PaperCraft3D: Paper-Based 3D Modeling and Scene Fabrication

Patrick Paczkowski, Julie Dorsey, Holly Rushmeier, and Min H. Kim *Member, IEEE*

Abstract—A 3D modeling system with all-inclusive functionality is too demanding for a casual 3D modeler to learn. There has been a shift towards more approachable systems, with easy-to-learn, intuitive interfaces. However, most modeling systems still employ mouse and keyboard interfaces, despite the ubiquity of tablet devices and the benefits of multi-touch interfaces. We introduce an alternative 3D modeling and fabrication paradigm using developable surfaces, inspired by traditional papercrafting, and we implement it as a complete system designed for a multi-touch tablet, allowing a user to fabricate 3D scenes. We demonstrate the modeling and fabrication process of assembling complex 3D scenes from a collection of simpler models, in turn shaped through operations applied to virtual paper. Our fabrication method facilitates the assembly of the scene with real paper by automatically converting scenes into a series of cutouts with appropriately added fiducial markers and supporting structures. Our system assists users in creating occluded supporting structures to help maintain the spatial and rigid properties of a scene without compromising its aesthetic qualities. We demonstrate several 3D scenes modeled and fabricated in our system, and evaluate the faithfulness of our fabrications relative to their virtual counterparts and 3D-printed fabrications.

Index Terms—multi-touch interface, 3D modeling, fabrication, papercraft.

1 Introduction

Creating 3D objects, through acquisition, modeling and physical fabrication, historically has been limited to professional users. For instance, 3D laser scanners or professional modeling tools were traditionally used for creating virtual models, while high-end, commercial 3D printers were used to fabricate these models as physical objects. Recently, the affordability and accessibility of these professional tools for modeling and printing have dramatically increased. 3D modeling and fabrication are now popular among casual users with little modeling experience. However, modeling and fabrication techniques still limit user creativity. We present a new modeling and fabrication system aimed at casual users[1].

3D Modeling. In 3D modeling, all-inclusive functionality is generally overwhelming for a casual user. Packages such as Blender or Maya provide many modeling techniques, e.g., polygonal modeling, modeling with NURBS or subdivision surfaces. However, learning how to use the majority of existing 3D modeling packages is challenging. An additional difficulty is the communication of 3D modeling coordinates through the standard user interface. Typical input devices, e.g., mice and pens, and screens provide 2D input and output. 2D devices are inadequate for specifying 6 degree-of-freedom (DoF) position and orientation. Conversely, devices with a full 6-DoF (e.g., Geomagic Touch) are inaccurate and difficult for novice users to operate.

Motion sensing interfaces with higher DoF have appeared, such as LeapMotion and Microsoft's Kinect, but these suffer from lack of accuracy, and lack of haptic feedback. By contrast, input devices with multi-touch interfaces have evolved in recent years, with products like the Apple iPad and the Samsung Galaxy Tab. Though each input touch is still two-dimensional, the combination of two or more touches has resulted in gestures with more information than the input of a traditional, single-point device. Due to their ubiquity and intuitiveness, these devices are particularly powerful for casual users.

In 3D modeling, problems such as 3D transformations on a multi-touch device have been studied, and there are domain-specific modeling systems for a multi-touch interface. However, systems specifically focused on casual 3D modeling by users with little modeling experience (e.g., sketch-based interfaces [2]) are still fairly limited either in scope, learnability or modeling representation. This paper presents a novel 3D modeling paradigm tailored for such use to fully leverage the benefits of a multi-touch interface.

3D Fabrication. Similar challenges exist in 3D. For casual users, producing a personally designed physical model is both entertaining and fulfilling. For professionals, rapid fabrication is a natural prototyping stage in many fields. Being able to hold and look at a physical model can provide a level of immersion that a digital model does not.

Despite the popularity of 3D printing, it is still challenging for casual modelers. Part of the issue is that there are constraints on what can be reliably 3D printed. Many models need adjustments to successfully print. There are strict limitations in size and resolution that may prevent the user from producing a model at the right scale or level of

- *P. Paczkowski, J. Dorsey, H. Rushmeier are with the Department of Computer Science, Yale University, New Haven, CT, 06511.*
- *M.H. Kim, the corresponding author, is with the School of Computing, KAIST, Daejeon, South Korea, 34141. E-mail: minhkim@kaist.ac.kr*

Manuscript received April 30, 2017; revised July 26, 2017.

1. The modeling part was published in conference proceedings of ACM UIST [1], this paper is a revised and extended version of the conference paper to additionally introduce the fabrication of 3D model designed with developable surfaces.

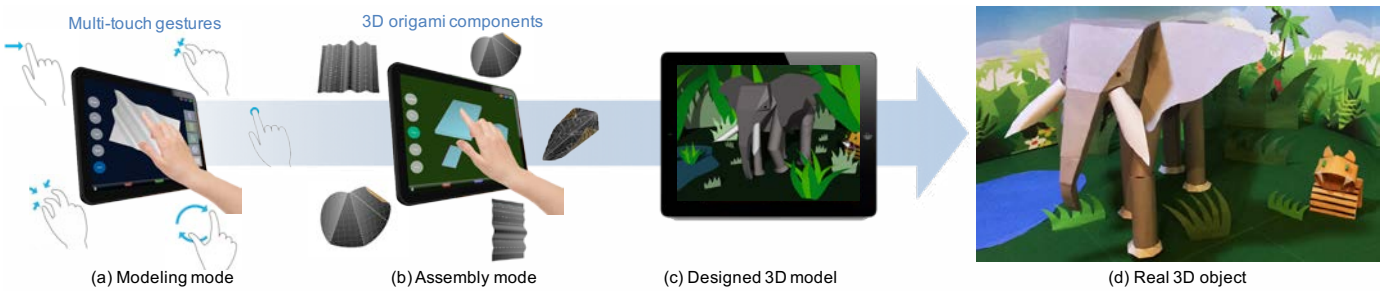


Fig. 1. An overview of our novel 3D modeling technique, designed for a multi-touch interface to create 3D models on a tablet device. (a) A user folds 2D sheets of paper in the modeling mode, using gesture-based modeling tools such as folding, bending, extending, and cutting. (b) A set of modeled 3D components is assembled together through pinning and taping, resulting in (c) a complex 3D model. (d) The user can create a real 3D scene object from the designed 3D model through our computer-aided paper crafting, resulting in (d) a real 3D scene.

detail. Lastly, 3D printing or other fabrication tools, such as Pepakura [3], are suited for printing individual models stripped of a global coordinate system, and of the spatial relationship between individual models in a scene.

To maintain the general spatial relations between scene components in fabrication, we add a small number of supporting structures. Building a robust structural model would require a full physical simulation of the model. This is inconsistent with the casual nature of our system. Instead we use a simple, practical approach for placing supports that are well-occluded in key user-specified views. Once supports are added, we add assembly guides and associated fiducial markers for accurate, guided assembly. The full mesh is then unfolded, segmented and printed onto sheets of paper. The guided user assembly of the scene is a simple, sequential process.

Overview. This work presents a novel 3D modeling and fabrication system tailored for creating inexpensive scenes composed of developable surfaces. First, inspired by traditional papercraft, we devise a 3D modeling technique in which deformations mimic physically-based operations on sheets of paper. We use theses simple, powerful techniques as a foundation, and extended them into a broader, practical technique for modeling developable surfaces. We design and integrate this technique on a multi-touch device [1]. We evolve the system design in collaboration with both casual and experienced users. Second, using the simple, yet powerful principles of traditional papercrafting, we extend this foundation of 3D modeling into a broader, practical technique for fabricating developable surfaces. We evolve our 3D modeling system [1], extending it to producing fully-textured fabricated scenes. Our method prepares the virtual scene for printing on sheets of letter-sized paper using a conventional printer. Through our fabrication process, we are able to simultaneously maintain scene structure and aesthetics. Our contributions to 3D modeling and fabrication are:

- A new modeling paradigm inspired by papercrafting, and extended by operations related to other physical actions;
- An implementation of a modeling system, based on this new paradigm, designed from the ground up to use multi-touch gestures;
- A new fabrication system for intuitive physical production of virtual 3D scenes modeled with developable surfaces,
- An algorithm for adding supports to a 3D scene, to maintain its structural stability and its aesthetics,
- A working implementation of our fabrication methodology, fully integrated into a system for digital 3D modeling and fabrication.

2 Related Work

This section surveys previous work on interfaces, 3D modeling and papercraft simulators, and fabrication.

2.1 3D Modeling Alternatives

2.1.1 Multi-Touch Interfaces

It is difficult to specify 3D coordinates using a 2D graphical interface (e.g., trackball, stylus, mouse). Indirect 3D graphical interfaces have gained popularity (e.g., Microsoft Kinect, LeapMotion, Geomagic Touch, etc.). Though these devices enable 6-DoF tracking, the lack of haptic feedback still limits their intuitiveness [4], [5]. From their inception, it was clear that multi-touch interfaces, pioneered by Lee et al. [6], offer significant advantages with their wide range of natural inputs, such as pinch gestures.

Multi-touch devices (Apple iPad, Microsoft Surface, etc.) are now widespread. This has resulted in pursuits such as developing intuitive 3D transformations on a multi-touch device [7]–[11]. Multi-touch devices have been explored for casual 3D modeling [12], [13]. In this work, we also use a multi-touch graphical interface for a 3D modeling system. The *intuitiveness* of such devices does not eliminate the ambiguity of mapping 2D screen inputs into 3D, but we have found that users, particularly casual modelers, find these devices more natural.

2.1.2 3D Modeling Systems

Desktop Modeling Systems. Commercial desktop systems such as AutoCAD, Maya or SolidWorks provide extensive modeling capabilities suited for industry. These tools are geared towards trained professionals. More accessible, alternative modeling systems have gained in popularity, such as SketchUp and ZBrush.

Gesture-Based Modeling Systems. Most modeling techniques have one-point-based interaction – e.g., dragging a point or set of points. Various papers describe and evaluate methods for transforming objects with multi-touch, e.g., [7]–[11]. However, this is just one small aspect of a modeling

system. Some researchers have begun to explore how a multi-touch interface could be used to model primitive and abstract shapes [12], [13] or to interact with 3D objects [9]. De Araùjo et al. [14] introduced a semi-immersive environment for preliminary conceptual modeling without haptic feedback. While their experience was satisfactory, the modeling system itself proved insufficient for precise control of 3D geometry. Walther-Franks et al. [15] performed a preliminary study by augmenting Blender, the 3D modeling tool, with a multi-touch interface. The multi-touch operations are mainly limited to object *animation* functions, rather than geometric modeling tools.

Autodesk launched a set of mobile products for modeling and design, including 123D Design and 123D Sculpt – notably simplified tools compared to their desktop counterparts. Li et al. [16] recently proposed a sketch-based interactive modeling system based on an RGB-D sensor input. Inexperienced users can create curved 3D surfaces using sweeping strokes. However, these are still basically *one-point-based* systems – beyond view manipulation, little is done to take advantage of a multi-touch interface. Several domain-specific modeling systems have been created for multi-touch interfaces, such as Eden (for constructing organic sets) [17], Sun et al.'s system for architectural design [18], and Wilson et al.'s physics simulator [19]. However, these systems are primarily targeted at domain-specific professional users.

