Single Image–Based 3D Tree and Growth Models Reconstruction

Jaehwan Kim and Il-Kwon Jeong

In this paper, we present a new, easy-to-generate system that is capable of creating virtual 3D tree models and simulating a variety of growth processes of a tree from a single, real tree image. We not only construct various tree models with the same trunk through our proposed digital image matting method and skeleton-based abstraction of branches, but we also animate the visual growth of the constructed 3D tree model through usage of the branch age information combined with a scaling factor. To control the simulation of a tree growth process, we consider tree-growing attributes, such as branching orders, branch width, tree size, and branch self-bending effect, at the same time. Other invisible branches and leaves are automatically attached to the tree by employing parametric branch libraries under the conventional procedural assumption of structure having a local self-similarity. Simulations with a real image confirm that our system makes it possible to achieve realistic tree models and growth processes with ease.

Keywords: Virtual 3D tree modeling, growth simulation, digital image matting, skeletonization.

I. Introduction

Realistic and natural 3D tree model construction and growth simulation have been an important goal in a variety of areas such as computer vision, graphics, and animations. However, the creation of realistic 3D tree models, as well as the interpretation of tree growth from existing tree models, is one of the most complicated and challenging tasks in the fields of botany and graphics, because of its inherent geometric complexity and ill-posedness. For these reasons, computer representations of trees need considerable efforts to achieve high levels of realism. With regard to creating 3D tree models, there are three well-known approaches: rule-based modeling, user-interactive modeling, and image-based modeling [1]. The conventional rule-based approaches, such as L-system [2], [3] or procedural generation [4]–[7], create recursive structured trees by introducing pre-defined rules or randomized parameters as a function of the position of the branches within a tree. These rule-based methods have been widely used in many areas as a general framework for tree development, because they are able to generate various complex tree structures from a small set of parameters. However, these methods have difficulties in modeling trees of a distinct shape specified by a user, as well as controlling shape parameters. For instance, creating a variety of similar tree models, each with the same trunk, is really difficult as any slight change to the rules or parameters of the growth model has adverse knock-on effects to the outcome of the final shape. User-interactive approaches [8], [9] generate a 3D tree model from a user's 2D sketches. In such a method, the creation and addition or omission of branches, can be considered as part of a natural process in the creation of tree shapes. Although such sketch-based methods are a more intuitive way of creating tree models than rule-

Manuscript received Apr. 3, 2013; revised Oct. 17, 2013; accepted Nov. 13, 2013.

This work was supported by the Ministry of Science, ICT and Future Planning (MSIP) and the Korea Creative Content Agency (KOCCA) in the Culture Technology (CT) Research & Development Program 2011 (2011A5020021098502001).

Jaehwan Kim (corresponding author, phone: +82 42 860 1070, jh.kim@etri.re.kr) is with the SW·Content Research Laboratory, Daejeon, ETRI, Rep. of Korea.

Il-Kwon Jeong (jik@etri.re.kr) is with the SW·Content Research Laboratory, ETRI, Daejeon, Rep. of Korea.

based methods, the approach relies heavily upon a user's extensive and skillful intervention in assisting with the creation of a complex tree. The image-based approaches [1], [10]–[12] construct a 3D tree model from one or several given images and although this generally provides an improved visual realism, it does not animate the visual growth of the constructed 3D tree model. In fact, there are some commercial products such as 'Speedtree' [13] and 'Xfrog' [14], which are readily available off-the-shelf software tools for generating tree models, which have adopted a sort of hybrid approach by combining rule-based and user-interactive methods.

Motivated by existing works [11], in this paper we introduce a new system, whereby it is possible to generate natural-looking 3D tree models using only a single, real tree image as input data. Furthermore, the system is capable of simulating a variety of growth processes from the constructed tree model by utilizing our proposed measurement of the growth age. In our system, we focus on the more important tasks of extraction and abstraction of the tree trunk object, as their intermediate results directly affect the final structure of the model. To precisely extract the visible branch structure from the input image, we propose improved methods of current solutions [15], [16]—namely, normalized matting and user-interactive skeletonization, respectively.

This paper is organized as follows: section II handles technical details of our system, section III validates the proposed system by performing several experiments, and finally, we conclude in section IV with summarizing the main contributions.

II. System Details

We begin with revisiting the digital image matting method and introducing our improved solution. Then, we illustrate how to construct the skeleton of a tree, how to apply a fractal starting from the obtained skeleton structure, and finally, how to simulate growth processes while taking the self-bending effect of the branches during growth into account from the existing tree model. In Fig. 1, our system flow is briefly illustrated.

1. Visible Trunk Extraction

As a first step, the tree input image is divided into the visible branch part and the other pixel colors. The visible branches can be extracted using the digital image matting method, where the goal of this method is to estimate the opacity (alpha matte) [15], [17], [18], under the assumption that each pixel value is a linear combination of the corresponding fore and background colors. In [15], Levin and others proposed the closed-form solution to

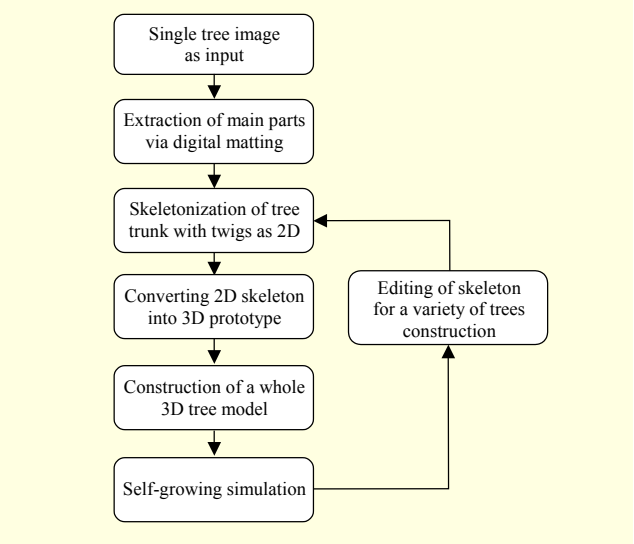

Fig. 1. Overview of our system.

