Metropolis Procedural Modeling

JERRY O. TALTON, YU LOU, STEVE LESSER, and JARED DUKE
Stanford University
RADOMÍR MĚCH
Adobe Systems
and
VLADLEN KOLTUN
Stanford University

Procedural representations provide powerful means for generating complex geometric structures. They are also notoriously difficult to control. In this article, we present an algorithm for controlling grammar-based procedural models. Given a grammar and a high-level specification of the desired production, the algorithm computes a production from the grammar that conforms to the specification. This production is generated by optimizing over the space of possible productions from the grammar. The algorithm supports specifications of many forms, including geometric shapes and analytical objectives. We demonstrate the algorithm on procedural models of trees, cities, buildings, and Mondrian paintings.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling

General Terms: Algorithms, Design

Additional Key Words and Phrases: Procedural modeling, geometry synthesis, context-free grammars, Markov chain Monte Carlo

ACM Reference Format:

Talton, J. O., Lou, Y., Lesser, S., Duke, J., Měch, R., and Koltun, V. 2011. Metropolis procedural modeling. ACM Trans. Graph. 30, 2, Article 11 (April 2011), 14 pages.
DOI = 10.1145/1944846.1944851
http://doi.acm.org/10.1145/1944846.1944851

This work was funded in part by NSF grants SES-0835601 and CCF-0641402, and by Adobe Systems.

Authors' addresses: J. O. Talton (corresponding author), Y. Lou, S. Lesser, and J. Duke, Department of Computer Science, Stanford University, 353 Serra Mall, Stanford, CA 94305; email: jtalton@cs.stanford.edu; R. Měch, Adobe Sytems; V. Koltun, Department of Computer Science, Stanford University, 353 Serra Mall, Stanford, CA 94305.

DOI 10.1145/1944846.1944851
http://doi.acm.org/10.1145/1944846.1944851

1. INTRODUCTION

Compelling 3D content is a prerequisite for the development of immersive games, movies, and virtual worlds. Unfortunately, the creation of high-quality 3D models is a notoriously difficult task, often requiring hundreds of hours of skilled labor. This problem is exacerbated in models that exhibit fine detail at multiple scales, such as those commonly encountered in biology and architecture.

Procedural modeling encompasses a class of powerful techniques for generating complex structures from a small set of formal rules. Intricate phenomena can be simulated by repeatedly applying the rules to each generated component of the structure. As a result, procedural representations have been used to model plants and trees, landscapes, ecosystems, cities, buildings, and ornamental patterns [Prusinkiewicz and Lindenmayer 1990; Wong et al. 1998; Deussen et al. 1998; Parish and Müller 2001; Ebert et al. 2002; Müller et al. 2006].

Some of the most common procedural representations are based on formal grammars, such as L-systems [Lindenmayer 1968] and shape grammars [Stiny and Gips 1971]. These languages consist of an alphabet of symbols, an initial symbol, and a set of rewriting rules. Each generated symbol encodes a set of geometric commands, which are executed to produce complex shapes [Prusinkiewicz 1986].

The power of procedural representations lies in their parsimonious expression of complicated phenomena. Unfortunately, controlling these representations is often difficult. Models based on formal grammars, in particular, tend to be "ill-conditioned," in that making slight alterations to the grammar or its parameters can result in global and unanticipated changes in the produced geometry.

The primary contribution of this article is an algorithm for controlling grammar-based procedural models. Given any parametric, stochastic, conditional, context-free grammar, the algorithm takes a high-level specification of the desired model and computes a production from the grammar that matches the specification. No interaction with the grammar itself is required, and the input specification can take many forms, such as a sketch, a volumetric shape, or an analytical objective.

The key idea behind our approach is to formulate modeling operations as probabilistic inference problems over the space of productions from the grammar. Given a high-level specification of the desired model, we define an objective function that quantifies the similarity between a given production and the specification. Our goal is to optimize over the space of productions and find one that maximizes this objective.

Since the space of productions may have complex, trans-dimensional structure, this problem is generally not amenable to traditional optimization techniques. A natural solution is to employ

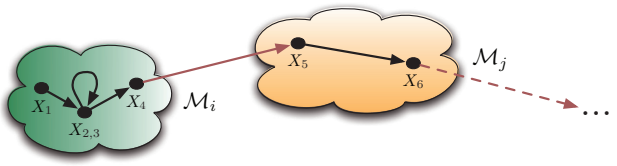


Fig. 1. The evolution of a trans-dimensional Markov chain. The chain alternates between diffusion moves within fixed-dimensional subspaces (black arrows) and jump moves between subspaces of differing dimension (red arrows).

Markov chain Monte Carlo (MCMC) methods, which reduce optimization to sampling. However, the classical MCMC formulation, due to Metropolis et al. [1953] and Hastings [1970], requires the function of interest to be defined over a fixed-dimensional space. In contrast, the dimensionality of productions from a given grammar can vary widely, since applying different rewriting rules to a particular symbol may result in strings of differing lengths. Therefore, we employ a generalization of traditional MCMC called Reversible jump Markov chain Monte Carlo [Green 1995], which allows the Markov chain to "jump" between spaces of varying dimension (see Figure 1).

Our algorithm for controlling grammar-based procedural models has surprisingly simple form, combining rigorous guarantees of correctness and convergence with implementation costs comparable to greedy local search. We demonstrate the algorithm on models of trees, cities, buildings, and Mondrian paintings.

