ExploreTree: Interactive Tree Modeling in Semantic Trait Space with Online Intent Learning

Yinhui Yang, Rui Wang*, Hongxin Zhang, Hujun Bao
State Key Lab of CAD&CG
Zhejiang University
rwang@cad.zju.edu.cn

Abstract

Perceptually modeling realistic trees is important for many graphics applications. However, existing methods are mainly rule-based. Few have directly associated control parameters with user modeling intent and semantic tree shape descriptions. In this paper, we propose a new interactive tree modeling system, ExploreTree, that automatically deduces user modeling intent and supports iteratively design of 3D tree models. It consists of two major phases. The first phase is an off-line learning process, where semantic tree traits perceived by humans are learned. Crowdsourced data on example tree models are collected and analyzed to construct the semantic trait space as well as the embedding of trees into this space. Built upon it, the second phase is an interactive exploration of tree models via a few user clicks, where a user intent evaluation model is learned online to guide the modeling process. Modeled trees and user studies demonstrate the efficiency and capability of ExploreTree.

1. Introduction

Trees are ubiquitous in our physical world, but for their large varieties and natural complexities in forms it is challenging to model them in a realistic way for the virtual environments. In the past few decades, researchers have proposed dozens of methods to make progress in modeling realistic trees for various graphics applications. Most of such methods are focused on designing specific computing mechanisms to create trees with highly visually pleasant results [7]. However, it is also important to design a powerful yet user-friendly interface that can support the ordinary users creating trees intuitively and efficiently, even without concerning the underneath synthesis algorithm or computing mechanism. Some sketch-based methods (e.g. [43, 21]) have already made a good progress in this direction, but further researches for the understanding of user perceptions on tree forms and their modeling intents are still needed and are valuable for designing more efficient systems.

Nowadays, a number of tree modeling systems are mainly designed for trained users or experts. Usually, a significant amount of efforts are required to produce desired good results, either in the sense of tuning tree growing parameters or in the sense of image processing and 3D reconstruction [40, 20]. For example, the parametric tree modeling methods [42, 19] involve a number of parameters to control a variety of effects. By incorporating them with a traditional trial-and-error style of interactions, users have to take a lot of time both on familiarizing themselves with the various parameters and finding the proper parameter combinations to achieve their final models.

To overcome aforementioned problems, in this paper, we propose a novel interactive tree modeling system, called *ExploreTree*. Here, the interactive modeling process is considered as a "conversation" between the system and the user [34]. First, from few strokes drawn by the user to indicate the desired tree shape, our system generates several initial tree candidates. Then, tree modeling is acted in a selection-and-recommendation scheme. From recommended tree models, users first select 'liked' trees that share some similar shape features with the desired one. Then, these selected trees are automatically processed by our system to generate top-k new candidates according to the underlying learnt user intent for a next recommendation. The entire modeling process is exploratory and iteratively, which is simple and convenient especially for novice and casual users. A typical example is demonstrated in Figure 1.

In our system, we use a parametric model derived from variational optimization [41] to generate trees. However, the parametric space of the computational model is not only computation-oriented but also high dimensional. Thus, to directly establish the exploratory interaction, in terms of selection-and-recommendation, on such a representation is awkward. Therefore, in this paper, we introduce a key technique of constructing a semantic trait space to represent users' various perceptions on tree forms based on the crowdsourced data. This tree trait space is acted as an intermediate representation, which bridges the user intent and

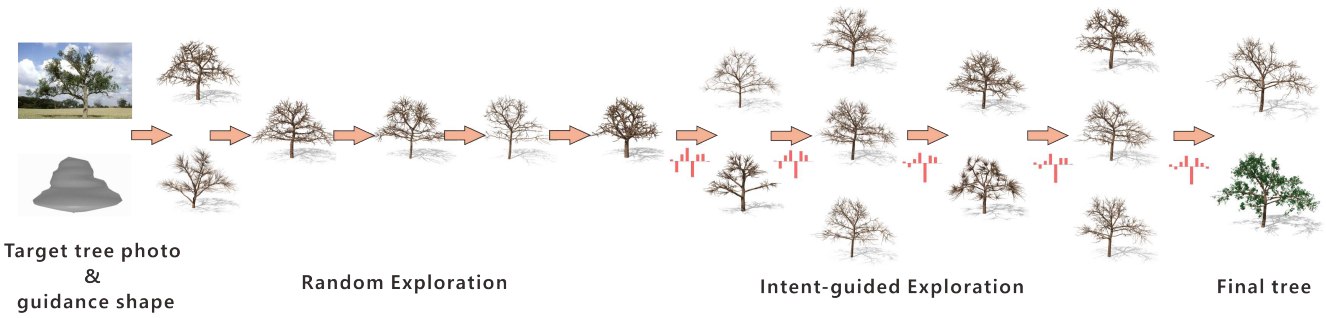


Figure 1. Illustrating the tree modeling procedure of our ExploreTree system. Given a guidance shape and a reference photo (Left), a desired tree (Right) can be iteratively modeled by selecting trees possesing similar traits with the reference (Middle). In this example, the first 6 trees are selected through a random exploration to warm-up the online learning procedure. Thereafter, at each iteration, our system employs a user intent evaluation model (visualized as red bars) to evaluate user intent from their selections and guide the generation of new candidate trees. Note, we only show the user selected trees at each iteration.

the exploration in the parametric shape space, therefor supports modeling trees mainly based on user perceptions.

In this paper, our major contributions are:

- By collecting data from designed crowdsourcing experiments, we compute a new semantic trait space to represent trees, where an off-line learning algorithm is leveraged to capture users' perceptions of tree shapes, and the mapping between the semantic trait space and the parametric tree modeling space is then built;
- We propose an online learning algorithm to learn the user intent evaluation model, which includes embedding modeling parameters into the semantic trait space, evaluating user intent scores and recommending new candidates successively during modeling;
- Based on those two key points, we present a new interactive exploratory method to model trees.

2. Related Work

In the past few decades, many methods on tree modeling have been published. Here, we only review some most relevant work. Please refer to [7] for an entire overview of this field.

Rule-based Tree Modeling Since the growth of trees implies strong botanic rules and patterns, the work of tree modeling began early with rule-based methods. In general, they are used to simulate botanical organs or growing process rather than model a real tree.

The work of [11] pioneered in the tree modeling work by proposing a few structural parameters and some simple rules to describe the form of a tree-like body. Since then, the rule-based method has been further developed, where the L-system [33, 31], and its various extensions [25, 17], become popular and have been widely used in the field.

The L-system based methods usually require the user to write botanical rules to generate a specific tree [32]. In spite of the expressive power, some expert knowledge are necessary for users, since the rule may involve a number of parameters to encode some botanical or geometric patterns. It prevents an ordinary user to model a specific tree efficiently. By addressing this limitation, a series of work [35, 28, 21] have been published to further improve the L-system to support modeling trees in a more convenient way.

Many other parametric models have also been proposed either to simulate the botanical growth mechanisms [6, 30] or to synthesize a tree with full control on its geometries [42]. Such models with heavily used parameters also prevent an ordinary user to synthesize a specific tree efficiently.

Lintermann and Deussen [19] proposed a new paradigm for the procedural tree modeling and have been implemented in some successful commercial softwares, such as Xfrog and SpeedTree. During the modeling process, the main interaction is to organize some predefined components into a structured tree graph. However, various parameters within each component are still required to be adjusted by users to model a specific tree.

Sketch-based methods [27, 13, 3, 43] provide a design-friendly tree modeling scheme guided by strokes. However, due to the complexity of the morphology of a tree, these methods always adopt some simple rules to automatically infer the full 3D structure of a tree from a 2D sketch with acceptable complexity to the user.

Recently, [41] proposed a variational framework to synthesize trees from a user sketched silhouette shape, which further simplified the complexity of the sketch input. [44] utilized an exemplar 3D tree-parts database and user sketches to build realistic tree models within few minutes. The tree-parts are connected following the allometry rules and the generation of twigs are guided by the user sketches.

In this paper, starting from a sketching input, we present a new tree modeling method with an intuitive way of inter-

action between the user and the system, where users explore in a semantic trait space built upon human perceptions of trees.

Exploration-based Modeling Recently, methods aimed at designing modeling tools for novice and casual users are developed. One of them, which is different from constructive modeling procedures for skillful users, has attracted much attentions. It can be viewed as an exploratory routine to find interesting results according to some visual perceptions without specialized knowledge. Parametric and design spaces are central in this category of computer graphics applications [22, 10, 16, 12, 8]. The dimensionality of such spaces plays an important role in designing the corresponding exploration procedures. Based on the parametric space, more representative spaces can be computed for further exploration through automatic methods [23, 2] or from crowdsourced data [39, 26, 45]. In [26, 45], the semantic traits are given by experts and an embedding of the objects in such a space then is learned from the crowdsourced data. Our work shares some similarities when learning such a semantic trait space, but the semantic traits are also learned from a crowdsouring study which is even more challenging.

By manually arranging the tree models through collaborations of many users, Talton et al. [39] embedded the tree model into a 2D map. Then, by exploring such a map, user could generate new trees interactively. The interaction between the system is intuitive since it is based on a space with high semantics. However, as has already been mentioned in [39], the intrinsic dimension of the morphological space of a tree is higher than 2. A 2D embedding may lead to huge jumps in tree traits from local to local, and bring cognitive difficulties when exploring in such regions. In this paper, we will further study the semantic trait space of the tree and design our modeling procedure on it.

The modeling procedure is an interactive process involving both the user and the system. The above work mainly focused on the design of the system itself and have paid little attention to model the intent of a user. Recently, [5] extends the concept of exploratory modeling by proposing a user interface that enables the user to quickly provide preference scores for selected shapes. Then a probability density function (pdf) is interactively designed based on such preference scores over the underlying shape space and new models are sampled and presented according to the designed pdf. Also, with the development of information retrieval systems [18, 36], the intent learning procedure has been successfully incorporated into the system to guide the overall retrieval procedure. Inspired by those work, we adapt a learning procedure for the user intent evaluation model to guide the tree synthesize process in our system.

