Received: 15 March 2018 | Accepted: 21 March 2018
DOI: 10.1002/cav.1821

SPECIAL ISSUE PAPER

# Sketch-based shape-preserving tree animations

Yutong Wang[1] | Luyuan Wang[1] | Zhigang Deng[2,3] | Xiaogang Jin[1]

[1]State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310058, China
[2]Virtual Reality and Interactive Technique Institute, East China Jiaotong University, Nanchang, China
[3]Department of Computer Science, University of Houston, Houston, TX, USA

**Correspondence**
Xiaogang Jin, State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310058, China.
Email: jin@cad.zju.edu.cn

Zhigang Deng, Virtual Reality and Interactive Technique Institute, East China Jiaotong University, Nanchang, China; or Department of Computer Science, University of Houston, Houston, TX, USA.
Email: zhigang.deng@gmail.com

**Funding information**
National Key R&D Program of China, Grant/Award Number: 2017YFB1002600; Key Research and Development Program of Zhejiang Province, Grant/Award Number: 2018C01090; National Natural Science Foundation of China, Grant/Award Number: 61732015

**Abstract**

We present a novel and intuitive sketch-based tree animation technique, targeting on generating a new type of special effect of smoothly transforming leafy trees into morphologically different new shapes. Both topological consistencies of branches and meaningful in-between crown shapes are preserved during the transformation. Specifically, it takes a leafy tree and a user's sketch describing the silhouette of the desired crown shape under a certain viewpoint as the input. Based on a self-adaptive multiscale cage tree representation, branches are locally transformed through a series of topology-aware deformations, and the resulting tree conforms to the user-designed shape, demonstrating better aesthetics compared to global single-cage-based methods. By interpolating the transformations, we are able to create visually pleasing shape-preserving animations of trees transforming between two crown shapes. Our proposed framework also provides an efficient way to interactively edit leafy trees toward desired shapes, demonstrating its potential to leverage existing tree modeling frameworks by providing flexible and intuitive tree editing operations.

**KEYWORDS**

shape preservation, sketch-based editing, special effects, tree animation

## 1 | INTRODUCTION

Despite its long history, the study of tree animation remains active. Other than animating real-world scenarios, such as the trees' growth process[1,2] and their interactions with the environment,[3,4] special requirements in animating stylized trees arise in the film and animation industry in order to nurture the mysterious atmosphere with compelling visual effects.[5] Consequently, topology-aware approaches have been proposed to generate smooth transformations between trees, achieving visually compelling special effects.[6–8] However, they do not put explicit efforts into preserving a sequence of morphologically continuous crowns when transforming between trees with specialized crown shapes, leading to less meaningful in-between trees (see Figure 9, top row).

In this paper, we propose a novel sketch-based shape-preserving tree animation method, which smoothly transforms a leafy tree into custom crown shapes defined by designers' sketches while preserving the morphological meanings of in-between trees. Our method contributes to a new type of special visual effect for leafy trees. Figure 1 shows an example of fluidly transforming a *Salix* with sphere-like crown into a heart-shaped one. It is noteworthy that both the topological consistencies of branches and the morphological meanings of crowns are preserved during the animation. One additional

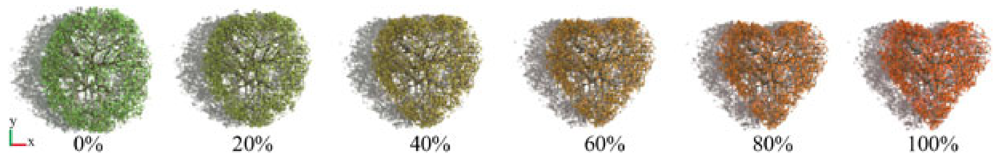


**FIGURE 1** Shape-preserving transformations of a sphere-shaped *Salix* into a heart-shaped one

benefit of our method is that it also provides an intuitive way to efficiently control and edit leafy trees into desired crown shapes using one user-designed sketch alone.

The key technical challenge to achieving shape-preserving tree animations is how to transform tree branches to form morphologically meaningful crown shapes while preserving their topological hierarchies. To tackle the challenge, we propose a self-adaptive multiscale cage-based deformation strategy to hierarchically propagate the transformations of crowns to tree branches. Specifically, the input leafy tree is first abstracted into a skeleton-lobe representation,[7] where the foliage is clustered into canonical geometries (i.e., *lobes*) in order to describe the tree in a hierarchical multiscale manner. The morphologically meaningful and continuous in-between trees are achieved by first gradually deforming the tree's crown according to the designer's sketch and then hierarchically propagating the deformation to tree branches based on the self-adaptive multiscale cage tree (MCT), in which the nodes are the view-dependent silhouettes serving as cages and the edges maintain their topological hierarchies. Since branches are transformed under both topology-aware and shape-aware deformations, the morphological meanings of the resulting trees are guaranteed.

**Contributions**. To the best of our knowledge, our work creates the first not only topologically consistent but also shape-preserving animations of transforming leafy trees into morphologically different new shapes. The contributions are twofold: (a) a novel tree animation algorithm that smoothly transforms a tree into crowns with custom shapes while preserving the topological consistency of branches and morphological meanings of crown shapes; (b) an intuitive tree editing method that allows designers to directly control the shape of the tree through view-dependent oversketching operations.

## 2 | RELATED WORK

**Shape-guided tree modeling**. To date, a large number of shape-guided tree modeling methods have been proposed to generate trees conforming to custom crown shapes. Specifically, there are 3D shape-guided methods[9–12] and sketch-based interactive methods.[13–16] However, both of them only generate static trees and are sensitive to the change of the guidance shapes. In other words, whenever the guidance shape is changed, the algorithm just re-runs to regenerate a new tree. Because of the low efficiency and no warranty of topologically consistent tree branches, they are not suitable for generating shape-preserving tree animations.

