Making and animating transformable 3D models

Yi-Jheng Huang, Shu-Yuan Chan, Wen-Chieh Lin, Shan-Yu Chuang

Abstract

Transformable models are 3D models whose shapes can be changed by rotating or translating their component parts. They have a variety of applications in our daily lives, being used in film props and sets, robots, furniture, tools, and toys. Successful transformable models, however, are challenging to create. In this paper, we present a new approach to designing and animating a transformable model, in which a source model is optimally segmented based on a target model and skeleton provided by users, and the motion of transformation is mapped from the source to target models. Our experimental results indicate that our system can transform a 3D model plausibly.

1. Introduction

Transformable models are 3D models whose overall shapes can be changed merely by rotating or translating their component parts. A good example of this is provided by the robots in the movie Transformers, but transformable models also exist in our daily lives, in the form of foldable furniture, tools, and toys whose shapes can be altered to save space or enhance functionality. Nevertheless, the successful creation of a transformable model is not an easy task. Designers have to consider how to shape and arrange each component part so that the model can take on two very different shapes by only translating and rotating its parts. They also need to carefully plan a transformation process, including the transformation order of the component parts and the type of joints between them, to ensure that the parts do not collide with each other and cause damage or even the failure of the entire system.

In this paper, we present a system that can generate a transformable model and its associated transformation motion. Taking two inputs—a source model and a target skeleton, with the latter representing a user’s desired figure—the system first adjusts the target skeleton and embeds it into the source model. Then, the simulated annealing (SA) is utilized to optimally segment the source model into parts. Finally, the transformation of these parts is animated based on the results of a two-level motion-planning process. The experimental results discussed below indicate that our system has considerable promise for assisting novice users to create transformable models easily, as well as providing professionals with feasible options for references.

The contributions of this paper can be summarized as follows: (1) We introduce the transformable-model problem to the fields of modeling and animation. (2) We propose an optimization-based segmentation approach to generate suitable parts for the construction of transformable models. (3) We propose a two-level motion-planning process to animate transformable models plausibly.

2. Related work

Making a transformable model is related to the geometric dissection problem that originated in ancient Greece [1]. Essentially, it is a segmentation problem. This section reviews related work including the dissection puzzles in mathematics and segmentation in computer graphics, as well as research that applies to 3D models in other applications.

Dissection puzzles: A dissection puzzle, also called a transformation puzzle, is a set of pieces that can be rearranged into two or more distinct geometric shapes. An excellent survey of which was provided by Frederickson [2,3], though study of such puzzles was undertaken by ancient Greek mathematicians more than two millennia ago [1]. The ancient Chinese also invented the tangram, a puzzle consisting of seven pieces [4], the objective of which is to use all the pieces to form different specific shapes.

Recently, research on geometric dissection has taken two main directions. The first is a search for optimal dissections that minimize the number of pieces. Cohen [5], for instance, studied economical dissections of a triangle into squares, and Akityama et al. [6] proposed an efficient dissection method for dividing a square into pieces that can be rearranged into two or more smaller squares. The other focus of research has been hinged dissections [3], in which all the pieces are linked at certain points. Piano hinges [7] and twisted hinges [8] have also been studied. However, the above-named studies only solved 2D hinged-dissection problems, whereas the focus of the present work will be on 3D hinged dissections.
With regard to 3D, Zhou and Wang [9] proposed a method for creating geometric dissection puzzles. Taking two regular grid figures of equal area, they partitioned one into separate pieces and reassembled them to form the other figure. Later, Zhou et al. [10] improved upon this work such that the pieces remained linked together while transforming into the alternative shape. Lo et al. [11] introduced a system which generated a polyomino puzzle by parameterizing the input to a quad-based surface then tiling the surface with polyominoes. Interlocking puzzles were discussed in [12,13], in which an input mesh is segmented into pieces that interlock with each other.

