TrussFab: Fabricating Sturdy Large-Scale Structures on Desktop 3D Printers

Robert Kovacs, Anna Seufert, Ludwig Wall, Hsiang-Ting Chen, Florian Meinel, Willi Müller, Si-jing You, Maximilian Brehm, Jonathan Striebel, Yannis Kommana, Alexander Popiak, Thomas Bläsius, and Patrick Baudisch

Hasso Plattner Institute, Potsdam, Germany

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

ABSTRACT

We present *TrussFab*, an integrated end-to-end system that allows users to fabricate large scale structures that are sturdy enough to carry human weight. TrussFab achieves the large scale by complementing 3D print with plastic bottles. It does not use these bottles as "bricks" though, but as beams that form structurally sound node-link structures, also known as *trusses*, allowing it to handle the forces resulting from scale and load. TrussFab embodies the required engineering knowledge, allowing non-engineers to design such structures and to validate their design using integrated structural analysis. We have used TrussFab to design and fabricate tables and chairs, a 2.5 m long bridge strong enough to carry a human, a functional boat that seats two, and a 5 m diameter dome.

Author Keywords

Fabrication; 3D printing; truss structure.

ACM Classification Keywords

H5.2 [Information interfaces and presentation]: User Interfaces: Input Devices and Strategies, Interaction Styles.

INTRODUCTION

Personal fabrication tools, such as 3D printers have become popular in HCI, where they have been used for fast prototyping [20] as well as to fabricate interactive objects [9], optical elements [34], or kinematic characters [5]. Since 3D printers today are available in a desktop form factor, they have been able to spread to the maker community [30] and are now increasingly reaching the consumer market [27].

In contrast, the fabrication of *large* objects still has remained a privilege of industry, which has access to specialized equipment, such as concrete printers that allow making houses [13] or robotic-arms capable of 3D printing [11]. The owners of the widespread desktop devices, in contrast, cannot participate in this evolution, because the underlying technology does not scale. Even techniques that break down

large models into printer-sized parts [16] ultimately do not scale, as large models consume material and time proportional to their size, which quickly renders 3D printing and related techniques intractable for larger-than-desktop-scale models.
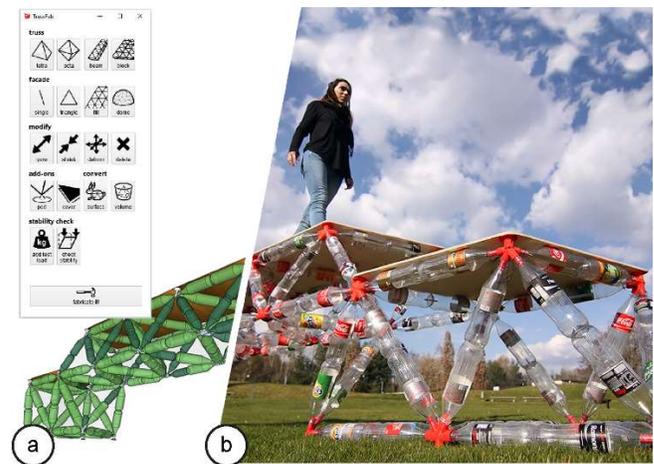

Figure 1: *TrussFab* is a system that allows users to fabricate large structures sturdy enough to carry human weight. Truss-Fab considers bottles as beams that form structurally sound node-link structures also known as *trusses*, allowing it to handle the forces resulting from scale and load.

As an alternative approach, fabrication enthusiasts have created large objects by combining 3D print with ready-made objects, such as plastic bottles [39]. In their simplest form, such objects wrapped in 3D print can serve as 3D voxel collages that approximate the volume of an object [36].

Going larger, however, is not only about scale and print volume. For large objects, the main design objective is typically to withstand large forces, as forces grow cubed with the size of the object. Also, large objects afford substantial external loads; furniture, bridges, and vehicles, for example, all must be engineered to hold the weight of a human. Designing for large forces, however, requires substantial engineering skill [40] from envisioning appropriate structures in the first place to verifying their structural integrity.

In this paper, we present *TrussFab*, an integrated end-to-end system that allows users to design large structures that are sturdy enough to carry human weight (Figure 1). TrussFab achieves this by taking a different perspective on bottles. Unlike previous systems that stacked bottles as if they were "bricks", TrussFab considers them as beams and uses them

to form structurally sound node link structures based on closed triangles, also known as *trusses*. TrussFab embodies the required engineering knowledge, allowing non-engineers to design such structures. TrussFab also allows users to validate their designs using integrated structural analysis (Figure 3). Our main contribution is this end-to-end system.

WALKTHROUGH OF THE TRUSSFAB SYSTEM

TrussFab allows users to create structures either by modeling from scratch or by converting existing 3D models. It supports an integrated workflow, which we summarize in the following and then discuss in detail throughout the rest of the paper.

Figure 2: (a) TrussFab's converter automatically turns this 3D model of a coffee table, into (b) a sturdy tetrahedral honeycomb structure, (c) which fabricated serves as a functional table.

Step 1: Automatic conversion. One way to create TrussFab structures is to convert an existing 3D model using TrussFab's *converter*. As shown in Figure 2 this converts the volume of the model into a tetrahedral honeycomb structure, allowing it to bear substantial load.
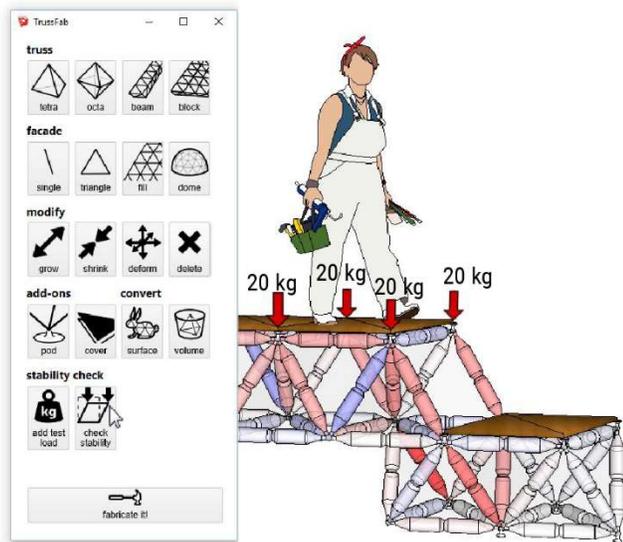
Figure 3: TrussFab's editor is implemented as an extension for *SketchUp*. Here the user is performing a stability check on the bridge from Figure 1.

