LinkEdit: Interactive Linkage Editing using Symbolic Kinematics

Moritz Bächer[1] Stelian Coros[1,2] Bernhard Thomaszewski[1]
[1]Disney Research Zurich [2]Carnegie Mellon University

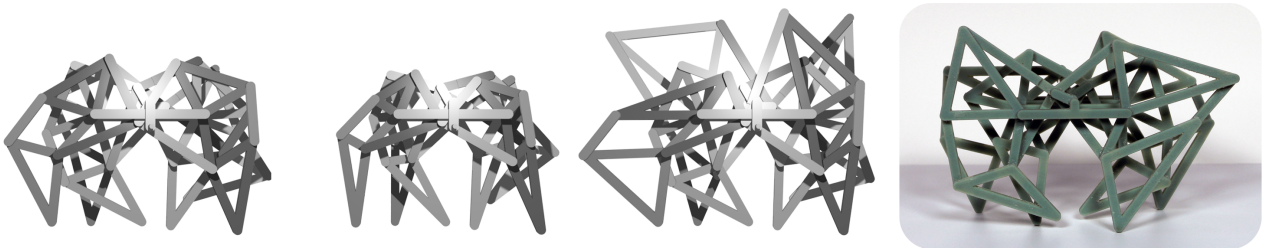


Figure 1: *Our interactive editing system allows users to adapt the shape and motion of planar linkages in an intuitive manner. As shown on the* Strandbeest *sculpture (*1^{st} from left*), we support diverse edits including changes to the enclosure of the assembly (*2^{nd}*) or the shape of individual components (*3^{rd}*). Our editing tools preserve correct functioning at all times, as illustrated on a 3D-printed prototype (*4^{th}*).*

Abstract

We present a method for interactive editing of planar linkages. Given a working linkage as input, the user can make targeted edits to the shape or motion of selected parts while preserving other, e.g., functionally-important aspects. In order to make this process intuitive and efficient, we provide a number of editing tools at different levels of abstraction. For instance, the user can directly change the structure of a linkage by displacing joints, edit the motion of selected points on the linkage, or impose limits on the size of its enclosure. Our method safeguards against degenerate configurations during these edits, thus ensuring the correct functioning of the mechanism at all times. Linkage editing poses strict requirements on performance that standard approaches fail to provide. In order to enable interactive and robust editing, we build on a symbolic kinematics approach that uses closed-form expressions instead of numerical methods to compute the motion of a linkage and its derivatives. We demonstrate our system on a diverse set of examples, illustrating the potential to adapt and personalize the structure and motion of existing linkages. To validate the feasibility of our edited designs, we fabricated two physical prototypes.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: linkages, animation, fabrication, interactive editing

1 Introduction

As 3D printers become widely available, more and more printable content is published online. Ranging from artwork to replacement parts and mechanically functional models, there are numerous catalogs with myriad printable models to download and replicate at home. While there is interest and value in replication, however, this usage forgoes the true potential of 3D printers: *personalization*—the opportunity to adapt content to individual needs and preferences.

A variety of existing software tools allow for intuitive editing of static digital models. Some of these tools even maintain or establish structural stability by virtue of finite element analysis. To date, however, no equivalent CAD software has been developed for mechanically-functional models. Yet, a vast array of mechanically-functional objects such as kinetic sculptures, fold-away furniture, electro-mechanical toys, and even robots stand to benefit from user-friendly editing tools and the corresponding potential for personalization that they would bring about.

We focus on the challenge of editing planar linkages, which are the beating heart of many mechanically-functional models—and notoriously difficult to design and edit: even for simple linkages, the relation between joint displacements and resulting change in motion-curves is complex and very difficult to predict. Within the group of planar linkages, our method can handle a large set of well-known and widely-used mechanisms.

Editing a linkage means changing certain aspects of its shape or motion while others are preserved. The reasons for editing linkages are diverse: for example, one may want to personalize an existing design by adapting the aesthetic appeal of its shape while maintaining its function, or one may want to alter the motion in order to reuse the linkage in a different context. In order to support such customization, we propose a set of interactive tools and high-level editing abstractions. Our method complements the recent body of work aimed at creating mechanically-functional objects [Zhu et al. 2012; Coros et al. 2013; Ceylan et al. 2013; Thomaszewski et al. 2014] and is designed to leverage the wealth of mechanical designs that can be found in online repositories such as *GrabCAD* and *Thingiverse*.

Overview & Contributions In this work, we present an interactive tool for intuitive editing of existing planar linkages, i.e., assemblies of interconnected rigid components that perform planar motion. Although mechanisms can involve other components such as gear-trains and cams, planar linkages are arguably the most challenging to design and edit. In order to accommodate the work flow that we propose, the design tool must *a*) be fast enough to provide interactive rates, *b*) offer abstractions and high-level metaphors for intuitive editing, and *c*) preserve correct functioning of the mecha-

nism at all times. Interactivity is a key requirement to enable quick and intuitive navigation of the design space of a given linkage, but we found that existing approaches cannot provide the necessary speed. The stringent performance requirements drive us to develop faster algorithmic solutions that distinguish our method from existing work on mechanism design. In particular, we depart from the typical constraint-based simulation in favor of a symbolic kinematics algorithm that uses closed-form expressions to determine the motion of a planar linkage. As a vital component of linkage editing, we further propose efficient solutions for computing state derivatives and preventing singular configurations.

We have used our system to edit a representative set of mechanisms, most of which are freely available on the internet. As indicated by the results, our method can accommodate a wide range of planar linkages and diverse edits. In addition to showing results in simulation, we also fabricated physically prototypes in order to demonstrate the validity of our personalized designs.

2 Related Work

Digital content generation has been a core topic of computer graphics since its very beginnings. Unsurprisingly, many techniques and concepts developed for creating virtual assets are now beginning to make their way into the digital fabrication pipeline. For example, CAD software packages specifically designed for 3D Printing, such as Autodesk's MeshMixer [2015], implement a variety of tools for mesh editing (sculpting, deformation, transferring parts between different 3D objects, etc) that find their roots in graphics research [Perry and Frisken 2001; Sorkine et al. 2004]. However, bringing virtual objects to the real world through 3D printing requires many additional challenges to be addressed.