2. RELATED WORK

Grammar-Based Modeling. Grammar-based procedural models were introduced to the graphics community by Prusinkiewicz [1986]. Early efforts demonstrated the utility of grammars for representing complex biological structures [Prusinkiewicz and Lindenmayer 1990]. Further research has resulted in the development of some of the most physically faithful growth models for plants, incorporating interaction with the surrounding environment and biological mechanisms such as competition for light and space [Prusinkiewicz et al. 1994; Měch and Prusinkiewicz 1996; Palubicki et al. 2009]. Grammars have also proven effective at representing man-made structures, such as ornamental patterns [Wong et al. 1998], street networks [Parish and Müller 2001], and building façades [Müller et al. 2006].

Outside computer graphics, grammar-based procedural models have seen extensive use in architecture and design. Shape grammars have been developed to describe a variety of architectural styles, including the villas of Andrea Palladio [Stiny and Mitchell 1978], Frank Lloyd Wright's prairie houses [Koning and Eizenberg 1981], Christopher Wren's city churches [Buelinckx 1993], and the Medina of Marrakech [Duarte et al. 2007]. Shape grammars have also been employed in the product design literature to describe artifacts such as chairs [Knight 1980], coffee makers [Agarwal and Cagan 1998], and motorcycles [Pugliese and Cagan 2002].

Markov Chain Monte Carlo. MCMC methods were first developed by Metropolis et al. [1953] to solve equations of state involving the Boltzmann distribution, and later extended by Hastings [1970] to allow sampling from more general functions. MCMC techniques have been used in a number of computer graphics applications. Szeliski and Terzopoulos [1989] applied stochastic relaxation to synthesize realistic terrain models that conform to a given input. Veach and Guibas [1997] applied the Metropolis-Hastings algorithm to light transport. Chenney and Forsyth [2000] used MCMC to generate rigid-body simulations that satisfy constraints. We extend this line of work to grammar-based procedural modeling.

Outside of graphics, a number of researchers have applied Markov chain Monte Carlo techniques to grammar-based modeling problems. Cagan and Mitchell [1993] discussed the application of MCMC to guiding derivations from shape grammars. Alegre and Dellaert [2004] parsed images of building façades using a variant of trans-dimensional MCMC that relies on an exhaustive search for parameter settings. Ripperda and Brenner [2006, 2009] similarly used RJMCMC and specialized, symmetric split-grammars to recognize building façades from image data. Schlecht et al. [2007] used a hand-tuned Reversible jump formulation to fit a particular L-system describing a species of *Altenaria* fungus to layered microscopy data. Our work unifies and generalizes this line of research, resulting in a robust algorithm that can be used on the complex geometric structures encountered in computer graphics.

3. CONTEXT-FREE GRAMMARS

In order to describe the probabilistic inference formulation, we must first define some basic concepts related to procedural grammars. A parametric, stochastic, conditional, context-free grammar is a tuple

$$G = \langle V, T, \Sigma, \omega, P \rangle,$$

where V is the set of *variables*, T is the set of *terminals*, Σ is the set of *formal parameters*, $\omega \in ((V \cup T) \times \mathbb{R}^*)^+$ is the *axiom*, and $P \subset (V \times \Sigma^* \times \mathcal{B} \times \mathcal{E}) \times ((V \cup T) \times \mathcal{E}^*)^*$ is a finite set of *productions*. Here $\mathcal{E}$ is the set of all correctly constructed arithmetic expressions with parameters from Σ and $\mathcal{B}$ is the set of all possible boolean expressions involving quantities from $\mathcal{E}$ [Prusinkiewicz and Hanan 1990]. Each production $\rho \in P$ is written as $\psi \rightarrow \chi$, where $\psi \in (V \times \Sigma^* \times \mathcal{B} \times \mathcal{E})$ is called the *predecessor*, and $\chi \in ((V \cup T) \times \mathcal{E}^*)^*$ is the *successor*. A simple example grammar is shown in Grammar 1.

Each production $p \in P$ has an associated *conditional expression* from $\mathcal{B}$ that is used to determine if the production is valid for a given set of parameters and an associated *probability expression* from $\mathcal{E}$ that controls the likelihood of p occurring in a derivation. In order to determine which production to apply for a particular variable, the set of productions with matching predecessors are found, and those with false conditionals are discarded. The probability expressions of the remaining productions are evaluated and normalized to create a discrete probability distribution, which is then sampled to choose a production.

Simple Branching Model

$$V = \{X\},\ T = \{F, +, -, [,]\},\ \Sigma = \{\ell\},\ \omega = X(1),$$

$$P = \left\{ \begin{array}{l} X(\ell) : \ell \leq 3 \xrightarrow{1-\ell/3} F\,(\mathcal{N}(\ell, .2))\,[-X(\ell+1)][+X(\ell+1)], \\ X(\ell) : \ell \leq 3 \xrightarrow{\ell/3} F\,(\mathcal{N}(\ell, .2)) \end{array} \right\}$$

Grammar 1. An example grammar describing a simple branching model. The terminal F has a single descriptive parameter sampled from a normal distribution $\mathcal{N}(\mu, \sigma^2)$ that governs the length of each branch.

