Making Burr Puzzles from 3D Models

Shiqing Xin Chi-Fu Lai Chi-Wing Fu
Nanyang Technological University

Tien-Tsin Wong
Chinese University of Hong Kong

Ying He
Nanyang Technological University

Daniel Cohen-Or
Tel Aviv University

Figure 1: *Left: A burr puzzle made from* BIMBA *and right: the eleven puzzle pieces after disassembly.*

Abstract

A 3D burr puzzle is a 3D model that consists of interlocking pieces with a single-key property. That is, when the puzzle is assembled, all the pieces are notched except one single key component which remains mobile. The intriguing property of the assembled burr puzzle is that it is stable, perfectly interlocked, without glue or screws, etc. Moreover, a burr puzzle consisting of a small number of pieces is still rather difficult to solve since the assembly must follow certain orders while the combinatorial complexity of the puzzle's piece arrangements is extremely high.

In this paper, we generalize the 6-piece orthogonal burr puzzle (a knot) to design and model burr puzzles from 3D models. Given a 3D input model, we first interactively embed a network of knots into the 3D shape. Our method automatically optimizes and arranges the orientation of each knot, and modifies pieces of adjacent knots with an appropriate connection type. Then, following the geometry of the embedded pieces, the entire 3D model is partitioned by splitting the solid while respecting the assembly motion of embedded pieces. The main technical challenge is to enforce the single-key property and ensure the assembly/disassembly remains feasible, as the puzzle pieces in a network of knots are highly interlocked. Lastly, we also present an automated approach to generate the visualizations of the puzzle assembly process.

Keywords: Recreational graphics, 3D burr puzzle

1 Introduction

Puzzles have always been fascinating, intriguing and entertaining adults and kids. Naturally, several computational methods have been developed for solving or generating puzzles [Freeman and Garder 1964; Goldberg et al. 2002; Kong and Kimia 2001; Cho et al. 2010]. In this work, we are interested in the making of burr puzzles, which are particularly attractive, complex, and highly challenging to solve (Figure 1). A burr puzzle is a 3D model that consists of interlocking components with a single-key property [Cutler 1978; Cutler 1994; IBM Research 1997]. That is, when the puzzle is assembled, all its parts are notched except one single key component which remains mobile. Unlike conventional puzzle games such as jigsaw puzzles, where the challenge mainly arises from the quantity of puzzle pieces, a burr puzzle attains a very high difficulty index with only a small number of puzzle pieces. Such difficulty index relates to the combinatorial complexity in the puzzle piece arrangement and assembling order.

The burr puzzle pieces have specially-designed geometric structures which yield the unique characteristic of being *interlocking*: once a burr puzzle is assembled by slipping in the last puzzle piece, no other pieces can be taken out unless we first move the last piece, which is called the key. Since the key piece locks the entire 3D model, the whole geometric structure of the 3D puzzle can remain stable without glue, screw, and nail, but at the same time, we can still disassemble and then re-assemble it like common puzzles.

In this paper, we take a computational approach to generate burr puzzles from a given 3D geometric model, in contrast to the traditional burr puzzles that are mainly cuboid in shape (Figure 2(a)). The result is a partition of the 3D shape into perfectly-interlocking puzzle pieces (Figure 1) that can be disassembled with a single

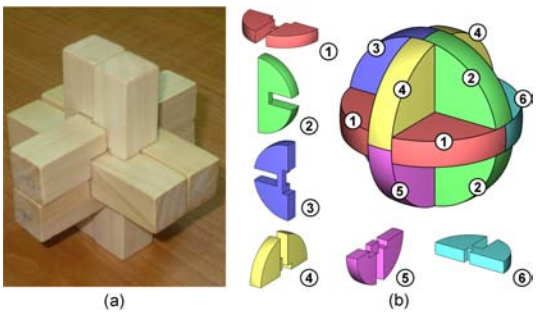

Figure 2: *(a) A traditional cuboid burr puzzle; and (b) the canonical six-piece burr puzzle.*

ACM Reference Format
Xin, S., Lai, C., Fu, C., Wong, T., He, Y., Cohen-Or, D. 2011. Making Burr Puzzles from 3D Models. *ACM Trans. Graph. 30*, 4, Article 97 (July 2011), 8 pages. DOI = 10.1145/1964921.1964992 http://doi.acm.org/10.1145/1964921.1964992.

moveable key piece. Our basic building block is a single canonical six-piece burr puzzle, in which we call it a *knot* for simplicity (Figure 2(b)). We illustrate our method by first generating a single-knot puzzle (6 puzzle pieces only) from the 3D model. Then we extend it to the more complicated multi-knot puzzle (a network of knots) that contains a larger number of puzzle pieces.

Note that extending to the multi-knot case does not naturally preserve the single-key property of the canonical single-knot case. Ensuring such property is a highly challenging issue since there might be a large number of puzzle pieces in the geometric construction, and yet the puzzle pieces have to be fully interlocked with suitable connection types (interdependency). Moreover, the assembly/disassembly of burr puzzles requires each piece to have a non-blocking motion space so that they can be assembled/disassembled in a specific trajectory. Thus, puzzle pieces have to be carefully generated to avoid blocking the trajectory of some other puzzle pieces. To help visualizing the assembly/disassembly of our generated burr puzzle, we further develop visualization methods to automatically illustrate and annotate the puzzle assembly process.

2 Related Work

Burr Puzzle. Burr puzzles have been massively produced in Asia as early as in the 18th century, but there is no consensus as to the origin of the burr puzzle. The IBM research website [IBM Research 1997] dated burr puzzles as early as 1803 in the Bestelmeier Toy catalogs while Coffin [2007] quoted Slocum's work, which traced its history to at least 1698 in Germany. The six-piece burr puzzle can be constructed by arranging three pairs of mutually-perpendicular rods, notching each other in a central region (Figure 2(a)). There are only very limited number of burr puzzle variations for a long time until 1978 when Cutler [1978; 1994] employed computers to systematically and thoroughly analyze all possible combinations of six-piece burr puzzles. Cutler is an American mathematician who worked on analyzing burr puzzles. In addition to six-piece burrs, he also worked on various kinds of burr puzzles, such as rectilinear burrs, non-rectilinear burrs, and box-filling burrs, see [Cutler 2000].

Puzzles in Graphics. Graphics researchers proposed techniques for *recreational assembly*, including the construction of papercraft toy models by Mitani and Suzuki [2004], the plush toys by Mori and Igarashi [2007], the digital bas-relief models by Weyrich et al. [2007], the paper-folding models by Kilian et al. [2008], the polyomino 3D puzzle models by Lo et al. [2009], the shadow-art models by Mitra and Pauly [2009], and the paper-popup architectural models by Li et al. [2010].