3. System Overview

In Figure 2, we illustrate the major computing modules and the data flows of our tree modeling system, called *ExploreTree*. Overall, it has two major phases. The first phase is an off-line learning procedure of the semantic trait space, which is learnt once. Built upon it, in the second phase, our system iteratively applies an online learning algorithm (Algorithm 1) to update a user intent evaluation model. This user intent evaluation model is the main driven force for modeling trees gradually approaching the user's desired tree. Note that user's interaction in our modeling system is only the selection of tree models in the second phase.

The rest of our paper is organized as follows. In Section 4, we briefly introduce the variational tree synthesis model. Then we describe the tree semantic space and the related learning mechanism in Section 5. In Section 6, we introduce the mathematic model behind our intent learning algorithm. The system implementation with various evaluation results are described in Section 7. Finally, we draw conclusions, discuss the limitations and the future work in Section 8 .

4. Parametric Tree Model

The computing framework of ExploreTree is build upon a parameter-driven tree representation. Theoretically, any parameter-driven approaches of tree modeling can be integrated as the basic tree representation of our framework. For the sake of convenient modeling for ordinary users, in this paper, we employ a recently developed parametric model based on a variational formulation [41]. After defining a *guidance shape* by the user, this model automatically synthesizes a tree by minimizing a variational error. Here, we briefly introduce the notations and formation of this model. And in the following sections, we build a semantic trait space based on this parametric model, where our online intent learning algorithm is designed to support modeling various trees.

A guidance shape is firstly defined as a shape of the desired tree crown. A number of small spheres (called *shape nodes*) is distributed on the guidance shape to form the constraint set $\mathcal{C}$. To find a best tree structure $\mathcal{B}$ according to the guidance shape $\mathcal{C}$, the tree modeling can be formulated as an optimization problem:

$$\mathcal{B}_{opt} = \arg\min_{\mathcal{B}} F(\mathcal{B}; \mathbf{p}_0, \mathcal{C}, \mathcal{R}) \tag{1}$$

where $F(\cdot)$ is the overall cost function, $\mathcal{R}$ is a set of botanical parameters and $\mathbf{p}_0$ is the root position of the tree. Three different costs regarding to the tree structure $\mathcal{B}$ are considered and combined as the overall cost. They are the intrinsic structure energy F_I, the exterior shape-guidance energy F_E due to the guidance shape $\mathcal{C}$, and the parameter control penalty F_P from botanical parameters $\mathcal{R}$.

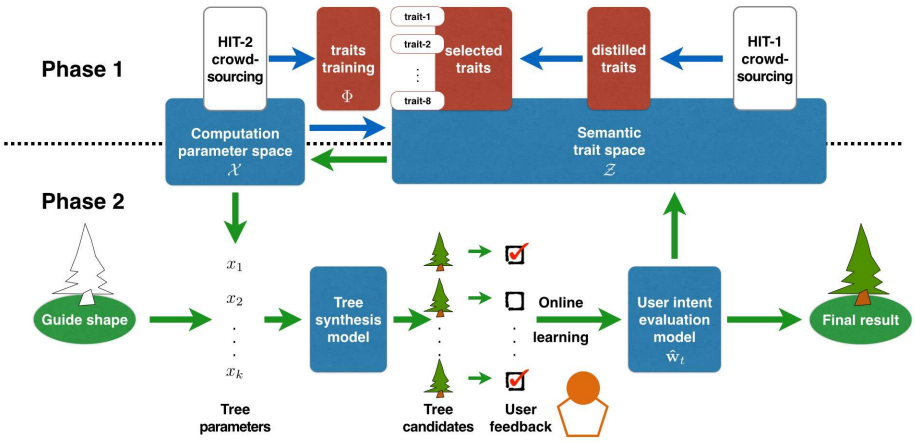


Figure 2. System overview of ExploreTree. In phase 1, the crowdsourced data HIT-1 are used to extract various semantic tree traits, based on which the HIT-2 crowdsourcing tasks are designed and finally used to learn and build such a semantic tree trait space. In phase 2, our intent-based tree modeling process is designed and implemented in a selection-and-recommendation scheme. The computation parameter space and the semantic trait space then build a bridge between the two major phases, where the loop (green arrows) in the center of the figure demonstrates the iterative and exploratory nature of our method.

In this paper, botanical parameters $\mathcal{R}$ is comprised with a set of structure feature descriptions of tree branches. Every branch is represented in a 5-dimensional vector:

$$\mathbf{R} = (N, P, n, \theta, \phi), \tag{2}$$

in the botanical parameter space, where N specifies the number of internodes along the branch axis, P provides an apical control and (n, θ, ϕ) define the phyllotaxic pattern of lateral branches. Please refer to [41] for more technique details. Since a tree is represented as a hierarchical structure, taking the notion of 'levels of recursion' [42], branches in different levels of a tree hierarchy can be either classified with a similar set of parameters $\mathbf{R}_i$ or specified with different values $\mathbf{R}_j$, where $i < j$ for endogenous effects. To simplify the representation, all parameter vectors of different levels are combined to be a botanical arguments set $\mathcal{R} = \{\mathbf{R}_0, \mathbf{R}_1, ...\mathbf{R}_L\}$ and all these control parameters are then treated as a feature vector of a tree. In general, mostly $L = 6$ levels are good enough to model realistic trees in this paper. The actual branches positions are computed from the variational model in Equation 1.

5. Learning Semantic Trait Space of Trees

Numbers of geometrical and botanical features have been proposed to describe and design a 3D tree model [42, 38]. However, these features are somewhat low-level and to our knowledge, a very few previous work in tree modeling had dealt with semantic traits that support users describing 3D tree models through their visual perceptions. Such traits are directly related to the visually structure of the tree and have their corresponding semantic meanings. Also, the parametric space spanned by the geometrical and botanical features often has a high dimension, e.g. [42], and this will incur the 'curse of dimensionality' problem for the intent learning algorithm in Section 6. So to construct a semantic space with a lower dimension will also facilitate our online intent learning process.

In this paper, we mainly focused on computing the branching structures of trees. Other traits, such as leaves and bark patterns, are not taken into consideration in the major computational pass but leave them to a post editing one. Following, we first present the new semantic trait space that we learned from human perceptions of trees, and then introduce the learning process on mapping these semantic traits to the parameters we used to generate trees.

Two parametric spaces We define two parametric spaces as following. One is the *tree semantic trait space* whose basis consists of the visual perception traits of a tree, is denoted as $\mathcal{Z}$. The other is the *tree computational parametric space* where the control parameters of the variational tree synthesis model constitute the feature vector of a tree, is denoted as $\mathcal{X}$. Also, we denote n and m to be the dimension of the space $\mathcal{X}$ and $\mathcal{Z}$ respectively. The embedding transform from the parametric space $\mathcal{X}$ to $\mathcal{Z}$ is defined as

$$\Phi : \mathcal{X} \rightarrow \mathcal{Z}.$$

In this paper, we adopt a linear mapping to constructed such a transform by setting

$$\Phi = [\mathbf{b}_1, \ldots, \mathbf{b}_m]^\top, \tag{3}$$

where each vector $\mathbf{b}_i$ is learnt via a linear SVM classifier to be further discussed in Section 5.2.

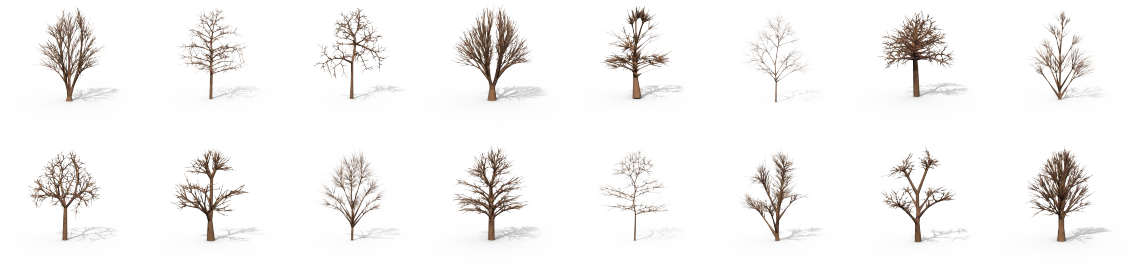


Figure 3. Illustrating of 16 random trees selected from the data set of 7500 example trees.

Example tree data set To learn the trait space, we generate a data set $\mathcal{S}$, including 7500 example tree models by sampling the tree computational parametric space $\mathcal{X}$. The guidance shape is treated as a required input to our system and is designed to be irrelevant to our trait space learning process. Thus, our example trees are all generated in a same guidance shape to simplify the work. Each tree in the data set corresponds to a 3D branch structure and its computational parameters in $\mathcal{X}$. In Figure 3, 16 random example trees selected from the data set are illustrated.

5.1. Semantic Traits

People will perceive a tree in different ways. To collect perceptional traits from different people, we design and publish HITs (Human Intelligence Tasks) on the Amazon Mechanical Turk for anonymous worldwide workers. Since the trees generated with certain parameter combinations will perceived similarly and for efficiency we first cluster 7500 example trees into 64 clusters in the tree computational parametric space using the K-means algorithm. Only one tree in each cluster is randomly selected as the representative tree to be used in the HITs. In each HIT, we randomly select 2 out of the 64 tree images and ask the worker to give descriptions on the similarities and differences of the two displayed trees. We published a batch of 64 HITs with each HIT allowing to have a maximum of 5 assignments. In the supplementary video, we show an example on how one HIT is taken. Finally, a total of 54 workers took our HITs and by eliminating some invalid results, we obtained a collection of 220 valid results, called HIT-1.