**Tree animations**. The existing literature on tree animation can be roughly classified into two categories: animating the trees' development and their interactions with the environment. Procedural methods such as in the works of Lintermann and Deussen,[1] Pirk et al.,[2] and Prusinkiewicz[17] have been proposed to create botanically plausible tree growth animations by interpolating a number of predefined developmental parameters. However, none of them are capable of generating shape-preserving animations of transforming leafy trees into custom shapes. The main reason is that these developmental parameters have no direct association with the trees' crown shapes, and interpolating them cannot guarantee the generation of morphologically continuous and meaningful in-between crowns. In addition to growth animations, various efforts have been conducted to deform and prune branches of a tree in order to generate realistic responses to environmental factors, such as wind,[4,18] light,[3] fire,[19,20] and obstacles.[3,21] Despite that the shape of the tree is changed according to the environment, it cannot be smoothly transformed into a user-defined shape since existing methods do not put explicit efforts into preserving the morphological meanings of the trees' crown shapes.

**Topology-aware tree animations**. Recent works[6,7] have created topologically consistent animations between two topologically varying trees. However, the work of Wang et al.[6] does not handle the animations of foliage. Although the work of Wang et al.[7] addresses the smooth transformations of foliage, it fails to preserve the morphological meanings of in-between trees' crowns, leading to less compelling visual effects when transforming between trees with custom crowns. In addition, this method requires exactly two trees as input, whereas ours only requires one tree and a designer's sketch describing the silhouette of the desired crown.

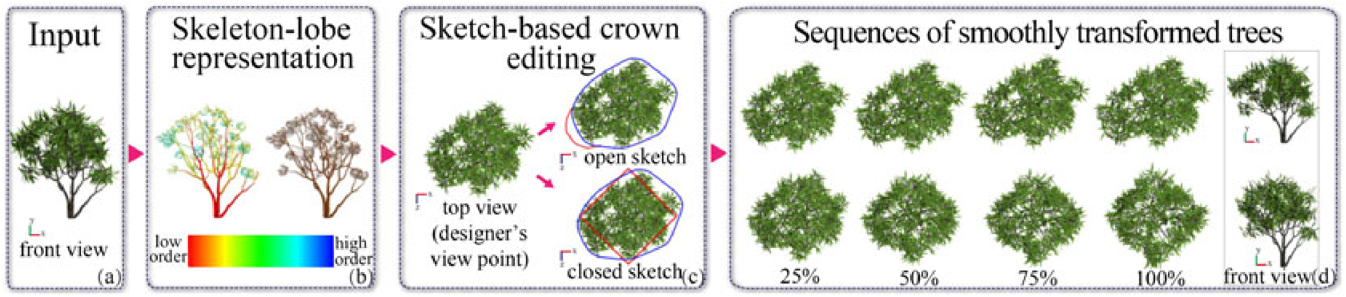

Input
front view
(a)
Skeleton-lobe representation
low order
high order
(b)
Sketch-based crown editing
top view (designer's view point)
open sketch
closed sketch
(c)
Sequences of smoothly transformed trees
25%
50%
75%
100%
front view
(d)

**FIGURE 2** The pipeline of our method

## 3 | APPROACH OVERVIEW

Figure 2 illustrates the main steps of our shape-preserving tree animation approach. Given a leafy tree (Figure 2a), it is first converted into a branching-pattern-aware skeleton-lobe representation using topologically consistent morphing[7] (Figure 2b). From a certain viewpoint, designers are allowed to change the shape of the tree by either oversketching the detected crown's silhouette (open sketch) or drawing a new shape describing the desired crown shape under the viewpoint (closed sketch); see Figure 2c, where the crown silhouettes are plotted in blue, whereas the designers' sketches are plotted in red). The crown deforms according to the new silhouette and hierarchically propagates the deformation to tree branches based on an MCT, which preserves the hierarchical topology of both the tree branches and their corresponding lobes. As a result, the tree is gradually transformed, and a fluid animation of transforming a tree into the sketched crown shape is generated (Figure 2d).

## 4 | SKELETON-LOBE REPRESENTATION

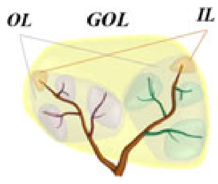
Given a leafy tree, we first abstract it into a branching-pattern-aware skeleton-lobe representation as in the work of Wang et al.[7] Below, we summarize the essential terms pertinent to our work. Readers are referred to the work of Wang et al.[7] for more algorithm details.

**Chain**. To describe the branching hierarchies of a tree, we label the tree branches with hybrid orderings[7] and define the consecutive branches without ordering changes as a *chain*.

**Branching pattern**. In the real world, the arrangement of branches at a branching point generally follows one of three patterns: *alternative*, *opposite*, and *whorled*.[22] In this paper, we encode the branching patterns into *chain groups*, which consist of chains growing at the same branching points while sharing the same orderings.

**Lobe**. The general term used for canonical geometry formed by a cluster of leaves.

**Multiscale lobe descriptions**. Following the work of Wang et al.,[7] three types of lobes (see inset) are defined in our work. Specifically, the *inner lobe* (*IL*), which only contains leaves growing at the tip of a chain, is interpreted as the finest scale of description; the *outer lobe* (*OL*) conveys a coarse description of a chain's foliage by describing the shape formed by the leaves that covers all its substructures; the *group outer lobe* (*GOL*), defined as the coarsest scale of description, depicts the general shape of a chain group's foliage and is mathematically equal to the union of *OL*s of the chains within the group. It is noteworthy that the *GOL* is equivalent to *OL* if and only if the chain group consists of only one chain. This leads to a multiscale description of a tree's foliage (see Figure 3a). At the coarsest scale, the tree is only described by the root chain group and the *GOL*, which is also referred to as the *crown* in this paper. More detailed foliage is described at finer scales, where the tree is described by higher-ordered chain groups and their *GOL*/*OL*s, and at the finest scale, the tree is described by all the chain groups and the *IL*s.



