

# Functionality-aware Retargeting of Mechanisms to 3D Shapes

RAN ZHANG, IST Austria  
THOMAS AUZINGER, IST Austria  
DUYGU CEYLAN, Adobe Research  
WILMOT LI, Adobe Research  
BERND BICKEL, IST Austria

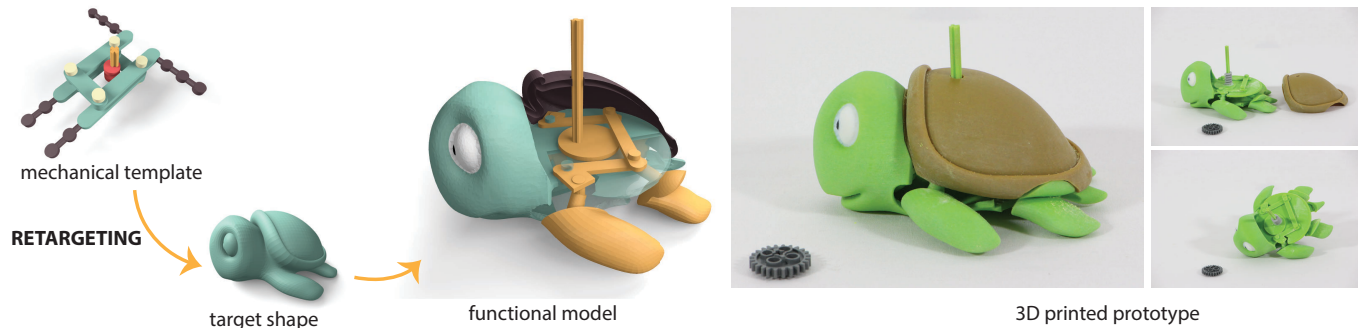


Fig. 1. Our interactive design system allows users to retarget a given mechanical template (top left) to an input shape (bottom left). Our optimization-in-the-loop approach generates a functional model (center) that can be 3D printed (right).

We present an interactive design system to create functional mechanical objects. Our computational approach allows novice users to retarget an existing mechanical template to a user-specified input shape. Our proposed representation for a mechanical template encodes a parameterized mechanism, mechanical constraints that ensure a physically valid configuration, spatial relationships of mechanical parts to the user-provided shape, and functional constraints that specify an intended functionality. We provide an intuitive interface and optimization-in-the-loop approach for finding a valid configuration of the mechanism and the shape to ensure that higher-level functional goals are met. Our algorithm interactively optimizes the mechanism while the user manipulates the placement of mechanical components and the shape. Our system allows users to efficiently explore various design choices and to synthesize customized mechanical objects that can be fabricated with rapid prototyping technologies. We demonstrate the efficacy of our approach by retargeting various mechanical templates to different shapes and fabricating the resulting functional mechanical objects.

CCS Concepts: • **Computing methodologies** → **Graphics systems and interfaces**; *Shape modeling*; • **Applied computing** → *Computer-aided manufacturing*;

Additional Key Words and Phrases: Fabrication, Functional structure, Mechanical structure

## ACM Reference format:

Ran Zhang, Thomas Auzinger, Duygu Ceylan, Wilmot Li, and Bernd Bickel. 2017. Functionality-aware Retargeting of Mechanisms to 3D Shapes. *ACM Trans. Graph.* 36, 4, Article 81 (July 2017), 13 pages.  
DOI: <http://dx.doi.org/10.1145/3072959.3073710>

This project has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 642841 and under the European Research Council (ERC) grant agreement No 715767.

© 2017 ACM. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *ACM Transactions on Graphics*, <https://doi.org/http://dx.doi.org/10.1145/3072959.3073710>.

## 1 INTRODUCTION

The increasing accessibility of rapid manufacturing devices and 3D printing services has made it possible for more and more users to fabricate a variety of functional objects. In recent years, many compelling examples have emerged from the maker community, including animated characters, mechanical automata, and simple robots. However, despite steady advances in computer-aided digital modeling tools, designing such functional objects is still very challenging and typically reserved for experts. Developing design tools that facilitate this task and make it accessible to a wider audience is an open research challenge at the intersection of computational fabrication and computer graphics.

The goal of our work is to address this problem for functional objects that are based on simple mechanisms, such as an assembly of gears and gear trains, cams, or linkages. Functional mechanical objects can be defined by a few high-level properties: *form*, which represents the shape and appearance of the design; *mechanical architecture*, which describes the configuration of mechanical parts; and finally, the *function* resulting from the combination of form and mechanical architecture. A challenge in designing such objects with conventional tools is that the user must develop the form and mechanism *concurrently*. This requires expertise both in shape modeling and mechanical engineering. For example, to design a given functional object, a typical workflow usually requires modeling an appropriate mechanism, shaping the form of the object, connecting mechanism and shape, and eventually adapting both the shape and the parameters of the mechanism to obtain a final working prototype. We envision a system that does not require modeling all properties from scratch, but instead enables the reuse of an existing mechanical architecture by retargeting it to a user-provided shape

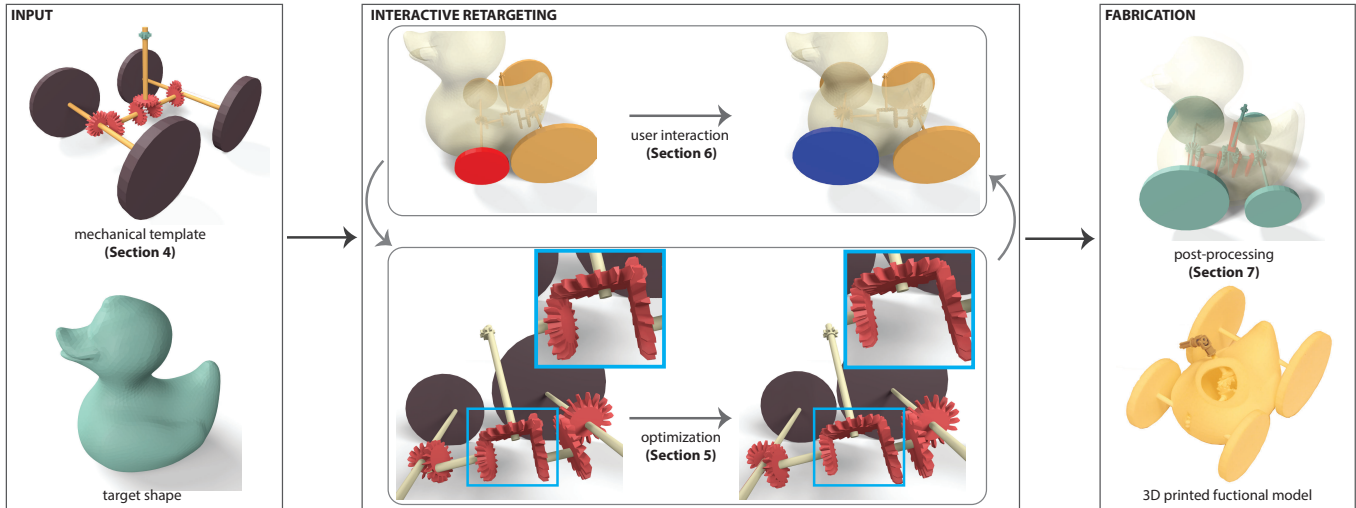


Fig. 2. Given a fully functional mechanical template and a target shape, we present a retargeting system which provides interactive tools to help retarget the functionality of the template to the target shape. As the user specifies placement options of the mechanism inside the target shape or edits the shape, the underlying optimization framework automatically computes a valid and functional configuration of the mechanical template that best matches the user preferences. Once the user is satisfied with the retargeting output, our system computes support structures necessary for 3D fabrication.

such that the resulting object matches the desired aesthetic and functional properties.

Our system is motivated by the observation that for many functional objects, the mechanical architecture remains the same while the form varies – e.g., in toy design where the same mechanism is reused for different versions of a given toy, in industrial design where the appearance/styling of an object may vary more often than the underlying mechanisms, and for prototyping, where exploration of a design space involves creating several variations of a given idea. In all cases, the different variations may require some modifications to the design parameters (e.g., due to geometric constraints) but probably not a completely different mechanical architecture.

In this paper we present an interactive retargeting system that enables users to adapt a given mechanical architecture to a target 3D shape in order to produce a functional mechanical object (see Figure 1). A central design element of our approach is a decomposition of the features into *form*, *mechanical architecture*, and *relationships* of form, mechanical architecture, and function. The mechanical architecture and their degrees of freedom, as well as rules that specify relationships, are encoded in a *mechanical template*. A core contribution of the paper is an interactive computational approach that, in combination with a tailored user interface, enables users to modify the placement of mechanical components and the shape of the object, while an underlying optimization procedure updates the overall design based on the relationship constraints to preserve the desired functionality. Our optimization model is based on the concept of differential manipulation [Gleicher and Witkin 1991], which supports constrained editing of geometric objects.

We have implemented a prototype design tool and validated our approach by creating several mechanical objects and showing their functionality as 3D printed models. Our system was tested by several users without any background knowledge in mechanical design.

As shown in the results section, all of them were able to create compelling functional models using our tools.

## 2 RELATED WORK

The advent of accessible fabrication methods has led to recent work on various fabrication-aware modeling approaches both for static and functional objects.

### 2.1 Design for Fabrication

3D printing is by far one of the most prevailing fabrication method accessible to both professionals and novices. Therefore, we have seen an increasing amount of research effort over the recent years focusing on optimizing 3D designs for 3D printing. Previous work has presented methods for decomposing large objects into smaller parts [Luo et al. 2012; Song et al. 2015a], identifying the parts of the object that may be exposed to high stress [Stava et al. 2012; Zhou et al. 2013], generating robust scaffolding [Dumas et al. 2014], and optimizing objects to reduce weight [2014] and fabrication time [Beyer et al. 2015].