Since all the results are described in natural language in a way like "Left tree is fuller than the right. Right tree is skinnier than the other." To process these texts, first, we manually distilled a set of trait keywords, and then parsed all texts to compute the occurrence of each keyword. Due to the polysemy of the natural language, e.g."fuller", "more branches" and "heavier secondary branching" are all related to the description of the branch density of a tree, the occurrence of such keywords with a similar meaning were manually merged. Finally, 24 trait keywords were mined from the data and in Figure 4 they are shown in a descending order according to their occurrences. See the supplemental document for more analysis on the crowdsourcing results.

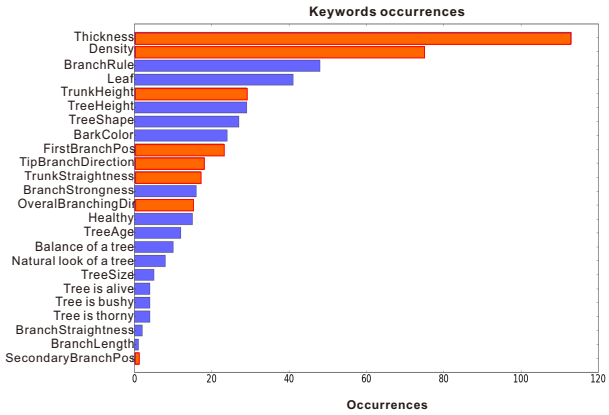


Figure 4. Trait keywords occurrences. The traits with orange bars are finally selected to be our tree semantic traits.

Semantic trait selection Not all traits mined are necessary for our modeling purpose. Firstly, since the variational tree model takes the tree guidance shape as input, all traits used to describe the overall shape of trees are discarded. Secondly, high level semantic traits (e.g. "health" and "alive") are also discarded, since they are implicitly correlated to the low level semantic traits (e.g. "thickness" and "density") and often have less occurrences compared to low level traits. Finally, some traits with occurrences less than a threshold are also discarded. Then, the final 8 traits are listed as in Table 1, where their corresponding cognitive properties summarized from the crowdsourcing results are also given.

5.2. Learning Semantic Trait Space

The directions along the 8 traits will span the tree semantic trait space $\mathcal{Z}$. Ideally, by projecting the trees along such a direction, we can observe a varying trends in the selected trait of the trees, please see Figure 5 for an illustration. We call such a direction as the *semantic trait vector* $\mathbf{b}_i \in R^n$, and further let $\{\mathbf{b}_i\}_{i=1}^m$ forms a basis of $\mathcal{Z}$. In our implementation, $n = 12$ and $m = 8$ are the dimensions of space

Trait name	Cognitive properties
thickness	thick, thin
density	fuller, sparse
trunk height	tall, short
1st branching position	low, high
2nd branching position	near outer layer, near main stem
tip branching direction	divergent, consistent
overall branching direction	upward, horizontal
trunk straightness	straight, bent

Table 1. Selected traits and their cognitive properties. The cognitive property gives two opposite varying directions for each trait.

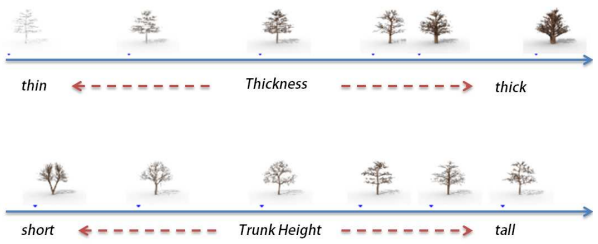


Figure 5. Illustration of trees distributed along 2 trait directions. See the supplemental video for all the 8 trait directions.

$\mathcal{X}$ and $\mathcal{Z}$, respectively. Though we can further construct an orthogonal basis from $\{\mathbf{b}_i\}_{i=1}^m$ for $\mathcal{Z}$, but then the semantic interpretability for each new trait will be lost. By learning the user intent to guide the tree modeling process as detailed in the next section, this interpretability is important for us. Thus, we just keep the semantic traits directly learned from our crowdsourcing study.

We then designed and published HITs for each trait on the Amazon Mechanical Turk. Anonymous workers were asked to label our data set $\mathcal{S}$ with respect to each sematic trait. Specifically, in each HIT, we randomly selected 15 trees from 7500 trees in the example tree data set, and asked workers to label them according to a given sematic trait. For each semantic trait, the workers were given some guidelines to choose the trees from the HIT to label them as positive examples. The maximum assignments of each HIT was set to be 3 with each HIT only labelling one sematic trait. We named this data set as HIT-2.

After labeling the data set $\mathcal{S}$ for all 8 traits, we obtained m data sets $\mathcal{D}^{(l)} = \{(\mathbf{x}_i^{(l)}, y_i^{(l)})\}_{i=1}^N$, where m is the number of selected traits, $l = 1, \ldots, m$, $\mathbf{x}_i^{(l)} \in \mathcal{X}$ and $y_i^{(l)} \in \{+1, -1\}$. Then we trained a binary classifier using the linear SVM with soft-margin [4] on each data set $\mathcal{D}^{(l)}$ and set $\mathbf{b}_i$ to be the normal vector of the separating hyperplane of the trained classifier. The details of the training procedure and various training results are described in Section 7.

6. Intent-based Tree Modeling

Algorithm 1 IntentLearningAlgorithm

Input: $\mathcal{S}$ {the example tree data set}
Initialization: $\sigma^2 \leftarrow 1, t_w \leftarrow 6, c \leftarrow 2, \mu \leftarrow 1,$
$t \leftarrow 1, n \leftarrow 1, K \leftarrow 9, \mathrm{Z}_t \leftarrow \{\Phi\}, \mathbf{r}_t \leftarrow \{\Phi\},$
$\hat{\mathbf{w}}_t \leftarrow \mathbf{0}, \hat{\mathbf{w}}^{(n|n-1)} \leftarrow \mathbf{0}, \Sigma^{(n|n-1)} \leftarrow \mathbf{0}$
{Warm-up the learning process}
for $t = 1$ **to** t_w **do**
  $\mathrm{Z}_t \leftarrow \mathrm{Z}_t \cup$ **randomSample**$(\mathcal{S}, K)$
  $\mathbf{r}_t \leftarrow \mathbf{r}_t \cup$ **userFeedback**(Z_t)
end for
$\hat{\mathbf{w}}^{(n|n-1)} \leftarrow (\mathrm{Z}_t{}^\top \mathrm{Z}_t + \mu \mathrm{I})^{-1} \mathrm{Z}_t{}^\top \mathbf{r}_t$
$\Sigma^{(n|n-1)} \leftarrow \sigma^2 (\mathrm{Z}_t{}^\top \mathrm{Z}_t + \mu \mathrm{I})^{-1} \mathrm{Z}_t{}^\top \mathrm{Z}_t (\mathrm{Z}_t{}^\top \mathrm{Z}_t + \mu \mathrm{I})^{-1}$
{The main online learning procedure}
while not get the target tree **do**
  $\hat{\mathbf{w}}_t \leftarrow \hat{\mathbf{w}}^{(n|n-1)}$
  $\mathrm{Z}_s \leftarrow$ **genCandidateTrees**$(\mathrm{Z}_t, \mathbf{r}_t, \hat{\mathbf{w}}_t, c, K)$ {Sec. 6.2}
  $\mathbf{r}_s \leftarrow$ **userFeedback**(Z_s)
  $\mathrm{Z}_t \leftarrow \mathrm{Z}_t \cup \mathrm{Z}_s$
  $\mathbf{r}_t \leftarrow \mathbf{r}_t \cup \mathbf{r}_s$
  for $(\mathbf{z}^{(n)}, r^{(n)})$ in $(\mathrm{Z}_s, \mathbf{r}_s)$ **do**
    $\mathbf{k}^{(n)} \leftarrow \frac{\Sigma^{(n|n-1)} \mathbf{z}^{(n)}}{\mathbf{z}^{(n)\top} \Sigma^{(n|n-1)} \mathbf{z}^{(n)} + \sigma^2}$
    $\Sigma^{(n|n)} \leftarrow (\mathbf{I} - \mathbf{k}^{(n)} \mathbf{z}^{(n)\top}) \Sigma^{(n|n-1)}$
    $\hat{\mathbf{w}}^{(n|n)} \leftarrow \hat{\mathbf{w}}^{(n|n-1)} + \mathbf{k}^{(n)} (r^{(n)} - \mathbf{z}^{(n)\top} \hat{\mathbf{w}}^{(n|n-1)})$
    $\hat{\mathbf{w}}^{(n+1|n)} \leftarrow \hat{\mathbf{w}}^{(n|n)}$
    $\Sigma^{(n+1|n)} \leftarrow \Sigma^{(n|n)}$
    $n \leftarrow n + 1$
  end for
end while

Based on the learnt tree sematic trait space, our ExploreTree system provides an intuitive and simple tree modeling process. The entire process of modeling a single tree is performed iteratively. That is, at the beginning of each iteration, the system will generate and display a number of candidate trees to the user. Then, the user needs to select "liked" trees, which share certain similar shape features with the target tree. The historical user interactions are all recorded and then used to train an user intent evaluation model. According to the learnt intent, the system will automatically compute top-k new candidates for the next iteration.

In ExploreTree, the modeling of the user intent is the key to drive the whole modeling process. In our work, We represent the intent evaluation model as an evaluation function, f. Such an intent evaluation function is learnt from the user interaction data, and is used to guide the generation of new candidate trees with higher intent scores. There are several challenges in learning f. First, at the early iterations of the

modeling process, the size of the user interaction data set is small and the learnt f may not be accurate. Second, there are difficulties on the trade-offs between the exploration and exploitation process, which are critical to guide users modeling their desired trees effectively. If we generate new candidate trees exactly according to f, which is called the exploitation process, will lead us to a local optimal. In order to avoid such local traps, we must also generate candidate trees that are not fully meet the current f but to some extent will lead us to a better estimate of f. This later process is called the exploration process.