Additionally, in our formulation, each terminal symbol $t \in T$ is associated with a set of real-valued *descriptive parameters* $\varphi_t = \{\phi_i\}$, drawn from a symbol-specific probability distribution Φ_t. These parameters are used to describe randomness in the model that

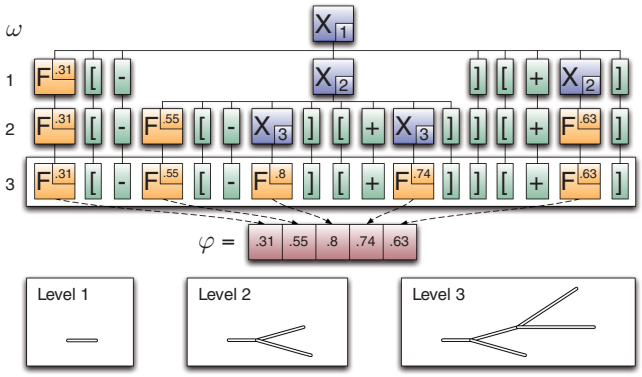


Fig. 2. A random derivation tree from Grammar 1. Nonterminals are shown in blue, terminals with descriptive parameters in gold, and terminals without parameters in green. The formal and descriptive parameter values are inset in the bottom and top of each symbol, respectively. The parameter vector φ corresponding to the derived string is shown in red.

cannot be naturally expressed in terms of the grammar's stochastic productions. Since each expression from $\mathcal{E}(\Sigma)$ is deterministic, the descriptive parameters facilitate the introduction of continuous variation to the model without exploding the number of productions: for instance, varying the length of the terminal branches in a tree without requiring a separate production for each value. Each multivariate distribution Φ is a product of univariate distributions governing the individual descriptive parameters; in our formulation, these univariate distributions may be uniform, Gaussian, or a mixture of Gaussians, and are allowed to depend on expressions from $\mathcal{E}$.

Furthermore, we define the set $\Delta(G)$ of all possible derivations from G: this is the space in which we will perform inference. Each derivation $\delta = (\tau, \varphi)$ from the grammar is uniquely represented by a derivation tree τ and the set of descriptive parameters $\varphi = \bigcup_t \varphi_t$ associated with the terminal symbols in the derived string. Each τ is rooted on ω, and every subsequent level in τ consists of a string produced by applying a matching production to each variable in the level above. The tree terminates once all variables in the string have been reduced to terminals (or when a global depth limit has been reached), and the last level of the tree contains the string representing the produced model $\delta_{\Rightarrow}$. A random derivation tree from Grammar 1 is shown in Figure 2.

Note that we define $\Delta(G)$ in terms of derivation trees, and not the strings that these trees produce. Many stochastic context-free grammars are ambiguous: for some strings, there may exist many valid derivations. By searching for the optimal derivation tree instead of the optimal string, we do not need to account for this ambiguity in the probability calculations, greatly simplifying the inference process.

4. PROBABILISTIC INFERENCE FOR GRAMMARS

Given a grammar G and a user-provided specification I describing a desired model, we would like to find the best derivation from G that matches the user's input. We formulate this as a probabilistic inference problem over the space of possible derivations $\Delta(G)$. Define a posterior

$$p(\delta|I) \propto L(I|\delta)\pi(\delta), \qquad (\star)$$

where $\delta \in \Delta(G)$, $L(\cdot|\cdot)$ is the likelihood of the input given a particular model, and $\pi(\cdot)$ is the model prior. Finding the best derivation from the grammar that matches the provided specification then reduces to maximizing $(\star)$.

The difficulty of performing this maximization is closely tied to the form of $p(\cdot|\cdot)$. The model prior $\pi(\cdot)$ is drawn directly from the grammar's stochastic production probabilities and descriptive parameter distributions, so that

$$\pi(\delta) \propto \prod_{s\in\delta} P\,(s|\text{parent}(s)) \prod_{t\in\delta_{\Rightarrow}} \Phi_t(\varphi_t),$$

where s ranges over all the successors contained within δ, and $P(\cdot|\cdot)$ is the production probability given in the grammar. In this way, the prior $\pi(\delta)$ is proportional to the probability of δ being randomly drawn from G.

The definition of $L(\cdot|\cdot)$, however, necessarily depends on the form of the user's provided specification: different likelihood functions must be developed for each of the different modes of control. The described inference procedure is largely agnostic to the likelihood formulation: virtually any function can be used, provided it can be evaluated efficiently.

In general, given the size of $\Delta(G)$ and the range of desirable specifications, it is unreasonable to expect analytic solutions to the inference problem to exist. Therefore, we use Markov chain Monte Carlo techniques to efficiently maximize the posterior.

5. INTRODUCTION TO MCMC INFERENCE

MCMC methods have a long history in computational statistics and machine learning. They operate by simulating a Markov chain that takes a random, memoryless walk through a space of interest and generates a set of correlated, unbiased samples from a given function defined over that space. These samples can then be used to efficiently solve integration or optimization problems involving this function. In fact, there are many high-dimensional problems for which MCMC techniques provide the only known efficient algorithms: for instance, computing the volume of a convex body in n dimensions, sampling from truncated multivariate Gaussians, or computing the permanent of a matrix [Andrieu et al. 2003].

A Markov chain $\mathbf{X}$ is a sequence of random variables $X_1, X_2, \ldots$ in a domain $\mathcal{X}$ which possess the Markov property $P(X_{n+1} = x_{n+1}|X_n = x_n, \ldots, X_1 = x_1) = P(X_{n+1} = x_{n+1}|X_n = x_n)$; that is, the value of the next sample in the chain depends only on the value of the sample that immediately precedes it. When the chain is constructed in such a way that these samples come from a particular probability distribution $p(\cdot)$, the chain is said to be *ergodic*, and $p(\cdot)$ is called the chain's *stationary distribution*. By running the chain for a sufficiently long time, we can use the generated samples to solve expected value problems of the form $E[f(x)] = \int_{\mathcal{X}} f(x)p(x)\,dx \approx \frac{1}{N}\sum_{i=1}^{N} f(X_i)$ and inference problems like $\max_{\mathcal{X}} p(x) \approx \text{argmax}_i\, X_i$.