define a quadratic cost function from the intrinsically under-constrained matting problem, under two important assumptions: fore and background color elements are approximately constant over a local window around each pixel, and each of the fore and background regions are a mixture of colors. As a result, the constrained quadratic cost function can be expressed as follows:

$$\begin{aligned} &\text{argmin}_{\boldsymbol{\alpha}}\ \boldsymbol{\alpha}^{\top}\boldsymbol{L}\boldsymbol{\alpha}, \\ &\text{s.t.}\ \ \boldsymbol{C}_s\boldsymbol{\alpha}=\boldsymbol{h}_s,\ 0\preceq\boldsymbol{\alpha}\preceq 1,\ \ \forall s\in S, \end{aligned} \tag{1}$$

where $\boldsymbol{\alpha}=[\alpha_1\cdots\alpha_N]^{\top}$ ($\alpha_i\in[0,1]$ represents a pixel's foreground opacity) and $\boldsymbol{L}$ is a matting Laplacian. This Laplacian can also be written as $\boldsymbol{L}=\boldsymbol{D}-\boldsymbol{A}$, where $\boldsymbol{A}$ is referred to as a matting affinity and $\boldsymbol{D}$ is a diagonal matrix that consists of diagonal elements $\boldsymbol{D}_{ii}=\sum_j \boldsymbol{A}_{ij}$. The set of constrained pixels is S, the diagonal matrix whose diagonal elements consist of 1's for constrained pixels and 0's for all other pixels is $\boldsymbol{C}_s$, and $\boldsymbol{h}_s$ is the vector containing pre-defined alpha values for the constrained pixels and zeros for all others. Levin and others obtain an optimal alpha matte solution through minimizing the Lagrangian of (1). However, as addressed in [15], the closed-form solution to image matting has a tendency to assign some erroneous non-opaque alpha values to subregions located far away from any constraints. This occurs when using small windows on a fine-resolution image or when being given insufficient constraints on the fore and background regions (see Fig. 2). Meanwhile, partitioning a graph with its edge weights assigned by pairwise similarities, serves as an important tool for data clustering (for example, image segmentation). The digital image matting problem can be considered as the classical binary partitioning (bi-partitioning)

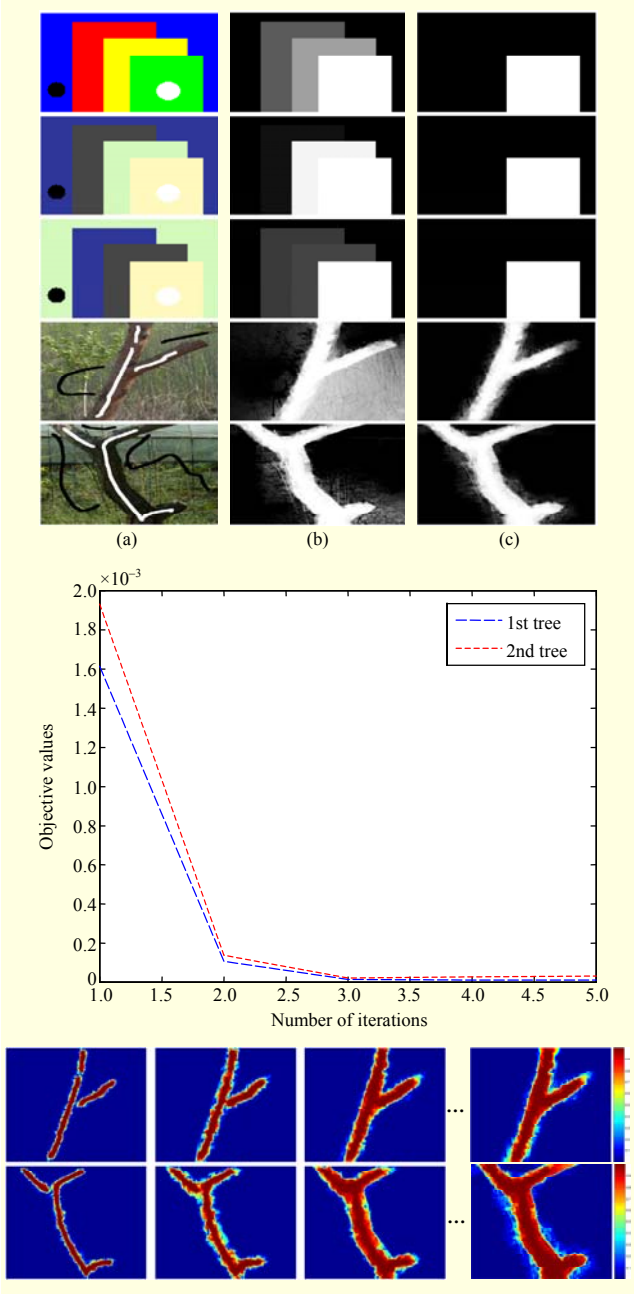

Fig. 2. Top: results of applying two different matting methods to sample images (toy and real tree images) are shown (using 3 × 3 window). (a) Input images (toy images of size 150 × 100 and tree images of size 420 × 400): white and black colors represent fore and background constraints, respectively. (b) Alpha matte using the closed-form solution (we used the λ from [15] = 100). (c) alpha matte using the proposed method. Mid & Bottom: objective values for two tree image data when applying our method, and some intermediate alpha mattes of two real tree images according to convergence level.