Our intent-based modeling framework is inspired by the LINREL algorithm [1], which was designed for exploratory searching and recently has been successfully applied in data mining [18] and knowledge management [36]. In this paper, we further extended the LINREL algorithm by incorporating the Kalman filtering [15] into the online intent learning process. Our online intent learning algorithm is illustrated in Algorithm 1 and the details of its steps are described in the following sections.

6.1. Learning User Intent

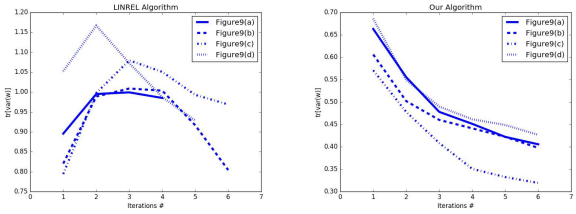
As in [1], we define a linear evaluation function to compute the user intent score for a given tree in the semantic trait space:

$$f(\mathbf{z_I}) = \mathbf{z_I} \cdot \mathbf{w} + \epsilon, \tag{4}$$

where $\mathbf{z}_I \in \mathcal{Z}$ is the semantic trait vector of the tree, $\mathbf{w} \in R^m$ is called the intent weights and $\epsilon \sim N(0, \sigma^2)$ is introduced to model the evaluation error. Since the size of the interaction data is limited and the data itself may often has noises, we can only compute an estimation of $\mathbf{w}$. Thus, our goal is to learn the estimation of $\mathbf{w}$, namely $\hat{\mathbf{w}}$, that best fits the historical user interaction data.

The original LINREL algorithm solved this problem within a regression formulation (Equation 10), where $\hat{\mathbf{w}}$ is updated at each iteration in a batch way (Equation 11). Thus all the data are weighted identically importantly and no obvious control had been exerted on the error of the learnt $\hat{\mathbf{w}}$. But, at each modeling iteration step, a more reasonable way is to treat each newly observed interaction data differently according to its prediction error with our current learnt $\hat{\mathbf{w}}$. The $\hat{\mathbf{w}}$ then is updated by combining its current estimate and the prediction error of the corresponding data. All these considerations motivated us to adopt the Kalman filter, which provides us a framework to continuously combine our predictions with observations, form beliefs, and then update our predictions for the future [37]. More importantly, we get an direct control over the error of $\hat{\mathbf{w}}$ when minimizing its variance to compute the Kalman gain (Equation 8).

In Figure 6, we give the error curves of $\hat{\mathbf{w}}$, when modeling the trees in Figure 9, with the LINREL algorithm and our algorithm with Kalman filter. The error measure



(a) The error curve of LINREL (b) The error curve of our method

Figure 6. Illustrating the error curves of $\hat{\mathbf{w}}$ when modeling the trees in Figure 9. Note, the two algorithms resulted in different iteration steps and to facilitate the comparison, we plot a maximum of 6 iterations after the warm-up process, which is enough for us to demonstrate the trends. The difference is obvious, with our method, the error is decreasing throughout the learning process, which demonstrates a better convergence property.

adopted here is the trace of the covariance matrix in Equation 9.

6.1.1 Learning User Intent From Interaction Data

The Historical User Interaction Data During the modeling process, we maintain a user interaction data matrix Z_t and a user feedback vector $\mathbf{r}_t$ to store the trait vectors of the user selected trees and the selection information respectively. More precisely, after iteration step t, we accumulated K new data points $\{(\mathbf{z}_i^{(t)}, r_i^{(t)})\}_{i=1}^K$. The matrix Z_{t-1} along with the $\mathbf{r}_{t-1}$ then are updated to include such new data:

$$\mathrm{Z}_t = [\mathbf{z}_1^{(1)}, \ldots, \mathbf{z}_K^{(1)}, \ldots, \mathbf{z}_1^{(t)}, \ldots, \mathbf{z}_K^{(t)}]^\top \tag{5}$$

$$\mathbf{r}_t = [r_1^{(1)}, \ldots, r_K^{(1)}, \ldots, r_1^{(t)}, \ldots, r_K^{(t)}]^\top, \tag{6}$$

where $\mathbf{z}_i^{(t)} \in \mathcal{Z}$ and $r_i^{(t)} \in \{0, 1\}$.

The Kalman Filter The key idea in incorporating the Kalman filter to learn $\hat{\mathbf{w}}$ is to utilize a 'prior' estimate of $\hat{\mathbf{w}}$ to compute the 'posterior' estimate based on the newly observed data. Following, to facilitate the discussion, we'll use $\hat{\mathbf{w}}^{(n|n-1)}$ and $\hat{\mathbf{w}}^{(n|n)}$ to denote the 'prior' and 'posterior' estimate of $\hat{\mathbf{w}}$ before and after observing the newly n-th data, respectively. Meanwhile, we use $\Sigma^{(n|n-1)}$, $\Sigma^{(n|n)}$ to denote the corresponding covariance matrices. Note, in ExploreTree, at each iteration there are K such new data points.

The rule for updating $\hat{\mathbf{w}}$ on each data point is:

$$\hat{\mathbf{w}}^{(n|n)} = \hat{\mathbf{w}}^{(n|n-1)} + \mathbf{k}^{(n)}(r^{(n)} - \mathbf{z}^{(n)} \cdot \hat{\mathbf{w}}^{(n|n-1)}), \tag{7}$$

here, $\mathbf{k}^{(n)} \in R^m$ is called the Kalman gain [15], $\mathbf{z}^{(n)} \in \mathcal{Z}$ and $r^{(n)} \in \{0, 1\}$.

The uncertainty of the 'posterior' estimate is characterized by $\Sigma^{(n|n)}$ and we want to reduce it after each update. Thus, the optimal Kalman gain is computed by minimizing

$\Sigma^{(n|n)}$. We define the norm of the covariance matrix to be its trace, $tr[\Sigma^{(n|n)}]$, as in [37] and by minimizing it we get:

$$\mathbf{k}^{(n)} = \frac{\Sigma^{(n|n-1)}\mathbf{z}^{(n)}}{\mathbf{z}^{(n)\top}\Sigma^{(n|n-1)}\mathbf{z}^{(n)} + \sigma^2}. \tag{8}$$

Finally, $\Sigma^{(n|n)}$ can be expressed with $\mathbf{k}^{(n)}$ in a compact form as below:

$$\Sigma^{(n|n)} = (\mathbf{I} - \mathbf{k}^{(n)}\mathbf{z}^{(n)\top})\Sigma^{(n|n-1)}. \tag{9}$$

The derivations are detailed in the supplemental document. Then, by observing the $(n+1)$-th data point, we set the 'prior' estimate $\hat{\mathbf{w}}^{(n+1|n)} = \hat{\mathbf{w}}^{(n|n)}$ and its 'prior' variance $\Sigma^{(n+1|n)} = \Sigma^{(n|n)}$. Following the rule in Equation 7, we denote $\hat{\mathbf{w}}_t$ to be the final estimate achieved after going through the K new data points at iteration step t.

Warm-up of the Online Learning Procedure To start the learning algorithm, we need to provide an initial estimate of $\hat{\mathbf{w}}^{(1|0)}$ and $\Sigma^{(1|0)}$. To this end, we adopt the common warm-up strategy in machine learning to compute such initial estimations. For the first t_w iterations, we randomly select trees from the dataset $\mathcal{S}$ without replacement for the user interactions. Then, as in [1], we compute the best $\hat{\mathbf{w}}_{t_w}$ by solving the following regularized regression problem

$$\underset{\mathbf{w}_{t_w}}{\operatorname{argmin}} \|\mathbf{r}_{t_w} - \mathrm{Z}_{t_w}\mathbf{w}_{t_w}\|^2 + \mu\|\mathbf{w}_{t_w}\|^2. \tag{10}$$

Such a quadratic minimization has a closed-form solution. Therefore, after t_w iterations, the best fit $\hat{\mathbf{w}}_{t_w}$ can be computed as:

$$\hat{\mathbf{w}}_{t_w} = (\mathrm{Z}_{t_w}{}^\top\mathrm{Z}_{t_w} + \mu\mathrm{I})^{-1}\mathrm{Z}_{t_w}{}^\top\mathbf{r}_{t_w}. \tag{11}$$

Then we set:

$$\hat{\mathbf{w}}^{(1|0)} = \hat{\mathbf{w}}_{t_w}$$
$$\Sigma^{(1|0)} =$$
$$(\mathrm{Z}_{t_w}{}^\top\mathrm{Z}_{t_w} + \mu\mathrm{I})^{-1}\mathrm{Z}_{t_w}{}^\top\mathrm{var}(\mathbf{r}_{t_w})\mathrm{Z}_{t_w}(\mathrm{Z}_{t_w}{}^\top\mathrm{Z}_{t_w} + \mu\mathrm{I})^{-1},$$

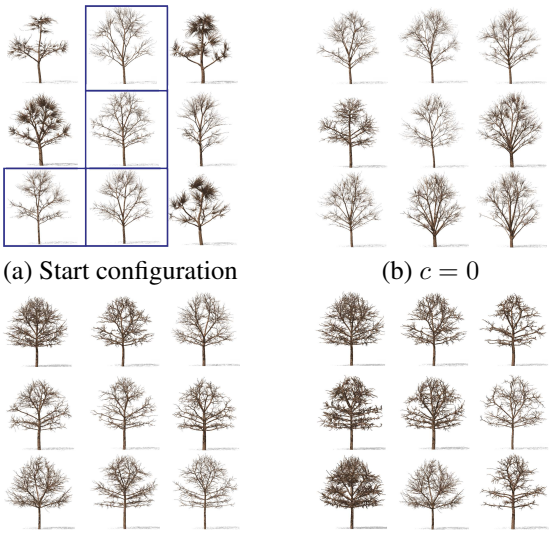
where we set $\mathrm{var}(\mathbf{r}_{t_w}) = \sigma^2\mathbf{I}$. Through experiments we found $\sigma^2 = 1$, $t_w = 6$ and $\mu = 1$ provide good results and we use them as default values in this paper.