5.1 The Metropolis-Hastings Algorithm

The most popular Markov chain Monte Carlo formulation is due to Metropolis et al. [1953] and Hastings [1970]. Given a state space $\mathcal{X}$ and a nonnegative function $p : \mathcal{X} \to \mathbb{R}^+$, the Metropolis-Hastings (MH) algorithm constructs an ergodic Markov chain whose stationary distribution is precisely p. In essence, the algorithm transforms the ability to evaluate a function at a point into an efficient method for sampling from that function. Unlike most other sampling methods, the MH algorithm does not require that p integrate to one, making it extremely useful for problems where $\mathcal{X}$ is so large as to make computing the normalizing constant for p infeasible.

The MH algorithm works as follows. First, X_1 is set to some arbitrary point in $\mathcal{X}$. To determine the value of X_{i+1} at each step, a "tentative" sample x^* is generated from a proposal density function $q(x|x_i)$. In theory, q can be *any* function that can be sampled directly and whose support includes the support of p. In practice, the proposal density must be carefully designed to match the shape of p as much as possible, in order to ensure good performance. This tentative sample $x^* \sim q(\cdot|x_i)$ is either accepted into the chain as x_{i+1} with probability

$$\alpha_{x_i \to x^*} = \min\left\{1, \frac{p(x^*)}{p(x_i)} \frac{q(x_i|x^*)}{q(x^*|x_i)}\right\},$$

or rejected, in which case $x_{i+1} = x_i$. It can be shown that this construction will produce at worst asymptotic convergence to $p(\cdot)$ regardless of the initial choice of x_1, and will often converge geometrically if the target domain is not too large [Tierney 1994].

6. REVERSIBLE JUMP MCMC

Unfortunately, the traditional Metropolis-Hastings algorithm breaks down when the dimensionality of the samples in the chain is allowed to vary. Intuitively, this is because comparing the densities of objects in different dimensions has no meaning, and the acceptance probability formulation is no longer valid. This is particularly confounding for applying MCMC techniques to procedural modeling problems, since different derivations from a single grammar may have drastically different lengths.

To overcome this problem, we employ a generalization of the traditional MH framework called Reversible jump MCMC [Green 1995]. RJMCMC works by supplementing the parameter-changing *diffusion* moves of MH with an additional set of dimension-altering *jump* moves, which allow the chain to move between subspaces of varying dimension. For instance, a "death" move could be used to remove a component from a statistical model and decrease its overall dimensionality, while a "split" move might replace an existing component of a model with two new components, increasing the dimension accordingly.

While the traditional MH algorithm can only be used for parameter fitting, the flexibility these jump moves impart makes RJMCMC a useful tool for solving model selection problems. For example, while traditional MCMC techniques could be used to fit a set of k Gaussians to a collection of points, the value of k must be known a priori. With RJMCMC, we can construct a chain in which k itself is allowed to vary, and use it to find both the optimal number of Gaussians as well as their means and variances. Essentially, RJMCMC is a method for inference problems in which "the number of things you don't know is one of the things you don't know" [Hastie and Green 2009].

6.1 Reversibility

Central to the application of RJMCMC techniques is the definition of the set of dimension-altering jump moves. Suppose we have a countable family of statistical models $\{\mathcal{M}_m\}$, each with an associated set of k_m parameters $\mathbf{x}_m \in \mathcal{X}_m \subseteq \mathbb{R}^{k_m}$. To guarantee ergodicity, each jump between these models must be carefully formulated to satisfy several strict theoretical requirements. First and foremost, each move must be reversible: for any move from a model m and associated parameters $\mathbf{x}_m$ to a model $(n, \mathbf{x}_n)$, there must also exist a reverse move from $(n, \mathbf{x}_n)$ back to $(m, \mathbf{x}_m)$. This is necessary to ensure that the chain does not get stuck in any particular subspace of the target domain $\mathcal{X} = \bigcup_m \{m\} \times \mathcal{X}_m$.

6.2 Dimension Matching

In addition, to perform a jump from a model m to a model n, a deterministic, differentiable, invertible *dimension matching* function

$$f_{m\to n} : \mathcal{X}_m \times \mathcal{U}_{m,n} \to \mathcal{X}_n \times \mathcal{U}_{n,m}$$

must be defined, where $\mathcal{U}_{m,n} \subseteq \mathbb{R}^{r_m}$ and $\mathcal{U}_{n,m} \subseteq \mathbb{R}^{r_n}$ with $k_m + r_m = k_n + r_n$. The precise formulation of these dimension matching functions is "rather obscure" [Green 1995, Section 3.2]. However, the intuition behind them is that we are extending both models into a common, intermediary parameter space in which their densities can be meaningfully compared. To do this, we supplement the parameters of the models to make their dimensions match using two sets of additional parameters $\mathbf{u}_{m,n}$ and $\mathbf{u}_{n,m}$ drawn from proposal density functions $q_{m\to n}(\cdot|m, \mathbf{x}_m)$ and $q_{n\to m}(\cdot|n, \mathbf{x}_n)$. Then, we can define $f_{m\to n}$ such that $f_{m\to n}(\mathbf{x}_m, \mathbf{u}_{m,n}) = (\mathbf{x}_n, \mathbf{u}_{n,m})$ and $f_{m\to n}(f_{n\to m}(\mathbf{x}_n, \mathbf{u}_{n,m})) = (\mathbf{x}_n, \mathbf{u}_{n,m})$. In many RJMCMC formulations, one of r_m or r_n is set to zero.