on the fore and background regions. Moreover, normalized cuts [19] is a method to deal with a graph partitioning problem in an efficient way by introducing a normalized cut criterion. In a graph theoretic term, the closed-form solution to image matting criterion (1) can be written as a total dissimilarity between the fore and background subregions, which is given by

$$\boldsymbol{\alpha}^{\top}\boldsymbol{L}\boldsymbol{\alpha} = \sum_{i\in F}\boldsymbol{D}_{ii} - \sum_{i\in F}\sum_{j\in F}\boldsymbol{A}_{ij} = \sum_{i\in F}\sum_{j\in B}\boldsymbol{A}_{ij}, \tag{2}$$

where F and B indicate the set of pixels of the fore and background regions, respectively. As formulated in (2), currently the existing closed-form approach only utilizes the sum of weights of connections across the distinct subregions (fore and background). To improve the performance (that is, alpha matte quality) of the closed-form solution, we propose a new criterion that is derived by way of incorporating a normalized term of the foreground region into the current closed-form solution's objective. By taking the normalized factor into account, we can not only minimize the sum of weights of connections across the different fore and background regions, but also maximize the cohesion of alpha matte within the foreground region at the same time. Then, the normalized matting criterion can be determined by

$$\begin{aligned} &\text{argmin}_{\boldsymbol{\alpha}} \frac{\boldsymbol{\alpha}^{\top}(\boldsymbol{D}-\boldsymbol{A})\boldsymbol{\alpha}}{\boldsymbol{\alpha}^{\top}\boldsymbol{D}\boldsymbol{\alpha}}, \\ &\text{s.t. } \boldsymbol{C}_s\boldsymbol{\alpha} = \boldsymbol{h}_s,\ 0 \preceq \boldsymbol{\alpha} \preceq 1,\ \forall s \in S. \end{aligned} \tag{3}$$

Note that only the volume of F in the denominator of the objective function is considered. To obtain an optimal matte result from our new criterion, we employ a global optimization technique. Motivated by the existing work in [20], we use a quadratic programming after getting its quadratic one. Defining

$$\boldsymbol{W} = \boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2},\quad \boldsymbol{q} = (\boldsymbol{\alpha}^{\top}\boldsymbol{D}\boldsymbol{\alpha})^{1/2},\quad \text{and}\quad \boldsymbol{w} = (\frac{1}{\boldsymbol{q}}\boldsymbol{D}^{1/2}\boldsymbol{\alpha})$$

the normalized matting problem (3) is equivalent to another constrained quadratic problem given by

$$\begin{aligned} &\text{argmin}_{\boldsymbol{w}}\ \boldsymbol{w}^{\top}\boldsymbol{W}\boldsymbol{w}, \\ &\text{s.t. } \boldsymbol{q}\boldsymbol{C}_s(\boldsymbol{D}^{-1/2})\boldsymbol{w} = \boldsymbol{h}_s,\ 0 \preceq \boldsymbol{w} \preceq \frac{1}{\boldsymbol{q}}\text{diag}(\boldsymbol{D}^{1/2}),\ \forall s \in S. \end{aligned} \tag{4}$$

If the objective function is convex quadratic and the constraint functions are affine, the convex optimization problem provides a globally optimal solution [20]. We use the sequential quadratic programming method for the optimization problem (4) with the non-convex constraint that $\boldsymbol{q}$ depends on $\boldsymbol{\alpha}$ (Here, we used Matlab's function, fmincon).

As shown in Fig. 2, our scheme helps the foreground constraint to propagate to the only foreground-like color region, strictly under given insufficient constraints on the fore and background regions. In Fig. 2 and Table 1, five matting results of applying the two methods (closed-form solver and our method) to both toy and natural images are shown. From

Table 1. Performance of digital image with data images of Fig. 2.

	CF solver $\alpha^{\top} L\alpha$	Our method $w^{\top} Ww$
1st row data	1.395E-005	1.000E-009
2nd row data	3.739E-005	2.600E-008
3rd row data	4.541E-004	1.750E-008
4th row data	3.179E-001	3.428E-006
5th row data	7.198E-001	3.309E-006

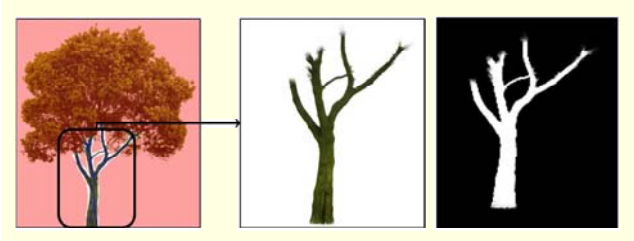

Fig. 3. Extracted tree trunk result from given image and its alpha matte.

several experimental results, we can notice that the performance of our method is more stable and prominent. Such a desirable result is due to the consideration of adding a foreground normalized factor. We incorporate our proposed digital image matting method into our system for extracting the visible branch structure (that is, the tree trunk) from the input image (see Fig. 3).

2. Trunk Skeletonization

Once the visible branches, consisting of the tree trunk with twigs, have been extracted from the given image through our proposed matting procedure, we next perform a skeleton-based abstraction of the branch object. The goal of skeletonization is to extract skeletons that are in accordance with human visual perception and to preserve the topological information of the original object, even in the presence of boundary deformations and protrusions. In our system, we try to construct tree skeletons that are as plausible as possible, so as to create a realistic virtual 3D tree model and its growth processes. However, for a complex tree branch consisting of lots of deformations and noise-like protrusions, it is quite difficult to obtain a precise skeleton—since noise leads to redundant or shortened skeleton branches (see red circles in Fig. 4 (Top)). To overcome such instability of the conventional skeletonizations, we employ a skeleton pruning method proposed by [16]. Most skeletonization algorithms consider only the local significance of skeleton points and therefore cannot guarantee to preserve the original topology. As shown in Fig. 4 (Top), the conventional skeletonization methods fail to obtain a precise skeleton from a tree-shaped image. The main idea of the current skeleton pruning method is to preserve the topology based on a contour partition into curve segments, which makes it possible to extract an exact skeleton of the global topology.