6.1.2 Augmenting with Upper Confidence Bound

New candidate trees could be recommended to the user simply based on the learnt $\hat{\mathbf{w}}_t$ for a new round iteration, which is the so-called exploitation process. For reasons as discussed earlier, here we introduce the upper confidence bound [1] to provide a mechanism making the trade-off between the exploitation and exploration process.

Given a candidate tree with trait vector $\mathbf{z}_I$ in space $\mathcal{Z}$, we treat it as a linear combination of the trait vectors of trees in the historical data Z_t:

$$\mathbf{z}_I = \mathrm{Z}_t^\top\mathbf{a}_I, \tag{12}$$



(a) Start configuration (b) $c = 0$ (c) $c = 5$ (d) $c = 10$

Figure 7. Illustrating the effect of different c values on candidate tree recommendations. The three different cases start from a common configuration in (a), where user selected trees are marked with blue boxes. Then, as the c value increases (b)-(d), more variant trees from the current configuration are recommended for the next round iteration.

where $\mathbf{a}_I$ is the coefficients vector of the linear combination. It follows that the estimation of the user intent score $\hat{r}_I$ could be computed as (see Appendix A):

$$\hat{r}_I = \mathbf{a}_I^\top\mathbf{r}_t, \tag{13}$$

which is a linear combination of the corresponding user historical feedbacks. The intuition here is that, the current intent score of the queried model is a weighted sum of the intent scores related to the models in the historical records. The weights are serving as a measure of the similarity between these models.

According to Equation 13, the variance of $\hat{r}_I$ is upper bounded by $\|\mathbf{a}_I\|^2/4$ (see Appendix A). In order to obtain a good estimation, we want to make the variance small by solving the following least-norm optimization problem to find $\mathbf{a}_I$:

$$\begin{aligned} \mathbf{minimize} \quad & \|\mathbf{a}_I\|_2^2 \\ \mathbf{s.t.} \quad & \mathbf{z}_I = \mathrm{Z_t}{}^\top\mathbf{a}_I. \end{aligned} \tag{14}$$

By applying the same regularization term as in Equation 11, the closed-form solution of $\mathbf{a}_I$ is given as follow:

$$\mathbf{a}_I = \mathrm{Z}_t(\mathrm{Z}_t^\top\mathrm{Z}_t + \mu\mathbf{I})^{-1}\mathbf{z}_I. \tag{15}$$

Finally, the intent score of the tree $\mathbf{z}_I$ is computed as:

$$\hat{f}'(\mathbf{z}_I) = \hat{f}(\mathbf{z}_I) + \frac{c}{2}\|\mathbf{a}_I\|, \tag{16}$$

where $\hat{f}(\mathbf{z}_I) = \mathbf{z}_I \cdot \hat{\mathbf{w}}$ is an estimation of the intent evaluation function defined in Equation 4 and $c > 0$ is a *confidence bound coefficient* (see Figure 7). The value of c could be set by the user, but a default value of 2 works well in most cases in our experiments. Thus, $\hat{f}'$ is our final intent evaluation function. As our method shares several similarities with the LINREL algorithm, please refer to [1] for more technical details.

From Equation 16 we know that a tree with a higher score may demonstrate the following two situations. In the first case, the tree is compatible with the current learnt user intent thus get a higher estimate from $\hat{f}(\mathbf{z}_I)$. In the second case, the tree is not compatible with the current learnt user intent but has a higher $\|\mathbf{a}_I\|$ which indicates that by selecting such a tree into the next iteration, we may have a chance to guide the user intent learning algorithm to jump out the current optimal state (see the supplemental document for a theoretical analysis).

6.2. Generating Candidate Trees

While learning the user intent, we want to guide the tree modeling process to generate trees approaching the target tree after several iterations. To this end, at each iteration, we take three steps to generate new tree samples. First, we generate new tree samples according to the historical user interaction data. Then, we rank newly generated sample trees with the user intent scores, and select K trees with highest scores. Finally, the K new candidate trees are synthesized and displayed to the user for the next iteration. These steps constitute the **genCandidateTrees** procedure in Algorithm 1.

6.2.1 Generating New Tree Samples

We use $\mathcal{T}_t$ to denote the set of tree samples generated at iteration t. These tree samples in $\mathcal{T}_t$ are generated from a probabilistic distribution learned from the user interaction data.

Formally, we define $\mathcal{P}(\mathbf{z}|r)$ to be the conditional probability distribution of a tree with feature $\mathbf{z}$ given its user feedback r. Here, $\mathbf{z} = (z_1, ..., z_m)^T$ is a feature vector in the semantic trait space. Thus, $\mathcal{T}_t$ will include tree samples all drawn from the conditional probability $\mathcal{P}(\mathbf{z}|r = 1)$.

To facilitate the learning of such a probability distribution we make two commonly used assumptions as adopted in the Gaussian Naïve Bayes[24], namely, $\forall i, j$, z_i is conditionally independent of z_j given r and $\mathcal{P}(z_i|r)$ is a Gaussian distribution. Then the final probability is:

$$\mathcal{P}(\mathbf{z}|r) = \prod_{i=1}^{m} \mathcal{P}(z_i|r) = \prod_{i=1}^{m} \mathcal{N}(z_i|\mu_i, \sigma_i^2). \quad (17)$$

The parameters μ_i and σ_i^2 are estimated from the user interaction data Z_{t-1} by the maximum likelihood estimation(MLE). Note that the estimated probability from this simple setting is not suffice to serve as a good evaluation of the user intent but only provided us a proper prior for sampling the semantic trait space to generate new candidate trees.

6.2.2 Selecting Candidate Trees

After generating $\mathcal{T}_t$, we want to select K samples from it as new candidate trees for a new round user interaction. We use Equation 16 to compute a score for each sample and the K tree samples with highest scores are selected. Precisely, we define $\tilde{\mathrm{Z}}_t$ to be the matrix with its rows arranged in an order as following:

$$\tilde{\mathrm{Z}}_t = [\mathbf{z}_{i_1}, \ldots, \mathbf{z}_{i_n}]^\top \text{ with } \hat{f}'(\mathbf{z}_{i_1}) \geq \cdots \geq \hat{f}'(\mathbf{z}_{i_n}), \quad (18)$$

here $\mathbf{z}_{i_k} \in \mathcal{T}_t$ and we set $|\mathcal{T}_t| = 60000$. Then $\mathrm{Z}_s = [\mathbf{z}_{i_1}, \ldots, \mathbf{z}_{i_K}]$ forms the set of the candidate trees.

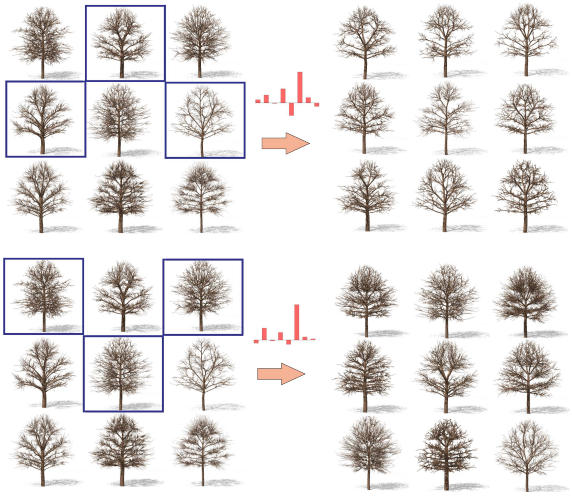


Figure 8. Illustrating the effect of different user selections on a new round recommendation. Left column, the same tree configurations with different user selections (marked as blue boxes). Right column, the candidate trees recommended. The learnt $\hat{\mathbf{w}}_t$ is visualized as a red bar plot in the middle column. The recommended candidate trees on the second row have more dense branches and their tip branch orientations are more consistent, which correspond to our 2nd and 6th semantic traits (see Table 1). These differences are reflected in the red bar plot, where the heights of the corresponding bars are increased.

6.2.3 The Inverse Transform from Semantic Trait Space to Parametric Space

The candidate trees in Z_s are vectors in the semantic trait space, we need to transform them back to the parametric space in order to finally synthesize their 3d branching structures. To map a vector $\mathbf{z}_I$ in space $\mathcal{Z}$ back to $\mathcal{X}$, we apply the Gaussian radial basis function (RBF) to interpolate its

corresponding vector $\mathbf{x}_I$ in space $\mathcal{X}$. By locating L nearest neighbors of $\mathbf{z}_I$ in the data set $\mathcal{S}$, we compute $\mathbf{x}_I$ as follow:

$$\mathbf{x}_I = \frac{\sum_{i=1}^{L} RBF(d_i)\mathbf{x}_i}{\sum_{j=1}^{L} RBF(d_j)}. \quad (19)$$

where d_i is the distance between $\mathbf{z}_I$ and its i-th nearest neighbor in space $\mathcal{Z}$. After that K new trees are generated based on the variational modeling engine for further model exploration and online recommendations.

In Figure 8, we show an example to demonstrate that different user selections will lead to different learnt user intents. Then, as a consequence, the generated candidate trees in the next iteration will differ. In Figure 8, we visualize the learnt $\hat{\mathbf{w}}_t$ as a bar plot, where the height of the bar indicates the magnitude of the corresponding semantic trait to the resulted intent score of a tree. And the bar directions indicates the positive or negative effect of the semantic trait on the intent score, respectively.