6.3 Acceptance Probability

With this machinery in place, the RJMCMC algorithm closely resembles traditional MH. If the current state of the chain is $(n, \mathbf{x}_n)$, we perform a jump to $(m, \mathbf{x}_m)$ by generating $\mathbf{u}_{n,m} \sim q_{n\to m}(\cdot|n, \mathbf{x}_n)$, setting $(\mathbf{x}^*_m, \mathbf{u}_{m,n}) = f_{n\to m}(\mathbf{x}_n, \mathbf{u}_{n,m})$, and accepting $\mathbf{x}^*_m$ into the chain with probability

$$\alpha_{n\to m} = \min\left\{1, \frac{p(m, \mathbf{x}^*_m)}{p(n, \mathbf{x}_n)} \frac{j(n|m)}{j(m|n)} \frac{q_{m\to n}(\mathbf{u}_{m,n}|m, \mathbf{x}^*_m)}{q_{n\to m}(\mathbf{u}_{n,m}|n, \mathbf{x}_n)} \mathcal{J}_{f_{n\to m}}\right\},$$

where $j(\cdot|\cdot)$ is the probability of selecting the jump and $\mathcal{J}_{f_{n\to m}}$ is the Jacobian of the transformation $f_{n\to m}$:

$$\mathcal{J}_{f_{n\to m}} = \left|\det \frac{\partial f_{n\to m}(\mathbf{x}_m, \mathbf{u}_{m,n})}{\partial(\mathbf{x}_m, \mathbf{u}_{m,n})}\right|.$$

Note that this formula clearly illustrates the need for guaranteeing the invertibility of the dimension matching functions, lest the Jacobian vanish and the acceptance probability collapse to zero. As a result, constructing such jump moves for even a single problem domain can be a tricky and time-consuming task. The difficulty of guaranteeing reversibility and dimension matching between models is widely held to be the primary impediment to the widespread adoption of RJMCMC techniques [Andrieu et al. 2003].

7. MCMC FOR GRAMMARS

This article presents a robust method for inferring reversible jump moves for any parametric, stochastic, conditional, context-free grammar. The presented formulation has a number of advantages over previous attempts at using trans-dimensional MCMC in procedural modeling. Unlike the method of Ripperda and Brenner [2009], which requires symmetric shape grammars in order to guarantee reversibility, our method places no restrictions on the form of the underlying grammar. In addition, the presented technique is entirely automatic: jump moves are derived directly from the grammar specification and need not be manually constructed for a particular grammar, as in Schlecht et al. [2007]. Moreover, we provide a simple mechanism for dimension matching that scales to models with hundreds of thousands of continuous parameters, obviating the need for approximate optimization algorithms like the one described by Alegre and Dellaert [2004].

Given an grammar G along with a user-provided model specification I and associated scoring function $L(I|\cdot)$, we initialize the Markov chain to $\delta \equiv (\tau, \varphi) \sim \pi(\cdot)$ by sampling from the prior. At

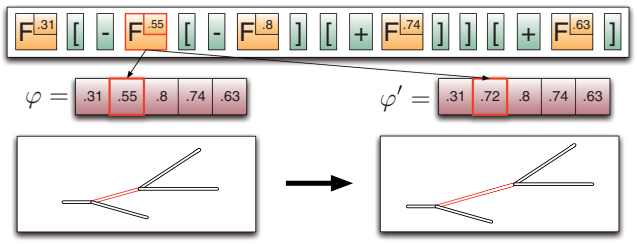


Fig. 3. A derivation undergoing a diffusion move. A random terminal is selected and its descriptive parameters resampled.

every subsequent iteration, we randomly choose to apply either a jump or diffusion move in order to evolve the chain.

7.1 Diffusion Moves

In a diffusion move, the topology of the derivation tree remains fixed, and the descriptive parameters of its derived string are modified (see Figure 3). To perform such a move, we first pick a random terminal t from $\delta_{\Rightarrow}$, and then resample $\varphi_t' \sim \Phi_t$ to generate a new derivation δ'. Since the distribution governing these parameters is specified a priori in the grammar, we use an independence sampler for the proposal density, where $q(\delta'|\delta) = q(\delta') = \Phi_t(\varphi_t')$. The acceptance probability for δ' then simplifies to

$$\alpha_{\delta\to\delta'} = \min\left\{1, \frac{p(\delta'|I)\Phi_t(\varphi_t)}{p(\delta|I)\Phi_t(\varphi_t')}\right\} = \min\left\{1, \frac{L(I|\delta')}{L(I|\delta)}\right\}.$$

7.2 Jump Moves

To derive a complete set of jump moves, we leverage the set of stochastic productions given in the grammar. In particular, to perform a jump on a given derivation (τ, φ), we first randomly select a variable v from within τ, and then rederive the subtree τ_v rooted at v by rerolling productions for v and its children. This new subtree τ_v' is evolved until it consists entirely of terminal symbols or reaches the global depth limit; this process is illustrated in Figure 4.

We now ensure that these jump moves are reversible, and formulate appropriate dimension matching functions for them. To begin, observe that sampling from a given descriptive distribution Φ is equivalent to sampling from one or more uniform distributions U and applying a deterministic transformation to the results [Box and Muller 1958]. Therefore, any parameter vector $\varphi \in \mathbb{R}^q$ has a corresponding representation $\tilde{\varphi} \in [0,1]^w$ with $q \le w$, and sampling $\varphi \sim \Phi$ is equivalent to sampling $\tilde{\varphi} \sim U_{[0,1]}$.

We may now consider jumps between subtrees $(\tau_v, \tilde{\varphi})$ and $(\tau_v', \tilde{\varphi}')$ where $\tilde{\varphi} \in \mathbb{R}^j$ and $\tilde{\varphi}' \in \mathbb{R}^k$. Without loss of generality, let $j \ge k$ and u' be a vector of $j-k$ uniform random numbers in the range $[0,1]$. For the forward move, we define $(\tilde{\varphi}', u') = f_{\tau\to\tau'}(\tilde{\varphi}) = \tilde{\varphi}$. For the reverse, $\tilde{\varphi} = f_{\tau'\to\tau}(\tilde{\varphi}', u') = (\tilde{\varphi}', u')$. Essentially, by putting all parameters on equal footing, we can copy the old values to the new state and reinterpret them accordingly, truncating or supplementing them with new random samples to match the target dimension.

