AutoConnect: Computational Design of 3D-Printable Connectors

Yuki Koyama[1,2] Shinjiro Sueda[1,3] Emma Steinhardt[1] Takeo Igarashi[2] Ariel Shamir[1,4] Wojciech Matusik[5]

[1]Disney Research Boston [2]The University of Tokyo [3]Cal Poly [4]IDC Herzliya [5]MIT

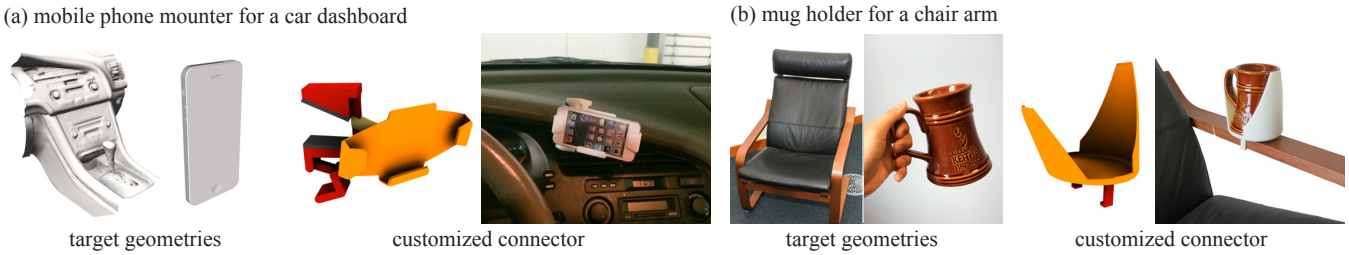


Figure 1: *Example scenarios of connecting two objects together with our method, AutoConnect. Based on geometries of the target objects and user-specified configurations, AutoConnect generates a 3D-printable customized connector. (a) A mobile phone is connected to a car dashboard. (b) A mug is connected to a chair arm*

Abstract

We present AutoConnect, an automatic method for designing a customized, 3D-printable connector that attaches two physical objects together. With our method, the user simply positions and orients virtual models of the two objects that he/she wants to connect, and our method creates several alternative designs. The user can then choose the connector that he/she likes or modify the design using a set of high-level parameters. The final connector can be printed using a commercial 3D printer. We broadly categorize the holders into two types. First, to hold standard objects such as pipes and planes, we utilize a database of parameterized mechanical fasteners and optimize the parameters based on the grip strength requirement and the material consumption. Second, to hold free-form objects, we procedurally generates shell-gripper designs using a region growing approach with geometric analysis. We demonstrate some application scenarios for connectors that can enhance our daily lives such as a mount that connects a mobile phone to your car dashboard or a cup-holder for a table.

1 Introduction

Modern manufacturing processes such as 3D printing allow users to personally design and manufacture complex 3D objects. One of the most exciting aspects of such personalized manufacturing is *customization*. Several recent works address different aspects of how to customize existing designs to suit some given specification or some user's needs and desires. Often, customization or new functionalities can be achieved not by creating a new object, but by combining two existing ones. In fact, many objects in digital modeling web sites such as Thingiverse are designed to connect and attach existing objects to other objects in new ways. For instance, one can download many different designs for attaching smart phones to bikes.

In this paper we address the problem of designing a custom connector that connects two objects together. Currently, if the user wants to connect two objects, she may search digital design sites in hopes of finding the right connector. However, more often than not, the specific design is difficult to find, or is not available at all. She can also try using 3D modeling tools to create a custom connector, but this demands expertise and is often challenging and complex. This is also tedious to do for every new pair of objects that need to be connected. In contrast, we present our AutoConnect method that requires the user to provide 3D models of two existing objects, such as an iPhone and a bike or a mug and a table, and only position them in a given configuration relative to each other. AutoConnect will automatically create a 3D model of a customized connector that attaches the two objects in the given configuration, is functional, and can be fabricated using a 3D printer. In essence, the user provides "what & where," and our AutoConnect method determines "how."

This problem is challenging because a totally new design of a 3D object that can connect the two input objects must to be created automatically from scratch. There are many different types of possible connections to choose the design from. The two objects can have a completely different geometric structure: some objects (*e.g.,* chairs, tables) contain common shapes such as pipes or planes, but others (*e.g.,* the Stanford bunny or a woman's shoe) have a free-form surface design. In addition, other considerations such as printing cost and aesthetics also come into play, and the connector must be physically fabricated and therefore, should hold and be sturdy enough.

Our design method works as follows. The input is a pair of 3D models that the user wants to connect. These models are loaded into a view and the user positions and orients them virtually. AutoConnect then provides a number of suggestions of potential designs that are tailored specifically to connect the two input objects. These suggestions are computed by running geometric analysis and force analysis in the background. The user can then choose a design she likes or modify the design interectively by specifying various constraints and objectives, such as the regions on the objects that the connector cannot cover and the maximum force that the connector must withstand.

To create the actual design we define two types of *holders* based on the categorization of the shape of the objects being connected. A connector consists of two holders, one for each object, and a rod connecting them. The first type of holder is for a structured object, which consists of simple, standard shapes such as a cylinder and box edges. For a structured object, we use predefined parameterized mechanical fasteners, such as pipe clamps, and customize them according to the object dimension and weight (§4). The second type is for freeform object that cannot be held using structured mechanical fasteners. For these shapes, we create a custom holder using a region growing algorithm based on geometric analysis (§5). The type of holder for the two objects is defined by the user when positioning them in place.

Our main contributions are as follows:

- We are first, to the best of our knowledge, to address the problem of computationally designing connectors.
- We present a technique to create 3D-printable mechanical fasteners by optimizing their shape parameters according to a grip-strength estimation based on measurements, while minimizing material consumption. We should also say that the DB is a contribution.
- We present a technique to generate holders for free-formed objects that ensures appropriate hold-ability and grip using a region growing algorithm based on geometry analysis.
- Using these techniques we present an automatic method to create connectors that can be physically printed and show various examples of generated connectors that are practical in everyday scenarios (Fig. 1).

2 Related Work

Commercial Sites and Systems Today, one of the most popular ways to obtain functional models for fabrication is to search for such models on the Internet. For example, [Thingiverse] is a web site for such a purpose where end-users can download and share 3D models. A user can find many iPhone cases and bike connectors, but if she wants a special kind of case or connector, chances are that this specific 3D model will not be found. Some 3D models allow customization by exposing some parameters, but customized connectors are rare and are usually limited to some trivial modifications—rarely can the user change the type of objects being connected. In contrast, our method allows creating a completely new connector for many types of different objects, and allows exchanging them freely.

Another option to create a connector is to design it using CAD tools such as Solidworks [Dassault Systèmes] or AutoCAD [Autodesk]. However, this requires expertise, and more importantly, the user would still need to take into account any functionality constraints while modeling. We target a method that will allow end users to simply generate a connector between any target shapes without having to consider its functionality or design.

Functional Design and Fabrication To allow non-expert users to create functional objects, researchers have investigated various computational design methods, each of which formulates a specific functionality for fabrication. For example, Prévost et al. [2013] proposed a method to evaluate and optimize the *stand-ability* of 3D-printed objects, while the method proposed by Bächer et al. [2014] enables to consider *spin-ability* of objects. Umetani et al. [2014] proposed a method to deal with *fly-ability* of lasercut paper airplanes. General *structural strength* of 3D-printed objects has also been investigated recently in [Stava et al. 2012; Lu et al. 2014]. In our method, we focus on two specific functionalities for designing connectors: *grip strength* of mechanical fasteners, and *hold-ability* of holders for free-form shapes.

Other works follow an interactive approach to modeling, while still constraining some functionalities in design. These include durability and validity in interactive furniture design in [Umetani et al. 2012], creation of chairs in [Saul et al. 2011], stacking ability in [Li et al. 2012], and fabricability by including connectors and fasteners in [Schulz et al. 2014]. In our work, the human interaction part is kept to a minimum–only placing the two objects relative to each other and choosing some initial parameters.

