TrussFormer: 3D Printing Large Kinetic Structures

Robert Kovacs[1], Alexandra Ion[1], Pedro Lopes[1], Tim Oesterreich[1], Johannes Filter[1], Philip Otto[1], Tobias Arndt[1], Nico Ring[1], Melvin Witte[1], Anton Synytsia[2], and Patrick Baudisch[1]

[1]Hasso Plattner Institute,
University of Potsdam, Germany
{firstname.lastname}@hpi.uni-potsdam.de

[2]College of Engineering,
Oregon State University, USA
anton.synytsia@gmail.com

ABSTRACT

We present TrussFormer, an integrated end-to-end system that allows users to 3D print large-scale *kinetic* structures, i.e., structures that involve motion and deal with dynamic forces. TrussFormer builds on TrussFab, from which it inherits the ability to create static large-scale truss structures from 3D printed connectors and PET bottles. TrussFormer adds movement to these structures by placing linear actuators into them: either manually, wrapped in reusable components called assets, or by demonstrating the intended movement. TrussFormer verifies that the resulting structure is mechanically sound and will withstand the dynamic forces resulting from the motion. To fabricate the design, TrussFormer generates the underlying hinge system that can be printed on standard desktop 3D printers. We demonstrate TrussFormer with several example objects, including a 6-legged walking robot and a 4m-tall animatronics dinosaur with 5 degrees of freedom.

Author Keywords

Fabrication; 3D printing; variable geometry truss; large scale mechanism.

INTRODUCTION

Personal fabrication tools [3], such as 3D printers, afford rapid prototyping [21] and empower non-experts with the ability to fabricate interactive objects [14]. The latter includes animated objects, such as kinematic animals [9] or actuated paper origami [23], and simple machines [25].

More recently, HCI researchers have started to explore how to enable non-expert users to fabricate *large-scale* structures. While professional users may have access to large-scale 3D printing equipment [16], non-experts are generally limited to the use of desktop 3D printers, causing these systems to achieve scale by combining 3D print with ready-made objects, such as empty plastic bottles [17]. The resulting systems also support users in creating structures capable of dealing with the substantial forces such structures are subject to. *TrussFab* [17], for example, achieves this by allowing users to combine already sturdy primitives and by checking stability during editing.

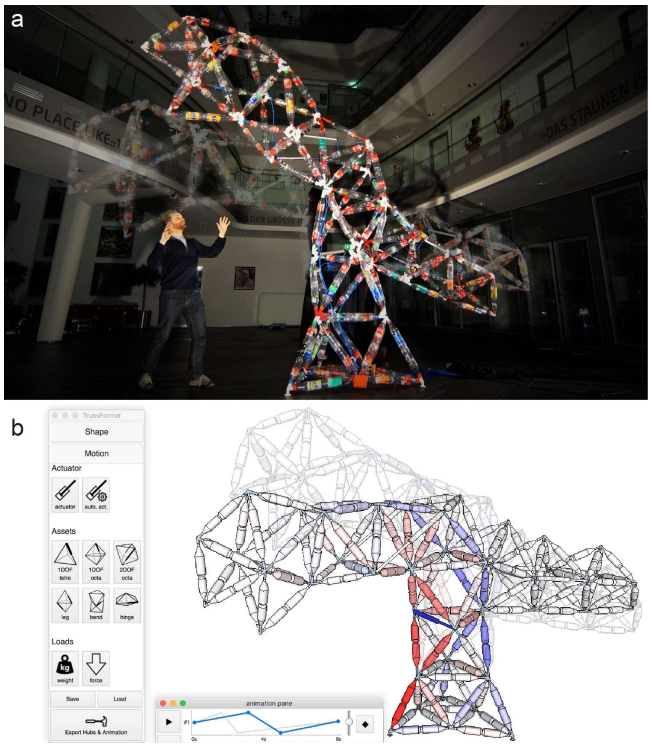

Figure 1: (a) TrussFormer is an end-to-end system that allows users to design and 3D print large-scale kinetic truss structures that deform and animate. (b) TrussFormer verifies that the designed structure can handle the forces resulting from its motion, as shown on this animatronics 4m tall T-Rex.

While large-scale fabrication systems like TrussFab have been shown to support a wide range of applications, from furniture to tradeshow pavilions, such systems are limited to creating static structures.

In this paper, we present a system that allows users to create large *kinetic* structures, i.e., structures that involve motion and deal with dynamic forces, as they occur as part of animatronics devices, such as the animated Tyrannosaurus Rex, illustrated by Figure 1, and other large-scale machinery. TrussFormer embodies the required engineering knowledge from creating the appropriate mechanism, verifying its

UIST '18, October 14–17, 2018, Berlin, Germany

ACM ISBN 978-1-4503-5948-1/18/10…$15.00
https://doi.org/10.1145/3242587.3242607

structural soundness, and generating the underlying hinge system printable on desktop 3D printers.

TRUSSFORMER

TrussFormer helps users to create the shape and design the motion of large-scale kinetic structures. It does this by incorporating linear actuators into rigid truss structures in a way that they move "organically", i.e., hinge around multiple points at the same time. These structures are also known as variable geometry trusses [1]. Figure 2 illustrates this on the smallest elementary truss, (a) the rigid tetrahedron. (b) We swap one of the edges with a linear actuator, (c) resulting in a variable geometry truss. The only required change for this is to introduce hubs that enable rotation at the nodes. We call these *hinging hubs*.

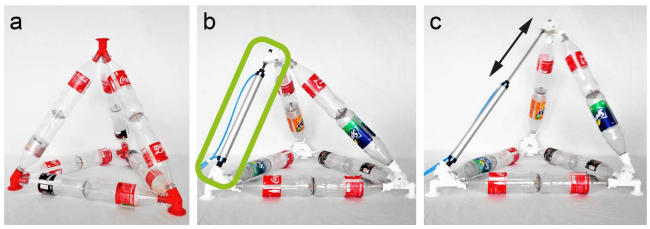

Figure 2: (a) The static tetrahedron (b-c) is converted into a deformable structure by swapping one edge with a linear actuator. The only required change is to introduce connectors that enable rotation.

This simple approach to create variable geometry truss mechanisms scales well to arbitrary larger structures. Our T-Rex model from Figure 1 contains five linear actuators and thus offers five degrees of freedom (DoF). It can (a) lift or lower its neck (1 DoF), (b) turn its head left and right (1 DoF), (c) sweep its tail (2 DoF), and (d) open its mouth (1 DoF), as shown in Figure 3.

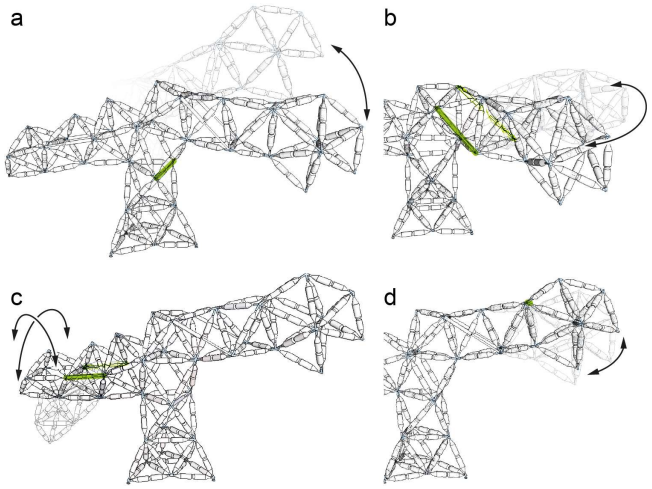

Figure 3: The T-Rex offers 5 degrees of freedom.

In the following, we demonstrate how TrussFormer allows non-expert users to create such structures in six steps.

Step 1: Creating a static structure

As shown in Figure 4, this particular model was created by first modeling the T-Rex as a static structure in TrussFormer. Our editor's ability to create static structures is based on TrussFab [17]: users design the shape of their T-Rex using structurally stable primitives (tetrahedra and octahedra).

