedure only takes the geometry into account. For example, we did not conduct any kind of stress analysis on the linkages, nor did we take into account the friction at each joint. The ease of contractibility can be increased by considering these mechanical properties, perhaps by adding additional scissor units for structural support, augmenting the optimization process with structural analyses, or by determining optimal locations for actuators to automatically contract and expand the linkage.

Uniform contraction is not the most optimal way of minimizing the bounding volume of the linkage. For concave objects, it would be more effective to design a linkage that changes its intrinsic shape while contracting. The method of Zhang et al. [ZWC*15] generates

	geometric complexity				contraction ratio			timings (min)	
	# nodes	# edges	# SUs	# joints	original	repositioned	deformed	reposition	deformation
BUNNY	27	34	68	120	0.776	0.293	0.227	32.5	40.2
BIRD	35	41	82	150	0.724	0.167	0.138	46.6	52.8
WHALE	32	43	86	160	0.774	0.354	0.302	40.5	64.2
BEANBAG	59	106	212	330	0.535	0.174	0.158	53.2	84.3
BOB	58	102	204	320	0.631	0.462	0.392	78.7	94.5
OCTOPUS	71	92	184	310	0.606	0.282	0.223	82.4	104.7

Table 1: ***Statistics.*** *The user-edited linkage graphs ("original") generate linkages which contract, and the amount it can contract is improved by the stochastic optimization procedure ("repositioned") and local deformation ("deformed"). Here, we report the complexity of the geometry in terms of number of graph nodes, edges, scissor units, and number of joints. The compactness is measured as the ratio of the bounding box volumes between the most contracted and expanded states.*

linkages that change shape, but they only demonstrated its applications on 2D linkages with one redundant constraint (i.e., a single loop of scissor units). For example, in the 3D linkages we generate, there can be arbitrarily many redundancies. Constructing a 3D scissor linkage that transforms from one shape to another shape remains an open problem. Lastly, in our examples, we fabricate the scissor units' bars individually and assemble them together with screws, bolts, and nuts. This is a laborious and time-consuming process, much longer than the computation time. Initially, we attempted to 3D print the entire linkage in one piece. However, the clearance needed at each joint causes the final linkage to be too loose. Thus, it is of practical importance to design new method for fabricating complex joints in a single piece.

Acknowledgments

This work is partially supported by the National Science Foundation of U.S. (CAREER-1453101) and China (No. 61303136), as well as generous gifts from Intel and Adobe. We also thank Chang Xiao for his help of assembling the linkages.

References

[AMO15] AGARWAL S., MIERLE K., OTHERS: Ceres solver. http://ceres-solver.org, 2015. 4, 8

[BBJP12] BÄCHER M., BICKEL B., JAMES D. L., PFISTER H.: Fabricating articulated characters from skinned meshes. *ACM Trans. Graph. (Proc. SIGGRAPH) 31*, 4 (2012). 2

[BCT15] BÄCHER M., COROS S., THOMASZEWSKI B.: Linkedit: Interactive linkage editing using symbolic kinematics. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2015) 34*, 4 (Aug. 2015). 1, 2, 5

[Ben03] BENNETT G.: A new mechanism. *Engineering 76*, 777 (1903), 2. 1

[BST*08] BERNHARDT P. A., SIEFRING C. L., THOMASON J. F., RODRIQUEZ S. P., NICHOLAS A. C., KOSS S. M., NURNBERGER M., HOBERMAN C., DAVIS M., HYSELL D. L., ET AL.: Design and applications of a versatile HF radar calibration target in low Earth orbit. *Radio Science 43*, 1 (2008). 1

[BWR*08] BERGOU M., WARDETZKY M., ROBINSON S., AUDOLY B., GRINSPUN E.: Discrete elastic rods. *ACM Trans. Graph. 27*, 3 (Aug. 2008), 63:1–63:12. 8

[CCA*12] CALÌ J., CALIAN D. A., AMATI C., KLEINBERGER R., STEED A., KAUTZ J., WEYRICH T.: 3d-printing of non-assembly, articulated models. *ACM Transactions on Graphics 31*, 6 (2012), 130. 2