The key differences between this paper and the above 3D puzzle studies are as follows. (1) Unlike prior researchers [9–13] whose methods limited the solution space to discrete regular grids, we link parts together and seek solutions in a continuous space. (2) With regard to research purposes, previous studies of interlocking puzzles [11–13] focused on cutting a model into a buildable, interlocking and maintainable puzzle, whereas the goal of our study is the building of a transformable model. (3) Our study takes into account the process of transformation, utilizing a rapidly-exploring random tree (RRT) method to generate transformation animation.

Mesh segmentation: The problem of 3D model segmentation has been extensively studied. Many methods have been proposed, including K-means [14], core extraction [15], primitive fitting [16], randomized cuts [17], and random walks [18]. Chen et al. [19] proposed a set of manually generated benchmarks for evaluating 3D segmentation. In contrast to traditional segmentation algorithms, which mainly consider the segmenting of mesh surfaces, we treat a 3D model as a solid object and cut it into blocks that are also 3D. Moreover, our goal is to generate parts that can roughly represent the figures before and after transformation.

Computational methods for recreation: There has been a great deal of recent work on applying 3D models to recreational purposes. Huang et al. [20] generated 3D mechanical collages by automatically assembling mechanical elements. Mitani and Suzuki [21] constructed papercraft toy models based on a particular mesh. (Papercraft puzzles are generated so that a 3D shape is converted into planar slices [22–24].) Kilian et al. [25] constructed an elegant surface via curved folding from a planar sheet. Mori and Igarashi [26] proposed a sketch system that helps users design plush toys. Bacher et al. [27] generated articulated deformable toys by estimating the joint positions of input meshes. Zhu et al. [28] presented a system that can plan an assembly mechanism to control the motion of a mechanical toy whose motion is specified by the designer; and Koo et al.’s [29] system assists users in generating prototypes of mechanical objects.

3. Approach

Fig. 1 is an overview of our system. The user provides a source model and a target skeleton as inputs, after which the system generates the output model and the motions of the transformation process in three stages: preprocessing, parts optimization, and transformation computation. We describe the first and third stages in this section, and the second stage in Section 4, below. We assume that all input models are manifold and symmetric (to a sagittal plane). This symmetry assumption saves computational time, in that only half of each model needs to be processed, because optimized segmentation and planned transformation can both be mirrored.

3.1. System input

Though the usual inputs of our system are a source model and a target skeleton, a user can also opt to provide a target model if she/he has a more specific shape in mind. Our system generates parts by segmenting the source model, and then rotates and/or translates the parts to create a transformed model. The source model is treated as the initial shape of a transformable model and the shape of the transformed model is determined by the target skeleton. Additionally, if a target model is given, our system will try to match the shape of the transformed model to that of the target model.

A target skeleton consisting of bones and joints not only depicts a transformed figure, but also indicates information on the connections between parts. There are two types of bones: bone and virtual bone. A bone has a corresponding part, while a virtual bone does not, as it merely denotes a connection relationship. We use virtual bones because they are more flexible for users assigning the orientation of parts. A joint may connect two parts, or more than two. Each joint has one translational degree of freedom (DOF) and three rotational DOFs ($r_x, r_y, r_z$). We adopt one translational DOF because we assume the bones in our system to be stretched along the bone’s orientation, i.e., they can be lengthened but not widened. It is a common setting in computer animation. However, if necessary, 3D translational DOFs also can be decomposed into three one DOF in our system. Also, we assume that the target model and its skeleton are both symmetrical. When creating the target skeleton, users are required to distinguish central joints, which are to be placed on the sagittal plane, from symmetric joints. They do not need to set the joint position precisely to make the target skeleton symmetric, however. Rather, the users just need to build a skeleton along the target model and our system will automatically create a symmetrical skeleton by adjusting joint positions during the preprocessing stage.

Guidelines for skeleton setting: To execute a design using our system, users should ideally provide a source model, a target model, and a skeleton rigged in the target model. The guidelines for users setting skeletons are as follows. First, the orientation of bones is important because our optimal segmentation ($Q_1$ term in Section 4) will produce a cylindrical part, which is corresponding to a bone, aligned with the orientation of the bone. Second, the computational time mainly depends on the number of parts, which is equal to the number of bones. As the number of virtual bones does not influence the computational time, users could
exploit the feature of virtual bones to describe a transformed model better.