Design for Digital Fabrication When it comes to digital fabrication, and in particular 3D-printing, a central challenge is to improve the reliability of the output. To this end, several methods have recently been developed to improve the strength-to-weight ratio of fabricated models [Stava et al. 2012; Zhou et al. 2013; Lu et al. 2014], or to create optimized support structures that enable complex 3D prints [Dumas et al. 2014]. However, the true power of digital fabrication is the ability to personalize the objects that are to be manufactured. Consequently, a variety of recent works propose methods for generating 3D-printable objects with, e.g., desired appearance properties [Weyrich et al. 2009; Hasan et al. 2010; Dong et al. 2010] or deformation behaviors [Bickel et al. 2010; Bickel et al. 2012; Skouras et al. 2012; Skouras et al. 2013]. Other methods have been proposed to allow casual users to design furniture pieces that are unique, stable and functional [Lau et al. 2011; Umetani et al. 2012], to quickly design prototypes that test and validate the way in which finished products will work [Koo et al. 2014], or to statically and dynamically balance intricate shapes [Prévost et al. 2013; Bächer et al. 2014]. The potential to reuse existing designs has been recognized in several of these works and was the topic of investigation for the recent method proposed by Schulz et al. [2014]. By empowering average users to intuitively edit planar linkages, our method shares the goal of reusing existing designs. However, rather than focusing on *static* objects such as furniture pieces, our focus is on *mechanically-functional* models that combine shape and motion through complex kinematic relationships.

Mechanism Design Understanding and designing mechanical objects by hand is notoriously difficult as it requires a great deal of skill, experience and engineering knowledge. In order to make this process accessible to the general public, a variety of computational methods have been proposed by the graphics community. For example, Mitra et al. [2010] introduced a method that generates intuitive visual aids to illustrate the motion of mechanisms. Several methods for synthesizing 3D printable mechanisms whose motions resemble those of virtual characters have also been proposed in the computer graphics community [Zhu et al. 2012; Coros et al. 2013; Ceylan et al. 2013; Thomaszewski et al. 2014; Megaro et al. 2014].

Outside the field of computer graphics, the problem of mechanism design has a rich history and continues to be studied extensively today [Burmester 1888; Freudenstein 1954; Kaufman and Maurer 1971; Erdman et al. 2001; Myszka et al. 2013]. As is the case for our work, these methods typically start from a template mechanism that is then edited to perform a desired motion or to fit a desired mechanical advantage profile. However, while powerful in the hands of a professional designer, methods developed in the mechanical engineering community are generally not meant to be used by casual users. For instance, state-of-the-art mathematical models allow singular configurations of closed-loop linkages to be traced out as individual design parameters vary [Myszka et al. 2013]. This level of information detail is very likely to be overwhelming for casual users as there are no obvious and direct ways in which it can be used to generate the envisioned designs. In this work, we therefore focus on developing intuitive interaction modes that allow casual users to exert direct and intuitive control over the design. Importantly, by safeguarding against singularities, our method automatically ensures the feasibility of the mechanical designs throughout the entire editing process. As discussed in prior work, singular configurations can be identified through zero singular values of the Jacobian of a mechanism's set of constraints [Erdman et al. 2001]. Based on this criterion, Thomaszewski et al. [2014] employed a stochastic optimization method to ensure that the generated linkages maintain a safe distance to singular configurations. However, due to its computational overhead, their formulation is only appropriate for an offline process. Instead, we seek to provide users of our system with immediate feedback in order to support an interactive design experience. To this end, we present a formulation that lends itself to efficient, derivative-based optimization methods.

Feature-Aware Model Editing Editing the shape of existing digital models is a core problem in geometry processing. Closest to our application are higher-level editing tools that aim to preserve both the global characteristics of the model and specific structures typically found in man-made objects [Gal et al. 2009; Bokeloh et al. 2011; Bokeloh et al. 2012]. Such high-level editing tools infer the parts of a model that can be deformed freely and the geometric features that must be preserved or parameterized in specific ways. Functional relationships or algebraic models are then used to map user inputs, typically handles that are interactively manipulated, to changes in the model's geometry. Inspired by the ease-of-use of these methods, our editing tools pursue the guiding principle of minimal invasion: in order to satisfy a user-specified editing objective, parts of the linkage should adapt automatically and as much as necessary, but the remainder should change as little as possible. Building on this principle, we propose a set of tools that allow the user to make selective edits to the shape and motion of a given linkage while preserving other, e.g., functionally-important aspects.

3 Closed-Form Kinematics

As a primordial requirement for interactive linkage editing, we must be able to compute their motion at sufficiently fast rates. For the sake of efficiency, we focus on purely kinematic approaches in this work—but even in this case, there is a large number of alternatives [Laulusa and Bauchau 2008]. A standard approach is to represent all components of a mechanism as rigid-bodies and impose constraints that model the different types of connections between the

bodies. While general and powerful, the computational burden of constraint-based simulation can be substantial, as it requires the solution of nonlinear systems of equations. An alternative that has been explored in computational engineering is symbolic processing [Kecskeméthy et al. 1997; Uchida and McPhee 2012], which provides solutions in closed-form using recursive computations. Rather than solving systems of equations that involve all degrees-of-freedom at once, the basic principle of this class of approaches is to decompose a mechanism, possibly with multiple loops, into independent parts that can be processed in isolation and in order. Due to its promise in computational efficiency, we pursue a symbolic kinematics approach in this work.

Relation to Existing Work Among the existing works on symbolic kinematics, our approach is perhaps closest to the one by Kecskeméthy et al. [1997], which we briefly summarize here for comparison. A given mechanism is decomposed into sets of overlapping loops such that, in each loop, only two rigid components meet in a given joint. The degrees of freedom (DOFs) are rigid-body transformations between neighboring components and a closure condition enforces that the transformations match up at the beginning and end of the loop. In this way, six DOFs can be expressed as a function of the remaining DOFs, a fact that is leveraged to derive coupling conditions between all loops that meet in a given joint. Finally, an ordering algorithm determines a numbering of the loops that allows for sequential processing. While we follow a similar principle, our approach allows us to readily compute derivatives of state with respect to parameters and to detect and avoid singular configurations. These are quintessential requirements for linkage editing: state derivatives are needed to transform a desired change in state to a corresponding change in parameters; singularities need to be prevented to maintain correct functioning of the linkages at all times.