**Branching-pattern-aware topology tree**. With the multiscale lobe descriptions, we encode their topological hierarchies into a branching-pattern-aware topology tree (i.e., the *BPTT*[7]; see Figure 3b). Nodes of the *BPTT* are defined as a tuple $N_i := < GOL_i, GC_i >$, where $GOL_i$ is the *GOL* of the chain group $GC_i$ exhibiting a certain branching pattern. Edges in the *BPTT* are defined in the form of $(N_i, N_j)$, implying a hierarchical relationship between $N_i$ and $N_j$ on the condition that $GOL_i$ is the parent of $GOL_j$.

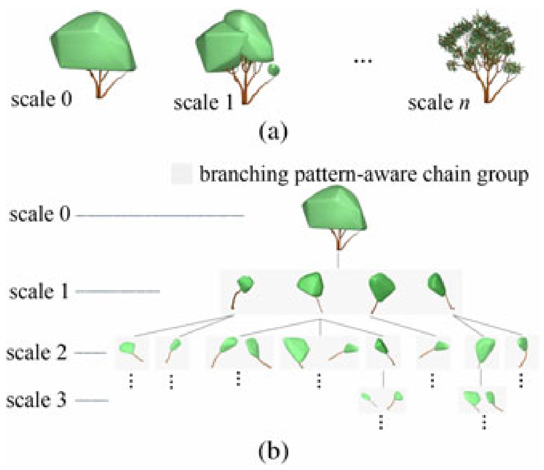


**FIGURE 3** Multiscale representations of a tree. (a) Multiscale representation of a tree. (b) Branching-pattern-aware topology tree

## 5 | SKETCH-BASED TREE ANIMATION

### 5.1 | Sketch-based crown editing

The shape-preserving tree animation is initiated by the change of the tree's crown. In our method, we employ view-dependent sketching operations to interactively edit a leafy tree's crown shape. The edit of a crown is accomplished in two steps: (a) detect the crown's silhouette; (b) oversketch a part of the silhouette (open sketch) or redraw a new (closed) sketch to describe the desired shape (see Figure 2c).

**Silhouette detection**. Among various silhouette detection approaches,[23] we choose the object-space silhouette (or simply the silhouette) for its accuracy and convenience to potential editing operations. Specifically, the silhouette is piecewise linear, denoted as $S = \{p_i \mid i \in (1, \dots, n)\}$, and $p_i$ satisfies $n_i \cdot (p_i - C) = 0$, where $n_i$ is the vertex normal and $C$ is the viewing position.

**Silhouette oversketching**. Designers are allowed to either draw an open sketch (red line in Figure 2c, top row) to slightly modify the crown shape (Figure 2d, top row) or redraw a closed sketch (red line in Figure 2c, bottom row) to define a new shape (Figure 2d, bottom row). Both of the sketches are view dependent, representing the projection of the desired crown shapes in the screen space, which are later transformed into the object space by establishing a mapping to the crown silhouette using screen space arc-length parameterization.[24]

### 5.2 | View-dependent MCT

According to the *BPTT*, under a certain viewpoint, the silhouettes of a tree's lobes, including *GOL*s, *OL*s, and *IL*s, also exhibit hierarchical features. In addition, the fact that coarser-scale lobes fully envelop the finer-scale lobes makes it reasonable to consider the coarser-scale silhouettes as cages of the finer-scale silhouettes. Therefore, we formalize the hierarchical silhouettes into an MCT (in graph-theoretic sense), that is, *MCT*, in which the nodes are the view-dependent silhouettes, viewed as cages, and the edges encode the parent–child relations between nodes (see Figure 4). Based on the objects that are directly influenced by the changes in cage vertices, two types of cages are distinguished: *leaf cages*,

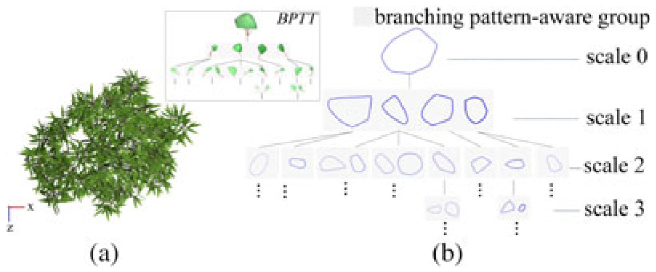


**FIGURE 4** The view-dependent multiscale cage tree. (a) Input leafy tree. (b) View-dependent multiscale cage tree

corresponding to the *IL*'s silhouettes, that directly deform tree branches and *internal cages* (none-leaf cages), corresponding to the silhouettes of the *GOL* and the *OL*, that only affect the finer-scale cages.

It is noteworthy that the hierarchical and multiscale nature of the *MCT* implies that a coarser-scale cage only directly controls the deformation of its child cages at the finer scale and that any cage vertex can only be controlled by the parent cage. In other words, when a cage at a certain scale changes, the transformation should only be propagated downward the *MCT* hierarchy. In addition to maintaining the topological hierarchies of the lobes' silhouettes, the *MCT* brings two additional benefits. It not only preserves the morphological consistencies of branches and lobes by ensuring that transformations can only be propagated from the coarser-scale cages to the finer-scale cages but also provides the capability of producing *local* transformations at any scale (see the self-adaptive cages in Section 5.4.1).

## 5.3 | The baseline transformation

Our algorithm hierarchically propagates the transformation induced by the sketch downward the *MCT* to the finer-scale internal cages and finally to leaf cages, resulting in a shape-preserving transformation of the tree. Specifically, it works in three steps (see Figure 5).

**Step 1: Transform internal cages (Figure 5b)**. We employ mean value coordinates (*MVC*)[25] to compute the new cage vertices resulted from the transformation of the coarser cages. By definition, the *MVC* of a point $p_i$ inside a given cage $C$ is a set of parameters $w_{ij}$ satisfying $p_i = \sum_{v_j \in C} w_{ij} v_j$, where $v_j$ is the cage vertex. This implies that any point inside a cage can be represented as a linear combination of the cage vertices. Therefore, when the coarser-scale cage is transformed, the new position of a finer-scale cage vertex $p'_i$ is calculated using the *MVC* $w_{ij}$ as $p'_i = \sum_{v_j \in C} w_{ij} v'_j$, where $C$ is the coarser-scale cage and $v'_j$ is the changed cage vertex.