3.2. Preprocessing

We first scale the target skeleton (and target model, if provided) to roughly the same size as the source model. Specifically, the target skeleton/model is scaled by $D_s/D_t$, where $D_s$ and $D_t$ are the diagonal lengths of the bounding boxes of the source and target models, respectively.

Since the target skeleton is initially built by users, its joint positions may not be symmetrical. To make them so, our system target models, respectively.

3.3. Animating transformation

A segmented source model is obtained after the optimal segmentation stage (Section 4, below). The segmented model is then animated by binding each part to a bone of the target skeleton. We observed that it does not look like the transformer motion if we move all joints simultaneously, so we only varied one or few DOFs at a time.

To fit the needs of our application, we designed a two-level motion-planning process, in which the order of moving joints is first determined, and then the motion path of a single joint is computed.

In the first level, we formulate the problem as one of finding the path with the least cost in a tree, and solve it using the Dijkstra algorithm. For a skeleton with $n$ joints, each node in the tree stores an $n$-digit binary number which shows the states of $n$ joints. There are two possible states of a joint: 0 if the four DOFs of the joint are at their initial values (initial pose), and 1 if the four DOFs reach their target values (transformed pose). Each node can be considered to be a pose. In the case of any two connected nodes, their node value would differ only by one digit, i.e., only one joint is transformed at a time. In this way, all possible movement sequences of $n$ moving joints can be encoded in an $n+1$-level tree. Each node also stores a value for the quantity of self-collisions for the represented pose. A path traced from the root to a leaf node represents a moving order of the joints. Fig. 4 presents an example of a tree for a model with three joints. Our goal is to find the path by which the joints move to their target poses with the fewest self-collisions.

In the second level, we adopt a bidirectional RRT to generate a transformation animation for every joint, resulting in a collision-free motion path from the source model to the transformed model. Specifically, our implementation uses the RRT-Blossom algorithm [30] to improve efficiency. The parameters of each joint to be planned are $(q, l)$, where $q$ denotes three rotational DOFs in quaternion and $l$ denotes one translational DOF.

In every iteration of RRT-Blossom, either rotational DOFs or translational DOF is changed. When changing its degree of rotation, the joint will rotate toward the target for $\epsilon_q$. When changing translation, the length $\epsilon_l$ is adjusted. In our experiments, we set $\epsilon_q$ to 30, and $\epsilon_l$ to between 10 and 30, depending on different cases.

4. Optimal segmentation

We formulated the segmentation of the source model as an optimization problem, with the objective of having joint position: (1) align correctly with bones, (2) minimize the number of collisions, (3) provide the closest possible match to the target model, (4) enable proper bone length, and (5) minimize the gaps between parts. In each iteration of the optimization, the joint positions of the skeleton vary, and a Voronoi diagram is applied to divide a source model into parts by referring sites at the center of bones, where these sites are used as the seeds to initialize a Voronoi diagram.
4. Objective function

We propose the following objective function to find an appropriate segmentation:

\[ Q = Q_1 + Q_2 + Q_3 + Q_4 + Q_5, \]

where \( Q_1 \) aligns segmented parts with their corresponding bones. According to [31], the ratio of a bone’s radius to its length is about 1:6. Hence, we approximate each bone by a cylinder \( C_s \) whose radius is one-sixth of its length. \( Q_1 \) is defined as

\[ Q_1 = \sum_{i} w_1 \left( 1 - \frac{V(P_i \cap C_i)}{V(M_i)} \right) + w_5 \left( 1 - \frac{V(M_i - C_i)}{V(M_s)} \right), \]

where \( P_i \) is a part and \( V(M_i) \) is the volume of the source model \( M_i \). The operators \( \cap \) and \( - \) denote the intersection and difference operations in constructive solid geometry (CSG), respectively. We approximate the volume of a part by the volume of its bounding box. The purpose of the first term is to increase the intersection volume of \( P_i \) and \( C_s \) while that of the second term is to decrease the difference between them. The weights \( w_1 = 1.6 \) and \( w_5 = 2.0 \) have been chosen for all the examples.