Step 2: Editing. TrussFab's editor allows users to refine an object created by automatic conversion (Figure 2) or to start a new object from scratch (Figure 3). We implemented

TrussFab's editor as an extension to the 3D modeling software *SketchUp* [41]. TrussFab's editor offers all the functionalities of the original SketchUp system, plus custom functions that help users create sturdy structures. In particular, TrussFab's editor offers primitives that are elementary trusses (tetrahedra and octahedra), tools that create large beams in the form of trusses, and tools for tweaking the shape of structures, while maintaining their truss structure. In Figure 3, the user placed a human weight on top of the bridge design. TrussFab's integrated structural analysis shows no warnings, suggesting that the bridge is structurally sound.

Step 3: Hub generation. After designing a structure, TrussFab's *hub generator* generates the 3D models of all hubs. Figure 4 shows our 3D printed hub design; the connector on the top snaps into the bottleneck, while the bottom ones are holding the bottles by their threaded neck. When designing structures to carry a human weight, these hubs experience large forces. We discuss the details of TrussFab's hub designs in section "Hubs and Members".
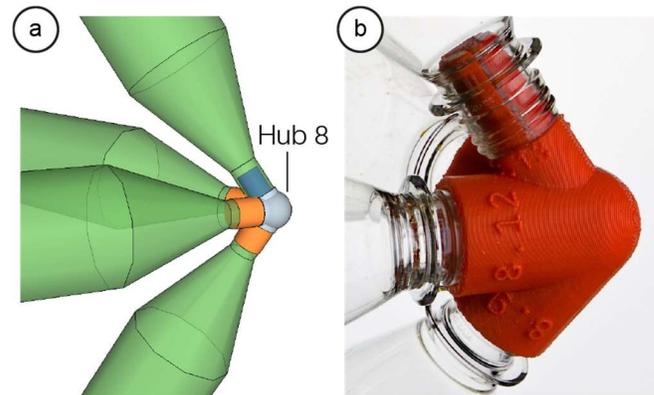
Figure 4: (a) TrussFab generates a hub for every node in the model. (b) The hubs are then 3D printed. Users assemble them following embossed IDs.

Step 4: Fabrication. Users then send the 3D model files produced by the hub generator to one or more 3D printers in order to manufacture them.

Step 5: Assembly. Users now manually assemble their structures by following unique IDs embossed into each hub (Figure 4b).

TrussFab's underlying structure achieves stability

The key ideas behind TrussFab are (1) to employ bottles in their structurally most sturdy way, i.e., as beams from bottom to bottleneck and (2) to afford sturdy "closed frame structures", also known as *trusses* [15].

While freestanding bottles tend to break easily (Figure 5a/b), truss structures essentially consist of triangles. In such an arrangement, it is the structure that prevents deformation, not the individual bottle. The main strength of trusses is that they turn lateral forces (aka bending moments) into tension and compression forces along the length of the edges (aka *members*). Bottles make great members: while they buckle easily when pushed from the side, they are *very* strong when

pushed or pulled along their main axis (see section "Strength test"). (c) The resulting structures, such as this tetrahedron, are strong enough to bear the weight of one or more humans. (d) TrussFab affords building trusses by combining tetrahedra and octahedra into so-called tetrahedral honeycomb structures. The table in Figure 2, for example, is cut from such a "tetra-octa" mesh. This structure is commonly used in truss design and provides structural stability [4].
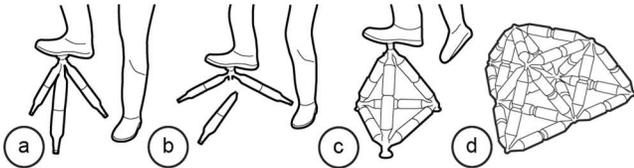

Figure 5: (a) Large objects involve large levers, causing them (b) to break under load. (c) TrussFab instead affords structures based on closed triangles, here forming a tetrahedron. Such structures are particularly sturdy. (d) TrussFab extends this concept to tetrahedron-octahedron trusses of arbitrary size.

Alternatively, we considered using larger stable primitives as building blocks, such as an icosahedron. We opted for the tetrahedra and octahedra, as they are space-filling, thus provide a simpler construction grid than other geometries.

Optimized construction using "facades"
TrussFab affords creating large structures by specifically supporting hierarchical construction. Figure 6 illustrates this. (a) Users start by creating a load-bearing structure in the form of trusses. (b) Users then fill in the non-load-bearing sides as *facades*. The benefit of this two-stage process is that facades are particularly efficient. First, they are single-layer, thus require fewer bottles. Second, the hubs that form a facade are flat; this allows TrussFab to fabricate such hubs using a laser cutter, which is very fast (40x faster than 3D print, see section "Laser-cut hubs for facades").
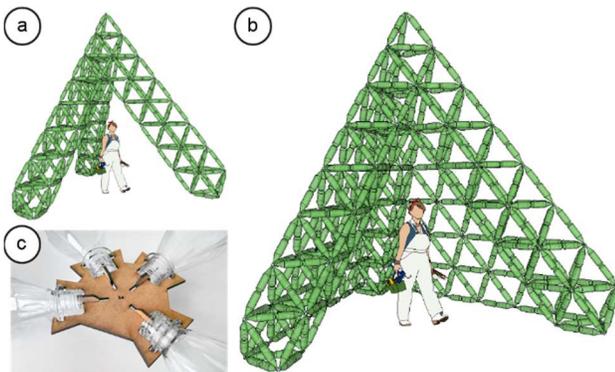

Figure 6: (a) The load-bearing structure of this tipi is best made from trusses; (b) its sides can be filled using *facades*. (c) TrussFab makes 2D facade hubs quickly on a laser cutter.

To support this classic layered architecture, TrussFab complements all of its truss tools with specialized facade tools. The editor offers facade tools for filling large opening with facades (Figure 6b). The hub generator offers the aforementioned laser-cut facade hubs. And, for object not expected to bear a load at all, TrussFab's surface converter turns the outer hull into a hollow facade structure (Figure 18).

CONTRIBUTION, BENEFITS, AND LIMITATIONS
Our main contribution is the integrated end-to-end system that allows users to fabricate large structures that are sturdy enough to carry human weight—on desktop 3D printers. Unlike previous systems that build on up-cycled plastic bottles combined with 3D print, TrussFab considers bottles not as "bricks", but as beams that form structurally sound node link structures also known as *trusses*, allowing users to handle the forces resulting from scale and load.

TrussFab embodies the required engineering knowledge, allowing non-engineers to design such structures, and allows users to validate their designs using integrated structural analysis. In particular, TrussFab's editor offers primitives that are trusses (tetrahedra and octahedra), tools that create large beams that are trusses, and tools for tweaking the shape of structures, while maintaining its truss structure. On the mechanical side, we contribute the key structural elements that allow creating trusses, i.e., the 3D-printed and laser-cut hub design.

We have validated our system by designing and fabricating tables and chairs, a 2.5 m bridge strong enough to carry a human, a 5 m diameter dome consisting of 512 bottles and a functional boat that seats two, shown in Figure 7.