3.1 Representation

We assume that a mechanical assembly consists of n_c rigid components interconnected by n_j joints. For the sake of efficiency, we renounce explicit representations of rigid components in terms of position and orientation. Instead, we retain only the joint positions as degrees of freedom but keep track of distance constraints between neighboring joints, i.e., pairs of joints that pertain to a given rigid component. The configuration of a linkage is thus defined through its time-varying joint positions $\mathbf{x}_i(t)$, $1 \leq i \leq n_j$, and we denote the initial configuration as $\bar{\mathbf{x}} = \mathbf{x}(0)$. For simplicity, we further assume that all joints are revolute (i.e., pin joints) and that there is a single motor $m(t)$ controlling the position of the first joint, $\mathbf{x}_1(t) = \mathbf{x}_1(m(t))$. Finally, we require that at least two of the joints remain fixed or move in prescribed ways and that the remainder of the assembly is fully-constrained. The motor value at any given time thus uniquely determines the configuration of the linkage.

3.2 Joint Ordering

Our symbolic kinematics algorithm follows a simple principle: since the distances between neighboring joints are constant, the position of a given joint can be determined as soon as the positions of two neighboring joints are known. This construction is analogous to determining the position of a triangle's node from its two other node positions and the lengths of its two incident edges. In order to successively determine the joint positions in this way, we first have to find an appropriate ordering of the joints. We start by setting up a graph with nodes corresponding to the linkage's joints and whose edges correspond to distance constraints imposed by the rigid components. Each node holds an index, a flag indicating whether it has been visited by the ordering algorithm yet, and a reconstruction rule that explains its geometric relation to previously visited nodes. As explained in Algorithm 1, we initially assign indices for the motor and all fixed joints and mark them as visited. We also insert all unvisited neighbors of these initial nodes into an active list. The algorithm then iterates over the active list, retrieving a node k and checking whether at least two of its neighboring nodes have been visited. If this is the case, we retain two of the visited neighbor nodes, i and j, and add a corresponding reconstruction rule $(i, j) \rightarrow k$. Node k is then assigned the next index, marked as visited and removed from the list, while all of its unvisited neighbors are added. As soon as rules for two joint positions on a given component are known, we add corresponding rules for all remaining joints on that component. The process terminates once the active list is empty.

Algorithm 1 Joint Ordering

Require: initialNodes //fixed joints and motor
Require: activeList
for all $i \in$ initialNodes **do**
  activeList.insert(neighbors(initialNodes(i)))
end for
while !activeList.empty() **do**
  k = activeList.pop_front()
  vn = visitedNeighbors(k) //list of visited neighbors
  if vn.size()> 1 **then**
    i = vn(1), j = vn(2)
    addRule(i,j,k) //$(i, j) \rightarrow k$
    assignNextIndex(k), setVisited(k)
    activeList.append(unvisitedNeigbors(k))
  else
    activeList.push_back(k)
  end if
end while

3.3 Symbolic Reconstruction

With the node ordering and reconstruction rules given by Algorithm 1, we can successively determine the positions of all joints from the current motor value $m(t)$ using closed-form expressions. Let k be the index of the current node and assume that the positions of all nodes $i < k$ have already been determined. Furthermore, assume that the reconstruction rule associated with node k is $(i, j) \rightarrow k$. Interpreting the three joint positions as the nodes of a *reconstruction triangle*, our goal is to analytically determine $\mathbf{x}_k$ from $\mathbf{x}_i$, $\mathbf{x}_j$, and the two constant edge lengths $\bar{l}_{ik}$, $\bar{l}_{jk}$ (see Fig. 3). To this end, we first scale the vector $\mathbf{x}_j - \mathbf{x}_i$ to length $\bar{l}_{ik}$, then rotate it about the known node $\mathbf{x}_i$ by an angle ϕ to obtain

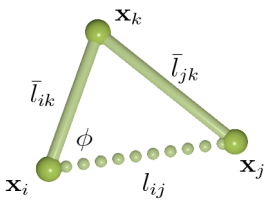


Figure 3: *Notation for reconstruction triangle.*

$$\mathbf{x}_k = \mathbf{R}(\phi)\bar{l}_{ik}\frac{\mathbf{x}_j - \mathbf{x}_i}{||\mathbf{x}_j - \mathbf{x}_i||} + \mathbf{x}_i \,, \tag{1}$$

where we use the Law of Cosines to obtain the rotation angle

$$\phi = \arccos\left(\frac{l_{ij}^2 + \bar{l}_{ik}^2 - \bar{l}_{jk}^2}{2l_{ij}\bar{l}_{ik}}\right). \tag{2}$$

Note that the 2×2 matrix $\mathbf{R}(\phi)$ either represents a counterclockwise or a clockwise rotation, depending on the orientation of the

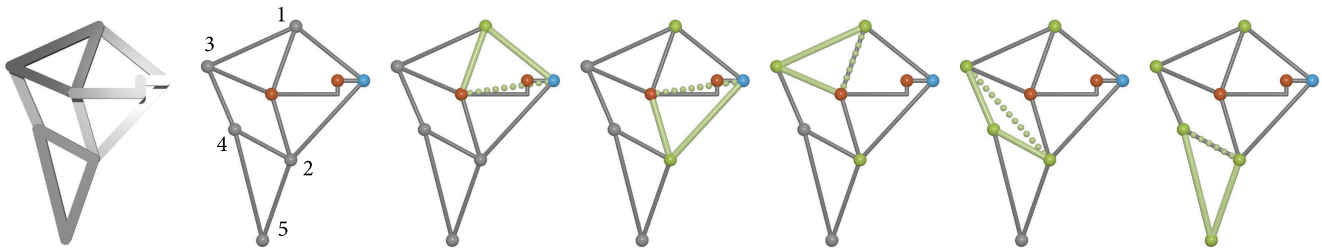
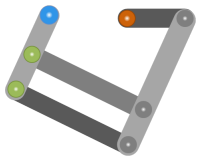