\( Q_2 \) penalizes the self-collision between the transformed parts:

\[ Q_2 = w_2 \sum_{i \neq j} \frac{V(P_i^t \cap P_j^t)}{V(M_i)}. \]

where \( P_i^t \) is a transformed part (\( w_2 \) has been set at 1.0 for all the examples).

\( Q_3 \) measures the shape difference between the transformed model and the target model \( M_t \):

\[ Q_3 = w_3 \sum_{i} \left( 1 - \frac{V(P_i^t \cap M_t)}{V(M_t)} \right). \]

The more \( P_i^t \) intersects with the target model \( M_t \), the smaller the value of \( Q_3 \) is. If the user has provided a target model, \( w_3 \) is set as 1.7; otherwise, \( w_3 \) is equal to zero.

\( Q_4 \) avoids dense distribution of sites. Since the Voronoi diagram partitions a space based on the positions of sites, sites distributed too densely will lead to fragmented parts and scattering of transformed parts. \( Q_4 \) is defined as follows:

\[ Q_4 = w_4 \sum_{i} \left( 1 - \frac{V(C_i)}{V(M_i)} \right) + w_{10} \left( 1 - \frac{V(S)}{V(M_i)} \right), \]

where \( V(S) \) is the volume of the bounding box of the skeleton in the current iteration. The first term penalizes short bones to avoid high site density. The second term globally adjusts the distribution of sites to cover the source model. We set \( w_4 = 3.2 \) and \( w_{10} = 1.1 \) in our experiment.

\( Q_5 \) penalizes the gaps between transformed parts:

\[ Q_5 = w_5 \sum_{i \neq j} \frac{H(\nu(P_i^t), \nu(P_j^t))}{D_i}, \]

where \( \nu(P_i^t) \) is the set of all vertices on the mesh of \( P_i^t \); \( H(\nu(\nu_i), \nu(\nu_j)) \) is the Hausdorff distance between the two sets \( \nu_i \) and \( \nu_j \); and \( D_i \) is the diagonal length of the bounding box of the source model \( M_i \). \( w_5 \) is set as 1.2 for all the examples. Fig. 5 illustrates the effects of each term in the objective function by showing the segmented results at different values. Please note that \( P_i^t \)'s and \( P_j^t \)'s in each objective term are determined by the joint positions of skeleton \( S \), which are the variables to be optimized.

4.2. Simulated annealing

We solve the optimization problem using simulated annealing (SA) [32], a generic probabilistic method for the global optimization problem. The original concept of annealing is that after a solid object is heated, its elements will be rearranged into a permutation that requires less energy. In the process of annealing, when the temperature is high, the energy will tend to be high as well.
With the decrease of the temperature in every iteration, the energy is lessened. Based on this concept, the SA algorithm was developed to find optimal (lowest energy) solutions, and can do so in a large space within a limited time. In our optimization problem, SA randomly changes the joint positions in each iteration. As the number of iterations increases, the movement range of each joint decreases.

In our implementation, we set the initial temperature as 0.7. When the temperature is 0.7 and $\Delta e_n$ is 0.5, the probability of transition between states is set as 0.5. We define $\Delta e_n$ using Eq. (2), where $e_n$ is the energy of the current state, i.e., the value of the objective function defined in Eq. (1) in the current iteration. $e_{\text{min}}$ is the minimal energy that ever reached from the initial to the current state. The temperature will decrease by $7/1_{\text{max}}$ in every iteration. If there are $1_{\text{max}}/10$ consecutive iterations in which lower energy is not reached, the temperature will be decreased to the lowest value, zero. If $e_n$ is lower than the current lowest energy $e_{\text{min}}$, the temperature will increase by $70/1_{\text{max}}$. If this occurs, the state could move toward high energy to find the global minimum instead of getting stuck in a local minimum.

$$\Delta e_n = \min \left( \frac{e_n - e_{\text{min}}}{e_{\text{min}}}, 1.0 \right)$$  \hspace{1cm} (2)

