Collaborative Mapping of a Parametric Design Space

Jerry Talton Daniel Gibson Pat Hanrahan Vladlen Koltun

Virtual Worlds Group
Stanford University

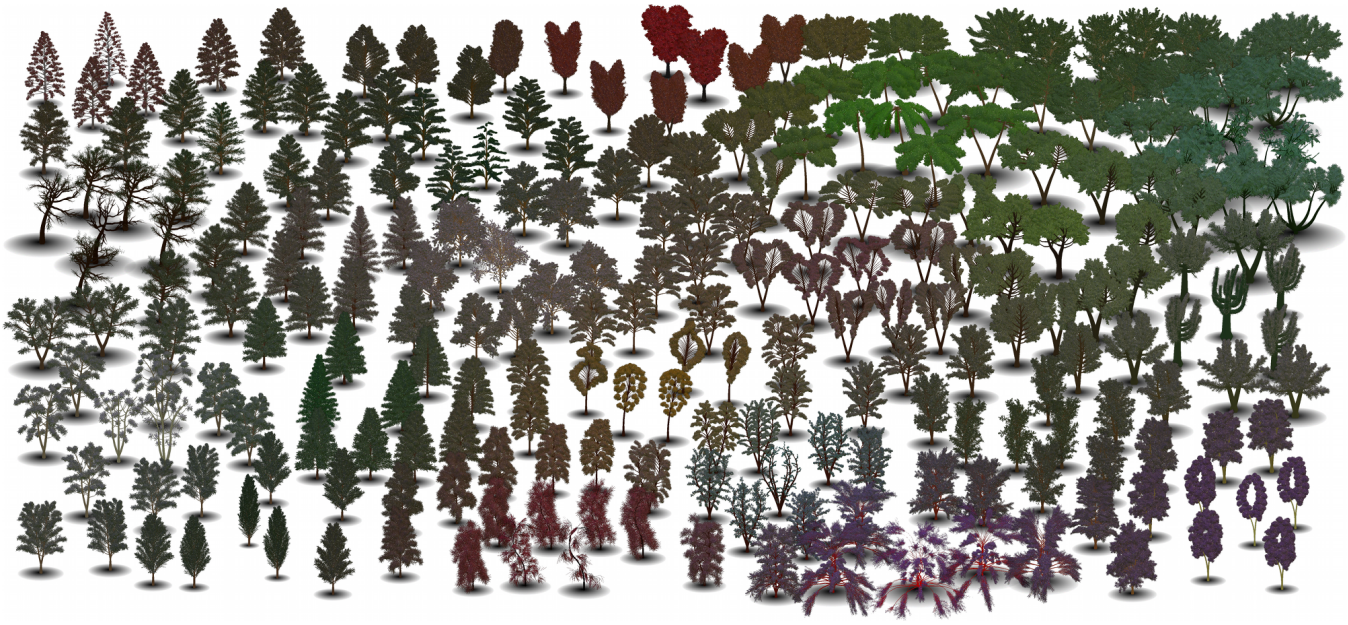


Figure 1: *A section of a parametric space of trees.*

Abstract

We utilize a high-dimensional parametric design space to support a novel and intuitive method for 3D modeling. Users visually explore the design space and pick models using a continuous, map-like interface. We leverage models created by the user community to learn underlying structure in the space via kernel density estimation. This mapping of the space is maintained by a server that synchronizes all the deployed design tools. The tools leverage the mapping to allow users with no prior modeling experience to easily create unique designs by interpolating between and extrapolating from landmark models. The result is a self-reinforcing design system that becomes easier to use as more people participate.

Our prototype tree modeling tool was downloaded by over six thousand users from more than eighty countries in the month following its release. Over fifteen hundred trees were voluntarily picked from the roughly hundred dimensional tree space. We report on usage patterns gathered through this deployment and on subsequent user surveys.

1 Introduction

The widespread availability of 3D graphics hardware and the increasing popularity of three-dimensional participatory media such as networked virtual worlds [Miller 2007], game mods [Newman 2006; Arendash 2004], and machinima [Marino 2004] are motivating many people to attempt to create three-dimensional content. Unfortunately, the creation of high-quality three-dimensional models is a notoriously difficult task. Current modeling tools require considerable training to be used effectively. As a result, 3D modeling is largely done by professionals in the movie and game industries, and not the general population.

Another approach to content creation is 3D scanning and motion capture, which is often called *data-driven graphics*. In addition to direct capture of specific objects, data-driven approaches can be used to construct more general models of motion, shape, and appearance. The basic recipe is to scan a large number of objects and then approximate them in a low-dimensional parametric space using machine learning techniques. For example, a library of motions can be used to construct a general model of human walking by embedding the joint angles in a low-dimensional manifold. This makes it easier for animators to construct new walking sequences, or to target motion to different skeletal models.

Currently this approach requires the ability to digitize real-world objects. We propose using data-driven graphics techniques with virtual objects that do not exist in the real world. Instead of scanning in objects, we propose to use a collection of objects created by a community of users. Each object in our system is an instance of a parametric model, and the set of all possible parameter values specifies the design space for a particular type of object. We learn the structure of the parametric space from specific designs, and then use this structure to enable a more accessible modeling process.

Given a large collection of designs, we use density estimation to approximate the distribution of good designs in the space. As the number of models grows, the estimated density function approaches this underlying distribution. Thus the structure of the design space

is mapped out collaboratively by the entire community of designers. It emerges as a byproduct of self-interested modeling activity on a community scale.

Using the density function, we construct a *map* of the design space. The map is a function from two dimensions to the parametric space. Thus, modeling can be performed by picking new points on this map. The map can also be used for navigation, mitigating the complexity of design tasks.

We created and deployed a prototype tree modeling tool to evaluate these techniques. Figure 1 illustrates the parametric space used by this tool. The tool was released to the public one month before the submission of this paper and has already been downloaded by more than six thousand unique users, resulting in the submission of over fifteen hundred tree models to our centralized database. Preliminary studies indicate that a majority of users believe our software allows them to create compelling tree models and is less cumbersome than existing tools.

2 Related Work

Intuitive modeling interfaces. The difficulty of traditional 3D modeling is well known. A popular alternative approach uses sketching to reduce the modeling task to a series of 2D freeform strokes. In the Teddy system [Igarashi et al. 1999], the user traces the silhouette of a 3D shape, which inflates accordingly. Many operations like extrusion, smoothing, and cutting are supported. SmoothSketch [Karpenko and Hughes 2006] interprets the sketched curves as perceptual contours and fills in hidden cusps. FiberMesh [Nealen et al. 2007] extends the sketching metaphor by embedding the defining curves on the surface of the three-dimensional shape. The curves can be directly manipulated, serving as handles that control the object.

Another prominent direction is image- and video-based modeling, which aims to reconstruct real-world objects from photographs and video sequences. A recent highlight in image-based modeling is Furukawa and Ponce [2007]. For a video-based approach see VideoTrace [van den Hengel et al. 2007], which reconstructs 3D scenes from video streams, inferring geometric information from user-drawn contours.