2.1.3 Papercraft Simulators

Papercrafting is a set of art forms that use paper to create physical 3D objects. In particular, origami, the traditional Japanese art of paper folding, has been extensively studied from a computer science and applied math perspective. Origami simulators have been implemented in various forms [20]–[23]. Rather than producing a faithful simulation of origami, we use the *principles* of origami to create a novel 3D modeling paradigm and system to design freeform 3D objects and scenes. Similar to the way sketching inspired the Teddy system by Igarashi et al. [24], we take inspiration from simple interactions with physical paper.

2.2 3D Fabrication Alternatives

3D Printing. 3D printing allows virtually anyone to independently design and manufacture products [25]–[27]. Many enhancements exit to prepare models for additive manufacture, such as computing efficient and temporary support structures [28] and analyzing stresses to determine the optimal print direction [29]. However, there are still several disadvantages. It can be difficult to learn how to 3D print a model for someone with little experience. There are printer-specific constraints on the size of models produced. The size constraint requires that objects in a 3D scene are printed separately, and later manually arranged by the user. Models with open meshes (e.g., stage sets) that can be created in a surface-based modeling system are not easily handled by a 3D printer. Structural aspects of a scene, such as the placement and distribution of mass of its various components need to be accounted for in the modeling, rather than printing, phase.

Other Digital Fabrication Methods. Aside from 3D printing, the most common form of output is printing an unfolded mesh on sheets of paper or cardboard. This is followed by user assembly of the printed sheets. Prior work focusing on the fabrication of 3D models, such as the seminal work of [30] and associated Pepakura software, or the work of Shatz et al. [31] are restricted to single, closed mesh models. Other software products, such as TreeMaker, are specific to creating origami models, and not generalizable to 3D models and scenes [32]. Several works [33]–[35] use planar slices to approximate closed meshes; Massarwi et al. [36] uses generalized cylinders. While simplifying the fabrication process, these are only approximate representations of the actual models. A larger shortcoming of all these methods is that separately producing multiple models from a larger 3D scene will not automatically preserve structural stability between the objects or between each object and the scene base. These relationships are often integral to the look of the scene. Lastly, there have been fabrication methods for very specific types of objects/materials, such as pop-up cards [37]–[39], knitted models [40], beadwork [41], and plush toys [42]. In addition, as opposed to providing an intuitive way to produce the shape in the first place, there have been efforts [43]–[45] to produce any given shape represented as a manifold mesh by paper folding [20]–[22]. Though conceptually interesting and creative, these works provide limited fabrication functionality.

Interactive Fabrication. An alternative paradigm is interactive fabrication, where users can fabricate objects in real-time (without first creating a 3D model) using a tangible user interface (TUI) [46]–[50]. Generally these fabricate a limited range of objects and require the use of specialized hardware. By contrast, our method only needs is an iPad and a paper printer.

Systems for Digital 3D Modeling and Fabrication. 3D printing systems normally work independently of modeling systems. However, there are benefits to tailoring a fabrication tool to a specific system. Autodesk's line of 3D modeling apps, for example, have built-in functionality that prepare models for successful 3D printing. A system in between interactive and digital fabrication was devised by Zoran and Paradiso [51]. FreeD employs a handheld milling device with a magnetic motion tracking system for fabrication. However, it relies on an interface with an existing 3D modeling system (Rhino). Lin et al. [52] designed a system that encompasses 3D modeling and fabrication, but models are required to be from scanned and extracted objects in the real world, as opposed to designed from scratch. Swaminathan et al. [53] designed a system for modeling and fabrication, but the system is restricted to outputting physical visualizations. Mueller et al. [54], [55] presented systems for fast fabrication of individual objects to facilitate the design and print cycle. McCrae et al. [56] introduced an integrated modeling and fabrication system, and evaluated the structural stability of 3D models, similarly to our proposed system. Their system approximates 3D models with planar sections, a coarse representation resulting in a more abstract appearance, whereas our system remains faithful to the original models. Stava et al. [57] process models before printing to reduce stress. The addition of supporting struts

is one method of stress-reduction that they use, and the ambient occlusion of the strut is used to assess its visual impact. Stava et al. consider only individual objects.

We introduce an integrated 3D framework, inspired by papercrafting and implemented on a multi-touch interface. We seamlessly integrate fabrication functionality, obtaining a full system for 3D modeling and fabrication.

3 3D Modeling with Developable Surfaces

To create an effective multi-touch modeling system, we observed that paper folding – a simple, physical process virtually anyone can identify with – has strong synergy with a multi-touch interface on a flat-surface device. We build on folding with additional physically-inspired operations including cutting, bending, pinning and taping. We then further expand the system to allow the assembly of sets of individually defined component objects into complex scenes. This allows us to focus on developable surfaces.

Traditional origami begins with a single, square sheet of paper. Only three basic folds are permitted: mountain folds (forming a ridge), valley folds (forming a trough), and creases. Origami, and particularly the work of [23] inspired several initial modeling functions in our system. We extend far beyond the constraints of traditional origami or origami simulators.

3.1 Intuitive, Gesture-Based Interactions

Our 3D modeling system is created for a tablet device with a multi-touch display (a third-gen. iPad). The majority of user interaction occurs through *intuitive* single- and multi-finger gestures. An intuitive interaction is one that has an easily understandable gesture mapping, is predictable and straightforward to replicate, and results in immediate visual feedback to the user. We focus on creating interactions that have a tangible, direct effect on a model. We use the notation t_i to represent the normalized screen coordinate of the $(i+1)^{th}$ gesture input, and define t_{i_start} and t_{i_end} as the initial and new/final screen coordinates of the user's finger. The corresponding points projected along the camera view direction onto the plane of current face f are p_{i_start} and p_{i_end}. The primary gestures we used are illustrated in Figure 2.

3.2 Scene Objects

Each individual object, or *3D component* in our system consists of interconnected *faces*, *edges* and *vertices*. Each face is a convex, planar area; its closed outline is composed of edges and vertices. Every edge has exactly two vertices, and is classified as a *boundary edge* (belongs to a single face, or two folded-over faces), a *fold* (between two non-coplanar faces), or a *crease* (between two coplanar faces). Two or more edges can have the same vertex; a vertex belonging to two boundary edges is a *corner*.

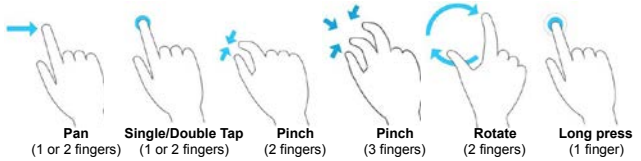


Fig. 2. Examples of multi-touch gestures used in our modeling system.

3.3 Modeling System Modes

The system has two design modes: *modeling* and *assembly*. Users can transition between the modes at any time. The *modeling mode* allows users to model sheets of source paper (one at a time) into 3D components through creasing, folding, bending, extending and cutting operations. In the assembly mode, users can insert models, transform them, and then group them together through pinning and taping operations [1].

Certain functions require the same gesture. To remove ambiguity, each mode has five radio buttons to the left, grouping sets of operations into submodes. In the modeling mode, the five submodes are *extend*, *fold*, *bend*, *crease/cut*, and *view*. In the assembly mode, the submodes are *insert*, *transform*, *pin/tape*, *color*, and *view*. A user may rest a finger of their non-dominant hand on a submode button, activating the submode only for as long as they are holding the button. This streamlines operations by allowing quick viewing of a model between modeling operations. In the modeling mode, thumbnails of existing models can be used to select a model for editing. In the insert submode of assembly mode, these thumbnails are used to drag model copies into the scene.

4 3D Modeling and Assembly Tools

4.1 Gesture-Based 3D Modeling Tools

Creating a 3D component begins with an initially flat sheet of paper that can be resized using a pinch gesture. Its shape may also be defined with a freeform outline tool, or by choosing a predefined (e.g., triangle, circle).

Planar Operations. Three fundamental operations available to the user are *creasing* (dividing one or more faces into two), *extending* (creating new, connected, coplanar faces), and *cutting* (dividing part of a model along a crease). The user defines a crease through two finger inputs p_0 and p_1, panning to adjust its position and orientation. (Figure 3a.) Upon releasing, the face is divided through a SPLIT() operation into two new faces connected along the crease. A outward pinch with fingers over a selected crease and one of its connected faces cuts that side of the model, either removing it or making it a separate component. (Figure 3b.) To extend the current sheet, as seen in Figure 3c, the user can drag two fingers along the outward normal of any boundary edge, creating a new face. Two of its vertices are of the crossed boundary edge; the remaining ones are defined by the two projected gestural inputs p_0 and p_1.

Extending. Users can bridge two edges using an *edge-to-edge extend*, by placing a finger over one edge and swiping the other edge towards the first (Figure 3d). A three-finger pinch on a face will extend the paper along all the boundary edges of a face simultaneously, along the normal to the original face. The distance the fingers are spread apart controls the width of the new faces, and the angle of the new faces can be subsequently adjusted (Figures 3e and 3f). A four-finger pinch performs an extend operation on all the faces of the component simultaneously, allowing users to quickly add thickness to their model. Any curved face is extended along its original normal prior to the added curvature.

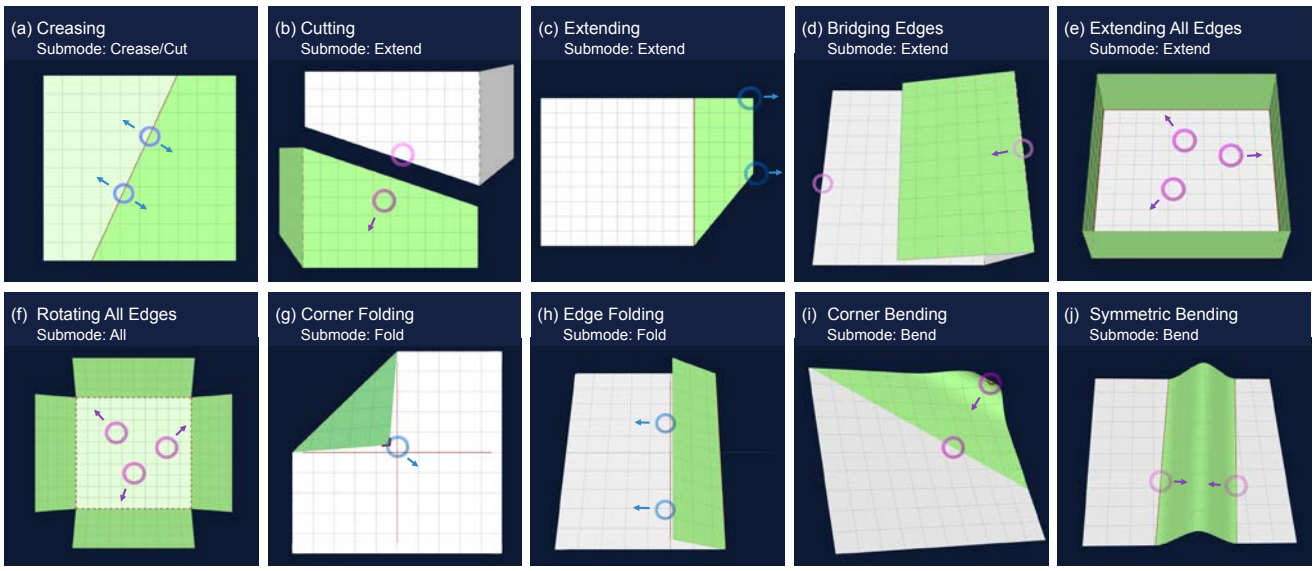


Fig. 3. Representative 3D modeling tools: (a) creasing a sheet of paper, (b) cutting along an edge, (c) extending along an edge of a face, (d) bridging edges of two faces, (e) extending all edges of a face, (f) rotating all faces generated in (e), (g) corner folding, (h) edge folding, (i) corner bending, and (j) symmetric bending. Circles and arrows indicate the position and motion of user touches.