The system proposed by Schulz et al. [2014] also utilizes existing functional models to create new ones by-parts. Such an example-based approach to modeling is promising, especially to help novice users with little knowledge to design new objects. Our method for designing mechanical fasteners is an implicit example-based method, since we use a database of parametric models of 3D-printable mechanical fasteners. However, the user does not need to explicitly pick or change the design–this is done automatically.

3D-Printing Mechanisms Modern commercial 3D printers allow printing not only static parts, but complex mechanical objects. By combining mechanism design with computational techniques, researchers have investigated several applications. For example, designing and printing articulated models [Calì et al. 2012; Bächer et al. 2012], toys [Zhu et al. 2012; Zhou et al. 2014], characters with designable motions [Coros et al. 2013; Thomaszewski et al. 2014], and even figures that mimic human motions [Ceylan et al. 2013]. Others allow printing small working-prototypes of mechanical designs for testing [Koo et al. 2014]. Our method is first to utilizes mechanical *fasteners* for effectively gripping target objects. In addition, we combine the design with real-world measurement data to estimate functionality given different shape parameters. This measurement-based approach encapsulates the complex mechanism of fasteners, and allows handling many types of mechanisms in a consistent manner.

Cost considerations also come into play when printing. Similar to previous work that examine the tradeoff between cost and strength [Wang et al. 2013], we consider the tradeoff between cost and grip strength while optimizing the shape parameters of mechanical fasteners.

Analysis of Holding Objects Methods to hold objects could be broadly classified into two categories: holding objects by using frictional forces (*e.g.,* pinching an object with two fingers), and holding objects by contact regardless of frictions (*e.g.,* enveloping an object in your palm). In robotic grasping theory [Bicchi and Kumar 2000], these two alternatives are called *force closure* and *form closure* respectively. In our work we use both methods. We use force closure for mechanical fasteners using data-driven measurements, and form closure for free-form objects. A force-closure approach is also possible for free-form objects, by using physical analysis. For example, Chen et al. [2014] introduce a force-closure example of a phone holder. However, robotic hands can be actively actuated, while 3D-printed holders are passive. Our form-closure approach to free-form objects uses a *hold-ability* criterion. This criterion is related to methods in robotic grasp, but is tailored to our application. In robotic, robotic-hands with a small number of contacts are used (*e.g.,* tips of robotic fingers), while we generate a continuous wrapper that fits the target free-form shape.

We are also interested in holders that do not completely restrain the objects. These holder have *free* directions, where the held object is able to move without being blocked by contact. This allows, for instance, easy insertion or removal of the object. Such *blocking* relationships are investigated in *assembly planning* domain [Wilson 1992; Hirukawa et al. 1994; Agrawala et al. 2003]. Our method uses a similar formulation to generate holders with free directions. However, as our target shapes is a free-form holder enclosing a free-form object, more contact points have to be dealt with than in assembly planning scenarios.

3 Overview

Given two 3D objects that are to be connected, our goal is to automatically create the design for a connector that can be 3D printed and used. Hence, the input to our method are two 3D models (i.e. water-tight triangle meshes) of the objects, or a proxy for the area being connected (e.g. a bar representing a bike's handle or a table's leg). For many mass-manufactured objects, such as mobile phones

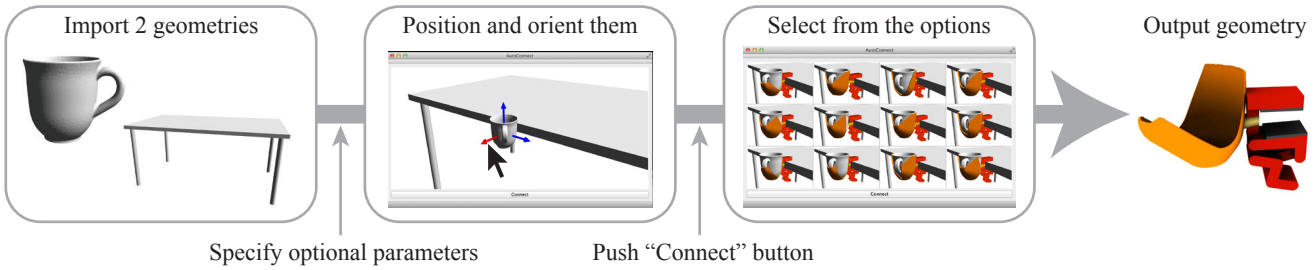


Figure 2: *User experience. Our method does not require any special interactions to obtain a connector; our method generates many connectors from a simple input and allows the user to choose the best one. The user can also specify some optional parameters such as a free motion and a region that should not be covered.*

and bikes, 3D meshes or CAD data are available online. Models of other objects can be acquired using scanners, 3D cameras and software such as Kinect Fusion. In some cases, the user needs to specify some additional information such as the object's mass or shape labeling (see §4), a free-motion direction, or a region that should remain uncovered (see §5).

To define the connector, the two objects must be positioned in the desired configuration relative to each other in 3D space. Therefore, we provide a simple user interface where the two objects to be connected are loaded and the user positions and orients them, and then presses the *Connect* button. After a short computation time (usually a few minutes and up to an hour), the system automatically generates several functional connector designs, presents them to the user, and allows the user to choose the most appropriate one for his/her purpose. To increase diversity among the candidate designs we implement various different strategies to produce connectors. Also, the user can revise the design by directly specifying or changing some design parameters or creation strategies. Fig. 2 shows this workflow from the user's perspective.

The actual design is created by classifying the input objects into two categories: structured objects (§4) and free-form objects (§5). The final connector is created by combining two such holders using a simple rod with a fixed radius. To hold structured objects such as pipes and planes, we utilize a database of parameterized mechanical fasteners and optimize the parameters based on the grip strength requirement and the material consumption. To hold free-form objects, we generate shell-gripper designs procedurally using a region growing approach with geometric analysis. Another case that is supported by our method is when one of the objects being connected is also 3D printed. In this case, only one "holder" is needed and it is attached directly to the printed object (see Fig. 18).

4 Fasteners for Structured Objects

We define a *structured* object as one in which its attachment area can be well approximated by a *standard* shape, which, in our current implementation, consists of: *cylinder*, *rectangular-prism*, *box-edge*, and *flat-plane* (Fig. 3). These standard shapes need not describe the whole shape—a single object can have multiple attachment areas. As an example, a table can be approximated by a flat-plane at the top, four cylindrical legs, and box-edge at the table edges. The attachment area of a 3D object can be classified into one of these shapes by analyzing the object using geometry processing methods such as slippage analysis [Gelfand and Guibas 2004], or manually labeled by the user. In this work we assume that we are given this information. We create a database of these fasteners annotated with their grip strength information (§4.1). This database of measurements is created once; it does *not* need to be recreated by each user. During run time, once the user has oriented and positioned

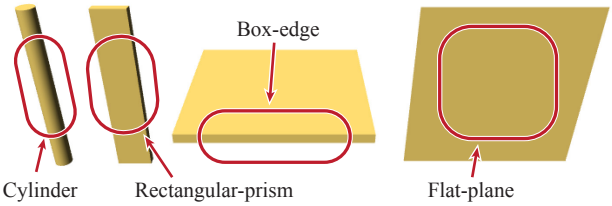


Figure 3: *Standard shapes in our current implementation.*

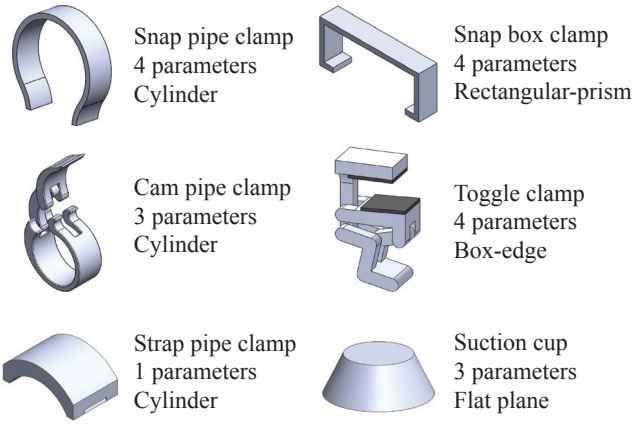


Figure 4: *Mechanical fasteners in our database.*

the objects, we query this database for a set of candidate fastener designs (§4.2) by optimizing for minimum material consumption while maintaining the requested grip strength. Any object that does not fall under one of these categories is treated as a free-form shape (§5).