As the recent advances in digital fabrication have paved the road to customization and personalization, a variety of approaches have proposed computational tools for designing objects with custom features, including desired appearance [Dong et al. 2010] and deformation properties [Bickel et al. 2010; Pérez et al. 2015], the ability to stand [Prévost et al. 2013], spin [Bächer et al. 2014], fly [Umetani et al. 2014], and swing [Zhao et al. 2016]. Umetani et al. [2012] present an interactive framework for stable furniture design. Skouras et al. [2014] focus on design of inflatable structures, and Zimmer et al. [2014] extend the scope of *Zometool* modeling to freeform, disk-topology surfaces. Recently, computational methods for designing connectors [Koyama et al. 2015], twisty joints and puzzles [Sun and Zheng 2015], scissor structures [Zhang et al. 2016; Zheng et al.

2016], and multicopters [Du et al. 2016] have been presented. While these approaches demonstrate compelling results, they all focus on mainly static objects. Our goal, in contrast, is to ease creation of mechanical objects with desired functional behavior.

## 2.2 Mechanism Design

Mechanism design is traditionally divided into two stages. While the *conceptual design* stage focuses on identifying the type of mechanical parts capable of realizing a desired motion [Chiu and Sridhar 1999; Han and Lee 2006; Roy et al. 2001], the subsequent *dimensional synthesis* stage determines the configuration parameters and the layout of these mechanical parts [Anantha et al. 1996; Haller et al. 2009]. We have seen increasing efforts in the graphics community to automate both of these stages to make the mechanism design process accessible to novice users. Some of these approaches have focused on automatically realizing user-specified motion requirements by utilizing a pre-defined set of mechanism types [Ceylan et al. 2013; Coros et al. 2013; Megaro et al. 2014; Song et al. 2015b; Zhu et al. 2012]. Other approaches have presented interactive systems that enable easy creation of mechanisms by exploring a template mechanism library [Bharaj et al. 2015; Kim et al. 2016; Koo et al. 2014] while configuration parameters of the mechanisms are automatically optimized to ensure physically and mechanically valid designs. More recently, Ureta et al [2016] present an interactive system that realizes physically valid joints for an input geometry and user specified kinematic hierarchy. This body of work focuses on a different fundamental design paradigm than our method. Given motion or geometry requirements by a user, these approaches aim at creating a design from scratch by assembling pre-defined mechanical blocks. On the other hand, we follow a *design by example* strategy where our goal is to explore the design space of mechanisms starting from working examples.

## 2.3 Design by Example

Creating designs based on examples is a paradigm that has been extensively studied in the graphics community to tackle challenging content creation problems. Researchers have proposed various structure-aware editing algorithms to synthesize variations of a given 3D model while preserving structural properties [Mitra et al. 2013a] and data-driven approaches that focus on deriving generative models (e.g. procedural grammars) from a set of exemplars [Xu et al. 2016a]. In the context of fabrication, Schulz et al. [2014] convert a set of expert-created designs to parameterized building blocks that can be used in an interactive system. Xie et al. [2015] present a shape editing tool that provides feedback based on finite element structural analysis. Starting from a working linkage-based character where each link is driven by a motor, Thomaszewski et al. [2014] leverage user guidance to explore different topologies to reduce the number of motors. More recently, Xu et al. [2016b] presents an interactive system to digitize a mechanisms from a set of input images. More closely related to our approach is the work of Bächer et al. [2015] and Megaro et al. [2015], which provides tools to edit both the shape and motion properties of a working mechanical assembly. While our tool supports similar edits to a working design, a key distinguishing feature of our method is that it enables *retargeting* of a working mechanism to a completely new 3D shape. To this

end, our system preserves the desired mechanical and functional properties of the original design and facilitates the creation of many functional models starting from only a single working design.

## 3 OVERVIEW

The primary input to our system is a *mechanical template*  $\mathcal{T}_M$  that describes both the mechanical architecture (i.e., the set of mechanical components and how they are connected) and functional properties (e.g., a mechanical car driving in a straight line without slipping) of a fully functional design. Given  $\mathcal{T}_M$ , the user provides a *target shape*  $\mathcal{S}$ , a closed watertight surface mesh that should be turned into a working object with the mechanical functionality of  $\mathcal{T}_M$ .

Our system provides interactive tools that help the user retarget  $\mathcal{T}_M$  to fit  $\mathcal{S}$  as well as possible, while preserving the relevant functional properties of the template. The final output is a working design that can be fabricated with additive manufacturing techniques.

### 3.1 Design Considerations

We designed our interactive system to address several specific challenges related to the task of mechanical retargeting.

**3.1.1 Correspondence ambiguity.** In general, automatically inferring how a mechanical template  $\mathcal{T}_M$  should align to the user-specified target shape  $\mathcal{S}$  is very difficult, since there are typically no obvious correspondences between the configuration of mechanical parts in  $\mathcal{T}_M$  and the geometry of  $\mathcal{S}$ . Even if  $\mathcal{T}_M$  is part of a complete design that has an associated 3D shape, the geometry of that shape will typically be sufficiently different from  $\mathcal{S}$  to make correspondences ambiguous. For example, imagine retargeting a driving mechanism associated with a car shape to a target duck shape (see Figure 2). Our system sidesteps this issue by providing an interface that helps users directly establish the alignment between  $\mathcal{T}_M$  and  $\mathcal{S}$ .

**3.1.2 Non-convexity.** Due to the complex interplay between *function* (formalized as constraints on the mechanical template parameters) and *form* (given by the potentially complex geometry of the target shape) the space of valid retargeting solutions is highly non-linear and non-convex making the problem of exploring the design space extremely challenging. Our system supports several user interactions that facilitate such design exploration.

**3.1.3 No “optimal” result.** Not only is the design space hard to explore, the notion of an optimal retargeting output is also not well-defined because the desired result is often subject to specific user preferences that are hard to formalize and quantify. In this sense, our system defers the final judgment to the user instead of automatically generating results that are likely to be *sub-optimal* from the user’s perspective.

### 3.2 Our System

In light of these challenges, we propose an interactive system that balances (i) fully automatic computational methods that ensure physical validity and preservation of functionality and (ii) intuitive user interactions to guide the retargeting process. More specifically, given  $\mathcal{T}_M$  and  $\mathcal{S}$ , we allow the user to directly manipulate both the mechanical components and the target shape geometry (Section 6).

In response to these edits, the system automatically optimizes the mechanism as a whole to ensure that it remains fully functional and physically valid (Section 5). This workflow allows the user to provide high-level guidance for how the template should be retargeted, while the automated algorithms make the necessary low-level adjustments to ensure that the overall design still works. Once the user is happy with the re-targeted result, the system generates support structures and component geometry to prepare the model for fabrication (Section 7).

In addition to interactively guiding the retargeting process, our system potentially requires the following user input. First, in order to robustly retarget specific functionalities of a template to certain parts of a target shape (e.g. a WIND-UP template which moves different limbs of a character), we require the user to segment the target shape in a pre-processing stage to match the topology of the template. Second, to ensure validity of the fabricated results, the user provides a per-component minimum size (e.g. minimum gear teeth count), a minimum clearance value between different components, and a shell thickness value based on the specific choice of the fabrication tool. The shell thickness parameter is used to convert the target surface mesh into a fabrication-ready solid shell by offsetting (see Section 7).

We summarize the different components of our system in Figure 2. We now first describe how we represent a mechanical template followed by a detailed description of the different components of our system.

#### 4 MECHANISM DESCRIPTION

In order to simplify the description of a mechanical template, we exploit several properties that can be found in a large class of mechanical assemblies:

**Reducibility:** We assume that a mechanical template  $\mathcal{T}_M$  consists of  $N_C$  ‘atomic’ components  $C$  (e.g., the individual gears, etc.).

**Rigidity:** Each of the components is assumed to be an undeformable, rigid object.

**Pairwise contact:** The interactions between the components is determined by their pairwise influences (e.g., an assembly of gears can be described by the pairwise contacts among them).

These assumptions allow us to represent  $\mathcal{T}_M$  as a graph, whose vertices are the components  $C$  of  $\mathcal{T}_M$  and whose edges indicate mechanical relationships between (ordered) pairs of components (see Figure 3). The actual nature of such a relationship (e.g., two gears are touching) is described by a *connection type*  $\tau_{\text{conn}}$  that is assigned to each edge. This representation is very similar to the *interaction graph* defined by Mitra et al. [2013b].

The components are categorized into *component types*  $\tau_{\text{comp}}$  (e.g., gears, linkages) to allow for a more structured description. Each component  $C_i$  is equipped with a set of continuous parameters  $\mathbf{p}_i$  that will be used to describe its geometric and functional properties. All components have a position  $\mathbf{x} \in \mathbb{R}^3$  and orientation  $\mathbf{d} \in \mathbb{R}^3 \setminus \mathbf{0}$ . Certain component types have additional geometric parameters; e.g., disk-shaped components, such as gears and wheels, possess a

radius  $r \in \mathbb{R}$ , while axles are assigned a length  $l \in \mathbb{R}$ . We denote the (closed) 3D shape of a component  $C_i$  with parameters  $\mathbf{p}_i$  as  $\mathcal{G}_{C_i}$ .

Possible connection types between a pair of components include **Fix** which enforces a fixed spatial relationship (e.g., a wheel placed at the end of an axle), **SLIDE** which allows a component to slide along the axis of another component (e.g. a gear to be placed along an axle), and **TOUCH** which ensures the surface of two components touch each other without colliding (e.g. that the teeth of two gears touch in a compatible way).

In order to support certain relationships between a mechanical template and its environment, we represent the environment as an additional component and define edges to denote these relationships. A mechanical car template, for example, requires all its wheels to be in contact with the ground plane, represented as a **TOUCH** connection type. Certain fixation relations between the components of the mechanical template and a possible target shape (e.g. an axle should be fixed to the geometry) are encoded similarly (Section 7 provides a detailed discussion).

Given such a decomposition, the parameters  $\mathbf{p}$  of a mechanical template are the aggregation of all parameters  $\mathbf{p}_i$  of its components  $C_i$ , i.e.,  $\mathbf{p} = (\mathbf{p}_1, \dots, \mathbf{p}_{N_C})$ . Moreover, associated with each connection type are a set of constraints that need to be satisfied to ensure a working mechanical template. We now describe the constraints supported by our system in detail and refer to the supplementary material for a complete description of an example mechanical template along with its parameters and associated constraints.