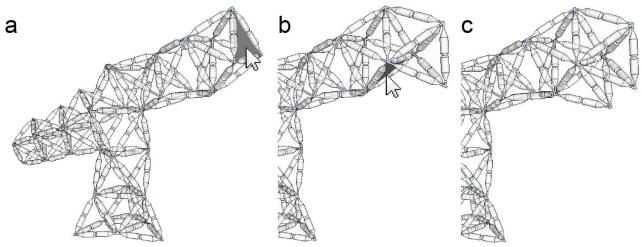

Figure 4: Modeling the static shape of the T-Rex. Here, the user creates the jaws of the T-Rex by attaching tetrahedron primitives through the steps (a, b, c).

Step 2: Adding movement

To add movement to the static structure, users select the *demonstrate movement* tool and pull the T-Rex head downwards, as shown in Figure 5. TrussFormer responds by placing an actuator that turns the T-Rex body into a structure that organically moves and bends down. Together with the *Demonstrate movement* tool, TrussFormer provides three different approaches to animating structures, ranging from this (1) automated placement (for novice users), through (2) placing elements with predefined motion, called *assets*, to (3) manual placement (as users acquire engineering knowledge). We discuss these in section "Adding motion to the structure".

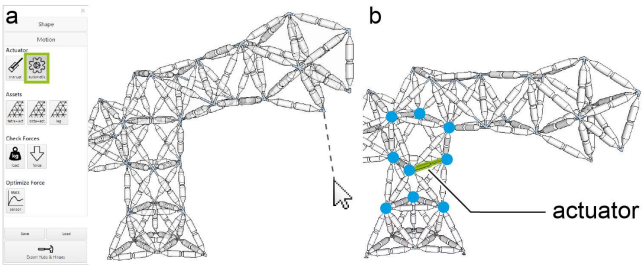

Figure 5: (a) The user selects the *demonstrate movement tool* and pulls the T-Rex head downwards. (b) TrussFormer responds by adding an actuator to the T-Rex body so that it is capable of performing this type of motion. At this point the system also places 9 hinging hubs to enable this motion (marked with blue dots).

Step 3: Stability check across poses

During this step, TrussFormer also verifies that the mechanism is structurally sound. In the background, TrussFormer finds the safe range of expansion and contraction of the placed actuator by simulating the occurring forces in a range of positions. If there is a pose where the forces exceed the pre-determined breaking limits or the structure would tip over, TrussFormer sets the limits for the actuator so it will not extend beyond them. This check prevents users from producing invalid configurations.

Step 4: Animation

To animate the structure users open the *animation pane* in the toolbar, as shown in Figure 6. First, they control the movement of the structure manually using sliders, to try out

the movement. When they find the desired pose, they simply add it as a keyframe to the animation timeline. With this TrussFormer allows users to orchestrate the movement of all actuators using a simple timeline/keyframe editor. In Figure 6 we program a "feeding" behaviour, where the T-Rex opens its mouth while reaching down and waving its tail.

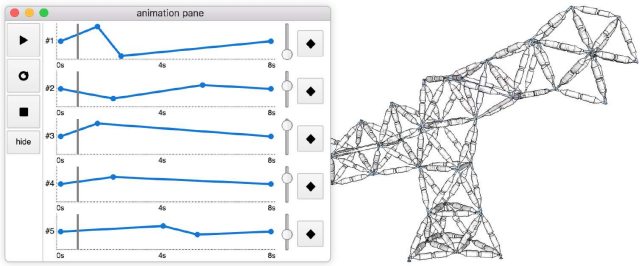

Figure 6: Animating the structure. Users sets the desired pose using the sliders in the animation pane and orchestrates the movement by placing key-frames on the timeline.

Step 5: Checking forces during the motion

Once a movement has been defined, TrussFormer computes the *dynamic* forces. As shown in Figure 7a, the user creates an animation that moves the T-Rex body up and down. (b) TrussFormer computes the forces while T-Rex's body comes back up quickly after dipping down; the large acceleration of the long neck leads to very high inertial forces, exceeding the breaking limit of the construction, (c) causing the structure to fail at the indicated time point. These situations are hard to foresee, because the *inertial forces* can be multiple times higher than the static load in the structure. (d) TrussFormer addresses this by automatically correcting the animation sequence by either limiting the acceleration or the range of the movement, assuring that the structure will now withstand the movement.

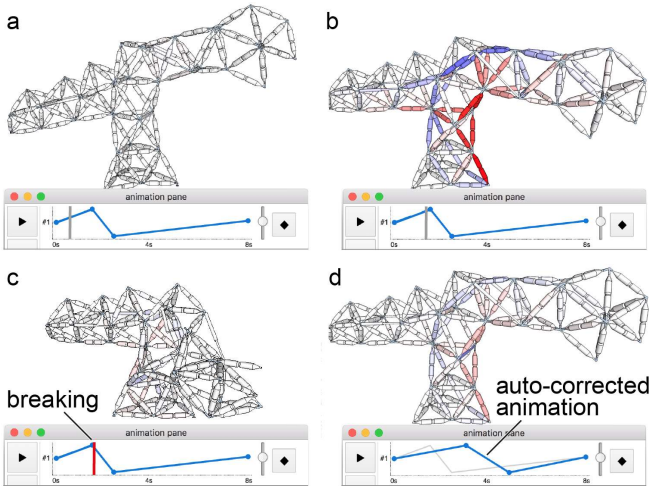

Figure 7: Verifying the inertial forces: (a-b) The forces are increasing with the acceleration of the structure. (c) The structure breaks when the direction of the movement changes rapidly. (d) TrussFormer resolves this by making the movement slower.

Step 6: Fabrication

When users are satisfied with their design (structure, movement and animation), they click the *fabricate* button, shown in Figure 8a. This invokes (1) TrussFormer's hinge generation algorithm, which analyzes the structure's motion and generates the appropriate 3D printable hinge and hub geometries, annotated with imprinted IDs for assembly. In the case of the T-Rex, the system exports 42 3D printed hubs, consisting of 135 unique hinging pieces. (2) Next, TrussFormer exports the created animation patterns as Arduino code that users upload to their microcontroller. (3) Lastly, it outputs a specification, containing the force, speed, and motion range of the actuators, in order to achieve the desired animation pattern. Users find these actuators as standardized components.

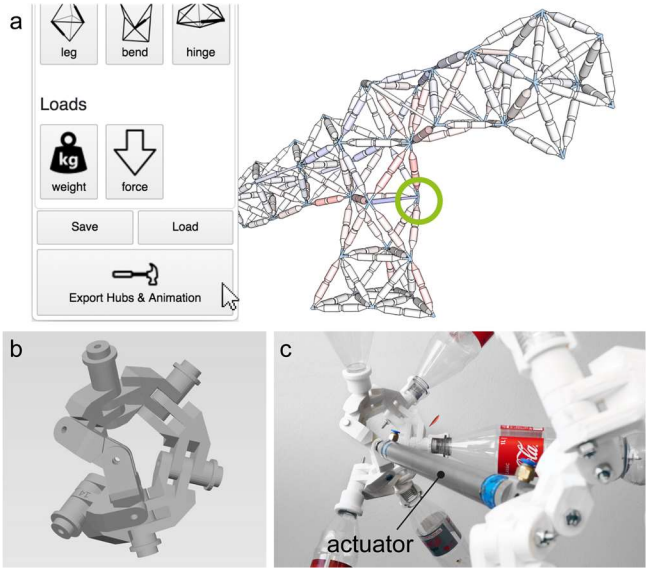

Figure 8: (a) To fabricate our T-Rex model, TrussFormer exports: (b) the appropriate 3D printable hinging-hubs, (c) and the specifications for the actuators that inform the users which one to buy. TrussFormer also exports the animation sequence for an Arduino.

CONTRIBUTION, BENEFITS, AND LIMITATIONS

Our main contribution is TrussFormer: an end-to-end system that enables non-expert users to create large-scale kinetic structures, such as the devices used in large-scale animatronics.