**Step 2: Transform lobes (Figure 5c)**. The change of an *internal* cage automatically induces the corresponding lobe to deform. By fixing the cage vertices as constraints, we use the Laplacian deformation algorithm[24] to compute the deformed shape by minimizing the following linear system:

$$\sum_{v'_i \in \mathcal{M}} \left\| \mathcal{L}\left(v'_i\right) - \sigma_i \right\|^2 + \sum_{v_j \in C} \left\| v'_j - v_j \right\|^2, \tag{1}$$

where $\mathcal{L}(\cdot)$ is the Laplacian operator, $\sigma_i$ is the Laplacian coordinate computed using cotangent weights, $v'_i$ is the transformed surface point, and $v_j$ is the constraint cage vertex with transformed new position $v'_j$. The first term perserves the local details of the shape in $\mathcal{L}(\cdot)$, and the second term penalizes the changes of the constraint points during the transformation.

**Step 3: Transform tree branches (Figure 5d)**. Tree branches are not transformed until the transformation is propagated to the *leaf* cages. To maintain the topological consistency, they are transformed by two consecutive deformation propagations: the *backward* and *forward* propagations. Given a branch and its transformed *IL*, the target position of the branch tip is calculated using the *MVC* coordinates.[25] Served as a constraint point, the branch tip is transformed into the target position, and the transformation is propagated backwards to the root of the branch using Equation (1). Once done, the branch forwards the transformation to its child branches by fixing the child roots as the constraint points and performing the transformation in a similar way using Equation (1).

By interpolating the transformation process, the tree is smoothly transformed into the designer's custom shape, without violating the branches' topological consistencies.

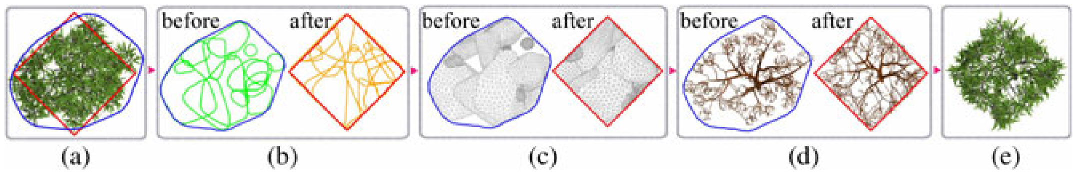


**FIGURE 5** The main steps of the baseline transformation. (a) Input tree. (b) Transform internal cages. (c) Transform lobes. (d) Transform tree branches. (e) Result

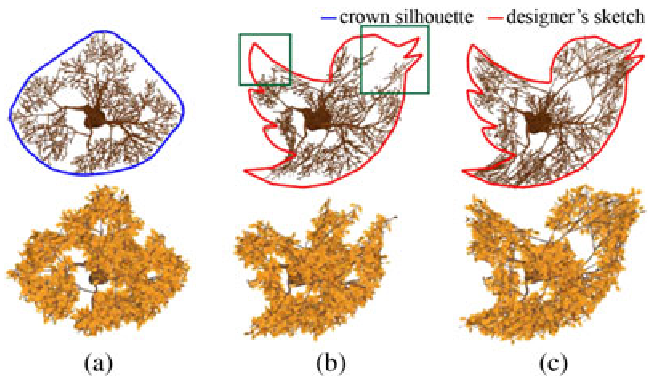


**FIGURE 6** (a) Input leafy tree and comparison of (b) the baseline transformation and (c) the self-adaptive shape-preserving transformation

## 5.4 | Shape-preserving transformation

Although maintaining the topological consistency, the above propagation does not guarantee the shape-preserving transformations of tree branches. In other words, the resulting trees might poorly match the designer's expectations (see the circled regions in Figure 6b). Two major factors that weaken the aesthetics of the resulting trees are *poor crown coverage* and *unnaturally deformed branches*, which can be solved by employing shape-aware self-adaptive cages and pruning branches outside the designer's desired shapes, respectively.

### 5.4.1 | Self-adaptive cages

The issue that a resulting tree poorly conforms to the target crown shape (green regions in Figure 6b) arises when the branches of the input leafy tree do not give full coverage of the crown (Figure 6a). Unfortunately, this is a natural phenomenon of real-world trees since the growth of both branches and leaves is influenced by environmental factors, which might shed branches and leaves because of light, wind, etc. In our baseline transformation algorithm, cages and tree branches are transformed according to the constraint points computed by *MVCs* that preserve the positions of the constraint points relative to their corresponding cages. Consequently, the uncovered regions of the input crown remain uncovered after the transformation, leading to unsatisfying results (see Figure 6b).

To solve the problem, we introduce the shape-preserving self-adaptive cage into the baseline transformation, which automatically deforms the finer-scale cages to "fill in" the uncovered regions of their parental cages at the coarser scale. Specifically, when an *internal* cage is transformed and prior to forwarding the transformation to its children, we analyze and partition the cage into segments based on the coverage of the child cages (see Figure 7a). Each of the segments is characterized as one of two types: the *covered* or *uncovered* segments, denoted as $O$ and $F$, respectively. A segment is labeled $O$ if there is at least one child cage that is partially matched within a distance threshold (10 pixels in the screen space); otherwise, it is labeled $F$ (see Figure 7a).