##### 4.1 Constraints

The functionality of a mechanical template is determined by the interactions among the individual components as well as the interactions between the components and the target shape. We identify three kinds of constraints to formalize these interactions: (i) low-level *mechanical constraints* ensure the validity of the mechanism; (ii) high-level *functional constraints* specify the intended functionality of the whole mechanism; and (iii) *spatial constraints* define the relationship between the mechanical components and the target shape. While all these constraints are represented as part of the mechanical template, spatial constraints can also be augmented during the interactive retargeting process with user-specified placement

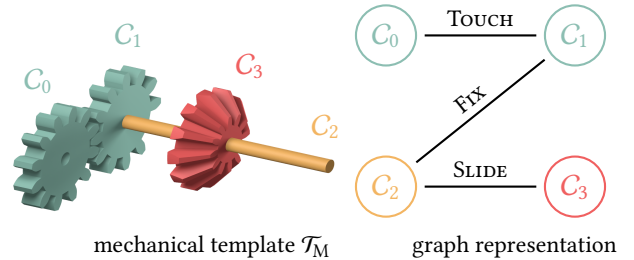


Fig. 3. We demonstrate a simple mechanical template  $\mathcal{T}_M$  (composed of two spur gears, one bevel gear, and an axle) and the corresponding graph representation which shows the pairwise connection types between the components.



preferences for certain components. We now provide a discussion of each type of constraint.

**4.1.1 Mechanical Constraints.** These constraints ensure a physically valid mechanical template. We distinguish between unary per-component and binary pairwise constraints which are either equality or inequality constraints:

$$\begin{aligned} c_{\text{mech}}(\mathbf{p}_i, \tau_{\text{comp}}(C_i)) &\geq 0 \quad \forall C_i \in \mathcal{C} \\ c_{\text{mech}}(\mathbf{p}_i, \mathbf{p}_j, \tau_{\text{conn}}(C_i, C_j)) &\geq 0 \quad \forall C_i, C_j \in \mathcal{C}. \end{aligned} \quad (1)$$

The former comprises validity constraints on component parameters and depend on the component type  $\tau_{\text{comp}}(C_i)$ ; e.g., the radii  $r$  for gears and wheels should be positive ( $r > 0$ ), and gears should have a sufficient number of teeth ( $n > 4$ ).

The interactions between different components are mediated by pairwise constraints, which depend on  $\tau_{\text{conn}}(C_i, C_j)$ , the *connection type* for components  $C_i$  and  $C_j$ . The most common among these are positional constraints that determine the spatial arrangement of parts. We also support orientation and co-planarity constraints (see Figure 4).

To ensure a fully working mechanism, it is furthermore necessary to prohibit unintended influences between components. Since we assume mechanical assemblies, this requirement can be fulfilled by precluding physical contact between parts that should not affect each other. Thus, we enforce that pairs of components that are not assigned a connection type are kept collision free. In this sense, we have the COLLISION constraint act as a default.

**4.1.2 Functional Constraints.** While mechanical constraints ensure the validity of the mechanical assembly, our system also supports *functional constraints* that encode the intended functionality of the mechanism. Unlike recent works on functionality analysis from 3D geometry which mostly focus on *static* interactions [Hu et al. 2016], we define functionality based on dynamic interactions between different mechanical parts or between a mechanism and its environment. Such desired functionality can be formulated as pairwise geometric relationships between parts, which are then enforced during optimization. Pairwise functional constraints can be combined to produce more global requirements. An example is that each wheel in a 4WD mechanism has the same linear velocity to avoid slipping (see Figure 5).

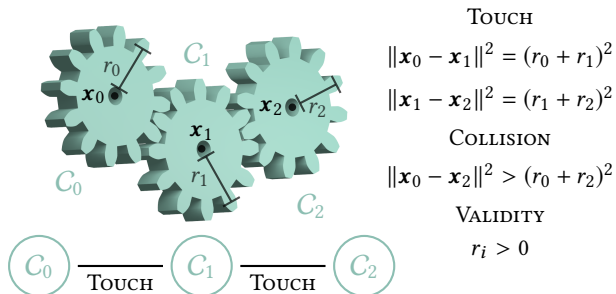


Fig. 4. Given a simple mechanical template  $\mathcal{T}_M$  composed of three gears, we provide examples of unary and binary mechanical constraints that ensure the validity of  $\mathcal{T}_M$ .

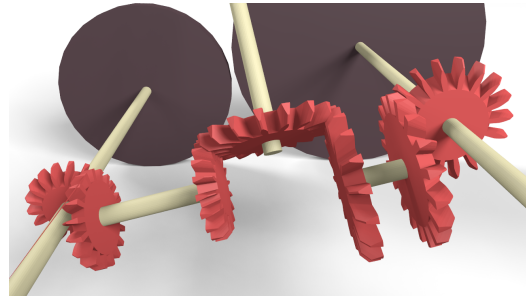


Fig. 5. For a mechanical car model, the functional constraints ensure, for example, that the wheels are parallel and have the same linear velocity. Thus, gear sizes are optimized by our system to provide suitable transmission ratios for differently sized wheels.

The formal description of functional constraints is similar to the mechanical constraints in Equation 1:

$$\begin{aligned} c_{\text{func}}(\mathbf{p}_i, \tau_{\text{comp}}(C_i)) &\geq 0 \quad \forall C_i \in \mathcal{C} \\ c_{\text{func}}(\mathbf{p}_i, \mathbf{p}_j, \tau_{\text{conn}}(C_i, C_j)) &\geq 0 \quad \forall C_i, C_j \in \mathcal{C}. \end{aligned} \quad (2)$$

Functionality also depends on the correct interaction between the mechanism and its environment. Such constraints are integrated into our system by describing these interactions with geometric proxies that represent the environment. All mechanical and functional constraints are part of the mechanical template  $\mathcal{T}_M$ .

**4.1.3 Spatial Constraints.** These constraints ensure the functionality of the mechanism. To ensure the *geometric* validity of the mechanism with respect to the user-supplied target shape  $\mathcal{S}$ , we provide two different types of spatial constraints: (i) spatial alignment preferences; and (ii) the containment of interior parts by the target shape  $\mathcal{S}$ . While the former are supplied by the user through interaction (e.g., specifying the desired position of a wheel with respect to the target shape), the latter is encoded in the mechanical template; e.g., apart from the wheels of a car (and their axles), all other components should be inside the target shape (see Figure 6).

Formally, user preferences for the spatial alignment of a part  $C_i$  are given by user-specified values  $\mathbf{p}_{\text{user},i}$  for a subset  $\tilde{\mathcal{P}}_i$  of its

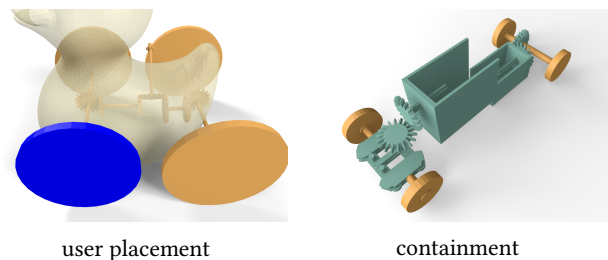


Fig. 6. Spatial constraints are defined either interactively by the user – to specify certain placement preferences for individual components (e.g. the user can specify a preferred location for the blue wheel with respect to the duck shape) – or are part of the mechanical template to ensure certain components (the green components in the car template) stay inside the target geometry.

parameters  $\mathbf{p}_i$ . The corresponding constraint is given by

$$\|\tilde{\mathbf{p}}_i - \mathbf{p}_{\text{user},i}\| = 0. \quad (3)$$

The containment constraints demand that the distance  $d$  between the 3D shape of a component  $\mathcal{G}_{C_i}$  and the surface of the target shape  $\mathcal{S}$  is greater than the sum of the shell thickness  $d_O$  and the minimal gap size  $d_{\text{gap}}$ . Since  $d_{\text{gap}}$  depends on the specific fabrication method used, it is provided by the user. In particular, since we use signed distances that are negative in the interior of  $\mathcal{S}$ , we require

$$d(\mathcal{G}_{C_i}, \mathcal{S}) < -d_O - d_{\text{gap}} \quad (4)$$

for all components that are specified to lie inside the final shape.

## 5 OPTIMIZATION

At each stage of our interactive re-targeting process, our system ensures the mechanical template stays physically valid and functional by maintaining all mechanical and functional constraints while enforcing spatial constraints where possible. This is achieved by solving a corresponding optimization problem, which we frame as an energy minimization task (see Section 5.1). As a solution strategy, we propose a form of *Differential Manipulation* [Gleicher and Witkin 1991], which has been successfully employed for constraint-based vector graphics editing [Bernstein and Li 2015] (see Section 5.2).

### 5.1 Energy Formulation

Our optimization problem is formulated based on the various constraints that we aim to enforce. As listed in Section 4.1, these are the unary and binary mechanical and functional constraints (see Equations 1 and 2) as well as the spatial constraints for component containment (see Equation 4) and user preferences (see Equation 3) for various component placement options. We aggregate all the *mechanical and functional* constraints into a set of vector-valued equality and inequality constraints

$$\begin{aligned} \mathbf{c}_{\text{eq}}(\mathbf{p}) &= \mathbf{0} \\ \mathbf{c}_{\text{ineq}}(\mathbf{p}) &\geq \mathbf{0}. \end{aligned} \quad (5)$$

We enforce the remaining constraints by minimizing the following energy function:

$$\begin{aligned} E(\mathbf{p}, \mathcal{S}, \mathbf{p}_{\text{user}}) &= \sum_{i: \mathbf{p}_{\text{user},i} \text{ defined}} \|\tilde{\mathbf{p}}_i - \mathbf{p}_{\text{user},i}\|^2 \\ &+ \lambda \sum_{i: C_i \text{ inside}} \left[ d(\mathbf{p}_i, \mathcal{S}, \tau_{\text{comp}}(C_i)) + d_O + d_{\text{gap}} \right]_+^2, \end{aligned} \quad (6)$$

where  $\mathbf{p}_{\text{user}}$  encodes all the user preferences about the alignment between the mechanical template and the target shape. The first term encourages each component included in  $\mathbf{p}_{\text{user}}$  to satisfy the corresponding requirements as close as possible. The second term encourages components that are required to be contained inside the target geometry to remain inside  $\mathcal{S}$ .  $[\cdot]_+$  denotes the function  $\max(0, \cdot)$  and  $\lambda$  is the relative weighting between the terms.