TrussFormer helps users in the 3 main steps along the design process. (1) It enables users to animate large truss structures by adding linear actuators to them. It offers three tools for this purpose: manual actuator placement, placement of assets performing predefined motion, and creating motion by demonstration. (2) TrussFormer validates the design in real time against static forces, static forces across all poses, and dynamic forces. (3) TrussFormer automatically generates the necessary 3D printable hinges for fabricating the structure. Its algorithm determines the placement and configuration of the hinges and their exact dimensions.

To validate our system, we created a series of example objects, including a 6-legged walking robot and a 4m-tall animatronics dinosaur with five degrees of freedom, comprising of 17 static and 25 hinging hubs.

TrussFormer is subject to some limitations. (1) TrussFormer's simulation relies on the Newton Dynamics physics engine [52], which offers only limited accuracy for engineering purposes. Higher precision could be achieved by replacing Newton Dynamics with a better physics engine (e.g., [7]). (2) When deployed, TrussFormer should be provided with additional safety features, such as the option to use stronger materials and additional safety margins in the computation.

WORKING PRINCIPLE BEHIND TRUSSFORMER'S KINETIC STRUCTURES

Before we discuss how TrussFormer allows users to define motion of the structure, we explain where actuators can be placed inside the truss structure and how their motion propagates.

A structure created in TrussFormer consists of unit cells, which can be tetrahedra or octahedra. Each cell can contain one or more linear actuators. When actuated, the actuators change the geometry of the cell and thus move the entire structure.

First, as an example, Figure 9 illustrates how inserting an actuator affects only its surrounding. (a) One way of thinking of the actuated tetrahedron is as a rotary hinge, with a triangle face at each side (shaded in gray). (b) Such structures can be extended by attaching a rigid structure to each of the two faces (here two octahedra). As a result the two structures are hinging around each other. Since the motion is localized, this type of actuator placement is intuitively graspable.

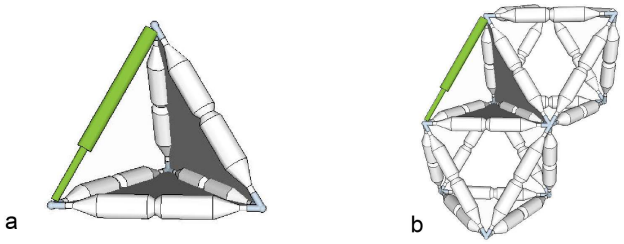

Figure 9: Attaching rigid primitives to (a) faces that do not contain an actuator (b) results in simple structures with only localized deformation. This structure acts as a hinge between the two octahedra.

Second, as illustrated by Figure 10, (a) we can produce more complex kinetic structures by attaching rigid primitives to the faces that contain an actuator (e.g., the one shaded in gray). (b) Now, the newly placed primitive will also contain this actuator and therefore the result is a structure that moves in whole, resulting in more complex behavior.

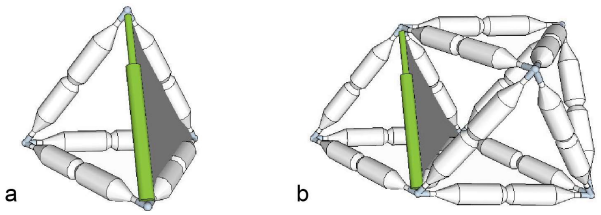

Figure 10: Attaching a rigid primitive (here an octahedron) to a face that contains an actuator results in a larger deforming structure.

The third way to propagate motion is to build structures where the cells are interconnected through two or more moving faces. Figure 11a shows an octahedron with one actuator in it. (b) We attach two tetrahedra to the marked faces and place a second octahedron in between them. Since the two original connecting faces are moving with respect to each other, the two tetrahedra are moving as well, causing the second octahedra to deform. The second octahedra requires removing one arbitrary edge (here on the top) to allow for deformation. (c) Applying this principle users can propagate the movement of one actuator through the truss.

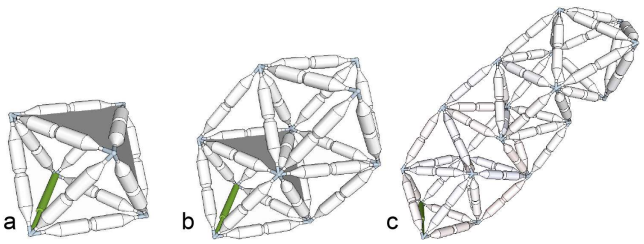

Figure 11: The motion caused by one actuator propagates throughout the entire truss beam, making it bend.

ADDING MOTION TO THE STRUCTURE

TrussFormer offers three ways for users to animate their structures: (1) by *demonstrating* the desired movement, as we discussed in our walkthrough, (2) using elements with predefined motion, which we call *assets*, and (3) by placing actuators *manually*.

The first two strategies are better suited for novice users, since they do not require knowledge about the mechanism, but rather focus on the shape of the structure and the movement they want to achieve. The third option is best suited for users with more experience, who have already gained a deeper understanding into variable geometry trusses.

1. Automatic actuator placement by demonstration

As we briefly discussed in the walkthrough section and in Figure 5, TrussFormer enables users to create moving structures by offering automatic actuator placement. Users can focus on only designing the shape of their structure first. Then, they invoke the *demonstrate movement* tool and drag the static structure in the direction they want it to move. TrussFormer then replaces the edge with an actuator at the position which best satisfies the movement.

To identify which edge should be replaced with an actuator TrussFormer runs an exhaustive search by virtually replacing every member with an actuator one by one. At every replacement it moves the actuator while measuring if the structure moved closer or further to the desired target position. Finally, it compares all the results and selects the actuator that produced the closest motion. A limitation of this simplistic method is that it works by naïve approximation, i.e., that it does not guarantee that the desired position will be exactly reached. To improve these results, further optimization algorithms can be utilized, similarly as demonstrated by Coros et al. [9] for planar mechanisms.

2. Creating kinetic structures based on assets

Because the resulting motion of variable geometry trusses tends to be hard to predict, TrussFormer encapsulates them into predefined sub-assemblies, which we call *assets*. Assets connect to the existing geometry through a dedicated triangle surface. This results in structures that contain the asset's movement which is localized and thus easy to understand.

Figure 12 shows a selection of assets. The triangles marked in gray are their connectors, i.e., the side that connects to existing geometry when the asset is added to a structure. TrussFormer offers a basic selection of assets, however, users can easily create their own asset library by saving a custom asset into an asset folder.

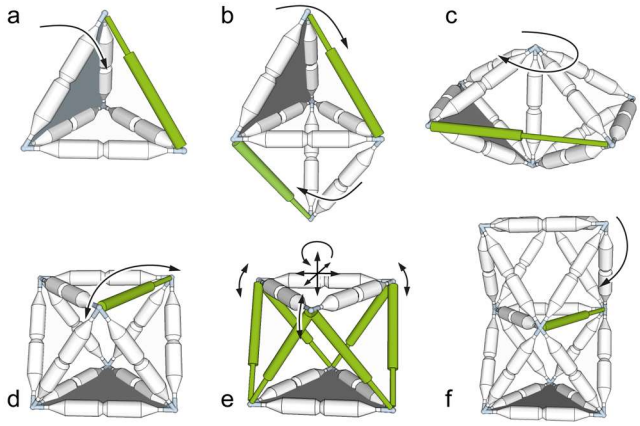

Figure 12: A selection of assets: (a) tetrahedron with 1DoF, (b) "robotic leg" asset, (c) hinging tetrahedra, (d) octahedron with 1DoF, (e) Stewart platform (6DoF), and (f) double-octahedron performing "bending" motion.

Figure 13 illustrates the workflow enabled by assets: a simple walking robot with six robotic legs. (a) Users start by creating the rigid body of the robot from tetrahedra and octahedra blocks. They design it to offer six connector faces, i.e., three on each side, (b) where they attach copies of the *robotic leg* asset, shown in Figure 12b. (c) This results in an autonomous walking structure.