We currently use six types of mechanical fasteners (see Fig. 4). Each fastener types corresponds to one of the four standard shapes (*e.g.,* cylinder, flat plane). These fasteners are designed using CSG-tree representations and are parameterized with a small number ($\leq$ 4) of parameters, such as "width" and "thickness." Some of these parameters are used to make the fastener exactly fit the target shape dimensions, and the remaining parameters are used to search for a set of designs that minimize the volume and provide adequate grip strength. All of the fasteners, including suction cups and multi-material toggle clamps, can be printed using a commercial printer. The database can be easily extended to include other types of fasteners created using any CAD software capable of parametric design.

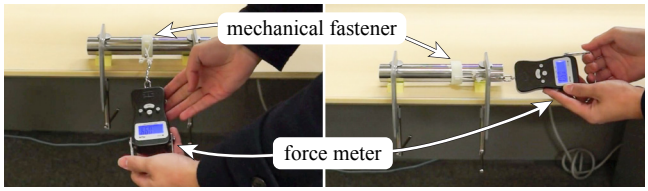


Figure 5: *Setting for measurement of the grip strength. We measured the minimum force that is required to (slightly) move the fastener by pulling in various directions.*

4.1 Estimating the Grip Strength

Under normal operating conditions, mechanical fasteners are expected to stay rigidly attached and not move. To be precise, we use the term *grip strength* to mean the minimum force required to perturb the fastener away from the gripping configuration. When we search the database for the appropriate fasteners §4.2, we require an estimate of the grip strength provided by the fastener for various parameter values. We take a data-driven approach by building a lookup table of real data measurements. Data-driven physics has been recently used for other applications including design of deformable objects [Bickel et al. 2010] and paper airplanes [Umetani et al. 2014].

An alternative approach for estimating the grip strength is to use physical simulation each time a parameter is changed. However, because mechanical fasteners operate based on a complex interplay of various factors, such as elasticity of the shape and friction at the contacting areas, accurately simulating the grip strength of the design is challenging and time consuming. Furthermore, other types of fasteners not in our implementation may work by some other means such as Velcro or screws, and so using a data-driven approach allows for more flexibility when adding new types of fasteners even when we do not have access to appropriate simulators. Note, however, that it is possible to use simulation results in the lookup table along with measured results.

The main idea is to use scattered-data interpolation to learn the relationship between the design parameters $\mathbf{x}$ and the grip strength. We physically measure the grip strength for a discrete set of sample points in parameter space, and use interpolation to define the grip strength function $G(\mathbf{x})$ on the whole parametric space. In our implementation, we use Radial Basis Function (RBF) interpolation [Anjyo et al. 2014], with the Gaussian kernel $K(x) = \exp(-\frac{x^2}{2})$. Other types of interpolation are also possible.

To compute the grip strength for the sample points, we print each fastener model with various parameter settings. Each new setting defines a point in the parametric space of the specific fastener. We physically measure the minimum force that is required to move each fastener by slowly pulling on the fastener as shown in Fig. 5. We use a digital force meter to measure forces, and a video camera to record the measurement process, so that we can read the maximum force before the fastener starts to move. We consider various possible directions for each printed fastener and use the minimum force from among them. For each mechanical fastener we obtained 16 to 25 sample points according to the number of design parameters. Details of the measurement setting and discussions of the quality of this approach can be found in the appendix. Although the accuracy of this approximation is not high, in practice, using our safety factor (§4.2) was enough to produce satisfactory results in our scenario.

4.2 Optimizing Shape Parameters

In most mechanical fastener designs, there is a trade-off between the volume (material consumption) and the functionality (grip strength) of the fastener. To balance such a trade-off, our method optimizes the parameters of mechanical fasteners so that the user can reduce the consumption of unnecessary materials while ensuring that the fastener satisfies the functional requirement.

The functional requirement, $G_{\text{target}} = s \times m$, is the required grip strength, which depends on the mass of the object, m, and the safety factor s, set by the user. (We use the default value of $s = 5$ unless otherwise stated.) The optimization guarantees that the generated fastener will be rigid (will not move) when applying disturbance force up to G_{target}. Hence, the optimization is formulated as minimizing a volume objective, with a constraint that the grip strength will be larger than G_{target}. This can be written as:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & V(\mathbf{x}) \\ \text{subject to} \quad & G(\mathbf{x}) \geq G_{\text{target}}, \\ & C_1(\mathbf{x}), \ldots, C_k(\mathbf{x}), \\ & 0 \leq x_i \leq 1 \ \ (i = 1, \ldots, n), \end{aligned} \tag{1}$$

where $\mathbf{x} \in [0,1]^n$ are the parameters that should be optimized, which are manually normalized so that they fit within the valid range, $V(\mathbf{x})$ is a function that provides the volume of the design, $G(\mathbf{x})$ is the function that provides the estimated grip strength for a given parameter setting $\mathbf{x}$ described above, and $C_1(\mathbf{x}), \ldots, C_k(\mathbf{x})$ are any additional constraints, such as minimum thickness.

Because we use CSG representations for the fastener designs, exact computation of the volume $V(\mathbf{x})$ is time consuming. To efficiently compute the volume during optimization, we approximate it again by using an RBF data-interpolation approach; as preprocessing, we generate many sample designs with various parameters $\mathbf{x}$, compute their exact volumes, and define the function $V(\mathbf{x})$ for the whole parametric space using interpolation. Other approaches, including analytical approaches, are also possible.

The optimization itself is solved by COBYLA (gradient-free optimization) algorithm [Powell 1998] from the NLopt library. To generate a set of fastener designs, and to avoid getting stuck in local minima, we generate 20 random initial solutions for the optimization, and use the best results, which usually takes less than 1 second to complete. Fig. 6 shows example results of the snap pipe-clamp for various target grip strength. Please see the accompanying video for more results.

5 Holders for Free-Form Objects

If the attachment area of an object does not have a standard shape, we categorize the object as a *free-form* object. For such an object, we use a geometry-based approach to generate a holder whose contact points prevent the object from moving in any direction. The user can optionally specify the ungrowable region and the free motion direction. Similar to the mechanical fasteners, we aim to generate for the user *many* different fabricable free-form holders for a given object by changing internal parameters and using different strategies.

The overview of the approach is shown in Fig. 7. The input is the mesh of the free-form object. We choose a starting point on the mesh and apply *region growing* using several priority biases (§5.1). As the subregion[1] grows, the contact area increases, making the subregion more and more capable of geometrically holding the

[1] We use the term *subregion* to emphasize that this region does not cover the whole object.