The re-targeted variation  $\mathcal{V}$  of the mechanical template  $\mathcal{T}_M$  is obtained by optimizing for the parameter set  $\mathbf{p}^*$ , which minimizes

Equation 6 and satisfies the hard constraints:

$$\begin{aligned} \mathbf{p}^* &= \arg \min_{\mathbf{p}} E(\mathbf{p}, \mathcal{S}, \mathbf{p}_{\text{user}}) \\ \text{such that } &\begin{cases} \mathbf{c}_{\text{eq}}(\mathbf{p}) &= \mathbf{0} \\ \mathbf{c}_{\text{ineq}}(\mathbf{p}) &\geq \mathbf{0} \end{cases} \end{aligned} \quad (7)$$

### 5.2 Differential Manipulation

In order to solve the energy minimization problem given by Equation 7, we adapt the *Differential Manipulation* method [Gleicher and Witkin 1991]. Originally developed for editing vector graphics, this method is especially suitable for interactive exploration of a constrained design space. In general, differential manipulation enables incremental changes to the state of an  $N$ -dimensional feature point under a set of differential constraints, assuming that the initial state satisfies the constraints. In other words, the technique enables manipulation of the feature point along the constraint manifold. In the domain of vector drawings, constraints are defined as geometric relationships in a drawing that the user wants to preserve (e.g., continuity constraints, parallelism, etc.). In our setting, the state vector represents the set of parameters  $\mathbf{p}$  that define the current variation  $\mathcal{V}$  of the mechanical template  $\mathcal{T}_M$ . Since this initial state by definition satisfies all of the relevant mechanical and functional constraints, differential manipulation enables users to interactively modify that state to re-target the mechanism.

We formulate our differential manipulation problem as follows. Given the current state  $\mathbf{p}$  of the template, which is both valid and functional (i.e. fulfills the constraints of Equation 7) and a proposed user edit, defined by the user-specified values  $\mathbf{p}_{\text{user}}$  for a subset  $\tilde{\mathbf{p}}$  of  $\mathbf{p}$ , we first set  $\tilde{\mathbf{p}} = \mathbf{p}_{\text{user}}$ . This modified state generally does not satisfy the constraints, so we project back to the constraint manifold via the following two steps. First, we account for the containment constraints by taking a fixed step size  $\Delta$  along the gradient of the second term of Equation 6 resulting in the intermediate parameter set  $\mathbf{p}^0$ . While this step moves the state closer towards satisfying the containment constraints, the resulting state generally does not satisfy the mechanical and functional constraints (Equation 7). Thus, in a second step, we project the intermediate parameter set  $\mathbf{p}^0$  onto the constraint manifold. This projection is performed by solving the following unconstrained optimization problem:

$$\mathbf{p}^1 = \arg \min_{\mathbf{p}} \left( \|\mathbf{c}_{\text{eq}}(\mathbf{p})\| + \|\mathbf{c}_{\text{ineq}}(\mathbf{p})\|_- \right) \quad (8)$$

using BFGS [Nocedal and Wright 2006], which performs a variant of gradient descent starting from the intermediate parameter set  $\mathbf{p}^0$ . The resulting parameter set  $\mathbf{p}^1$  represents a fully functional variation of the mechanical template.

Depending on the user specified set of parameters  $\mathbf{p}_{\text{user}}$ , the energy  $E(\mathbf{p}^1, \mathcal{S}, \mathbf{p}_{\text{user}})$  may or may not evaluate to zero. In the first case (i.e.  $E(\mathbf{p}^1, \mathcal{S}, \mathbf{p}_{\text{user}}) = 0$ ), all spatial constraints are fully satisfied: all components that should lie inside the target shape are located as such and all components are placed according to the user preference. In case  $E(\mathbf{p}^1, \mathcal{S}, \mathbf{p}_{\text{user}}) \neq 0$ , however, the optimization process fails to find such a configuration that satisfies all the constraints including the user specified ones. In this case, we repeat the aforementioned two-step optimization process while iteratively halving

the step size  $\Delta$  until  $E(\mathbf{p}^1, \mathcal{S}, \mathbf{p}_{\text{user}})$  is zero or the step size is below a predefined threshold. This helps to avoid configurations where the parameter values oscillate between violating different constraints due to a large step size.

Both steps of the optimization process described above utilize the derivatives of the constraint functions, i.e.,  $\frac{\partial c(\mathbf{p})}{\partial \mathbf{p}}$  for a constraint function  $c$  defined over the parameters  $\mathbf{p}$  of the mechanical template. For most of the constraints, these derivatives are analytically defined and have closed-form functions, with only two exceptions: the containment constraint (see Equation 4) and the collision constraints. We now describe how we handle these.

**5.2.1 Containment Constraints.** For interactive performance, the exact evaluation of pairwise mesh distances between the component geometries  $\mathcal{G}_C$  and the target shape  $\mathcal{S}$  is prohibitively expensive. Instead, we equip each component  $C_i$  with a set  $\ell_i$  of representative sample locations on the surface of its geometry. The distance term  $d(\mathbf{p}_i, \mathcal{S}, \tau_{\text{comp}}(C_i))$  between  $C_i$  and  $\mathcal{S}$  (see Equation 4) is then realized as

$$d(\mathbf{p}_i, \mathcal{S}, \tau_{\text{comp}}(C_i)) \approx \max_{\mathbf{l} \in \ell_i} d(\mathbf{l}, \mathcal{S}),$$

with  $d(\mathbf{l}, \mathcal{S})$  denoting the signed point-mesh distance. This computation can be efficiently performed using axis-aligned bounding box hierarchies, i.e. AABB trees. As corresponding gradient direction, we use the vector between the nearest sample location and its closest point on the target shape  $\mathcal{S}$ . Note that the hierarchy has to be regenerated each time  $\mathcal{S}$  is edited, which takes in the order of seconds.

**5.2.2 Collision Constraints.** Real-time collision detection suffers from the same high computing requirements as the containment detection. Thus, we use geometric proxies (e.g., spheres, boxes, etc.) instead of the actual component geometry. Collisions between such proxies can be efficiently detected and we use the worst overlap direction as the gradient direction to push the components apart.

## 6 USER INTERACTION

Our system enables the user to guide the retargeting process by supporting both direct manipulation of mechanical components and simple geometric edits to the target shape.

### 6.1 Mechanism Editing

To change the position of a mechanical component  $C_i$ , the user simply drags it to the desired position. As the user drags, the system continuously updates the state of the entire mechanism, by solving the optimization problem stated in Equation 7. More specifically, at each time step, we update the energy function (Equation 6) with a spatial constraint  $\|\mathbf{p}_i - \mathbf{p}_{\text{user},i}\| = 0$ , where the spatial coordinates of  $\mathbf{p}_{\text{user},i}$  are set to the current mouse position  $\mathbf{x}_i$ . We then optimize to find the new position of all mechanical components. Once a user has manipulated a component, we keep the corresponding spatial constraint in the energy function so that, during subsequent editing operations, previously edited components tend to stay in their user-specified positions. We visualize these “active” spatial constraints by highlighting the relevant components in blue. Users can also deactivate a spatial constraint at any point.

While the user interaction constitutes a continuous dragging operation, it is possible that at some point, a larger non-continuous modification of the mechanism is required. A purely differential approach would not allow such a behavior and the system could get ‘stuck’. Thus, we strictly enforce the user placement *during* dragging, i.e., we fix the spatial coordinates of the selected component  $C_i$  to match the specified location  $\mathbf{x}_i$  and solve the minimization for the remaining parameters. At the end of the dragging operation – marked by the release of the mouse button – the spatial coordinates are unfixed and the full minimization problem solved. Note that the aforementioned spatial constraint is still maintained in order to realize the desired component placement; the constraint can be removed by a right click on the component.

### 6.2 Shape Editing

In some cases, editing the mechanism alone may not be sufficient to achieve the desired re-targeting result. This may be due to fundamental incompatibilities between the template  $\mathcal{T}_M$  and target shape  $\mathcal{S}$ . For example,  $\mathcal{S}$  may simply be too small to contain all the components that should lie in its interior under the specified mechanical and functional constraints. In such scenarios, it is necessary to adapt the geometry of  $\mathcal{S}$  itself. To support such edits, our system simply re-runs the mechanism optimization as the user modifies  $\mathcal{S}$ . As the geometry changes, we update the spatial containment constraints that define the relationship between the mechanism and  $\mathcal{S}$ . These updates can produce different optimization results (e.g., as the interior of  $\mathcal{S}$  grows, components can make use of the extra space). Since updating the signed distance constraint gradients requires more computation than updating the per-component spatial constraints required for mechanism editing, we run the optimization only after the user mouses up after a given geometric edit, rather than during dragging.

### 6.3 Visualization

The system visualizes the current state of the target shape  $\mathcal{S}$  and mechanical variation  $\mathcal{V}$  by rendering a semi-transparent view of

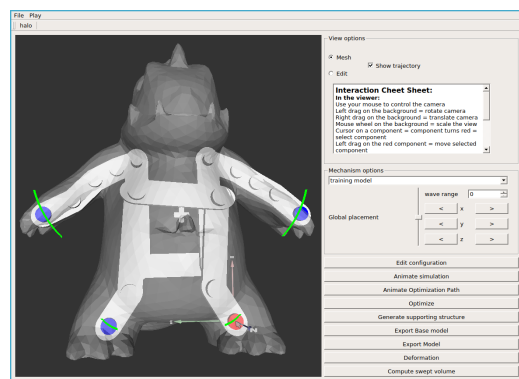


Fig. 7. Graphical user interface of our system. Simulated component trajectories are shown in green. Components are initially colored white and turn to blue if they are assigned preferred user placements. During dragging, the corresponding component is marked in red.