Anderson et al. [2000] utilize physical creation processes to assist digital modeling. Their first system uses instrumented Lego™-like building blocks. The user assembles these into a physical structure, which is converted to a geometric model and embellished by the system. A second prototype infers the geometry of a physical clay sculpture from images and retrieves a matching model from a library. The stored library models can be rigged and animated in advance, thus aiding the creation process.

A number of systems in the game industry let a player assemble a 3D object from parts, each chosen from a catalogue. An advanced implementation, in which the components are deformable and the resulting object can be animated, is described by Choy et al. [2007].

Parametric spaces in graphics. High-dimensional parametric spaces are central to data-driven graphics. The landmark work of Blanz and Vetter [1999] constructs a space of 3D faces from 200 example models. Every point is a linear combination of the examples. High-level attributes that correspond to meaningful features like age, weight, and gender can be represented as vectors and learned from tagged examples. New faces can be modeled by adjusting these high-level attributes via sliders.

Allen et al. [2003] extend this approach to entire human models, reconstructing a space of body shapes from 250 scanned human figures using a sophisticated registration algorithm. In appearance modeling, Matusik et al. [2003] generate a space of bidirectional reflectance distribution functions (BRDFs) through the acquisition of over 130 real-world examples. Both of these works represent high-level features as vectors in the space and allow individual manipulation of these features.

Blanz and Vetter [1999] attempt to avoid low-quality regions of the face space ("unlikely faces") by fitting a single Gaussian distribution to the examples. Our work uses more principled density estimation inspired by practices in applied statistics and is capable of capturing local quality variations, such as clusters of quality in the space. Matusik et al. [2003] extract the relevant portion of a BRDF space using linear and non-linear dimensionality reduction, successfully reducing the dimension from roughly 100 to about 10. Our approach is complementary and is applicable even when the intrinsic dimensionality of the space is high. Another distinction is that we do not rely on *a priori* availability of large collections of examples: we instead source these directly from the user community.

Ngo et al. [2000] propose to avoid low-quality regions ("a typical parameterized graphic ... contains nonsense images in much higher proportion than desirable images") by modeling the desirable portion of the space as a product of simplicial complexes. Simplicial complexes are also used by Matusik et al. [2005] to circumscribe parts of a space of textures. Their simplicial complex is constructed from 1500 examples. Unlike [Ngo et al. 2000], Matusik et al. [2005] describe a technique for computing the topology of the complex from the underlying geometry, rather than relying on manual specification by the user. Both approaches require explicit availability of extremal instances from the space, which serve as vertices. These extremal instances are used to sharply demarcate the quality regions. While widely applicable, this technique is less appropriate in situations where only an incomplete set of representative instances is initially available, and when quality is a continuous (rather than binary) function.

One of the contributions of our work is a continuous map-like interface for exploring high-dimensional parametric spaces. The interface maintains perceptual continuity in two dimensions, while still allowing the entire space to be explored by the user community. Navigation in high-dimensional design spaces is a recognized problem in computer graphics. Traditional approaches centered around direct parameter control (e.g., via sliders) quickly become intractable as the number of parameters grows. Marks et al. [1997] describe a set of interfaces for parameter setting that typically present a discrete set of landmarks sampled from the space. The sampling is guided by criteria designed to maximize the landmarks' usefulness. Ngan et al. [2006] describe an interface for parameter setting in a space of BRDFs, making extensive use of an original image-based distance metric.

Large scale collaboration. Our system harnesses self-interested actions by members of a large distributed user community. Statistical techniques are applied to derive global knowledge from this pool of collective activity. This knowledge is then used to improve the experience of each user, creating a positive feedback loop. This approach can also be seen in collaborative filtering, which aggregates the preferences of many users to generate personalized recommendations for individuals [Adomavicius and Tuzhilin 2005]. Since its introduction in the early 90s [Goldberg et al. 1992], collaborative filtering has become a Web mainstay [Linden et al. 2003]. More recently, community-scale self-interested activity has been used for image labeling, object recognition in images, and the collection of semantic information [von Ahn and Dabbish 2004; Su et al. 2007].

Another example of mining collective activity in computer graphics is recent work leveraging large image collections sourced from the Web. Snavely et al. [2006] present an interface for interactive exploration of image collections taken at particular geographic locations, leading to an engaging virtual tourism experience. Lalonde et al. [2007] use image collections to seamlessly embed novel objects in existing photographs. Finally, Hays and Efros [2007] describe a novel scene completion algorithm using an

Internet-scale image database.

Tree modeling. Although the focus of our work is not tree modeling per se, our prototype tree modeler builds upon extensive prior research in the field. As described in Section 6, we extend the work of Weber and Penn [1995], which follows earlier approaches [Honda 1971; Oppenheimer 1986] in specifying trees as vectors from a high-dimensional parametric space. Much other work in the area focuses on grammars [Prusinkiewicz and Lindenmayer 1990], culminating in powerful systems that can yield striking visual results [Měch and Prusinkiewicz 1996; Prusinkiewicz et al. 1994; Prusinkiewicz et al. 2001]. Lintermann and Deussen [1998] describe a tree modeler that combines rule-based and parametric interfaces. Substantial recent progress has also been made on sketch- and image-based approaches [Neubert et al. 2007; Okabe et al. 2005; Quan et al. 2006; Reche-Martinez et al. 2004; Shlyakhter et al. 2001; Tan et al. 2007]. A comprehensive introduction to the field can be found in [Deussen and Lintermann 2005].

3 Overview

We propose using the self-interested design activity of a community of users to build better tools for 3D modeling. Our system has two main components. The first is a central server that stores a large number of individual designs created by the community. The second is a client modeling tool that allows individuals to design new objects.

The first key idea behind our system is that we *learn* properties of the space of good designs from the collection of specific models selected by users. Our conjecture is that this space has structure, and that good designs are not just random collections of parameters. We learn this underlying structure using density estimation, a technique common in machine learning and computational statistics. Our approach is described in detail in Section 4.

The modeling system itself is based on parametric models, which are ubiquitous in graphics. For example, the parametric model of an articulated rigid skeleton might be a set of joint angles. A parametric model of a face could be a collection of weights for a set of blend shapes. In this paper, we employ a parametric model of trees. Section 6 describes both the general requirements for a parametric model to be compatible with our method and the particular tree model used in our system.

The second key idea is that we can greatly improve the modeling process itself by leveraging the structure learned from the community. In Section 5 we describe an intuitive map-like interface for selecting models and exploring design spaces. This extends techniques common in the dimensionality reduction literature. In particular, we designate a smooth two-dimensional manifold that passes through known high-quality regions of the design space and adaptively refine this manifold as users pan and zoom across it. This refinement process, which makes extensive use of our density estimation technique, ensures that we allow navigation of the entire design space, instead of unnecessarily restricting users to some subset of it.