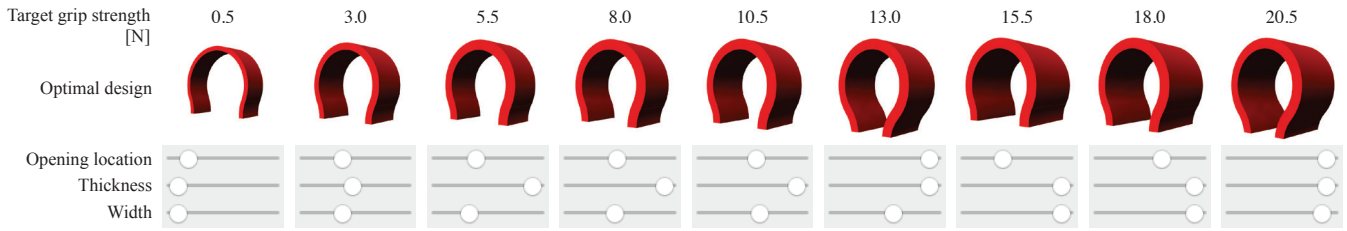


Figure 6: *Example results of our optimization while increasing the target grip strength (from left to right) for the snap pipe clamp. This clamp has 4 design parameters, one of which defines the target diameter and is fixed in the example. The other 3 parameters are optimized so that the volume is minimized while the target grip strength is achieved.*

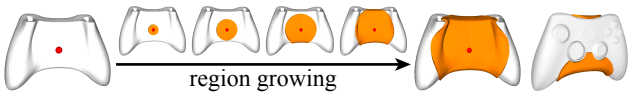


Figure 7: *Generating a free-form holder (orange) by growing the subregion from a seed point (red) until the hold-ability criteria is achieved.*

object. Region growing terminates when the *holdability criterion* is satisfied (§5.2). The result is a set of triangles that provide contact points that hold the object rigidly, which is then converted into a 3D-printable mesh by thickening. In this work, we do not consider elastic forces and friction but rather assume that everything is rigid and rely on geometric hold due to contact. Because there are many possible subregions that can achieve holdability, and there is no clear *optimal* solution, we generate many candidate designs and allow the user to pick the most suitable one.

5.1 Region Growing

Region growing is governed by three principal characteristics, in addition to the stopping criteria §5.2: (1) what target shape to fit, (2) where to start (what seed triangle to use), and (3) what growing priority to use.

By default, the target shape is the object's mesh itself. However, to create more alternative designs, we also use the convex hull of the shape as an additional target shape for the creation of free-form holders. Other targets such as the lower envelope are also possible. In addition, the user can specify certain regions on the mesh, such as the screen of a mobile phone, that should *not* be covered by the holder. The triangles in these regions are marked as *ungrowable* and are not inserted to the queue, making the region growing process skip over these triangles.

The starting point of the region-growing process has a large effect on the generated holder shape. For aesthetic reasons we use symmetry cues to generate these starting points. We detect the global reflective symmetry planes of the object ([Mitra et al. 2006; Podolak et al. 2006]), and choose the seed points by uniformly sampling from these planes. If no symmetry is found, we generate seeds randomly.

The priority used in the queue also has a significant effect on the final shape of the holder, as it biases the growing direction of the subregion by determining which triangles will be added next. The basic priority used for region growing is the geodesic distance of the triangle t in question to the seed triangle s. We use the weighted sum of the geodesic distance from the seed and two additional terms as our priority:

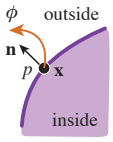
$$w_1 D_{\text{geod}}(t,s) + w_2 \min_k D_{\text{symm}}(t,P_k) + w_3 D_{\text{norm}}(n(t),N) \quad (2)$$

where $D_{\text{geod}}(t,s)$ is the geodesic distance between the barycenters of triangle t and triangle s, $D_{\text{symm}}(t,P_k)$ is the distance between the barycenter of t to the k^{th} symmetry plane P_k, and $D_{\text{norm}}(n(t),N)$ is the angle between the normal $n(t)$ of triangle t and a global direction vector N. The first term in is used simply to grow at a constant speed on the target shape from the seed. The second term is used to encourage growth along the symmetry planes, which were computed previously for finding the seeds. Note that we can use a negative w_2 weight to grow in directions perpendicular to the symmetry planes. The third term is used to bias the growth to cover certain mesh regions. For example, by setting N to the gravity direction, we can increase the coverage of the holder in the bottom part of the object (*e.g.,* for cup holders). Fig. 12 shows some results.

5.2 Holdability Criterion

To determine when to terminate the region-growing process, we use the *holdability* measure. Roughly speaking, holdability indicates how well the contact area of the subregion prevents the object from moving. Each time we add a triangle, we recompute the holdability measure and stop growing if it is above a certain threshold.

Our approach combines and extends two related concepts: *form closure*[2] from robotics [Bicchi and Kumar 2000] and *slippage analysis* from geometry processing [Gelfand et al. 2003; Gelfand and Guibas 2004]. Like slippage analysis, our approach deals with the geometry, or the triangle mesh, of the object, and thus handles a large number of contact points compared to grasp analysis in robotics. However, unlike slippage analysis, our approach deals with unilateral (inequality) contact constraints, which are handled by grasp analysis. Proper handling of unilateral constraints is critical because we need to differentiate between penetration and separation. For example, slippage analysis would mark all of the cases in Fig. 8 as rigid, since any motion of the object causes the surface of the object to penetrate into or separate from the surface of the holder. Our approach correctly identifies the rigid and non-rigid cases because it can differentiate between penetration and separation.



To define holdability, we start with the 6-dimensional rigid motion (or twist), ϕ. Let p be a surface point with position $\mathbf{x}$ and normal $\mathbf{n}$. Given p and ϕ, we can compute how much the surface point p moves along its *outward* normal. This value measures the amount of blockage experienced by the point p when the object is moved infinitesimally by ϕ. We call this the *contact blockage*, denoted $b(p,\phi)$. If the point moves away from its outward normal, then there is no blockage, and so b always takes on a non-negative value. (See §B.1 for the derivation of b.)

As the subregion grows, the number of contact points increases. For

[2] "A condition of complete restraint in which the grasped body can resist any external disturbance wrench, irrespective of the magnitude of the contact forces."

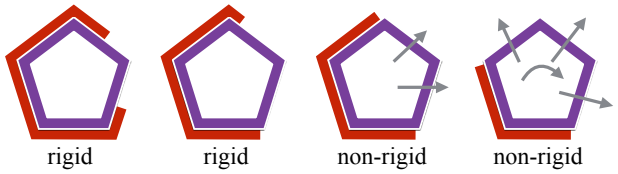


Figure 8: *2D schematic examples of our rigid and non-rigid cases. If there is no free-motion, we consider it is rigid; otherwise, it is non-rigid. Purple and red parts correspond to the target meshes and generated holder meshes respectively. Gray arrows correspond to free-motions.*

each triangle, we use the barycenter as the contact point, assuming that the input mesh is well-conditioned. Let $\mathcal{T}$ be the set of triangles of the current subregion. Given $\mathcal{T}$ and ϕ, We define a non-negative valued *subregion blockage* as a sum of contact blockages:

$$B(\mathcal{T}, \phi) = \sum_i b(p_i, \phi), \tag{3}$$

where p_i is the barycenter of the i^{th} triangle. Intuitively, B represents the amount of blockage the holder applies to the object when the object tries to move in the direction ϕ. If $B(\mathcal{T}, \phi) = 0$, then none of the triangles in the subregion blocks ϕ, and thus ϕ is a *free-motion*. In this case, the object can locally move away from the holder defined by $\mathcal{T}$ without any collisions. If $B(\mathcal{T}, \phi) > 0$, then there is at least one triangle in $\mathcal{T}$ that blocks the object from moving in the ϕ direction.

We define the holdability of the subregion $\mathcal{T}$ as the minimum subregion blockage with regard to all the possible motions.

$$H(\mathcal{T}) = \begin{cases} \underset{\phi}{\text{minimize}} & B(\mathcal{T}, \phi), \\ \text{subject to} & \|\phi\| = 1. \end{cases} \tag{4}$$

We treat ϕ as a 6D direction, and so we restrict the search to the unit hypersphere $\|\phi\| = 1$. (We can also use any small positive number instead of 1.) We use hyper-spherical coordinates to parameterize ϕ, giving us an unconstrained minimization problem with 5 degrees of freedom that allows us to deal with the constraint $\|\phi\| = 1$ implicitly. The holdability criterion is met when $H(\mathcal{T}) > 0$. In other words, we achieve holdability when $B(\mathcal{T}, \phi) > 0$ for any ϕ.