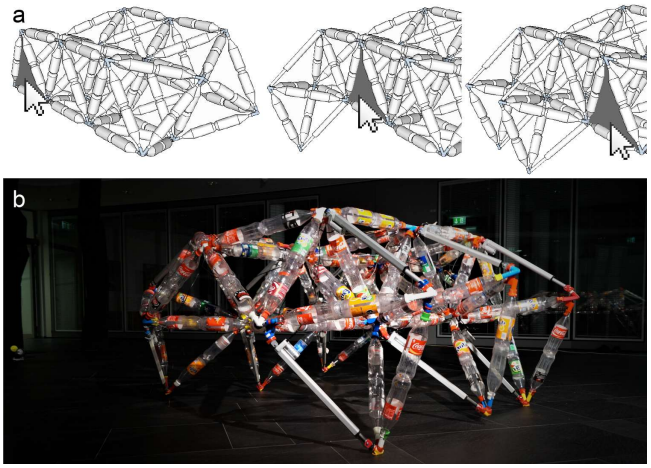

Figure 13: (a) Users start the design by making the body of the walking robot. The predesigned 2DoF "leg" asset is added to the side triangles 6 times. (b) The fabricated robot.

The concept of assets is useful beyond the use of actuators. Figure 14 for example, shows a bike we designed around the *hinge* asset that forms the steering column.

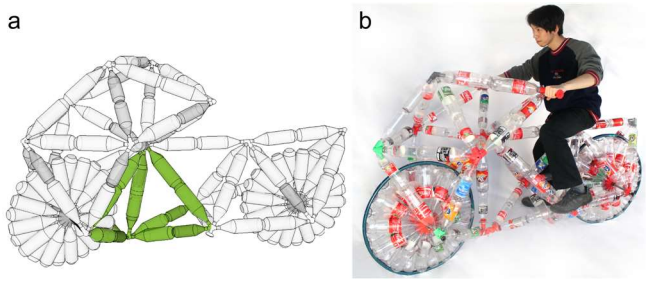

Figure 14: This bike's steering column is based on the hinge asset, which is used without the actuator in this example.

3. Manual actuator placement

As users gain expertise in creating variable geometry trusses, they may prefer to place actuators directly into their structures. TrussFormer's *turn edge to actuator* tool allows users to transform rigid edges into actuators by simply clicking on them, as illustrated by Figure 15.

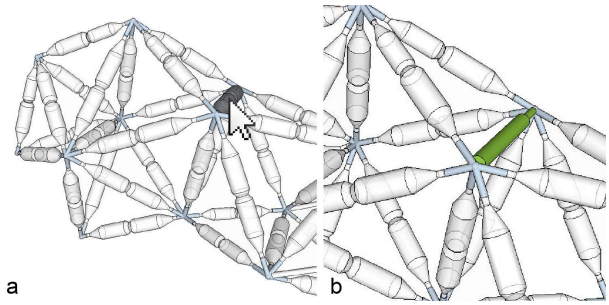

Figure 15: The *turn edge to actuator* tool allows users to turn any edge into an actuator. Here user replaces one edge in the T-Rex's head to make its jaws move.

We designed this tool deliberately as a "turn existing edge into actuator" tool and not as a "place new actuator" tool. Normally, placing a new actuator edge into an already rigid

structure would not allow for movement, however, by turning an existing edge into an actuator, users are essentially adding a degree of freedom to the structure. If user

VERIFYING AND ADAPTING FORCES

Our system helps users to create the shape and the motion of large-scale kinetic structures. To accomplish this, it helps users handle the dynamic forces that occur when large structures move, such as the T-Rex in Figure 1.

TrussFormer enables users to (1) constantly monitor the forces that occur within the structure at interactive rates. Furthermore, it (2) validates the poses of the structure and adapts the motion range of the actuators to not damage the model and (3) automatically adapts the user-defined animation sequence in case it breaks the structure.

To perform these tasks, TrussFormer takes into account the breaking limits of the building materials. The model is considered broken when the simulated peak stress value exceeds the entered breaking limit of the building material. We acquired these values from fracture testing the materials, in our case the plastic bottles, as described in TrussFab [17], resulting in max.85 kg compression and max.135 kg tension. If users decide to use different building materials, we recommend testing the forces these elements can withstand again. However, we expect users to share this information on platforms such as *thingiverse.com*.

1. Constantly visualizing forces during animations

While the user animates the structure, TrussFormer is continuously simulating the forces using its built-in physical simulation. The forces are visualized as colored edges: red indicates compression, blue indicates tension, while the saturation signalizes the intensity of the force.

This allows users not only to preview artifacts that arise from their current animation, e.g., the structure wobbling too much due to rapid changes from pose to pose; but, more importantly, it allows them to preview how the stress is distributed in the structure and even foresee breaking points when rapidly actuated.

2. Validating possible poses

After users have placed an actuator in their structure, TrussFormer automatically determines their motion range, i.e., how far can it expand without damaging the structure.

Figure 16 shows that the structure can break due to various causes, such as the structure falling over, hitting the ground from too high of a movement allowance, or simply exerting too much force on another structural element (e.g., an edge).

To determine the limits of an actuator, TrussFormer iteratively increases the expansion until the simulated model breaks. TrussFormer then stores the previous valid expansion as the maximum length for that actuator. This value is then set as the upper bound for the motion in the keyframe editor. This way the user is never able to over-actuate them.

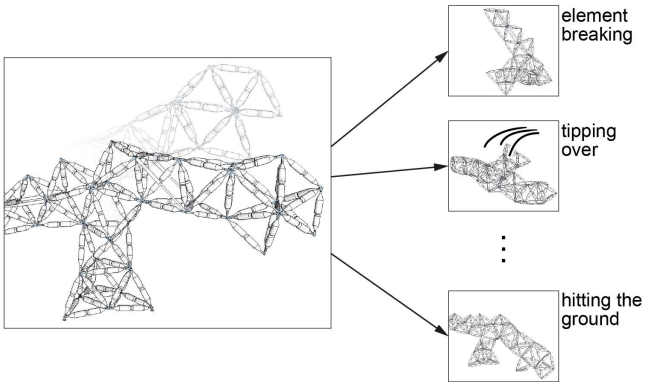

Figure 16: In the background, TrussFormer tests each actuator to see if its extension leads to invalid position, such as the structure tipping over, hitting the ground, or braking any structural elements.

This check is performed for each actuator individually. While a full factorial cross check would be necessary to detect damaging interaction effects, unfortunately, such an exhaustive search does not scale well with the increasing number of actuators and would deteriorate the software's interactivity. Therefore, TrussFormer still checks if the structure breaks in the later animation step and offers automatic correction.

3. Automatically adapting forces

After users create an animation sequence using the keyframe editor, shown in Figure 17a, TrussFormer continues to validate if the structure can withstand user-defined accelerations.

As we previously demonstrated in Figure 7, TrussFormer predicts that the T-Rex breaks if its neck is actuated too rapidly between a neck-down and a neck-up pose. This happened due to the large inertial forces. Since the structure is large, its mass is large as well. Forces that act on the elements of the structure increase proportionally with the acceleration of the movement ($\boldsymbol{F} = m\boldsymbol{a}$). While the mass is a constant in the structure, the acceleration is what TrussFormer can alter to prevent it from breaking. When the model breaks in the simulation, TrussFormer offers two options to reduce the occurring inertial forces, as shown in Figure 17.

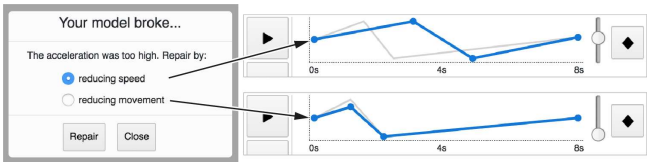

Figure 17: If the user-defined animation breaks the model, TrussFormer offers to automatically reduce the speed or the motion range.