To maximize the coverage, we greedily deform the child cages. For instance, given an $F$ segment with neighboring segments $O_i$ and $O_j$, for every child cage that is partially matched to either $O_i$ or $O_j$ within a threshold (10 pixels in the screen space), we perform Laplacian deformation[24] to adjust the cage with strategically selected constraint points. With the entire child cage considered as the region of interest, the vertices of the cage that are partially matched to the

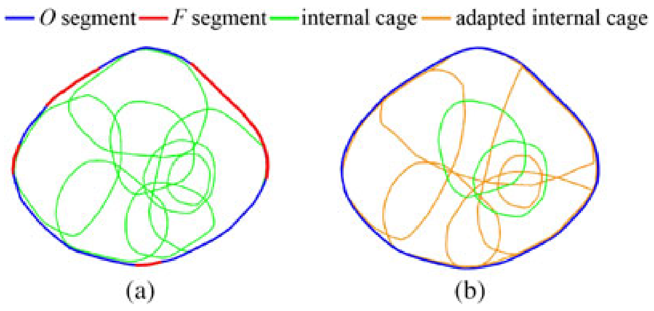


**FIGURE 7** Self-adaptive cages. (a) Partition silhouette into $F$ and $O$ segments. (b) Self-adaptive internal cages covering the $F$ segments

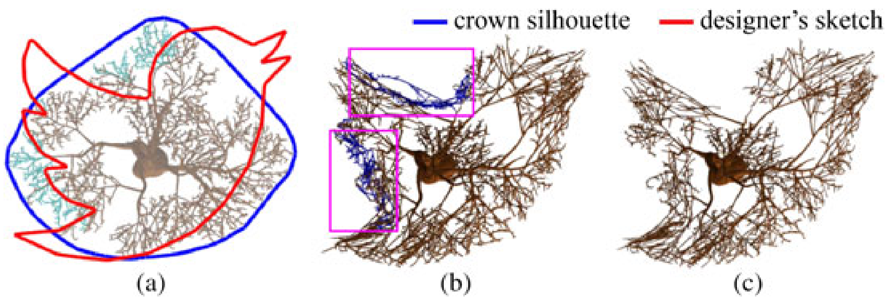


**FIGURE 8** Comparison of pruning branches. (a) Input with pruned branches (blue). (b) Transform without branch pruning. (c) Transform with shape-preserving branch pruning

$O_i$ or the $O_j$ segment are automatically selected as constraint points. Furthermore, with the mappings induced by the screen-space arc-length parameterization[24] of the child cage and the $F$ segment, the cage vertices that are matched with the $F$ segment are also selected as additional constraint points. As a result, the cage is deformed to cover the $F$ segment and then propagates the transformation to the finer-scale cages based on the *MCT*.

### 5.4.2 | Pruning branches

In order to avoid unnaturally transformed branches, we propose a pruning algorithm to strategically wilt several branches (see the blue branches in Figure 8b). Branch pruning is hierarchically performed based on the *MCT*. Starting from the coarsest scale, we clip cage $C$ with the designer's sketch using *Vatti*'s polygon clipping algorithm.[26] In case that the cage is clipped into multiple pieces, we choose the one with the largest area as the clipped cage $C'$. The areas of both $C$ and $C'$, as well as the ratio $r$ of the $C'$'s area to the $C$'s area, are calculated. Based on $r$, branches are pruned by performing the following tests.

(i) If $r$ is smaller than a certain threshold, which is experimentally set to 30%, the cage is marked as pruning. Based on the *MCT*, all of the child cages are also marked as pruning to avoid the violation of the topological consistency.
(ii) If $r$ is approximate to 1, it implies that the cage is nearly completely inside the sketch and requires no more pruning tests.
(iii) Otherwise, we test the finer-scale cages based on the *MCT* and repeat the process until the *leaf* cages are tested or either the above condition (i) or (ii) is satisfied.

In the end, branches whose corresponding cages are marked as pruning are gradually wilted using the topology-aware method.[7] Figure 8 demonstrates an example of the branch pruning algorithm.

## 6 | RESULTS AND DISCUSSION

### 6.1 | Results

We collected 60 real-world trees from the Internet (i.e., http://www.evermotion.org/). In addition, 10 designers' hand-modeled virtual trees were also incorporated into the data set to evaluate the effectiveness and flexibility of our algorithm. Figure 2 shows the result of smoothly transforming an oval-shaped *Cabbage* tree into slightly modified crown shape and a completely new diamond shape. Figures 1, 9, and 10 demonstrate the visually pleasing shape-preserving transformations for different tree species. Please refer to our animation results in the supplemental video.

**Performance**. We implemented our algorithm in C++ on a desktop equipped with Intel© i7 4.0GHz CPU and 32GB RAM. The runtime statistics are presented in Table 1. In sum, our method demonstrated its efficiency of generating smooth results on an off-the-shelf computer.

### 6.2 | Discussion

**Comparison with the space colonization method**. Figure 11 compares our work with the space colonization method,[10] which generates trees with custom crown shapes using a procedural growth model to simulate the natural

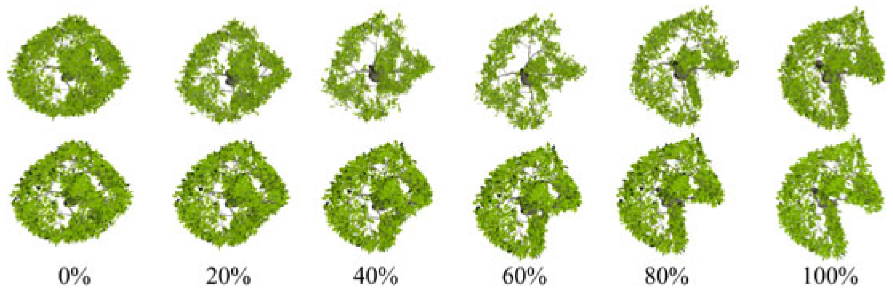


**FIGURE 9** Comparison between our method (bottom row) and topologically consistent morphing[7] (top row). The latter generates less meaningful in-between crowns due to the branch correspondences unaware of the crown's morphological meanings