Since 60's, computational methods have been developed for solving puzzles, in particular 2D jigsaw puzzles. Freeman and Garder [1964] were the first to approach this problem by considering the geometry of puzzle pieces. Wolfson et al. [1988] introduced a two-stage puzzle assembly algorithm. Goldberg et al. [2002] further developed techniques to improve the performance and robustness. Rather than apictorial, Cho et al. [2010] recently developed a probabilistic approach for solving 2D image jigsaw puzzles. While previous works mostly focus on solving 2D jigsaw puzzles, our method practically creates interlocking 3D burr puzzles from general 3D shapes. Without the computer aid, such interlocking 3D puzzles are extremely difficult to design, and can only be developed by highly skilled craftsmen.

3 Overview

The problem of making a burr puzzle from a 3D model is a volume partitioning problem. The given 3D model is split into disjoint components that can be assembled to form an interlocking burr puzzle. Our solution is based on extending the pieces of a known canonical burr puzzle into larger pieces that conform to the shape of the input model without violating their interlocking properties. To simplify the description of the method, we first start with a basic case of making a six-piece burr puzzle, or a single-knot burr puzzle, and then extend our solution to the general multi-knot burr puzzle.

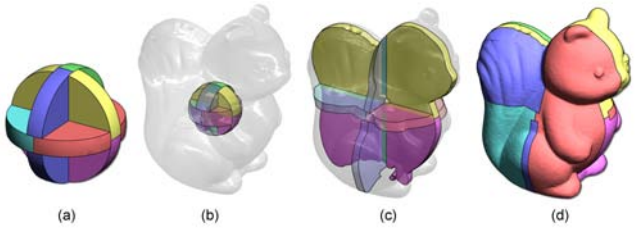

Figure 3: *(a) Single knot: the canonical six-piece burr; (b) embedding the single knot into the 3D model,* SQUIRREL*; (c) extending the "blade" to partition the volume without touching the center burr lock; and (d) pieces after flesh attachment.*

To generate a single-knot burr puzzle, we embed a canonical six-piece burr (knot) in the input 3D model. The method is illustrated in Figure 3. The knot in Figure 3(a) is first partitioned into inner and outer parts. The inner part, which remains a canonical six-piece, is placed entirely inside the 3D model (Figure 3(b)). Then the outer pieces are extruded using an anisotropic (axial) scaling till they go beyond the 3D model. Then we can apply a CSG intersection between the extruded six pieces and the given 3D model to produce the puzzle "skeleton" as shown in Figure 3(c). The last step is to compute the missing octant volumes ("fleshes") and attach them to the "skeleton" pieces to yield the single-knot burr puzzle shown in Figure 3(d) (Section 4).

The general case of making a multi-knot burr puzzle is based on the basic case. The goal is to connect multiple knots into a network so that the resultant puzzle owns the single-key property and all puzzle pieces are interlocked but still can be disassembled. The network is in general an orthogonal grid structure with each knot corresponding to a unique grid point. To work out the connection, we first compute the shortest path tree of the network to obtain a partial order of the knots. Then, we propose novel strategies to orient neighboring knots and modify adjacent puzzle pieces to connect them, aiming at enforcing the single-key and disassemblability properties. Finally, we compute the disassembly order of the pieces, and generate the geometry of each piece with CSG boolean operations (Section 5).

To help understanding how the puzzle can be disassembled/assembled or preparing an "instruction manual," we develop a series of visualization methods in the spirit of [Mitra et al. 2010] to *automatically* generate annotated disassembly/assembly animation for each created puzzle (Section 6). In Section 7, we present our results of various burr puzzles made from different 3D models, and the rapid-prototyped burr puzzles. Finally, Section 8 draws the conclusion.

4 Single-Knot Burr Puzzle

As mentioned before, we employ the canonical six-piece burr puzzle (single-knot) (Figure 2(b)) as the basic building block for our burr puzzle construction. Then we extend it to the general multi-knot puzzle in Section 5.

Locking Mechanics. Before we go on, we first describe the locking mechanics of a knot. Without loss of generality, we denote the six puzzle pieces of a single knot as $\mathbf{A}_1$, $\mathbf{A}_2$, ..., and $\mathbf{A}_6$ (for knot ($\mathbf{A}$)) (Figure 2(b)), and consistently color-code each of them in order to facilitate discussion throughout this paper. The disassembly of a single knot must start with the only moveable key $\mathbf{A}_1$. $\mathbf{A}_1$ can

only slide a little bit outward as it is later blocked by $\mathbf{A}_4$ and $\mathbf{A}_5$. After such little move, the burr lock becomes *partially unlatched* (namely the *activated* state), and we can then take out $\mathbf{A}_2$ or $\mathbf{A}_3$ entirely. The selection of a removable piece at each step may affect the disassembling ordering (path) of the remaining pieces. In general, as one more piece is removed, the choice of further removable pieces increases. Figure 4 shows all possible disassembling sequences (paths) in the form of a graph. Note that the notation $\mathbf{A}$ is dropped for clarity in the graph and the *unlocked* state means that all remaining pieces can be taken out in any order. A video sequence of assembling a knot is shown in the supplementary video.

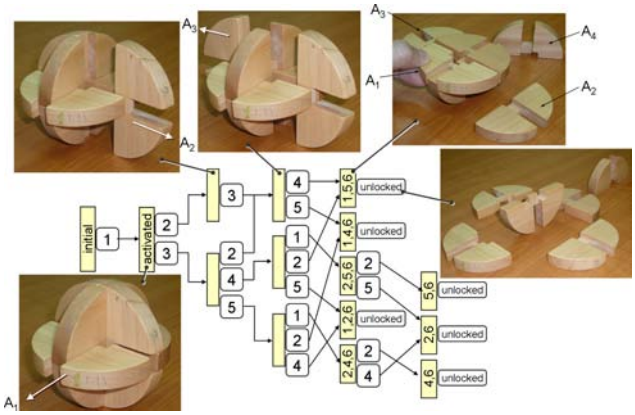

Figure 4: *Possible disassembling orders: the vertical rectangular boxes and rounded squares next to each of them denote the states and the removable puzzle piece(s) at each state, respectively.*

Knot Embedding. To create burr puzzles from a given 3D shape, we can simply embed the knot into the shape (Figure 3(b)) and extend the "blades" of burr pieces along the three principle axes to partition the shape (Figure 3(c)). By this, we regard the problem of making a burr puzzle from a 3D model as a volume partitioning problem. The extension of "blades" can be achieved by applying an anisotropic (axial) scaling only on the blades without touching the center notch of the burr lock. Note that such extension does not violate the interlocking property since each piece is extended only radially from the knot center. After that, the six burr pieces are formed by CSG intersection with the given 3D shape (Figure 3(c)).