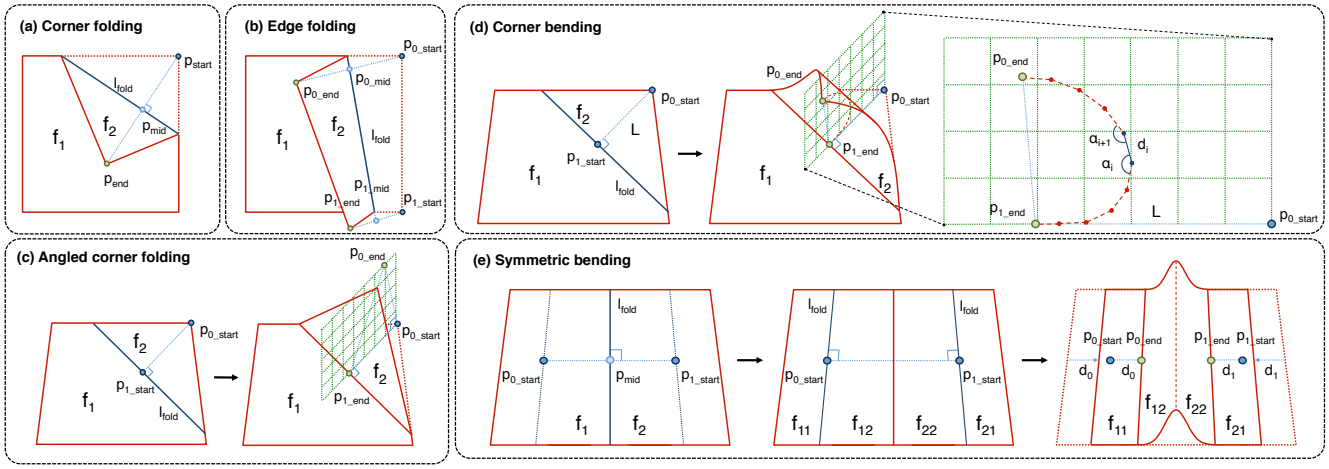


Fig. 4. Representative functions of 3D modeling tools. (a) and (b) show how the folding line l_{fold} is calculated for corner and edge folds. (c) illustrates how the two user inputs are projected onto a plane perpendicular to the current face, allowing the user to specify both the fold angle and orientation. (d) shows how a curve is fit between the two projected input points, resulting in the curved surface. (e) shows the effect a symmetric bend has on a face of the model, translating each half inwards as the bend is increased.

Folding. A one-finger drag over one of the adjoining faces of a selected crease adjusts the angle of the crease (through a call to ROTATE()), turning it into a folded edge, with the angle defined by t_0 projected onto the normal of the edge. All faces connected to the rotating face (on the same side of the crease/fold) are rotated together with it. The angle of any existing crease or fold of the 3D component may be adjusted in this way. In a *corner fold* operation, a user's finger is dragged across the screen over a corner of a face (selecting it), and the user starts to fold over the face (Figure 3g). The folding line l_{fold} is defined as the line perpendicular to $l(p_{start}, p_{end})$ and passing through the midpoint of p_{start} and p_{end}. The new, folded-over face f_{folded} is rotated 180° about l_{fold}. Similarly, *edge folding* lets the user fold over the paper by grabbing any boundary edge of the component (Figure 3h), with its final position unambiguously defined through a two-finger pan gesture. The midpoints of the two start and end touches, p_{0_mid} and p_{1_mid}, define the folding line l_{fold}, and the new face f_{folded} is rotated 180° about l_{fold} (Figures 4a and 4b).

Angled Folding and Bending. A pinch gesture on a multi-touch device lets us define an *angled corner folding* tool, allowing a user to simultaneously fold over a corner of the sheet of paper, while also controlling the folding angle between the fixed and folded-over parts. The folding line l_{fold} is defined as the line perpendicular to LINE(p_{0_start}, p_{1_end}) and passing through p_{1_end} (Figure 4c). The angle of rotation of face f_{folded} is found using p_{0_end}, the 3D point found by projecting p_{0_end} onto the plane passing through p_{0_start} with normal parallel to l_{fold}. *Angled edge folding* is analogous, except it starts on a boundary edge instead of a corner. In *corner bending*, we extend angled folding to allow users to curve parts of their model (Figure 3i). The angled face f_{folded} is curved into n bend strips, through a recursive sequence of n SPLIT() and ROTATE() operations. Dividing lines l_1 to l_n are parallel to the initial dividing line l_{fold}. Following [23], we determine the line spacing and the angle of each rotation by minimizing the energy function

$$E = \alpha \Sigma_i (d_i - L)^2 + \beta \Sigma_i (a_i - a_{i+1})^2, \qquad (1)$$

where L is a constant equal to $1/k$ multiplied by the distance between the line l and the selected corner at p_{0_start}, and d_i and a_i are the widths of and angles between the bend strips. Constants $\alpha = 0.6$ and $\beta = 0.4$ were determined experimentally (Figure 4d). If the user's fingers are close together, the paper is curved into a cylindrical shape (instead of using the energy minimization function), connecting points p_{0_end} and p_{1_end}. *Edge bending* instead bends over an edge of the model, but is otherwise identical.

Symmetric Folding and Bending. A *symmetric bend* is activated when a user pinches an inner region of the paper. As the user's two fingers are drawn together, the paper bends upwards, forming a peak at the midpoint (Figure 3j). In the implementation, we define w_{orig} as half the original distance between the user's initial, projected contact points p_{0_start} and p_{1_start}. w_{bend} and h_{bend} are defined as the width and height of half the bent portion of the paper, computed based on the new/final contact points p_{0_end} and p_{1_end}. The original face f_{orig} is first divided along the line passing through the midpoint of p_{0_end} and p_{1_end}, and perpendicular to the line through those points. The two resulting faces are translated inwards by a distance equal to $w_{orig} - w_{bend}$, and then each is split into two new faces along the fold lines through points p_{0_end} and p_{1_end}. Finally, the inner two faces f_{0_folded} and f_{1_folded} are curved, such that they meet symmetrically at the peak of the bend p_{peak}, and maintain their original dimensions. (See Figure 4e). The orientation of the bend and the thickness and height of the bend are readjusted in real-time in response to user input, until the user's fingers are lifted off the screen. Subsequently, the position of the peak can be adjusted horizontally and vertically. Analogous to symmetric bending, a *symmetric fold* forms a peak at the midpoint of the two points of contact, where the portion of paper between the two pinched fingers is folded upwards, creating a folded peak in the middle.

4.2 Transforming, Grouping and Texturing Tools

Modeled single sheet components can be inserted into the scene and then transformed and assembled into more complex models by pinning and taping.

Pinning and Taping. A user pins two objects together by first tapping on a face f_0 of a model m_0. This creates a pin at projected location p_0, with normal equal to f_0's normal. A subsequent tap of face f_1 of a second model m_1 indicates a desired location and alignment of the first model. The two faces are pinned together; this is highlighted through animation. If m_0 has been pinned to the wrong side of f_1, it can be flipped over using a rotate gesture. Adjustments can subsequently be made by translating within the plane of f_1 or rotating about the pin axis. Taping is similar to pinning, except a user first taps an edge e_0 of m_0 (with two fingers, to distinguish from pinning), and then taps an edge e_1 of model m_1. The same transformation occurs as above, e_1 is constrained to lie on e_0. The user can rotate m_0 about the axis of the tape (the line passing through e_1), or translate along this axis.

Transforming and Duplicating. Individual and assembled models can both be transformed through uniform scaling (pinch gesture), translation within the plane of any face of the model (two-finger pan), and rotation about any edge of the model (one-finger pan). In addition, users can temporarily drop a pin or tape onto a model (without grouping), and translate and rotate about the pin/tape. Both individual models and groups can be duplicated using a one-finger drag operation.

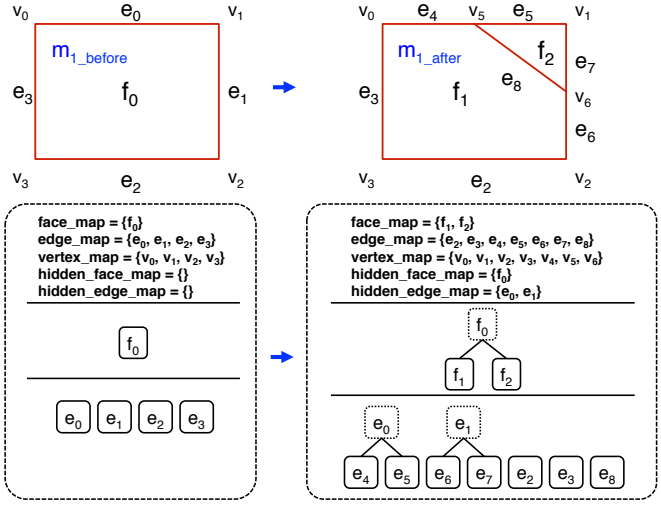


Fig. 5. Overview of the object data structure in our system. The right-hand side shows how the data structure is updated when a face of the model is split (e.g., through folding).

Coloring and Texturing. In the assembly mode, we provide coloring and texturing tools. In the coloring/texturing submode, a color picker is used, and a single one/two/three-finger tap on a face of a model will change the color of the face/model/group to the selected color. A long press over a model sets the current color to that of the indicated face. A list of texture thumbnails is provided. Once a texture is selected, it can be applied to a face/model/group in the same way as a color. The texture mapping can be adjusted using a pinch gesture.

4.3 Object Storage

Each object instance in our system (i.e., vertices; edges; faces) is stored dynamically, and has a unique identifier. Standard map containers store pointers to these objects for easy access. Each model stores three separate maps for its faces, edges and vertices, respectively. As shown in Figure 5, each face stores a map of its edges and an ordered list of its vertices, while edges store pointers to their associated faces and vertices.