TrussFormer offers to fix the animation slopes in two ways: (1) by reducing the speed of the motion, i.e., by stretching the time of the animation, or (2) by reducing the range of the movement. TrussFormer finds the valid actuation profiles by simulating the structure in the background and gradually

reducing the speed or the extension of the actuation, depending on the users' choice.

The predicted force values during the simulation are also used to inform users about properties of the actuators they need to buy to fabricate their structures, i.e., the minimum force that actuators must exert and the speed set in the animation. This force is defined as the maximum force that we measure during the simulation while the structure is performing the programmed animation.

Matching simulated and real forces

In order to accurately predict the forces within a structure, we measured the forces in our T-Rex example and tuned our simulation based on these measurements.

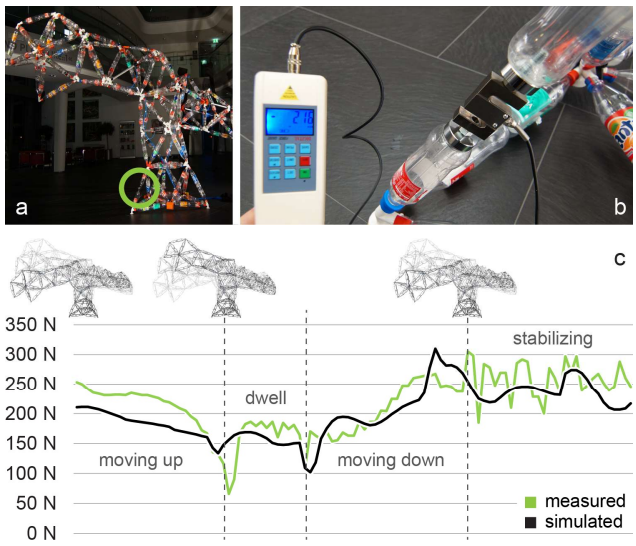

Figure 18: (a) We measured the forces on the bottom front edge of the T-Rex (b) using a digital force gauge. (c) The measured forces agree with the simulated forces.

As Figure 18a-b shows, we inserted the external force sensor (capacity: 5000 N, error: 0.5%) between two bottles at the bottom of the T-Rex structure. We chose this element as it bears the largest forces. We then actuated the T-Rex to move its entire head up to its highest position and down again to its lowest position. Figure 18c shows the measured and the simulated forces. The simulation is in agreement with the forces we measured.

Our simulated results hold for structures other than the T-Rex example, under the condition that the physical components remain the same.

TRUSSFORMER'S HINGE SYSTEM

A key element behind TrussFormer's kinetic structures is the 3D printable hinge system, that enables multiple edges to spherically pivot around a node point. While traditional ball joints allow for spherical motion, they are limited to connect only two edges. To address this shortcoming, TrussFormer uses the generic design of a *spherical joint mechanism* [5], shown in Figure 19a, that allows for multiple edges to pivot around the same center point, as they were connected via ball joint. Figure 19b shows TrussFormer's rendering of the spherical joint mechanism, adopted for 3D printing.

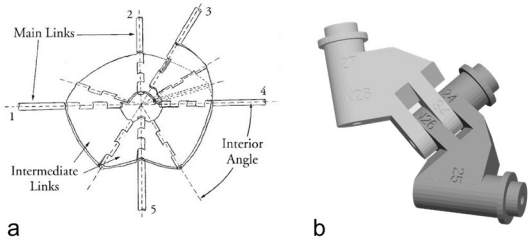

Figure 19: (a) Spherical joint mechanism [5] connecting 5 edges. (b) TrussFormer's 3D printable hinge design.

To achieve the motion that users designed, TrussFormer arranges the necessary spherical joints automatically in the structure. Traditionally, determining the required mobility of the joints is done by evaluating the Grübler–Kutzbach mobility criterion. However, this analytical approach is hard to fit for spatial (i.e., 3D) parallel mechanisms, and it's still subject to active research [19]. Therefore, instead of attempting an analytical solution to this problem, TrussFormer tests the motion of the user-defined structure by using its built-in physical simulation and arranges the hinges heuristically. In the following, we describe TrussFormer's four step hinge placement routine on the example of an octahedra with one actuator, shown in Figure 20a.

Step 1: Placing all possible hinges. As a first step, TrussFormer assigns the intermediate link connections of the spherical joint mechanism from Figure 19a, between all the edges forming a triangle in the structure, as illustrated with blue lines in Figure 20b. This provides 2DoF to all the edges, as they were connected via ball joints. This already gives a mechanically satisfying solution, however it can be further optimized. In variable geometry truss structures, most of the edges are confined in triangles and larger rigid substructures, therefore not all movements are possible. Placing unnecessary hinges only adds complexity for assembly and reduce mechanical stability.

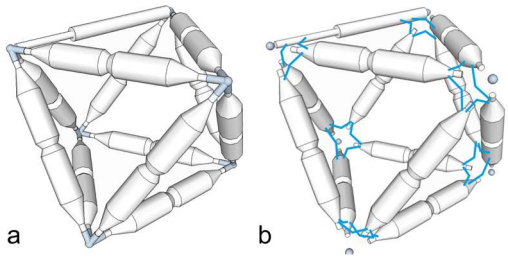

Figure 20: (a) Octahedron with an actuator, still with rigid hubs. (b) After the first step, intermediate links are assigned inside all triangles, creating spherical joints.

Step 2: Identifying rigid substructures. To identify rigid substructures in the structure, TrussFormer now runs the physical simulation and moves all the actuators simultaneously. It observes the angular movement between the edges and if the angle between two or more connected edges never changed, TrussFormer considers them as a *rigid*

substructure. Figure 21a shows the rigid substructures identified in the octahedron, visualized in distinct colors. The triangles containing the actuator are not considered as rigid, since it's internal angles are changing. Figure 21b shows the result of this step on the example of the T-Rex. Here, the rigid substructures consisting of single triangles are left uncolored, for clarity.

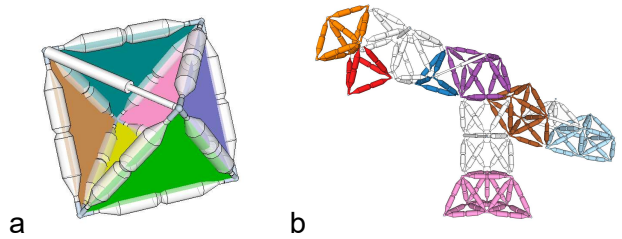

Figure 21: TrussFormer identified the rigid substructures (a) in the octahedron from Figure 20, and (b) in the T-Rex.

Step 3: Reducing the excess of hinges. Now that TrussFormer knows which parts of the structure are rigid, it can remove the unnecessary hinges between the edges which belong to the same rigid substructure and don't move in regards to each other. In Figure 22 we show this step on our octahedron example. Between the edges forming rigid substructures, the intermediate link connections (before blue lines) are reduced to rigid connections (black lines). Rigid substructures will still rotate with respect to each other. At this stage, the final hinge chain is already found for the octahedron example and the resulting 3D print is shown in Figure 22b.

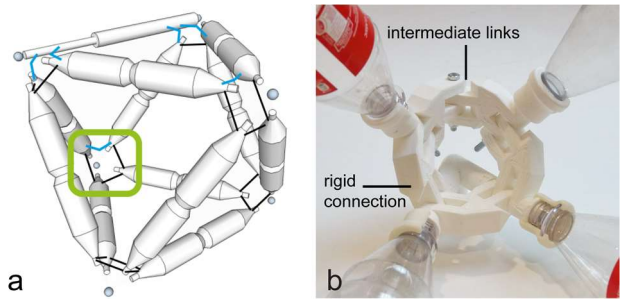

Figure 22: (a) The intermediate hinge connections are reduced to rigid connections where rigid substructures are identified (black lines). (b) The fabricated hinging hub of the marked node of the octahedron.