We define the *normalized* holdability measure as

$$\tilde{H}(\mathcal{T}) = \frac{H(\mathcal{T})}{H(\mathcal{T}_{\text{whole mesh}})}, \tag{5}$$

so that $0 \leq \tilde{H} \leq 1$. This normalization allows us to use the same threshold to get reasonable results for all our examples. If $\tilde{H} \approx 0$ the subregion holds very lightly, and if $\tilde{H} \approx 1$, the subregion holds very rigidly. To balance the scaling effect of the rotations vs. translations, we regularize (scale and translate) the mesh into the unit bounding box before evaluating this function. This has the effect of roughly equalizing the maximum torque to the unit translational force [Gelfand et al. 2003]. To solve this minimization problem (Eqs. 4-5), we use COBYLA, a gradient-free method [Powell 1998]. We run this optimization every time we add a new triangle and terminate the region-growing process when $\tilde{H}(\mathcal{T})$ becomes greater than a threshold. In our implementation, we set this threshold to 0.1. The region-growing process finishes in a few seconds for a smaller mesh and around 15 seconds for a large mesh.

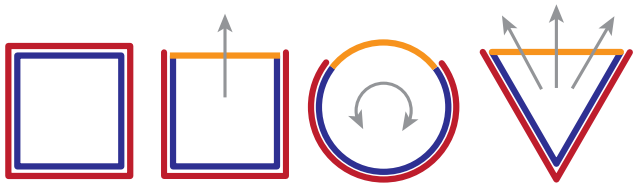


Figure 9: *Concept of intrinsic free motions. These 2D schematic examples show growable regions (blue), constrained regions (orange), "maximum" holders (red), and intrinsic free motions (gray).*

5.3 Free Motions

In some cases, the holdability measure is zero even if all triangles of the mesh are included in the triangle set $\mathcal{T}$. For example, for objects with primitive shapes such as a sphere or a cylinder, there remains a rotational motion that cannot be blocked by the holder composed of all the triangles of the object mesh. We call these unblockable motions the *intrinsic free motions* of the object. Before starting the region growing process, we analyze the input mesh to find these intrinsic free motions, and then we ignore these motions for the computation of the holdability measure to determine when we stop the region-growing process.

The algorithm for computing the set $\mathcal{F} = \{\phi_i^{\text{free}}\}$ of intrinsic free motions of an object is given in Alg. 1 (listed in §B.2). We run this algorithm on the whole input mesh as a preprocessing step before we start the region-growing process.

Some examples of intrinsic free motions are given in Fig. 9. (The orange regions are marked as *ungrowable* by the user.) Let us first consider the box example with one free intrinsic motion. Let ϕ^{free} be this free motion. When we solve for ϕ in Eq. 5, we want ϕ to not point in the same direction as ϕ^{free}. The can be expressed as

$$\phi \cdot \phi^{\text{free}} \leq \alpha. \tag{6}$$

This constraint forces the solution, ϕ, to not point within a cone of directions centered around ϕ^{free}. The parameter α controls the size of this cone. For the box example in Fig. 9, since there is a single free direction, $\alpha = 1$ would work well, preventing ϕ from pointing exactly in the direction of ϕ^{free}. However, for the triangular object in Fig. 9, there is a cone of intrinsic free motions, indicated by the three gray arrows. If we know exactly what this cone is, then we can set the correct value of α, and a single ϕ^{free} would be sufficient to fully cover the free motion directions. However, since we do not know the extent of these cones for an arbitrary mesh, we set $\alpha < 1$ and sample the cone of free directions. If α becomes closer to 1 then the approximation becomes more accurate and we will obtain more solutions at the expense of requiring more ϕ_i^{free}. For the triangular shape in Fig. 9, we have $i = 3$ with each ϕ_i^{free} covering roughly a third of the total cone. In our implementation, we use $\alpha = 0.5$ for all the examples.

In some cases, the user may manually specify additional free motion directions, which we call *extrinsic free motions*. For example, the user may specify that the holder should not block the object from moving vertically up, so that the user can insert and remove the object freely in that direction. (In this case, $\phi^{\text{free}} = (0\,0\,0\,0\,0\,1)^T$, assuming gravity points in the $-z$ direction.) We add these additional motion directions to the set $\mathcal{F}$ of intrinsic free motions computed by Alg. 1.

To incorporate $\mathcal{F}$ into our pipeline, we slightly modify both the region growing process and the holdability criterion. First, when growing, we do not add a triangle that blocks any of the specified

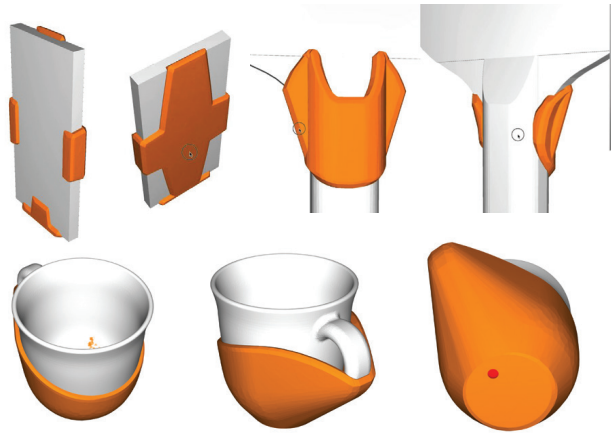


Figure 10: *Leave-one-motion examples, where the up direction is specified as an extrinsic free motion. For the bottom example, the convex hull is used for region growing.*

Figure 11: *A failure example of leave-one-motion (up direction is specified here). The bottom part of this mug is slightly wider than its mouth, and thus the outward normal of almost all of the triangles of the convex hull face upward. The region growing process stops before achieving the stop criteria because no triangle can be added without blocking the up motion (thus there are many free-motions left).*

free motions in $\mathcal{F}$. In other words, we only add triangles that satisfy the following criteria.

$$b(p, \phi_i^{\text{free}}) < 0, \quad \forall \phi_i^{\text{free}} \in \mathcal{F}, \tag{7}$$

where c is the contact penetration function from §5.2, and p is the contact point on a triangle. Second, we modify the computation of $H(\mathcal{T})$ in Eq. 4 by adding additional constraints

$$\phi \cdot \phi_i^{\text{free}} < \alpha, \quad \forall \phi_i^{\text{free}} \in \mathcal{F}. \tag{8}$$

These constraints keep ϕ away from all the intrinsic and extrinsic free motions.

Fig. 10 shows some examples of free-form holders, where the *up* direction is specified as an extrinsic free-motion. Fig. 11 shows a failure case, where region growing terminates before achieving the stopping criterion because no more triangles can be added due to the condition Eq. 7.

Splitting and Attaching Snapfits When the user does not specify any (extrinsic) free-motion, we split the shell into multiple parts so that we can physically attach/detach the shell to/from the target object without any intersection. We use the symmetry planes of the target objects if available or ask the user to specify these cutting planes. We then add snapfit mechanisms (see the inset image) for each split part to ensure that the parts can snap to each other. In our implementation we use only translational snapfits, but it is possible to attach rotational snapfits (*e.g.,* locking hinge) when a rotational motion path is available. For more details, see Appendix §B.3.

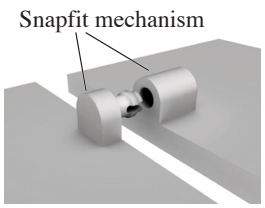


Figure 13: *Mobile phone holder with a suction cup attaching to a glass plane. The top direction is specified as a free motion. Note that the whole thing including suction cup is printed.*

Figure 14: *Pingpong paddle connected to the desk leg by using snap pipe clamp. The top direction w.r.t. the paddle is specified as a free motion.*

To summarize, the generation of the free-form holder designs is composed of the following steps:

- Analyze the input mesh to find *intrinsic-free* motions (§5.3).
- Generate a shell that can hold the target object (§5.1 & §5.2).
- Optionally split the shell into multiple parts (§B.3).

Fig. 12 shows generated various holders for some free-form shapes. We use random combinations of region-growing weights (Eq. 2) $w_1 \in \{1.0\}, w_2 \in \{-1.0, 0.0, 3.0, 6.0\}, w_3 \in \{0.0, 0.5\}$ and also start with several different seed points (§5.1) to generate many different designs. Selecting a small number of diverse, distinguishable options from the possible design space is an important future work, but is out of the scope of this research. For this purpose, there are many previous methods available [Marks et al. 1997; Won et al. 2014].