[CDXF14] CAI J., DENG X., XU Y., FENG J.: Constraint analysis and redundancy of planar closed loop double chain linkages. *Advances in Mechanical Engineering 6* (2014), 635423. 3

[CLM*13] CEYLAN D., LI W., MITRA N. J., AGRAWALA M., PAULY M.: Designing and fabricating mechanical automata from mocap sequences. *ACM Trans. Graph. 32*, 6 (Nov. 2013), 186:1–186:11. 2

[CSPF12] CHEN X., SAPAROV A., PANG B., FUNKHOUSER T.: Schelling points on 3D surface meshes. *ACM Transactions on Graphics (SIGGRAPH 2012)* (Aug. 2012). 7

[CTN*13] COROS S., THOMASZEWSKI B., NORIS G., SUEDA S., FORBERG M., SUMNER R. W., MATUSIK W., BICKEL B.: Computational design of mechanical characters. *ACM Trans. Graph. 32*, 4 (July 2013), 83:1–83:12. 1, 2

[Hob90] HOBERMAN C.: Reversibly expandable doubly-curved truss structure, July 24 1990. US Patent 4,942,700. 2, 3

[Hob91] HOBERMAN C.: Radial expansion/retraction truss structures, June 18 1991. US Patent 5,024,031. 1, 2, 3

[Ies08] IESAN D.: *Classical and generalized models of elastic rods*. CRC Press, 2008. 8

[Kin95] KING H. C.: Planar linkages and algebraic sets. 2

[KJM08] KALDOR J. M., JAMES D. L., MARSCHNER S.: Simulating knitted cloth at the yarn level. *ACM Trans. Graph. 27*, 3 (Aug. 2008). 8

[LHAZ15] LI H., HU R., ALHASHIM I., ZHANG H.: Foldabilizing furniture. *ACM Transactions on Graphics, (Proc. of SIGGRAPH 2015) 34*, 4 (2015). 2, 9

[LPW*06] LIU Y., POTTMANN H., WALLNER J., YANG Y.-L., WANG W.: Geometric modeling with conical meshes and developable surfaces. In *ACM Transactions on Graphics (TOG)* (2006), vol. 25, ACM, pp. 681–689. 2

[McC06] MCCARTHY J. M.: *Geometric design of linkages*, vol. 11. Springer Science & Business Media, 2006. 2

[Piñ65] PIÑERO E. P.: Three dimensional reticular structure, May 1965. US Patent 3185164. 2

[RMDT13] ROOVERS K., MIRA L. A., DE TEMMERMAN N.: From surface to scissor structure. In *Proceedings of the First Conference Transformables, Seville, Editorial Starbooks, Seville, Spain* (2013). 2

[SZ15] SUN T., ZHENG C.: Computational design of twisty joints and puzzles. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2015) 34*, 4 (Aug. 2015). 2

[TCG*14] THOMASZEWSKI B., COROS S., GAUGE D., MEGARO V., GRINSPUN E., GROSS M.: Computational design of linkage-based characters. *ACM Trans. Graph. 33*, 4 (July 2014), 64:1–64:9. 1, 2, 5

[VLA87] VAN LAARHOVEN P. J., AARTS E. H.: *Simulated annealing: theory and applications*. Springer Science & Business Media, 1987. 7

[YP97] YOU Z., PELLEGRINO S.: Foldable bar structures. *International Journal of Solids and Structures 34*, 15 (1997), 1825–1847. 2, 3

[ZSMS14] ZHOU Y., SUEDA S., MATUSIK W., SHAMIR A.: Boxelization: Folding 3d objects into boxes. *ACM Trans. Graph. 33*, 4 (July 2014), 71:1–71:8. 2

[ZWC*15] ZHANG R., WANG S., CHEN X., DING C., JIANG L., ZHOU J., LIU L.: Designing planar deployable objects via scissor structures. *IEEE Trans. Vis. Comp. Graph.* (2015). 2, 9