The discrete curve convolution proposed by [21], can reduce the boundary noise and simplify the contour without changing relevant shape features so that the accuracy of the skeleton position with one that has a similar perceptual appearance is guaranteed. The discrete curve evolution of polygons is constructed in the following way [21], [22], [16]: In every evolutional step, a pair of consecutive line segments s_1, s_2 is replaced by a single line segment joining the endpoints of $s_1 \cup s_2$. The key property of this evolution is the order of the substitution. The substitution is achieved according to a relevance measure R, given by $R(s_1, s_2) = (\beta(s_1, s_2) l(s_1) l(s_2)) / (l(s_1) + l(s_2))$, where $\beta(s_1, s_2)$ is the turn angle at the common vertex of segments s_1 and s_2 and where l is the length function normalized with respect to the total length of a polygonal curve. The main property of the relevance measure is that the higher the value of $R(s_1, s_2)$, the greater the contribution to the shape of the curve of arc $s_1 \cup s_2$. Given the input boundary polygon P, with n vertices, the discrete curve evolution produces a sequence of simpler polygons P^{n-k} $(0 \le k \le n-3)$ by removing a single vertex $v \in P^{n-k} - P^{n-(k+1)}$ from the less-evolved polygon whose

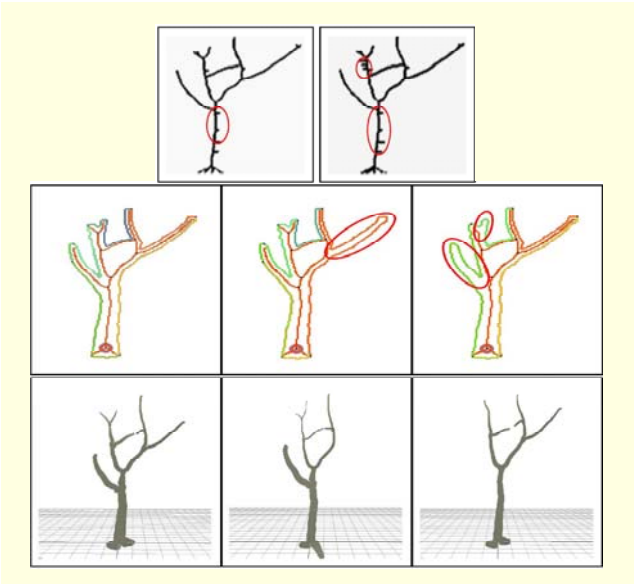

Fig. 4. Top: two conventional skeletonizaiton methods: Gonzalez's (left) & Zhang's (right) thinning results [23]. Mid & Bottom: differently segmented contours (different parts marked as circles), final skeletons (inner red line) obtained through our user-interactive skeletonization method and their constructed 3D tree models.

shape contribution measured by R is the smallest. In [16], they incorporated the discrete curve evolution into their skeleton pruning method to remove all skeleton points whose boundary points tangential to their maximal circle, lie on the same contour segment.

Algorithm 1. Algorithm of user-interactive skeletonization.

Require : an alpha matte & the number of vertices m
1 : Extract the input boundary polygon P with n vertices
2 : **while** final # vertices ≥ m vertices **do**
3 : Evaluate and sort the relevance measure R
4 : **if** R is the smallest value **then**
5 : remove the corresponding vertex
6 : join the two consecutive end points
7 : **else**
8 : remain the vertex as a shape contributor
9 : allow user-interaction in case of need
10 : (i.e., remove or create a vertex by user-selecting)
11 : **end if**
12 : **end while**

In our system, to design various user-wanted skeleton shapes, we propose a user-interactive skeletonization method that provides direct control (that is, merging two differently segmented contours into one, so that some protrudent sub-branches, such as noise, can be easily removed) of the segmented contours obtained from the discrete curve evolution (see Algorithm. 1). As shown in Fig. 4, this makes it possible to achieve both precision and controllability in skeletonization.

3. 3D Tree Modeling and Growth

After obtaining the tree branch skeleton, we tackle the skeleton as an undirected acyclic graph. For converting into a graph, we represent a connected graph $G(V, E)$ (where V and E denote a set of vertices and a set of edges, respectively) by taking each skeleton point x_i as a node and connecting to its neighborhood with edges.

In our system, to simulate the visual growth of the input tree in a natural-looking way, we take the branch age information combined with a scaling factor and the self-bending effect of the branches into account. We define a measurement of the i node's growth age $T(i)$—which is in inversely proportional to the growth order—and a scaling control value of the whole tree size $M(j)$, based on the pre-defined assumptions: (a) the new branch is the farthest away from a tree root on the branches' manifold; (b) any branch that is thinner than others around it, is more likely to be a branch of recent growth; and (c) the self-bending is caused by the weight of the child branches during growth. The measurement can be expressed as follows:

$$T(i \neq r) = \left\| x_i - x_r \right\|_{geo}^{-1} + w_p * EDT(x_i),$$
$$M(j \neq r) \propto T(j)^{-1}, \text{ s.t. } 0 < M(\bullet) \leq 1, \tag{5}$$

where j is an index of the latest branching node, x_r is a tree root node, and $EDT(x_i)$ is a Euclidean distance transform of the binary object F that assigns to each node $x_i \in F$ its distance $\min_{b \in F^c} \| x_i - b \|$ to the background F^c. The symbol w_p is a weight value (we generally used w_p[1] = 0.001). The geodesic distance (Dijkstra) on a neighborhood graph well reflects the local structure [24] (see Fig. 5), so we employ the geodesic distance when computing the i node's growth age $T(i)$. The computed growth age $T(i)$, specifies how much faster or slower any sub-branches should be formed compared with others being sorted (from the root node to any apical nodes). The whole tree growth process can be achieved by using the growth age value. In Fig. 6, 3D growth model examples of the initial tree trunk of Fig. 4 are shown. After finding out branch-wise growth rates to control the simulation of tree growth

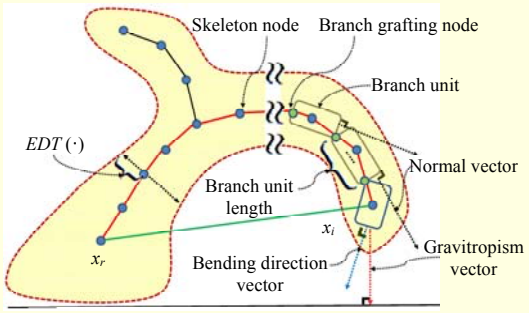

Fig. 5. Tree structure consisting of skeleton nodes and branch units. Euclidean distance between nodes x_r and x_i, $\left\| x_i - x_r \right\|$, is denoted by a green-colored line on a space of skeleton nodes. Dijkstra geodesic distance, $\left\| x_i - x_r \right\|_{geo}$, is computed along shortest path (red-colored path) between x_r and x_i.