4 Density Estimation

Given a set of landmarks points that represent specific user-selected designs, we wish to estimate the density of quality designs in different regions of the space. Let $\{\mathbf{x}_i\}$ be a set of N landmarks from an n-dimensional parametric space $\mathbb{D}$, assumed to be drawn from a probability density function $f(\mathbf{x})$. An approximation $\hat{f}(\mathbf{x})$ of $f(\mathbf{x})$ can be recovered with the Parzen estimate (Figure 2):

$$\hat{f}(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} K(\mathbf{x}, \mathbf{x}_i),$$

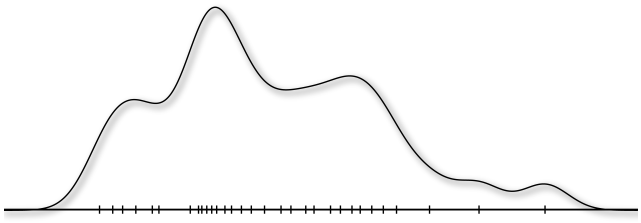


Figure 2: *Parzen density estimation for a set of points in one dimension.*

where K is an appropriate kernel. We use the Gaussian kernel

$$K(\mathbf{x}, \mathbf{x}_i) = \mathcal{G}(\mathbf{x}; \mathbf{x}_i, \mathbf{\Sigma}) = \frac{1}{(2\pi)^{n/2}\left|\mathbf{\Sigma}\right|^{1/2}} \exp\left[-\frac{1}{2}\left(\mathbf{x}-\mathbf{x}_i\right)^T \mathbf{\Sigma}^{-1}\left(\mathbf{x}-\mathbf{x}_i\right)\right],$$

where the k^{th} entry of the diagonal covariance matrix $\mathbf{\Sigma}$ is set to a constant multiple of the sample variance of the k^{th} parameter of points in $\{\mathbf{x}_i\}$:

$$\mathbf{\Sigma}_{k,k} = \frac{c}{N}\sum_{i=1}^{N}\left[\left(\mathbf{x}_i\right)_k - \overline{\left(\mathbf{x}_i\right)}_k\right]^2.$$

Here c is an appropriate constant and $\overline{\left(\mathbf{x}_i\right)}_k$ is the sample mean of the k^{th} parameter.

Our system uses the density function $\hat{f}(\mathbf{x})$ to draw samples from the design space. Since $\hat{f}(\mathbf{x})$ is a uniform mixture of Gaussian kernels, it can be sampled efficiently by first choosing uniformly a random index $1 \le i \le N$ and then drawing a sample from $\mathcal{G}(\mathbf{x}; \mathbf{x}_i, \mathbf{\Sigma})$.

Our modeling interface, described in Section 5, further relies on the ability to sample from $\hat{f}(\mathbf{x})$ *in the local neighborhood* of a point $\mathbf{x}_0$. To accomplish this, we draw samples from probability distributions of the form

$$\frac{1}{\varphi}\,\mathcal{G}\big(\mathbf{x}; \mathbf{x}_0, \mathbf{\Sigma}_0\big)\cdot\hat{f}(\mathbf{x}),$$

where φ is a normalizing constant. Intuitively, we bias the sample to be close to $\mathbf{x}_0$ while still adhering to the quality distribution. To perform this sampling efficiently we notice that

$$\begin{aligned}
\frac{1}{\varphi}\,\mathcal{G}\big(\mathbf{x}; \mathbf{x}_0, \mathbf{\Sigma}_0\big)\cdot\hat{f}(\mathbf{x}) &= \\
\frac{1}{\varphi}\,\mathcal{G}\big(\mathbf{x}; \mathbf{x}_0, \mathbf{\Sigma}_0\big)\cdot\left(\frac{1}{N}\sum_{i=1}^{N}\mathcal{G}\big(\mathbf{x}; \mathbf{x}_i, \mathbf{\Sigma}\big)\right) &= \\
\frac{1}{\varphi N}\sum_{i=1}^{N}\left(\mathcal{G}\big(\mathbf{x}; \mathbf{x}_0, \mathbf{\Sigma}_0\big)\mathcal{G}\big(\mathbf{x}; \mathbf{x}_i, \mathbf{\Sigma}\big)\right) &= \\
\frac{1}{\varphi N}\sum_{i=1}^{N}\left(\mathcal{G}\big(\mathbf{x}_0; \mathbf{x}_i, \mathbf{\Sigma}_0+\mathbf{\Sigma}\big)\mathcal{G}\big(\mathbf{x}; \mathbf{x}_i', \mathbf{\Sigma}'\big)\right),
\end{aligned}$$

where

$$\begin{aligned}
\mathbf{\Sigma}' &= \left(\mathbf{\Sigma}_0^{-1}+\mathbf{\Sigma}^{-1}\right)^{-1} \qquad \text{and} \\
\mathbf{x}_i' &= \mathbf{\Sigma}'\left(\mathbf{x}_0\mathbf{\Sigma}_0^{-1}+\mathbf{x}_i\mathbf{\Sigma}^{-1}\right).
\end{aligned}$$

These inverses can be computed rapidly since $\mathbf{\Sigma}$ and $\mathbf{\Sigma}_0$ are diagonal. Each of the summands is now a scaled Gaussian and the

contribution of summand i to the probability mass is proportional to the constant $\mathcal{G}(\mathbf{x}_0; \mathbf{x}_i, \mathbf{\Sigma}_0 + \mathbf{\Sigma})$. Thus we can sample from $\frac{1}{\varphi}\mathcal{G}(\mathbf{x}; \mathbf{x}_0, \mathbf{\Sigma}_0)\hat{f}(\mathbf{x})$ efficiently by choosing an index $1 \leq i \leq N$ with probability

$$\frac{\mathcal{G}(\mathbf{x}_0; \mathbf{x}_i, \mathbf{\Sigma}_0 + \mathbf{\Sigma})}{\sum_i \mathcal{G}(\mathbf{x}_0; \mathbf{x}_i, \mathbf{\Sigma}_0 + \mathbf{\Sigma})}$$

and then drawing a sample from $\mathcal{G}(\mathbf{x}; \mathbf{x}'_i, \mathbf{\Sigma}')$.

5 Navigational Manifold

To make the problem of navigating a high-dimensional space tractable, traditional approaches often impose significant restrictions on users. For instance, at any given time a user might be forced to choose between a discrete set of landmark points, or to move only along axis-aligned orthogonal paths in the design space. Traditional dimensionality reduction techniques typically force navigation to stay within a single static low-dimensional submanifold, which is unsatisfactory if the intrinsic dimensionality of the desirable regions in the space is believed to be high.

Our learned quality distribution $\hat{f}(\mathbf{x})$ enables the design of a new interface for navigating through high-dimensional parametric spaces. To allow users to explore the space and select particular designs, we define a mapping from a rectangular portion of screen space, R, to points in $\mathbb{D}$. This mapping defines a curved two-dimensional manifold $\mathcal{M}$ in $\mathbb{D}$ that interpolates the set of user-designated landmarks $\{\mathbf{x}_i\}$ and is monotone with respect to a two-dimensional subspace Π of $\mathbb{D}$. This monotonicity ensures that $\mathcal{M}$ can be efficiently projected and visualized in two dimensions, allowing users to explore $\mathcal{M}$ by panning and zooming R like an interactive map.

In contrast to traditional manifold embedding techniques, $\mathcal{M}$ is not held static but is instead adaptively refined as users explore R. In particular, when a user navigates to a region of R that contains few points from $\{\mathbf{x}_i\}$, new points are sampled from $\hat{f}(\mathbf{x})$ and added to $\mathcal{M}$. In this manner our method allows the user community to collectively explore the parametric space, while guiding users to regions of the design space that are believed to be high quality.

