SpringFit: Joints and Mounts that Fabricate on Any Laser Cutter

Thijs Roumen, Jotaro Shigeyama, Julius Cosmo Romeo Rudolph, Felix Grzelka, and Patrick Baudisch
Hasso Plattner Institute at University of Potsdam
Potsdam, Germany

{firstname.lastname}@hpi.uni-potsdam.de

ABSTRACT

Joints are crucial to laser cutting as they allow making three-dimensional objects; mounts are crucial because they allow embedding technical components, such as motors. Unfortunately, mounts and joints tend to fail when trying to fabricate a model on a different laser cutter or from a different material. The reason for this lies in the way mounts and joints hold objects in place, which is by forcing them into slightly smaller openings. Such "press fit" mechanisms unfortunately are susceptible to the small changes in diameter that occur when switching to a machine that removes more or less material ("kerf"), as well as to changes in stiffness, as they occur when switching to a different material.

We present a software tool called *springFit* that resolves this problem by replacing the problematic press fit-based mounts and joints with what we call *cantilever-based* mounts and joints. A cantilever spring is simply a long thin piece of material that pushes against the object to be held. Unlike press fits, cantilever springs are robust against variations in kerf and material; they can even handle very high variations, simply by using longer springs. SpringFit converts models in the form of 2D cutting plans by replacing all contained mounts, notch joints, finger joints, and t-joints. In our technical evaluation, we used springFit to convert 14 models downloaded from the web.

Author keywords

Laser cutting; fabrication; portability; reuse

CSS Concepts

• **Human-centered computing~Interactive systems and tools**

INTRODUCTION

Laser cutting is one of the key technologies in personal fabrication [3]. Unlike 3D printing, it allows processing natural materials, such as wood, making it relevant to arts & crafts [6, 15, 37, 38]. Laser cutting also fabricates physical 3D models orders of magnitude faster than 3D printing, making it a key contender for *rapid* prototyping [4, 5].

UIST '19, October 20–23, 2019, New Orleans, LA, USA

ACM ISBN 978-1-4503-6816-2/19/10…$15.00
https://doi.org/10.1145/3332165.3347930

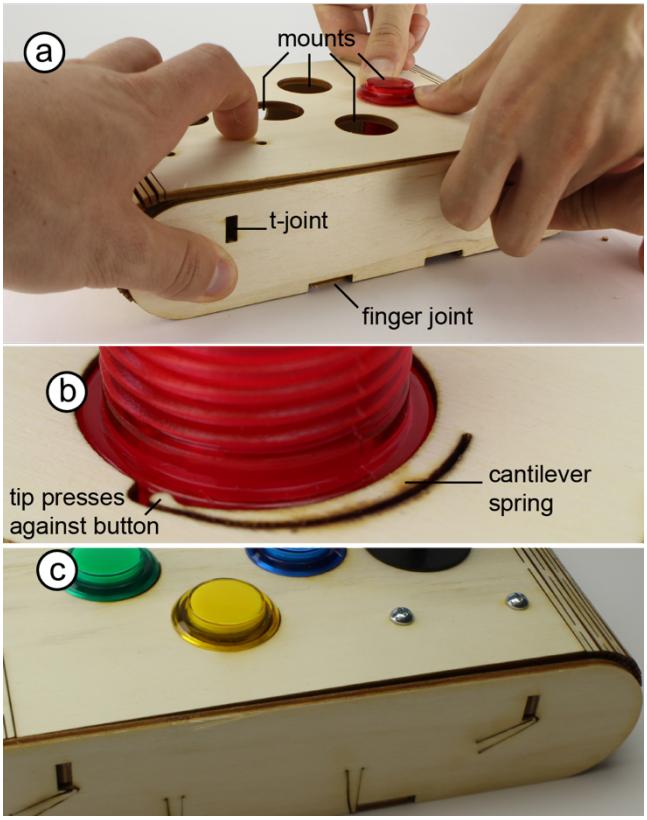

Figure 1: (a) This model downloaded from the Internet does not assemble properly, because the user's laser cutter has removed less material (aka smaller "kerf") than the machine this model was designed for. Similar issues occur as a result of *larger* kerf and when switching to a different material. (b) SpringFit tackles this by replacing traditional (press-fit-based) mounts and joints with what we call "cantilever-based" joints and mounts. (c) Processing the model using springFit before laser cutting resolves the problems and it can now be assembled properly.

While laser cutting does not support reproducing the same wealth of shapes as 3D printing, laser cutting does allow fabricating mounts and joints [24]. These affordances allow users to assemble multiple parts into larger and/or three-dimensional objects, and to embed components, such as motors or LEDs [33, 23].

Unfortunately, the resulting compound models tend to fail when shared (Figure 1a). The reason is that the mounts and joints contained in the shared 2D cutting plans (e.g., Thingiverse) are inherently specific to the single material and laser cutter they were designed for. When fabricated on

another machine or from another material, the result often does not assemble. As we discuss in more detail in the following section, the reason is that mounts and joints are susceptible to even the smallest variations in the width of the laser beam (aka *kerf*) and material properties.

Such variations, however, inevitably occur when switching to a different laser cutter or material, e.g., as part of Internet-based sharing. In addition, even the original designer of a model is affected when switching to new equipment—a step very likely to happen over time.

In this paper, we analyze the problem caused by press-fit-based mounts and joints. We then present a variation of mounts and joints that are not subject to these limitations. As illustrated by Figure 1b and c, the key idea is to replace the current mounts and joints with ones that contain what we call "cantilever-based" mounts and joints. These make the model machine- and material-independent.

We have created the model shown in Figure 1c automatically using our simple web-based software tool "springFit" (Figure 2). It takes a 2D cutting plans as input (e.g. svg), locates mounts and joints and replaces them with spring-based mounts and joints, and produces the same type of 2D cutting plans as output. SpringFit thereby makes models fabricate reliably from a range of materials and on a range of laser cutters.

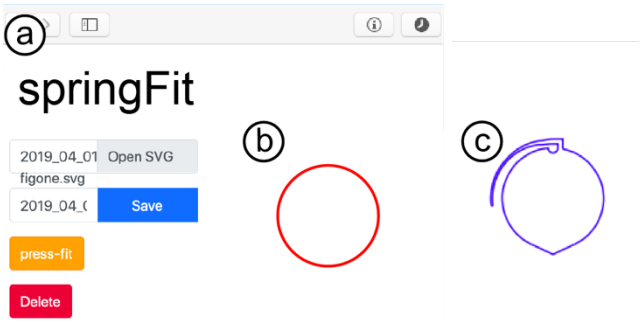

Figure 2: (a) SpringFit (b) parses 2D cutting plans, and (c) returns a cutting plan in which all press-fit based mounts and joints have been replaced with cantilever-based designs.

CHALLENGE: MOUNTS AND JOINTS NOT "PORTABLE"

We begin by taking a closer look at the underlying challenge. The problem with mounts and joints lies in the way these have traditionally been (and continue to be) implemented.

As illustrated by Figure 3, mounting a plastic arcade button requires the model designer to add a mount to the model. The commonly accepted solution is to create a hole that has the same shape as the button, but is a *tiny* bit smaller than the button. This type of mount is called a "*press fit*" (or "force fit", "interference fit", etc.). When *forcing* the button into such a press fit mount, the tightness of the hole causes the plywood to stretch a tiny bit to accommodate the button. This means that the plywood now acts as a *spring* and it is the stretching of that spring that holds the button in place.