Step 4: Resolving impossible connections. At this point, TrussFormer has already assigned an optimized valid hinge configuration, however, not all the connections might be physically possible to assemble. TrussFormer's hinge design have the limitation that it only supports one-on-one hinge connections, as shown in Figure 19b. Three way connections are not possible, i.e., three parts cannot physically hinge around the same axis.

However, after Step 3, there might be hubs violating this condition, e.g., where three hinges are meeting at the same axis. We demonstrate this case in Figure 23 on the example of the double-octahedra structure with one actuator, similar to the one found in the body of the T-rex. In Figure 23b we highlight the hub where three-way hinge connections are present after performing Step 3. Fortunately, these connections are redundant in TrussFormer's kinetic structures, and they can be resolved by eliminating some of the hinges, while still maintaining the hub's structural integrity, i.e., all the edges remain interconnected via continuous hinge chain.

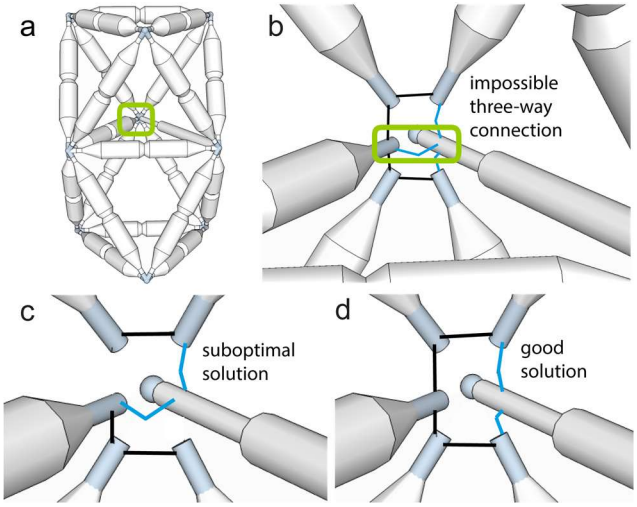

Figure 23: (a) Double-octahedral structure, where (b) violating three-way hinge connections appear. (c-d) TrussFormer finds the valid configurations by heuristic elimination and (d) chooses the structurally more stable closed chain.

TrussFormer resolves violating connections using a backtracking algorithm that removes connections heuristically. After each removed connection, it checks the validity of the resulting hinge configuration for two constraints: (1) all edges around the node are still interconnected directly or indirectly with each other, and (2) no more than two hinges are connected at each axis. If these constraints are satisfied, a valid hinge configuration was found. The algorithm continues until it finds all valid configurations. If available, TrussFormer will select the configuration with closed hinge chain (Figure 23d) over open chain (Figure 23c), for stability reasons. The fabricated hinge for this example was shown earlier in Figure 8b-c.

Generating the hinge geometries for 3D printing

After determining where the hinges should be placed in the structure, TrussFormer has all the necessary information to export the 3D printable geometries in the form of OpenSCAD [51] files. These files contain the information about the angle and connector lengths of the hinging pieces, as well as their imprinted IDs (as visible in Figure 23a). Users assemble the hinging hubs by matching the corresponding IDs. These IDs also contain the information about the placement of the actual hub within the structure, the IDs of the neighboring hubs, and the bottle type to be inserted.

RELATED WORK

TrussFormer builds on previous efforts in animatronics, robotics, software tools for creating mechanisms in HCI and graphics, and creating variable geometry truss mechanisms.

Software tools for Animatronics

Many HCI researchers have built software tools to empower users to animate robots [20, 25]. This is especially challenging when the users are novices and the intended results are expressive movements, such as imitating animal (organic) movements, i.e., animatronics [10, 28].

Animatronics interfaces follow several designs, from manual control [20] to puppeteering using skeletal tracking [27]. Marti et al. designed an early example of an animatronics software tool for a small (puppet sized) phone call handling agent, demonstrating two methods: manual control (user directly controls each single actuator using one GUI fader) and programming motion patterns using a sequencer [20]. Later work integrated robotics into keyframe editors, as for 3D animation [43] or video editing [44].

Previous tools suffice for animating small robots because actuating these robots (typically via small servo motors) does not involve moving large loads. With smaller robots, software tools do not have to simulate the adversarial effects of dynamics, e.g., inertia and resonance. However, when animating large animatronics, such as our T-Rex (Figure 1), these forces affect not only the stability of the structure but also the desired animation.

Software tools for designing mechanisms and dealing with forces

TrussFormer draws from work on systems that assist users with creating mechanisms that involve motion or forces.

Algorithmic tools can help users create moving mechanisms. For example, kinematic synthesis of mechanisms [34], or generation of personalized walking toys from a library of predefined template mechanisms [2]. These can be embedded in design support systems, for example, generating moving toys from motion input [42], or synthetized planar kinematic mechanisms from sketch-based motion input [9].

Several software tools directly help the manual design of linkage-based mechanisms, such as LinkageDesigner [48] and Mechanism Perfboard [15]. These tools sometimes include physical simulation of simple mechanisms (e.g., hinges); examples include Crayon Physics [50] and freeCAD [49].

Researchers in the domain of personal fabrication have started to investigate the effects of dynamic forces in the resulting models, such as balancing rotating objects [22], interactively designing model airplanes [39], and approximating the elastic behavior of 3D printed materials [7].

TrussFormer extends these approaches by using physical simulation interactively in its editor. This combination is necessary to provide an editor that embodies the domain knowledge needed to produce large scale animated truss structures.

Programming robotic manipulators

Programming robotic manipulators is a similar task to creating animation patterns for TrussFormer's mechanisms. The manufacturers of industrial robots usually provide their proprietary software packages for expert users, like ABB RobotStudio [45] or KUKA.Sim [46]. Recently, also visual block-based interfaces become popular for non-expert users, like KUKA|prc [47] and CoBlox [40]. These software tools provide advanced programming capabilities, however they still lack real time physical simulation to simulate forces during the motion.

Variable geometry truss mechanisms

TrussFormer's mechanisms are based on variable geometry trusses (VGT) [1, 24, 31]. An example of a VGT is the *Stewart Platform* [33], a common mechanism found in haptics/HCI. A Stewart platform uses actuators in every member to enable 6 DoF motion while maintaining the stability of a truss, crucial for scenarios that involve large inertial forces.

VGTs have been used extensively in robotics. *Tetrobot* [11, 12] is built by chaining the tetrahedron edges with linear actuators, which unite at a vertex in a spherical joint. The design and mechanics behind this type of spherical joints have been extensively analyzed [30, 32]. Tetrobot was designed to enable robots to reconfigure into different usages by reusing the same basic primitives. Researchers and engineers have explored variations of this VGT design in different contexts, for example: space applications [1], reconfigurable robotic manipulators [1, 11, 36], and shape morphing trusses [30].

Other researchers introduced design variations in this basic cell, allowing the resulting structure to afford new qualities. For instance, the *Spiral Zipper* [8] is an extendable edge, based on extending a cylinder that allows for extreme expansion ratios (e.g., 14:1). Similarly, *Pneumatic Reel Actuator* [13] is based on a mechanism that extrudes and retracts a plastic (tape-like) tubing, to act as an actuator. The mechanism is designed to be lightweight and low-cost ($4 USD) while being limited in its robustness.

TrussFormer takes inspiration from VGTs and builds on the conceptual design of Tetrobot. To this work, TrussFormer contributes a spherical joint design that is automatically generated based on the designed truss geometry.

IMPLEMENTATION

To help readers replicate our results, we now describe the implementation of the main components of the TrussFormer software system and the hardware we used to actuate our prototypes.

Software system

We extend the TrussFab editor [5], which provides the core functionality to create, save, load, and export static

structures. TrussFormer further allows users to add movement and animate these structures. Like TrussFab, TrussFormer is also implemented as a plugin for the 3D modelling software SketchUp [53]. The native programming language for SketchUp plugins is Ruby, which most of the features that we described throughout the paper, such as the hinge placement algorithm, are written in.