Figure 7: A functional boat created using TrussFab. (a) Using the editor, users can design their structures efficiently. (b) The bottle-frame is simply covered by tarp. (c) The fabricated boat seats two.

Our approach is subject to the general limitations faced by ready-made objects. In particular, TrussFab can reproduce neither details smaller than a bottle nor closed surfaces.

RELATED WORK
TrussFab builds on previous efforts in the following branches: large-scale personal fabrication, design with existing objects and construction kits, and tools for creating structurally sound objects.

Large-scale personal fabrication

Architects and engineers have made efforts to scale up the additive manufacturing process for constructing large-scale structures, such as houses or sculptures. These efforts mostly involve a scaled-up version of the common machinery, like concrete printers [13], or breaking down the objects into smaller parts to print on desktop machines [16,17].

Another approach to fabricating architectural-scale objects is the use of mobile printing robots, which move on the ground [12] or fly [35] around the printed object. Yet another approach is to use human assisted devices, such as Protopiper [1]. Similarly, Yoshida et al. [37] proposed a computer-assisted fabrication method for large-scale architecture that combines a hand-held chopstick dispenser with a projector-based guiding system. Lafreniere et al. [14] coordinate multiple workers while collaboratively fabricating a pavilion.

Construction kits

Construction kits are popular for fast prototyping and fabrication. They offer a repertoire of prefabricated elements, which can be combined in various ways. Henrik and Kobbelt [38] developed a system to accurately approximate complex shapes using the Zometool mathematical modeling kit. Skouras et al. [28] created an interactive editor to computationally combine interlocking elements into a desired shape.

Designing with ready-made objects

MixFab [33], and Encore [3] allow users to integrate existing objects into their design. For creating objects enclosing electronic components Ashbrook et al. [2] developed an augmented fabrication system. Devendorf and Ryokai [6] proposed a human-assisted fabrication system that helps users incorporate everyday objects into 3D print. Beady [10] approximates models from beads and offers an editor for refining them.

Gellért assembled wooden boards combined with 3D printed connectors in node-link structure [8]. Combining carbon tubes with 3D printed metal has been proposed for creating functional cars [40].

Skilled individuals have stacked or tied plastic bottles in order to make art pieces, furniture, rafts or houses [39]. Yamada et al. [36] proposed a system for arranging ready-made objects into 3D shapes using 3D-printed connectors.

Tools for creating structurally sound objects

Smith et al. [29] developed a system that automatically generates truss structures using non-linear optimization. Makris et al. [18] proposed a design tool that generates parametrically defined, semi-automatically analyzed, and visualized structures. Wang et al. [32] developed an automated method that minimizes material cost by converting solid 3D models into a skin-frame structure. SketchChair [26] is an interactive chair design system that allows users to validate the structural integrity of their design by subjecting it to the weight of a human rag doll. Similarly, Umentani et al. [31] created a system for exploring physically valid shapes in furniture design.

Commercial tools for engineering truss structures include SkyCiv [42], which allows users to analyze the force distribution in trusses. MiTek's PAMIR| [43] is a specialized tool for creating timber rooftops. TrussTool [44] allows assembling constructions from ready-made trusses. In contrast, TrussFab affords creating trusses, rather than just assembling them.

WALKTHROUGH OF THE TRUSSFAB EDITOR

We now zoom in on step 2 of our workflow: the TrussFab editor. The main design rationale behind the editor is to afford stable structure. It achieves this by using bottles as the members of trusses. The key idea behind TrussFab's editor is to start any design with primitives that already are trusses, i.e., tetrahedra and octahedra; additional functions then allow users to extend and tweak the structure while maintaining the truss property at all times. Once the main truss structure has been created, users may add facades and decorative details.

Design based on tetrahedron-octahedron primitives

In this section we demonstrate the use of the TrussFab editor on our *chair with backrest* design.

(a) ⬦ As illustrated by Figure 8a, we start our design by creating the base of the chair. We select the *octahedron* tool from TrussFab's drawing toolbar and click on the ground plane of our workplace, which creates an octahedron. By default, this octahedron is made from small (half-liter) bottles. We check its height using the standard SketchUp measurement tool—it is 45cm, which is a good height for an average person to sit on.
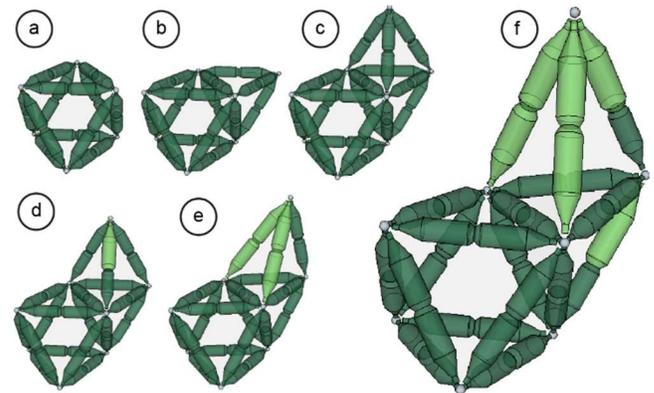

Figure 8: Creating a *chair with backrest* in TrussFab's editor

(b) △ To make the backrest, we now select the *tetrahedron* tool. We click on one of the sides of the base octahedron, which attaches a tetrahedron to it. (c) We click the top surface of the tetrahedron we just made, attaching one more tetrahedron to it. This gives us the rough shape of our chair and its backrest.

(d) ↗ The backrest is a little short. We select the *grow* tool and click one of the three upper members of the backrest. This elongates one of the two bottles to the next supported size, i.e., a 1.5-liter bottle, here shown as light green. We click again, which causes the second bottle of this member

to grow as well. (e) Repeating this on the other side scales the backrest to the desired size.

(f) The chair is now too "laid back". Still holding the grow tool, we click the rear edge of the backrest until the backrest is more vertical.

Alternatively, instead of using the discrete grow/shrink tools, user can also use the *deform* tool, which allows freely dragging individual hubs in space, while preserving the truss nature of the model (see section "Editing larger objects efficiently"). In addition to the standard member lengths governed by bottle sizes, TrussFab can implement extra-long members and in-between sizes by extending hubs with 3D print, as shown in Figure 9. This allows users for more freedom in their design.
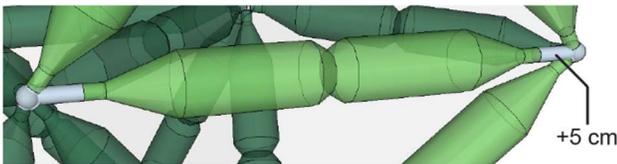

Figure 9: TrussFab implements the extra-long members using extra 3D print.

We now select the *pod* tool and use it to add *pods* to the bottom of our chair, as illustrated by Figure 10a. As discussed earlier, bottles are sturdy only when forces apply *along* their main axis. This is not the case for an octahedron directly touching the ground. If we tried to sit on it, our weight would cause the members touching the ground to buckle and break. The *pods* avoid this by propagating the user's weight into the truss, making the design robust.