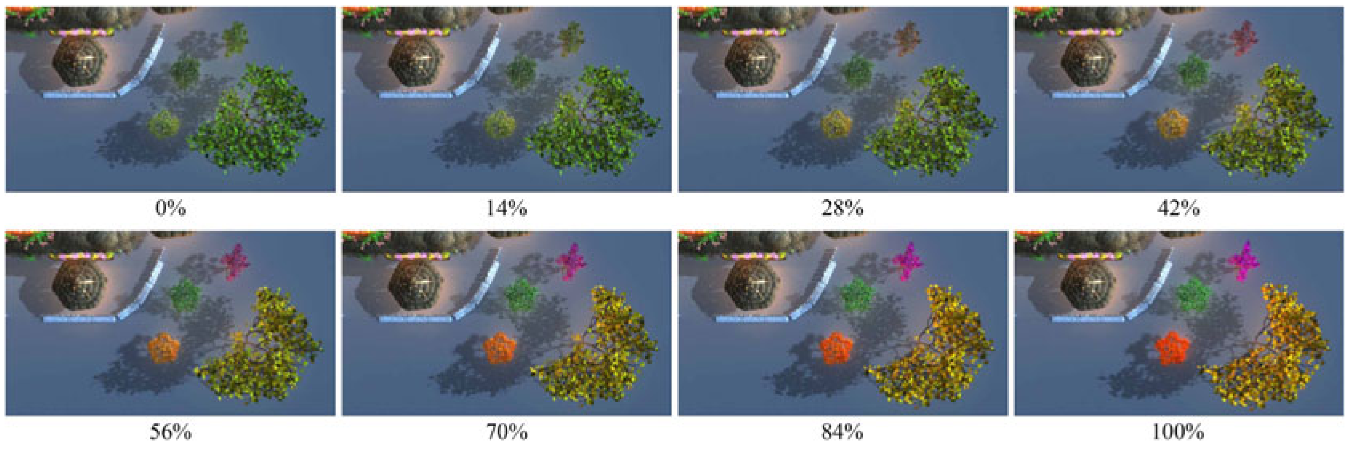


**FIGURE 10** A fairy scene of transforming leafy trees into a moon and three stars

**TABLE 1** Runtime statistics

| Ex. | Input | | Time |
|---|---|---|---|
| | # Branches | # Leaves | (per frame) |
| Figure 1 | 5,031 | 3,830 | 1.14 s |
| Figure 2 | 1,800 | 1,702 | 0.32 s |
| Figure 6 | 3,917 | 3,787 | 2.05 s |
| Figure 9 | 3,976 | 3,852 | 0.67 s |
| Figure 13 | 1,774 | 1,766 | 0.57 s |

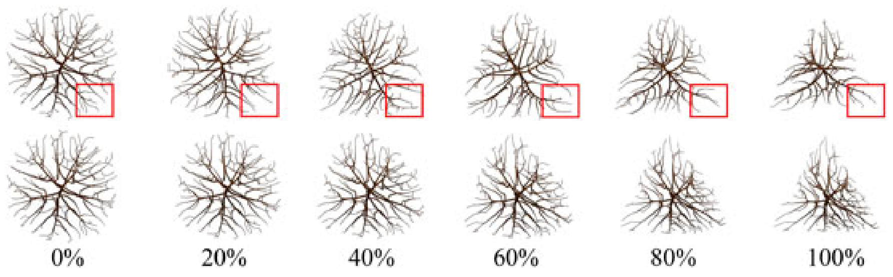


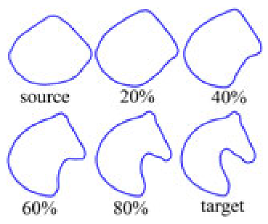
**FIGURE 11** Comparison between results generated by our method (bottom row) and by the space colonization algorithm[10] (top row). The latter produces discontinuous tree branches between frames (marked in red) due to a lack of preservation of branches' topological consistencies

competitions between branches. Although targeting on generating static trees, the method of Runions et al.[10] demonstrates the possibility of generating a shape-preserving animation of tree transformation from one crown shape to another when a sequence of consistently transformed crown shapes is provided. Unfortunately, it failed to create a smoothly transformed tree sequence and generated discontinued branches (circled in red), as shown in Figure 11 (top row).

The reason is that the algorithm in the work of Runions et al.[10] must regenerate a new tree whenever the crown changes. Even with smoothly transformed crowns, the procedural nature of the algorithm cannot guarantee the generation of topologically consistent tree branches between consecutive frames, leading to unsuccessful animation. In contrast, our algorithm (the bottom row in Figure 11) avoids regenerating trees for each frame. Instead, it gradually deforms the crown and hierarchically propagates the crown's transformation to tree branches based on the self-adaptive *MCT*, therefore generating more visually pleasing animation.

**Comparison with topologically consistent morphing**. We compared our work with the most recent and relevant work by Wang et al.,[7] which is capable of generating fluid morphing effects between two leafy trees. As shown in the bottom row of Figure 9, our method demonstrates the equal capability of preserving branches' topological consistency when transforming a tree into a new shape, but exhibits an evident advantage over the preservation of the meaningful in-between crown shapes. In addition, our algorithm only takes a leafy tree and a designer's sketch to achieve a visual effect, whereas the work of Wang et al.[7] requires at least two leafy trees to create a morphing animation.

In addition to visual comparisons, we quantify the shape preservation of the resulting tree sequences by measuring the similarities between the in-between crowns' silhouettes and the morphologically meaningful reference silhouettes (see inset), which are achieved by linearly interpolating the source and the target crown silhouettes. The similarities between two crown silhouettes is measured by the discrete *Fréchet* distance,[27] and the smaller the distance is, the greater the similarity is. Figure 12 plots the *Fréchet* distances of both methods against time during the animation. Evidently, our method (red plot) demonstrates smaller distances to the reference silhouettes, implying greater similarities to the morphologically meaningful in-between crown shapes compared to the work of Wang et al.[7] (blue plot).