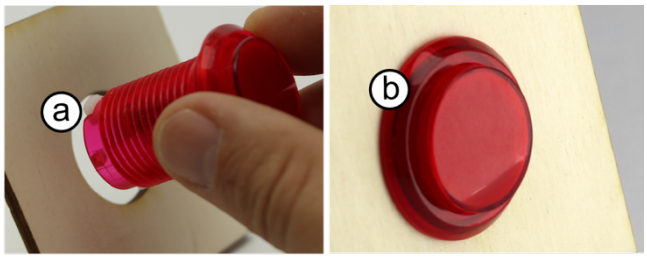

Figure 3: (a) A key design element in laser cutting is the "press-fit" where a physical object is forced into a slightly smaller opening. (b) The opening now acts as spring, securely holding the object in place.

Unfortunately, when other users try to reproduce such a model, they switch to *their* laser cutter and *their* material and this introduces variation in the size and stiffness of the mount, which commonly causes the mount to fail. As shown in Figure 4, (a) some user's lasers produce a thinner cut (aka smaller "kerf"). This makes it impossible to assemble the model, as the button does not fit into the mount anymore. (b) other users' lasers produce a wider cut (larger "kerf"). This causes the button to fall out of the mount. (c) This user tries to reproduce the model from acrylic. Acrylic is more brittle than plywood. The hole size that worked fine with plywood now causes the material to pop when the user forces the button in.

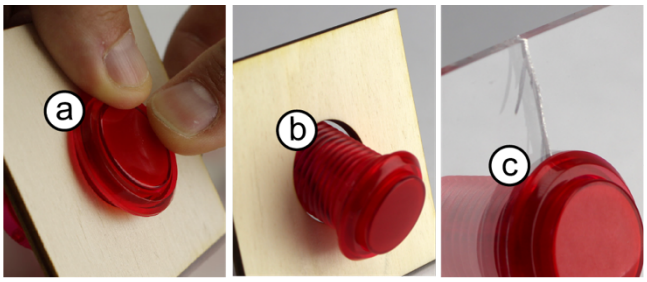

Figure 4: The carefully tuned press fit from Figure 3 fails when fabricated on different equipment. (a) On machines with smaller kerf, it cannot be inserted completely. (b) On machines with wider kerf, it is loose and falls out. (c) When cut from brittle material, it breaks the model.

In summary, compound laser cut models require mounts and joints; these are typically based on press fits; press fits fail if conditions vary; and such variations are inevitable when switching materials or machines, be it as part of sharing, when upgrading to a new laser cutter model, or even just as the result of daily wear on a single machine.

MOUNTS & JOINTS BASED ON CANTILEVER SPRINGS

To address this problem, we propose replacing press fit-based mounts and joints with a different type of mounts and joints based on *cantilever springs* [14]. These can generally be manufactured using a laser cutter as part of cutting the model.

The model shown in Figure 1b features a mount based on a cantilever spring (generated by springFit) that holds the button in place. A cantilever spring is a long and thin element that is connected to the laser cut model at one end, while the tip at the free end makes physical contact with the

object it is supposed to hold, here the button. This particular spring is curved in order to accommodate the shape of the button; cantilever springs generated by our system are otherwise straight.

As shown in Figure 5, the user mounts the button by inserting it into the mount (this works best if done at an angle, so that the button holds the cantilever spring back until fully inserted).

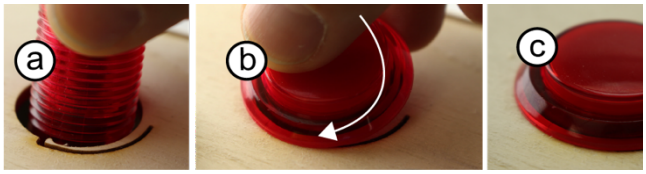

Figure 5: (a) A cantilever spring-based mount. (b) The button is best inserted by sliding it in at an angle and optionally spinning it against the direction of the spring. (c) Done.

Figure 6 shows the resulting benefit of using cantilever-based mounts: these mounts continue to work irrespective of what machine they are fabricated on and what material they are made of. In particular, they fabricate reliably on (a) a machine of small kerf, (b) a machine of wide kerf, (c) from different material. (d) They even allow inserting a slightly bigger button.

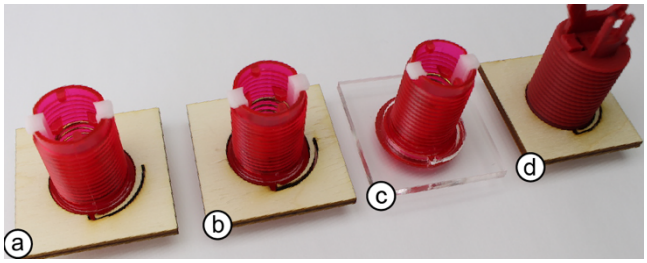

Figure 6: The use of cantilever-based mounts and joints allows one and the same models to fabricate reliably (a) on machines with small kerf, (b) with wide kerf (here simulated by eroding the model by 0.2mm), and (c) different material, and (d) even slightly different sized buttons (this one is 0.3mm bigger in diameter).

Why it works

Figure 7 illustrates why cantilever springs succeeds where press fits fail. (a) The springs formed by a press fit require the surrounding material to *compress*. Such "compression-based" springs are very *stiff*, i.e., even a small compression requires a large force. Implementing a certain desired force thus requires a *very* specific diameter, while minor changes in diameter easily result in a force of zero or a force large enough to break the model.

(b) Cantilever springs, in contrast, act by *bending* material, which makes them much less stiff. Obtaining a certain desired force can thus be achieved with a wider range of diameters. Since the cantilever *tolerates* comparably large changes in diameter, switching to a different fabrication machine or material is less of an issue.

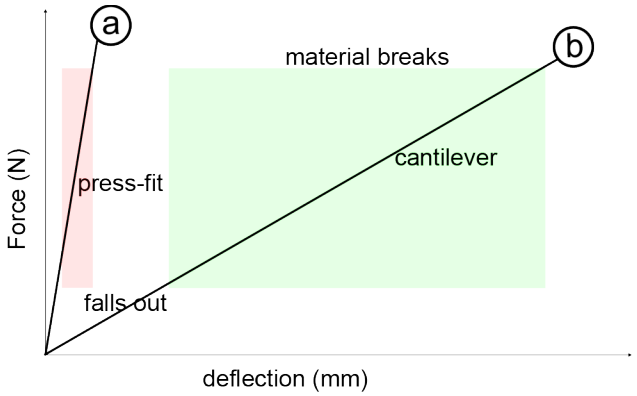

Figure 7: (a) Traditional press fit-based mounts and joints are very stiff, thus only a tiny range of "deflection" allows it to stay in the desired force range. (b) The cantilever solution affords a substantially bigger range of deflection.

Cantilever springs make it easy to tune their stiffness. As illustrated by Figure 8, we can increase (a) a spring's stiffness by a factor of 8 either (b) by doubling its diameter or (c) by cutting its length in half (as both parameter affect stiffness *cubed*).