Finally, we select the *cover* tool and click the top of the octahedron (Figure 10b). This adds a plywood plate for users to sit on; it is supported by three upward-facing pods. Plates will later be exported as *SVG* files. We tend to fabricate them on a laser cutter.
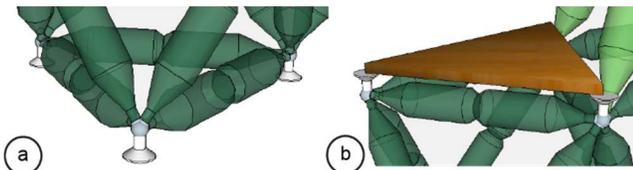

Figure 10: (a) Adding *pods* to the bottom hubs and (b) a cover to the top adds stability to our design, as any load is now propagated through the hubs, saving members from buckling.

To verify the chair's ability to carry a human user, we select the *add weight* tool and click on the seat plate (Figure 11). This places 10 kg weight on each of the three corners of the plate. We click three more times, which increases the weight in 5 kg steps, to sum up in total of 75 kg. A click at the backrest adds another 10kg pushing load into the backrest.

Clicking the *check stability* icon causes TrussFab to compute the effect of these weights onto the structure. This happens in two steps. First, the software looks for flaws in the truss structure, i.e., it searches for parts that are not completely locked in place by other members and are subject to shearing forces (see section "Implementation"). If found, the software would suggest placing additional stabilizing members. Our chair, however, is rigid, so there are no warnings.

Second, the software checks whether the structure will hold up the weight we placed on it. Using finite element analysis, the software calculates the forces that apply to every member of the structure. As show in Figure 11a, TrussFab shades all members accordingly. The six vertical members of the octahedron now appear in shades of red, suggesting that these are experiencing compression. So does the chair's "backbone". All other members are tinted blue, suggesting that these are subject to tension.
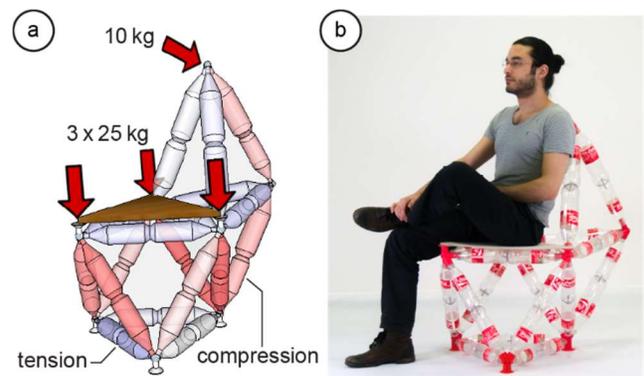

Figure 11: (a) To verify the chair's structural stability we add the load forces expected during use. The system now calculates internal compression and tension forces. Here, no forces are exceeding the limit; thus no warnings are displayed. (b) As predicted, the fabricated chair holds the human weight.

TrussFab compares these forces with the maximal load members and hubs can hold (see section "Hubs and Members"). It warns the user if the limits are exceeded. This is not the case here, so we now know that our chair model is structurally sound.

Finally, we click the *fabricate it!* button. This causes TrussFab to generate 3D models of all hubs and export them in *STL* format to the 3D printer. For the wooden seat cover, TrussFab creates a 2D line drawing in *SVG* format and sends it to a laser cutter. Users now assemble hubs and bottles based on the embossed hub IDs, resulting in the chair shown in Figure 11b. This particular design prints in 90 min per hub on a *MakerBot 2X*, and takes about 20 minutes to assemble.

Note how the interaction we described afforded creating a stable structure. In particular, the octahedron we started out with was a truss and thus stable. We then added tetrahedra, which turned our initial truss into a larger truss. Tweaking the length of individual members, finally, did not affect the structure of our design, so it remained a truss at all times.

Editing larger objects efficiently

To create larger objects, TrussFab offers a number of tools that create larger trusses in a single interaction, thus resulting in a more efficient design process.

The *beam* tool creates entire beams in one go. The bridge in Figure 1 and the pavilion in Figure 12 were created this way.

The *block* tool creates a tetra-octa plane in one go. We used it to create the roof of the pavilion in Figure 12. It can also serve, for example, as a stage.

The *deform* tool allows users to deform trusses. In Figure 12b we applied this tool in order to obtain a curved roof. Using the tool, we grabbed a hub located in the middle of the roof and dragged it upwards. The tool accommodates this by growing and shrinking members throughout the truss (see section "Implementation").
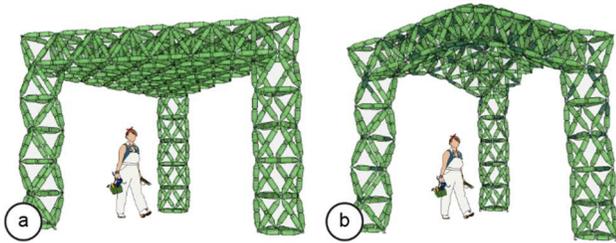

Figure 12: (a) Pavilion created using the *beam* and *block* tools. (b) The roof is freely deformed by pulling upwards using the *deform* tool.

Going even larger with facades

As mentioned earlier, single-layer triangle meshes, which as call *facades,* are an efficient way to add surfaces to a structure. TrussFab's editor offers two specific truss tools.

The *facade* tool allows filling in a facade between two trusses. This tool flood fills the plane in between two trusses with a triangle mesh. In Figure 6, we used this tool to create the walls of a tipi.

TrussFab supports a special type of facade, namely *domes* (Figure 13). Domes are particular in that they support themselves by means of their own curvature, i.e., *without* any underlying truss structure. We created the shown tent by (a) creating a dome by selecting TrussFab's *dome* tool and clicking on the ground. (b) We create an opening in the front using TrussFab's *delete* brush. (c) Using the *triangle* and the *line* tools we add the decorative ears on top of the dome. (d) The resulting dome is assembled from 512 bottles, 68 3D-printed and 63 laser cut hubs.

Figure 13: (a) Dome created using the *dome* tool. (b) We make the opening using the *delete* tool, and (c) add decoration using the *triangle* and *line* tools. (d) The resulting dome is built using 512 bottles.

HUBS AND MEMBERS

As mentioned earlier, TrussFab subjects hubs to loads in the range of a human weight, making their design crucial for achieving sturdy structures.

Each hub connects two or more bottle members by their necks. Consequently, each hub has two or more regions that hold a bottle; we call these regions *connectors*. We have designed two connector types that complement each other.

1. Threaded connectors hold bottles by their thread, as shown in Figure 14. To connect, users simply screw the respective bottle into the connector. Unfortunately, trusses cannot be assembled from threaded connectors alone. The reason is that screwing in the last member of a closed contour unscrews the bottle at the opposite end of that member. TrussFab therefore complements threaded connectors with a second type of connector that is not susceptible to rotation.