Figure 2: *Symbolic reconstruction illustrated on a leg of the Jansen linkage.* 2^{nd} from left*: starting with two fixed joints (*red*) and a joint controlled by a motor (*blue*), the ordering algorithm determines the sequence in which to compute the remaining joints (*as numbered*).* $3^{rd} - 7^{th}$*: joints (*green*) are reconstructed from two known joint positions and two edge-lengths as indicated by the green triangle.*

triangle. We choose among the two solutions by asking that the triangle orientation in the initial configuration of the linkage is preserved. It is important to note that this choice is not arbitrary, but a requirement for non-degeneracy of the linkage. If, for contrast, we assume that its motion is such that the orientation changes, the three joints must become collinear at some point in time. This, in turn, results in a bifurcation point beyond which the motion of $\mathbf{x}_k$ is no longer uniquely determined through $\mathbf{x}_i$ and $\mathbf{x}_j$. Such singularities have to be avoided at all costs and, as will be shown in Sec. 4.2, the orientation of the reconstruction triangle will play a crucial role for detecting and preventing such degenerate configurations.

Limitations The algorithm described above assumes that the input linkage can be decomposed into kinematic loops that can be processed sequentially such that only one unknown joint position has to be determined at a time. We refer to these basic building blocks as *simple kinematic loops*. But while many planar linkages can be decomposed into simple kinematic loops, this is not true for all cases.

The inset figure shows an example of a complex kinematic loop consisting of 5 links, a motor (*blue*), and a fixed joint (*red*). Given the motor value, we can determine the positions of two joints (*green*), but none of the three remaining joints can be directly determined from known quantities. It is a simple matter to detect such higher-order kinematic loops. Whenever the ordering algorithm returns an order for a given linkage, our symbolic kinematics algorithm is applicable. If the linkage exhibits a complex kinematic loop, the algorithm will not be able to further reduce the active list at some point and signal failure. In order to deal with these cases, one could design dedicated analytic reconstruction rules or use an embedded constraint-based simulation solver. As a matter of fact, however, a large range of practical mechanisms can be decomposed into basic kinematic loops, which is why we leave this extension as future work.



We have investigated the behavior of our symbolic kinematics algorithm on a diverse set of example linkages and, as we show in Sec. 5, the performance is orders of magnitudes better than a Newton-based solver. Building on this approach, we develop an interactive editing system as described next.

4 Linkage Editing

Editing a linkage means changing certain aspects of its geometry or motion while preserving others. Starting from a working linkage, the user specifies desired edits through interactive tools that issue *editing objectives* of different granularity. During editing, our system provides constant and immediate feedback through animations of the entire mechanism and motion curves that visualize the movement of selected points. In addition to meeting these objectives, we must imperatively maintain the correct functioning of the linkage at all times during the edits. To this end, we introduce *functionality constraints* that safeguard against singular configurations.

4.1 Objectives

The shape and motion of a linkage are defined by the positions of its joints and their evolution over time. For the type of linkages that we consider here—fully-constrained and driven by a single motor—the joint positions at any time are determined by the time-varying motor value and the constant initial configuration of the mechanism. Therefore, by changing the initial state of the linkage, we can control its shape and motion. Our system provides a set of tools that allow the users to intuitively change a linkage's initial state. These tools give rise to various editing objectives that we describe in the following. Examples of different use cases are given in Sec. 5.

Joint Displacements The most basic way of editing a linkage is to directly change its initial configuration by imposing displacements on individual joints. For this purpose, the user selects a given joint $\bar{\mathbf{x}}_i$ in the initial configuration and drags it to a desired location $\hat{\mathbf{x}}_i$ as illustrated in Fig. 4. The system responds to this edit by creating an energy term that encourages this operation as

$$f_e^{\text{Disp}}(\bar{\mathbf{x}}_i) = \tfrac{1}{2}(\bar{\mathbf{x}}_i - \hat{\mathbf{x}}_i)^2 \,. \tag{3}$$

Apart from editing the initial configuration, the user can also stop the motion of the mechanism at any time and specify changes directly on the current configuration. In such cases, we replace $\bar{\mathbf{x}}$ with $\mathbf{x}_i(\bar{\mathbf{x}})$ in (3). This editing mode is useful if, e.g., the position of a given joint at a single point in time is important. If the entire motion matters, the user can edit the trajectory of selected points as described next. The per-joint energy is also easily extended to higher-order editing tools such as rigid transformations of sets of joints or global scaling of the linkage. Sec. 5 and the accompanying video show examples that demonstrate these tools.

Trajectory Editing Instead of manually displacing joints, it is often more intuitive to specify a desired effect and compute the required joint displacements automatically. We support this editing mode by allowing the user to change the motion of selected *tracking points*. The user defines tracking points by specifying a location relative to a selected component. The time-varying position $\mathbf{p}(t) = \mathbf{p}(\mathbf{x}(t))$ of the point is determined from the current configuration of the linkage or, equivalently, from the initial configuration and the motor value as $\mathbf{p}(t) = \mathbf{p}(m(t), \bar{\mathbf{x}})$. In order to edit the trajectory of a given tracking point $\mathbf{p}_i$, we define an energy that

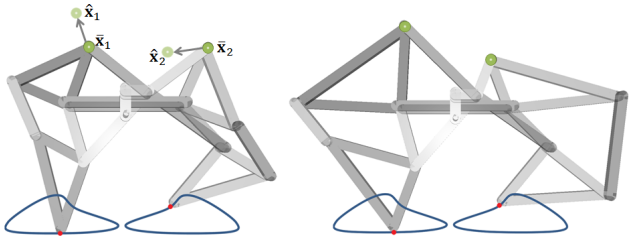


Figure 4: *Joint-displacement and trajectory objectives are used to change the shape of a Jansen linkage while preserving the motion of its feet.*

penalizes deviations from the discrete target trajectory $\hat{\mathbf{p}}_i^j$ as

$$f_e^{\text{Track}}(\bar{\mathbf{x}}) = \tfrac{1}{2}\sum_j (\mathbf{p}_i^j(\bar{\mathbf{x}}) - \hat{\mathbf{p}}_i^j)^2 \,. \tag{4}$$

We note that this formulation also allows us to demand that the motion of selected points remain unchanged. As shown in Fig. 1, this is crucial if, e.g., the motion of an end-effector needs to be tracked precisely in order to maintain correct functioning. Our system supports two ways of generating target trajectories for tracking points. A global transformation mode allows the user to translate, rotate, and scale the motion curve. In order to change the shape of motion curves, the user can drag on selected points on the curve, using a soft-selection metaphor to specify the range of points that are influenced.