The main reason is that the work of Wang et al.[7] directly transforms tree branches based on topology-aware correspondences but does not put explicit efforts into maintaining the morphological meanings of the transformed crown shapes. In contrast, out method interpolates the crown shapes during the animation and hierarchically propagates the transformation of crowns to tree branches based on the *MCT*. Therefore, not only the topological hierarchy but also the morphologically consistent and meaningful in-between crown shapes are preserved by our algorithm.

**Comparison with single-cage transformation**. In Figure 13, we compared our self-adaptive multiscale transformation (Figure 13d) with the single-cage-based global transformation (Figure 13c). It is clear that our method creates better results because of the locally transformed branches based on the self-adaptive *MCT*. Another noteworthy feature of our algorithm is the preservation of the tree's natural branches and foliage. This can be explained by the

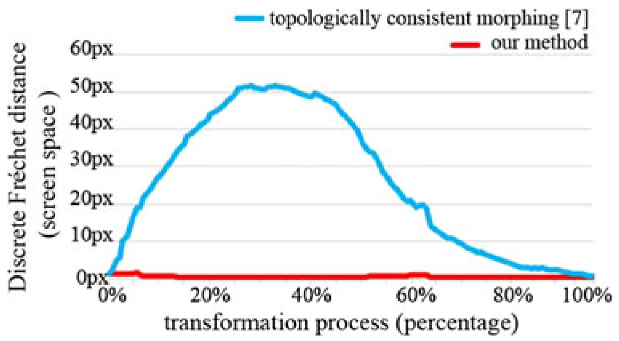


**FIGURE 12** Comparison of the discrete Fréchet distance between our method and topologically consistent morphing[7]

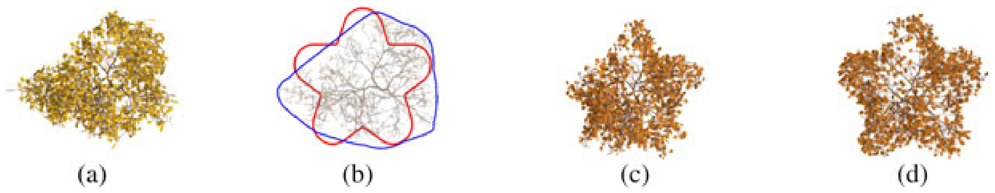


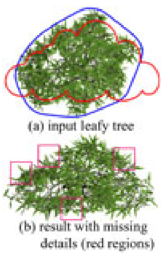
**FIGURE 13** Comparison with the single-cage-based global transformation. (a) The input. (b) The crown silhouette (blue) and the designer's sketch (red). (c) The result by the single-cage-based global transformation. (d) The result by our method

multiscale skeleton-lobe representation, which provides an efficient way to reconstruct botanically meaningful trees from transformed skeletons, radii, and lobes.[7]

## 7 | CONCLUSION

Aiming at generating a new type of visual effect, we have presented a novel and intuitive sketch-based tree animation method. The method requires a leafy tree and a designer's sketch as input and smoothly transforms the tree into the custom crown shape. Both topologically consistent tree branches and morphological meaningful in-between crowns are maintained by the *MCT*. Local deformations are supported by the self-adaptive multiscale cages, resulting in aesthetically pleasing results. In addition, our method could be potentially integrated into existing sketch-based tree modeling frameworks, such as those in the works of Wither et al.,[14] Chen et al.,[14] and Longay et al.,[16] as an efficient and convenient way to generate special effects for leafy trees.

**Limitations and future work**. Our method still leaves room for improvement. Because of the cage-based transformation strategy, the results approximate to the designers' sketches by conforming to the dominant features and ignoring the subtle details (inset). In addition, since we employ a view-dependent sketch to edit a tree's crown shape, due to the lack of 3D information, the crowns of the resulting trees only transform meaningfully when observed from the viewpoint from which they are modified. Despite this, thanks to the multiscale skeleton-lobe representation, the transformed trees remain to be botanically meaningful when viewed from other viewpoints (see the top view results shown in Figure 2d). In the future, we plan to improve the crown editing operations by introducing direct 3D controls and make the results meaningful under multiple viewpoints. Another promising future direction is to extend the current framework by generating shape-preserving animations between two topologically different trees as in the work of Wang et al.[7]



### ACKNOWLEDGEMENTS

The work of Xiaogang Jin was supported by the National Key R&D Program of China (2017YFB1002600), the Key Research and Development Program of Zhejiang Province (2018C01090), and the National Natural Science Foundation of China (61732015).

### ORCID

*Yutong Wang* http://orcid.org/0000-0002-8686-6598
*Xiaogang Jin* http://orcid.org/0000-0001-7339-2920

### REFERENCES

1. Lintermann B, Deussen O. Interactive modeling of plants. IEEE Comput Graph Appl. 1999;19(1):56-65.
2. Pirk S, Niese T, Deussen O, Neubert B. Capturing and animating the morphogenesis of polygonal tree models. ACM Trans Graph. 2012;31(6):169.
3. Pirk S, Stava O, Kratt J, et al. Plastic trees: interactive self-adapting botanical tree models. ACM Trans Graph. 2012;31(4):1-10.
4. Pirk S, Niese T, Hädrich T, Benes B, Deussen O. Windy trees: computing stress response for developmental tree models. ACM Trans Graph. 2014;33(6):204.
5. Shek A, Lacewell D, Selle A, Teece D, Thompson T. Art-directing Disney's tangled procedural trees. ACM SIGGRAPH 2010 Talks. 2010;53.
6. Wang Y, Xue X, Jin X, Deng Z. Creative virtual tree modeling through hierarchical topology-preserving blending. IEEE Trans Vis Comput Graph. 2017;23(12):2521-2534.
7. Wang Y, Wang L, Deng Z, Jin X. Topologically consistent leafy tree morphing. Comput Animat and Virtual Worlds. 2017;28(3–4).
8. Wang G, Laga H, Xie N, Jia J, Tabia H. The shape space of 3D botanical tree models. ACM Trans Graph. 2018;37(1):1-18.
9. Prusinkiewicz P, James M, Měch R. Synthetic topiary. Paper presented at: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '94); 1994 Jul 24–29; Orlando, FL. New York, NY: ACM; 1994. p. 351-358.
10. Runions A, Lane B, Prusinkiewicz P. Modeling trees with a space colonization algorithm. Paper presented at: Proceedings of Eurographics Workshop on Natural Phenomena; 2007; Prague, Czech Republic. Aire-la-Ville, Switzerland: Eurographics Association; 2007. p. 63-70.
11. Wang R, Yang Y, Zhang H, Bao H. Variational tree synthesis. Comput Graph Forum. 2014;33(8):82-94.
12. Stava O, Pirk S, Kratt J, et al. Inverse procedural modelling of trees. Comput Graph Forum. 2014;33(6):118-131.