2. Snap-fit connectors *slide* into the bottleneck and hold the bottle from the inside (Figure 14). A three-way slit in the connector forms a set of cantilever springs that are compressed when the connector is inserted into a bottle, allowing the tip to slip in. When it reaches the point where the bottle widens, it expands and now resists being pulled out. To give the connector stability, users insert a pin into the connector, as shown in Figure 14, which locks the connector into place. Users can disassemble a snap-fit connector by pushing the pin all the way into the bottle.
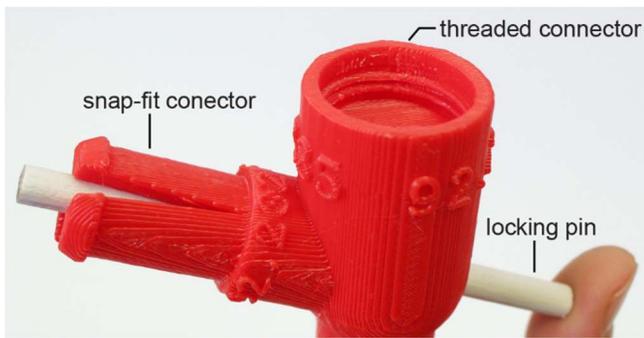
Figure 14: 3D printed hub with snap-fit and threaded bottle connectors.

Similar to the threaded connectors, it is not generally possible to implement a TrussFab structure using snap connectors alone. The reason is that if two snap connectors are on the same hub placed opposite to each other, it is impossible to insert both locking pins.

TrussFab resolves the challenge by using at least one snap connector per closed contour and by using threaded connectors opposite to any snap connector. TrussFab's editor resolves this automatically.

By default, TrussFab creates connectors that fit the bottle-necks of the most common bottles (*PCO 28mm* thread); other ready-made objects can be handled by loading an OpenSCAD description of the fitting connector design.

Creating members from bottles

As shown in Figure 15, TrussFab generally uses (a) long wood screws to connect the bottoms of two bottles or (b) double-ended screws, tightened by rotating the bottles in opposite directions. (c) In the rare case of short, single-bottle members, TrussFab uses bottom-to-hub wood screw connectors.
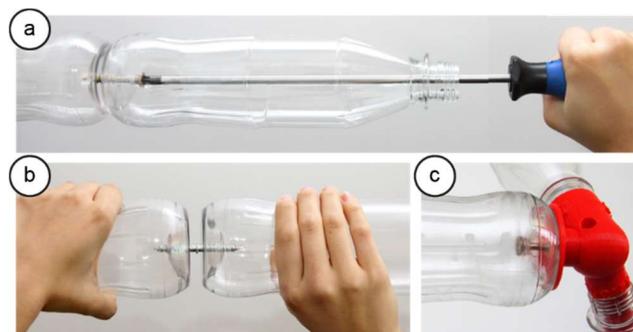

Figure 15: (a) Connecting bottles with a wood screw using an extra-long screwdriver and (b) with a double-ended screw. (c) Single-bottle edges require a bottom connector

We made all objects shown in this paper from *refillable* plastic bottles. Since these bottles are designed to be used multiple times, they feature thicker walls, resulting in sturdier structures.

For fabricating facades or non-load-bearing structures we connect bottles using 6" wide adhesive tape. This leaves the bottles intact, allowing us to return the bottles. It also works well with thinner, disposable bottles.

Stability and safety

Structures intended to carry a human weight need to undergo careful design. Before building, users should verify the stability of *their particular* bottle members and hubs using an appropriate testing procedure.

In order to determine the maximum load that our bottle members can undergo, we used the mechanical break testing machine shown in Figure 16a. We used the 3D-printed test connectors shown in Figure 16b to attach the bottles to the machine. The machine then applied increasing tension or compression, until the tested element breaks, resulting in the strain-stress diagram shown in Figure 16c.
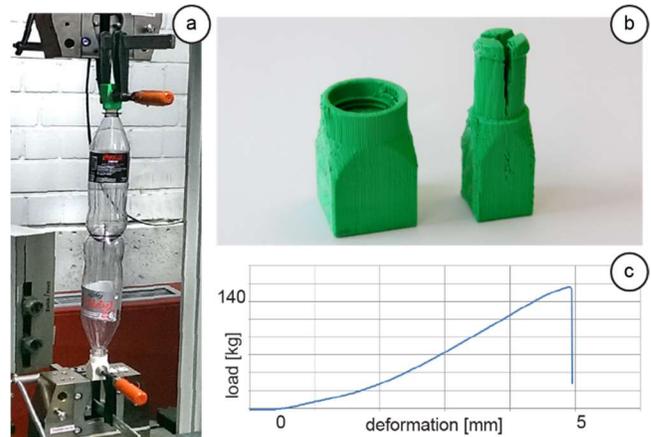

Figure 16: (a) During break strength testing of this truss member (b) the machine held the member by these square hubs. (c) Resulting strain-stress diagram shows that the member broke at 135 kg of tensile force.

Table 1 shows the loads *our* hubs and members withstand under idealized conditions, i.e., indoors and without any dynamic loads. For our members, we were able to apply up to 85kg of pressure (at which point the bottle buckles and collapsed) or 135 kg of tension (at which point the FDM-printed ABS hubs tore).

	threaded connector	snap connector	bottle member
compression	(any)	(any)	85 kg
tension	135 kg	145 kg	180 kg

Table 1: Breaking points of our threaded and snap connectors and bottle members.

Note that these measurements were obtained with refillable bottles—disposable bottles tend to break under smaller loads. Also slicer settings, print-orientations, and hub materials may lead to different results. Thus the testing procedure needs to be performed with the respective bottles and 3D printing technology at hand.

These measured values need to be complemented with additional factors that represent the expected dynamic loads and in the case of outdoor deployment also factors resulting from environmental conditions, such wind forces, wear and weather decay. Finally, a substantial factor for safety should be applied.

Laser-cut hubs for facades

Figure 17 shows the laser-cut connectors we use for facades. We tend to fabricate them from particleboard or optionally plywood for extra stability.
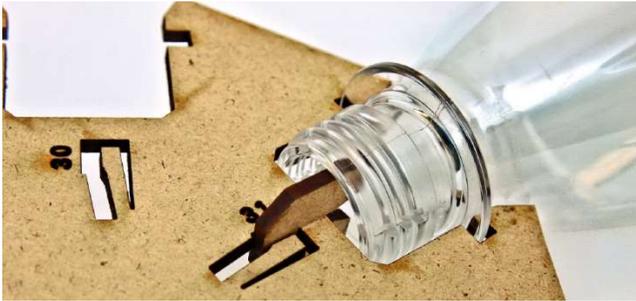

Figure 17: Laser-cut snap connectors are secured using a self-locking wedge.