The manifold $\mathcal{M}$ is defined with the help of Principal Component Analysis (PCA), although $\mathcal{M}$ itself is not linear. Each parameter of every point in $\{\mathbf{x}_i\}$ is normalized to the range $[0, 1]$ using the absolute range of the parameter if this range is bounded and the current range of the parameter in the set $\{\mathbf{x}_i\}$ if it is not. We then recenter this normalized set about the origin, perform PCA on it, and pick the two dominant eigenvectors to obtain the two-dimensional subspace Π. The orthogonal projection matrix from $\mathbb{D}$ onto Π is denoted by $\mathbf{P}^{\perp}$. For each i, define $\bar{x}_i = \mathbf{P}^{\perp}\mathbf{x}_i$.

The manifold $\mathcal{M}$ is defined through Π. Let $\mathbf{M}$ be the following function from Π to $\mathbb{D}$:

$$\mathbf{M}(x) = \begin{cases} \dfrac{\sum_i \phi_x(\bar{x}_i)\,\mathbf{x}_i}{\sum_i \phi_x(\bar{x}_i)} & : \quad \forall i \,.\, x \neq \bar{x}_i \\ \mathbf{x}_i & : \quad x = \bar{x}_i \end{cases},$$

where $\phi_x(y)$ is an appropriate radial basis function such as $\phi_x(y) = \|x - y\|^{-3}$. The manifold $\mathcal{M}$ is simply the set $\{\mathbf{M}(x) \mid x \in \Pi\}$. Note that adding a new landmark $\mathbf{x}$ to $\{\mathbf{x}_i\}$ alters every point in $\mathcal{M} \setminus \{\mathbf{x}_i\}$, though in practice this is noticeable only for points near $\mathbf{x}$.

5.1 Adaptive refinement

Consider a particular rectangular region $\tilde{R}$ on Π that a user wishes to examine. If the number of landmarks from $\{\mathbf{x}_i\}$ that project onto $\tilde{R}$ exceeds some threshold k, we consider $\mathcal{M}$ to be sufficiently detailed in this region. Otherwise, we refine this section of $\mathcal{M}$ by introducing *transient* landmarks sampled from $\hat{f}(\mathbf{x})$. This refinement generally happens when a user zooms or pans to a region of Π that represents an unmapped area of the design space, and is one of the primary ways in which users explore and map out new regions of $\mathbb{D}$. The transient landmarks introduced in this manner are kept only for the duration of the current modeling session and are not propagated to a central database or other modeling tools.

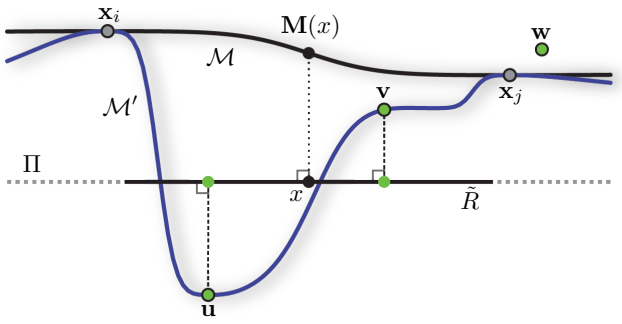


Figure 3: *Refining a section of $\mathcal{M}$ to generate a new manifold $\mathcal{M}'$. We sample $\frac{1}{\varphi}\,\mathcal{G}\big(\mathbf{x}; \mathbf{M}(x), \mathbf{\Sigma}_0\big) \cdot \hat{f}(\mathbf{x})$ to generate $\{\mathbf{u}, \mathbf{v}, \mathbf{w}\} \in \mathbb{D}$. $\mathbf{u}$ and $\mathbf{v}$ are added as transient landmarks. $\mathbf{w}$ is not used since it is too far away from $\tilde{R}$.*

Let x be the center of $\tilde{R}$. To generate a transient landmark, we sample from $\hat{f}(\mathbf{x})$ in the neighborhood of $\mathbf{M}(x)$, as described in Section 4. If the generated sample projects sufficiently close to $\tilde{R}$, we add it to the set of landmarks; otherwise we try again. If the user navigated to a region that is far from all current landmarks and no $\{\mathbf{x}_i\}$ project in the neighborhood of $\tilde{R}$ to begin with, it is unlikely that this process will terminate within an acceptable amount of time. In this case we sample from $\mathcal{G}(\mathbf{x}; \mathbf{M}(x), \mathbf{\Sigma}_0)$ directly. We repeatedly generate transient landmarks until the threshold k is reached. This process is illustrated in Figure 3.

5.2 Visualization

At any given time, the manifold $\mathcal{M}$ is visualized through the screen-space rectangle R, which corresponds to the rectangular region $\tilde{R}$ on Π. In our system, as users mouse over R, the cursor position is registered and is used to update a three-dimensional visualization of the corresponding model from $\mathbb{D}$. To assist the exploration process, we display a set of icons at points in R that depict the corresponding three-dimensional models in $\mathcal{M}$. We heuristically determine a target number of icons and their desired screen-space size. This visualization method is illustrated in Figure 4.

To determine which icons to display, we first compute the set of landmarks that project onto $\tilde{R}$. We shuffle this set and iterate through it, attempting to place each corresponding icon on R. A Poisson-disk rejection criterion [Cook 1986] is used to prevent icons from overlapping. Placement ends when the number of icons reaches the target or when the set of suitable landmarks is exhausted. In the latter case, a set of novel icon locations is generated by Poisson-disk sampling. Since it is not essential that the target number of icons is reached, this process is capped at a small number of iterations to prevent it from becoming stuck in problematic areas.

6 Parametric Models

We define a parametric model to be a mapping from a set of parameters to a renderable object. The number of parameters in the model defines the dimensionality of the parametric design space. Such models are ubiquitous in computer graphics and used in a variety of applications.

In order for a parametric space to be mappable with our method,

Figure 4: *(Left) A screenshot of our prototype tree modeling tool. (Right) Users can toggle between this direct parameter adjustment interface and the map interface shown on the left.*

it must possess two important properties. In particular, the space must be:

Metric: The parameter space must have a metric so that the distance between points can be computed. Ideally, this metric should correspond to *perceptual* distances between models, but determining such metrics is extremely difficult and well beyond the scope of this paper. In practice, we have had satisfactory results with the standard Euclidean metric.

Continuous: In order for our technique to be effective, we require a sort of geometric and perceptual Lipschitz continuity. Namely, points that are close in the design space should correspond to perceptually similar models. In particular, we must be able to meaningfully interpolate between design space points, although it is not strictly necessary that all parameters be real-valued.

6.1 Tree space

To test the ideas presented in this paper, we built a tool to explore a parametric space of trees. We chose trees for a number of reasons. First, trees are ubiquitous in multi-user 3D environments like games and virtual worlds and there is significant demand for good tree models. Second, trees have complex and interesting structure that encourages playful exploration, and are familiar to casual computer users and expert modelers alike. Last, there exists a large body of literature on tree construction and several excellent parametric models are available. We based our space on the model developed by Weber and Penn [1995].