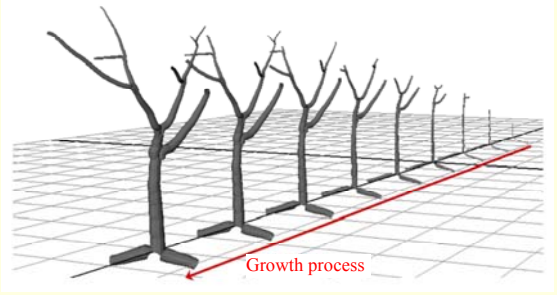

Fig. 6. For initial tree trunk in Fig. 4, 3D growth model examples generated according to our growth process.

1) Growth ages of some nodes can be changed according to the weight value, which leads to various intermediate growth models. In here, we regard the branch manifold structure as more important factor than the branch thickness when expressing growth.

http://dx.doi.org/10.4218/etrij.14.0113.0069

processes, the 2D skeletons are further extended to 3D according to Gaussian distribution and directions of the sub-branches, while their orthogonal projection in the input image remains unchanged. Our goal is to just construct a 3D tree model that is as plausible as possible when viewed from different viewpoints. Converting the 2D skeleton graph into a 3D model is represented by a geometric transformation based on the following assumptions: for a parent branch having k sub-branches, all distinguished sub-branches are rotated around the vertical axis (that is, y-axis) of the parent branch coordinate.

The rotation angles $\psi_i (0 \le i \le k)$, have Gaussian distributions $N(\mu_{\psi_i} + (2\pi(k-i)/k), \sigma_{\psi_i})$ so that we not only express a spatial variance of sub-branches, but also adjust the distance between the lateral sub-branches to be as large as possible.

Additionally, to create the whole tree model in a natural-looking way, we add invisible branches to the tree by employing parametric branch libraries where the parameters (that is, three rotation angles around the x-, y-, and z-axis and the scaling factor on the parent-child branches) are estimated based on the shape of the initial skeleton 3D model. Then, we utilize the inferred parameters with some randomness, under the assumption that a branch can be regarded as any subset of the whole tree; that is, the structure has a local self-similarity. The tree extends its branches by grafting new sub-branch units to the existing branch tips and sides. With these parametric sub-branch units, three different examples according to the branch propagation levels are illustrated in Fig. 7. The growth order of newly attached sub-branches to the existing tree model is assigned in the same way (5).

Meanwhile, the self-bending caused by the weight of the adjacent child branches is very important for the realism of growth animation. To achieve such a self-bending effect during the tree growth process, we assume newly added branches to be affected by gravitropism, which guides tree growth downward. As illustrated in Fig. 5, a branch is discretized into a string of skeleton nodes and branch units. The branch units are affected from the bending direction $\boldsymbol{\rho}$ in (6); a direction bent due to gravitropism. Each branch unit contains an affine transformation that represents its contribution to the curvature and length of the branch unit. The transformation of the tip-branch unit from global-space to the local-space $\boldsymbol{T}$, can be expressed as a product of the transformation $\boldsymbol{U}_i$, of the mid-branch units from the attached point to the current tip-branch unit. The transformations, $\boldsymbol{T}$ and $\boldsymbol{U}_i$, are defined as follows:

$$\boldsymbol{T} = \prod_{i=0} \boldsymbol{U}_i, \text{ where } \boldsymbol{U}_i = \begin{bmatrix} \boldsymbol{\phi}_x & \boldsymbol{\eta}_x & \boldsymbol{\rho}_x & l_x \\ \boldsymbol{\phi}_y & \boldsymbol{\eta}_y & \boldsymbol{\rho}_y & l_y \\ \boldsymbol{\phi}_z & \boldsymbol{\eta}_z & \boldsymbol{\rho}_z & l_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (6)$$