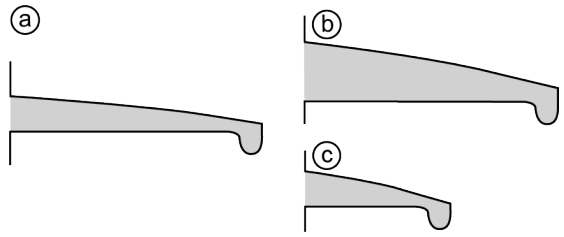

Figure 8: We can increase (a) a spring's stiffness by a factor of 8 either (b) by doubling its diameter or (c) by cutting its length in half.

As shown in Figure 9, the combination of thickness and length allows tuning the spring's desired tolerance. (a) This cantilever spring was designed to allow for typical variation in kerf (e.g. 0.1mm at 10N). (b) This longer (yet thicker) cantilever spring is as stiff as the previous spring, but accommodates 7.5x more variation of up to 1.5mm, allowing users to even swap out the button for an (up to 1.5mm) larger button, such as the one from Figure 6d.

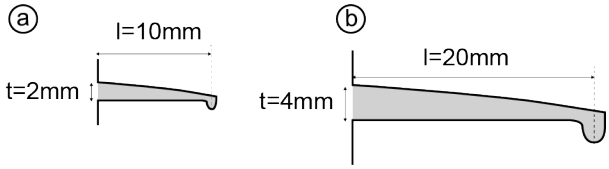

Figure 9: (a) This short and thin cantilever spring and (b) this long and thick cantilever spring are equally stiff. The latter one can deform further though, thus accommodates, for example, larger variations in kerf.

Non-round mounts

Our approach equally applies to mounts of other convex shapes. As shown in Figure 10a, springFit generally fits non-round shapes with a straight cantilever spring that applies to one corner of the object from a diagonal angle. (b) In addition, springFit produces two additional protruding points along the other sides, that form an (ideally equilateral) triangle with the spring tip. This prevents the held object from rotating out of plane.

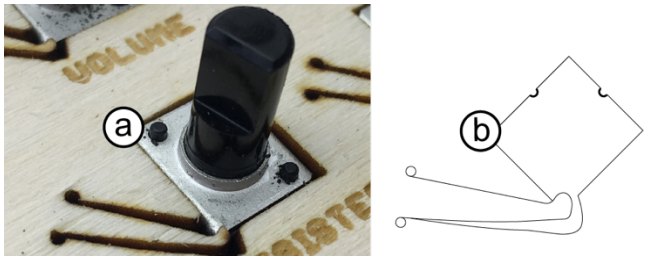

Figure 10: (a) Cantilever-based mount for a rectangular shape. The cantilever spring pushes against one of the corners of the cutout. (b) two additional protruding points along the other sides prevents the held object from rotating out of plane.

Cantilever-based notch-, finger- and T-joints

While so far we have talked only about *mounts*, we have created cantilever-based equivalents for press-fit *joints* as well. Figure 11 shows cantilever-based (a) finger joints (c) notch joint (aka cross joint) and (e) t-joints and how they are assembled. Many models combine multiple joint types, such as the model shown in Figure 1.

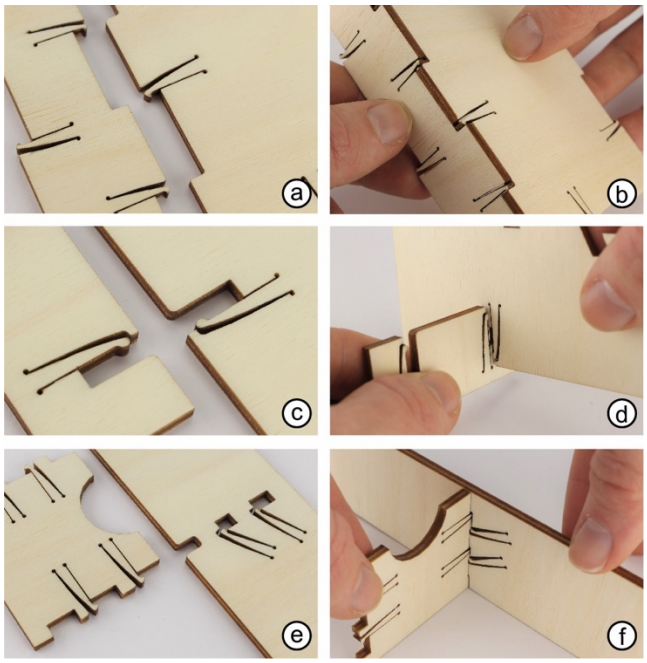

Figure 11: Cantilever spring versions of (a) finger joints and (b) how they assemble. (c,d) notch joints or cross joints and (e,f) T-joints.

To simplify assembly, finger-joints generated by springFit provide rounded corners, as shown in Figure 12.

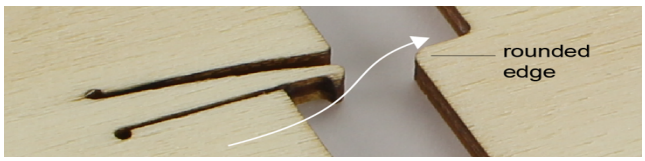

Figure 12: the rounded edge helps to push the spring smoothly when assembling.

CONTRIBUTION, BENEFITS & LIMITATION

In this paper, we (1) analyze the problem caused by mounts and joints and the implications for the overall model. (2) We present "cantilever-based mounts and joints" that are not subject to these limitations. (3) We present a software tool called springFit that locates mounts and joints in laser cut models and replaces them with mounts and joints not subject to the above limitation. It achieves this by making good initial guesses automatically, through a browser UI, users can fine tune or override these guesses.

Models processed using springFit fabricate reliably from a range of materials and on any laser cutter (in any state of wear). The resulting models use the traditional 2D cutting plan format (svg). This allows them to be stored and shared alongside any traditional model.

Our approach is subject to four limitations. (1) Some generated models come apart more easily than their press fit counterparts, as cantilever springs cannot reach the stiffness of a press fit mounted with the help of a rubber hammer. (2) Cantilever springs add additional incisions to a model., which may affect the aesthetic qualities of the model. (3) Models need to offer sufficient space for the cantilever springs (see also "Technical Evaluation"). (4) When cantilever springs cut into structurally relevant regions, they make the resulting models more prone to breaking.

RELATED WORK

Our work builds on research into parametric joint generation, embedding objects, and multi-material fabrication. Most of the related work stems from 3D printing.

Parametric models for laser cutting

One traditional way of allowing a model to fabricate on a different machine is to make the model parametric (e.g., openSCAD(openscad.org) or OnShape (onshape.com), in kerf and material. Also 3D editing systems for laser cutting that export 2D geometry are inherently parametric (e.g., Kyub [4] for finger joints, joinery [35] and CutCAD [15] for flat joints in general and FlatFitFab [12] for cross joints). With these systems users do not share the 2D cutting plans, but the 3D models. The conversion to 2D cutting plans then takes place in the local context of the downloading user, allowing that user to generate cutting plans for their local machine's kerf. In practice, people do not share those high level descriptions, rather they share SVGs as anyone can open and cut them. SpringFit, for that reason, takes SVG as input and exports the generalized version. We do not believe the sharing habits will change any time soon.

Creating mounts and embedding objects in 3D printing

In the context of 3D printing, several research systems have explored how to allow users to embed physical object into their fabricated models. *RetroFab* [21], for example, allows users to embed electronics.