6 Application Scenarios

We create many connectors that could enhance our daily lives by using our AutoConnect. In Figures 13-16, we connect a structured object and a free-form object, and specified a free-motion when generating connectors. For many uses cases, it is desirable to specify a free-motion, since it allows the user to easily attach and detach the object.

Sometimes, the user may want to connect two objects very firmly. Fig. 17 is an example where strong external forces can be applied to the target objects when using this connector. For this scenario, we do not specify any free-motion and specify a large safety factor. Our method generates design options by using the strap pipe clamp, which has a very strong grip strength compared to the other mechanical fasteners for cylinders.

Our method can also be applied when one side of the connected objects is directly 3D-printed. Fig. 18 shows such cases; first, we attach a 3D-printed dragon's head to a high-heel shoe by using

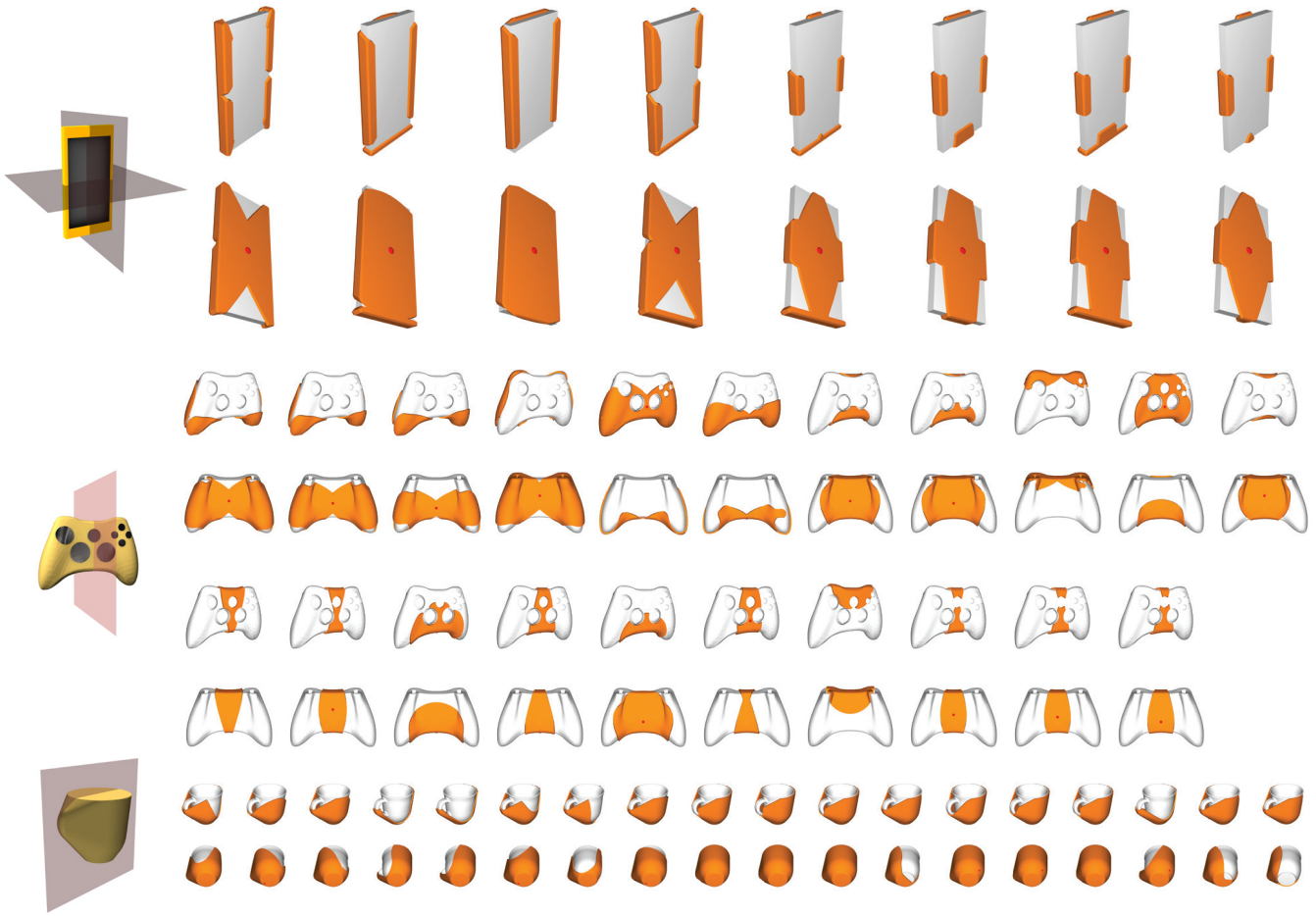


Figure 12: *Variations of generated free-form holders. The red points are the seed points for region growing. Top row: a rectangular mobile phone, where the display part is specified as a contrained (non-growable) region. The top direction is specified as a free-direction. As there are two reflective planes, the center of back side is always selected as a seed point. Middle row: game controller. Bottom row: a mug, where the convex hull is used for region growing and the top direction is specified as a free-direction.*

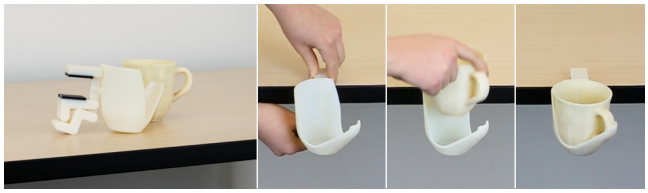


Figure 15: *Mug holder at the desk edge. The convex hull is used for region growing, and the top direction is specified as a free motion.*

Figure 16: *A connector that allows us to put a game controller to a chair arm.*

the free-form holder. Second, we attach a 3D-printable object to a structured object by using one of the mechanical fasteners. We can also connect structured-structured combinations; we show such scenarios in Fig. 19.

7 Discussion

In this paper, we presented AutoConnect, a method to automatically design 3D-printable connectors that are tailored to the two input geometries and user's specifications. Our method classifies the target geometries into two categories, structured objects and free-form objects, and applies different strategies to generate holders/grippers for each of them. For structured objects, we perform force-based optimization to predefined mechanical fasteners. We use a data-driven approach to estimate the grip-strength based on real measurements. For free-form objects, we perform procedural region growing with geometry-based criterion of holdability. We showed various possible scenarios for the usage of AutoConnect, which include mounting various objects to the desk or chair (phone, ping-pong paddle, mug), creating phones mounts to the bike or the dashboard of the car, and creating decorative objects such as wall mounted bunny and a shoe dragon.

Our final goal is that anyone would be able to pick two objects and get a connecting design for them. We presented a first step towards this goal, but there are many limitations and possibilities for future

Figure 17: *Mobile phone can be attached to your mountain bike in your favorite configuration. This connector consists of a strong strap pipe clamp and a free-form holder without free motions.*

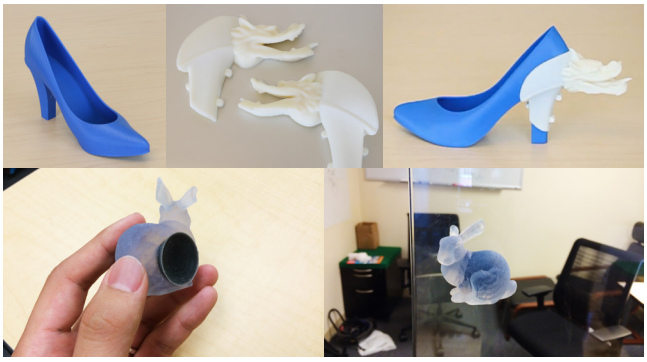


Figure 18: *One of the connected objects can be a directly-printable object. (Top) Attaching printed Stanford dragon's head to a free-form high-heel. (Bottom) Attaching a printed bunny to a flat glass plane.*

work. First, we need correct 3D virtual models for the two objects to create a functional connector. We have demonstrated the use of Kinect-fusion to scan the car dashboard to produce a connector, but this process can be time consuming. It would be nice if one could use photographs or videos or even simple scanning devices.