This formulation has several desirable properties. First, it is trivial to implement and highly efficient. Second, it is clearly reversible, since $f_{\tau\to\tau'}(f_{\tau'\to\tau}(\tilde{\varphi}', u')) = (\tilde{\varphi}', u')$. Third, the Jacobians simplify to

$$\mathcal{J}_{f_{\delta\to\delta'}} = \left|\frac{\partial(\varphi', u')}{\partial(\varphi)}\right| = \mathcal{J}_{f_{\delta'\to\delta}} = \left|\frac{\partial(\varphi)}{\partial(\varphi', u')}\right| = 1.$$

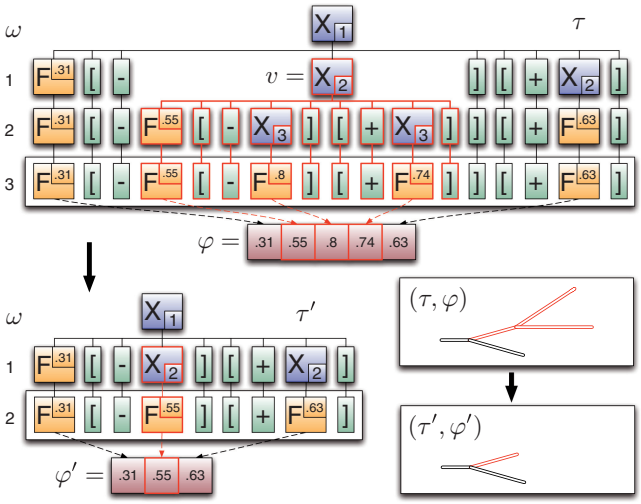


Fig. 4. A derivation tree (τ, φ) undergoing a jump move to (τ', φ'). A random nonterminal symbol v is selected, and the subtree rooted at v is replaced with a different production from Grammar 1. Here the dimension of φ is reduced by the jump.

To compute the final acceptance probability for the move, we first define the jump probability of replacing the selected subtree τ_v with a new subtree τ_v'

$$j(\tau_v'|\tau_v) = q_\tau(v) \prod_{s\in\tau_v'} P(s|\text{parent}(s)),$$

where $q_\tau(v)$ is the probability of selecting a nonterminal v in τ. Then,

$$\begin{aligned}\alpha_{\delta\to\delta'} &= \min\left\{1, \frac{p(\delta'|I)}{p(\delta|I)}\frac{j(\tau_v|\tau_v')}{j(\tau_v'|\tau_v)}\frac{U_{[0,1]}(u')}{1}\mathcal{J}_{\delta\to\delta'}\right\} \\ &= \min\left\{1, \frac{L(I|\delta')\,\pi(\delta')}{L(I|\delta)\,\pi(\delta)}\frac{q_{\tau'}(v)}{q_\tau(v)}\frac{\prod_{s\in\tau_v} P(s|\text{parent}(s))}{\prod_{s\in\tau_v'} P(s|\text{parent}(s))}\right\} \\ &= \min\left\{1, \frac{q_{\tau'}(v)}{q_\tau(v)}\frac{L(I|\delta')}{L(I|\delta)}\frac{\prod_{t\in\tau_v'}\Phi_t(\varphi_t)}{\prod_{t\in\tau_v}\Phi_t(\varphi_t')}\right\}. \qquad (\diamond)\end{aligned}$$

This formula illustrates the behavior of the chain: it randomly walks through the derivation space, generally preferring to move towards models with higher likelihoods. Unlike greedy local search, however, it will occasionally accept worse moves in order to avoid becoming stuck in local maxima. The final algorithm is given in Algorithm 1.

8. OPTIMIZATION

Although this formulation guarantees asymptotic convergence to $p(\cdot|\cdot)$, it is generally impossible to find the true global maximum of the posterior, given the size of $\Delta(G)$ and the complexity of the likelihood formulation. However, we can still obtain useful approximations for many computer graphics applications by establishing a reasonable computational budget and ensuring that the chain *mixes* well: that is, that it explores the derivation space rapidly enough to produce meaningful results. Guaranteeing good mixing is complicated by the fact that the posterior may be highly multimodal: in order to mix, the chain must be able to jump out of local maxima efficiently.

8.1 Nonterminal Selection

One natural way to encourage good mixing in grammar-based inference is to exploit the structure of the derivation tree. While the most straightforward policy for $q_\tau(v)$ is to select a nonterminal uniformly at random from τ, this strategy will generally result in long mixing times given the exponential increase in symbols between successive levels of the derivation. While rederiving the subtree rooted at a nonterminal symbol in the lower levels of a derivation will produce only small changes in the current model, selecting a nonterminal high in the tree will cause a significant portion of the model to be replaced. To ensure that the chain proposes large and small moves in roughly equal measure, we take

$$q_\tau(v) = b^{[\text{depth}(\tau)-\text{depth}(v)]},$$

where is b is an estimate of the grammar's branching factor.

8.2 Parallel Tempering

Several more general methods for improving the mixing of Markov chain Monte Carlo methods have been proposed. The most popular are based on tempering schemes that "melt down" the peaks of the target distribution to minimize the role of local maxima as false attractors and then "cool" back to the original distribution [Marinari and Parisi 1992; Neal 1994]. Unfortunately, these schemes typically require a series of initial simulations to estimate distribution parameters that cannot be known a priori, making them unattractive as components of a general system.