Motion Envelope Editing Editing a linkage for reuse in a different context often demands that we change its motion envelope, e.g., to fit into a narrower enclosure. We define the motion envelope of a linkage as the convex hull of all of its joint positions for any point in time. A simple way of defining enclosure constraints is to impose limits on the extents of an axis-aligned bounding box for the motion envelope as shown in Fig. 5. Whenever a joint moves outside this bounding box at a given time step, we add a penalty term that pulls the joint back to the admissible region. The formulation is analogous to (3), but applies only to the time steps in which the enclosure constraints are violated.

Regularization Most of the objectives described above involve only a subset of the joints. Furthermore, there can be a multitude of joint displacements that satisfy a given editing objective equally well. In selecting a particular solution, we generally prefer those that induce the least amount of change in the linkage. We quantify the amount of change by measuring the change in distance between

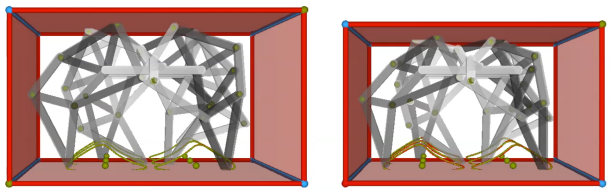


Figure 5: *Editing the motion envelope of a mechanism while preserving the motion of its end-effectors. Bounds on the envelope are specified per coordinate using an axis-aligned bounding box.*

pairs of joints on the same component as

$$f_e^{\text{Reg}}(\bar{\mathbf{x}}) = \tfrac{1}{2}\sum_{i,j} (l_{ij}(\bar{\mathbf{x}}) - \tilde{l}_{ij})^2 \,, \tag{5}$$

where l_{ij} and $\tilde{l}_{ij}$ denote the current and reference distance between joints i and j. Instead of regularizing towards the original state, we update the reference distances after each editing operation. While this regularizer is slightly more involved than a simple L_2-norm penalty on the joint displacements, it encourages shape-preservation without penalizing rotations or translations.

4.2 Avoiding Singularities

The objectives described in the previous section promote fast and intuitive linkage editing. However, without further action, these editing operations are bound to drive the mechanisms towards singularities, i.e., parameter values for which the motion becomes indefinite or grinds to halt altogether. Singularities are not merely a hypothetical peril, they arise very frequently, making editing tedious at best. In a constraint-based approach, a standard way of detecting singular configurations is by analyzing the Jacobian of the constraints, i.e., their derivatives with respect to the states of the components. As long as this matrix maintains full rank, the system stays clear of singularities. In order to prevent a linkage from moving too close to these problematic configurations, Thomaszewski et al. [2014] employ a penalty term proportional to the inverse of the smallest singular value. However, the computational costs of this approach are too high for our interactive editing application.

Fortunately, the symbolic simulation algorithm described in Sec. 3.3 provides us with a simple yet effective means to detect and prevent singular configurations. In the reconstruction rule described by (1), we established that, for non-degenerate linkages, the orientation of the reconstruction triangle described by three points involved in the rule does not change. Using the reverse of this statement, we obtain a strong predicate for singularities: the linkage undergoes a singularity whenever the three points of any reconstruction triangle become colinear at any time. This observation provides a direct recipe for preventing singularities. To this end, we quantify the distance to a singular configuration as

$$d_s(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k) = 1 - \cos(\alpha(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k))^2 \,, \tag{6}$$

where $\alpha(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$ denotes the angle formed by the two edges incident to the reconstructed joint $\mathbf{x}_k$. One possible approach would be to introduce inequality constraints for each reconstructed joint and each sample in time that prevent the distance to singularities from becoming too small. However, this would mean a drastic increase in problem size. We therefore pursue a penalty approach and define a corresponding energy as

$$f_s(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k) = -\tfrac{1}{2}\log^2(\tfrac{1}{\varepsilon} d_s(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)) \,. \tag{7}$$

We implement this term as a one-sided function, returning a zero value whenever the distance defined by (6) is larger than a threshold ε. Note that this formulation ensures continuity in the first derivative.

It is worth noting that, compared to the approach described in [Thomaszewski et al. 2014], the simplicity of our construction is owed to additional information and simplifying assumptions that we can use to our advantage. By exploiting knowledge on kinematic causality and focusing on simple kinematic loops, we are able to construct an efficient and robust way of avoiding singular configurations. Next we describe how to formulate and solve the optimization problem that integrates the various objectives and constraints.

4.3 Optimization

The objectives and constraints described above are functions of the time-varying configuration of the linkage $\mathbf{x}(t)$. However, we can only change its initial configuration $\bar{\mathbf{x}}$ directly. We therefore formulate a minimization problem where we seek to find changes $\Delta\bar{\mathbf{x}}$ to the linkage's initial state that satisfy

$$\Delta\bar{\mathbf{x}} = \arg\min_{\Delta\tilde{\mathbf{x}}} \quad \mu_e f_e(\mathbf{x}(\bar{\mathbf{x}} + \Delta\tilde{\mathbf{x}})) + \mu_s f_s(\mathbf{x}(\bar{\mathbf{x}} + \Delta\tilde{\mathbf{x}})) \,, \quad (8)$$

where the coefficients μ_e and μ_s are weights for the editing objectives and singularity penalties, respectively. In order for this energy to attain a minimum, its gradient has to vanish. This requirement leads to a system of nonlinear equations,

$$\left(\mu_e \frac{\partial f_e}{\partial \mathbf{x}} + \mu_s \frac{\partial f_s}{\partial \mathbf{x}}\right) \frac{\partial \mathbf{x}}{\partial \bar{\mathbf{x}}} = \mathbf{0} \,, \quad (9)$$

which we solve using Newton's method. An aspect that deserves further attention is the computation of the derivative $\frac{\partial \mathbf{x}}{\partial \bar{\mathbf{x}}}$, relating a change in initial state to the corresponding change in the current configuration.