Each laser-cut connector consists of two parts. The plate forms the hub itself. It is cut to accommodate the flange of the bottle, which prevents the bottle from moving in-and-out along its main axes. Inserting a wedge prevents the bottle from slipping out of the plane of the connector.

Even though hubs are flat when fabricated, assembling them into a curved structure, such as a dome (Figure 13), requires hubs to assume this curvature. TrussFab fabricates laser-cut connectors with play to allow for this.

IMPLEMENTATION

To help readers replicate out results, we now describe the implementation of the main components of the TrussFab system: TrussFab's editor, converter, force analyzer, and hub generator.

TrussFab editor

We implemented TrussFab as an extension to the 3D editor SketchUp. It is written in Ruby and JavaScript. It allows users to create 3D models, verify stability, and to trigger the TrussFab Hub Generator.

The grow and shrink tools affect the lengths of members and consequently the angles between members. TrussFab restores the consistency of the 3D model by running a dynamic relaxation algorithm [21], i.e., neighboring members start to push-pull each other until they find the position that accommodates the change. TrussFab iterates up to 10,000 cycles or until 0.1 mm accuracy has been reached.

TrussFab converter

TrussFab's converter offers two modes of operation: the volumetric and surface conversion.

The volumetric conversion procedure is similar to traditional voxelization methods. However, instead of intersecting the given 3D model with a regular cubical grid, TrussFab intersects the model with a tetrahedral honeycomb, as shown earlier on the example of a table in Figure 2. The algorithm also iterates to find those angles and positions that maximize the number of fully enclosed edges. Further elaborate space-filling algorithm can be found in Mitra et al. [19].

The surface conversion procedure reproduces the object's facade as members, as illustrated by Figure 18. The main challenge here is to ensure that every edge of the 3D model either fits the length of one of the bottle primitives or is slightly longer, in which case the converter will lengthen the edge by extending the respective connector.
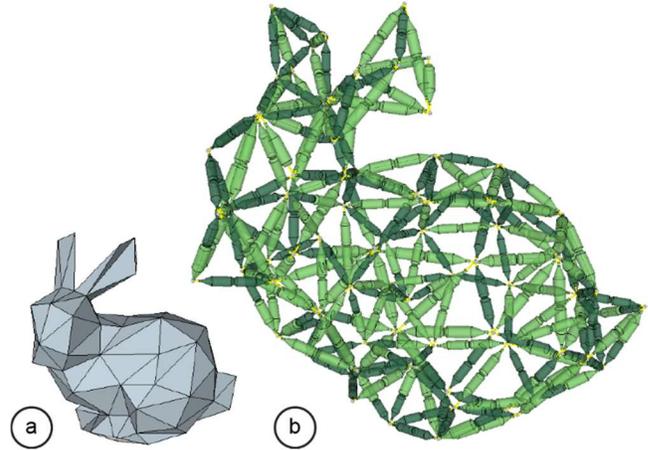

Figure 18: Stanford bunny converted using the TrussFab converter in *facade* conversion mode .

Our surface conversion tool, inspired by Richter and Alexa's *Beam Meshes* [24], consists of two stages: mesh simplification and surface remeshing. In the mesh simplification stage, we use the quadric-based edge collapse function in *MeshLab* [45] until it reaches the desired number of edges. We preserve certain features, such as the ears of the bunny in Figure 18, by manually simplifying the 3D model using the simplification brush in Autodesk *MeshMixer* [46].

In the surface remeshing stage, we optimize the vertex position of the model so that all edges are of the valid length of bottle primitives and the distortion of the final mesh is minimized. The energy function has two terms, where the first term is the minimum distance between an edge and the bottle primitives and the second term is the distance between the vertex position and the original simplified mesh.

More specifically, the energy function is of the form

$$E(\boldsymbol{V}) = \sum_{i=0}^{m} \min(\Sigma E_i - \boldsymbol{B}) + \alpha \sum_{i=0}^{n} Di \quad (v_i, \boldsymbol{S})$$

where $\boldsymbol{V}$ are the vertices of the simplified 3D model, n and m are the number of vertices and edges respectively, E_i is the length of the edge i, $\boldsymbol{B}$ is the set of valid length for all bottle primitives, $Dist(v_i, \boldsymbol{S})$ is the distance between vertex i and the surface $\boldsymbol{S}$ of the given 3D model. We calculate the optimized vertex positions using Powell's COBYLA optimization routine [22].

The algorithm does not account for structural stability; therefore, optional reinforcement needs to be added manually. Also, physical self-intersections need to be corrected by the user. The converted models are exported to the TrussFab editor in the form of a JSON file.

Finite elements

TrussFab uses *karamba3D* [23] as its finite element engine. This method models each edge as a spring of particular stiffness and calculates the displacement of the nodes under the given force [7]. TrussFab treats all hubs as ball joints, allowing for deformations without breaking. The bottle members are modelled as filled cylinders, which are rigid in shear. The pods touching the ground are considered anchor points.

TrussFab sends the geometry of the model together with the specified load forces to Karamba3D in JSON format, which returns the resulting compression and tension forces for each member.

Rigidity check

To check rigidity [25], TrussFab represents the 3D model as a node-link diagram. From this graph G TrussFab forms a rigidity matrix. If the rank of this matrix is equal $3n - 6$ where n is the number of vertices in G TrussFab considers the structure rigid.

To shortly explain this, consider a movement of the vertices given by specifying a velocity $\mu_i(t)$ for each vertex v_i at every point in time t. Let p_i be the initial position of v_i. Then the movement preserves the length of an edge $v_i v_j$, if and only if

$$\big(\mu_i(t) - \mu_j(t)\big) \cdot (p_i - p_j) = 0$$

holds for every point in time. Thus, to check G for rigidity, we can instead test whether velocities satisfying this equation for every edge exist. As each equation is linear, we obtain a system of linear equations. This system can be written as $A\mu = 0$ where μ is the vector of all velocities and each row of the matrix A corresponds to one equation. The matrix A is the above mentioned rigidity matrix. Note that μ has dimension $3n$ as we have one velocity for each vertex and each velocity is 3-dimensional. Thus, if $rank(A) = 3n - 6$ then the solution space of $A\mu = 0$ is 6-dimensional, which covers exactly the trivial movements of rotating (in three dimensions) and translating (in three dimensions) the whole graph. Hence, if $rank(A) = 3n - 6$, no other edge-length preserving movement can exist, i.e. G is rigid.

TrussFab Hub Generator

The TrussFab Hub Generator generates the 3D models of the hubs using the mathematical solid modeling tool OpenSCAD [47].

The TrussFab Hub Generator receives its input from the TrussFab editor in OpenSCAD script file format (*.scad*). (1) For 3D printed hubs, this data file describes each connector using a direction vector for each connection, annotated with connector type, elongation, and ID. (2) For laser-cut hubs, the plug-in projects the connections onto a plane before exporting the hub as a 2D geometry.