Instead, we employ an optimization scheme called parallel tempering [Geyer 1991]. The idea behind parallel tempering is to run a series of N independent chains at different temperatures; periodically, these chains propose to swap their states via Metropolis updates. We define a geometric progression of N temperature profiles $T_i = 1/t_c^i$ and set one chain to sample from each stationary distribution $p_i(\cdot) = p^{T_i}(\cdot)$. After each iteration (in which all N chains are updated independently) we pick a random index $j \in [1, N-1]$ and propose a Metropolis move to swap the state of chains j and $j+1$. This swap is accepted with probability

$$\alpha_{j \leftrightarrow j+1} = \frac{p_j\left(X_k^{j+1}\right) p_{j+1}\left(X_k^j\right)}{p_j\left(X_k^j\right) p_{j+1}\left(X_k^{j+1}\right)},$$

where X_k^j is the kth iteration of the jth chain. In our implementation, we take $N = 10$ and set $t_c = 1.3$ so that a move which has a 70% acceptance probability in the warmest chain has a 1% probability in the coldest.

In this manner, hot chains (which may freely move out of local maxima and across the space) are able to transfer their information to cold chains (which can efficiently find local maxima) and vice versa. The method is particularly well-suited to our purposes since it requires almost no parameter tuning, and can be parallelized efficiently.

8.3 Delayed Rejection

Ensuring good mixing in Reversible jump MCMC schemes is often more difficult than in their fixed-dimensional counterparts due to a phenomenon called persistent rejection. If the proposal function is poorly calibrated to the target distribution in some parts of the space, most jump moves proposed in those regions will be rejected. Since the dimension matching functions described in Section 7.2 lack strong semantics, our method is occasionally susceptible to this phenomenon. Copying values from one subspace to another may result in unlikely parameter assignments even if the target subspace contains better samples from the distribution of interest.

ALGORITHM 1: The following is the pseudocode for the basic inference algorithm described in Section 7

Metropolis Procedural Modeling
Input: a grammar G, likelihood function $L(I|\delta)$, target specification I, and computational budget N.
Output: a MAP estimate of δ.

Sample $\delta_0 \sim \pi_G$
Initialize $\delta_{max} \leftarrow \delta_0, max \leftarrow p(\delta_0|I)$
for $n = 1$ **to** N
  switch RANDOMMOVE()
  case diffusion
    $t \leftarrow$ RANDOMTERMINAL $((\delta_{n-1})_\Rightarrow)$
    Compute δ_n' by sampling $\varphi_t' \sim \Phi_t$
    $\alpha \leftarrow \min\left\{1, \frac{L(I|\delta_n')}{L(I|\delta_{n-1})}\right\}$
  case jump
    Sample $v \sim q_{\delta_{n-1}}$
    $\tau_v' \leftarrow$ REDERIVE τ_v
    $\delta_n' \leftarrow$ DIMENSIONMATCH τ_v', τ_v
    $\alpha \leftarrow \min\left\{1, \frac{q_{\tau'}(v)}{q_\tau(v)} \frac{L(I|\delta_n')}{L(I|\delta_{n-1})} \frac{\prod_{t \in \tau_v'} \Phi_t(\varphi_t)}{\prod_{t \in \tau_v} \Phi_t(\varphi_t')}\right\}$
  Sample $t \sim [0, 1]$
  if $t < \alpha$ **then** $\delta_n \leftarrow \delta_n'$
    else $\delta_n \leftarrow \delta_{n-1}$
  if $max < p(\delta_n|I)$
    $\delta_{max} \leftarrow \delta_n,\ max \leftarrow p(\delta_n|I)$
return δ_{max}

To mitigate this issue, we employ a technique called delayed rejection [Tierney and Mira 1999]. When a jump from a model δ to δ' is not accepted, we temporarily forestall the rejection and target an auxiliary diffusion move on δ' in an effort to "clean up" the dimension-matched parameters. We then accept or reject the two moves in concert, with a specialized acceptance probability formulated to preserve ergodicity and the Markovian property of the chain [Green and Mira 1999]. In this way, the chain is able to mix well even when jump proposals generate unlikely parameters.

Suppose we propose a jump move in the usual way, choosing a random nonterminal v, rederiving the subtree τ_v, dimension-matching its parameters to produce a new model δ', and computing the acceptance probability $\alpha_{\delta \to \delta'}$. If the move is rejected, we retain δ' instead of discarding it, and resample $\varphi_t' \sim \Phi_t$ for all terminals t in τ_v' to generate a new candidate model δ''. This augmented model is then accepted into the chain with probability

$$\alpha_{\delta \to \delta''} = \min\left\{1, \frac{p(\delta''|I)}{p(\delta|I)} \frac{j(\tau_v|\tau_v'')}{j(\tau_v''|\tau_v)} \frac{[1 - \alpha_{\delta'' \to \delta_*}]}{[1 - \alpha_{\delta \to \delta'}]}\right\},$$

where δ_* is δ with parameters dimension-matched from δ'', and $\alpha_{\delta'' \to \delta_*}$ and $\alpha_{\delta \to \delta'}$ are calculated as in ($\diamondsuit$). In essence, this formulation preserves ergodicity by weighing the utility of a path from $\delta \to \delta' \to \delta''$ against a hypothetical sequence of moves from $\delta'' \to \delta_* \to \delta$, though these latter moves are never actually attempted.