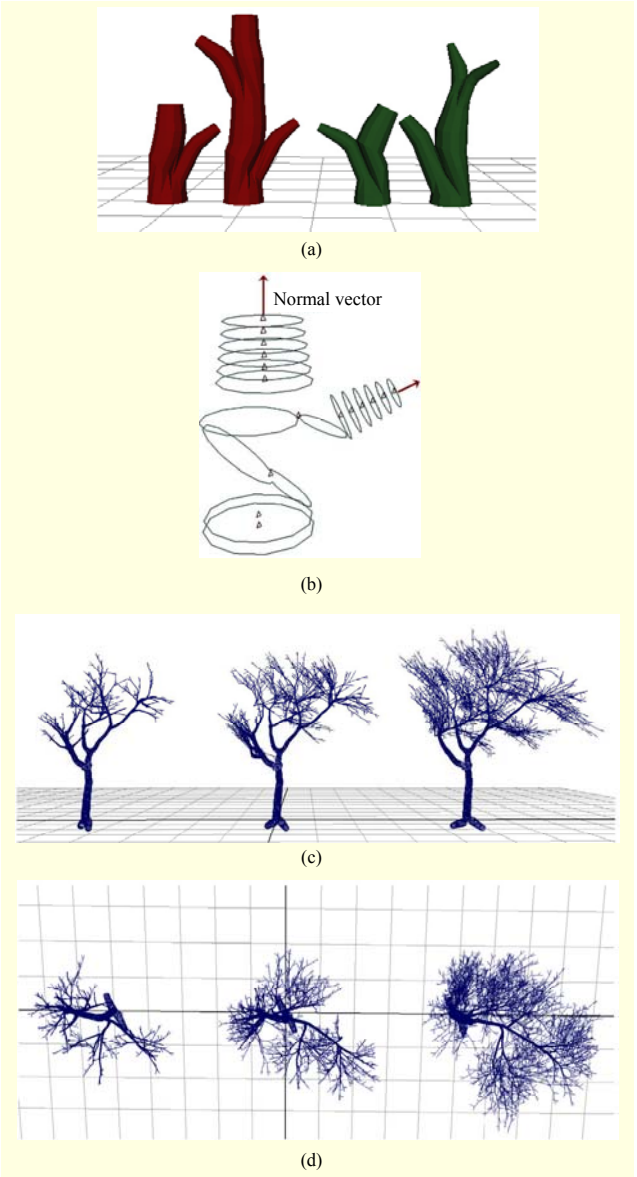

Fig. 7. (a) Four sub-branch units: 'L' (red) and 'Y' (green) types, (b) one unit model: consisting of several skeleton nodes (red triangles), circles (blue) are cross-sections of branches, and normal directions (red arrows) of branches, (c) several examples according to branch propagation levels (2-4 level), and (d) their top views.

$\boldsymbol{\eta}$ is a normal vector of the branch unit, $\boldsymbol{\rho}$ is a bending direction vector, $\boldsymbol{\phi}$ is $\boldsymbol{\rho} \times \boldsymbol{\eta}$, and l is a branch unit length.

The tip-branch unit undergoes rotation due to gravitropism. To rotate the branch unit toward a gravitropism direction, we need to obtain the axis of rotation in local branch space, which is given by

$$\gamma = \boldsymbol{T}(\boldsymbol{T}^{-1} \frac{\boldsymbol{\rho}}{\|\boldsymbol{\rho}\|} \times \boldsymbol{g}), \quad (7)$$

where $\boldsymbol{g}$ is the gravitropism vector. Using the axis of rotation γ

in local branch space, we get the rotational matrix for the rotation of the direction of the tip-branch unit toward the direction of the gravitropism with a strength of δ

$$\boldsymbol{R} = \begin{bmatrix} (\gamma_x)^2 c + a & \gamma_y \gamma_x c + \gamma_z b & \gamma_z \gamma_x c - \gamma_y b & 0 \\ \gamma_x \gamma_y c - \gamma_z b & (\gamma_y)^2 c + a & \gamma_z \gamma_y c + \gamma_x b & 0 \\ \gamma_x \gamma_z c + \gamma_y b & \gamma_y \gamma_z c - \gamma_x b & (\gamma_z)^2 c + a & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad (8)$$

where $a = \cos(\delta)$, $b = \sin(\delta)$, and $c = 1 - \cos(\delta)$. The tip-branch unit, as well as the mid-branch units, are rotated toward the direction of the gravitropism by multiplying $\boldsymbol{R}$ with the local transformation of the branch units. The strengths of the tropisms are chosen empirically.

III. Experiments

In this section, we show the usefulness of our system through modeling a single, real tree image and simulating a variety of growth processes.

In Fig. 8, we show the animation of branches affected by external gravitropism and growth processes, taking the self-bending effect of the branches during growth into account. Gradually bent branches according to the growing level are illustrated in Fig. 8 (Bottom).

Table 2. The number of skeleton nodes, memory size, and tree growth simulation times for the three different results in Fig. 9.

Model	# of skeleton nodes	Memory size of model	# of growth models	Time growth. processes
(a)	21,860	28.8 MB	184	24 sec
(b)	13,348	16.4 MB	165	22 sec
(c)	25,629	30.7 MB	186	24 sec

As shown in example results, our system is able to generate a natural-looking 3D tree trunk model that is very similar to the trunk shape of the input tree image. Results in Fig. 9 show three different growth processes for one constructed final tree model: one is without considering the scaling factor at the left; another takes into account branch thicknesses, tree size, and branching order attributes during growth; and the other considers only the scaling factor at the right.

Our current C++ implementation constructs a single tree model, as well as its growth models, in just a few seconds (see Table. 2). All times are recorded on a 2.9 GHz dual-core Processor with 4 GB of RAM. Our system also provides flexible user control for editing the output tree model by means of sketching (for example, adding and deleting branches) the intermediate result (matte image).