7. Implementation and Results

System implementation The user interface of ExploreTree is designed to both support the interactive modeling and post-editing. Our system mainly accepts two kinds of shape inputs namely 3D mesh or 2D sketch. We adopted a spreadsheet-like interface inspired by [14] to support the interactive modeling process. Through this user interface design, the user can view and select multiple trees by simple mouse clicks. In our implementation, 3×3 sub-viewports are displayed so as to provide $K = 9$ candidate trees in each round to make the balance of the simplicity of user cognition and the diversity of selections. But other value of K could be employed in our computation framework without any difficulty. The final tree then is generated and various post-editings are provided to make the tree even more realistic, such as attaching leaves and bark textures to it. Our prototype system is implemented in C++ and Qt in a desktop computer and see the supplementary video for a demonstration of our UI.

Traits training We adopt a leaning scheme to compute the semantic trait vector for each trait as described in Section 5. Here, $m = 8$ traits are selected in our implementation. We first collected a data set of 7.5K labeled trees from Amazon MTurk. For each trait, based on this data, we trained a binary classifier using the linear SVM model with soft-margin. We utilized the Python library scikit-learn[29] to train all our classifiers. For the linear SVM model with soft-margin, we need to set the penalty parameter ξ as the error control term. Thus, our training procedure consists of three passes. First, we split the labeled data set of each trait into two sets: the training set (80%) and the testing set (20%). Then, we used 5-fold cross-validation to train the model on the training set with a series of different values of ξ. The final value of ξ is selected to be the one with the lowest cross-validation mean error. Finally, we trained the model on the whole training set with the selected ξ and used the testing set to evaluate the trained model. In Table 2, we list the selected ξ and the prediction accuracy on the testing set for each trait.

Trait	ξ	Accuracy
thickness	1.0	0.88
density	1.0	0.79
trunk height	0.1	0.69
1st branching position	55.0	0.91
2nd branching position	0.1	0.76
tip branching direction	10.0	0.73
overall branching direction	0.005	0.74
trunk straightness	0.1	0.65

Table 2. SVM training results for each trait.The accuracies are computed from the testing data by comparing the predictions to the crowdsourced labels of the data. The chance accuracy for our binary classification is 0.5.

Intent-based tree modeling evaluation We designed tree matching tasks to evaluate our intent-based tree modeling system. We randomly select 4 target trees from our 7500 example trees and use our modeling system to generate corresponding models. For each target tree, we also record all the candidate trees generated during the modeling iterations. Then we compute the average distances from the candidate trees to the target tree at each iteration. These distance trends for different matching tasks are shown in Figure 9.

The distance metric here should offer a proper way to measure the perceptual similarity between two trees. However, the naive similarity metric with Euclidean distance is not good enough to this end. To illustrate such a problem, in Figure 10 we show three example trees, with the tree in the top row as the reference tree and the two trees at the bottom row as alternative trees. According to the Euclidean distance metric, the bottom left tree will be chosen as the most similar tree to the reference from the two alternatives. This result obviously violates our observation that the bottom right tree is most similar to the reference tree. In [26], when predicting the perceptual similarity between fonts, they also encountered such a problem. Since we share the same situation here, we adopt a similar distance metric:

$$d_{i,j} = \|\mathbf{W}(\mathbf{z}_i - \mathbf{z}_j)\|, \quad (20)$$

where $\mathbf{z}_i$ and $\mathbf{z}_j$ are tree feature vectors in the semantic trait space and $\mathbf{W}$ is a $p \times m$ embedding matrix. Here $m = 8$, is the dimension of the semantic trait space. And $p = 8$ is selected by cross validation (see the supplemental document). According to such a distance metric, the bottom right tree in Figure 10 will be chosen as the most similar tree to the reference.

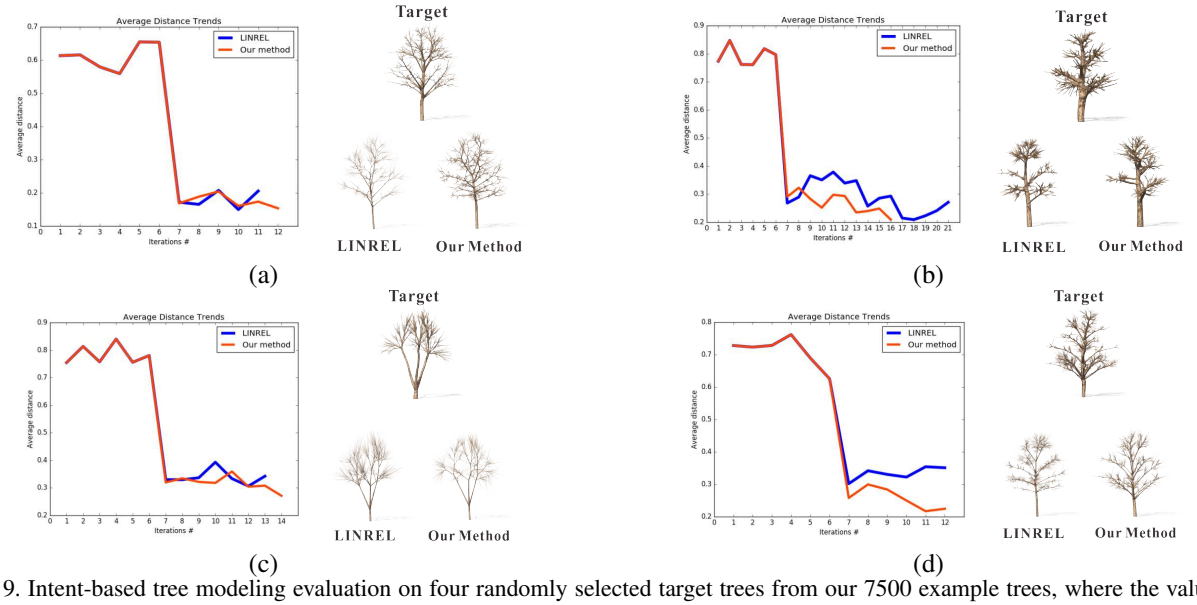


Figure 9. Intent-based tree modeling evaluation on four randomly selected target trees from our 7500 example trees, where the values of the semantic traits of each tree are known. In each sub-figure(a)-(d), the left plot shows the average distance trend against the modeling iterations, where the red curve indicates our method and the blue curve is the LINREL algorithm. The target tree and the resulted trees are shown in the right column. From these distance curves, we can see a better convergent tendency for our methods.

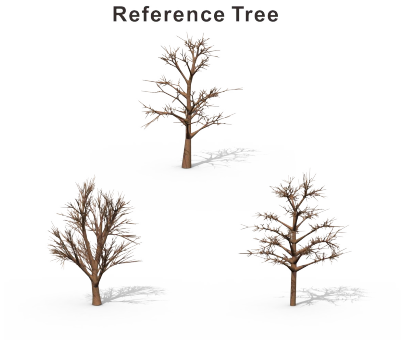
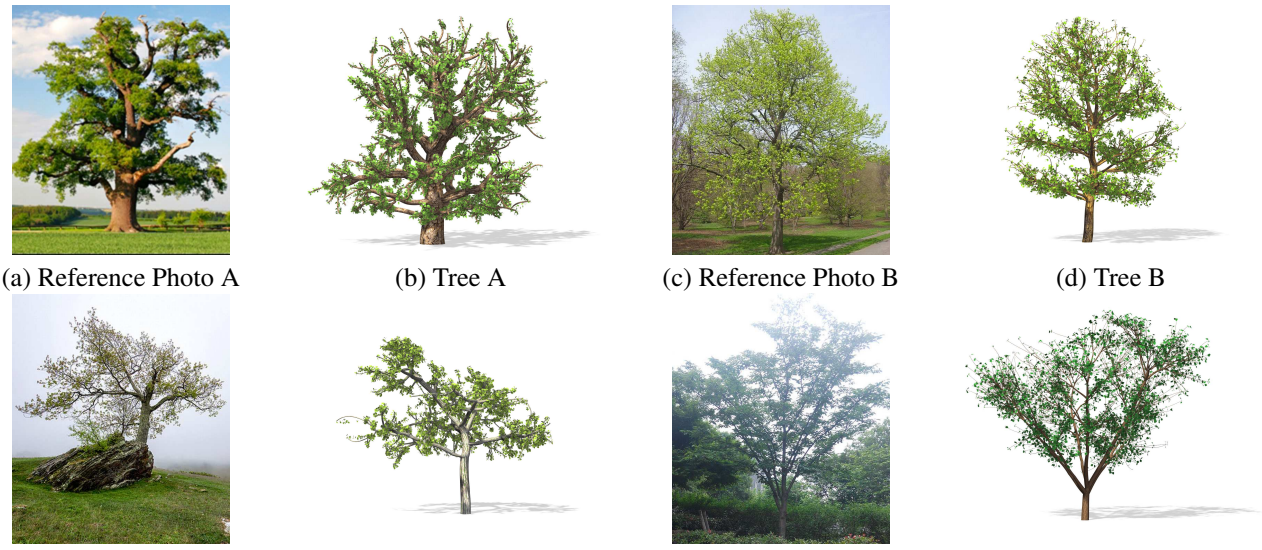


Figure 10. Example task for tree distance study. The worker decides which alternative trees at bottom is more similar to the reference tree above.

In Figure 9 (a)-(d), we show four resulted trees modeled according to their corresponding target trees through our intent-based tree modeling system. Both of the results by using the LINREL algorithm and our method with the Kalman filter are given. All the distances are computed using the learnt distance metric. The candidate trees at the first six iterations are selected randomly from the 7500 example trees to warm-up the learning process. At each iteration since then, the intent evaluation model is updated and new candidate trees are generated according to the learnt intent evaluation model. From these plots, first, we can observe that our method has a nicer convergent tendency than the LINREL algorithm and the resulted trees are more closer to the targets. These are mostly due to our direct control with the error of $\hat{\mathbf{w}}$ at each iteration. Second, the distances between the candidate trees generated by our intent evaluation model and the target tree are declined overall in a fluctuant way. This demonstrates that our tree modeling system can support the users getting their desired models by learning their modeling intents.