Medley [8] allows users to embed objects into their 3D-printed models. Their process requires manual calibration. *Reprise* [9] follows a similar approach, but create mounts from soft 3D printed materials. The resulting solution is still material and machine-specific. AutoConnect [18] goes a step further by embedding 3D objects in 3D models, it does so by making holders that surround the object in 3D printed geometry. Maker's Marks [23] does a great job at prototyping the mounting of electronics and then creates the required modifications to a model so it can be directly 3D printed.

Kim et al. [17] identified the challenge of creating precise mounts. Their system *FitMaker* inserts mechanisms (i.e. ball joints, sliders) in the models that allow users to tune the dimensions after fabrication.

Enclosed [33] allows users to make laser cut casings to then fit the electronic components in. Lau et al. [19] generate connectors and parts for 3D models of furniture.

Material aware fabrication in 3D printing

Ulu et al. [30] used material properties in 3D printed objects to specifically design their compliant behavior. They make snap-fit holders for objects using given input geometry. Foundry [31] is an approach to describe the variation of materiality within a single (multimaterial) 3D print. While tied to multimaterial 3D printing, the description of the models afford rendering in various materials. This is exploited in OpenFab [32], in which material properties are programmed. Yang et al. [34] describe a method that adjusts the shape of fabricated objects based on the materials used. Stress Relief [28] is a system that uses material properties to find out where objects are weak and reinforces the models where needed.

Portability in 3D printing

Grafter [22] allows users to recombine elements from multiple parent models found on online repositories into a new device. It does so by keeping mechanisms together and joining them with each other. This makes the mechanical design portable across models.

Similarly, *PARTS* [16] is a clever framework that makes functional entities in models parametric and enables models to be portable across a range of use contexts. These systems solve nasty portability problems, and could be further enhanced with springFit as none of them deal with the portability issues related to fit, raised in this paper.

Tolerance Optimization and springs

Mechanical engineers optimize their models for a very specific fabrication set-up. They do consider the tolerances and material behaviors but then optimize for the given set-up. For example Shin et al. [25] carefully design the behavior of connectors to have reliable force exchange between elements of modular robots. Chen et al. [10] optimize their design of snap fit geometry for injection molding, they create a constant stress model for snap fit mechanisms using this method. Sigmund [27] approaches the problem of manufacturing tolerances on a Nano fabrication scale. They use topology optimization considering best and worst case tolerances. They then make the structures more or less compliant to achieve the optimal result.

The idea of using springs as a method to achieve a wider range of tolerances has been explored in ME, as part of a "robust design methodology" [11]. In particular, Downey et al. [13] suggest embedding "smart features" in models to make them more robust to production or usage errors.

For specific applications in industry various spring optimization strategies have been developed. Liu et al. [20] for leaf springs in truck suspension, and Shokrieh [26] for light vehicles in general. And Zhu et al. [36] in the context of aerospace.

Joints based on Springs

Various joints commonly used in Mechanical Engineering contain the concept of springs. The most typical application is in the form of snap-fits [10]. Various ME handbooks describe the precise specification of such snap-fit joints [1, 14]. In this paper, we apply those insights in the context of modifying existing joints and mounts.

CLASSIFICATION AND CONVERSION ALGORITHM

SpringFit proceeds in two automatic steps. First, it analyzes the cutting plan at hand in order to locate press fit-based mounts and joints, i.e., mounts, finger joints, cross joints, and t-joints. Second, it replaces these mounts and joints with cantilever-based counterparts. It thereby computes optimal springs for each individual mount and joint. In a third manual step, a user can come in and override the suggested joints and mounts using a simple browser UI (as shown in Figure 19).

SpringFit's mount and joint classifier

As a first overview, Figure 13 illustrates the criteria springFit's joint classifier uses to locate (a) a circular mount (b) a cross joint, (c) a finger joint, or (d) a T-joint in the svg file it is processing.

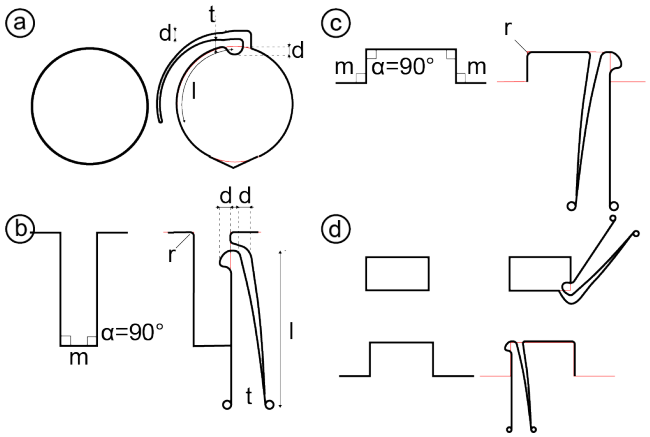

Figure 13: SpringFit's joint classifiers.

As shown in Algorithm 1, springFit uses the joint classifiers of Figure 13 to detect which segments in the svg repre-

sent what type of joint. It relies on the heuristic to find material thickness *m*, by plotting all line segments in a histogram ranging from 0 to 30mm with 100 equal bins. The most frequently occurring dimension is considered *m*. After classification, the SpringOptimizer (see section "Implementation") provides springFit with the optimal spring parameters for the given force, tolerance and spring types. A final pass creates the actual output file in which the identified spring elements are exchanged for actual spring geometry.

ALGORITHM 1: SPRING PLACEMENT

```
Input:  labeled SVG file D
        Pressfit force F
        Required tolerance t
Output: SVG file d*
lineLengths ← new array()
for each element ∈ D do
        add element.length to lineLengths
m = max(bucket_sort (lineLengths from 1 to 10mm))
for each class ∈ D do
    if class is labeled("press-fit")
     for each element ∈ class
        if element= "circle" or "rectangle" then
                delete element
                insert mount-element
        if element= "path" then
         for each segment ∈ path
            if segment.lentgh = m and
            segment[-2].length = m
                if segment[-2].angle =
                segment.angle = 90 or -90 then
                        replace segment with
                        "fingerjoint"
                if segment[-1].lentgh =
                segment.length[-3] then
                        replace segment with
                        "crossjoint"
spring = SpringOptimizer(springTypes,F,t)
for each element ∈ D do
    if element.type = "fingerjoint" then
            insert spring.finger to element
    if element.type = "crossjoint" then
            insert spring.cross to element
    if element.type = "mount" then
            insert spring.mount to element
    add element to d*
export d*
```

Generating mounts

As illustrated by Figure 14, springFit generates round mounts so as to make sure that any matching object will be held at three points forming an equilateral triangle. (1) SpringFit finds the inscribed equilateral triangle in the circle, (2) scales it down to fit the minimal required circle (by using the given tolerance requirement), (3) translates that circle until it intersects with the original cutout, and (4) overlaps these two circles so the final cutout will have shape reminiscent of an oval.

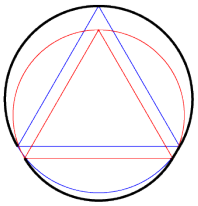

Figure 14: Mounts generated by springFit have the shape of the black shape shown here. It allows holding round physical objects at three points that together form an equilateral triangle. The red circle and the blue circle illustrate this for two specific diameters.

Cross joints

As illustrated by Figure 15, springFit classifies cross-joints by locating pairs of opposing line segments of the same length in the model with a segment of material thickness in between and straight angles.