Flesh Attachment. So far, we only form the "skeleton" of the burr puzzle. There are eight missing octant volumes ("fleshes") as in Figure 3(c), and these fleshes have to be attached to the six extended puzzle pieces to complete the final puzzle. There can be many ways to attach the fleshes. The key criteria we used is to make the cutting plane of the flesh radial from the knot center to avoid potential occlusion in the assembling/disassembling trajectory of the puzzle pieces.

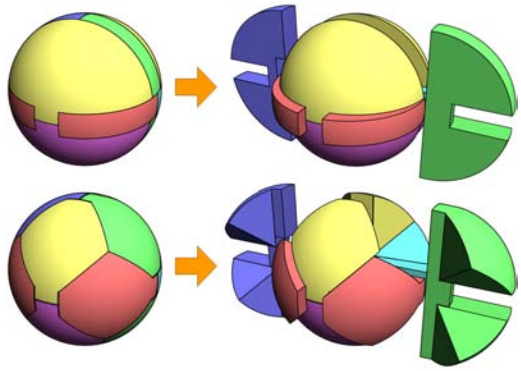

Figure 5: *Top: Two-way attachment; Bottom: Even attachment by three 45-degree cutting planes.*

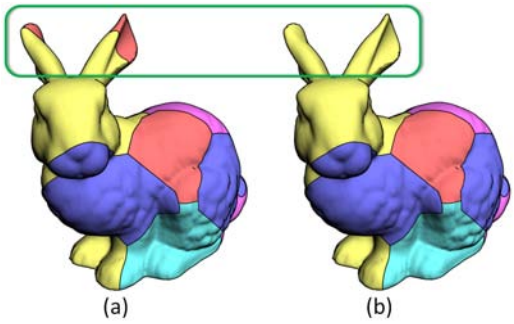

Figure 6: *Fragments may be resulted during partitioning.*

Here, we suggest three kinds of attachment schemes, namely *two-way*, *multi-way*, and *even* attachments. The two-way scheme attaches the eight fleshes to two out of the six pieces on the opposite side, e.g., $\mathbf{A}_1$ and $\mathbf{A}_6$, $\mathbf{A}_2$ and $\mathbf{A}_3$, or $\mathbf{A}_4$ and $\mathbf{A}_5$ (Figure 5 (top)). The multi-way scheme generalizes the two-way one by arbitrarily attaching a flesh to one of its three neighboring puzzle pieces, resulting in 3^8 possible choices altogether. The even attachment scheme strives for a more balanced partitioning by first partitioning each flesh into three subvolumes and then attaching each subvolume to the neighboring puzzle piece (Figure 5 (bottom)). The choice of flesh attachment schemes is decided by the users. In practice, we couple the flesh attachment step with the skeleton construction step, by subtracting unwanted subvolumes from the given object using CSG operations.

Note that a puzzle piece may result in multiple disjoint parts during the partitioning in knot embedding and/or flesh attachment (Figure 6). We have to constrain the partitioning process to a single connected component to avoid fragmentation during the CSG operations.

5 Multi-Knot Burr Puzzle

Extending the burr puzzle from single-knot to multi-knot results in more complicated and challenging 3D puzzles with a larger number of interlocking puzzle pieces. To do so, we need a series of procedures: designing the knot network, computing the knot interdependency and orientation to maintain the single-key property, modifying certain pieces to connect neighboring knots, and finally generating the puzzle pieces.

Knot Network Design. Given an input 3D object, our first step is to design a network of knots that fits inside the volumetric space of the object (Figures 7(a)-(b)). We employ an interactive GUI to load the 3D model, position the knots inside the volume, and connect them into an undirected network. Since the 6 pieces of a knot are axis-aligned (orthogonal) in nature, edges in the knot network are also axis-aligned. Therefore, the placement of neighboring knots is constrained to be axis-aligned. During the knot network design, the system renders partition planes, i.e., the mid-plane between neighboring knots (Figure 7(c)). This allows us to visualize the partitioned sub-volumes corresponding to each knot, thereby avoiding uneven subdivision of the object volume among the knots. The last input in this step is the specification of the *key knot*. That is the knot containing the key piece of the entire puzzle (Figure 7(b)).

Note that, when designing the knot network, edges are undirected (the knot dependency is not yet decided) and the knot orientation is not yet determined. Our methods to be described later can automatically compute them.

Knot Orientation. Due to its axis-aligned nature, the knot network is a subgraph of a complete 3D grid graph. For any two neighboring grid points in the graph, there must be an edge connecting them to ensure the graph connectivity. When placing a knot at a grid

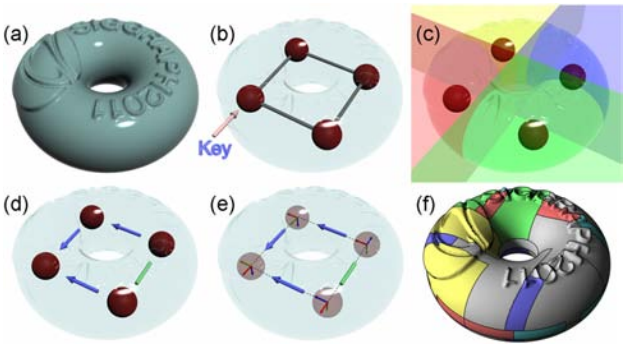

Figure 7: *Constructing a multi-knot burr puzzle: (a) a given 3D model; (b) a network of knots; (c) partition the model volume for each knot; (d) compute the knot interdependency; (e) compute the knot orientation; and (f) generate the puzzle pieces.*

point, we have 48 possible choices of orientations via rotation and mirror-reflection. This is due to the fact that there are 24 different orientations by applying axis-to-axis rotations; in addition, we can obtain another 24 different orientations by applying mirror reflection of them because the burr structure is asymmetrical. Note that mirror reflection of a knot is still a valid burr puzzle. To facilitate the knot connection, we further divide the 48 orientations into two