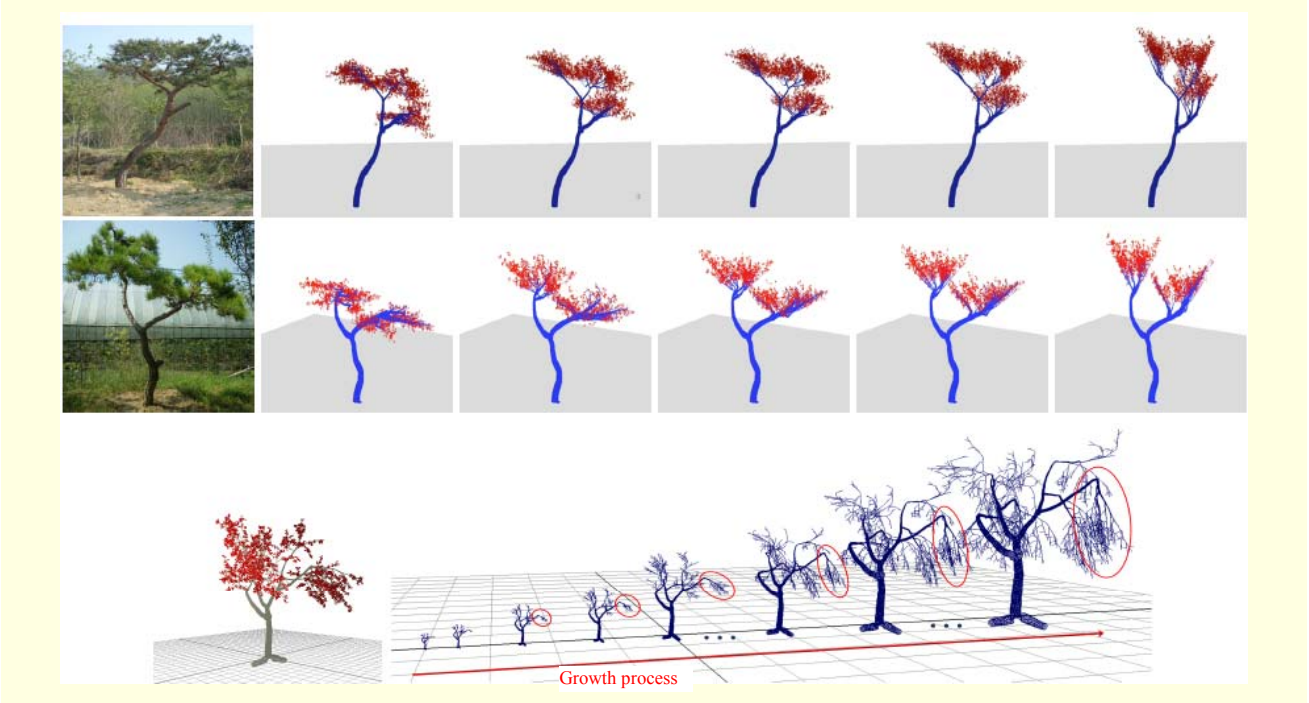

Fig. 8. Top and Middle: two results generated from input images and their simulation results with self-bending effects according to level of gravitropism (from left to right, we used $\delta \approx 20$, 0, –7, –14, and –25). Bottom: an example model and its growth processes with self-bending effects by gravitropism.

http://dx.doi.org/10.4218/etrij.14.0113.0069

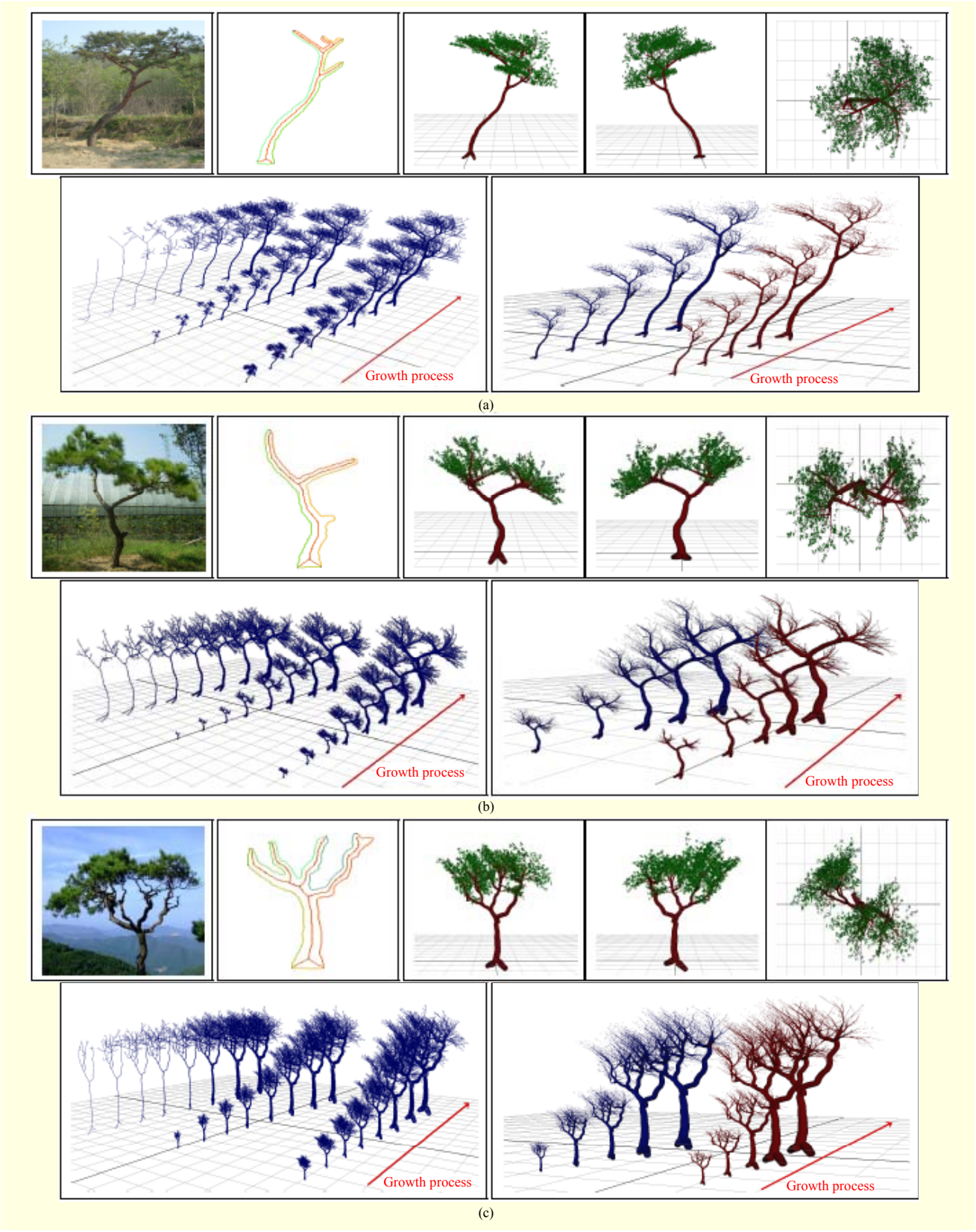

Fig. 9. Several results generated from input images of different trees ((a), (b), and (c)). From left to right: input image, post-(trunk skeleton) extraction, and final tree models from different viewpoints (front, back, and top). Input image's three different growth simulation results and first few magnified results of rightmost (dark blue) and center (dark brown) growth models.