The feature detector finds a cross joint in the model shown in Figure 15. First, (a) the path is split in segments (the dashes are the start of a new segment). (b) springFit loops through segments until one is found with length *m* (material thickness). (c) it then loops 2 more segments and checks whether element *n-1* and *n-3* have the same length (d) if so, it checks whether the angles are both 90° (or 270°) and uses the direction of the angles to determine how the spring will be inserted.

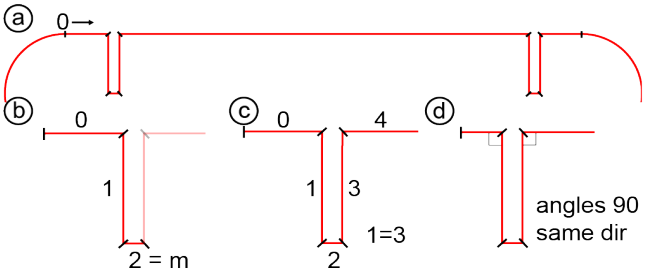

Figure 15: Notch joint classification.

SpringFit modifies this joint by inserting a cantilever spring next to the slit. It rounds the edges to prevent more brittle materials from cracking at sharp corners. The spring for cross joints lines up with the cutout.

As shown in Figure 11c, springFit does not place the cross joint spring all the way at the edge of the material as this would make the material much weaker to torque.

Finger joints

SpringFit locates finger joints as illustrated by Figure 16. (a) springFit loops through the segments (b) until a segment with length *m* is found. (c) it considers two segments forwards. springFit checks whether the *n*th segment has also length *m* (d) and confirms by checking the angles. It again uses the direction of the angles to determine how the spring will be inserted.

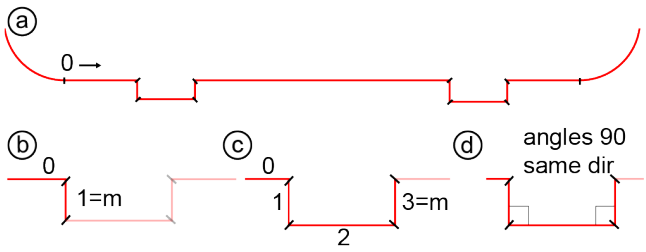

Figure 16: Finger joint classification.

SpringFit's finger joint classifier shown in Figure 13c, identifies two parallel edges of "material thickness" length and a 90 degree connector between them as well as 90 degree connections on both other sides of the segment. The replacing cantilever spring has the three core properties of length, displacement and thickness, are generated by springFit's spring optimizer (more detail in Section "Implementation").

T-joints

SpringFit classifies and converts T-joints as a side effect of mount and finger joint classification and conversion. One side of the T-joint is equivalent to a finger joint. The other side is a rectangular cutout—these are recognized and processed as rectangular mounts.

TECHNICAL EVALUATION OF CONVERSION

To validate the functionality of springFit, we ran it on 14 models, which we downloaded from Thingiverse. We picked these models as to show the maximum variety of joints and mounts.

We used 7 of these models to test our claim of *material*-independence by converting models and then fabricating them from both plywood and acrylic. We used the other 7 models to test our claims of *machine*-independence by fabricating them on two laser cutters that deferred quite substantially in terms of kerf (0.12 vs. 0.20mm).

Results

Eight of the fourteen models converted in fully automated fashion and Figure 17 shows four of them.

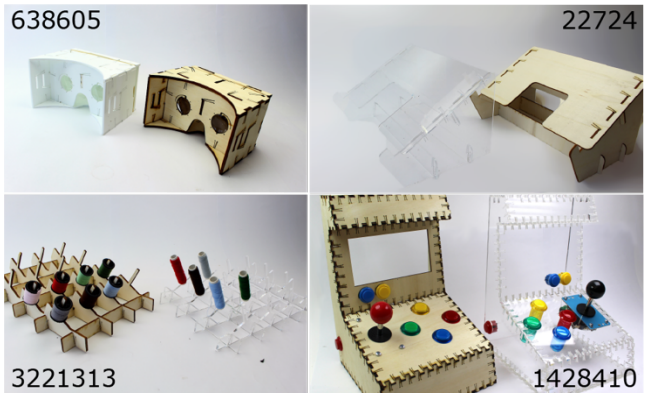

Figure 17: Four of the models we converted and fabricated as part of the first technical evaluation. (thingiverse IDs on the label)

SpringFit completed the conversion also for the other six models; the results, however, required manual fixing. (1) Three models had produced intersecting cantilever springs, which we resolved by manually deleting one or more springs (springFit allows for this, as shown in Figure 19). (2) Two models contained parts that were too small to contain the cantilever springs as shown in Figure 18. (3) One model (a heart shaped box) could not be converted because it contained bent fingers as shown in Figure 18c, springFit was unaware of this joint type.

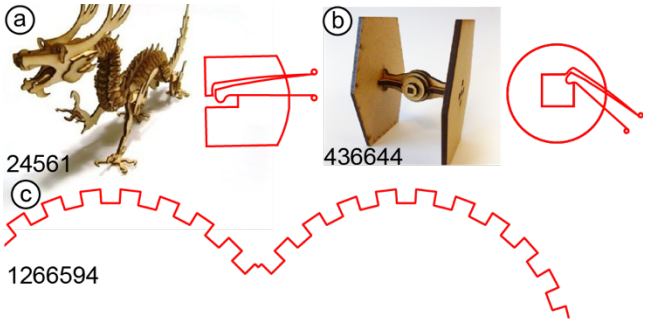

Figure 18: Models springFit could not convert (a,b) two of the models that contained parts too small to hold the required cantilever springs. (c) a model with non-straight finger joints.

All three issues are solvable in the long run. Future versions of springFit should address them by adding a better "routing algorithm" for the cantilever springs, by folding cantilever springs into the available space (similar to how springFit already produces curved springs), and by adding additional joint classifiers.

SpringFit identified all mounts and joints contained with 4% false positives. We inserted on average 105 springs per model (with the arcade of Figure 17c as extreme outlier with 477 springs). On average 4.1 springs were placed at cutouts that are not press-fit (and thus have been removed in the UI). The model with the most redundant springs was a Theremin model which has a lot of holes for bolts that are not press-fit (14 springs (27%) are not needed).

Users may choose to simply leave them (they generally do not really affect the model's functionality) or choose to remove them in the UI. For familiar models, the simple interaction shown in Figure 19 typically takes only a few seconds.

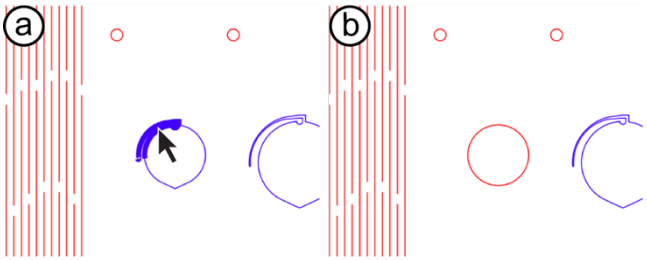

Figure 19: (a) SpringFit falsely classified this cutout as a press-fit and consequently created a cantilever spring for it. Leaving it in does not affect functionality. Alternatively, a mouse click in springFit reverts this mount to (b) the original version.

CANTILEVER SPRING DESIGN

At the lowest level, springFit is about cantilever spring design. When converting models, springFit calls the function *generateSpring*(force,tolerance) that generates optimal springs for the mount or joint at hand.