$\mathcal{S}$  on top of the mechanical component geometry (see Figure 7). Hovering the mouse cursor over a component highlights it in red to help users discover the individual, editable elements of the mechanism. As mentioned above, after a component has been manipulated, it turns blue to denote an active spatial constraint on its position. Finally, we visualize the effect of different mechanism variations by rendering component motions with green trajectories. We compute the trajectories using a forward simulation of the mechanism given a predefined range of input states for the driving components. This visualization allows the user to better see and evaluate the functional impact of different re-targeting candidates.

## 7 POST-PROCESSING AND FABRICATION

Once the user is satisfied with the re-targeting result, we perform several post-processing steps to ensure the final result is fabricatable. The post-processing step takes the re-targeted mechanism  $\mathcal{V}$  and the (potentially edited) target shape  $\mathcal{S}$  as input. As output, a fabrication-ready set of 3D surfaces of the final model is provided. This includes – in addition to the component geometry of the mechanism – a description of the outer shell of the model, cutaways of this shell to avoid collisions with the mechanism, and fixations of the components. We note that a typical workflow of our system does not require iterating between interactive re-targeting and fabrication steps since fabrication-related parameters, e.g. the minimum gap parameter in Equation 4, are already considered in the formulation of the optimization problem solved during interactive re-targeting.

### 7.1 Outer Shell

We convert the target shape  $\mathcal{S}$  to a shell  $\mathcal{O}$  of thickness  $d_{\mathcal{O}}$  by interior offsetting. Since  $d_{\mathcal{O}}$  has a direct influence on factors such as material consumption and appearance, we defer its choice to the user. However, the recent work of Musialski et al [2016] on structural stability of offset surfaces can be exploited for assistance. A solid model can be achieved by setting  $d_{\mathcal{O}} = \infty$ .

### 7.2 Cutaways

During interactive editing, we prohibit unintended collisions between the individual components of the mechanism. However, components can possibly collide with the target shape. In order to resolve such collisions, we introduce cutaways of the outer shell of the target geometry without modifying the re-targeting result. For static components, it is sufficient to assign an exclusion region to each of them, depending on the minimal gap size  $d_{\text{gap}}$  of the fabrication method. As illustrated on the right, this can be realized by computing outer offset surfaces or use geometric primitives (e.g., cylinders for axles and gears). For moving components, this process has to be repeated for each timestep. Thus, we perform a dense sampling of the state of the mechanism during its time evolution and compute the union of all exclusion regions for each component. Finally, we obtain a

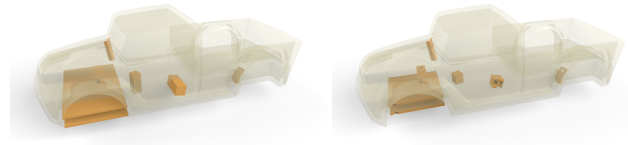
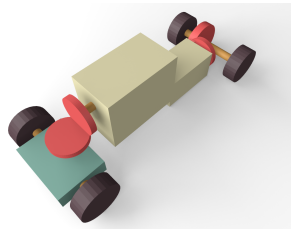


Fig. 8. Fixations of a DRIVETRAIN mechanism re-targeted to the PICK-UP shape. The fixations are shown before (left) and after (right) intersection with the cutaways and the outer shell.

set of cutaway geometries  $\mathcal{G}_{\text{cut}}$  – one for each component – that is subtracted from the outer shell via boolean mesh operations.

### 7.3 Fixations

So far, the mechanical template  $\mathcal{T}$  and its variations  $\mathcal{V}$  were essentially free-floating. For fabrication, various fixations have to be added to maintain the correct placement of each static component. However, only a subset of the components needs to be fixated as the mechanical constraints propagate the fixations' effect to the remaining parts of the mechanism (e.g., a fixated axle also fixates the gear connected to it).

As illustrated in Figure 8, fixations are solid geometric primitives, such as boxes. Their location is determined by (i) certain connection types (e.g., at one or both sides of a gear which is fixed to an axle); or (ii) by components themselves (e.g., to fix a motor holder to the outer shell). In both cases, the fixation are connected to a suitable location on the outer shell with an emphasis on keeping the size of the fixation as small as possible.

### 7.4 Final Model

Given the outer shell  $\mathcal{O}$  as well as the geometries for cutaways  $\mathcal{G}_{\text{cut}}$ , fixations  $\mathcal{G}_{\text{fix}}$ , and mechanism components  $\mathcal{G}_{\mathcal{C}}$ , the geometry of the final model  $\mathcal{G}$  is given as

$$\mathcal{G} = \mathcal{G}_{\mathcal{C}} \cup \left( (\mathcal{O} \cup \mathcal{G}_{\text{fix}}) \setminus \mathcal{G}_{\text{cut}} \right)$$

where  $\cup$  and  $\setminus$  denote the boolean union and boolean difference of all geometries, in the sense of constructive solid geometry (CSG).

For the fabrication of the final model, two additional steps may be necessary (see Figure 9).

### 7.5 Support Removal

For various additive manufacturing techniques, temporary support structures are required during the fabrication process. This is the case for photopolymerization of a liquid resin – which we primarily employ –; accordingly, we manually insert cutaways to facilitate the cleanup of the manufactured model.

### 7.6 Component Insertion

If non-printable parts are required for the functionality of the mechanism, sufficient accessibility of the mechanism needs to be available in the manufactured model. For results with the DRIVETRAIN mechanism, we manually create openings in the model for insertion of the motors.



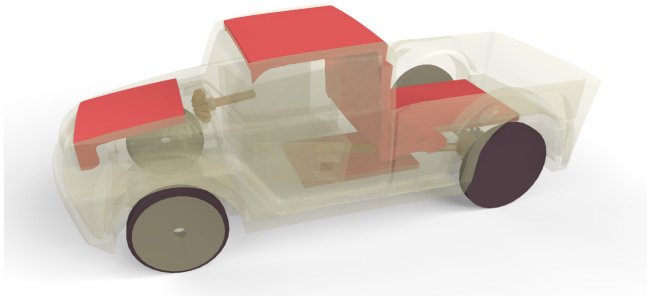


Fig. 9. Fabrication-ready version of the PICK-UP result. Manually added openings for support removal and component insertion are shown in red. The final model consist of several parts (shown in orange and black) that are separated by the minimal gap size  $d_{\text{gap}}$  specified by the user.

## 8 EVALUATION

We evaluate our system on several retargeting scenarios consisting of mechanical templates of various complexity. We fabricate several of these examples using a Stratasys<sup>®</sup> J750 3D printer. We further conduct a small user study to better evaluate the effectiveness of our system. We now discuss our main findings in detail.

### 8.1 Implementation

The implementation of our system utilizes libigl [Jacobson et al. 2017] to compute the spatial hierarchy for the containment constraints and to generate the cutaways. We use libQGLViewer [Debunne 2017] for visualizations and CppOptimizationLibrary [Wieschollek 2017] for the numerical optimization routines. Various components were generated using OpenSCAD [Kintel et al. 2017]. Manual post-processing of the model geometry was performed with Autodesk Netfabb<sup>®</sup>.

### 8.2 Results

We used our system to retarget four different mechanical templates (WIND-UP, DRIVETRAIN, TAPPING, and ROTORS) to various target shapes, which are either obtained from online resources or generated by the participants of our user study. We note that a user of our system may initiate the retargeting process by selecting or creating either a target shape or a mechanical template first. We provide details of the shapes and templates used in our evaluations in Figure 10 and Figure 11, respectively. Figure 12 shows the retargeting results for pairs of templates and target shapes. Figure 13 shows the 3D printed models, and the supplemental video depicts the dynamic behavior.

Our mechanical templates contain between 18 (DRIVETRAIN) and 48 (TAPPING) components. Our design system makes it easy for novice users to retarget these mechanisms to their desired shapes. For example, retargeting the TAPPING mechanism with existing tools requires accurately placing the mechanism inside a hand, which is a very confined space for all components to fit inside. With our optimization-in-the-loop approach, edits are propagated at interactive rates and important geometrical relationships in the mechanism are preserved while the user identifies an aesthetically pleasing

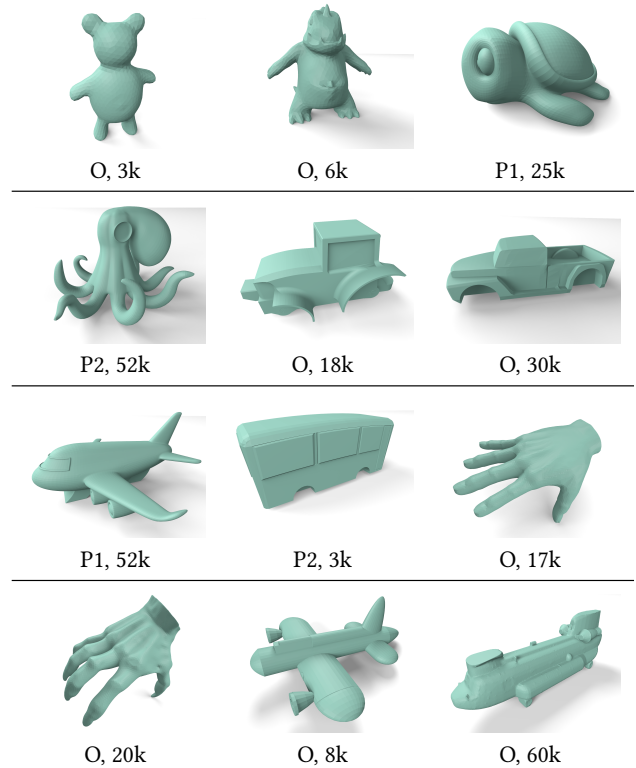


Fig. 10. Target shapes used for retargeting including labeling of the source (O: online model databases as denoted in the acknowledgments, P1-P3: participants of the user study) and the number of triangles.

result. Every participant of our user study, all of whom have a background in 3D modeling but not in mechanical engineering, stated that they would not be able to perform this task with a conventional modeling tool. We also evaluated the performance of our system. Each iteration of our optimization procedure takes between 20ms to 150ms, depending on the complexity of the template. For our most complex example (TAPPING), which contains a total of 459 variables and 64 hard constraints, our optimization was typically able to enforce all constraints in about 150ms.