4.4 Modeling Updates

Figure 5 shows an example of how the internal data structures of the system are updated after a crease operation. Original face f_0 is divided into new faces, f_1 and f_2. The outline of f_0 is split: vertices v_0, v_5, v_6, v_2, and v_3 now make up the outline of f_1, while v_5, v_1 and v_6 comprise the outline of f_2. We maintain a history of operations performed on the models, as shown in the tree structures in the same figure. Each face has a parent face and two children, all initially set to null. In this example, the parent face f_0 is labeled as hidden, and its left and right children are set to f_1 and f_2, respectively. Edges are divided and updated similarly: e_0 is split into e_4 and e_5, e_1 is split into e_6 and e_7, while a new

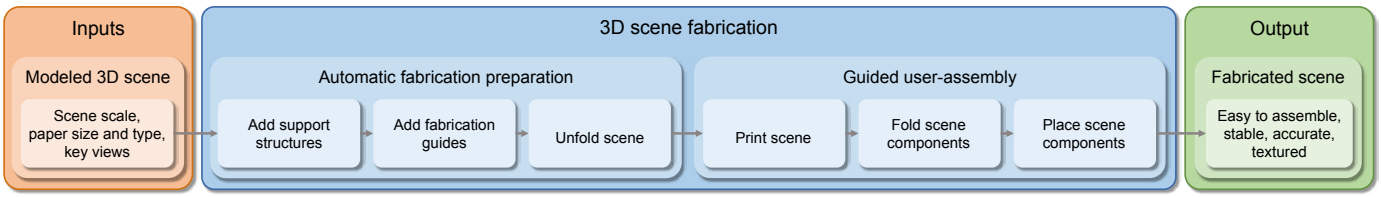


Fig. 6. Overview of our two-stage 3D fabrication process. Inputs are a 3D modeled scene along with its physical scale, paper type and size, and key scene views. Minimally-visible supporting structures are added for scene stability, while fabrication guides later help with assembly. The resulting meshes are unfolded, divided, and saved as images ready for printing. After cutting out the printed meshes, users assemble the scene by following the numbered guides in order. Folding and gluing along the indicated fiducials yields the fabricated 3D scene.

edge e_8 is created between f_1 and f_2. Parent and child edges are updated, as are edge map containers. If Figure 5 instead showed a corner bend operation, f_2 would subsequently be replaced by a curved surface, by recursively dividing it into a set of bend strips of near-equal length and parallel to edge e_8. Undoing an operation simply involves accessing the tree structures to determine the original objects.

5 Fabrication for Aesthetics and Stability

For casual users, a key aspect of any fabrication is preserving the approximate visual, characteristics of their scenes in a simple fabrication process. To achieve this, we introduce a small number of unobtrusive support structures to the model. A full physical simulation of the paper and glue fabricated system to optimize support placement would be computationally demanding. This would be inappropriate for the relative imprecision of a manually assembled papercraft-based 3D scene. Instead, we develop an algorithm that takes into account key scene views selected by the user. We choose supports to maintain the approximate scene structure, while reducing their visual impact. Reducing support visibility in key views is important in the construction of scenes (such as stage sets) as opposed to visibility constraints averaged over all views (e.g., as in [57]). The characterization and placement of the supports is diagrammed in Figure 6.

5.1 Overview

The primary type of support in our system is a thin, triangular prism that extends from the center of a model face to the ground plane. This shape provides a good balance between sturdiness and simplicity to fold. Cylindrical supports, for example, would be harder to fold at the base. The cross-section used can vary in size: thicker supports can support greater weight. In certain situations, a triangular prism cannot or should not be used. An example is when a face needing support is close to or fully upright. The angle between the support and the plane of the face would be too small. If there is a boundary edge on the lower end of the face, we can instead extend it downward until it reaches the ground, and end it with a folded tab. This is only used when the mass to support is small, and the distance from the boundary edge to the ground is small. Alternatively, we attach a regular support extended at an angle equal to half the angle between the face normal and the ground normal. This will generally provide more stability, but at an aesthetic cost. Lastly, if certain parts of a model are already resting on the ground, these faces are treated as already-existing supports with zero vertical length (ideal from an aesthetic perspective). This includes any face of a model that is either pinned to or resting on the ground.

Performing a physical bend operation precisely can be difficult. Though the system can approximate a bend with a number of thin strips folded perpendicular to the axis of the bend, creating such a large number of folds for each curved surface is impractical. Instead, we add a single, optional *curvature support* that attaches to the middle of the beginning and end strips of the bend. These are identical to regular supports, except they attach at the beginning and end of the curved surface.

5.2 Preserving Overall Scene Stability

A rigid object is in static equilibrium if the sum of the external forces $\mathbf{F}$ and the sum of the external torques $\boldsymbol{\tau}$ are both 0: $\sum \mathbf{F} = 0$ and $\sum \boldsymbol{\tau} = 0$. The only external force that directly affects a printed scene is gravity. We assume for the moment that a fabricated 3D model is rigid; in order to establish whether or not it is properly supported, we consider its *center of mass*. Since models in the scene are constructed from uniform material, the center of mass is located at the centroid of the model $c(m)$. The instability of the model m on ground g can be defined by $S(m, g)$ as the distance between the centroid of the model and the centroid of the model's base; the larger the distance between these two centroids, the more unstable the model. This is illustrated in Figure 7a. If $S(m, g)$ is below an acceptable threshold S_{min}, we can assume that the model is sufficiently stable.

$$S(m, g) = \|\text{proj}(c(m), g) - c(m \cap g)\| \leq S_{min}, \qquad (2)$$

where $c(m)$ is defined as the average of each face's centroid weighted by the face's area $a()$:

$$c(m) = \frac{1}{a(m)} \sum_{f \in F(m)} \left[a(f) \frac{\sum_{v \in V(f)} v}{|V(f)|} \right], \qquad (3)$$

and $\text{proj}(c(m), g)$ specifies the centroid of the model projected onto the ground plane. Table 1 summarizes notations and operators used for describing our method.

5.3 Preserving Local Scene Stability

Being made out of paper, the model can deform as a result of gravity. A model's unsupported vertices may bend downwards and deform it. Similarly, if a large face is supported at its edges, bending may occur in the middle. To correct for this, we must ensure that no point on a model's manifold

TABLE 1
Terminology used for describing properties of a scene

Term	Interpretation
$m_1 \cup m_2$	Combined model resulting from attaching model m_1 to m_2 along one or more of their faces. Same notation applies for combining a model to a support.
$m_1 \cap m_2$	Intersection of model m_1 and m_2, i.e., the planar region where they are attached. Same notation applies for combining a model to a support.
$F(m)$	Set of faces of a model m.
$V(m)$	Set of vertices of a model m; similarly, $V(f)$ is a set of vertices of a face f of a model.
$c(m)$	The ground-projected centroid of a model.
$a(m)$	Surface area of a model; similarly, $a(f)$ is the surface area of a face f of a model.

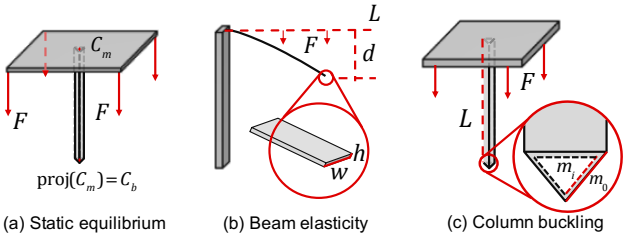
is too far away from a support. Accounting for this can also ensure general stability of a model.

We apply the principle of Euler-Bernoulli beam theory [58]. We treat a vertex of the sheet of paper extending away from a supporting structure as a cantilever beam. The following equation determines the vertical displacement d of such a beam with applied uniform force:

$$d = \frac{\mu G L^3}{8EI} = \frac{3\rho G L^4}{2Eh^2}. \quad (4)$$

Here, the mass of the beam is μ, G is the gravitational force due to gravity L is the length of the beam, E is the modulus of elasticity (or Young's modulus [58]) of the beam's material, and I is the second moment of area. For a regular cross-section (sheet of paper), $I = \frac{wh^3}{12}$, where w and h are the width and height of the beam, respectively. The mass of the paper can also be computed as $\mu = \rho w h L$, where ρ is the mass density of the beam, giving the final equation for the deflection of the beam. See Figure 7b.

The height and density of the paper can be measured, and we can experimentally determine Young's modulus by observing beam deflections at different lengths. For the two types of paper we primarily worked with, printer paper and photo paper, we found that $E = 2.3$ GPa and $E = 1.5$ GPa, respectively. Knowing these elasticity coefficients, we can set a maximum allowable value for d substitute in the density of the paper, and get a maximum possible length of a cantilevered beam (unsupported sheet of paper) before a support needs to be added. We can determine if all vertices of a model are close enough to a supporting structure to prevent them from bending unacceptably.



(a) Static equilibrium (b) Beam elasticity (c) Column buckling

Fig. 7. Aspects of structural stability used to determine the necessary support of a fabricated scene. (a) illustrates *static equilibrium*: an object will be unstable if the ground-projected centroid of the model is too far from the centroid of its base. (b) demonstrates beam elasticity: for an outward-extending cantilever beam, its displacement is determined by its material *elasticity*. (c) illustrates column buckling, which occurs when too much weight is placed on a column.

We measure the deformation $D(m)$ of a model due to its fabrication material bending at inadequately supported areas. Prior to adding any supports to m, we subdivide its mesh to find a set of roughly evenly distributed vertices on the mesh. The density of the vertices is proportional to the elasticity of the material, as more supports will likely be needed. Given a subset of all possible supports $X_m = x_0, x_1, ..., x_n \subset X$, this term is then computed as

$$D(m) = \frac{1}{|V(m)|} \sum_{v \in V(m)} \min_{x \in X_m} d_D(\|v - c(m \cap x)\|, c(m \cap x)), \quad (5)$$

where

$$d_D(l, f) = \begin{cases} \frac{l}{l_m(v)} - 1 & \text{if } l \leq 2 l_m(v) \\ 1 & \text{otherwise} \end{cases}, \quad (6)$$

and $l_m()$ computes the maximum distance a support can be away from an unsupported vertex. Based upon the input area, the paper type chosen by the user and a maximum acceptable vertical displacement d_m, l_m is computed using Equation (4). d_D will then range from 1 (large deformation at vertex) to -1 (no deformation at vertex). Overall, $D(m)$ averages the deformations across all vertices.