Computing State Derivatives As described in Sec. 3.3, our symbolic kinematics algorithm relies on recursive computations in order to determine the state of the linkage from a given motor value and initial configuration. Consequently, we also compute derivatives of the state with respect to the initial configuration using a recursive algorithm based on the same graph structure. We focus on the algorithm for combining and propagating the derivatives of the reconstruction rules (see supplemental material for derivations), which are readily obtained analytically. While the true parameters of our system are the joint positions in the initial configuration of the linkage, notational simplicity justifies the detour of using distances between the joints as intermediate parameters. The relation between positions and lengths is obvious, as are the corresponding derivatives. For a given reconstruction rule $(i, j) \rightarrow k$, we need to compute the derivatives of Eq. 1 with respect to the joint positions $\mathbf{x}_i$ and $\mathbf{x}_j$, as well as the two distances $\bar{l}_{ik}$ and $\bar{l}_{jk}$. Only the latter depend directly on the initial configuration, but we also need to account for the fact that the two nodes $\mathbf{x}_i$ and $\mathbf{x}_j$ can potentially depend on all other nodes that have a lower index and thus the distances involved in the corresponding reconstruction rules. Using the graph structure described in Sec. 3.3, we keep track of these dependencies and compute the corresponding derivatives recursively using the chain rule as described in the supplemental material. While this scheme could be extended to compute second derivatives, we instead use the BFGS-algorithm to compute an incrementally-updated estimate of the Hessian of (8) from its gradients.

Our approach for computing state derivatives is very fast and, together with the symbolic kinematics algorithm and our formulation for preventing singularities, it constitutes the enabling technology for interactive linkage editing. Sec. 5 provides evidence for this fact in terms of performance comparisons with a state-of-the-art numerical kinematics solver.

5 Results

We have used our interactive editing system to apply various changes to the shape and motion of a diverse set of linkages. In the following, we demonstrate common use cases for our method and highlight its main features on three example linkages. But before we proceed to examples, we compare the performance of our symbolic kinematics algorithm to a reference solution for numerical kinematics.

5.1 Symbolic vs. Numerical Kinematics

In order to quantify the computational benefits of our *symbolic* kinematics algorithm, we consider the implications and performance of an alternative approach based on standard *numerical* kinematics [Laulusa and Bauchau 2008]. Following the model described in [Coros et al. 2013], we represent a linkage as a set of rigid components interconnected through constraints. The state of the linkage is given by the position and orientation of the components, whereas the parameters are the positions of the joints in the local coordinate systems of the components. For the purely kinematic case, computing the motion of the linkage can be cast as an unconstrained minimization problem, which amounts to solving a set of nonlinear equations with Newton's method.

We have investigated the behavior of our symbolic kinematics algorithm on a diverse set of example linkages and compared its performance to the numerical kinematics solver. We generally observed that the solutions produced by the two approaches agree within tight margins, but large speedups result for the symbolic kinematics solver: for the Stranbeest with three leg pairs (Fig. 1), the numerical solver needed 1.1ms on average to simulate a single step, whereas our symbolic algorithm took only 0.01ms. Repeating the experiment for a structure with 10 leg pairs, the numerical solver required 7.5ms while the symbolic approach took 0.03ms, indicating that the symbolic algorithm also scales favorably with increasing complexity. Besides forward simulation, editing a linkage also requires the derivatives of state with respect to parameters. For numerical kinematics, however, there is no closed-form relation between the state of a linkage and its parameters, which is why the derivatives have to be determined indirectly by solving a system of equations. Finally, in order to guarantee the correct functioning at all times, one has to explicitly prevent singularities during the edits. While stochastic optimization is a valid option for offline applications [Thomaszewski et al. 2014], this approach is intractable for interactive editing. In an effort to improve performance, we combined the numerical kinematics approach with singularity constraints with analytical derivatives. But even when using an industrial-grade constraint optimization package, the performance is far beyond interactive rates. This analysis shows that the performance gain of our symbolic kinematics approach is not only substantial, but a vital condition for interactive linkage editing.

5.2 Editing Examples

Function-Preserving Editing When editing an existing linkage, we often want to change the aesthetic aspects of its shape and motion while leaving its function intact. We demonstrate this use case on the Jansen linkage—the building block of the famous kinetic sculpture *Strandbeest* by artist Theo Jansen. Driven by a single actuator, this mechanism generates walking-like motions for its two end-effectors. This design has inspired myriad imitations and interpretations, and while the applications are diversely creative, the linkage itself remains largely unchanged. Indeed, editing the shape or motion is a difficult task since inadvertent tampering with the inter-joint distances is bound to end in dysfunction of the mechanism. By contrast, our system makes function-preserving edits easy and intuitive. In the first edit, the user sets up tracking objectives for the end-effectors, asking that their motion remain unchanged to maintain the characteristics of the gait. As illustrated by the edits shown in Fig. 1, this constraint still leaves ample room for changing the shape of the structure, e.g., through joint displacements. We noticed that significant edits to the rest pose can lead to parts of the linkage colliding with the ground plane. But these problems are conveniently resolved using our motion envelope tool.

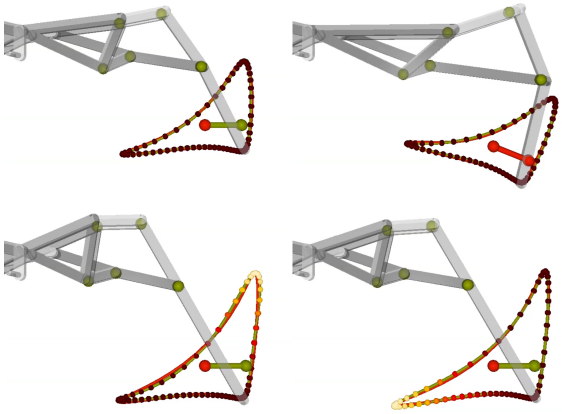


Figure 6: *Motion-curve editing demonstrated on a leg of the Klann linkage.*

In the second edit, we use the motion envelope tool to rescale the Jansen linkage while again leaving the motion of the feet intact. As can be seen in Fig. 1 (second from left), the changes to the geometry are quite significant, allowing the joints of the linkage to fit in a much narrower confinement.