Currently we use only six models in our parametric fasteners database. This limits variation of possible designs, but in the future it is easy to add more parametric fasteners to build a larger database, or even use a web-based service, where dedicated users can upload their fastener designs. To add such a design to the database it needs to be parametric, and the grip-strength needs to be measured and stored for various parameter values. The lookup and interpolation-based approach during optimization of the mechanical-fasteners sidesteps the complexity of real-time simulation with real data and measurement. On the other hand, using explicit simulation does not require pre-processing. In the future, it would be interesting to combine the two approaches as they need not be mutually exclusive.

As most fasteners are easy to detach (*e.g.,* a toggle clamp has a handle for detaching), we do not consider the difficulty of detaching in our optimization. However, since some fasteners, such as the snap pipe clamp, must be difficult to move but easy to detach, it would be interesting to find a way to include such conflicting requirements in our optimization. We also assume currently that both the target objects and the printed connectors are rigid and infinitely strong. Considering soft deformation and structural strength is an important future work.

Figure 19: *Our method can be applied even when both target objects are structured. (Left) Attaching the cylindrical light to the bike handle. (Center) A beverage can holder connected to the belt. For this example, we assume that the belt is rectangular-prism-shaped. (Right) We can also attach a can to the desk leg.*

For free-form holders, when the target shape is highly complex or includes many concave parts, such as in the Stanford dragon's mouth or legs or the armadillo model, our method can generate a complex holder that is difficult to attach/detach because of blocking. In this case, our split-and-verify algorithm (§B.3) often fails, or needs many splitting planes. In most practical scenarios, this is not a problem, as many artificial artifacts have nearly convex shapes, and can easily achieve attachability/detachability using a single splitting. In addition, the ability to use the convex hull of the target shape can provide some solution to this problem.

Although we try to create many alternative designs for the user to choose from, some of them can be very similar. A measure of similarity between the designs could be used to provide more distinct alternatives and assist the user. Lastly, it would be interesting to allow more user input either in the form of more declarative constraints such as where to position the cutting planes, where to place the connecting rod, or how many parts to divide the connector, or in the form of a fully interactive modeling tool.

We believe that the problems of automatic creation of geometry and customization of everyday objects for 3D printing are challenging but are indeed very useful. We hope that our work, which only starts to tackle these problems, will inspire future research as well.

References

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

ANJYO, K., LEWIS, J. P., AND PIGHIN, F. 2014. Scattered data interpolation for computer graphics. In *ACM SIGGRAPH 2014 Courses*, ACM, New York, NY, USA, SIGGRAPH '14, 27:1–27:69.

AUTODESK. Autocad. http://www.autodesk.com/products/autodesk-autocad.

BÄCHER, M., BICKEL, B., JAMES, D. L., AND PFISTER, H. 2012. Fabricating articulated characters from skinned meshes. *ACM Trans. Graph. 31*, 4 (July), 47:1–47:9.

BÄCHER, M., WHITING, E., BICKEL, B., AND SORKINE-HORNUNG, O. 2014. Spin-it: Optimizing moment of inertia for spinnable objects. *ACM Trans. Graph. 33*, 4 (July), 96:1–96:10.

BICCHI, A., AND KUMAR, V. 2000. Robotic grasping and contact: a review. In *Proceedings of IEEE International Conference on Robotics and Automation*, 348–353.

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. In *ACM SIGGRAPH 2010 Papers*, ACM, New York, NY, USA, SIGGRAPH '10, 63:1–63:10.

CALÌ, J., CALIAN, D. A., AMATI, C., KLEINBERGER, R., STEED, A., KAUTZ, J., AND WEYRICH, T. 2012. 3d-printing of non-

assembly, articulated models. *ACM Trans. Graph. 31*, 6 (Nov.), 130:1–130:8.

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

CHEN, X., ZHENG, C., XU, W., AND ZHOU, K. 2014. An asymptotic numerical method for inverse elastic shape design. *ACM Trans. Graph. 33*, 4 (July), 95:1–95:11.

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

DASSAULT SYSTÈMES. Solidworks. http://www.solidworks.com/.

GELFAND, N., AND GUIBAS, L. J. 2004. Shape segmentation using local slippage analysis. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, ACM, New York, NY, USA, SGP '04, 214–223.

GELFAND, N., IKEMOTO, L., RUSINKIEWICZ, S., AND LEVOY, M. 2003. Geometrically stable sampling for the icp algorithm. In *3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings. Fourth International Conference on*, 260–267.

HIRUKAWA, H., MATSUI, T., AND TAKASE, K. 1994. Automatic determination of possible velocity and applicable force of frictionless objects in contact from a geometric model. *Robotics and Automation, IEEE Transactions on 10*, 3 (Jun), 309–322.

KOO, B., LI, W., YAO, J., AGRAWALA, M., AND MITRA, N. J. 2014. Creating works-like prototypes of mechanical objects. *ACM Trans. Graph. 33*, 6 (Nov.), 217:1–217:9.

LI, H., ALHASHIM, I., ZHANG, H., SHAMIR, A., AND COHEN-OR, D. 2012. Stackabilization. *ACM Trans. Graph. 31*, 6 (Nov.), 158:1–158:9.

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. *ACM Trans. Graph. 33*, 4 (July), 97:1–97:10.

MARKS, J., ANDALMAN, B., BEARDSLEY, P. A., FREEMAN, W., GIBSON, S., HODGINS, J., KANG, T., MIRTICH, B., PFISTER, H., RUML, W., RYALL, K., SEIMS, J., AND SHIEBER, S. 1997. Design galleries: A general approach to setting parameters for computer graphics and animation. ACM, SIGGRAPH '97, 389–400.

MITRA, N. J., GUIBAS, L. J., AND PAULY, M. 2006. Partial and approximate symmetry detection for 3d geometry. *ACM Trans. Graph. 25*, 3 (July), 560–568.

PODOLAK, J., SHILANE, P., GOLOVINSKIY, A., RUSINKIEWICZ, S., AND FUNKHOUSER, T. 2006. A planar-reflective symmetry transform for 3d shapes. *ACM Trans. Graph. 25*, 3 (July), 549–559.

POWELL, M. J. D. 1998. Direct search algorithms for optimization calculations. *Acta Numerica 7* (1), 287–336.

PRÉVOST, R., WHITING, E., LEFEBVRE, S., AND SORKINE-HORNUNG, O. 2013. Make it stand: Balancing shapes for 3d fabrication. *ACM Trans. Graph. 32*, 4 (July), 81:1–81:10.

SAUL, G., LAU, M., MITANI, J., AND IGARASHI, T. 2011. Sketchchair: An all-in-one chair design system for end users. In *Proceedings of the Fifth International Conference on Tangible, Embedded, and Embodied Interaction*, ACM, New York, NY, USA, TEI '11, 73–80.

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.

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

THINGIVERSE. Thingiverse. `http://www.thingiverse.com/`.

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

UMETANI, N., IGARASHI, T., AND MITRA, N. J. 2012. Guided exploration of physically valid shapes for furniture design. *ACM Trans. Graph. 31*, 4 (July), 86:1–86:11.

UMETANI, N., KOYAMA, Y., SCHMIDT, R., AND IGARASHI, T. 2014. Pteromys: Interactive design and optimization of free-formed free-flight model airplanes. *ACM Trans. Graph. 33*, 4 (July), 65:1–65:10.

WANG, W., WANG, T. Y., YANG, Z., LIU, L., TONG, X., TONG, W., DENG, J., CHEN, F., AND LIU, X. 2013. Cost-effective printing of 3d objects with skin-frame structures. *ACM Trans. Graph. 32*, 6 (Nov.), 177:1–177:10.

WILSON, R. H. 1992. *On Geometric Assembly Planning*. PhD thesis, Stanford, CA, USA. UMI Order No. GAX92-21686.