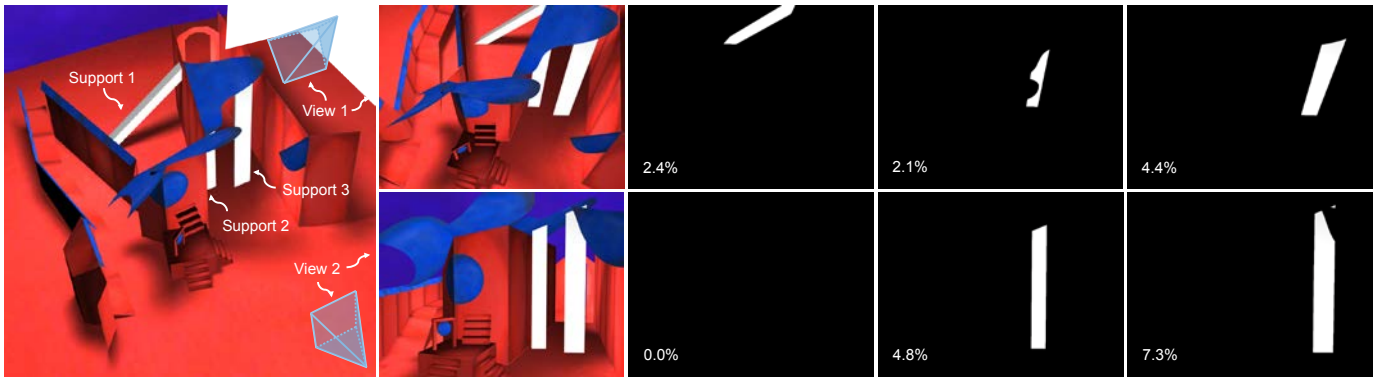
Another engineering principle is column buckling (see Figure 7(c)). If the cross-section of the structure of a support is too narrow relative to its own height, it may *buckle* or collapse. Euler's formula for the *critical buckling load* of a column with fixed ends is: $F = 4\pi^2 EI/L^2$, and is dependent upon the elasticity of the material from which the column was fabricated. E and I are defined as in Equation (4) , and L is the length of the supporting structure. In this case, since supporting structures are shaped as hollow triangular prisms, with their base being an equilateral triangle, $I = \frac{\sqrt{3}}{32}({m_o}^4 - {m_i}^4)$, where m_o and m_i are the outer and inner side lengths of the triangle, and $m_i \approx m_o - 2\delta_m$, where δ_m is the width of the paper used for creating the support.

5.4 Preserving Scene Aesthetics

To preserve aesthetics, we focus on minimizing the visibility of a support from one or more user-selected views. Minimizing the visibility of the supports is often less important globally than it is for specific, important views of the scene (e.g., the front of a stage set). Ideally, added supports are entirely hidden from view, but at a minimum, they should not be visible from a set of key views.

We calculate the visibility $I(m)$ of a model's supports in proportion to the visibility of the model itself. We render a color-coded version of the model with all its supports: white pixels are supports, transparent pixels are ignored, while the other pixels are rendered black and totaled to get the visibility of the model. We normalize by the remaining quantity of visible pixels of the model. When considering multiple views, the average of the ratios computed from each corresponding rendering is calculated. See Figure 8 for an example. The formula is as follows:

$$I(m) = \frac{\sum_{r \in R(m)} \sum_{p \in P(r)} [p_i = 255 \wedge p_\alpha > 0]}{\sum_{r \in R(m)} \sum_{p \in P(r)} [p_\alpha > 0]}, \quad (7)$$



(a) 3D scene with two user-selected key views and three representative support candidates

(b) Rendering from user-selected viewpoints

(c) Support 1 visibility I(m) = 0.012, D(m) = 0.77

(d) Support 2 visibility I(m) = 0.035, D(m) = 0.68

(e) Support 3 visibility I(m) = 0.059, D(m) = 0.81

Fig. 8. An example of how support visibility is computed and its impact on support selection. A scene with two user-selected key views and three candidate supports is shown in (a). A rendering of the scene from these two viewpoints in shown in (b). In (c)-(e), a visibility rendering from each viewpoint of the three supports is shown, respectively. The ratio of visible support area to visible scene area, averaged across the selected views, determines the visibility cost for each support. These are combined with the stability cost to select the best support. Support 1 is the least visible (lowest $I(m)$), while support 2 most significantly decreases the instability of the scene (lowest $D(m)$). With visibiilty and stability both taken into account, support 2 is selected as the best candidate.

where p_i is a pixel intensity, p_α is the alpha transparency value, $R(m)$ is a set of renderings of model m, and $P(r)$ is the array of pixels comprising a rendering r.

Note that as we increase the number of views taken into account, each support is more likely to be seen from at least one of them. We found experimentally that accounting for more than 3–4 views did not yield significantly different results than support visibility been ignored.

6 3D Scene Fabrication

In this section, we describe how scenes are prepared for fabrication. Our fabrication method automatically prepares a virtual scene for printing on sheets of letter-sized paper using a conventional printer. Only scissors and glue are required to assemble the physical scene.

Overview. In the fabrication mode of the system, users specify the scale at which they want to print their scene, the paper type they will be using, and what views they want to consider for adding support structures. Thumbnails, like the ones seen in Figure 8, represent user-bookmarked views of the scene that are used in the support structure calculations. In addition to support structures, a set of *assembly indicators* are added during the fabrication stage to guide users to correctly assemble their fabricated models. The meshes comprising the 3D scene are subsequently unfolded and, if necessary, segmented, to allow printing.

6.1 Determining Fabrication Supports

Overview. The general idea behind the algorithm for determining fabrication supports is to do the following: (1) start with no supports, other than the ones already present in the scene, (2) look at a large pool of candidate supports that could potentially be added to the scene to improve stability; (3) add a support (from this pool of candidates) that best improves the stability of the model while not disrupting the aesthetic look of the model; (4) repeat step (3), choosing the next "best" support (from the ones not already chosen, until acceptable model stability is reached.

Initialization. Any 3D scene contains a set of component models M; let us refer to any individual model in the scene as $m \in M$ and the ground plane (itself a model) g. We start by finding a larger pool of possible supports X that could be added to each model m of the scene. Each support $x \in X$ has one end attached to the ground, and the other attached to m. To get a distribution of initial supports, we segment the faces of the scene until all edge lengths are below a predefined length (larger than the maximum edge length of the base of a support), and then attach supports at several different thicknesses to the centroid of each of these newly-created faces. All faces of the model already lying on the ground are treated as pre-selected supports.

Maximization. We want to determine a subset of all possible supports $X_m = x_0, x_1, ..., x_n \subset X$, for which the aesthetic look of the model is maintained, while the stability of m is kept high. We formulate a maximization problem as follows:

$$\max_{X_m \subset X} \left[Z(m, g) - Z(m_n, g) \right] , \tag{8}$$

where $Z(m, g)$ is the following cost function that factors in the visibility of supports and the instability of a fabricated model m on a ground g, and $m_i = m \cup x_0 \cup x_1 \cup \ldots \cup x_i$:

$$Z(m, g) = k_I I(m) + k_D D(m) , \tag{9}$$

where $I(m)$ computes the visibility of the model's supports relative to the visibility of the model, and $D(m)$ computes the overall deformation and instability of the model.

The two terms are weighted, with the constraint that $k_I + k_D = 1$ in Equation (9). We experimentally determine the weights for each term of the scoring function Z. We tested different combinations to see what provided the best compromise between quickly achieving stability but also retaining the aesthetic qualities of the model. Figure 10 gives an example of how the supports added to a 3D model can vary depending on the weights of visibility versus stability. The weights that gave us the best results were $k_D = 0.7$ and $k_I = 0.3$.

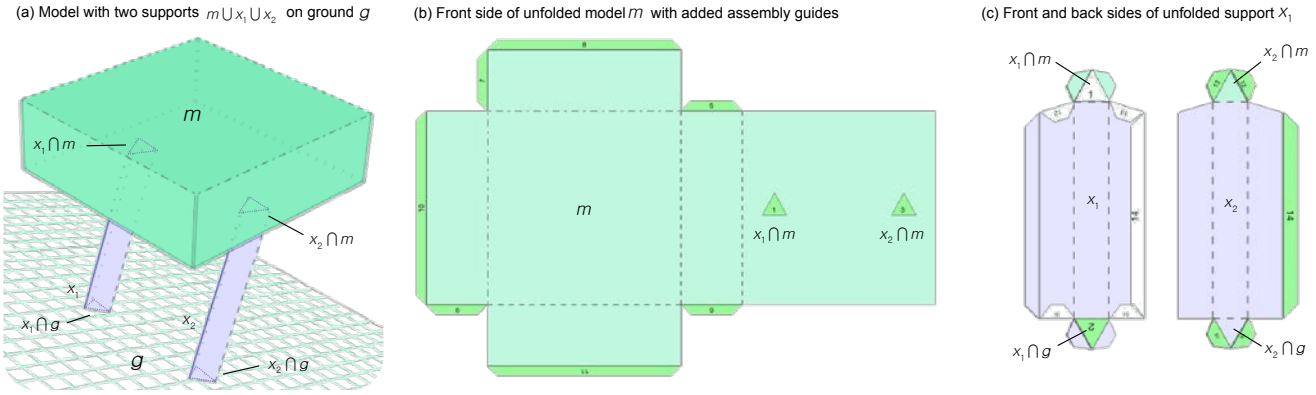
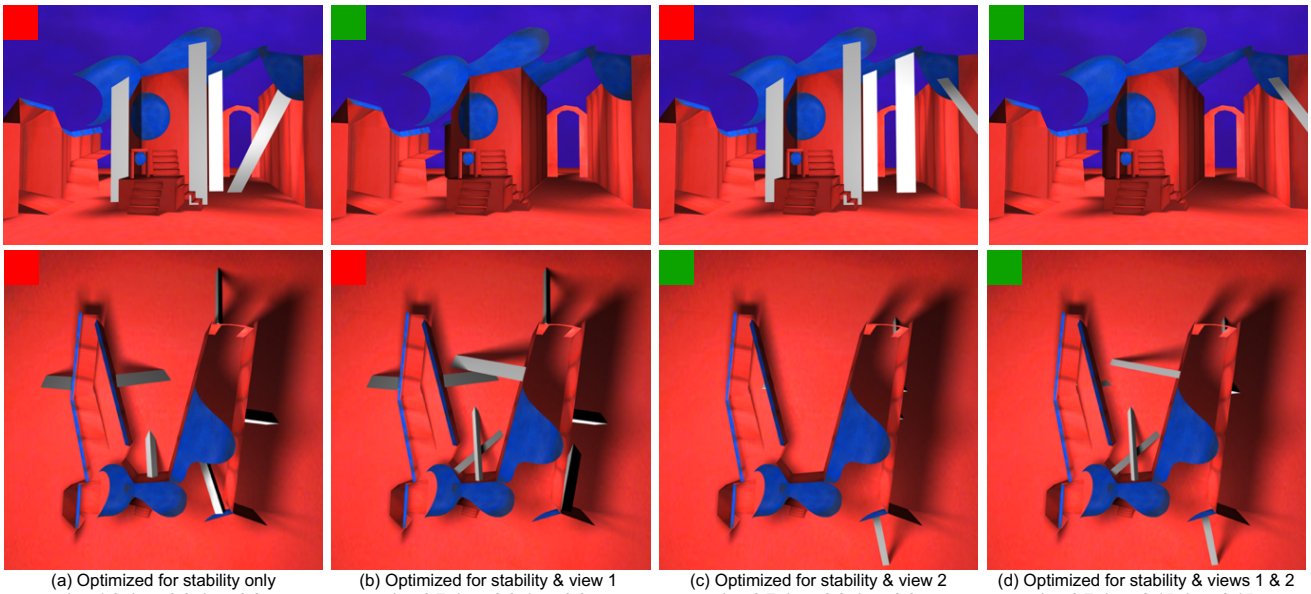


Fig. 9. In (a), a basic example model with two added supports, denoted $m \cup x_1 \cup x_2$, is shown standing on the ground plane g. (b) shows the unfolded model m with added assembly guides, while (c) shows both sides of the unfolded support x_1.