TrussFab Hub Generator generates hubs by arranging the individual connector primitives around a sphere. The connector geometry is loaded from separate modular files, allowing users to include their own, custom connector types for using different ready-made objects in the design.

Fabrication and assembly

We fabricated the 3D hubs of all models shown in this paper on MakerBot 2X desktop FDM printers. Each hub consumed about 50-150 g of filament, resulting in \$2-5 cost. One average hub printed in about 1.5-2.5 h, using a 0.5 mm nozzle. We mostly printed ABS, but also included recycled PET materials. The laser cut hubs are made from 5 mm particleboard, which took about 3 minutes to cut on a *Universal UL 150D* laser cutter.

Table 2 summarizes the bottle/hub count, printing and assembly time for all presented objects. The refillable bottles were acquired for their deposit value (\$0.15/piece).

	number of bottles	number of hubs	printing time	assembly time
chair	36	8	~16 h	~10 min
table	48	10	~20 h	~20 min
boat	124	31	~62 h	~2 h
dome	512	68 (3D) 63 (2D)	~136 h ~3 h	~8 h (2 person)
bridge	174	30	~60 h	~4 h

Table 2: Summary of bottle/hub count, printing and assembly time per example object.

CONCLUSION

TrussFab is an integrated end-to-end system that allows users to fabricate large structures that are sturdy enough to carry human weight on desktop 3D printers. Unlike previous systems that built on up-cycled plastic bottles combined with 3D print, TrussFab considers bottles not as "bricks", but as beams that form structurally sound node link structures also known as *trusses*, allowing users to handle the forces resulting from scale and load. TrussFab embodies the required engineering knowledge, allowing non-engineers to design such structures and allows users to validate their designs using integrated structural analysis.

Personal fabrication of large-scale objects opens up a range of new challenges. Unlike for desktop-scale objects, software systems need to consider how to (1) assure structural integrity to guarantee safety. (2) Consider dynamic forces, such as human movements and wind forces. (3) Use material consciously. (4) Optimize fabrication time. (5) Take into account the effect of environmental factors on longevity, such as temperature, weather, UV light, etc.

REFERENCES
1. Harshit Agrawal, Udayan Umapathi, Robert Kovacs, Johannes Frohnhofen, Hsiang-Ting Chen, Stefanie Mueller, and Patrick Baudisch. 2015. Protopiper: Physically Sketching Room-Sized Objects at Actual Scale. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*

(UIST '15), ACM, New York, NY, USA, 427–436. http://doi.org/10.1145/2807442.2807505

2. Daniel Ashbrook, Shitao Guo, and Alan Lambie. 2016. Towards Augmented Fabrication: Combining Fabricated and Existing Objects. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '16)*, ACM, New York, NY, USA, 1510–1518. http://doi.org/10.1145/2851581.2892509

3. Xiang "Anthony" Chen, Stelian Coros, Jennifer Mankoff, and Scott E. Hudson. 2015. Encore: 3D Printed Augmentation of Everyday Objects with Printed-Over, Affixed and Interlocked Attachments. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology (UIST '15)*, ACM, New York, NY, USA, 73–82. http://doi.org/10.1145/2807442.2807498

4. Kenneth C Cheung and Neil Gershenfeld. 2013. Reversibly Assembled Cellular Composite Materials. *Science* 341, 6151: 1219–1221. http://doi.org/10.1177/0892705714554493

5. Stelian Coros, Bernhard Thomaszewski, Gioacchino Noris, Shinjiro Sueda, Moira Forberg, Robert W. Sumner, Wojciech Matusik, and Bernd Bickel. 2013. Computational Design of Mechanical Characters. *ACM Transactions on Graphics* 32, 4: 1. http://doi.org/10.1145/2461912.2461953

6. Laura Devendorf and Kimiko Ryokai. 2015. Being the Machine: Reconfiguring Agency and Control in Hybrid Fabrication. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*, ACM, New York, NY, USA, 2477–2486. http://doi.org/10.1145/2702123.2702547

7. Jacob Fish and Ted Belytschko. 2007. *A first course in finite elements*. Wiley New York.

8. Ollé Gellért. Print To Build, 3D printed joint collection. Retrieved September 15, 2016 from https://www.behance.net/gallery/27812109/Print-To-Build-3D-printed-joint-collection

9. Scott E. Hudson. 2014. Printing Teddy Bears. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*, ACM, New York, NY, USA, 459–468. http://doi.org/10.1145/2556288.2557338

10. Yuki Igarashi, Takeo Igarashi, and Jun Mitani. 2012. Beady: interactive beadwork design and construction. *ACM Transactions on Graphics (TOG)* 31, c: 49. http://doi.org/10.1145/2185520.2185545

11. Sasa Jokic and Petar Novikov. Mataerial - A Radical New 3D Printing Method. Retrieved September 15, 2016 from http://www.mataerial.com/

12. Sasa Jokic, Petr Novikov, Shihui Jin, Stuart Maggs, Cristina Nan, and Dori Sadan. Minibuilders: Robots for 3D printing in construction and design. Retrieved September 15, 2016 from http://robots.iaac.net/

13. Behrokh Khoshnevis. 2004. Automated Construction by Contour Crafting—Related Robotics and Information Technologies. *Automation in Construction* 13, 1: 5–19. http://doi.org/10.1016/j.autcon.2003.08.012

14. Benjamin Lafreniere, Marcelo H. Coelho, Nicholas Cote, Steven Li, Andy Nogueira, Long Nguyen, Tobias Schwinn, James Stoddart, David Thomasson, Ray Wang, Thomas White, Tovi Grossman, David Benjamin, Maurice Conti, Achim Menges, George Fitzmaurice, Fraser Anderson, Justin Matejka, Heather Kerrick, Danil Nagy, Lauren Vasey, Evan Atherton, and Nicholas Beirne. 2016. Crowdsourced Fabrication. In *Proceedings of the 29th Annual ACM Symposium on User Interface Software & Technology (UIST '16)*, 15–28. http://doi.org/10.1145/2984511.2984553

15. Tien T. Lan. 2005. *Structural Engineering Handbook - Space Frame Structures*. CRC Press.

16. Manfred Lau, Akira Ohgawara, Jun Mitani, and Takeo Igarashi. 2011. Converting 3D Furniture Models to Fabricatable Parts and Connectors. *ACM Transactions on Graphics* 30, 212: 1–6. http://doi.org/10.1145/1964921.1964980

17. Linjie Luo, Ilya Baran, Szymon Rusinkiewicz, and Wojciech Matusik. 2012. Chopper: Partitioning Models into 3D-Printable Parts. *ACM Transactions on Graphics* 31, 6: 1. http://doi.org/10.1145/2366145.2366148

18. Michael Makris, David Gerber, Anders Carlson, and Doug Noble. 2013. Informing Design through Parametric Integrated Structural Simulation. In *eCAADe 2013: Computation and Performance– Proceedings of the 31st International Conference on Education and research in Computer Aided Architectural Design in Europe*, Delft University of Technology, 69–77.