Whether the state changes is determined by $\Delta e_n$ and temperatures. We apply the probability function proposed by Metropolis (Eq. (3)) to calculate the transitional probability of a state. When $\Delta e_n < 0$, it implies that we find a new state with lower energy, so we will move to the new state. When $\Delta e_n \geq 0$, the transitional probability will be determined by $\Delta e_n$ and temperature. Assuming that the temperature remains constant, the state will be more stable when $\Delta e_n$ is larger. On the contrary, if $\Delta e_n$ remains the same, the state will be more likely to change when the temperature is higher.

$$P(e_n, e_{\text{min}}, \text{temp}) = \begin{cases} 1 & \text{if } \Delta e_n < 0 \\ e^{-\frac{\Delta e_n}{\text{temp}}} & \text{if } \Delta e_n \geq 0 \end{cases}$$  \hspace{1cm} (3)

The optimization process is terminated if any of the following conditions holds: (1) When the number of iterations exceeds the largest number of iterations set by the user ($l_{\text{max}}$); (2) When the energy is lower than the threshold provided by the user ($e_{\text{threshold}}$); and (3) If the energy does not decrease after $l_{\text{max}}/10$ consecutive iterations.

5. Results and discussion

5.1. Results

We tested our approach on three examples: a car-robot, a cylinder-toy, and a board-chair, with the first two demonstrating typical applications of transformable robots or other toys, and the last, that of foldable furniture. Table 1 shows the statistics for these examples, the inputs and outputs of which are shown in Figs. 1 and 6. Please see also the accompanying animation video.

In Fig. 1, the transformed model forms a humanoid shape that approximately represents the target model. Fig. 6(a) demonstrates a board transforming into a chair, with the transformed model representing a chair shape well. In Fig. 6(b), a cylinder is transformed into a toy. In these three examples, although the transformed model is similar to the figure of the target model, they are far from identical. Several factors may explain the phenomenon.
First, the region segmented by a Voronoi Diagram is always a convex polytope. Thus, it is difficult to generate a thin-disk part.

Second, the distribution of sites determined by the bone centers may limit the number of possible ways of segmenting a model.

Third, the objective function (for measuring the differences between the transformed model and the target model) may not be very accurate.

5.2. Evaluation

To evaluate the quality of our results, we conducted a user study which compared our method against a naïve approach. The naïve approach cuts a source model into several regular pieces through a voxelization process, in which the number and size of voxels are set manually according to the number of bones and the shape of the skeleton. It then binds these regular pieces with the nearest bone and extends the skeleton to the corresponding position.

![Fig. 7](image)

Fig. 7. A comparison of the results with and without Q₃. (a) The car-robot result optimized with all terms. (b) The car-robot result optimized without Q₃. The input source and target models are shown in Fig. 6(b). (a) All. (b) Without Q₃.

Table 3

<table>
<thead>
<tr>
<th>Preference</th>
<th>Car-robot</th>
<th>Cylinder-toy</th>
<th>Board-chair</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td># Same order</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td># Prefer ours</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td># Prefer theirs</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

5.2. Evaluation

To evaluate the quality of our results, we conducted a user study which compared our method against a naïve approach. The naïve approach cuts a source model into several regular pieces through a voxelization process, in which the number and size of voxels are set manually according to the number of bones and the shape of the skeleton. It then binds these regular pieces with the nearest bone and extends the skeleton to the corresponding position.

![Fig. 7](image)

Fig. 7. A comparison of the results with and without Q₃. (a) The car-robot result optimized with all terms. (b) The car-robot result optimized without Q₃. The input source and target models are shown in Fig. 6(b). (a) All. (b) Without Q₃.