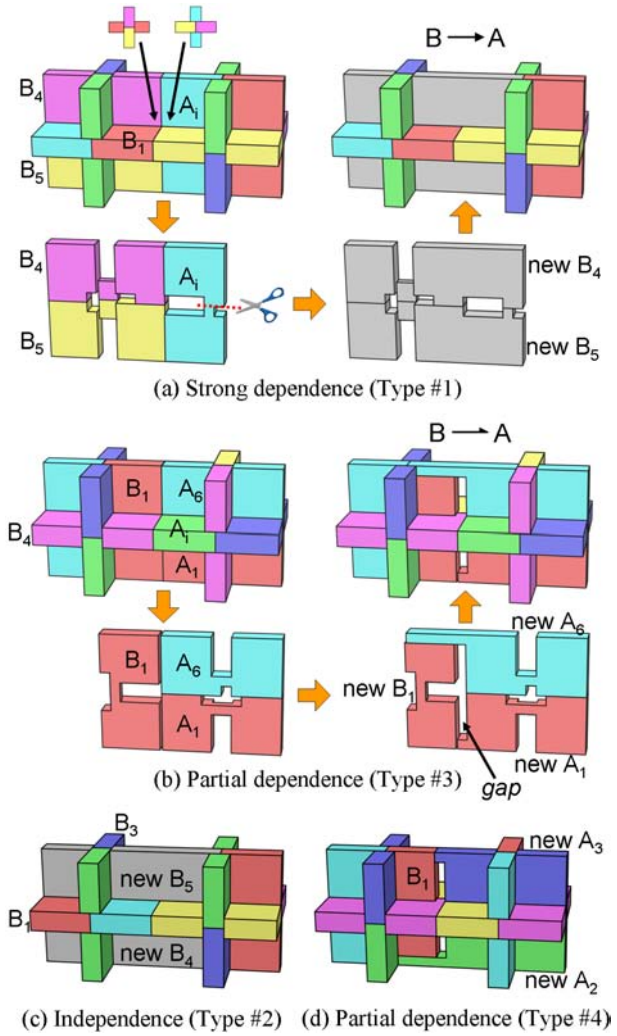

Figure 8: *Connection types of neighboring knots.*

groups using the following group operations: 180°-rotation about X, Y, or Z axis and mirror reflection. This gives us groups G_1 and G_2, each with 24 orientations.

To connect neighboring knots, we found that the orientation of two neighboring knots, say $\mathbf{A}$ and $\mathbf{B}$, must be chosen from different orientation groups. That is, if $\mathbf{A}$ is from G_1 (or G_2), $\mathbf{B}$ must be from G_2 (or G_1). In this way, the proposed constraint can always result in a distinctive contact (90-degree-off) between all neighboring knots, as depicted by the top-left thumbnails (cross-sections) in Figure 8(a). This pattern can support the connection technique described in the next step.

Knot Connection and Dependency. With the above orientation constraint, we can connect neighboring knots and then create interdependency to maintain the single-key property. This requires modification on the puzzle pieces. To facilitate the discussion, we denote a knot as $\mathbf{A}$ and its i-th piece as $\mathbf{A}_i$. We also employ the same subscript numbering and color-coding of pieces as in Figure 2(b).

Connection type #1: Strong dependence. Consider two neighboring knots $\mathbf{A}$ and $\mathbf{B}$. In this type, we orient $\mathbf{B}_1$ to face $\mathbf{A}_i$ where $\mathbf{A}_i$ can be any piece of $\mathbf{A}$ other than $\mathbf{A}_1$. Next, we cut $\mathbf{A}_i$ into two halves as in Figure 8(a) and merge them into $\mathbf{B}_4$ and $\mathbf{B}_5$, respectively. Since $\mathbf{B}_1$ is the key piece of knot $\mathbf{B}$ (Figure 4), by orienting $\mathbf{B}_1$ to face knot $\mathbf{A}$, we block the movement of $\mathbf{B}_1$ and make $\mathbf{B}$ fully dependent on $\mathbf{A}$. That is, $\mathbf{B}$ cannot be unlocked without first unlocking $\mathbf{A}$. We denote such strong dependence by $\mathbf{B} \rightarrow \mathbf{A}$.

Connection type #2: Independence. Two neighboring knots can be connected but not dependent on each other in the assembly/disassembly order. This connection type is called independence. One example is shown by the green edge in Figures 7(d)&(e). Geometrically, it is a variant of Type #1, and achieved by orienting $\mathbf{B}$ so that both $\mathbf{B}_1$ and $\mathbf{B}_3$ face away from $\mathbf{A}$, and then by modifying the puzzle pieces as in Type #1 (the result is shown in Figure 8(c)). With this arrangement, both $\mathbf{A}$ and $\mathbf{B}$ can be unlocked independently, even though they are connected. We denote this connection type as $\mathbf{B} - \mathbf{A}$.

Connection type #3: Partial dependence. Figure 8(b) demonstrates how pieces are modified to construct this type. First, we orient $\mathbf{A}_1$ and $\mathbf{A}_6$ to block $\mathbf{B}_1$. Then, we modify these three pieces as in Figure 8(b) (lower right) so that $\mathbf{B}_1$ is blocked only by $\mathbf{A}_1$. The introduced gap between modified $\mathbf{B}_1$ and $\mathbf{A}_1$ & $\mathbf{A}_6$ is just large enough to allow $\mathbf{B}_1$ to activate $\mathbf{B}$ (the little forward move) after $\mathbf{A}_1$ has moved outwards (downwards in the figure). Recall that a knot is activated by slightly moving outwards its key piece (Figure 4). So when $\mathbf{A}$ is locked, so as $\mathbf{B}$. But when $\mathbf{A}$ is activated, so as $\mathbf{B}$, even though other pieces of $\mathbf{A}$ have not been moved yet. In other words, $\mathbf{B}$ can still be unlocked as long as $\mathbf{A}$ can be activated. This connection type is useful to provide flexibility when the knot network is heavily interlocked. We denote this type as $\mathbf{B} \rightharpoonup \mathbf{A}$.

Connection type #4: Partial dependence. The last connection type is a variant of Type #3, in which we orient $\mathbf{A}_2$ and $\mathbf{A}_3$ instead of $\mathbf{A}_1$ and $\mathbf{A}_6$ to contact $\mathbf{B}_1$ (Figure 8(d)). In this way, $\mathbf{B}$ can be activated only if $\mathbf{A}_3$ moves out of its place. Note that $\mathbf{A}_3$ is in a later stage of the disassembly compared to $\mathbf{A}_1$, thus making it a more secure type of partial dependence as compared to Type #3. We denote it as $\mathbf{B} \rightharpoondown \mathbf{A}$.

Determining Knot Orientations and Connections. To determine the dependency among knots and the orientation of each knot, our method performs the following steps:

Step 1: Compute a shortest path tree. With the user specified key knot $\mathbf{K}$, we apply the Dijkstra's algorithm starting from $\mathbf{K}$ to compute a shortest path tree, $\mathcal{T}$, with $\mathbf{K}$ as the root of $\mathcal{T}$.