IV. Conclusions and Future Work

We have presented a novel, easy-to-generate system that is capable of creating natural-looking 3D tree models and a variety of growth processes from a single given tree image. Useful aspects of our proposed system could be summarized as follows: (a) our proposed 'normalized matting' and 'user-interactive skeletonization' methods can provide an efficient way of extracting a tree-trunk part from complex backgrounds —with a few constraints (scribbles in here)—and can make a precise skeleton structure. Using these methods, we are able to construct a natural-looking 3D tree trunk model, as well as design more user-wanted 3D tree models. (b) The system is simple, and a variety of growth processes are created with ease by controlling growth-associated attributes (that is, branching orders, branch width, tree size, and branch self-bending factors).

In the future, we plan to improve by creating more realistic tree models and showing comparison results with other competing approaches. For more realistic models, a variety of sub-branch units should also be provided.

References

[1] B. Neubert, T. Franken, and O. Deussen, "Approximate Image-Based Tree-Modeling Using Particle Flows," *J. ACM Trans. Graph.*, vol. 26, no. 3, July 2007, pp. 88:1–88:8.

[2] A. Lindenmayer, "Mathematical Models for Cellular Interactions in Development II. Simple and Branching Filaments with Two-Sided Inputs," *J. Theoretical Biology*, vol. 18, no. 3, Mar. 1968, pp. 300–315.

[3] P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants*, NY: Springer-Verlag, 1990.

[4] P.E. Oppenheimer, "Real Time Design and Animation of Fractal Plants and Trees," *J. ACM Siggraph Comput. Graph.*, vol. 20, no. 4, Aug. 1986, pp. 55–64.

[5] J. Weber and J. Penn, "Creation and Rendering of Realistic Trees," *Proc. Annual Conf. Comput. Graph. Interactive Techn.*, Los Angeles, CA, USA, Aug. 6–11, 1995, pp. 119–128.

[6] P. Przemyslaw et al., "The Use of Positional Information in the Modeling of Plants," *Proc. Comput. Graph. Intractive Tech.*, Los Angeles, CA, USA, Aug. 12–17, 2001, pp. 289–300.

[7] O. Deussen and B. Linatermann, *Digital Design of Nature: Computer Generated Plants and Organics*, Berlin: Springer-Verlag, 2005.

[8] M. Okabe, S. Owada, and T. Igarashi, "Interactive Design of Botanical Trees Using Freehand Sketches and Example-Based Editing," *Proc. Eurographics*, Dublin, Ireland, vol. 24, no. 3, Aug. 29 – Sept. 2, 2005, pp. 487–496.

[9] X. Chen et al., "Sketch-Based Tree Modeling Using Markov Random Field," *J. ACM Trans. Graph.*, vol. 27, no. 5, Dec. 2008, pp. 109:1–109:9.

[10] P. Tan et al., "Image-Based Tree Modeling," *J. ACM Trans. Graph.*, vol. 26, no. 3, July 2007, pp. 87:1–87:7.

[11] P. Tan et al., "Single Image Tree Modeling," *J. ACM Trans. Graph.*, vol. 21, no. 5, Dec. 2008, pp. 108:1–108:7.

[12] C. Li et al., "Modeling and Generating Moving Trees From Video," *J. ACM Trans. Graph.*, vol. 30, no. 6, Dec. 2011, pp. 127:1–127:12.

[13] Interactive Data Visualization Inc., 2000. Accessed Apr. 3, 2013. http://www.speedtree.com

[14] Xfrog Inc., 1996. Accessed Apr. 3, 2013. http://www.xfrog.com

[15] A. Levin, D. Lischinski, and Y. Weiss, "A Closed-Form Solution to Natural Image Matting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, Feb. 2008, pp. 228–242.

[16] X. Bai, L. Latecki, and W. Liu, "Sekeleton Prunning by Contour Partitioning with Discrete Curve Evolution," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 3, Mar. 2007, pp. 449–462.

[17] Y.-Y. Chang et al., "A Bayesian Approach to Digital Matting," *Proc. IEEE Comput. Vision, Pattern Recogn.*, Kauai, HI, USA, vol. 2, Dec. 8–14, 2001, pp. 264–271.

[18] J. Wang and M.F. Cohen, "An Iterative Optimization Approach for Unified Image Segmentation and Matting," *Proc. IEEE Int. Conf. Comput. Vision*, Beijing, China, vol. 2, Oct. 17–20, 2005, pp. 936–943.

[19] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, Aug. 2000, pp. 888–905.

[20] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge, UK: Cambridge University Press, 2004.

[21] L. Latecki and R. Lakamper, "Polygon Evolution by Vertex Deletion," *Proc. Int. Conf. Scale-Space Theories Comput. Vision*, Corfu, Greece, Sept. 26–27, 1999, pp. 398–409.

[22] L. Latecki et al., "Continuity of the Discrete Curve Evolution," *J. Electron. Imag.*, vol. 9, no. 3, July 1, 2000, pp. 317–326.

[23] R. Gonzalez and R.E. Woods, *Digital Image Processing*, Boston, MA: Addison-Wesley Longman, 2001.

[24] J. Kim, K.-H. Shim, and S. Choi, "Soft Geodesic Kernel K-Means," *IEEE Int. Conf. Acoust., Speech, Signal Process.*, Honolulu, HI, USA, vol. 2, Apr. 15–20, 2007, pp. 429–432.

Jaehwan Kim received his BS degree in computer science & engineering from Inha University in 2002, and his MS degree in computer science & engineering from Pohang University of Science and Technology (POSTECH), Pohang, Rep. of Korea, in 2005. He was an engineering staff in the Department of Computer Science & Engineering at POSTECH in 2002 and 2005. He has been a senior researcher at ETRI since 2006. His research interests are in the areas of machine learning and computer vision.

Il-Kwon Jeong received his BS, MS, and PhD degrees in electrical engineering from Korea Advanced Institute of Science and Technology, Daejeon, Rep. of Korea, in 1992, 1994, and 1999, respectively. Since 1999, he has been with the Department of Visual Content Research, ETRI, where he was at first a senior researcher, and then later a director (2008), and a principal researcher (2010). His current research interests include next-generation generation 3D content, computer graphics, and virtual reality.