13. Okabe M, Owada S, Igarash T. Interactive design of botanical trees using freehand sketches and example-based editing. Comput Graph Forum. 2005;24(3):487-496.
14. Wither J, Boudon F, Cani M-P, Godin C. Structure from silhouettes: a new paradigm for fast sketch-based design of trees. Comput Graph Forum. 2009;28(2):541-550.
15. Chen X, Neubert B, Xu YQ, Deussen O, Kang SB. Sketch-based tree modeling using Markov random field. ACM Trans Graph. 2008;27(5).
16. Longay S, Runions A, Boudon F, Prusinkiewicz P. Treesketch: interactive procedural modeling of trees on a tablet. Paper presented at: Proceedings of the International Symposium on Sketch-based Interfaces and Modeling; 2012 Jun 4–6; Annecy, France. Aire-la-Ville, Switzerland: Eurographics Association. p. 107-120.
17. Prusinkiewicz P. Modeling plant growth and development. Curr Opin Plant Biol. 2004;7(1):79-83.
18. Zhao Y, Barbič J. Interactive authoring of simulation-ready plants. ACM Trans Graph. 2013;32(4):84.
19. Liu S, An T, Gong Z, Hagiwara I. Physically based simulation of solid objects burning. In: Transactions on Edutainment VII. Berlin, Germany: Springer; 2012.
20. Pirk S, Jarząbek M, Hädrich T, Michels DL, Palubicki W. Interactive wood combustion for botanical tree models. ACM Trans Graph. 2017;36(6):197.
21. Palubicki W, Horel K, Longay S, et al. Self-organizing tree models for image synthesis. ACM Trans Graph. 2009;28(3):58.
22. Dirr MA. Manual of woody landscape plants: their identification, ornamental characteristics, culture, propagation and uses. Champaign, IL: Stipes Publishing Co; 1990.
23. Hertzmann A. Introduction to 3D non-photorealistic rendering: silhouettes and outlines. Paper presented at: SIGGRAPH 99: Non-Photorealistic Rendering; 1999 Aug 8–13; Los Angeles, CA. ACM SIGGRAPH; 1999.
24. Nealen A, Sorkine O, Alexa M, Cohen-Or D. A sketch-based interface for detail-preserving mesh editing. ACM Trans Graph. 2005;24(3):1142-1147.
25. Ju T, Schaefer S, Warren J. Mean value coordinates for closed triangular meshes. ACM Trans Graph. 2005;24(3):561-566.
26. Vatti BR. A generic solution to polygon clipping. Commun ACM. 1992;35(7):56-63.
27. Eiter T, Mannila H. Computing discrete Fréchet distance. Vienna, Austria: Department of Information Systems, Technical University of Vienna; 1994. Technical report. CD-TR 94/64.

**Yutong Wang** received her BSc degree in software engineering from Chongqing University, China, in 2012. She is currently a PhD candidate at the State Key Lab of CAD&CG, Zhejiang University. Her main research interests include creative modeling, sketch-based modeling, and computer animation.

**Luyuan Wang** received her BSc degree in digital media from Hunan University, China, in 2016. She is currently a PhD candidate at the State Key Lab of CAD&CG, Zhejiang University. Her main research interests include creative modeling and virtual try-on.

**Zhigang Deng** is currently a Full Professor of Computer Science at the University of Houston (UH) and the Founding Director of the UH Computer Graphics and Interactive Media (CGIM) Lab. His research interests include computer graphics, computer animation, virtual human modeling and animation, and human–computer interaction. He earned his PhD degree in computer science from the Department of Computer Science at the University of Southern California in 2006. Prior that, he also completed his BS degree in mathematics from Xiamen University, China, and his MS degree in computer science from Peking University, China. He was the recipient of a number of awards, including the ACM ICMI Ten-Year Technical Impact Award, the UH Teaching Excellence Award, the Google Faculty Research Award, the UHCS Faculty Academic Excellence Award, and the NSFC Overseas and Hong Kong/Macau Young Scholars Collaborative Research Award. Besides the CASA 2014 Conference General Co-chair and SCA 2015 Conference General Co-chair, he currently serves as an Associate Editor for several journals, including Computer Graphics Forum and the Computer Animation and Virtual Worlds Journal. He is a Senior Member of the ACM and a Senior Member of the IEEE.

**Xiaogang Jin** is a Professor of the State Key Lab of CAD&CG, Zhejiang University, China. He received his BSc degree in computer science in 1989 and his MSc and PhD degrees in applied mathematics in 1992 and 1995, respectively, all from Zhejiang University. His current research interests include traffic simulation, insect swarm simulation, physically based animation, cloth animation, special effects simulation, implicit surface computing, non-photorealistic rendering, computer-generated marbling, and digital geometry processing. He received an ACM Recognition of Service Award in 2015 and the Best Paper Award from CASA 2017. He is a member of the IEEE and the ACM.

## SUPPORTING INFORMATION

Additional supporting information may be found online in the Supporting Information section at the end of the article.

**How to cite this article:** Wang Y, Wang L, Deng Z, Jin X. Sketch-based shape-preserving tree animations. *Comput Anim Virtual Worlds*. 2018;e1821. https://doi.org/10.1002/cav.1821