Modeling capability To evaluate the capability of our tree modeling system, we mainly asked users to model trees from various photos of real trees. This is more convenient and illustrative than just asking the user to generate trees freely. The photo is only served as a visually guide for the users to select trees during the modeling process and is not an input to the computation procedure during the exploration. In Figure 11, four typical trees are modeled using our system. After the warm-up process ($t_w = 6$ for all the four trees), the number of explorations to generate the final trees are 9, 4, 6 and 6 respectively. In average, each tree takes the user 5 minutes to achieve the final model. Since our system is not a image-based modeling one, it is difficult to capture the exact details from the image. But, by inspecting the results, we found that the tree models possess some obvious traits of the reference trees, though some local details, like branch geometries, may vary. A detailed example is also shown in Figure 1, where it is easy to see that $\hat{\mathbf{w}}$ converges and leads to the final result.

Modeling diversity In Figure 13, we show four different tree modeling results from users with the same reference tree and guidance shape. Though the modeling target is the same, people may have different perceptions on the same tree and this will lead them to generate different trees under varying intents. Nevertheless, it is worth noting that all results captured some sort of features from the reference



(a) Reference Photo A (b) Tree A (c) Reference Photo B (d) Tree B

(e) Reference Photo C (f) Tree C (g) Reference Photo D (h) Tree D

Figure 11. Illustrating the modeling results of trees according to four reference photos using ExploreTree. The reference photos here are not directly involved with any computations in ExploreTree, but serving as a visually guide for the user to select trees during the modeling process. The final tree models ((b), (d), (f), (h)) then possess a number of similar traits to the reference.

tree.

User studies We conducted user studies on a group of 21 subjects to further evaluate our tree modeling system in many aspects. All subjects have no previous experience on our modeling system, and covers the age from 12 to 45. Every subject was asked to create tree models according to a fixed set of 2 reference photos of real trees using our modeling system. All subjects accomplished the modeling tasks in less than 20 minutes. For example, Figure 13 illustrates the modeling results using one of the photo by differen users. When finishing the tree modeling task, each subject was presented with a series of statements, including 14 questions, of our modeling system. For each statement, a subject was asked to give a score in $[1, 5]$ with 5 indicates 'strongly agree' and 1 for 'strongly disagree'. The survey results are summarized in Figure 12. From the survey results, almost

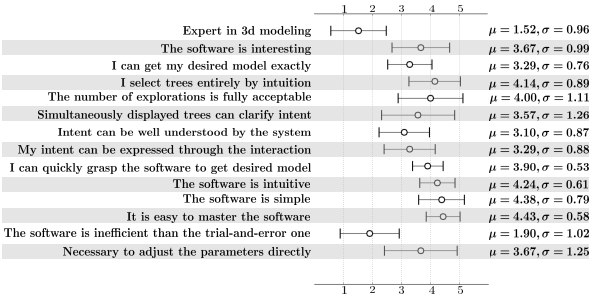


Figure 12. The survey results from 21 subjects of our intent-based tree modeling interface. The subject rates 1 for 'strongly disagree', 3 for 'neutral' and 5 for 'strongly agree'. Error bars indicate one standard deviation.

no subject claimed his/her 3D modeling knowledge as "expert" and the majority of subjects were agreed on the modeling system to be simple and intuitive. More importantly, the subjects also had a common agreement on the fact that our modeling system can "understand" their modeling intents and generate their desired models.

Besides, the subjects also showed an agreement on providing a direct way of parameter adjustment to support model fine-tunings for more realistic results. Thus, we conducted another user study on 10 subjects to further evaluate our intent-based tree modeling system over the traditional trial-and-error one. We developed an interface with the 8 semantic traits directly adjustable through sliders to provide the subjects a tree modeling system with a trial-and-error style. Then for each system, the subjects were asked to model 5 target trees both within a time limit of 10 minutes. The target trees are randomly selected from our example tree data set and are kept the same for all subjects when using both of the two systems. In Table 3 there is a summary of the results for the two systems, where the mean distances from the resulted trees to the target ones are given. These results showed that our intent-based tree modeling system can support the users achieving their modeling targets more closely in a limited modeling time. From Table 3, the standard deviations for our intent-based tree modeling system are also smaller, which indicates that our system provided the user a more consistent way for tree modeling. The main reason here is that there exists different learning curves for the users when mastering the system with a trial-and-error modeling style. Within a limited time, the consequences of these differences are magnified and the resulted tree models are more diverse. Since our intent-based system does not

Target tree	intent-based	trial-and-error
1	0.35 ± 0.19	0.36 ± 0.28
2	0.24 ± 0.12	0.32 ± 0.22
3	0.20 ± 0.14	0.36 ± 0.31
4	0.29 ± 0.15	0.23 ± 0.25
5	0.34 ± 0.19	0.36 ± 0.21

Table 3. The mean distances from the resulted trees to the target ones for two modeling systems. In the second column, the mean distance with one standard deviation for our intent-based tree modeling system are given. In the third column, the same results for the tree modeling system with a trial-and-error style are given. All the distances are computed using Equation 20.

involve any explicit parameter tunings during the modeling process such problems can be avoided.

To generate tree models with certain semantic traits, our exploratory modeling system guided by user intents will conform to the major perceptions of the target tree. But for some detailed lower level controls, like the geometries of the branches, it is seemed more convenient to manipulate parameters directly. Fortunately, our prototype system already integrated such functionalities into the post editing mode.

8. Conclusion

In this paper, we proposed a novel tree modeling system with an intuitive user interface design. The main goal, when designing our system, is to ease the learning and cognitive burdens on using our system to model various trees. Thus, the main difference of our system to the previous ones on providing professional modeling solutions for senior users is on supporting ordinary users modeling trees through a selection-and-recommendation scheme, where their modeling intents are automatically deduced to guide the exploration in a semantic tree trait space.

The semantic traits of the tree are distilled and learnt from the crowdsourced data collected through the Amazon MTurk, where an off-line learning procedure is carried out to learn the final semantic trait space. These tree semantic traits provide us a way to understand the human perceptions on different tree shapes and support our online intent learning algorithm to learn the user intent evaluation function in a space that is close to the user perceptions. Finally, intensive user studies have been conducted, which demonstrate the efficiency of our system both for the amateur and experienced users, on modeling trees and supporting their explorations in such a tree space.

Limitations and Future Work Although our system is effective for generating realistic trees through a few interactions, it bears some limitations in the current state. First, the size of our textual data collected to distill the semantic traits are limited and the process for selecting semantic tree traits from the textual data are mainly manually, which is inefficient for a larger data set. In the future, we'd like to adopt advanced algorithms, like the deep learning technology [9], to automatically mining such traits from the text data. Second, currently we only consider the semantic traits from the perceptions of ordinary users. By also collecting and analysing the data from botanists, we may complement our semantic traits to support modeling more realistic trees. Also, the semantic traits currently are learned through batch comparisons and the resulted loss in accuracy when comparing to the pairwise comparisons as in [26, 45] should be further studied. Third, our current system incorporating no image-based modeling techniques, by combining which the final results may even be more compelling.

There are also several other valuable directions for our future work. For instance, we could improve our method based on other non-linear parameter embedding techniques. And of course, a larger scale size of crowdsourcing study will help us finding more dimensions of shape perceptions. Also, we can integrate collaborative filtering or personalized recommendation algorithms from the field of machine learning and data mining to further customize the modeling process for a individual user. Since our computing framework is a data-driven one, the presented system could be extended for other geometry design issues (e.g. buildings, faces, machinery and 2D artworks) without large work, if a suitable parametric representation can be found to match our computation framework.

Acknowledgement

This work was partially supported by National Key R&D Program of China (No. 2016YFB1001503), NSFC (No. 61472350, No. 61232011), the Fundamental Research Funds for the Central Universities (No. 2016FZA5013).

Appendix A. Upper confidence bound

As already mentioned in Section 6, the LINREL algorithm [1] built a linear map between the data feature and its response. Precisely, after iteration step t, this linear regression problem implies that $\mathrm{Z}_t\hat{\mathbf{w}} \simeq \mathbf{r}_t$. Then by incorporating Equation 12 we get:

$$\hat{r}_I = \mathbf{z}_I^\top \hat{\mathbf{w}} = (\mathrm{Z}_t^\top \mathbf{a}_I)^\top \hat{\mathbf{w}} = \mathbf{a}_I^\top (\mathrm{Z}_t \hat{\mathbf{w}}) \simeq \mathbf{a}_I^\top \mathbf{r}_t. \quad (21)$$

To compute an upper confidence bound for $\hat{r}_I$, the LINREL algorithm treats $\mathbf{r}_t$ as a Bernoulli random vector since its elements take binary values. Then by definition, we get:

$$var(\hat{r}_I) = \mathbf{a}_I^\top var(\mathbf{r}_t)\mathbf{a}_I \leq \frac{1}{4}\mathbf{a}_I^\top \mathbf{a}_I, \quad (22)$$

where we also take the assumption that the elements of $\mathbf{r}_t$ are i.i.d Bernoulli random variables and the fact that the variance of a Bernoulli random variable is upper bounded by $\frac{1}{4}$.