Step 2: Candidate knot orientations. Then, we can rotate $\mathbf{K}$ so that its key piece $\mathbf{K}_1$ does not face any neighboring knots, thereby ensuring $\mathbf{K}_1$ to be always movable when the puzzle disassembly starts. Based on the parent-child relationship in $\mathcal{T}$, we can determine a disassembly order, and hence the interdependency constraint, for each pair of neighboring knots. If $\mathbf{A}$ and $\mathbf{B}$ are a pair of parent and child knots in $\mathcal{T}$, we arrange $\mathbf{B}_1$ to face $\mathbf{A}$ so that $\mathbf{B}$ is dependent on $\mathbf{A}$, either fully or partially. Note that, at this moment, the orientation of $\mathbf{B}$ can still have four possible choices (by a 180° rotation along $\mathbf{B}$ to $\mathbf{A}$ and/or mirror reflection about plane of $\mathbf{B}_1$) in an orientation group that is different from that of $\mathbf{A}$.

Step 3: Candidate connection types. In addition to knot orientation, we determine a set of candidate connection types for each pair of neighboring knots. If the pair is associated with an edge in $\mathcal{T}$, we must connect the two knots with Type $\#1$, $\#3$ or $\#4$, so that $\mathbf{B}$ can always depend on $\mathbf{A}$ to maintain the interlocking. On the other hand, if the pair is not associated with any edge in $\mathcal{T}$, we use the independence connection Type $\#2$ to connect them so that the resultant puzzle can be more strongly connected.

Step 4: Determining orientation and connection. With the candidate orientations and connection types over the knot network, we can simulate the disassembly starting from knot $\mathbf{K}$ with its key piece. The goal is to determine the orientation and connection for all knots so that all pieces can be disassembled in order during the simulation. Note that we can follow the logics in Figure 4 to compute the valid puzzle piece moves locally in each knot together with the logics previously defined for the four connection types.

In our current implementation, we use a greedy algorithm to make choices during the simulation. That is, we attempt to arrange $\mathbf{B}_3$ and also $\mathbf{B}_5$ to face away from neighboring knots of $\mathbf{B}$ when choosing the knot orientation of $\mathbf{B}$ because $\mathbf{B}_3$ is a crucial milestone piece in the disassembly (Figure 4). In addition, we prefer Type $\#1$ over $\#3$ and $\#4$ as the dependence connection because it has the strongest dependency among the three.

Generating Puzzle Pieces. After determining the knot orientations and connections, we can generate the "skeleton" of multi-knot burr puzzles as in the single-knot case. At the same time, we can also modify the puzzle pieces according to the determined connection type. Then, starting from the key piece, we can attach the "flesh" into the skeleton puzzle pieces by computing the motion space of each puzzle piece while ensuring a non-blocking trajectory. Our implementation employs the CGAL library API to generate an action list, and performs the CSG boolean operations in 3D Studio Max via scripting. An action list encodes a sequence of CSG operations for constructing the geometry of each puzzle piece. Each action in the list is basically a CSG operation such as $A(\cap/\cup/-)B = C$, where A, B, and C are mesh models.

Extensions. Our multi-knot puzzle modeling framework can be further extended in the following two ways:

Non-Orthogonal Connection. So far, neighboring knots are arranged exactly axis-aligned. We can generalize orthogonal knot connection to be non-orthogonal by shearing the volumetric space in-between neighboring knots (Figure 9 (left)). Since such shearing does not block the motion trajectory of puzzle pieces in the assembly, we can retain the interlocking and connection structure among the puzzle pieces. To construct the non-orthogonal connection as shown in Figure 9 (left), we first arrange three knots in the DRAGON body in zigzag way (with 55° turning angle). Then, we can apply a piecewise shearing along one specific direction (while isometric along the others) to deform the subspace in the puzzle skeleton structure. Finally, we can continue with the standard CSG operations in our multi-knot framework to generate the puzzle pieces.

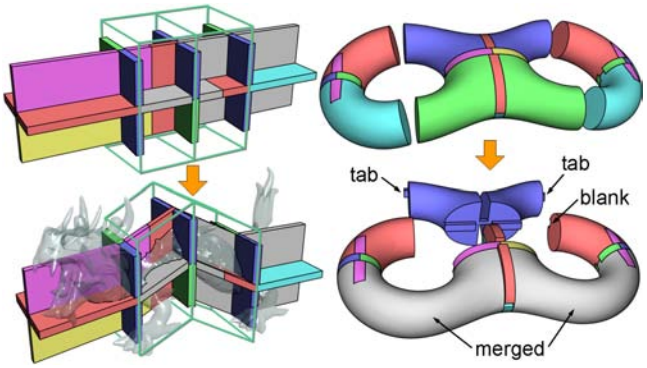

Figure 9: *Non-orthogonal connection (left) and connecting disjoint knot networks (right).*

Disjoint Knot Networks. Another interesting variation is that we can produce disjoint knot networks within a 3D model, and yet we can connect them by merging puzzle pieces among them. As shown in Figure 9 (right), we in fact start with three disjoint knots here. By merging relevant contacting pieces (those become grey) between the middle and left knots (as well as the middle and right knots), we can connect these three disjoint knots. However, it is worth to note that such connection does not enforce the single-key property. In this example, we can add in two pairs of tabs and blanks on the blue and the two red pieces (left and right) as indicated in the figure. The tabs on the blue piece can block the motion volume of the two red pieces, which are the keys of the left and right knots. As a result, these two knots become dependent on the blue piece of the middle knot, hence leaving a single key in the entire system. The general rule is that we have to pick one of the knot networks as the key, and block the key pieces of all the other disjoint knot networks by pieces subsequently dependent on the key network.

Please refer to the supplementary video for animations to present these results.

6 Assembly/Disassembly Illustration

Assembling burr puzzles can be very challenging, particularly for multi-knot burr puzzles. To aid the understanding on the puzzle assembly/disassembly process, we develop illustrative visualization methods to automatically generate animations and figures to illustrate and annotate the assembly/disassembly process. Related literatures on illustration generation include: Mitra et al. [2010] used cap, side and translational arrows to illustrate the spatial movement of mechanical parts. Agrawala et al. [2003] used motion arrows as a guidance. Our method follows the principle of these works to illustrate the spatial configuration and object motion of our puzzle pieces. In particular, simultaneous movement of multiple puzzle pieces is also possible in our illustration.

Camera Movement. To ensure the visibility of puzzle piece in-motion, we change the camera view on the puzzle pieces. Given an initial camera position and the puzzle assembly order, we compute the visibility of each puzzle piece in-motion by rendering it together with all the other pieces in the camera view. If the proportion of its occlusion in the view is higher than a threshold ($\tau_o = 50\%$), we move our camera to a new orientation such that the camera viewing direction and the piece movement direction is within a prescribed threshold ($\tau_a = 60$ degrees). The upward vector of the camera frame remains constant to minimize the annoying tilting. For the case of multi-knot puzzles, we also automatically translate the camera to center our view on the next knot to be assembled or disassembled.