(a) Optimized for stability only k_D= 1.0, k_{I1} = 0.0, k_{I2} = 0.0
(b) Optimized for stability & view 1 k_D= 0.7, k_{I1} = 0.3, k_{I2} = 0.0
(c) Optimized for stability & view 2 k_D= 0.7, k_{I1} = 0.0, k_{I2} = 0.3
(d) Optimized for stability & views 1 & 2 k_D= 0.7, k_{I1} = 0.15, k_{I2} = 0.15

Fig. 10. Comparing the placement of support structures (rendered in white) with different term weights. In (a), the support placement is only optimized towards scene stability, and does not attempt to hide the supports from either view. In (b) and (c), support visibility is minimized for views 1 (top) and 2 (bottom), respectively. In (d), the support position is optimized for the visibility of both views simultaneously, as well as stability.

We can expand Equation (8) into the following:

$$\max_{X_m \subset X} [(Z(m,g) - Z(m_0,g)) + (Z(m_0,g) - Z(m_1,g)) + \ldots + (Z(m_{n-1},g) - Z(m_n,g))]. \quad (10)$$

In order to avoid testing 2^n subsets of X to find the true maximum, we approximate this maximization with

$$\max_{x_0 \in X}(Z(m,g) - Z(m_0,g)) + \max_{x_1 \in X \setminus \{x_0\}}(Z(m_1,g) - Z(m_0,g)) + \ldots + \max_{x_n \in X \setminus \{x_0, x_1, \ldots x_{n-1}\}}(Z(m_n,g) - Z(m_{n-1},g)). \quad (11)$$

This allows us to iteratively test each of the remaining supports in X, choose the one that results in the biggest increase in stability (or decrease in instability computed by Z), and add this to X_m.

Termination. After each new support is added, we check whether the result of Equation (2) falls below a minimum threshold, and we verify that all sheets of paper are properly supported to avoid excessive bending of the scene. Once the termination conditions are met, no further supports are added to the scene. Note that supports cannot pass through any model in the scene, nor through other supports. At each iteration of the algorithm, we discard supports remaining in X that intersect the previously added support.

Supporting Structure Limitations. There are several limitations to the algorithm for supporting structures. First, it does not allow for "cascading" support structures that fall between two surfaces of a model, instead of between the model and the ground. In some cases, this would be preferred, or in fact required. There are potential models where the algorithm would still be unable to find a solution that satisfies all three termination conditions. This could be improved by considering the addition of more than one support at a time.

6.2 Addition of Assembly Guides

Since none of the modeling operations in the system deform the model, a user should clearly be able to recreate a hard-copy version of a model simply by redoing the equivalent operations on the printed out sheets of paper (e.g., folding, bending, etc.). However, there are several difficulties. In

the case of pins and tape, most models will have a large number of edges and faces, making it difficult for the user to recognize which two should be taped or pinned together, respectively. Second, in the case of folding, it can be difficult to estimate the precise angle and curvature, respectively.

For these reasons, we add pairs of *assembly guides* to each model. An assembly guide is either a new face added to the model, or a portion of a face of an existing model. The guides are shaded green, always appear only on one side of the model, and each has a number printed on them. Guides with the same number and with matching outlines are considered a pair; each pair should be glued together to reconstruct the model. Since the printouts are double-sided, if a guide appears on a portion of a face, the texture belonging to that area is transferred to the back of the other guide in the pair to ensure no loss of texture. We create three types of assembly guides, for pins, tape and folds. Figure 9 shows an example of unfolded supports with assembly guides.

Tape Guides. If two faces have been taped together, or if the unfolded model was divided earlier, a tape guide is added along each of the two taped edges. One of the guides extends outward from the original model, while the other extends inward, overlapping the face of the model.

Pin Guides. If two faces have been pinned together, a pin guide is added to each of these faces, representing the overlap of the two faces. This includes pins between the added supporting structures and both the ground and a face of the supported model. Pin guides do not add any new faces to the model.

Folding Guides. In recreating each component model of their 3D scene, users mimic the folding operations performed in the system by physically folding the printed mesh cutouts. Though folding is straightforward, it can be difficult to replicate the precise angle. To be able to replicate a particular fold unambiguously, it is enough to check all the edges originating from the same vertex as one of the vertices of the fold. If all of these edges are folds themselves (alternatively, if none of them are boundary edges), then as long as we know the direction of the folds, there is only one possible solution. We define such a vertex (triangle fan) as *locally-closed*; it is otherwise *locally-open*. For each fold with both vertices belonging to locally-open portions of the mesh, we add a *guiding face* for precise angle recreation. If a mesh is locally-open, the vertex will have exactly two boundary edges connected to it. We add a triangular face that extends from one of the two boundary edges, such that its own opposite edge will align exactly with the second boundary edge. A complementary tab extends from the other boundary edge. Guiding faces are temporary indicators of the angle of a fold, and can be cut off once the corresponding fold has been creased.

6.3 Mesh Unfolding

The models created using the system are unfolded into flat sheets of paper. Due to possible extending operations, parts of an unfolded model may overlap. If this is the case, the model is divided along an edge between the overlapping faces, and internally, the two models are taped along this edge. Similarly, based on the scale of the scene defined by user through scaling the grid of the ground plane, if an unfolded model is too big to fit on a single sheet of paper, it is divided along one or more edges. Note that we could also optimize this process using a host of existing mesh unfolding techniques and subsequent packing techniques. The result of this unfolding procedure is a set of non-overlapping, polygonally-shaped sheets of paper. Each side of these sheets is stored as an image on the iPad (we want to print both sides of an open mesh).

6.4 Order of Assembly

Once the meshes are printed, the user cuts out all the unfolded meshes. Then, the user follows a sequence of folding and assembly steps. The system produces one (of potentially many) numbered sequence of steps for the user.

The creation of the supports is prioritized, placing them on the ground as a scaffold for the model. This also allows the user to get a rough sense of the layout, and a better understanding of subsequent steps. Afterwards, the models are folded in order, starting from the smallest and ending with the largest, as it is generally easier to place smaller models. The order of operations mimics the order of the modeling operations, making the process more natural to the user. Curvature supports are added last to a fabricated component of the scene. Pinning and taping operations that combine component models together are performed after both models have been folded, before they are collectively glued to their supports.

7 Results

PaperCraft3D went through a natural, user-guided evolution. Once we had a stable, preliminary version of our system, we showed our system to four modelers: three casual 3D modelers with only about three months of modeling experience, and a professional designer with extensive modeling experience. These four modelers learned to use our system (5–10 hours of practice), and each created a selection of 3D models using our system, while providing feedback. This feedback resulted in modifications to the system.

Implementation. Our system was implemented natively for a tablet device with a multi-touch display (a third generation iPad). The majority of user interaction with the system occurs through single- and multi-finger gestures. Our system is implemented in a mix of C++ and Objective C, with OpenGL ES as the rendering API. The data structures of [1] are naturally extended to include all additional fabrication components. Since support structures and assembly guides can be treated as components themselves, they are stored in exactly the same way as any of the user's models. These structures are either pinned or taped to the faces to which they are attached.

7.1 Modeling and Fabrication Results

To illustrate the potential and limitations of our method for producing a physical realization of a virtual design, we show modeling and fabrication results for three scenes

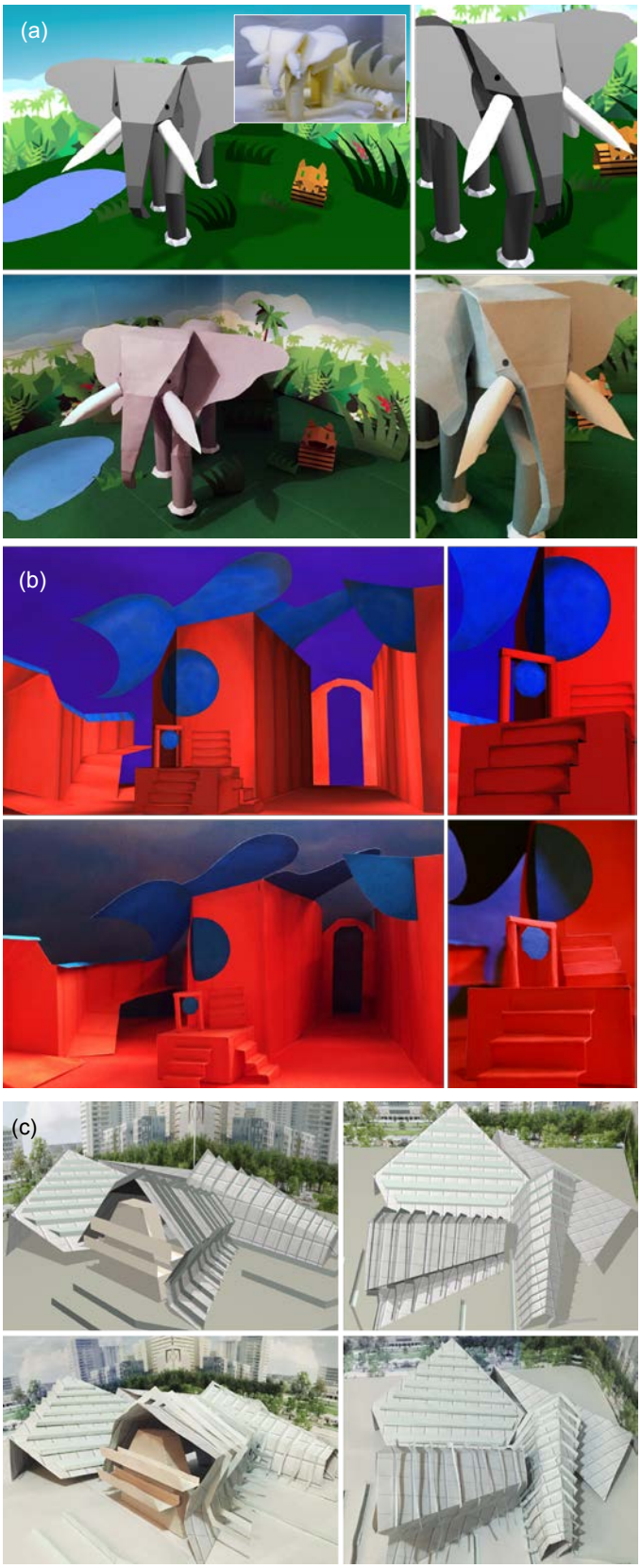


Fig. 11. (a) A 3D scene of animals in a jungle is shown, with the first row presenting the rendered 3D scene, and second row showing the fabricated scene (Inset: 3D printer output). (b) Presented here are rendered and fabricated versions of a David Hockney inspired stage set are shown in the first and second row, respectively. (c) Shown here is an innovative design for a new media center. The top row shows the rendered 3D scene, while the bottom row shows the fabrication.

of varying type and complexity: a jungle scene in Figures 11a, a stage set in Figure 11b, and a proposed design for a media center in Figure 11c. Modellers spent an estimated average of 4–5 hours conceptualizing and creating each scene. In each Figure, we compare computer renderings of the design and photographs of the fabricated result. The size of the scenes ranged from 55×43×26 cm (stage set) to 82×73×34 cm (jungle). While the overall fabrication time was somewhat longer than we originally expected, each part of the process could be sped up in the future. Cutting out the meshes could be sped up with, for example, a cutting machine. Precise folding could similarly be simplified if small perforations were automatically added to the folds after printing. Gluing time could be somewhat cut down if the scene were printed on larger paper, as fewer models would need to be segmented prior to printing.