To simulate the movement and the force distribution in the 3D model, we use the physical simulation engine MSPhysics [35], a Ruby wrapper for the C++ physics library Newton Dynamics [52]. To achieve interactive performance, the only simulated components are the hubs, the edges are just animated on the scene. The hubs contain all the necessary information, such as weight, breaking force, and the stiffness determining how much hubs can move in relation to their neighbors.

User interface elements (e.g., the control or the animation pane) are displayed in a SketchUp-integrated Web Browser View. We implemented the UI in HTML and JavaScript to take advantage of UI frameworks such as React [53].

To generate the 3D printable hinge, we use the parametric 3D modeling tool OpenSCAD [51]. When users export their kinetic structure, TrussFormer determines the hinges and static hubs and calls the pre-defined OpenSCAD scripts with the relevant parameters (e.g., angle, connection type, or length of the connection). These scripts describe the resulting parametrized 3D model, which are rendered in OpenSCAD as *.stl* files.

Control system and actuators

Figure 24 shows the hardware we use to actuate our T-Rex example. We use pneumatic actuators with interfaced with proportional valves (Festo VPPE and MPYE series) that are controlled by an Arduino Nano. The pneumatic cylinders have diameters from 25 to 35 mm and produce forces between 390 N and 770 N. We use an *Airpress HL 360* compressor that can provide up to 8 bar of pressure.

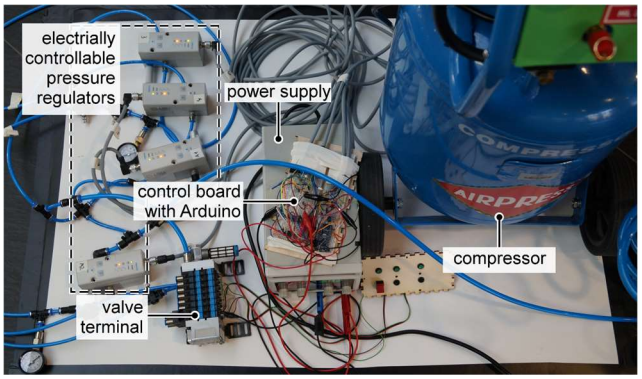

Figure 24: Hardware setup for controlling the T-Rex, with Arduino, electric pressure control valves, and compressor.

Our spider example in Figure 13 uses electric linear actuators similar to those found in garage doors. These actuators have a motion range of 45 cm and move rather slowly: 0.03 m/s compared to the pneumatic actuator speed of 20 m/s.

Building materials

For creating our models we used refillable soda bottles and 3D printed the hubs on an Ultimaker 3 3D printer using PLA material. To increase stability we set 3mm wall thickens for our hubs. While the 3D printing process is rather time consuming (5-8 hours/hub) the assembly of the hubs is quite fast (10-15min/hub). The overall structure is assembled in reasonable short time; our T-Rex took approximately 1-2 hours for 3 person.

We use plastic bottles as building material as they are ecologically friendly and commonly available all around the world. However, TrussFormer also supports any other type of building materials. Users only need to create and copy the 3D models of their material primitives into TrussFormer's material library folder.

To create more realistic looking animatronic creatures, users can also cover the structure with stretchable textile or other materials and attach smaller features (e.g. ears, fingers, etc) using 3D printing or other fabrication techniques.

CONCLUSIONS

We presented TrussFormer, an end-to-end system that enables novice users to design and build large animated structures. Such structures are usually a privilege of industry such as theme parks. TrussFormer encapsulates domain knowledge about the occurring dynamic forces so that even novice users can build such animated structures.

We showed how TrussFormer enables users to add motion to static structures in three ways, including simply pulling on the virtual model and letting the system find the placement of an actuator to enable this motion. Furthermore, we showed how TrussFormer finds valid motion and force ranges for actuators to realize user-defined animations. TrussFormer detects and automatically suggests corrections for animations that would break the simulated structure, thereby ensuring that the physical structure will function as desired. As a last step, TrussFormer generates all connectors and hinges that users print on their desktop 3D printer and exports the actuator specifications.

In the future, we plan to investigate how to conserve energy by incorporating springs and dampers into our system and taking advantage of resonant frequencies.

REFERENCES

1. V. Arun, Charles F. Reinholtz, and Layne Terry Watson. 1990. Enumeration and analysis of variable geometry truss manipulators. Department of Computer Science, Virginia Polytechnic Institute and State University.
2. Gaurav Bharaj, Stelian Coros, Bernhard Thomaszewski, James Tompkin, Bernd Bickel, and Hanspeter Pfister. 2015. Computational design of walking automata. In Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA '15). ACM, New York, NY, USA, 93-100. DOI: https://doi.org/10.1145/2786784.2786803

3. Patrick Baudisch and Stefanie Mueller. Personal Fabrication. *Foundations and Trends® in Human–Computer Interaction* Vol. 10: No. 3–4, 165-293. 2017.
4. Bernd Bickel, Moritz Bächer, Miguel A. Otaduy, Hyunho Richard Lee, Hanspeter Pfister, Markus Gross, and Wojciech Matusik. 2010. Design and fabrication of materials with desired deformation behavior. *ACM Trans. Graph.* 29, 4, Article 63 (July 2010), 10 pages. DOI: https://doi.org/10.1145/1778765.1778800
5. Paul Bosscher, Imme Ebert-Uphoff. 2003. A novel mechanism for implementing multiple collocated spherical joints. In *Robotics and Automation, Proceedings. ICRA'03. IEEE International Conference on* (Vol. 1, pp. 336-341). IEEE.
6. Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D³ data-driven documents. In *IEEE transactions on visualization and computer graphics* 17, no. 12: 2301-2309. DOI: 10.1109/TVCG.2011.185
7. Desai Chen, David I. W. Levin, Wojciech Matusik, and Danny M. Kaufman. 2017. Dynamics-aware numerical coarsening for fabrication design. *ACM Trans. Graph.* 36, 4, Article 84 (July 2017), 15 pages. DOI: https://doi.org/10.1145/3072959.3073669
8. Foster Collins, and Mark Yim. 2016. Design of a spherical robot arm with the spiral zipper prismatic joint. In *Proceedings of the IEEE International Conference on Robotics and Automation* (ICRA), 2137-2143.
9. 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
10. Paul H. Dietz and Catherine Dietz. 2007. The animatronics workshop. In ACM SIGGRAPH 2007 educators program (SIGGRAPH '07). ACM, New York, NY, USA, Article 36 . DOI: https://doi.org/10.1145/1282040.1282078
11. Gregory J. Hamlin and Arthur C. Sanderson. 2013. Tetrobot: A Modular Approach to Reconfigurable Parallel Robotics. In Volume 423 of The Springer International Series in Engineering and Computer Science, Springer Science & Business Media. ISBN: 1461554713, 9781461554714
12. Gregory J. Hamlin and Arthur C. Sanderson. 1997. Tetrobot: A modular approach to parallel robotics. *IEEE Robotics & Automation Magazine 4, no. 1*:42-50. DOI: 10.1109/100.580984
13. Zachary M. Hammond, Nathan S. Usevitch, Elliot W. Hawkes, and Sean Follmer. 2017. Pneumatic Reel Actuator: Design, modeling, and implementation. In *Proceedings of the IEEE International Conference on Robotics and Automation* (ICRA), 626-633.
14. 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. DOI: http://doi.org/10.1145/2556288.2557338
15. Yunwoo Jeong, Han-Jong Kim, and Tek-Jin Nam. 2018. Mechanism Perfboard: An Augmented Reality Environment for Linkage Mechanism Design and Fabrication. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (CHI '18). ACM, New York, NY, USA, DOI: https://doi.org/10.1145/3173574.3173985
16. Behrokh Khoshnevis. 2004. Automated Construction by Contour Crafting—Related Robotics and Information Technologies. *Automation in Construction* 13: 5–19. http://doi.org/10.1016/j.autcon.2003.08.012
17. Robert Kovacs, Anna Seufert, Ludwig Wall, Hsiang-Ting Chen, Florian Meinel, Willi Müller, Sijing You, Maximilian Brehm, Jonathan Striebel, Yannis Kommana, Alexander Popiak, Thomas Bläsius, and Patrick Baudisch. 2017. TrussFab: Fabricating Sturdy Large-Scale Structures on Desktop 3D Printers. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (CHI '17). ACM, New York, NY, USA, 2606-2616. DOI: https://doi.org/10.1145/3025453.3026016
18. 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
19. Wenjuan Lu, Lijie Zhang, Yitong Zhang, Yalei Ma, Xiaoxu Cui. 2014. Modified Formula of Mobility for Mechanisms. In *Proceedings of the International Conference on Intelligent Robotics and Applications* (ICIRA'14). Springer, Cham, Germany, 2014, 535-545. DOI : https://doi.org/10.1007/978-3-319-13963-0_54
20. Stefan Marti and Chris Schmandt. 2005. Physical embodiments for mobile communication agents. In *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology* (UIST'05). ACM, New York, NY, USA, 231-240. DOI: http://dx.doi.org/10.1145/1095034.1095073
21. 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