19. Niloy J Mitra and Mark Pauly. 2009. Shadow art. *ACM Transactions on Graphics* 28, 5: 1. http://doi.org/10.1145/1618452.1618502

20. Stefanie Mueller, Sangha Im, Serafima Gurevich, Alexander Teibrich, Lisa Pfisterer, François Guimbretière, and Patrick Baudisch. 2014. WirePrint: 3D Printed Previews for Fast Prototyping. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software & Technology (UIST '14)*, ACM, New York, NY, USA, 273–280. http://doi.org/10.1145/2642918.2647359

21. Joseph Reuben Harry Otter, Alfred Carlo Cassell, and Roger Edwin Hobbs. 1966. Dynamic Relaxation. In *Proceedings of the Institution of Civil Engineers* 35, 4: 633–656. http://doi.org/10.1680/iicep.1966.8604

22. Michael J. D. Powell. 1964. An Efficient Method for

Finding the Minimum of a Function of Several Variables without Calculating Derivatives. *The computer journal*: 155–162. http://doi.org/10.1093/comjnl/7.2.155

23. Clemens Preisinger. Karamba3D - Parametric Structural Modeling. Retrieved March 15, 2016 from http://www.karamba3d.com/

24. Ronald Richter and Marc Alexa. 2015. Beam Meshes. *Computers & Graphics*, 1: 1–8. http://doi.org/10.1016/j.cag.2015.08.007

25. B. Roth. 1981. Rigid and Flexible Frameworks. *Mathematical Association of America* 88, 1: 6–21.

26. Greg Saul, Manfred Lau, Jun Mitani, and Takeo Igarashi. 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 (TEI '11)*, ACM, New York, NY, USA, 73. http://doi.org/10.1145/1935701.1935717

27. Rita Shewbridge, Amy Hurst, and Shaun K. Kane. 2014. Everyday Making. In *Proceedings of the 2014 conference on Designing interactive systems (DIS '14)*, ACM, New York, NY, USA, 815–824. http://doi.org/10.1145/2598510.2598544

28. Melina Skouras, Stelian Coros, Eitan Grinspun, and Bernhard Thomaszewski. 2015. Interactive Surface Design with Interlocking Elements. *ACM Transactions on Graphics* 34, 6: 224. http://doi.org/10.1145/2816795.2818128

29. Jeffrey Smith, Jessica Hodgins, Irving Oppenheim, and Andrew Witkin. 2002. Creating Models of Truss Structures with Optimization. *ACM Transactions on Graphics*. 21, 3: 295–301. http://doi.org/10.1145/566654.566580

30. Joshua G. Tanenbaum, Amanda M. Williams, Audrey Desjardins, and Karen Tanenbaum. 2013. Democratizing Technology: Pleasure, Utility and Expressiveness in DIY and Maker Practice. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*: 2603–2612. http://doi.org/10.1145/2470654.2481360

31. Nobuyuki Umetani, Takeo Igarashi, and Niloy J. Mitra. 2012. Guided exploration of physically valid shapes for furniture design. *ACM Transactions on Graphics* 31, 4: 1–11. http://doi.org/10.1145/2185520.2335437

32. Weiming Wang, Tuanfeng Y. Wang, Zhouwang Yang, Ligang Liu, Xin Tong, Weihua Tong, Jiansong Deng, Falai Chen, and Xiuping Liu. 2013. Cost-effective Printing of 3D Objects with Skin-frame Structures. *ACM Transactions on Graphics* 32, 6: 1–10. http://doi.org/10.1145/2508363.2508382

33. Christian Weichel, Manfred Lau, David Kim, Nicolas Villar, Hans W. Gellersen, Christian Weichel, Manfred Lau, David Kim, Nicolas Villar, and Hans W. Gellersen. 2014. MixFab: A Mixed-reality Environment for Personal Fabrication. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*, 3855–3864. http://doi.org/10.1145/2556288.2557090

34. Karl Willis, Eric Brockmeyer, Scott Hudson, and Ivan Poupyrev. 2012. Printed Optics. In *Proceedings of the 25th annual ACM symposium on User interface software and technology (UIST '12)*, 589. http://doi.org/10.1145/2380116.2380190

35. Jan Willmann, Federico Augugliaro, Thomas Cadalbert, Raffaello D'Andrea, Fabio Gramazio, and Matthias Kohler. 2012. Aerial Robotic Construction Towards a New Field of Architectural Research. *International Journal of Architectural Computing* 10, 3: 439–460. http://doi.org/10.1260/1478-0771.10.3.439

36. Suguru Yamada, Hironao Morishige, Hiroki Nozaki, Masaki Ogawa, Takuro Yonezawa, and Hideyuki Tokuda. 2016. ReFabricator: Integrating Everyday Objects for Digital Fabrication. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '16)*, ACM, New York, NY, USA, 3804–3807. http://doi.org/10.1145/2851581.2890237

37. Hironori Yoshida, Syunsuke Igarashi, Takeo Igarashi, Yusuke Obuchi, Yosuke Takami, Jun Sato, Mika Araki, Masaaki Miki, Kosuke Nagata, Kazuhide Sakai, Kyungeun Sung, and Tim Cooper. 2015. Architecture-scale Human-assisted Additive Manufacturing. *ACM Transactions on Graphics* 34, 4: 88:1-88:8. http://doi.org/10.1145/2766951

38. Henrik Zimmer and Leif Kobbelt. 2014. Zometool Rationalization of Freeform Surfaces. *IEEE Transactions on Visualization and Computer Graphics* 20, 10: 1461–1473. http://doi.org/10.1109/TVCG.2014.2307885

39. Homes Made from Plastic Bottles. Retrieved September 15, 2016 from http://www.inspirationgreen.com/plastic-bottle-homes

40. Divergent3D: The World First 3D Printed Super Car. Retrieved September 15, 2016 from http://www.divergent3d.com/

41. Trimble SketchUp. Retrieved March 15, 2016 from http://www.sketchup.com/

42. SkyCiv cloud engineering software. Retrieved December 16, 2016 from https://skyciv.com/

43. MiTek-PAMIR Software. Retrieved December 16, 2016 from http://www.mitek.co.uk/PAMIR/

44. GlobalTruss TRUSSTOOL. Retrieved December 16, 2016 from https://trusstool.com/

45. MeshLab. Retrieved December 16, 2016 from http://meshlab.sourceforge.net

46. Autodesk - MeshMixer. Retrieved December 16, 2016 from http://meshmixer.com

47. Open SCAD. Retrieved December 16, 2016 from http://openscad.org