The first objective of cantilever design is to create a spring with a well-defined *holding force*. Holding forces for small buttons and joints may range anywhere from 1 to 5N. We made 5N the default in springFit, but users can override it.

The second parameter springFit optimizes for is the amount of *tolerance* the spring offers, i.e., the size difference between the smallest and the largest object this mount or joint will be able to hold. This parameter defines the deflection that the spring-fit will be able to accept. For example, when the user defines 1mm of tolerance, the system will generate a spring that exerts the desired holding force around ±1mm from the original point of fit.

The *generateSpring* function takes these two input parameters and minimizes the size of the spring. It has to conform 3 additional constraints: (1) the material should not break, (2) the force needs to be consistently applied within the given tolerance and (3) the resulting spring should be able to fabricate (not too small, not too big).

Design parameters and constant stress springs

As described in the "why it works" section, the stiffness of a cantilever can be varied by changing its shape. SpringFit specifies the stiffness k in the $F=kd$ relationship of the cantilever by using the shape parameters, l: length, t: thickness and d: deflection.

The smallest possible spring is one that has no redundant material. This happens when the stress of the spring is distributed evenly across the material. To achieve this, springFit uses the *constant stress* cantilever model. Similar to Shin et al. [25].

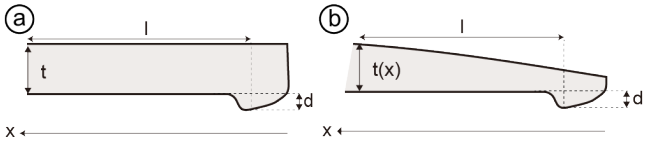

Figure 20: Design for a constant-strength cantilever. (a) Compared to usual 'bar' cantilever, (b) the *constant-stress* cantilever has constant bending stress along its length induced by input force at the end and thus is more space efficient.

Force/deflection relationship for cantilever springs

Next, springFit needs to know how the shape parameters influence each other and the required spring behavior. This is defined by the force-deflection relationship.

To acquire the force-deflection relationship for a cantilever, we use the Euler-Bernoulli beam theory [1]. The deflection of a cantilever d under the bending moment M is described by the elastic curve equation for the path along x:

$$d''(x) = -\frac{M}{EI} \tag{1}$$

E is Young's modulus of the material and I is its moment of inertia of a cross-section which is a shape dependent value which can be calculated as $I=mt^3/12$. The bending moment M is calculated by multiplying the distance from the point of force F to the point of interest (in case of a straight cantilever this will be $M=Fx$).

For curved springs with radius R and angle θ_e, springFit uses Castigliano's method [[7]] to derive the deflection. With the given elastic energy in the cantilever U, we calculate the deflection at the end.

$$\omega = \frac{\partial U}{\partial F} \tag{2}$$

F is the force applied at the end of the cantilever. To calculate the elastic bending energy stored in the spring, we integrate given the path C and small element ds of the cantilever. We neglected tension/compression energy here.

$$U = \int_C \frac{M^2}{2EI} ds \tag{3}$$

Thus we can derive $F=kd$ relationship resulting from solving equations (1)-(3). Table 1 shows the equations, which shows how much deflection can be caused by force F given the shape of cantilever.

Table 1: The calculations based on the cantilever dynamics model for each cantilever design we used in by springFit. I_0 here means moment of inertia of section at $x=l$.

shape	Constant-stress deflection d	thickness t
Straight	$\frac{2Fl^3}{3EI_0}$	$\sqrt{\frac{6Fx}{b\sigma_b}}$
Curved	$\frac{FR^3}{EI_0}\sqrt{sin\theta_e}\int_0^{\theta_e}\sqrt{sin\,\theta}\,d\theta$	$\sqrt{\frac{6FR}{b\sigma_b}sin\theta}$

Optimization and criteria

SpringFit aims to produce the smallest possible springs as this minimizes aesthetic and structural impact in the model. Since there are multiple parameter configurations that lead to the same spring stiffness (see Figure 8), springFit uses an optimization algorithm to pick the optimal design.

We write these criteria as an objective function L that penalizes for size of the spring (sizeof(π)). To let springs work across materials, we choose the minimum value of Young's modulus E and maximum stress σ_{max} from given set of materials specified by users (default are the typical materials used for laser cutting: cardboard, plywood, acrylic), the parameter s is a safety factor to compensate for slight variations within the material (f.e. grains of the wood). We furthermore have a lower bound for the force (otherwise it would not be press-fit) Following these criteria, the optimization problem of the cantilever design π with design parameters length, thickness, and deflection π:$\{l,t,d\}$ can be described as follows:

$$\pi^* = \arg\min_{\pi} L(\pi)$$
$$\text{s.t.} \begin{cases} L(\pi) = \text{sizeof}(\pi) \\ \sigma_{max} < \frac{\sigma_+}{s} \\ F \quad > F_{min} \end{cases} \tag{4}$$

We obtain the actual values of Young's modulus E and the maximum strength σ_{max} of the materials from material testing (see Section "Technical Evaluation"). For solving the nonlinear optimization problem with multiple constraints we use the COBYLA algorithm from NLopt C++ library (nlopt.readthedocs.io).

With help of this procedure, springFit determines the optimal length, thickness and deflection of the cantilever that produces required force across different materials or kerfs.

SpringFit as a service

SpringFit's spring optimizer is running as a service that is called by the browser application, when the user annotates parts, springFit returns springs. The service architecture furthermore allows for easy integration in existing editors.

The annotations are passed as CSS class elements in the svg. Tech-savvy users can directly modify the CSS classes and run springFit as a command line tool.

TECHNICAL EVALUATION OF SPRING PERFORMANCE

To verify that (1) cantilever springs made from plywood actually deliver reliable repeatable force and tolerance, (2) to test our claim that a single spring design including its dimensions works in plywood *and* acrylic, and (3) to verify the design parameters of our springs, we measured forces and tolerances of springFit-generated springs using a testing setup.

The test setup we created is shown in Figure 21. It uses a linear actuator to automatically push a spring in increments of 0.1 mm against a force gauge (SAUTER FK-100).

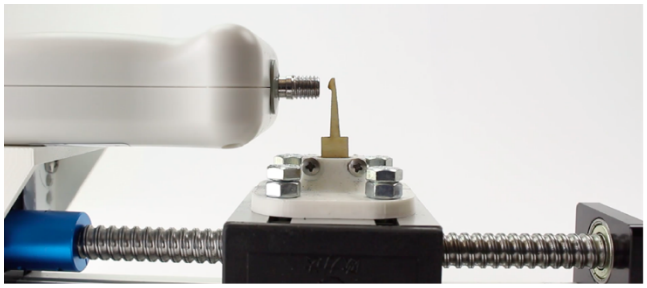

Figure 21: Spring strength example setup. Linear actuator that moves test pieces generated by springFit into a digital force gauge.

The springs we tested were at least 3mm thick at their base and were designed to allow for at least 1mm deflection. We repeated each test with 10 springs. We generated springs optimized by springFit with two different forces. We tested straight and curved cantilever springs, from plywood and acrylic (so 2x2x2x10 samples). For plywood test pieces, we laser cut the springs *along* the grain of the material of the outer layers which is the easier side to break with the applied force.

Results Figure 22 shows the results for the straight cantilever springs. As the diagrams show, the tested springs behaved well, i.e., produced reliable and repeatable force, which proves our optimization criterion (1).