The WIND-UP and DRIVETRAIN templates were each retargeted to four target shapes. Starting from the WIND-UP mechanism, users were able to create several compelling functional characters. Although we do not run a full simulation of the resulting motion including all geometric parts during interactive editing, our interface displays the trajectories of the end effectors to intuitively visualize the motion. In all the examples, the configuration of the mechanism had to be significantly adapted to the input shapes. For example, for the PICK-UP and TRACTOR, the motor placement shows important differences. While in the PICK-UP both motors are aligned horizontally, in the TRACTOR the driving motor had to be aligned vertically such that it fits inside the shape. Furthermore, the PICK-UP model required a shape change (slightly increased length of the driver's cabin and increased height of the rack body) such that the

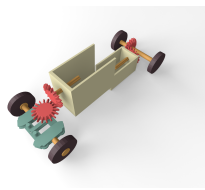
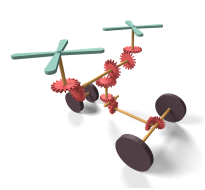
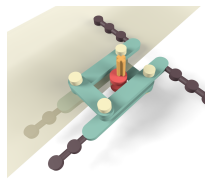
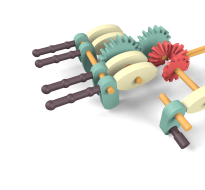
Mechanical template	DRIVETRAIN	ROTORS	WIND-UP	TAPPING
				
<b>Component count</b>	18	26	46	48
<b>Mechanical constraints</b>	16	22	67	64
<b>Functional constraints</b>	6	7	0	0
<b>Containment constraints</b>	16	15	36	46
<b>Components</b>	6 axles, 4 gears, 4 wheels, 1 steering, 2 motors, 1 ground plane	8 axles, 11 gears, 4 wheels, 2 propellers, 1 ground plane	1 axle, 2 driving disks, 10 bars, 8 pins, 10 rods, 14 handles, 1 split board	6 axles, 2 spur gears, 3 bevel gears, 5 elliptic gears, 5 bars, 13 rods, 14 handles

Fig. 11. Mechanical templates used for our results with functionalities ranging from a motorized steering-capable DRIVETRAIN, over wheel-driven ROTORS and a WIND-UP mechanism for waving arms and legs, to an elliptic-gear-based mechanism that realized periodic TAPPING motions.

whole mechanism fits. Our system supports such edits by efficient iterations between shape and mechanism editing.

For all our results, the fixations of the mechanism to the shape were generated automatically. Segmentations of the input geometry into rigid subparts, for example into body, arms, and legs for the TEDDY and MONSTER shape, were performed by the user. As can be seen in all examples, our system subtracts cutaways of the moving parts from the remaining model. Although this might lead to small visual artifacts in the object shape, as can be seen for example at the wheel arch of the TRACTOR model, we enforce this operation to ensure proper movement of the mechanism.

### 8.3 User Evaluation

To better understand the overall usability and effectiveness of our re-targeting system, we conducted a small, exploratory user study. We recruited three participants with professional 3D modeling background. After an initial training phase, in which they accustomed themselves with the controls of our system, they were individually guided through a simple re-targeting task to get an overview of the possible interactions. Afterwards, they were given three re-targeting tasks, which they performed on their own:

**Task 1:** Retarget a WIND-UP mechanism to a bi-pedal shape.

**Task 2:** Retarget a DRIVETRAIN mechanism to a car shape.

**Task 3:** Retarget each of the previous mechanism to a new shape created by themselves.

All three tasks were performed on a conventional workstation provided by us. For modeling new shapes, each user used tools of their preference (Autodesk Maya [Autodesk 2017], MODO [Modo 2017], and ZBrush [Pixologic 2017]). During the user study, we performed screen capturing and sound recording. We measured the completion time of each task and also asked questions with freeform responses.

**8.3.1 Findings.** All participants were able to produce functional models for all the tasks using our system. In particular, we were pleased that users were able to create functional versions of their own models. The users required between 2 to 8 min to complete

each task, with 4.8 min on average and a standard deviation of 2.1 min. Typically, in each interactive session the users manipulated 4-5 components, while all the remaining parameters were optimized automatically by our system. Our demo video shows an example editing session. The qualitative feedback was largely positive. All participants agreed that our system was conceptually easy to use and fast to learn. They found the system predictable and felt they had control over the re-targeting process. All participants appreciated that our system helped automate the time consuming process of designing functional objects. Users also suggested several straightforward improvements to our system, such as allowing a ranking of user constraints to enforce certain constraints more strictly and the option to deactivate the optimization temporarily when making several edits at once. Figure 10 shows a subset of shapes created by the users. In all cases, the users were able to turn them into functional models, two of which we fabricated: TURTLE (Figure 1) and OCTOPUS (Figure 13).

### 8.4 Limitations and Future Work

While our system enables users to create a wide set of re-targeting examples of varying complexity, the main limitation is the possibility of *deadlock* configurations that arise due to conflicting constraints. In our user study, we encountered in total two deadlock cases due to users fixing too many component positions resulting in over-constrained systems. The current workaround is to remove some constraints manually. In the future, we would like explore several options, as suggested by Gleicher et al. [1994] (e.g. actively detecting and visualizing conflicting constraints, temporarily disabling a constraint to enable further editing which may possibly recover from the deadlock configuration), to avoid and recover from such configurations.

Our optimization strategy is based on differential manipulation which walks along a constraint manifold in the gradient direction. While this makes it intuitive for the user to interactively observe the optimization performed by the system, the optimization can

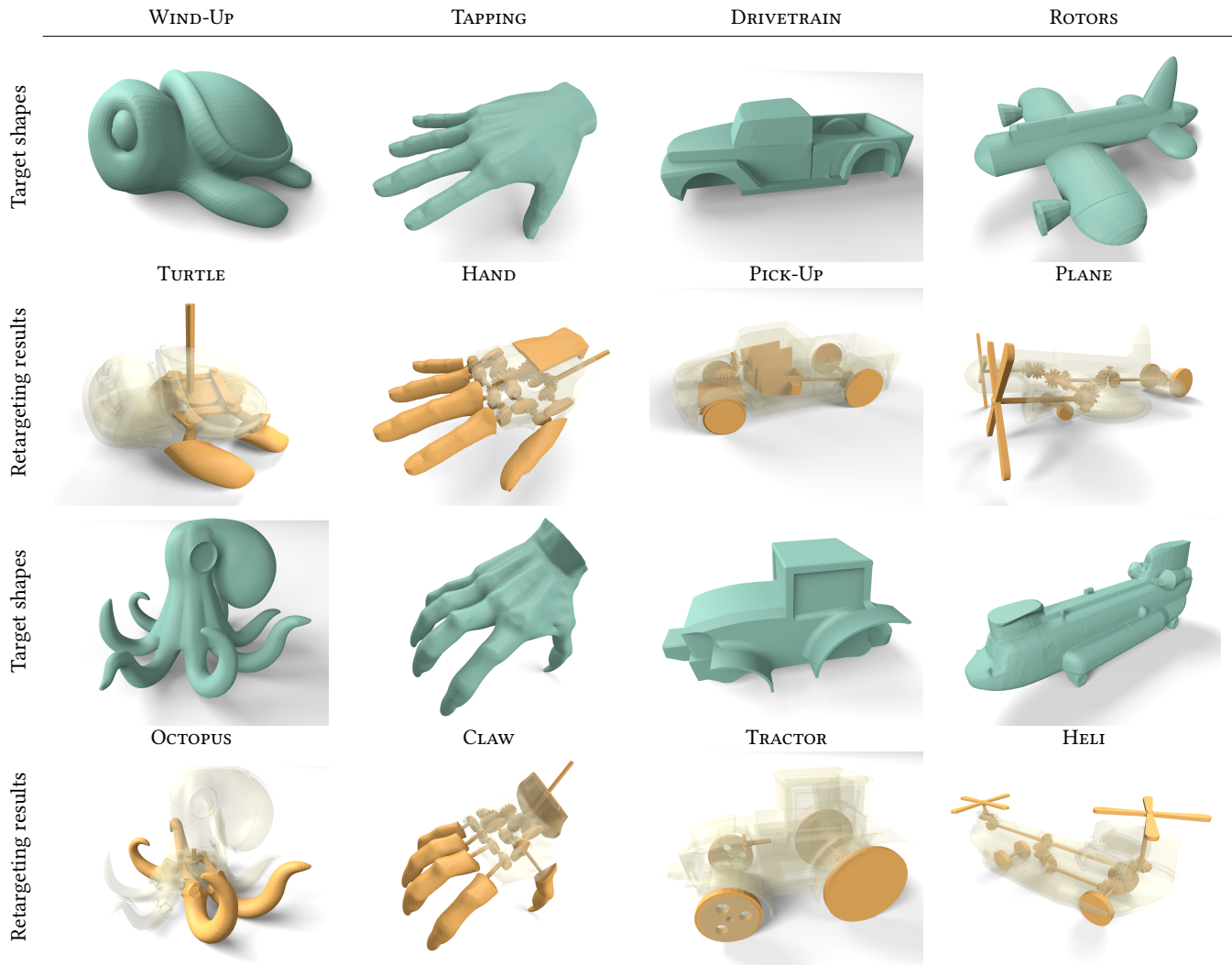


Fig. 12. Retargeting results for different mechanical templates and target shapes. In each column, a mechanical template (see Figure 11) is retargeted to two different target shapes. Below each target shape (given as a surface mesh), the corresponding retargeting result is given. Selective transparency is applied to fixations and parts of the outer shell to reveal the mechanical components.

get trapped in local minima (as is the case with any continuous gradient-based approach). While we depend on user guidance to recover from such configurations, an interesting research direction is to exploit user intuition during optimization in a more principled way, e.g. via a sketch-based interface.