Table 3

<table>
<thead>
<tr>
<th>Preference</th>
<th>Car-robot</th>
<th>Cylinder-toy</th>
<th>Board-chair</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td># Same order</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td># Prefer ours</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td># Prefer theirs</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

5.3. Validation

Validation of the objective function. We further validated our system’s objective function. Fig. 8 compares the optimized results with and without each term of this function. Fig. 8(b) and (e) shows that the sizes of bones are uneven without Q₃ or Q₅, while Fig. 8(c) indicates that the parts penetrate each other without Q₃. The effect of Q₂ is not obvious in this case, because the source model and the target model differ significantly. However, Q₃ is effective in the

![Fig. 8](image)

Fig. 8. Verification of the objective function. (a) The car-robot result using all terms. (b)–(f) The car-robot results with one term of the objective function removed. (a) All, (b) without Q₁, (c) without Q₂, (d) without Q₃, (e) without Q₄, (f) without Q₅.
cylinder-toy case, as shown in Fig. 9. Fig. 8(f) shows that there are more gaps between the parts without $Q_2$.

**Validation of first-level motion planning:** We conducted a user study to validate the effectiveness of automatic ordering, i.e., first-level motion planning. Five participants aged between 22 and 24, including one male and four females, were recruited to experience the process of manual ordering.

In this experiment, participants were asked to decide the order of joint movements for all three of the above-mentioned objects. Some information was provided to assist the participants in this task: (1) they were given models of skeleton before and after transformation, and the optimally segmented parts (Section 4); and (2) the self-collision cost was provided for each order the participants came up with. They were allowed 15 min per object to accomplish first-level motion planning. Once a participant decided on a provisional movement order, she/he was allowed to use our system to run second-level motion planning to obtain result animation, and then repeatedly adjust the movement order until satisfied with the animation. (Since second-level motion planning might take up to several minutes, the self-collision cost information we provided could assist the participants in predicting the result more quickly.) Then, we asked the participants to compare their animations with ours if their joint movement order is different from that generated by our first-level motion planning.

The results of this experiment are shown in Table 3. Since each of the five participants provided three orders, 15 orders were generated. Among these 15, four were as same as ours and 11 were different. For those 11 different orders, we asked the participants whether they prefer the movement orders generated by them or by our system. It turns out that our system are preferred in 10 of 11 comparisons. This result validates the effectiveness of our joint movement order planning. It should also be noted that it was hardest for the participants to arrive at a satisfactory order within the time limit in the case of the car-robot, which was clearly the most complicated of the three. Additionally, in the interviews we conducted after the experiment, all participants agreed that self-collision cost was an important factor in deciding movement order.

### 5.4. Limitations and discussion

There are still several limitations in our approach. Some possible improvements to the system should be considered. First, the current version only optimizes the skeleton, which determines the segmentation of a model. A two-level optimization, which first optimizes the skeleton then the segmentation, may generate results that better fit users’ needs. Second, an improved version of the system could incorporate more 3D geometry techniques, such as the extraction of geometry features from a target model or the deformation of a source model, which could render the transformed model more similar in appearance to the target model. Third, because our system connects pairs of parts using a telescopic rod that can extend and shrink arbitrarily, our current transformable models can only be used in animation, and cannot be 3D printed as physical objects. To solve this problem, the connection positions between parts and the geometry of connection joints should also be taken into account when segmenting the source model. It might also be possible to adapt the approach proposed by Cali et al. [33] for generating printable joints for a 3D mesh.

### 6. Conclusion

An approach for making and animating a transformable model is proposed in this paper. Users only need to provide a source model and a target skeleton. Our system then can automatically segment the source model and generate a transformation animation. This can assist artists to design transformable models. We believe our system could lead to a number of practical applications in making transformable animations and designing toys, robots, tools, or furniture in the future.

### Acknowledgements

We thank the anonymous reviewers for their insightful comments. This work was supported in part by Taiwan Ministry of Science and Technology (MOST) under grants 101-2628-E-009-021-MY3 and 102-2221-E-009-082-MY3, and UST-UCSD International Center of Excellence in Advanced Bioengineering sponsored by the MOST I-RiCe Program under grant number: MOST 103-2911-I-009-101-.

### Appendix A. Supplementary material

Supplementary data associated with this article can be found in the online version at [http://dx.doi.org/10.1016/j.cag.2015.07.014](http://dx.doi.org/10.1016/j.cag.2015.07.014).

### References