We made several changes to Weber and Penn's original parameter specification in order to create a design space appropriate for our technique. Although we highlight the most important modifications in the following paragraphs, space does not permit us to completely describe the resultant model, which has ninety-eight dimensions. For those interested in reproducing our results, we will make source code for our parametric model available along with this paper.

In the Weber and Penn model, parameters frequently trigger conditional modes of execution. For example, the interpretation of a particular parameter can vary greatly depending upon its sign. This behavior violates our continuity requirement and we modified all such parameters to be interpreted consistently, either by discarding one of the operational modes or by splitting the parameter into two.

Enforcing continuity required several other changes to the parameter space. One such modification is the reinterpretation of *Levels* as a floating point (rather than integer) parameter. This parameter is reinterpreted in a natural way, so that a tree with *Levels* $= \ell$ has $\lceil \ell \rceil$ physical levels, with proportion $\ell - \lfloor \ell \rfloor$ of its total leaves distributed at level $\lfloor \ell \rfloor - 1$ and the remaining $1 - \ell + \lfloor \ell \rfloor$ at level $\lfloor \ell \rfloor$. Furthermore, the length of the branches on level $\lfloor \ell \rfloor$ is set to be $\ell - \lfloor \ell \rfloor$ times the length of the branches on level $\lfloor \ell \rfloor - 1$. Therefore, trees that differ only in *Levels* appear to "morph" smoothly into one another as one interpolates between them, instead of popping discontinuously into visual equivalence. For the same reason, we are meticulous when assigning random seeds, so that the seeds of corresponding branches in different trees are always equal. In this manner it is possible to smoothly interpolate one tree into another by traversing a contiguous path in the design space.

We found some of the parameters in Weber and Penn to be unnecessary, either because they were redundant, linearly dependent, or merely contributed little to the expressive power of the space. We removed as many of these parameters as possible while still retaining perceptual completeness.

7 Tree Modeling Prototype

In our tree modeling prototype, the map interface is displayed in the right pane and the user's currently selected tree is shown in the left (see Figure 4, left). By clicking and dragging in the left pane, the camera can be rotated around the displayed tree. The tool initializes to a default Christmas tree, and in this fashion the cognitive stumbling block of starting with a blank screen is avoided. The current tree can be saved by the user at any time as an OBJ file; when this occurs, the corresponding design space point is uploaded to our centralized server. The interface also provides standard undo/redo functionality and a "randomize" button that produces a variant of the current tree by changing the random seed.

A toggle switches between the map navigation interface (Figure 4, left) and a slider interface that gives precise control over most of the individual parameters in the space (Figure 4, right). The user is free to alternate between the two at any time. When a user

Figure 5: *(Top) Six points chosen uniformly at random from the tree space. (Bottom) Six points sampled from our density function.*

modifies the tree using sliders and then returns to the map interface, the tool adds the modified tree to the set of transient landmarks and recenters the map around it.

As the user mouses over the map, the displayed three-dimensional tree model continuously updates to the corresponding point in the parameter space. The user can click on any point in the map to select it as the current tree. The map itself may be panned and zoomed: clicking and dragging causes the displayed region $\tilde{R}$ on Π to translate accordingly, while moving a zoom slider causes $\tilde{R}$ to expand or contract.

Our modeling client is written in C++ and uses OpenGL for rendering. To facilitate crossplatform development, we made extensive use of the Qt framework [Trolltech Inc. 2007]. Because the geometry generated by the client is complex and because we employ many advanced real-time rendering techniques (such as environment-based lighting, bump-mapping, and render-to-texture), our modeler requires a relatively modern graphics card.

Our centralized tree server is written in Ruby and is based on Mongrel [Shaw 2007], a fast HTTP server library. SQLite [Hwaci 2007] is used to store and manage tree data and user information on the backend. To ensure the stability of the system in the face of many simultaneous users, we employ Pound [Apsis Security 2007], a load-balancing reverse HTTP proxy.

At the beginning of a modeling session, the client sends a GET request to the server, which responds with a list of parameters for any newly-added landmarks from the tree space that are not present in the client's cache. The server also sends several numeric constants, allowing us to modify settings after the initial deployment of the software. The set of landmarks is updated only when the client initializes so as to avoid perceptual discontinuities within a given session. When a user saves a local copy of a tree mesh, the client sends a PUT request with the parameters of the corresponding point from the parametric design space. These are parsed by the server and added to the central landmark database. Additionally, each client keeps a log of user interactions with the interface and periodically uploads this log to server, where it is hashed with a randomly generated per-client ID. In this manner we are able to observe how large numbers of users interact with the software, while protecting their anonymity.

8 Results

Our tree modeling prototype was made available to the public exactly one month before the time of this writing. In this period it has been downloaded 6,868 times by users in 88 countries, resulting in the creation of 1,545 unique tree models. To analyze the effectiveness of our technique, we performed statistical analysis on the set of mapped landmarks in the parametric space. We also conducted a detailed user study by mining the database of anonymous usage logs and soliciting responses to a short survey.

8.1 Tree space analysis

We analyzed the set of user-submitted landmarks to test a number of hypotheses about the tree space and our methodology.

First, we conjectured that the distribution of desirable trees in the space has structure: that it is not uniform. To this end we compared the set of $N = 1,545$ user-submitted landmarks with a set of N samples picked uniformly at random from within the minimal bounding box of the user-submitted set. Figure 6(a) shows the pairwise-distance histograms of the two sets. For a given integral distance value, each histogram shows the number of pairs of points from the corresponding point set whose rounded pairwise distance is at that value. A tell-tale sign of random point sets in high-dimensional spaces is that the set of their pairwise distances is sharply concentrated around the mean [Weil and Wieacker 1993]. This is reflected by the spiky appearance of the distance histogram for the random point set. The set of user-submitted landmarks, on the other hand, exhibits a much wider distance distribution, indicating significant non-uniformity. The same conjecture can be qualitatively confirmed through simple examination, as demonstrated in Figure 5. The top row shows typical random samples from the tree space. The bottom row shows typical random samples from the probability density function $\hat{f}(\mathbf{x})$ learned from the set of user-mapped landmarks.

We also found that the set of user-submitted landmarks appears to have high intrinsic dimensionality. This lends further support to the use of density estimation in this context, instead of dimensionality reduction alone. We employed two established dimensionality reduction techniques: Principal Component Analysis and Isomap [Tenenbaum et al. 2000]. PCA finds the best-fit hyperplane through the data, while Isomap attempts to fit a nonlinear manifold. To evaluate the extent to which these methods approximate the data, we examined the residual variance [Tenenbaum et al. 2000]. The

residual variance is given as $1 - R^2(\hat{\mathbf{D}}_m, \mathbf{D}_y)$ where $\mathbf{D}_y$ is the matrix of Euclidean distances recovered by the algorithm, $\hat{\mathbf{D}}_m$ is the algorithm's best estimate of the intrinsic manifold distances, and R is the standard linear correlation coefficient taken over all entries of $\mathbf{D}_y$ and $\hat{\mathbf{D}}_m$. Figure 6(b) shows the residual variance plotted against dimensionality.