TABLE 2
Fabrication timing for each of the three results scenes

Scene/ Component	Cutting (hours)	Folding (hours)	Gluing (hours)	Total (hours)
Stage set	2.00	1.50	1.50	5.00
Media center	5.00	2.00	3.50	10.50
Jungle	4.00	4.50	4.00	12.00
Elephant head	1.25	2.25	2.00	5.50
Elephant body	0.75	1.00	0.75	2.50
Tiger	0.50	0.25	0.25	1.00
Plants	1.00	0.25	0.25	1.50

7.2 Accuracy of Fabrication

To estimate the accuracy of our fabrication tool, we picked ten pairs of vertices in our fabricated stage set scene, measured the distances between these pairs of points, and compared them to the distances between the corresponding virtual vertices. Figure 12 shows the chosen points, while Table 3 lists the measurements and percent error in each case. Across the ten measurements, we found that our fabrication had an average percent error of 1.5% ±1%.

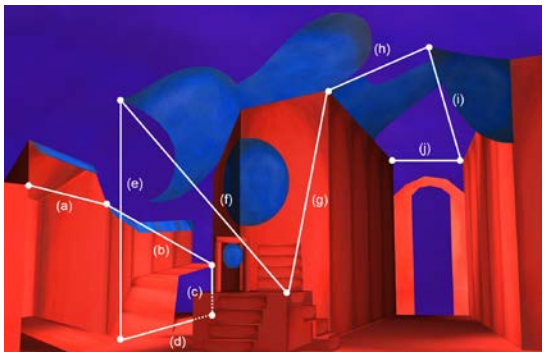


Fig. 12. Vertices and distances chosen to compare the virtual and fabricated scene measurements of the stage set model. As seen in Table 3, these distances are all accurate to within 3%.

7.3 Comparison to 3D Printing

To compare our fabrication method to 3D printing, we 3D printed the jungle scene in the inset of Figure 11a. We used a commodity 3D printer, Opencreators Almond. The fabrication using our method far exceeded the maximum printable

TABLE 3
Measurements to estimate stage set fabrication accuracy

Distance	Virtual Scene (cm)	Fabrication (cm)	Percent Error
(a)	6.2	6.2	0.0%
(b)	23.2	23.1	0.4%
(c)	4.0	4.1	2.5%
(d)	23.2	23.5	1.3%
(e)	10.7	10.4	2.8%
(f)	10.8	10.9	0.9%
(g)	10.9	11.0	0.9%
(h)	6.7	6.9	3.0%
(i)	26.2	26.5	1.1%
(j)	5.9	5.8	1.7%

size of the 3D printer. Thus, we instead chose to print at a 1:6 scale, or an approximate size of 12.1×13.6×6.3 cm. The total time it took to 3D print the scene was 15.5 hours: approximately 1 hours to prepare the scene for printing, 14 hours for the actual printing, and 0.5 hours to clean up the printed scene.

The 3D printed scene has many artifacts, particularly at model endpoints such as the elephant trunk, its tusks, and the ends of the plants. There are other artifacts because the printer generates its own supports. These must be manually removed after printing, a task difficult to do when there are many small-scale features. In addition, thin structures such as the elephant eyes cannot be produced accurately. Lastly, there is obviously a lack of texture in the 3D printed scene, which significantly reduces its aesthetic quality.

8 Limitations and Future Work

Modeling. Though we used papercrafting as an inspiration, it was often not appropriate to directly simulate interactions, but rather to use gestures appropriate with a multitouch device. For example, the crease gesture in PaperCraft3D uses a pinch gesture to define the orientation of the crease, even though this does not mimic physically creasing paper. In some cases, gestures with multiple fingers led to significant screen occlusion, affecting our choices. Users observed that a mouse provides better precision than hand gestures, though in general, this did not negatively affect their modeling process. This agreed with our assumption that while the precision of a tablet device may not be sufficient for a professional modeler, it should suffice for a casual one.

Fabrication. While we expected the thickness of the paper to impact the sturdiness of the scene, there is an additional tradeoff that we observed in practice. While thicker paper minimizes deformations in the model, it is also more difficult to make small folds, and it is more likely that the fold will damage the paper. While this is not noticeable on plain paper, the effect on textured paper is visible white lines along the folds. In addition, folds made with thicker paper influence distance measurement; if not accounted for, the fiducials will not perfectly align with each other.

The glue used during assembly similarly influences both the aesthetics and the accuracy of the scene. The thicker the paper, the stronger the glue required, particularly when glued along a fold. On the other hand, glue applied to thinner paper is more likely to damage the outer surface, leaving visible marks. We typically paired conventional glue sticks with thinner paper, and stronger liquid glue for thicker paper.

Conclusion. In summary, we have presented PaperCraft3D, a novel 3D modeling and fabrication system designed specifically for multi-touch interfaces. Our system is easily learned by experienced and casual modelers alike, allowing users to create a wide range of developable surfaces and subsequently fabricate the model as a 3D scene. In particular, our fabrication tool maintains the spatial and rigid properties of a scene, despite the fact that the scene is constructed out of everyday paper.

Several interesting future directions of our modeling system remain, such as improving input precision, experimenting with additional forms of curvature, improving rendering efficiency and quality, extensions to support animation, and hardcopy output. In the fabrication tool, the supporting structure optimization could be extended to allow for inter-model supports, and we could incorporate some of the planar section ideas from related works to add new types of supports. We could explore fabrication with sturdier materials, which are less prone to deformities during assembly and over time.

Acknowledgment

Julie Dorsey acknowledges support from the National Science Foundation under Award 1018470. Min H. Kim acknowledges Korea NRF grants (2016R1A2B2013031, 2013M3A6A6073718) and additional support by KOCCA in MCST of Korea, Cross-Ministry Giga KOREA Project (GK17P0200), and an ICT R&D program of MSIT/IITP of Korea (2017-0-00072, 2016-0-00018). We also thank Daniel Jeon for helping, and Jennifer Lackie for helping us with scene fabrication.

References

[1] P. Paczkowski, J. Dorsey, H. Rushmeier, and M. H. Kim, "Paper3D: Bringing casual 3D modeling to a multi-touch interface," in *Proc. ACM UIST*, Honolulu, USA, 2014, pp. 23–32.
[2] C. Li, H. Lee, D. Zhang, and H. Jiang, "Sketch-based 3d modeling by aligning outlines of an image," *Journal of Computational Design and Engineering (JCDE)*, vol. 3, no. 3, pp. 286 – 294, 2016.
[3] J. Mitani, "Tama software pepakura designer," http://www.tamasoft.co.jp/pepakura-en/, 2018.
[4] J. Keijser, S. Carpendale, M. Hancock, and T. Isenberg, "Exploring 3D interaction in alternate control-display space mappings," in *Proc. Symp. on 3D User Interfaces*, 2007, pp. 526–531.
[5] H. Aoki, J. Mitani, Y. Kanamori, and Y. Fukui, "Ar based ornament design system for 3d printing," *Journal of Computational Design and Engineering (JCDE)*, vol. 2, no. 1, pp. 47 – 54, 2015.
[6] S. K. Lee, W. Buxton, and K. C. Smith, "A multi–touch three dimensional touch–sensitive tablet," in *Proc. ACM CHI*, 1985, pp. 21–26.
[7] O. K.-C. Au, C.-L. Tai, and H. Fu, "Multitouch gestures for constrained transformation of 3D objects," *Comput. Graph. Forum*, vol. 31, no. 2, pp. 651–660, 2012.
[8] A. Cohé and M. Hachet, "Beyond the mouse: Understanding user gestures for manipulating 3D objects from touchscreen inputs," *Comput. Graph.*, vol. 36, no. 8, pp. 1119–1131, Dec. 2012.
[9] M. S. Hancock, M. S. T. Carpendale, and A. Cockburn, "Shallow-depth 3D interaction: design and evaluation of one-, two- and three-touch techniques," in *Proc. ACM CHI*, 2007, pp. 1147–1156.
[10] J. Liu, O. K.-C. Au, H. Fu, and C.-L. Tai, "Two-finger gestures for 6DOF manipulation of 3D objects," *Comput. Graph. Forum*, vol. 31, no. 7-1, pp. 2047–2055, 2012.

[11] A. Martinet, G. Casiez, and L. Grisoni, "The design and evaluation of 3D positioning techniques for multi-touch displays," in *Proc. the Symposium on 3D User Interfaces*, 2010, pp. 115–118.

[12] S. H.-H. Chang, L. Stuart, B. Plimmer, and B. Wünsche, "Origami simulator: a multi-touch experience," in *ACM CHI Extended Abstracts*, 2009, pp. 3889–3894.

[13] A. Joshi, G. Robertson, B. Wünsche, and B. Plimmer, "Bubbleworld builder - 3D modeling using two-touch and sketch interaction," in *Proc. GRAPP*, 2010, pp. 116–122.

[14] B. R. De Araùjo, G. Casiez, and J. A. Jorge, "Mockup Builder: Direct 3D modeling on and above the surface in a continuous interaction space," in *Proc. Graphics Interface*, 2012, pp. 173–180.

[15] B. Walther-Franks, M. Herrlich, and R. Malaka, "A multi-touch system for 3D modelling and animation," in *Proc. Smart Graphics*. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 48–59.

[16] Y. Li, X. Luo, Y. Zheng, P. Xu, and H. Fu, "Sweepcanvas: Sketch-based 3d prototyping on an rgb-d image," in *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '17. New York, NY, USA: ACM, 2017, pp. 387–399. [Online]. Available: http://doi.acm.org/10.1145/3126594.3126611

[17] K. Kin, T. Miller, B. Bollensdorff, T. DeRose, B. Hartmann, and M. Agrawala, "Eden: A professional multitouch tool for constructing virtual organic environments," in *Proc. ACM CHI*. New York, NY, USA: ACM, 2011, pp. 1343–1352.