22. Romain Prévost, Emily Whiting, Sylvain Lefebvre, and Olga Sorkine-Hornung. 2013. Make it stand: balancing shapes for 3D fabrication. *ACM Trans. Graph.* 32, 4, Article 81 (July 2013), 10 pages. DOI: https://doi.org/10.1145/2461912.2461957
23. Jie Qi and Leah Buechley. 2012. Animating paper using shape memory alloys. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems* (CHI '12). http://doi.org/10.1145/2207676.2207783
24. Marvin D. Rhodes, and Martin M. Mikulas. *Deployable controllable geometry truss beam*. 1985 National Aeronautics and Space Administration, Scientific and Technical Information Branch (Technical report).
25. Tiago Ribeiro and Ana Paiva. 2012. The illusion of robotic life: principles and practices of animation for robots. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction* (HRI '12). ACM, New York, NY, USA, 383-390. DOI: https://doi.org/10.1145/2157689.2157814
26. Thijs Roumen, Willi Mueller and Patrick Baudisch. 2018. Grafter: Remixing 3D Printed Machines. In *Proceedings of the 36th Annual ACM Conference on Human Factors in Computing Systems* (CHI '18). ACM, New York, USA DOI: https://doi.org/10.1145/3173574.3173637
27. Mose Sakashita, Tatsuya Minagawa, Amy Koike, Ippei Suzuki, Keisuke Kawahara, and Yoichi Ochiai. 2017. You as a Puppet: Evaluation of Telepresence User Interface for Puppetry. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (UIST '17). ACM, New York, NY, USA, 217-228. DOI: https://doi.org/10.1145/3126594.3126608
28. Shooter, P.E., Steven B. Animatronics. 2015, in International Journal of Advanced Research in Computer Science and Software Engineering, pg. 1260, Volume 5, Issue 3, March 2015, ISSN: 2277.
29. Mélina Skouras, Bernhard Thomaszewski, Stelian Coros, Bernd Bickel, and Markus Gross. 2013. Computational design of actuated deformable characters. *ACM Trans. Graph.* 32, 4, Article 82 (July 2013), 10 pages. DOI: https://doi.org/10.1145/2461912.2461979
30. A Y N Sofla, D M Elzey, and H N G Wadley. 2009. Shape morphing hinged truss structures. *Smart Materials and Structures* 18: 65012. http://doi.org/10.1088/0964-1726/18/6/065012
31. Alexander Spinos, Devin Carroll, Terry Kientz, and Mark Yim. 2017 Variable topology truss: Design and analysis. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (IROS), Vancouver, BC, 2017, 2717-2722. DOI: 10.1109/IROS.2017.8206098
32. Alexander Spinos and Mark Yim. 2017. Towards a variable topology truss for shoring. 2017. In *Proceedings of the 14th International Conference on Ubiquitous Robots and Ambient Intelligence* (URAI), 244–249. http://doi.org/10.1109/URAI.2017.7992723
33. Doug Stewart. 1965. A platform with six degrees of freedom. In *Proceedings of the institution of mechanical engineers* 180.1, 371-386.
34. Devika Subramanian. 1995. Kinematic synthesis with configuration spaces. In *Research in Engineering Design* 7, no. 3, 193-213.
35. Anton Synytsia. *MSPhysics* physics simulation. Retrieved on April 3, 2018 from https://extensions.sketchup.com/en/content/msphysics
36. Emil Teutan, S.D. Stan., D. Verdes, R. Balan. 2009. Virtual Reality Simulation of Tetrobot Parallel Robot for Medical Applications. In *Proceedings of International Conference on Advancements of Medicine and Health Care through Technology*, Vol 26. pp. 177-180.
37. Bernhard Thomaszewski, Stelian Coros, Damien Gauge, Vittorio Megaro, Eitan Grinspun, and Markus Gross. 2014. Computational design of linkage-based characters. *ACM Trans. Graph.* 33, 4, Article 64 (July 2014), 9 pages. DOI: https://doi.org/10.1145/2601097.2601143
38. Nobuyuki Umetani, Takeo Igarashi, and Niloy J. Mitra. 2012. Guided exploration of physically valid shapes for furniture design. *ACM Trans. Graph.* 31, 4, Article 86 (July 2012), 11 pages. DOI: https://doi.org/10.1145/2185520.2185582
39. Nobuyuki Umetani, Yuki Koyama, Ryan Schmidt, and Takeo Igarashi. 2014. Pteromys: interactive design and optimization of free-formed free-flight model airplanes. *ACM Trans. Graph.* 33, 4, Article 65 (July 2014), 10 pages. DOI: https://doi.org/10.1145/2601097.2601129
40. David Weintrop, Afsoon Afzal, Jean Salac, Patrick Francis, Boyang Li, David C. Shepherd, and Diana Franklin. 2018. Evaluating CoBlox: A Comparative Study of Robotics Programming Environments for Adult Novices. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (CHI '18). ACM, New York, NY, USA. DOI: https://doi.org/10.1145/3173574.3173940
41. Nora S. Willett, Wilmot Li, Jovan Popović, and Adam Finkelstein. 2017. Triggering Artwork Swaps for Live Animation. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'17)*. ACM, New York, NY, USA, 85-95. DOI: https://doi.org/10.1145/3126594.3126596

42. Lifeng Zhu, Weiwei Xu, John Snyder, Yang Liu, Guoping Wang, and Baining Guo. 2012. Motion-guided mechanical toy modeling. *ACM Trans. Graph.* 31, 6, Article 127 (November 2012), 10 pages. DOI: http://dx.doi.org/10.1145/2366145.2366146
43. Blender's Keyframe editor, https://docs.blender.org/manual/en/dev/animation/keyframes/editing.html , last accessed at 31/03/2018.
44. Adobe Premier's keyframe editor, https://helpx.adobe.com/premiere-pro/using/adding-navigating-setting-keyframes.html, last accessed at 31/03/2018.
45. ABB RobotStudio. Retrieved on July 13, 2018 from https://new.abb.com/products/robotics/robotstudio
46. KUKA.sim. Retrieved on July 13, 2018 from https://www.kuka.com/en-de/products/robot-systems/software/planning-project-engineering-service-safety/kuka_sim
47. KUKA|prc. Retrieved on July 13, 2018 from http://www.robotsinarchitecture.org/kuka-prc
48. Linkage designer. Retrieved on April 3, 2018 from http://www.linkagedesigner.com/
49. freeCAD. Retrieved on April 3, 2018 from http://www.ar-cad.com/freecad/
50. Crayon Physics. Retrieved on April 3, 2018 from http://crayonphysics.com/
51. OpenSCAD. Retrieved on April 3, 2018 from http://openscad.org
52. Newton Dynamics. Retrieved on April 3, 2018 from http://newtondynamics.com/forum/newton.php
53. React. Retrieved on April 3, 2018 from https://reactjs.org/SketchUp. Retrieved on April 3, 2018 from https://www.sketchup.com/