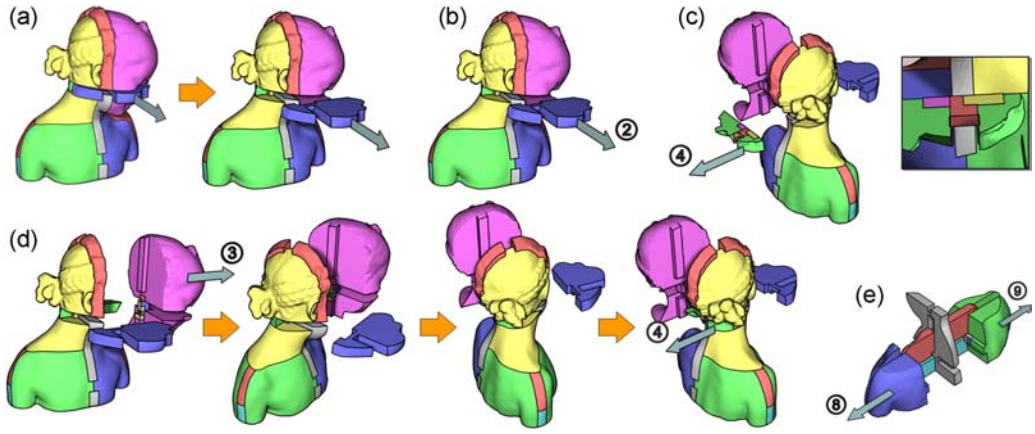

Figure 10: *Illustrating the puzzle disassembly: (a) motion arrow placement to indicate puzzle piece movement; (b) number label to show the movement stage; (c) contact areas coding with zoom-in view to illustrate the interlocking structure; (d) successive views showing the camera movement to expose the next puzzle piece in-motion; and (e) annotating multiple puzzle pieces in motion.*

Motion Arrows and Numbering. Motion arrows have been used in many conventional composition diagrams in instruction manuals to indicate how parts should be assembled. With our previous puzzle formation result, we can determine the assembly/disassembly sequence, and thus, produce the visualization animation. Our method first analyses the puzzle piece movement during the assembly/disassembly, and then automatically places the motion arrows for illustration. To place a motion arrow on the moving piece, we compute the centroid of the largest visible region occupied by this piece in image space, and then project this point onto the puzzle piece surface in object space. The arrow tail is then anchored at this 3D point and its head is oriented towards the moving direction of the puzzle piece. The arrow is further translated towards the camera in order to avoid intersection with the puzzle piece (Figure 10(a)) and scaled properly relative to the size of the related puzzle piece.

Numbering is another common feature in many conventional illustrations. We automatically generate number labels to show the assembly/disassembly steps. We place the number labels next to the arrow head (Figure 10(b)).

NPR Stylization and Color-Coding. To better convey the geometric structure, we employ non-photorealistic rendering style similar to [Li et al. 2008]. We color-code the canonical six pieces with RGBYMC colors, and for multi-knot puzzles, we highlight those merged pieces (modified after knot connection) with a light grey color, see Figures 10(a)-(e).

Furthermore, to help visualizing how burr pieces touch each other, we color-code the contact area at the innermost burr lock region. The color applied to each contact area is the color of the contacting piece (Figure 10(c)). Such visualization is applied only to the puzzle piece in motion, and we display also a zoom window to expose this contact area information.

7 Results

We applied our computer-aided-geometric design system to make a number of burr puzzles from a variety of 3D models using different knot network arrangements. Figure 12 shows all the 3D puzzles created from our modeling system (see the mini-map indices for the name of the puzzles). Regardless of the knot arrangements and the number of puzzle pieces contained, all the presented puzzle models can enforce the single-key property, and thus can be perfectly-interlocking upon the puzzle assembly. Note also that the small gap in the middle of CUBICBOX is a result of Type #3 connection; such a gap is introduced to leave a movement space for taking out the related key piece. Figure 11 depicts the corresponding knot network arrangement for each of these puzzles, using blue arrow, green edge, yellow arrow, and magenta arrow to indicate Type #1, #2, #3, and #4 connections, respectively; in addition, the key knots are colored in orange. Note that since knots connected by Type #2 are independent of each other, thereby no arrow heads are drawn. In addition, 2-TORUS is created by connecting disjoint knot networks. Its knots are not connected using any of the standard connection types.

Table 1 provides the detail of the 3D puzzles in three sections. The first section refers to the 3D puzzles created using the extended techniques: DRAGON and 2-TORUS. The second section refers to those produced with a rapid-prototyping counterpart, whereas the last section is for the rest. The last column in the table shows the time taken for our system to compute the knot orientations and connections, as well as to generate the puzzle pieces. These experimental results also demonstrate the possibility of using very different arrangements when designing the knot network.

Rapid-prototyping Results. Figure 13 shows the three rapid-prototyping burr puzzles we have. They are created from 3D plastic parts printers (SOMOS and SLA machines) with different slot widths, i.e., the size of the slot in the inner part of the burr lock. The slot widths for CHESS, MOAI, and TORUS are 2mm, 2.5mm, and 3mm, respectively, and such value is controllable in the puzzle piece generation.

The CHESS model is particularly interesting because it demon-

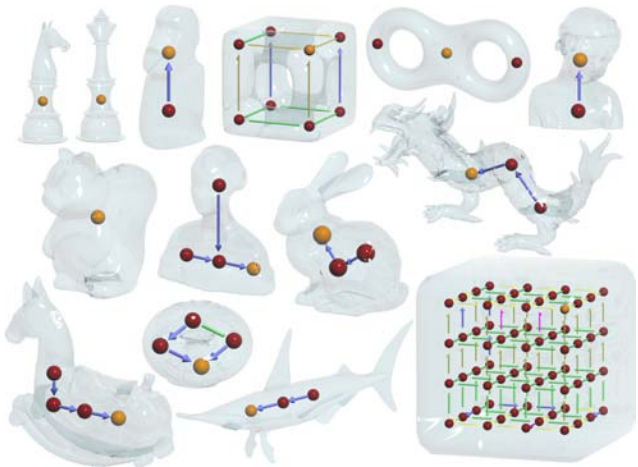

Figure 11: *Knot networks employed to construct the 3D burr puzzles presented in Figure 12.*