The straight cantilever springs produced a consistent force of around 5[N] largely irrespective of the material (blue line = acrylic vs. orange line = plywood). This means that the tested springs were functional across materials, which also supports our second criterion (2).

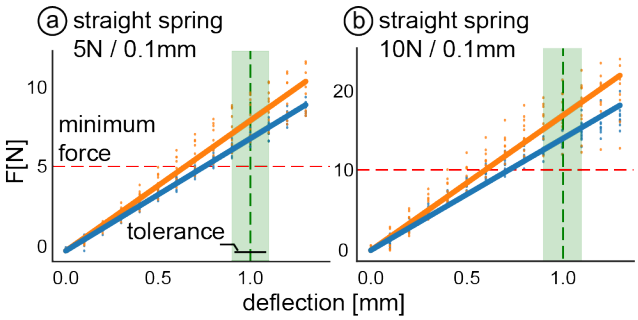

Figure 22: (a) Force-deflection diagram of generated cantilever springs with input force of up to 10N. Red dashed line shows the input minimum force and green band shows the input tolerance=0.1mm. (Blue = acrylic, orange = plywood) (b) Same diagram of bar spring with 10N.

Figure 23 shows the corresponding results for the curved springs, as used in the round mounts shown in earlier figures. As the diagram shows, the curved acrylic springs produced a slightly larger deflection given the same force compared to the straight cantilever springs. Surprisingly, the curved *plywood* springs were roughly half as stiff as their straight counterparts, i.e., they deflected twice as much.

This was caused by the non-elastic material behavior of the curved springs which breaks the assumption on Castigliano's elastic energy method. Since the curved springs have larger deflection compared to bar springs, it will likely fail to produce the linear elasticity assumed in the cantilever model.

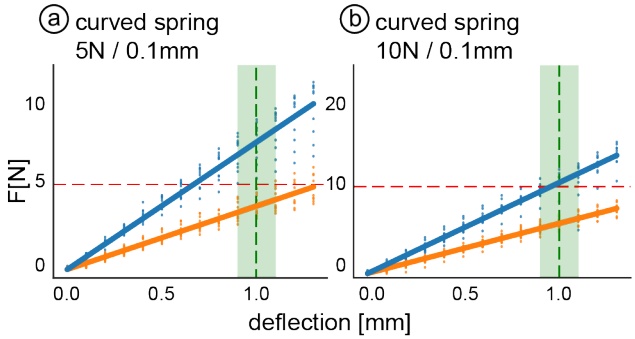

Figure 23: Same diagram for *curved* cantilever springs.

In summary, all tested springs behaved well. The stiffness of the curved plywood springs was unexpected, but still predictable. This allowed us to embody these findings into springFit by tuning our models there. This now allows springFit to produce springs of desired properties reliably

irrespective of materials and shape. More testing would be required to find out whether the springs keep their characteristics under highly frequent or long term (dis)assembly.

CONCLUSION

Mounts and joints are both crucial building elements of laser cutting, as they enable users to produce 3D models as well as a wide range of physical extensions. In this paper, we discussed why such models tend to fail when shared. We then demonstrated how to overcome problem by replacing the problematic press fit-based mounts and joints with, what we call, cantilever-based mounts and joints.

Zooming out into future work, we argue that springFit is the first facet of a much larger puzzle, which is the notion of "portability" and "maintainability" in digital fabrication. Today, fabrication machines are still in a phase where new technologies emerge on a fast clock. Under such circumstances, how can a fabrication ecosystem guarantee that today's designs still fabricate reliably a year from now? What about in 10 or 100 years, i.e., after the machines and materials a model was initially designed for have long disappeared from the market? As our exploration into machine-independent mounts and joints suggests, not everything needed to understand and reproduce a digital models is contained in the files supposed the represent it. So how to represent digital fabrication models so as to make them portable and maintainable?

REFERENCES

[1] BASF, Snap-Fit Design manual, BASF Corporation Engineering Plastics, USA, 2007 http://www8.basf.us//PLASTICSWEB/displayanyfile?id=0901a5e1801499d5

[2] Bauchau, O. A., and J. I. Craig. "Euler-Bernoulli beam theory." In Structural analysis, pp. 173-221. Springer, Dordrecht, 2009. DOI: https://doi.org/10.1007/978-90-481-2516-6_5

[3] Patrick Baudisch and Stefanie Mueller. 2017. Personal Fabrication, Foundations and Trends in Human–Computer Interaction: Vol. 10: No. 3–4, pp 165-293. http://dx.doi.org/10.1561/1100000055

[4] Patrick Baudisch, Arthur Silber, Yannis Kommana, Milan Gruner, Ludwig Wall, Kevin Reuss, Lukas Heilman, Robert Kovacs, Daniel Rechlitz, and Thijs Roumen. 2019. Kyub: a 3D Editor for Modeling Sturdy Laser-Cut Objects. In 2019 CHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019), May 4–9, 2019, Glasgow, Scotland, UK. ACM, New York, NY, USA. https://doi.org/10.1145/3290605.3300796

[5] 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. DOI: https://doi.org/10.1145/2702123.2702225

[6] Leah Buechley and Hannah Perner-Wilson. 2012. Crafting technology: Reimagining the processes, materials, and cultures of electronics. ACM Trans. Comput.-Hum. Interact. 19, 3, Article 21 (October 2012), 21 pages. DOI: https://doi.org/10.1145/2362364.2362369

[7] Alberto Castigliano. 1879. Théorie de l'équilibre des systèmes élastiques et ses applications. Vol. 1. AF Negro, 1879.

[8] Xiang 'Anthony' Chen, Stelian Coros, and Scott E. Hudson. 2018. Medley: A Library of Embeddables to Explore Rich Material Properties for 3D Printed Objects. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18). ACM, New York, NY, USA, Paper 162, 12 pages. DOI: https://doi.org/10.1145/3173574.3173736

[9] Xiang 'Anthony' Chen, Jeeeun Kim, Jennifer Mankoff, Tovi Grossman, Stelian Coros, and Scott E. Hudson. 2016. Reprise: A Design Tool for Specifying, Generating, and Customizing 3D Printable Adaptations on Everyday Objects. In Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16). ACM, New York, NY, USA, 29-39. DOI: https://doi.org/10.1145/2984511.2984512

[10] Yi-Ho Chen, and Chao-Chieh Lan. 2012. Design of a constant-force snap-fit mechanism for minimal mating uncertainty. *Mechanism and Machine Theory* 55: 34-50. DOI: https://doi.org/10.1016/j.mechmachtheory.2012.04.006

[11] Martin Ebro Christensen, Thomas J. Howard, and Janus Juul Rasmussen. 2012. The foundation for robust design: enabling robustness through kinematic design and design clarity. In 12th International design conference. Design Society.

[12] James McCrae, Nobuyuki Umetani, and Karan Singh. 2014. FlatFitFab: interactive modeling with planar sections. In Proceedings of the 27th annual ACM symposium on User interface software and technology (UIST '14). ACM, New York, NY, USA, 13-22. DOI: https://doi.org/10.1145/2642918.2647388

[13] Kris Downey, Alan Parkinson, and Ken Chase. 2003. An introduction to smart assemblies for robust design. Research in Engineering Design 14, no. 4 (2003): 236-246. DOI: https://doi.org/10.1007/s00163-003-0041-5

[14] Erhard, Gunter, and Martin Thompson. *Designing with plastics. Chapter 8: Flexing Elements* Munich, Germany: Hanser, 2006.