WON, J., LEE, K., O'SULLIVAN, C., HODGINS, J. K., AND LEE, J. 2014. Generating and ranking diverse multi-character interactions. *ACM Trans. Graph. 33*, 6 (Nov.), 219:1–219:12.

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

ZHU, L., XU, W., SNYDER, J., LIU, Y., WANG, G., AND GUO, B. 2012. Motion-guided mechanical toy modeling. *ACM Trans. Graph. 31*, 6 (Nov.), 127:1–127:10.

A Details of Measurement of Grip Strength

Number of Data Points For the mechanical fastener that has n design parameters, we consider $1+2n+2^n$ parameter sets consisting of 1 default parameter set (*i.e.,* $(0.5, 0.5, \ldots, 0.5)$), $2n$ one-parameter-minimized/maximized parameter sets (*e.g.,* $(1.0, 0.5, \ldots, 0.5)$), and 2^n every-parameter-minimized/maximized parameter sets (*e.g.,* $(1.0, 0.0, \ldots, 0.0)$). For example, for the snap pipe clamp, which has 4 parameters, we print $1 + 2 \times 4 + 2^4 = 25$ clamps and measure each clamp's grip strength.

The strap pipe clamp is an exception. This clamp has almost infinitely strong grip strength in a realistic scenario (we applied over 50 kg-weight force several times, but it never moved). Thus, we do not gather any data for this strap pipe clamp. Note that the strap pipe clamp is parameterized by only one *geometrically-deterministic* parameter (the diameter of the target pipe) and thus there is no need to optimize the design according to the grip strength.

For the toggle clamp, which has 4 parameters, there are 7 invalid designs (*e.g.,* some parts are intersected, or it breaks before applying enough force to measure) out of 25 designs. Thus we interpolate the grip strength using the rest of 18 data points. Similarly, for the cam

clamp, which has 3 parameters and thus 15 clamps are printed, we failed to measure 3 clamps, so we use other 12 data points for the interpolation.

Measurement Settings For the fasteners for cylinder, we use stainless polished pipes with the diameter of 20.0mm, 30.0mm, 40.0mm. For the flat-plane fastener (*i.e.,* suction cup), we use a glass plate. For the rectangular-prism fastener (*i.e.,* box clamp) and the box-edge fastener (*i.e.,* toggle clamp), we use printed prisms and printed boxes.

Every fasteners and standard shapes for measurements are printed with the material called *Endur*, which is Simulated Polypropylene, except for suction cups and contact areas in toggle clamps. For those exceptions, we use the meterial called *Tango*, which is Rubber-like material. All fasteners are printed by using a commercial inkjet 3D printer called *Objet Connex 500*.

For measuring force, we use a force meter whose resolution is 0.040 kg-weight. When printing fasteners for measurement, we always print a small hook at the point defined for each fastener, then the force meter is attached to this small hook. Starting from pulling small force, we increase the force gradually, and when the mechanical fastener moves, we record the maximum force that the force meter displays during this process[3]. It takes 2 to 5 hours to obtain data for each fastener type.

Validation We performed a simple validation for the snap pipe clamp, which has 4 parameters. We printed an additional set of snap pipe clamps (testing set) and measured data for them. The size of learning data set is 25, and we choose the size of testing set as 12 in total (4 random parameter sets for each diameter). Let $\mathbf{x}_1, \ldots, \mathbf{x}_{12}$ be the testing parameters, and $G_1, \ldots, G_{12}$ be the corresponding measured data. The root mean square (RMS) error of our estimation is

$$\mathrm{RMS} = \sqrt{\frac{\sum_{i=1}^{N} \left(G(\mathbf{x}_i) - G_i\right)^2}{N}} = 0.14[\text{kg-weight}]. \quad (9)$$

The average of 25 measured grip strength is 0.57 kg-weight and the maximum value is 3.54 kg-weight. We consider this estimation is not very accurate; however, we use this estimation with the safety factor, which is typically specified by rough values (*e.g.,* $s = 5$ or $s = 10$). Thus, this accuracy could be enough for our context. Note that we can add data points when more accuracy is necessary.

B Free form details

B.1 Contact displacement function

To define holdability, we require the 6-dimensional rigid motion (or twist)

$$\phi = \begin{pmatrix} \omega \\ \nu \end{pmatrix} \in \mathbb{R}^6, \quad (10)$$

where $\omega \in \mathbb{R}^3$ is the rotational velocity, and $\nu \in \mathbb{R}^3$ is the translational velocity expressed in body local coordinates. [4] Let p be a surface point with position $\mathbf{x}$ and normal $\mathbf{n}$. Like ϕ, both $\mathbf{x}$ and $\mathbf{n}$ are expressed in body local coordinates. Given p and ϕ, we can evaluate how much the surface point p moves along its normal when the object is moved infinitesimally by ϕ. We call this the contact blockage of p given ϕ:

$$b(p, \phi) = \max(\mathbf{n}^T \Gamma \phi, 0), \quad (11)$$

where $\Gamma = ([\mathbf{x}]^T \; I)$ is the 3×6 matrix that transforms a spatial velocity, ϕ, to a point velocity. The 3×3 skew symmetric matrix, $[\mathbf{x}]$, is the cross product matrix, such that $[\mathbf{x}]\mathbf{a} = \mathbf{x} \times \mathbf{a}$, and I is the 3×3 identity matrix. If b is positive, then p moves along its normal. Thus, this surface point p blocks the object from moving in the direction ϕ.

Algorithm 1 Computing intrinsic free motions

$\mathcal{F} \leftarrow$ empty set of free motions;
$\mathcal{C} \leftarrow$ empty set of constraints;
$\mathcal{T} \leftarrow$ all triangles from mesh;
loop
  Solve: $H \leftarrow \min_\phi B(\mathcal{T}, \phi)$ s.t. $\mathcal{C}$;
  $\phi^{\min} \leftarrow$ the optimal argument;
  if $H < \epsilon$ **then**
    $\mathcal{F} \leftarrow \mathcal{F} \cup \{\phi^{\min}\}$;
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{\phi \cdot \phi^{\min} < \alpha\}$;
  else
    return $\mathcal{F}$;
  end if
end loop

B.2 Finding Intrinsic Free Motions

We start with an empty set $\mathcal{F}$ of free motions, and incrementally build this set by iteratively solving the subregion blockage equation (Eq. 3) subject to the constraint set $\mathcal{C}$, which is also initially empty. Each new free motion we find adds the corresponding constraint to $\mathcal{C}$ of the form Eq. 6, so that motion directions close to the newly found direction are no longer considered. Once we achieve holdability, the algorithm terminates, returning a set of intrinsic free motions of the input mesh. The thresholding parameter (*e.g.,* $\epsilon = 0.01$) is necessary to be robust against noise in the input mesh and inaccuracy due to discretization. For our examples, this algorithm takes less than one second to find the intrinsic free motions.

B.3 Splitting

To verify that every part of the holder can be attached to and detached from the object, we first ensure that the local holdability of each holder part is zero, and then ensure that there is an intersection-free motion path for each part.To find such an intersection-free path, we use the information of the optimal twist motion ϕ_k that satisfies $H(\mathcal{T}_k, \phi_k) = 0$, where $\mathcal{T}_k$ is the sets of triangles for the split parts so that $\mathcal{T} = \bigcup \mathcal{T}_k$. Then, we run a dynamic rigid body simulation, where every holder part and the target object are assumed to be rigid, with virtual spring forces that pull each holder part towards their computed twist motion ϕ_k. If the global intersection check fails, the system rejects the current cut plane, and a different one is chosen either automatically (if symmetry planes are used) or by the user.

[3] For the cam clamp, we took a different approach; we measured the force not to slightly move the fastener (by slippage), but to remove the fastener completely from the cylinder (by breaking). We empirically found that scaling this data by 0.1 is a reasonable approximation of the grip strength.

[4] More precisely, $\phi \in se(3)$, the Lie algebra of SE(3), the special Euclidean group in 3 dimensions.