In both cases, the residual variance decreases as the dimensionality is increased. PCA eventually becomes stuck, demonstrating its inability to properly approximate the set of desirable trees. Isomap is able to obtain a good reconstruction only at about forty dimensions. The lack of a sharp "elbow" (point at which the curve ceases to decrease significantly with added dimensions) demonstrates that neither method is capable of conclusively determining the intrinsic dimensionality of the space.

Finally, we tested to what extent the initial set of landmarks that existed when the tool was first made publicly available influenced the eventual set of user-submitted trees. The initial set of landmarks comprised 26 trees, most of which are shown in Figure 8. Our conjecture was that while correlation certainly exists, the user community was not confined by the initial set in the long term, due to adaptive refinement of the navigation manifold and the ability to reach almost any point in the space through the direct sliders interface.

As a preliminary test of this hypothesis, we performed clustering on the final set of landmarks and measured the number of clusters that are necessary to optimally represent the data. Intuitively, a good clustering maximizes intra-cluster distance and minimizes inter-cluster similarity. We iteratively applied k-means clustering [Lloyd 1982] with increasing k to the set of landmarks and measured the quality of the clustering after convergence using two different quality measures [Raskutti and Leckie 1999]. Since no accepted metric exists for determining the optimal number of clusters in general, we employed two distinct quality measures: the first biased towards small clusters, and the second towards larger ones. The first quality measure we evaluated was minimum total distance, given by

$$\sum_{j=1}^{k} \left(\sum_{\mathbf{x}_i \in C_j} \|\mathbf{x}_i - C_j^{\text{avg}}\| \right) + \sum_{j=1}^{k} \|C_j^{\text{avg}} - \mathbf{x}^{\text{avg}}\| \,,$$

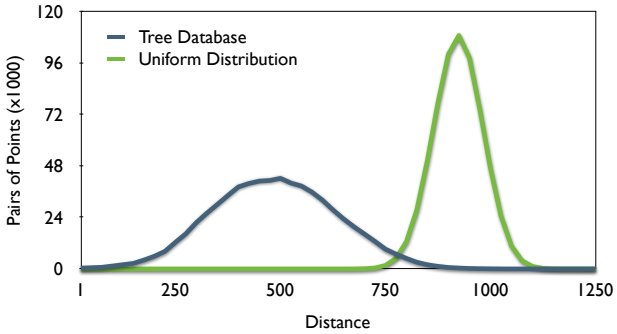
where C_j is set of database points in cluster j, C_j^{avg} is the cluster's centroid, and $\mathbf{x}^{\text{avg}}$ is the global centroid. The second measure was cluster separation:

$$\left[\sum_{j=1}^{k} \left(\frac{\max_{\mathbf{x}_i, \mathbf{x}_l \in C_j} \|\mathbf{x}_i - \mathbf{x}_l\|}{\min_{i \neq j} \|C_i^{\text{avg}} - C_j^{\text{avg}}\|} \right) \right]^{-1} .$$

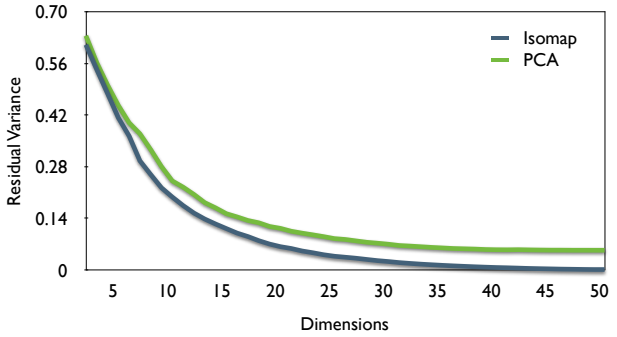
To ensure a reliable estimate for each k, we ran k-means a number of times until the quality measure did not significantly diminish for several iterations. Figure 6(c) shows the clustering quality for each k. For both measures, the optimal number of clusters indicated is well over 100. We conclude, therefore, that the users collectively mapped out clusters of desirable trees in the space that were not represented by the initial 26 landmarks. This point can also be appreciated by examining Figures 8 and 9. The former shows almost all of the initial landmarks, while the latter demonstrates models created by users.

8.2 User studies

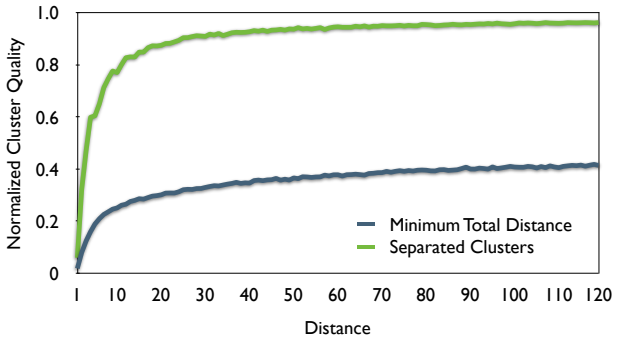
After the number of downloads of our tool topped five thousand, we solicited responses to a short survey from users who had requested to be added to our mailing list during the download process. These users were presented with a series of statements about the software and asked to evaluate these via the standard 5-point Likert scale [1932], where 1 = 'strongly disagree', 3 = 'neutral',



(a) Pairwise-distance histograms for the set of landmarks and a comparable set of points chosen uniformly at random. The two sets exhibit markedly different signatures, indicating non-uniform structure in the set of landmarks.



(b) Residual variance of the set of landmarks versus dimensionality for PCA and Isomap. The gradual decline of both curves suggests that the dimensionality of the set of desirable trees is high.



(c) Clustering quality versus number of clusters for the set of landmarks. The optimal number of clusters is clearly greater than the number of initial landmarks that were used to seed the space.

Figure 6: *Statistical analysis of the set of user-submitted trees.*

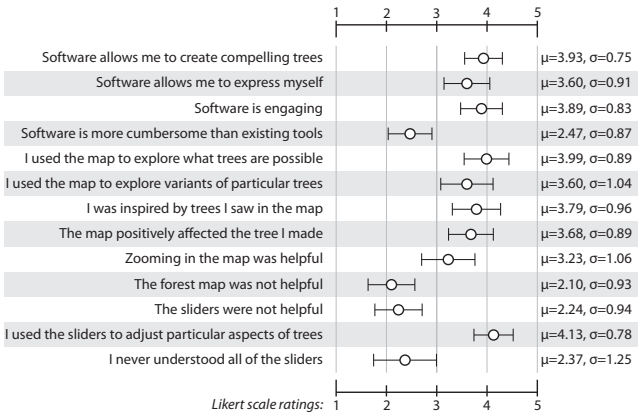


Figure 7: *Survey results from 72 users of our software. Error bars indicate $\frac{1}{2}$ standard deviation in each direction.*

and 5 = 'strongly agree'. We received complete responses from 72 users; the results of their evaluations are summarized in Figure 7.

The responses seem to largely validate our technique, with a majority of users indicating that the prototype allowed them to create compelling models and was less cumbersome than existing tools. Especially encouraging was the degree to which users found the software allowed them to express their personal sense of aesthetics and provide an engaging and even inspiring experience. The survey also reinforced our hypothesis that users would use the map navigation interface to explore the space and the slider controls to make fine-detail changes to particular trees.