[18] Q. Sun, J. Lin, C.-W. Fu, S. Kaijima, and Y. He, "A multi-touch interface for fast architectural sketching and massing," in *Proc. ACM CHI*. New York, NY, USA: ACM, 2013, pp. 247–256.

[19] A. D. Wilson, S. Izadi, O. Hilliges, A. Garcia-Mendoza, and D. S. Kirk, "Bringing physics to the surface," in *Proc. UIST 2008*. ACM, 2008, pp. 67–76.

[20] E. J. Nitsch, "When pigs fly: a study of computer generated paper folding," M.S. Thesis, Texas A&M University, 2008.

[21] T. Tachi, "Rigid-foldable thick origami," *Origami*, vol. 5, pp. 253–264, 2011.

[22] J. Mitani, "The folded shape restoration and the rendering method of origami from the crease pattern," in *Proc. Int. Conf. on Geometry and Graphics*, 2008, pp. 1–7.

[23] S.-Y. Miyazaki, T. Yasuda, S. Yokoi, and J.-I. Toriwaki, "An origami playing simulator in the virtual space," *J. of Vision and Computer Animation*, vol. 7, no. 1, pp. 25–42, 1996.

[24] T. Igarashi, S. Matsuoka, and H. Tanaka, "Teddy: A sketching interface for 3D freeform design," in *Proc. SIGGRAPH*, 1999, pp. 409–416.

[25] M. Lau, J. Mitani, and T. Igarashi, "Digital fabrication," *Computer*, vol. 45, no. 12, pp. 76–79, 2012.

[26] J. S. Sadar and G. Chyon, "3D scanning and printing as a new medium for creativity in product design," in *Proc. Conf. Creativity and Innovation in Design (DESIRE)*. New York, NY, USA: ACM, 2011, pp. 15–20.

[27] C. Mota, "The rise of personal fabrication," in *Proc. ACM Conf. Creativity and Cognition*. New York, NY, USA: ACM, 2011, pp. 279–288.

[28] R. Schmidt and N. Umetani, "Branching support structures for 3d printing," in *ACM SIGGRAPH 2014 Studio*. ACM, 2014, pp. 9:1–9:1.

[29] N. Umetani and R. Schmidt, "Cross-sectional structural analysis for 3d printing optimization," in *SIGGRAPH Asia 2013 Technical Briefs*, ser. SA '13. New York, NY, USA: ACM, 2013, pp. 5:1–5:4.

[30] J. Mitani and H. Suzuki, "Making papercraft toys from meshes using strip-based approximate unfolding," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 259–263, Aug. 2004.

[31] I. Shatz, A. Tal, and G. Leifman, "Paper craft models from meshes," *The Visual Computer*, vol. 22, no. 9-11, pp. 825–834, 2006.

[32] R. J. Lang, "Treemaker," http://www.langorigami.com/science/computational/treemaker/treemaker.php, 2013.

[33] D. Chen, P. Sitthi-amorn, J. T. Lan, and W. Matusik, "Computing and fabricating multiplanar models," *Computer Graphics Forum*, vol. 32, no. 2pt3, pp. 305–315, 2013.

[34] K. Hildebrand, B. Bickel, and M. Alexa, "Crdbrd: Shape fabrication by sliding planar slices," *Comp. Graph. Forum*, vol. 31, no. 2pt3, pp. 583–592, 2012.

[35] G. Saul, M. Lau, J. Mitani, and T. Igarashi, "Sketchchair: An all-in-one chair design system for end users," in *Proc. Int. Conf. Tangible, Embedded, and Embodied Interaction*. ACM, 2011, pp. 73–80.

[36] F. Massarwi, C. Gotsman, and G. Elber, "Papercraft models using generalized cylinders," in *Computer Graphics and Applications, 2007. PG '07. 15th Pacific Conference on*, Oct 2007, pp. 148–157.

[37] S. Iizuka, Y. Endo, J. Mitani, Y. Kanamori, and Y. Fukui, "An interactive design system for pop-up cards with a physical simulation," *Vis. Comput.*, vol. 27, no. 6-8, pp. 605–612, Jun. 2011.

[38] S. Okamura and T. Igarashi, "An interface for assisting the design and production of pop-up card," in *Proc. Int. Symp. Smart Graphics (SG)*, 2009, pp. 68–78.

[39] X.-Y. Li, C.-H. Shen, S.-S. Huang, T. Ju, and S.-M. Hu, "Popup: Automatic paper architectures from 3D models," in *Proc. ACM SIGGRAPH 2010*, 2010, pp. 111:1–111:9.

[40] Y. Igarashi, T. Igarashi, and H. Suzuki, "Knitting a 3D model," *Computer Graphics Forum*, vol. 27, no. 7, pp. 1737–1743, 2008.

[41] Y. Igarashi, T. Igarashi, and J. Mitani, "Beady: Interactive beadwork design and construction," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 49:1–9, 2012.

[42] Y. Igarashi and T. Igarashi, "Designing plush toys with a computer," *Commun. ACM*, vol. 52, no. 12, pp. 81–88, Dec. 2009.

[43] E. D. Demaine and T. Tachi, "Origamizer: A Practical Algorithm for Folding Any Polyhedron," in *33rd International Symposium on Computational Geometry (SoCG 2017)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), B. Aronov and M. J. Katz, Eds., vol. 77. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, pp. 34:1–34:16.

[44] R. Guseinov, E. Miguel, and B. Bickel, "Curveups: Shaping objects from flat plates with tension-actuated curvature," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 64:1–64:12, Jul. 2017.

[45] H. Shimanuki, T. Watanabe, K. Asakura, and H. Sato, "Construction and analysis of easily fold-able processes for computer-aided origami," in *Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication*, ser. IMCOM '17. New York, NY, USA: ACM, 2017, pp. 96:1–96:8.

[46] Y. Huang, M. D. Gross, E. Y.-L. Do, and M. Eisenberg, "Easigami: A reconfigurable folded-sheet TUI," in *Proc. Int. Conf. Tangible and Embedded Interaction*. New York, NY, USA: ACM, 2009, pp. 107–112.

[47] Y. Huang and M. Eisenberg, "Easigami: Virtual creation by physical folding," in *Proc. Int. Conf. Tangible, Embedded, and Embodied Interaction*. New York, NY, USA: ACM, 2012, pp. 41–48.

[48] G. Saul, C. Xu, and M. D. Gross, "Interactive paper devices: End-user design & fabrication," in *Proc. Int. Conf. Tangible, Embedded, and Embodied Interaction*. New York, NY, USA: ACM, 2010, pp. 205–212.

[49] S. Mueller, P. Lopes, and P. Baudisch, "Interactive construction: Interactive fabrication of functional mechanical devices," in *Proc. ACM UIST*, 2012, pp. 599–606.

[50] K. D. Willis, C. Xu, K.-J. Wu, G. Levin, and M. D. Gross, "Interactive fabrication: New interfaces for digital fabrication," in *Proc. Int. Conf. Tangible, Embedded, and Embodied Interaction*. ACM, 2011, pp. 69–72.

[51] A. Zoran and J. A. Paradiso, "Freed: A freehand digital sculpting tool," in *Proc. ACM SIGCHI*, New York, NY, USA, 2013, pp. 2613–2616.

[52] J. Lin, H. Nishino, and T. Kagawa, "A digital fabrication assistant for 3D arts and crafts," in *Int. Conf. Broadband and Wireless Computing, Communication and Applications (BWCCA)*, Nov 2014, pp. 395–400.

[53] S. Swaminathan, C. Shi, Y. Jansen, P. Dragicevic, L. A. Oehlberg, and J.-D. Fekete, "Supporting the design and fabrication of physical visualizations," in *Proc. ACM SIGCHI*, 2014, pp. 3845–3854.

[54] S. Mueller, T. Mohr, K. Guenther, J. Frohnhofen, and P. Baudisch, "fabrickation: Fast 3d printing of functional objects by integrating construction kit building blocks," in *Proc. ACM SIGCHI*, 2014, pp. 3827–3834.

[55] S. Mueller, S. Im, S. Gurevich, A. Teibrich, L. Pfisterer, F. Guimbretière, and P. Baudisch, "WirePrint: 3D Printed Previews for Fast Prototyping," in *Proc. ACM UIST*, 2014, pp. 273–280.

[56] J. McCrae, N. Umetani, and K. Singh, "Flatfitfab: Interactive modeling with planar sections," in *Proc. ACM UIST*, New York, NY, USA, 2014, pp. 13–22.

[57] O. Stava, J. Vanek, B. Benes, N. Carr, and R. Méch, "Stress Relief: Improving Structural Strength of 3D Printable Objects," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 48:1–48:11, Jul. 2012.

[58] R. C. Hibbeler, *Mechanics of Materials*, 9th ed. London, UK: Pearson PLC, 2014.

Patrick Paczkowski is currently Vice President of Software at IsoPlexis. He earned his Master's in Computer Science from Yale University, and received his Ph.D. in Computer Science in 2017. Prior to his time at Yale, he was an undergraduate at Duke University, where he double-majored in Computer Science and Mathematics. He is an avid software developer, researcher and entrepreneur, with particular interests in gesture-based 3D modeling, image processing techniques, and UI design and visualization.

Julie Dorsey is a professor of Computer Science at Yale University, where she teaches computer graphics. She came to Yale in 2002 from MIT, where she held tenured appointments in both the Department of Electrical Engineering and Computer Science (EECS) and the School of Architecture. She received undergraduate degrees in architecture and graduate degrees in computer science from Cornell University. Her research interests include photorealistic image synthesis, material and texture models, and sketch-based modeling. Her current and recent professional activities include service as the Editor-and-Chief of ACM Transactions on Graphics (2012-15) and membership on the editorial boards of Foundations and Trends in Computer Graphics and Vision, Computers and Graphics, and IEEE Transactions on Visualization and Computer Graphics. She has received several professional awards, including MITs Edgerton Faculty Achievement Award, a National Science Foundation Career Award, an Alfred P. Sloan Foundation Research Fellowship, along with fellowships from the Whitney Humanities Center at Yale and the Radcliffe Institute at Harvard. She is co-author of Digital Modeling of Material Appearance and the founder of Mental Canvas, a software company that is developing a new type of interactive graphical media and a system to design this form of media.

Holly Rushmeier is a professor of Computer Science at Yale University. She received the PhD degree from Cornell University. She is a fellow of the ACM and of Eurographics. Her research interests include shape and appearance capture, applications of perception in computer graphics, modeling material appearance and developing computational tools for cultural heritage.

Min H. Kim is an associate professor of computer science at KAIST, leading the Visual Computing Laboratory. Prior to KAIST, he worked as a postdoctoral researcher at Yale University. He received his PhD in computer science from University College London in 2010 with a focus on color reproduction in computer graphics. In addition to serving on many conference program committees, such as SIGGRAPH Asia and Pacific Graphics, he has been working as an associate editor in various journals: ACM Transactions on Graphics, ACM Transactions on Applied Perception and Elsevier Computers and Graphics. His research interests include computational imaging, computational photography, 3D imaging, and hyperspectral imaging, in addition to color and visual perception.