Our approach relies on the availability of a mechanical template that often must be generated by an expert. This template also needs to correctly reflect fabrication constraints, such as required tolerances by the 3D printer. In the future, it would be interesting to find more automated ways to generate templates, perhaps by building on related work that can infer from a geometric model of a mechanical assembly how the individual parts move and interact with each other [Mitra et al. 2013b] or recent approaches on scanning and reconstructing mechanism models from images [Xu et al. 2016b].

As a usage scenario, we envision that creating a template will be a one-time effort for a specific mechanism. Such templates may also be shared to benefit the broader maker community.

While our system supports a wide range of mechanical and functional constraints and takes into account several important rules to obtain a valid design, there are additional aspects that can provide further improvement. Supporting symmetry constraints and dynamic mechanical components that follow a collision free trajectory are some examples. Analyzing emerging forces and torques in the system as well as structural stability during the design would also be very interesting.



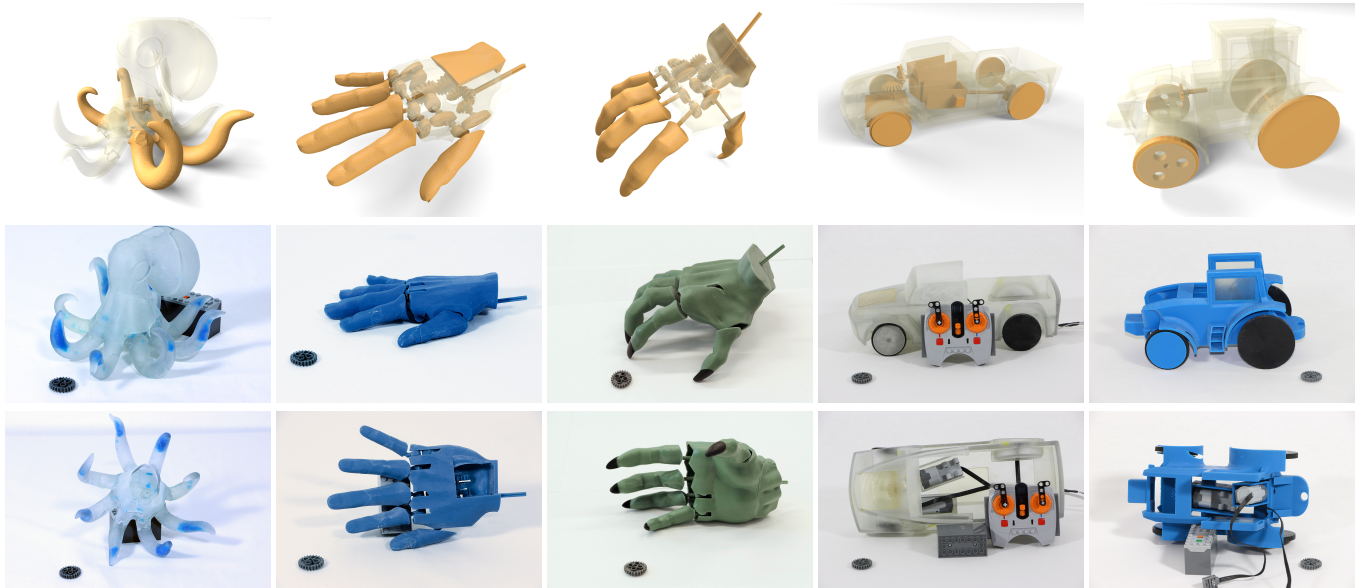


Fig. 13. From left to right: Retargeted (top) and fabricated (middle and bottom) results for the WIND-UP, TAPPING, and DRIVETRAIN template to the OCTOPUS, HAND, CLAW, PICK-UP, and TRACTOR shapes.

## 9 CONCLUSION

In this paper, we have proposed a new workflow for users to efficiently create functional mechanical objects by re-targeting an existing mechanical template to a given input shape. Based on the observation that for many functional objects the mechanical architecture remains similar while the form varies, we proposed a novel representation of the mechanism and functional features encoded in a mechanical template. The mechanical template represents a parametrized mechanism, spatial relationships of mechanical parts to the user-provided shape, and functional constraints that specify an intended functionality. Users can efficiently explore various design choices and instances of the mechanical template by interactively manipulating components in a user interface. We proposed an optimization-in-the-loop approach that supports finding a valid configuration such that low-level mechanical constraints, spatial relationships between form and mechanism, and higher-level functional goals are met. We demonstrated the efficacy of our system by re-targeting several mechanical templates to various shapes and fabricating the resulting customized functional objects.

## ACKNOWLEDGMENTS

We would like to thank everyone who contributed to this paper, especially the artists Abbas Saleh, Daniel Bösze, and David Ronnes for participating in our user study and allowing us to use their models created during the user study; furthermore we thank the authors of the remaining models for sharing them on Thingiverse as Rubber Duck (by Willie, CC0 1.0), Monster Mama (by mcallaghan95, CC BY-NC 3.0), Devilman Hand (by Renato T., CC BY-SA 3.0), B-17 Bomber (by Ethan F. at the Mastics-Moriches-Shirley Community Library's Teen 3D Print Club, CC BY-SA 3.0), Snap Together Farm Tractor (by Jon Stephenson, CC BY-NC 3.0), chinook helicopter

(by Paul Johnson, CC BY-NC-SA 3.0), and Pickup Truck (by Kalvin Daniels, CC BY-NC-SA 3.0), as well as on the McGill 3D Shape Benchmark [Siddiqi et al. 2008] as teddy10. We also want to express our gratitude to all proof-readers and anonymous reviewers.

## REFERENCES

- Ram Anantha, Glenn A Kramer, and Richard H Crawford. 1996. Assembly modelling by geometric constraint satisfaction. *Computer-Aided Design* 28, 9 (1996), 707 – 722.
- Autodesk. 2017. Maya, Computer Animation and Modeling Software. <http://www.autodesk.com/products/maya/overview>. (2017).
- Moritz Bäcker, Stelian Coros, and Bernhard Thomaszewski. 2015. LinkEdit: Interactive Linkage Editing Using Symbolic Kinematics. *ACM Trans. Graph.* 34, 4, Article 99 (July 2015), 8 pages.
- Moritz Bäcker, Emily Whiting, Bernd Bickel, and Olga Sorkine-Hornung. 2014. Spin-it: Optimizing Moment of Inertia for Spinnable Objects. *ACM Trans. Graph.* 33, 4, Article 96 (July 2014), 10 pages.
- Gilbert Louis Bernstein and Wilmot Li. 2015. Lillicon: Using Transient Widgets to Create Scale Variations of Icons. *ACM Trans. Graph.* 34, 4, Article 144 (July 2015), 11 pages.
- Dustin Beyer, Serafima Gurevich, Stefanie Mueller, Hsiang-Ting Chen, and Patrick Baudisch. 2015. Platener: Low-Fidelity Fabrication of 3D Objects by Substituting 3D Print with Laser-Cut Plates. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 1799–1806.
- Gaurav Bharaj, Stelian Coros, Bernhard Thomaszewski, James Tompkin, Bernd Bickel, and Hanspeter Pfister. 2015. Computational Design of Walking Automata. In *ACM SCA (SCA '15)*. ACM, New York, NY, USA, 93–100.
- Bernd Bickel, Moritz Bäcker, Miguel A. Otaduy, Hyunho Richard Lee, Hanspeter Pfister, Markus Gross, and Wojciech Matusik. 2010. Design and Fabrication of Materials with Desired Deformation Behavior. *ACM Trans. Graph.* 29, 4, Article 63 (July 2010), 10 pages.
- Duygu Ceylan, Wilmot Li, Niloy J. Mitra, Maneesh Agrawala, and Mark Pauly. 2013. Designing and Fabricating Mechanical Automata from Mocap Sequences. *ACM Trans. Graph.* 32, 6, Article 186 (Nov. 2013), 11 pages.
- Shean-Juinn Chiou and Kota Sridhar. 1999. Automated conceptual design of mechanisms. *Mechanism and machine theory* 34, 3 (1999), 467–495.
- Stelian Coros, Bernhard Thomaszewski, Gioacchino Noris, Shinjiro Sueda, Moira Forberg, Robert W. Sumner, Wojciech Matusik, and Bernd Bickel. 2013. Computational Design of Mechanical Characters. *ACM Trans. Graph.* 32, 4, Article 83 (July 2013), 12 pages.