Our final evaluation was to analyze the collected log information to determine the modeling times for some of the more visually appealing and complex trees submitted to the database. We measured this quantity as the time elapsed between the start of a user's session and the time the tree was submitted to the database. The average creation time for the first 16 trees presented in Figure 9 was 11.6 minutes; more detailed timing information is given in the caption. It is important to note that these numbers are not from a controlled setting: all trees shown were created entirely "in the wild."

9 Discussion and Future Work

The ability to learn the structure of a parametric design space leads to simple interactive tools for exploring the space and creating models. One area for improvement is the way we represent the navigational manifold. If the manifold has complex folds and singularities, the method we present may have problems. Additionally, adaptive density estimation techniques may be more effective than kernel density estimation if the density of samples varies widely. Finally, we normalize the distance in each dimension using the standard deviation of that dimension. Some machine learning techniques, in contrast, first learn the distance metric, and then extract the structure.

One area that we are actively investigating is the applicability of this technique to other domains. We have outlined the general characteristics of a parametric model necessary for our technique to work. Parametric models of faces, bodies, creatures, vehicles, furniture all deserve investigation. More challenging are models that have a combinatorial structure, such as L-systems and recursive procedural models. Whether the presented techniques can be applied to these more complex models is an open question.

Finally, we are very excited by the opportunities of collaborative modeling. As the Web has taught us, distributed collaboration infrastructure will catalyze exciting new approaches to solving difficult problems. We are eager to add support for ratings, recommendations, tagging, and other community filtering mechanisms. Ratings and recommendations will allow us to associate a quality metric with each model, which will improve the density estimation and manifold representation. Tagging will add semantic information, enabling new ways of searching and clustering models. Our hope is that these technologies can be used to build a new generation of 3D modeling tools.

References

ADOMAVICIUS, G., AND TUZHILIN, A. 2005. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *Transactions on Knowledge and Data Engineering 17*, 6, 734–749.

ALLEN, B., CURLESS, B., AND POPOVIĆ, Z. 2003. The space of human body shapes: reconstruction and parameterization from range scans. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, ACM, New York, NY, USA, 587–594.

ANDERSON, D., FRANKEL, J. L., MARKS, J., AGARWALA, A., BEARDSLEY, P., HODGINS, J., LEIGH, D., RYALL, K., SULLIVAN, E., AND YEDIDIA, J. S. 2000. Tangible interaction + graphical interpretation: a new approach to 3d modeling. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 393–402.

APSIS SECURITY. 2007. *Pound*. `http://www.apsis.ch/pound`.

ARENDASH, D. 2004. The Unreal editor as a Web 3D authoring environment. In *Web3D '04: Proceedings of the ninth international conference on 3D Web technology*, ACM, New York, NY, USA, 119–126.

BLANZ, V., AND VETTER, T. 1999. A morphable model for the synthesis of 3d faces. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 187–194.

CHOY, L., INGRAM, R., QUIGLEY, O., SHARP, B., AND WILLMOTT, A. 2007. Rigblocks: player-deformable objects. In *SIGGRAPH '07: ACM SIGGRAPH 2007 sketches*, ACM, New York, NY, USA, 83.

COOK, R. L. 1986. Stochastic sampling in computer graphics. *ACM Trans. Graph. 5*, 1, 51–72.

DEUSSEN, O., AND LINTERMANN, B. 2005. *Digital Design of Nature*. Springer.

FURUKAWA, Y., AND PONCE, J. 2007. Accurate, dense, and robust multiview stereopsis. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1–8.

GOLDBERG, D., NICHOLS, D., OKI, B. M., AND TERRY, D. 1992. Using collaborative filtering to weave an information tapestry. *Commun. ACM 35*, 12, 61–70.

HAYS, J., AND EFROS, A. A. 2007. Scene completion using millions of photographs. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, ACM, New York, NY, USA, 4.

HONDA, H. 1971. 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*, 331–338.

HWACI. 2007. *SQLite software*. `http://www.sqlite.org`.

IGARASHI, T., MATSUOKA, S., AND TANAKA, H. 1999. Teddy: a sketching interface for 3d freeform design. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 409–416.

KARPENKO, O. A., AND HUGHES, J. F. 2006. Smoothsketch: 3d free-form shapes from complex sketches. *ACM Trans. Graph. 25*, 3, 589–598.

LALONDE, J.-F., HOIEM, D., EFROS, A. A., ROTHER, C., WINN, J., AND CRIMINISI, A. 2007. Photo clip art. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, ACM, New York, NY, USA, 3.

LIKERT, R. 1932. A technique for the measurement of attitudes. *Archives of Psychology 140*, 1–55.

LINDEN, G., SMITH, B., AND YORK, J. 2003. Amazon.com recommen-

dations: item-to-item collaborative filtering. *IEEE Internet Computing 7*, 1, 76–80.

LINTERMANN, B., AND DEUSSEN, O. 1998. A modelling method and user interface for creating plants. *Computer Graphics Forum 17*, 1, 73–82.

LLOYD, S. 1982. Least squares quantization in pcm. *Information Theory, IEEE Transactions on 28*, 2 (March), 129–137.

MARINO, P. 2004. *3D Game-Based Filmmaking: The Art of Machinima*. Paraglyph.

MARKS, J., ANDALMAN, B., BEARDSLEY, P. A., FREEMAN, W., GIBSON, S., HODGINS, J., KANG, T., MIRTICH, B., PFISTER, H., RUML, W., RYALL, K., SEIMS, J., AND SHIEBER, S. 1997. Design galleries: a general approach to setting parameters for computer graphics and animation. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 389–400.

MATUSIK, W., PFISTER, H., BRAND, M., AND MCMILLAN, L. 2003. A data-driven reflectance model. *ACM Trans. Graph. 22*, 3, 759–769.

MATUSIK, W., ZWICKER, M., AND DURAND, F. 2005. Texture design using a simplicial complex of morphable textures. *ACM Trans. Graph. 24*, 3, 787–794.

MILLER, G. 2007. The promise of parallel universes. *Science 317*, 1341–1343.

MĚCH, R., AND PRUSINKIEWICZ, P. 1996. Visual models of plants interacting with their environment. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 397–410.

NEALEN, A., IGARASHI, T., SORKINE, O., AND ALEXA, M. 2007. Fibermesh: designing freeform surfaces with 3d curves. *ACM Trans. Graph. 26*, 3, 41.

NEUBERT, B., FRANKEN, T., AND DEUSSEN, O. 2007. Approximate image-based tree-modeling using particle flows. *ACM Trans. Graph. 26*, 3, 88.

NEWMAN, G. 2006. *Garry's Mod software, version 10*. `http://www.garrysmod.com`.

NGAN, A., DURAND, F., AND MATUSIK, W. 2006. Image-driven navigation of analytical BRDF models. In *Proceedings of the Eurographics Symposium on Rendering*, 389–400.

NGO, T., CUTRELL, D., DANA, J., DONALD, B., LOEB, L., AND ZHU, S. 2000. Accessible animation and customizable graphics via simplicial configuration modeling. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 403–410.

OKABE, M., OWADA, S., AND IGARASHI, T. 2005. Interactive design of botanical trees using freehand sketches and example-based editing. *Computer Graphics Forum 24*, 3, 487–496.