[15] Florian Heller, Jan Thar, Dennis Lewandowski, Mirko Hartmann, Pierre Schoonbrood, Sophy Stönner, Simon Voelker, and Jan Borchers. 2018. CutCAD - An Open-source Tool to Design 3D Objects in 2D. In Proceedings of the 2018 Designing Interactive Systems Conference (DIS '18). ACM, New York, NY, USA, 1135-1139. DOI: https://doi.org/10.1145/3196709.3196800

[16] Megan Hofmann, Gabriella Hann, Scott E. Hudson, and Jennifer Mankoff. 2018. Greater than the Sum of its PARTs: Expressing and Reusing Design Intent in 3D Models. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18). ACM, New York, NY, USA, Paper 301, 12 pages. DOI: https://doi.org/10.1145/3173574.3173875

[17] Jeeeun Kim, Anhong Guo, Tom Yeh, Scott E. Hudson, and Jennifer Mankoff. 2017. Understanding Uncertainty in Measurement and Accommodating its Impact in 3D Modeling and Printing. In Proceedings of the 2017 Conference on Designing Interactive Systems (DIS '17). ACM, New York, NY, USA, 1067-1078. DOI: https://doi.org/10.1145/3064663.3064690

[18] Yuki Koyama, Shinjiro Sueda, Emma Steinhardt, Takeo Igarashi, Ariel Shamir, and Wojciech Matusik. 2015. AutoConnect: computational design of 3D-printable connectors. ACM Trans. Graph. 34, 6, Article 231 (October 2015), 11 pages. DOI: https://doi.org/10.1145/2816795.2818060

[19] Manfred Lau, Akira Ohgawara, Jun Mitani, and Takeo Igarashi. 2011. Converting 3D furniture models to fabricatable parts and connectors. ACM Trans. Graph. 30, 4, Article 85 (July 2011), 6 pages. DOI: https://doi.org/10.1145/2010324.1964980

[20] Xin Liu, and Y. S. Chadda. 1993. Automated optimal design of a leaf spring. No. 933044. SAE Technical Paper. DOI: https://doi.org/10.4271/933044

[21] Raf Ramakers, Fraser Anderson, Tovi Grossman, and George Fitzmaurice. 2016. RetroFab: A Design Tool for Retrofitting Physical Interfaces using Actuators, Sensors and 3D Printing. In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16). ACM, New York, NY, USA, 409-419. DOI: https://doi.org/10.1145/2858036.2858485

[22] Thijs Jan Roumen, Willi Müller, and Patrick Baudisch. 2018. Grafter: Remixing 3D-Printed Machines. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18). ACM, New York, NY, USA, Paper 63, 12 pages. DOI: https://doi.org/10.1145/3173574.3173637

[23] Valkyrie Savage, Sean Follmer, Jingyi Li, and Björn Hartmann. 2015. Makers' Marks: Physical Markup for Designing and Fabricating Functional Objects. In Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology (UIST '15). ACM, New York, NY, USA, 103-108. DOI: https://doi.org/10.1145/2807442.2807508

[24] Yuliy Schwartzburg and Mark Pauly. 2013. Fabrication‐aware design with intersecting planar pieces. In Computer Graphics Forum, vol. 32, no. 2pt3, pp. 317-326. Oxford, UK: Blackwell Publishing Ltd DOI: https://doi.org/10.1111/cgf.12051

[25] Sung Ho Shin. 2004 Analytic integration of tolerances in designing precision interfaces for modular robotics. PhD diss., UTexas. http://hdl.handle.net/2152/2195

[26] Mahmood Shokrieh, and Davood Rezaei. 2003 "Analysis and optimization of a composite leaf spring." Composite structures 60, no. 3: 317-325. DOI: https://doi.org/10.1016/S0263-8223(02)00349-5

[27] Ole Sigmund, 2009 Manufacturing tolerant topology optimization. Acta Mechanica Sinica 25, no. 2: 227-239. DOI: https://doi.org/10.1007/s10409-009-0240-z

[28] Ondrej Stava, Juraj Vanek, Bedrich Benes, Nathan Carr, and Radomír Měch. 2012. Stress relief: improving structural strength of 3D printable objects. ACM Trans. Graph. 31, 4, Article 48 (July 2012), 11 pages. DOI: https://doi.org/10.1145/2185520.2185544

[29] Francisca Gil Ureta, Chelsea Tymms, and Denis Zorin. 2016. Interactive modeling of mechanical objects. In Computer Graphics Forum, vol. 35, no. 5, pp. 145-155. DOI: https://doi.org/10.1111/cgf.12971

[30] Nurcan Gecer Ulu, Stelian Coros, and Levent Burak Kara. "Designing coupling behaviors using compliant shape optimization." *Computer-Aided Design* 101 (2018): 57-71. DOI: https://doi.org/10.1016/j.cad.2018.03.008

[31] Kiril Vidimce, Alexandre Kaspar, Ye Wang, and Wojciech Matusik. 2016. Foundry: Hierarchical Material Design for Multi-Material Fabrication. In Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16). ACM, New York, NY, USA, 563-574. DOI: https://doi.org/10.1145/2984511.2984516

[32] Kiril Vidimče, Szu-Po Wang, Jonathan Ragan-Kelley, and Wojciech Matusik. 2013. OpenFab: a programmable pipeline for multi-material fabrication. ACM Trans. Graph. 32, 4, Article 136 (July 2013), 12 pages. DOI: https://doi.org/10.1145/2461912.2461993

[33] Christian Weichel, Manfred Lau, and Hans Gellersen. 2013. Enclosed: a component-centric interface for designing prototype enclosures. In Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction (TEI '13). ACM, New York, NY, USA, 215-218. DOI: https://doi.org/10.1145/2460625.2460659

[34] Yong-Liang Yang, Jun Wang, and Niloy J. Mitra. 2015. Reforming shapes for material-aware fabrication. In Proceedings of the Eurographics Symposium on Geometry Processing (SGP '15). Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 53-64. DOI: http://dx.doi.org/10.1111/cgf.12696

[35] Clement Zheng, Ellen Yi-Luen Do, and Jim Budd. 2017. Joinery: Parametric Joint Generation for Laser Cut Assemblies. In Proceedings of the 2017 ACM SIGCHI Conference on Creativity and Cognition (C&C '17). ACM, New York, NY, USA, 63-74. DOI: https://doi.org/10.1145/3059454.3059459

[36] Ji-Hong Zhu, Wei-Hong Zhang, and Liang Xia. 2016. Topology optimization in aircraft and aerospace structures design. Archives of Computational Methods in Engineering 23, no. 4: 595-622. DOI: https://doi.org/10.1007/s11831-015-9151-2

[37] Amit Zoran. 2015. Hybrid craft: showcase of physical and digital integration of design and craft skills. In ACM SIGGRAPH Art Gallery (SIGGRAPH '15). ACM, New York, NY, USA, 384-398. DOI: http://dx.doi.org/10.1145/2810185.2810187

[38] Zoran, Amit, and Leah Buechley. "Hybrid reassemblage: an exploration of craft, digital fabrication and artifact uniqueness." Leonardo 46, no. 1 (2013): 4-10. DOI: https://doi.org/10.1162/LEON_a_00477