Motion-Curve Editing Another application of linkage editing is to adapt the function of a mechanism by changing the motion of an end-effector. By way of illustration, we explore a range of possible end-effector motions for the Klann linkage—another famous mechanism. Navigating the space of feasible motions for a given linkage is a difficult task without assistance from an interactive tool. Our motion-curve tool provides rigid-transformation and soft-selection editing metaphors, allowing the user to deform the target trajectory in an intuitive way while the optimization works in the background to comply with these requests. Fig. 6 shows a selection of different end-effector motions, a more extensive set is provided in the video. Clearly, the structure of a given linkage will only admit a subset of the requested edits, but the quick turn-around time of our system allows the user to navigate around such road blocks and discover alternate motions.

Aesthetic Retargeting Conceiving a complex linkage is beyond the capabilities of average users, but existing designs often inspire new interpretations or reuse in a different context. Starting from an 11-bar linkage whose shape and motion suggest the hind leg of a bull-like creature, we edit both its geometry and motion in order to evoke the characteristics of a rhinoceros. Targeting a stout and sturdy appearance, we first shorten and broaden the hind leg using joint-displacements while tracking the motion of a marker on the foot. The hind leg is then duplicated and re-edited to serve as front leg using joint-displacements and spline-based geometry edits. We then use the motion-curve tool to fine-tune the trajectory of the foot. The result is a personalized physical character with appealing shape and motion.

Finally, it is worth noting that, for all examples that we have investigated, the system remains robust and responsive even when trying to force the linkages into singularities.

5.3 Limitations & Future Work

We presented an interactive system for intuitive editing of planar linkages. In the future, we plan to extend our system to accommodate other mechanical components such as gear trains or cam shafts.

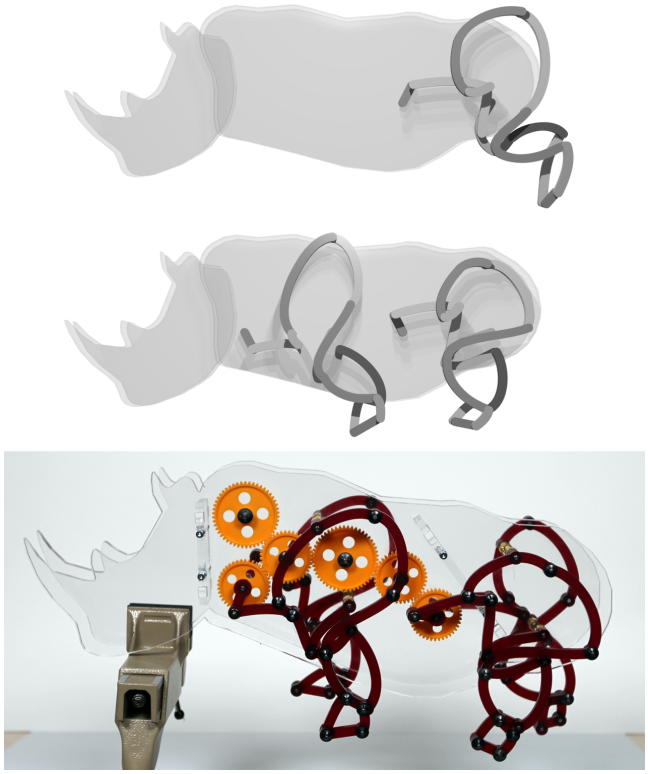


Figure 7: *Rhino example.* From top to bottom*: input mechanism, result of user edits, and physically-fabricated prototype.*

Our symbolic simulation algorithm can currently only handle simple kinematic loops, i.e., linkages for which the position of any free joint can be determined in isolation from the position of two neighboring joints. While using an embedded constraint-based solver is an option, we would like to investigate geometric reconstruction rules for complex kinematic loops as well as corresponding singularity conditions. As another interesting direction, we would like to extend our system to detect and handle collisions among moving parts. While collisions can be resolved for planar mechanisms by arranging components in different depth layers, this approach does not apply when editing spatial mechanisms, which is another interesting direction for future work.

Acknowledgments

We thank the anonymous reviewers for their helpful comments; Jan Wezel for fabrication and assembly of our Rhino example; Maurizio Nitti and Nobuyuki Umetani for model design; Romain Prévost and Oliver Wang for video assistance.

References

AUTODESK. 2015. *Autodesk MeshMixer*. Available at `http://www.123dapp.com/meshmixer`.

BÄCHER, M., WHITING, E., BICKEL, B., AND SORKINE-HORNUNG, O. 2014. Spin-it: Optimizing moment of inertia for spinnable objects. *ACM Trans. Graph. (Proc. SIGGRAPH) 33*, 4.

BICKEL, B., BÄCHER, M., OTADUY, M. A., LEE, H. R., PFISTER, H., GROSS, M., AND MATUSIK, W. 2010. Design and

fabrication of materials with desired deformation behavior. *Proc. of ACM SIGGRAPH '10*.

BICKEL, B., KAUFMANN, P., SKOURAS, M., THOMASZEWSKI, B., BRADLEY, D., BEELER, T., JACKSON, P., MARSCHNER, S., MATUSIK, W., AND GROSS, M. 2012. Physical face cloning. In *Proc. of ACM SIGGRAPH '12*.

BOKELOH, M., WAND, M., KOLTUN, V., AND SEIDEL, H.-P. 2011. Pattern-aware shape deformation using sliding dockers. In *Proc. of ACM SIGGRAPH '11*.

BOKELOH, M., WAND, M., SEIDEL, H.-P., AND KOLTUN, V. 2012. An algebraic model for parameterized shape editing.

BURMESTER, L. 1888. *Lehrbuch der Kinematik*. Arthur Felix, Leipzig.

CEYLAN, D., LI, W., MITRA, N. J., AGRAWALA, M., AND PAULY, M. 2013. Designing and fabricating mechanical automata from mocap sequences. In *Proc. of ACM SIGGRAPH Asia '13*.

COROS, S., THOMASZEWSKI, B., NORIS, G., SUEDA, S., FORBERG, M., SUMNER, R. W., MATUSIK, W., AND BICKEL, B. 2013. Computational design of mechanical characters. In *Proc. of ACM SIGGRAPH '13*.