- Gilles Debunne. 2017. libQGLViewer. <https://github.com/GillesDebunne/libQGLViewer/>. (2017).
- Yue Dong, Jiaping Wang, Fabio Pellacini, Xin Tong, and Baining Guo. 2010. Fabricating Spatially-varying Subsurface Scattering. *ACM Trans. Graph.* 29, 4, Article 62 (July 2010), 10 pages.
- Tao Du, Adriana Schulz, Bo Zhu, Bernd Bickel, and Wojciech Matusik. 2016. Computational Multicopter Design. *ACM Trans. Graph.* 35, 6, Article 227 (Nov. 2016), 10 pages.
- J er mie Dumas, Jean Hergel, and Sylvain Lefebvre. 2014. Bridging the Gap: Automated Steady Scaffolding for 3D Printing. *ACM Trans. Graph.* 33, 4, Article 98 (July 2014), 10 pages.
- M. Gleicher and A. Witkin. 1991. Differential Manipulation. In *Proceedings of Graphics Interface '91 (GI '91)*. 61–67.
- M. Gleicher and A. Witkin. 1994. Drawing with Constraints. *The Visual Computer* 11, 1 (1994), 39–51.
- Kirk Haller, Audrey Lee-St. John, Meera Sitharam, Ileana Streinu, and Neil White. 2009. Body-and-cad Geometric Constraint Systems. In *Proceedings of the 2009 ACM Symposium on Applied Computing (SAC '09)*. ACM, New York, NY, USA, 1127–1131.
- Young-Hyun Han and Kunwoo Lee. 2006. A Case-based Framework for Reuse of Previous Design Concepts in Conceptual Synthesis of Mechanisms. *Comput. Ind.* 57, 4 (May 2006), 305–318.
- Ruizhen Hu, Oliver van Kaick, Bojan Wu, Hui Huang, Ariel Shamir, and Hao Zhang. 2016. Learning How Objects Function via Co-analysis of Interactions. *ACM Trans. Graph.* 35, 4, Article 47 (July 2016), 13 pages.
- Alec Jacobson, Daniele Panozzo, and others. 2017. libigl: A simple C++ geometry processing library. (2017). <https://libigl.github.io/libigl/>.
- Han-Jong Kim, Yunwoo Jeong, Ju-Whan Kim, and Tek-Jin Nam. 2016. M.Sketch: Prototyping Tool for Linkage-Based Mechanism Design. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16 Adjunct)*. ACM, New York, NY, USA, 75–77.
- Marius Kintel, Clifford Wolf, and others. 2017. OpenSCAD. <https://github.com/openscad/openscad/>. (2017).
- Bongjin Koo, Wilmot Li, JiaXian Yao, Maneesh Agrawala, and Niloy J. Mitra. 2014. Creating Works-like Prototypes of Mechanical Objects. *ACM Trans. Graph.* 33, 6, Article 217 (Nov. 2014), 9 pages.
- Yuki Koyama, Shinjiro Sueda, Emma Steinhart, Takeo Igarashi, Ariel Shamir, and Wojciech Matusik. 2015. AutoConnect: Computational Design of 3D-printable Connectors. *ACM Trans. Graph.* 34, 6, Article 231 (Oct. 2015), 11 pages.
- Lin Lu, Andrei Sharf, Haisen Zhao, Yuan Wei, Qingnan Fan, Xuelin Chen, Yann Savoye, Changhe Tu, Daniel Cohen-Or, and Baoquan Chen. 2014. Build-to-last: Strength to Weight 3D Printed Objects. *ACM Trans. Graph.* 33, 4, Article 97 (July 2014), 10 pages.
- Linjie Luo, Ilya Baran, Szymon Rusinkiewicz, and Wojciech Matusik. 2012. Chopper: Partitioning Models into 3D-printable Parts. *ACM Trans. Graph.* 31, 6, Article 129 (Nov. 2012), 9 pages.
- Vittorio Megaro, Bernhard Thomaszewski, Damien Gauge, Eitan Grinspun, Stelian Coros, and Markus Gross. 2014. ChaCra: An Interactive Design System for Rapid Character Crafting. In *ACM SCA (SCA '14)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 123–130.
- Vittorio Megaro, Bernhard Thomaszewski, Maurizio Nitti, Otmar Hilliges, Markus Gross, and Stelian Coros. 2015. Interactive Design of 3D-printable Robotic Creatures. *ACM Trans. Graph.* 34, 6, Article 216 (Oct. 2015), 9 pages.
- Niloy J. Mitra, Michael Wand, Hao Zhang, Daniel Cohen-Or, and Martin Bokeloh. 2013a. Structure-Aware Shape Processing. In *EUROGRAPHICS State-of-the-art Report*.
- Niloy J. Mitra, Yong-Liang Yang, Dong-Ming Yan, Wilmot Li, and Maneesh Agrawala. 2013b. Illustrating How Mechanical Assemblies Work. *Commun. ACM* 56, 1 (Jan. 2013), 106–114.
- Modo. 2017. MODO Creative Modeling Software. <https://www.thefoundry.co.uk/products/modo/>. (2017).
- Przemyslaw Musialski, Christian Hafner, Florian Rist, Michael Birsak, Michael Wimmer, and Leif Kobbelt. 2016. Non-linear Shape Optimization Using Local Subspace Projections. *ACM Trans. Graph.* 35, 4, Article 87 (July 2016), 13 pages.
- Jorge Nocedal and Stephen J. Wright. 2006. *Numerical optimization* (2. ed.). Springer, New York, NY.
- Jes s P rez, Bernhard Thomaszewski, Stelian Coros, Bernd Bickel, Jos  A. Canabal, Robert Sumner, and Miguel A. Otaduy. 2015. Design and Fabrication of Flexible Rod Meshes. *ACM Trans. Graph.* 34, 4, Article 138 (July 2015), 12 pages.
- Pixologic. 2017. Pixologic: Home of ZBrush. <http://pixologic.com/>. (2017).
- Romain Pr vost, Emily Whiting, Sylvain Lefebvre, and Olga Sorkine-Hornung. 2013. Make It Stand: Balancing Shapes for 3D Fabrication. *ACM Trans. Graph.* 32, 4, Article 81 (July 2013), 10 pages.
- U. Roy, N. Pramanik, R. Sudarsan, R.D. Sriram, and K.W. Lyons. 2001. Function-to-form mapping: model, representation and applications in design synthesis. *Computer-Aided Design* 33, 10 (2001), 699 – 719.
- Adriana Schulz, Ariel Shamir, David I. W. Levin, Pitchaya Sitthi-amorn, and Wojciech Matusik. 2014. Design and Fabrication by Example. *ACM Trans. Graph.* 33, 4, Article 62 (July 2014), 11 pages.
- Kaleem Siddiqi, Juan Zhang, Diego Macrini, Ali Shokoufandeh, Sylvain Bouix, and Sven Dickinson. 2008. Retrieving Articulated 3-D Models Using Medial Surfaces. *Mach. Vision Appl.* 19, 4 (May 2008), 261–275.
- M lina Skouras, Bernhard Thomaszewski, Peter Kaufmann, Akash Garg, Bernd Bickel, Eitan Grinspun, and Markus Gross. 2014. Designing Inflatable Structures. *ACM Trans. Graph.* 33, 4, Article 63 (July 2014), 10 pages.
- Peng Song, Zhongqi Fu, Ligang Liu, and Chi-Wing Fu. 2015a. Printing 3D objects with interlocking parts. *Computer Aided Geometric Design* 35-36 (2015), 137 – 148. Geometric Modeling and Processing 2015.
- S. Song, J. Kim, and K. Yamane. 2015b. Development of a bipedal robot that walks like an animation character. In *IEEE ICRA*. 3596–3602.
- Ondrej Stava, Juraj Vanek, Bedrich Benes, Nathan Carr, and Radom r M ch. 2012. Stress Relief: Improving Structural Strength of 3D Printable Objects. *ACM Trans. Graph.* 31, 4, Article 48 (July 2012), 11 pages.
- Timothy Sun and Changxi Zheng. 2015. Computational Design of Twisty Joints and Puzzles. *ACM Trans. Graph.* 34, 4, Article 101 (July 2015), 11 pages.
- Bernhard Thomaszewski, Stelian Coros, Damien Gauge, Vittorio Megaro, Eitan Grinspun, and Markus Gross. 2014. Computational Design of Linkage-based Characters. *ACM Trans. Graph.* 33, 4, Article 64 (July 2014), 9 pages.
- Nobuyuki Umetani, Takeo Igarashi, and Niloy J. Mitra. 2012. Guided Exploration of Physically Valid Shapes for Furniture Design. *ACM Trans. Graph.* 31, 4, Article 86 (July 2012), 11 pages.
- Nobuyuki Umetani, Yuki Koyama, Ryan Schmidt, and Takeo Igarashi. 2014. Pteromys: Interactive Design and Optimization of Free-formed Free-flight Model Airplanes. *ACM Trans. Graph.* 33, 4, Article 65 (July 2014), 10 pages.
- Francisca Gil Ureta, Chelsea Tymms, and Denis Zorin. 2016. Interactive Modeling of Mechanical Objects. *Computer Graphics Forum* (2016).
- Patrick Wieschollek. 2017. CppOptimizationLibrary. <https://github.com/PatWie/CppNumericalSolvers/>. (2017).
- Yue Xie, Weiwei Xu, Yin Yang, Xiaohu Guo, and Kun Zhou. 2015. Agile structural analysis for fabrication-aware shape editing. *Computer Aided Geometric Design* 35-36 (2015), 163–179. Geometric Modeling and Processing 2015.
- Kai Xu, Vladimir G. Kim, Qixing Huang, Niloy Mitra, and Evangelos Kalogerakis. 2016a. Data-driven Shape Analysis and Processing. In *SIGGRAPH ASIA 2016 Courses (SA '16)*. ACM, New York, NY, USA, Article 4, 38 pages.
- Mingliang Xu, Mingyuan Li, Weiwei Xu, Zhigang Deng, Yin Yang, and Kun Zhou. 2016b. Interactive Mechanism Modeling from Multi-view Images. *ACM Trans. Graph.* 35, 6, Article 236 (Nov. 2016), 13 pages.
- Ran Zhang, Shiwei Wang, Xuejin Chen, Chao Ding, Luo Jiang, Jie Zhou, and Ligang Liu. 2016. Designing Planar Deployable Objects via Scissor Structures. *IEEE Transactions on Visualization and Computer Graphics* 22, 2 (Feb. 2016), 1051–1062. <http://dx.doi.org/10.1109/TVCG.2015.2430322>
- Haiming Zhao, Chengkuan Hong, Juncong Lin, Xiaogang Jin, and Weiwei Xu. 2016. Make It Swing. *Comput. Aided Geom. Des.* 43, C (March 2016), 226–236.
- Changxi Zheng, Timothy Sun, and Xiang Chen. 2016. Deployable 3D Linkages with Collision Avoidance. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '16)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 179–188.
- Qingnan Zhou, Julian Panetta, and Denis Zorin. 2013. Worst-case Structural Analysis. *ACM Trans. Graph.* 32, 4, Article 137 (July 2013), 12 pages.
- Lifeng Zhu, Weiwei Xu, John Snyder, Yang Liu, Guoping Wang, and Baining Guo. 2012. Motion-guided Mechanical Toy Modeling. *ACM Trans. Graph.* 31, 6, Article 127 (Nov. 2012), 10 pages.
- H. Zimmer and L. Kobbelt. 2014. Zometool Rationalization of Freeform Surfaces. *IEEE Transactions on Visualization and Computer Graphics* 20, 10 (Oct 2014), 1461–1473.