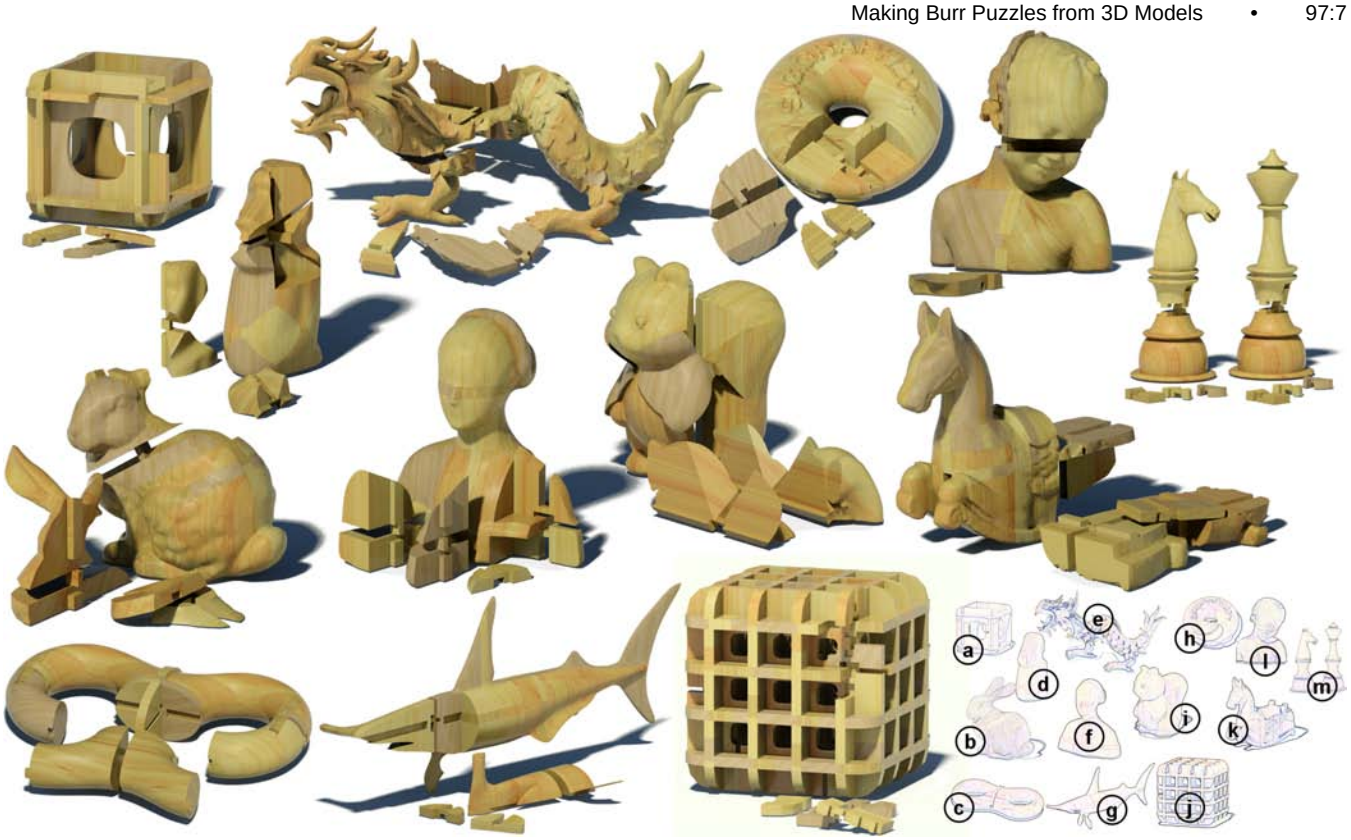

Figure 12: *Single-knot and multi-knot burr puzzles created from assorted 3D models: a)* CUBICBOX*, b)* BUNNY*, c)* 2-TORUS*, d)* MOAI*, e)* DRAGON*, f)* LAURANA*, g)* SHARK*, h)* TORUS*, i)* SQUIRREL*, j)* MEGABOX*, k)* ISIDORE HORSE*, l)* BIMBA*, and m)* CHESS*.*

Table 1: *Information about the burr puzzles in Figure 12.*

Puzzle model	#knots	#pieces	Timing (sec.)
DRAGON	3 (non-ortho.)	16	175
2-TORUS	3 (disjoint knots)	16	95
CHESS	1 (interchangeable)	7	107
MOAI	2 (evenly attachment)	11	141
TORUS	4 (2×2)	20	113
SQUIRREL	1 (single-knot)	6	73
BIMBA	2 (linear)	11	92
BUNNY	3 (L-shaped)	16	163
SHARK	3 (linear)	16	151
ISIDORE HORSE	4 (L-shaped)	21	187
LAURANA	4 (T-shaped)	21	171
CUBICBOX	8 ($2 \times 2 \times 2$)	36	196
MEGABOX	64 ($4 \times 4 \times 4$)	240	1405

strates a unique characteristic of using the burr structure to connect and compose 3D models: "*interchangeable*." This model consists of seven puzzle pieces with two of them originated from exactly the same skeleton puzzle piece out of the six pieces in the canonical set. The headpieces for the knight and bishop are interchangeable, see again Figure 13 (right). In addition, it is worth highlighting that the rapid-prototyped TORUS puzzle is a 4-knot model with 20 pieces; these pieces are perfectly interlocking with only a single key.

Implementation Issues. The automatic visualization engine was implemented by using offscreen rendering with OpenGL to obtain the depth buffer view at different time frame. Hence, we can obtain scene conditions at the viewpoint and automatically generate an animation script for 3D Studio Max to position and move the camera, to arrange the motion arrows (through scripting in 3D Studio Max), as well as to layout other illustrative elements shown in the assembly/disassembly video, see also Figure 14 for the assembly sequence of the BUNNY puzzle. Compared to the case of disassembly, these motion arrows are reversed. In addition, we also developed a lightweight program to help preview the puzzle disassembly/assembly sequence in a generated animation script; see the supplementary material for this lightweight program and the simplified version of the animation scripts.

Limitations. We cannot create burr puzzles from 3D shapes with parts that are too narrow or flat, and the key must be on the outer side of the knot network. Although in Section 5 we demonstrate how to extend the connection strategies to more complicated geometric shapes by attempting to relax the requirement of orthogonal structure, we must point out that such an extension technique is not general. The essential limitation of the orthogonal structure, in our mind, lies in the intrinsic structure of burr. Moreover, there are also no general methodologies that can always ensure that a single key locking a disjoint knot network.