DONG, Y., WANG, J., PELLACINI, F., TONG, X., AND GUO, B. 2010. Fabricating spatially-varying subsurface scattering. In *Proc. of ACM SIGGRAPH '10*.

DUMAS, J., HERGEL, J., AND LEFEBVRE, S. 2014. Bridging the gap: Automated steady scaffoldings for 3d printing. In *Proc. of ACM SIGGRAPH '14*.

ERDMAN, A. G., SANDOR, G. N., AND KOTA, S. 2001. *Mechanism Design: Analysis and Synthesis*. Prentice-Hall, Englewood Cliffs, NJ. Vol. 1, No 4.

FREUDENSTEIN, F. 1954. *Design of Four-link Mechanisms*. Ph. D. Thesis, Columbia University, USA.

GAL, R., SORKINE, O., MITRA, N. J., AND COHEN-OR, D. 2009. iwires: An analyze-and-edit approach to shape manipulation. In *Proc. of ACM SIGGRAPH '09*.

HASAN, M., FUCHS, M., MATUSIK, W., PFISTER, H., AND RUSINKIEWICZ, S. 2010. Physical reproduction of materials with specified subsurface scattering. In *Proc. of ACM SIGGRAPH '10*.

KAUFMAN, R. E., AND MAURER, W. G. 1971. Interactive linkage synthesis on a small computer. In *Proceedings of the 1971 26th Annual Conference*, ACM '71, 376–387.

KECSKEMÉTHY, A., KRUPP, T., AND HILLER, M. 1997. Symbolic processing of multiloop mechanism dynamics using closed-form kinematics solutions. *Multibody System Dynamics 1*, 1, 23–45.

KOO, B., LI, W., YAO, J., AGRAWALA, M., AND MITRA, N. J. 2014. Creating works-like prototypes of mechanical objects. *ACM Transactions on Graphics (Special issue of SIGGRAPH Asia 2014)*.

LAU, M., OHGAWARA, A., MITANI, J., AND IGARASHI, T. 2011. Converting 3D furniture models to fabricatable parts and connectors. In *Proc. of ACM SIGGRAPH '11*.

LAULUSA, A., AND BAUCHAU, O. A. 2008. Review of Classical Approaches for Constraint Enforcement in Multibody Systems. *Journal of Computational and Nonlinear Dynamics 3*, 1.

LU, L., SHARF, A., ZHAO, H., WEI, Y., FAN, Q., CHEN, X., SAVOYE, Y., TU, C., COHEN-OR, D., AND CHEN, B. 2014. Build-to-last: Strength to weight 3d printed objects. In *Proc. of ACM SIGGRAPH '14*.

MEGARO, V., THOMASZEWSKI, B., GAUGE, D., GRINSPUN, E., COROS, S., AND GROSS, M. H. 2014. Chacra: An interactive design system for rapid character crafting. In *Proc. of Symp. on Computer Animation '14*.

MITRA, N. J., YANG, Y.-L., YAN, D.-M., LI, W., AND AGRAWALA, M. 2010. Illustrating how mechanical assemblies work. In *Proc. of ACM SIGGRAPH '10*.

MYSZKA, D. H., MURRAY, A. P., AND WAMPLER, C. W. 2013. Computing the branches, singularity trace, and critical points of single degree-of-freedom, closed-loop linkages. *Journal of Mechanisms and Robotics 6*, 1.

PERRY, R. N., AND FRISKEN, S. F. 2001. Kizamu: A system for sculpting digital characters. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, ACM, New York, NY, USA, SIGGRAPH '01, 47–56.

PRÉVOST, R., WHITING, E., LEFEBVRE, S., AND SORKINE-HORNUNG, O. 2013. Make It Stand: Balancing shapes for 3D fabrication. *ACM Trans. Graph. (Proc. SIGGRAPH) 32*, 4.

SCHULZ, A., SHAMIR, A., LEVIN, D. I. W., SITTHI-AMORN, P., AND MATUSIK, W. 2014. Design and fabrication by example. *ACM Trans. Graph. 33*, 4 (July), 62:1–62:11.

SKOURAS, M., THOMASZEWSKI, B., BICKEL, B., AND GROSS, M. 2012. Computational design of rubber balloons. In *Proc. of Eurographics '12*.

SKOURAS, M., THOMASZEWSKI, B., COROS, S., BICKEL, B., AND GROSS, M. 2013. Computational design of actuated deformable characters. In *Proc. of ACM SIGGRAPH '13*.

SORKINE, O., COHEN-OR, D., LIPMAN, Y., ALEXA, M., RÖSSL, C., AND SEIDEL, H.-P. 2004. Laplacian surface editing. In *Proceedings of the EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing*, ACM Press, 179–188.

STAVA, O., VANEK, J., BENES, B., CARR, N., AND MĚCH, R. 2012. Stress relief: improving structural strength of 3d printable objects. In *Proc. of ACM SIGGRAPH '12*.

THOMASZEWSKI, B., COROS, S., GAUGE, D., MEGARO, V., GRINSPUN, E., AND GROSS, M. 2014. Computational design of linkage-based characters. In *Proc. of ACM SIGGRAPH '14*.

UCHIDA, T., AND MCPHEE, J. 2012. Using grbner bases to generate efficient kinematic solutions for the dynamic simulation of multi-loop mechanisms. *Mechanism and Machine Theory 52*, 0, 144 – 157.

UMETANI, N., IGARASHI, T., AND MITRA, N. J. 2012. Guided exploration of physically valid shapes for furniture design. In *Proc. of ACM SIGGRAPH '12*.

WEYRICH, T., PEERS, P., MATUSIK, W., AND RUSINKIEWICZ, S. 2009. Fabricating microgeometry for custom surface reflectance. In *Proc. of ACM SIGGRAPH '09*.

ZHOU, Q., PANETTA, J., AND ZORIN, D. 2013. Worst-case structural analysis. In *Proc. of ACM SIGGRAPH '13*.

ZHU, L., XU, W., SNYDER, J., LIU, Y., WANG, G., AND GUO, B. 2012. Motion-guided mechanical toy modeling. In *Proc. of ACM SIGGRAPH Asia '12*.