OPPENHEIMER, P. E. 1986. Real time design and animation of fractal plants and trees. *SIGGRAPH Comput. Graph. 20*, 4, 55–64.

PRUSINKIEWICZ, P., AND LINDENMAYER, A. 1990. *The algorithmic beauty of plants*. Springer-Verlag New York, Inc.

PRUSINKIEWICZ, P., JAMES, M., AND MĚCH, R. 1994. Synthetic topiary. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 351–358.

PRUSINKIEWICZ, P., MÜNDERMANN, L., KARWOWSKI, R., AND LANE, B. 2001. The use of positional information in the modeling of plants. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 289–300.

QUAN, L., TAN, P., ZENG, G., YUAN, L., WANG, J., AND KANG, S. B. 2006. Image-based plant modeling. *ACM Trans. Graph. 25*, 3, 599–604.

RASKUTTI, B., AND LECKIE, C. 1999. An evaluation of criteria for measuring the quality of clusters. In *IJCAI '99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 905–910.

RECHE-MARTINEZ, A., MARTIN, I., AND DRETTAKIS, G. 2004. Volumetric reconstruction and interactive rendering of trees from photographs. *ACM Trans. Graph. 23*, 3, 720–727.

SHAW, Z. 2007. *Mongrel library*. `http://mongrel.rubyforge.org`.

SHLYAKHTER, I., ROZENOER, M., DORSEY, J., AND TELLER, S. 2001. Reconstructing 3d tree models from instrumented photographs. *IEEE Comput. Graph. Appl. 21*, 3, 53–61.

SNAVELY, N., SEITZ, S. M., AND SZELISKI, R. 2006. Photo tourism: exploring photo collections in 3d. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, ACM, New York, NY, USA, 835–846.

SU, Q., PAVLOV, D., CHOW, J.-H., AND BAKER, W. C. 2007. Internet-scale collection of human-reviewed data. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, ACM, New York, NY, USA, 231–240.

TAN, P., ZENG, G., WANG, J., KANG, S. B., AND QUAN, L. 2007. Image-based tree modeling. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, ACM, New York, NY, USA, 87.

TENENBAUM, J. B., DE SILVA, V., AND LANGFORD, J. C. 2000. A global geometric framework for nonlinear dimensionality reduction. *Science 290*, 5500 (December), 2319–2323.

TROLLTECH INC. 2007. *Qt: Cross-Platform Rich Client Development Framework*. `http://www.trolltech.com/products/qt`.

VAN DEN HENGEL, A., DICK, A., THORMÄHLEN, T., WARD, B., AND TORR, P. H. S. 2007. Videotrace: rapid interactive scene modelling from video. *ACM Trans. Graph. 26*, 3, 86.

VON AHN, L., AND DABBISH, L. 2004. Labeling images with a computer game. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM, New York, NY, USA, 319–326.

WEBER, J., AND PENN, J. 1995. Creation and rendering of realistic trees. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 119–128.

WEIL, W., AND WIEACKER, J. A. 1993. Stochastic geometry. In *Handbook of Convex Geometry*, P. M. Gruber and J. M. Wills, Eds. North Holland, 1391–1431.

Figure 8: *Nineteen trees that were used as initial landmarks for the parametric tree space.*

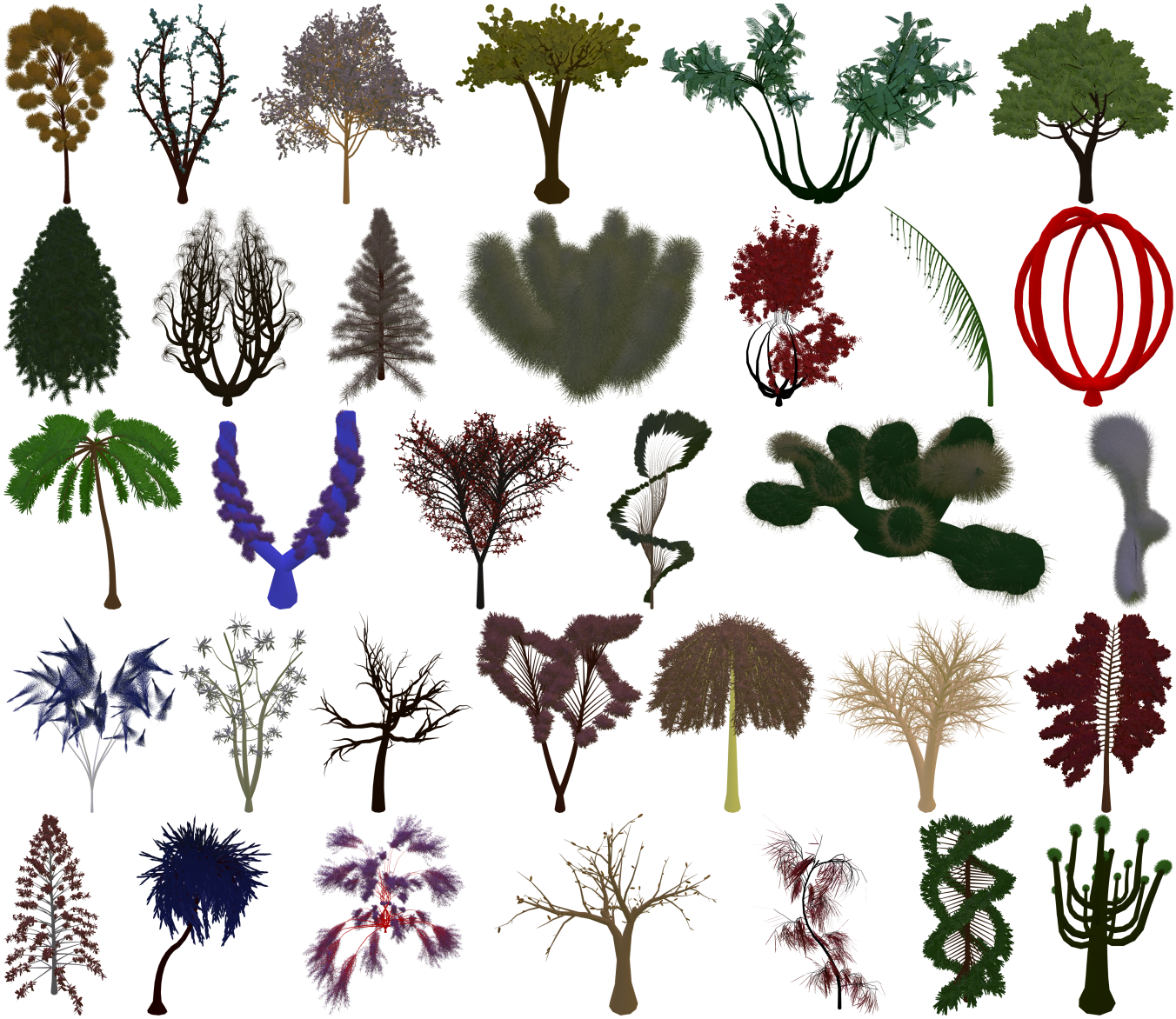


Figure 9: *Trees created by users of our modeling software within the first month of release. Trees in the top row were created in under 6 minutes. Trees in the second row were modeled in under 17 minutes.*