Figure 13: *Rapid prototyping models of our 3D burr puzzles.*

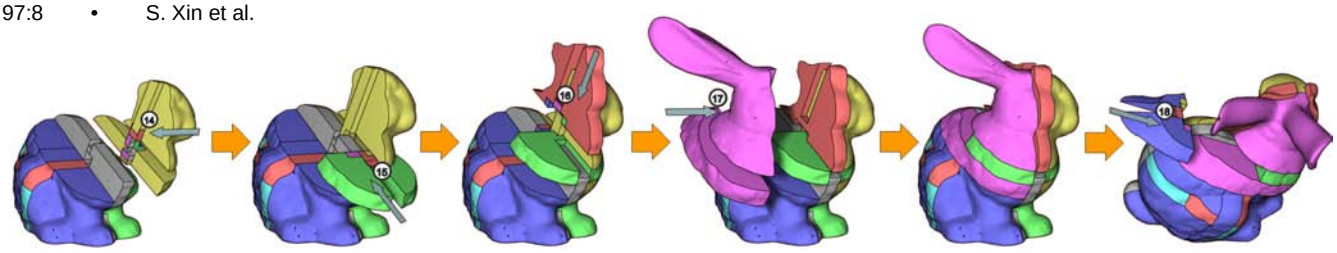

Figure 14: *An assembly sequence of the* BUNNY *puzzle.*

More generally, a burr puzzle should better be made up of pieces that are roughly equal in size, similar in shape, as well as solid and strong. In addition, input 3D models that are more symmetric are likely to produce more balanced puzzle pieces. However, these observations are also not absolute. Building burr puzzles from asymmetrical models may also be nicely shaped, for example, for artistic purpose.

8 Conclusion

This paper generalizes the well-known 6-piece orthogonal burr puzzle to multi-knot burr puzzles and introduces a geometric modeling system to design and create burr puzzles from 3D models. Our geometric modeling techniques can enforce the single-key property while connecting and interlocking the puzzle pieces partitioned from the 3D models, meaning that a 3D puzzle model can be locked by a single key piece regardless of the number of puzzle pieces it consists of. Hence, the resultant 3D puzzles can be glue-less, screw-less, and nail-less, as well as interchangeable, thus allowing us to replace or reconfigure parts in the puzzle.

Key techniques proposed include the formulation of knot network to model multi-knot burr puzzles, geometric modification methods to connect knots, the computational method for knot orientation and connection, and the volumetric partitioning schemes to shape the puzzle pieces from "skeleton" and "flesh," as well as the visualization methods to illustrate the puzzle assembly or disassembly process. All these enable us to make the 3D puzzle models perfectly-interlocking with only a single key, as demonstrated in a variety of examples we presented.

Acknowledgements. We thank the anonymous reviewers for their constructive comments, and Shuang-Min Chen for her help in preparing some of the figures and video demonstration. This project was partially supported by NRF2008IDM-IDM-004-006, AcRF Tier1 69/07, AcRF Tier1 RG 13/08, Hong Kong RGC General Research Funds (Project No. CUHK417107), and the Israel Science Foundation. For the data models used: BUNNY and DRAGON are courtesy of the Stanford University Computer Graphics Laboratory; CHESS is produced by our technician Max Lim; the following models are provided by the AIM@SHAPE Shape Repository: BIMBA, ISIDORE HORSE, and SHARK courtesy of INRIA; and LAURANA courtesy of the Visual Computing Lab of ISTI-CNR.

References

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

CHO, T. S., AVIDAN, S., AND FREEMAN, W. T. 2010. A probabilistic image jigsaw puzzle solver. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2010*, 183 –190.

COFFIN, S. T. 2007. *Geometric Puzzle Design*. A. K. Peters.

CUTLER, W. H. 1978. The six-piece burr. *Journal of Recreational Mathematics 10*, 4, 241–250.

CUTLER, W. H., 1994. A computer analysis of all 6-piece burrs. Self published.

CUTLER, W. H., 2000. Bill Cutler Puzzles, Inc. puzzle works. `http://home.comcast.net/~billcutler/index.html`.

FREEMAN, H., AND GARDER, L. 1964. Apictorial jigsaw puzzles: The computer solution of a problem in pattern recognition. *IEEE Transactions on Electronic Computers EC-13*, 2, 118 –127.

GOLDBERG, D., MALON, C., AND BERN, M. 2002. A global approach to automatic solution of jigsaw puzzles. In *Proceedings of the Eighteenth Annual Symposium on Computational Geometry*, ACM, New York, NY, USA, 82–87.

IBM RESEARCH, 1997. The burr puzzles site. `http://www.research.ibm.com/BurrPuzzles/`.

KILIAN, M., FLÖERY, S., CHEN, Z., MITRA, N. J., SHEFFER, A., AND POTTMANN, H. 2008. Curved folding. *ACM Tran. on Graphics (Proc. of SIGGRAPH) 27*, 3.

KONG, W., AND KIMIA, B. B. 2001. On solving 2D and 3D puzzles using curve matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2001*.

LI, W., AGRAWALA, M., CURLESS, B., AND SALESIN, D. 2008. Automated generation of interactive 3D exploded view diagrams. *ACM Trans. Graph. 27* (August), 101:1–101:7.

LI, X.-Y., SHEN, C.-H., HUANG, S.-S., JU, T., AND HU, S.-M. 2010. Popup: Automatic paper architectures from 3D models. *ACM Tran. on Graphics (Proc. of SIGGRAPH) 29*, 3. Article 111.

LO, K.-Y., FU, C.-W., AND LI, H. 2009. 3D Polyomino puzzle. *ACM Tran. on Graphics (Proc. of SIGGRAPH Asia) 28*, 5. Article 157.

MITANI, J., AND SUZUKI, H. 2004. Making papercraft toys from meshes using strip-based approximate unfolding. *ACM Tran. on Graphics (Proc. of SIGGRAPH) 23*, 3, 259–263.

MITRA, N. J., AND PAULY, M. 2009. Shadow art. *ACM Tran. on Graphics (Proc. of SIGGRAPH Asia) 28*, 5. Article 156.

MITRA, N. J., YANG, Y.-L., YAN, D.-M., LI, W., AND AGRAWALA, M. 2010. Illustrating how mechanical assemblies work. *ACM Tran. on Graphics (Proc. of SIGGRAPH) 29*, 3. Article 58.

MORI, Y., AND IGARASHI, T. 2007. Plushie: an interactive design system for plush toys. *ACM Tran. on Graphics (Proc. SIGGRAPH) 26*, 3. Article 45.

WEYRICH, T., DENG, J., BARNES, C., RUSINKIEWICZ, S., AND FINKELSTEIN, A. 2007. Digital bas-relief from 3D scenes. *ACM Tran. on Graphics (Proc. of SIGGRAPH) 26*, 3. Article 32.

WOLFSON, H., SCHONBERG, E., KALVIN, A., AND LAMDAN, Y. 1988. Solving jigsaw puzzles by computer. *Annals of Operations Research 12*, 51–64.