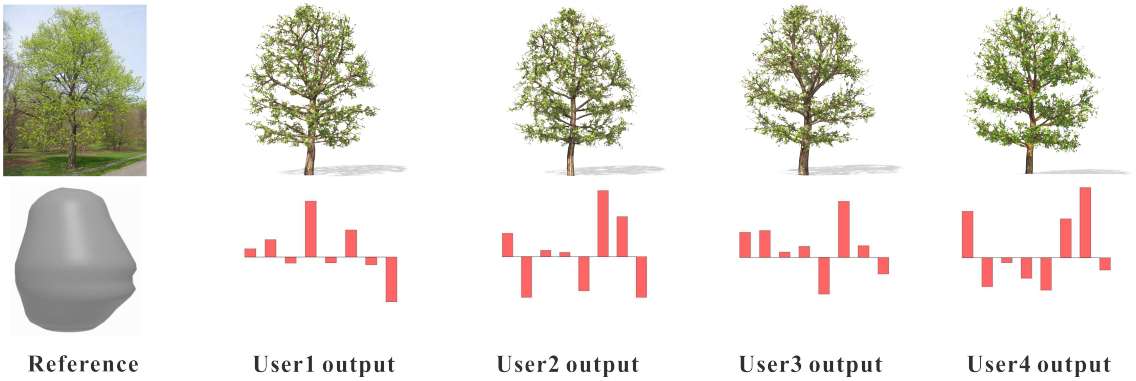


Figure 13. Illustrating the modeling outputs of different users from the same reference photo. The upper left image is the given reference photo and the lower left one is an input 3D mesh as the guidance shape. The upper right four images are rendered results generated by four different users selected from the data of our user study. The lower right four red bar plots are the corresponding intent weights $\hat{\mathbf{w}}$.

References

[1] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *J. Mach. Learn. Res.*, 3:397–422, Mar. 2003. 7, 8, 9, 13

[2] M. Averkiou, V. Kim, Y. Zheng, and N. J. Mitra. Shapesynth: Parameterizing model collections for coupled shape exploration and synthesis. *Computer Graphics Forum (Special issue of Eurographics 2014)*, 2014. 3

[3] X. Chen, B. Neubert, Y.-Q. Xu, O. Deussen, and S. B. Kang. Sketch-based tree modeling using markov random field. *ACM Transactions on Graphics*, 27(5):109, 2008. 2

[4] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. 6

[5] M. Dang, S. Lienhard, D. Ceylan, B. Neubert, P. Wonka, and M. Pauly. Interactive design of probability density functions for shape grammars. *ACM Trans. Graph.*, 34(6):206:1–206:13, Oct. 2015. 3

[6] P. de Reffye, C. Edelin, J. Françon, M. Jaeger, and C. Puech. Plant models faithful to botanical structure and development. In *ACM SIGGRAPH '88*, pages 151–158, 1988. 2

[7] O. Deussen and B. Lintermann. *Digital Design of Nature: Computer Generated Plants and Organics*. Springer-Verlag New York, Inc., 2005. 1, 2

[8] E. Garces, A. Agarwala, D. Gutierrez, and A. Hertzmann. A similarity measure for illustration style. *ACM Trans. Graph.*, 33(4):93:1–93:9, July 2014. 3

[9] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org. 13

[10] M. Hermann, A. C. Schunke, and R. Klein. Semantically steered visual analysis of highly detailed morphometric shape spaces. In *IEEE BioVis 2011*, pages 151–158, 2011. 3

[11] H. Honda. Description of the form of trees by the parameters of the tree-like body: Effects of the branching angle and the branch length on the shape of the tree-like body. *Journal of Theoretical Biology*, 31(2):331–338, May 1971. 2

[12] S.-S. Huang, A. Shamir, C.-H. Shen, H. Zhang, A. Sheffer, S.-M. Hu, and D. Cohen-Or. Qualitative organization of collections of shapes via quartet analysis. *ACM Trans. Graph.*, 32(4):71:1–71:10, July 2013. 3

[13] T. Ijiri, S. Owada, and T. Igarashi. The sketch lsystem: Global control of tree modeling using free-form strokes. *Smart Graphics*, pages 138–146, 2006. 2

[14] T. J. Jankun-Kelly and K.-L. Ma. Visualization exploration and encapsulation via a spreadsheet-like interface. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):275–287, July 2001. 10

[15] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960. 7

[16] Y. Kleiman, N. Fish, J. Lanir, and D. Cohen-Or. Dynamic maps for exploring and browsing shapes. In *Proceedings of the Eleventh Eurographics/ACMSIGGRAPH Symposium on Geometry Processing*, SGP '13, pages 187–196, 2013. 3

[17] W. Kurth and B. Sloboda. Growth grammars simulating trees: An extension of l-systems incorporating local variables and sensitivity. *Silva Fennica*, 31(3):285–295, 1997. 2

[18] A. P. Leung and P. Auer. An efficient search algorithm for content-based image retrieval with user feedback. In *IEEE ICDM 2008), December 15-19, 2008, Pisa, Italy*, pages 884–890, 2008. 3, 7

[19] B. Lintermann and O. Deussen. Interactive modeling of plants. *IEEE Computer Graphics and Applications*, 19(19):56–65, 1999. 1, 2

[20] Y. Livny, F. Yan, M. Olson, B. Chen, H. Zhang, and J. Elsana. Automatic reconstruction of tree skeletal structures from point clouds. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2010)*, 29:151:1–151:8, 2010. 1

[21] S. Longay, A. Runions, F. Boudon, and P. Prusinkiewicz. Treesketch: interactive procedural modeling of trees on a tablet. In *SBIM '12*, pages 107–120, 2012. 1, 2

[22] J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber. Design galleries: a general approach to setting parameters for computer graphics and animation. In *ACM SIGGRAPH '97*, pages 389–400, 1997. 3

[23] W. Matusik, H. Pfister, M. Brand, and L. McMillan. A data-driven reflectance model. In *ACM SIGGRAPH 2003*, pages 759–769, 2003. 3

[24] T. M. Mitchell. *Machine Learning*. WCB McGraw-Hill, 1997. 9

[25] R. Měch and P. Prusinkiewicz. Visual models of plants interacting with their environment. In *ACM SIGGRAPH '96*, pages 397–410, 1996. 2

[26] P. O'Donovan, J. Libeks, A. Agarwala, and A. Hertzmann. Exploratory Font Selection Using Crowdsourced Attributes. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 33(4), 2014. 3, 10, 13

[27] M. Okabe, S. Owada, and T. Igarashi. Interactive design of botanical trees using freehand sketches and example-based editing. In *ACM SIGGRAPH 2006 Courses*, 2006. 2

[28] W. Palubicki, K. Horel, S. Longay, A. Runions, B. Lane, R. Mech, and P. Prusinkiewicz. Self-organizing tree models for image synthesis. *ACM Trans. Graph.*, 28(3), 2009. 2

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 10

[30] J. Perttunen, R. Siev Anen, E. Nikinmaa, H. Salminen, H. Saarenmaa, and Akev. Lignum: A tree model based on simple structural units. *Ann Bot*, 77(1):87–98, Jan. 1996. 2

[31] P. Prusinkiewicz, M. Hammel, J. Hanan, and R. Mech. L-systems: from the theory to visual models of plants. *Plants to ecosystems. Advances in Computational Life Sciences*, 1:1–27, 1997. 2

[32] P. Prusinkiewicz, J. Hanan, and R. Měch. An l-system-based plant modeling language. In M. Nagl, A. Schurr, and M. Munch, editors, *Applications of graph transformations with industrial relevance.*, volume 1779 of *Lecture Notes in Computer Science*, pages 395–410, 1999. 2

[33] P. Prusinkiewicz and A. Lindenmayer. *The algorithmic beauty of plants*. Springer-Verlag New York, Inc., 1996. 2

[34] A. Ribeiro and T. Igarashi. Sketch-editing games: Human-machine communication, game theory and applications. In *ACM UIST '12*, pages 287–298. ACM, 2012. 1

[35] A. Runions, B. Lane, and P. Prusinkiewicz. Modeling trees with a space colonization algorithm. In *Eurographics NPH 2007*, pages 63–70, 2007. 2

[36] T. Ruotsalo, J. Peltonen, M. Eugster, D. Glowacka, K. Konyushkova, K. Athukorala, I. Kosunen, A. Reijonen, P. Myllymäki, G. Jacucci, and S. Kaski. Directing exploratory search with interactive intent modeling. In *ACM CIKM 2013*, pages 1759–1764, 2013. 3, 7

[37] R. Shadmehr and S. Mussa-Ivaldi. *Biological Learning and Control: How the brain builds representations, predicts events, and makes decisions*. MIT Press, 2012. 7, 8

[38] O. Stava, S. Pirk, J. Kratt, B. Chen, R. Mech, O. Deussen, and B. Benes. Inverse procedural modelling of trees. *Computer Graphics Forum*, 2014. 4

[39] J. O. Talton, D. Gibson, L. Yang, P. Hanrahan, and V. Koltun. Exploratory modeling with collaborative design spaces. In *ACM SIGGRAPH Asia 2009 papers*, SIGGRAPH Asia '09, pages 167:1–167:10, 2009. 3

[40] P. Tan, T. Fang, J. Xiao, P. Zhao, and L. Quan. Single image tree modeling. *ACM Trans. Graph.*, 27(5):108, 2008. 1

[41] R. Wang, Y. Yang, H. Zhang, and H. Bao. Variational tree synthesis. *Comput. Graph. Forum*, 33(8):82–94, 2014. 1, 2, 3, 4

[42] J. Weber and J. Penn. Creation and rendering of realistic trees. In *ACM SIGGRAPH '95*, pages 119–128, 1995. 1, 2, 4

[43] J. Wither, F. Boudon, M.-P. Cani, and C. Godin. Structure from silhouettes: a new paradigm for fast sketch-based design of trees. *Comput. Graph. Forum*, 28(2):541–550, 2009. 1, 2

[44] K. Xie, F. Yan, A. Sharf, O. Deussen, B. Chen, and H. Huang. Tree modeling with real tree-parts examples. *IEEE Transactions on Visualization and Computer Graphics*, PP(99):1–1, 2016. 2

[45] M. E. Yumer, S. Chaudhuri, J. K. Hodgins, and L. B. Kara. Semantic shape editing using deformation handles. *ACM Trans. Graph.*, 34(4):86:1–86:12, July 2015. 3, 13