8.4 Annealing

It is also important to remember that the ability to generate unbiased samples from the posterior is not necessarily a prerequisite

for maximizing it. Although the global maximum of $p(\cdot)$ can be estimated by $\text{argmax}_i\ p(X_i)$, this approximation is inefficient for posteriors in which the bulk of probability mass lies far from the modes of the distribution. Instead, we can employ a simple annealing technique to simulate an inhomogeneous Markov chain whose stationary distribution at iteration i is $p^{1/\mathcal{T}_i}(\cdot)$, where $\mathcal{T}_i$ is a decreasing cooling schedule with $\lim_{i\to\infty} \mathcal{T}_i = 0$. Under weak regularity assumptions this chain will concentrate itself on the global maxima of $p(\cdot)$, saving much unnecessary computation [White 1984].

9. IMPLEMENTATION

To evaluate the presented inference framework, we implemented a modular procedural modeling system in the C++ programming language. Grammars are written in a domain-specific language defined by a parsing expression metagrammar, translated into C++ classes by a Ruby frontend, and compiled into dynamic plugin libraries. This architecture allows the inference framework to operate exclusively on primitive data types, an important feature given the high overhead of string operations in Monte Carlo simulations. These plugins can then be loaded by the main application, which implements the inference framework described in Section 7 as well as the optimizations described in Section 8. The framework also exposes an extensible interface for likelihood computations. This simple design allows grammars and targeting functions to be mixed and matched. The implementation and all the grammars used in this article are available online at http://graphics.stanford.edu/projects/mpm.

10. GRAMMARS

We tested the implemented modeling system on stochastic grammars for several classes of procedural models, including trees, buildings, and the paintings of Dutch artist Piet Mondrian. Random derivations from these grammars are shown in Figure 5.

Trees. We developed tree grammars that simulate sympodial (as in the young oak, old oak, willow, and acacia) and monopodial (for the conifer and poplar) growth. To achieve different growth patterns within each branching class, we adjust the probability distributions that control branching angles, tropisms, and leaf patterns. To simulate the effects of competition for light in the absence of Open L-systems [Měch and Prusinkiewicz 1996], we use a budget-based approach. Each tree starts with a budget of branches that is depleted during growth. This budget controls the probability of creating new geometry at each stage of the derivation.

Architecture. We developed two architectural grammars: a spatial subdivision grammar for building generation, and an incremental growth grammar for city modeling. The building grammar can be used to target edifices with complex, irregular mass models. It functions by applying a sequence of split and shrink operations to an initial box. The city grammar is more restrictive, producing a sequence of rectangular buildings laid out in city blocks. To approximate urban sprawl, the size of each building decreases probabilistically with distance from city center.

Mondrian Art. We also constructed a grammar for the distinctive grid-based paintings of Piet Mondrian, a Dutch painter who cofounded the *De Stijl* art movement in early twentieth century. The Mondrian grammar is a probabilistic two-dimensional discrete kd-tree with bounded depth and a random color assigned to each leaf node.

11. LIKELIHOOD FORMULATIONS

We implemented image- and volume-based likelihood functions, as well as a special likelihood function for Mondrian art. Wherever possible, these computations are performed via CUDA on the GPU to exploit parallelism.

11.1 Image- and Volume-Based Modeling

For image- and volume-based modeling, we employ a discretized likelihood formulation which takes as input a target image or voxelization I, and treats each pixel or voxel element as being independently and identically Gaussian distributed. For a particular derivation δ, we rasterize $\delta_{\Rightarrow}$ into a buffer I_δ and calculate

$$\log L(I|\delta) = -\frac{1}{2\sigma^2}\sum_{x\in\mathcal{D}} \|I(x) - I_\delta(x)\|^2,$$

where $\mathcal{D}$ is the domain of I, and σ^2 is a tunable variance parameter. In our implementation, $\sigma^2 \approx .22\sqrt[n]{|\mathcal{D}|}$ where $n = \dim \mathcal{D}$, so that a scanline-sized degradation in model representation results in roughly a 10% acceptance rate in the chain.

11.2 Mondrian Modeling

Piet Mondrian was a Dutch painter in the early twentieth century, famous largely for pioneering a distinctive nonrepresentational art form which he called neoplasticism. Mondrian's iconic paintings depict two-dimensional, axis-aligned subdivisions, with cells colored in yellow, white, red, blue, or black. The paintings are characterized by a careful balance of geometric structure and color, which Mondrian said was led by "high intuition" rather than "calculation." It is therefore difficult to manually construct a set of rules that reproduce Mondrian's style [Andrzejewski et al. 2010].

Many of Mondrian's compositions live within the space of two-dimensional kd-trees: a space that can be compactly described by a stochastic, parametric, context-free grammar. We developed a likelihood function that rewards tree configurations with aesthetic balance, rhythm, and symmetries that are evocative of Mondrian. Distinct modes of this function can then be sampled to produce Mondrian-like paintings.

To construct this function we examined nine of Mondrian's representative works painted between 1920 and 1930 and assembled a set of terms that account for some of the variation within this set. These terms include number of cells, average cell size, average aspect ratio, and average area of horizontally- and vertically-dominant cells. Since grid lines in Mondrian's art tend to align across kd-tree boundaries, we added the minimum nonnegative distance between parallel edges. To account for the distinctive spatial color distributions of Mondrian paintings, we also included the average area occupied by each color in Mondrian's palette and the number of adjacent cells with the same nonwhite color.

The form of the likelihood function follows directly from this set of terms. Given a derivation δ from the grammar, we compute each term s on the derivation and calculate a penalty value

$$D(s_\delta) = \text{erf}\left(\frac{|s_\delta - \mu_s|}{\sigma_s\sqrt{2}}\right),$$

where $\text{erf}(\cdot)$ is the Gauss error function, and μ_s and σ_s are the mean and standard deviation of s in the training data. $L(\delta)$ is defined simply as

$$L(\delta) = \exp\left(-\sum_{s\in S} w_s D(s_\delta)\right),$$

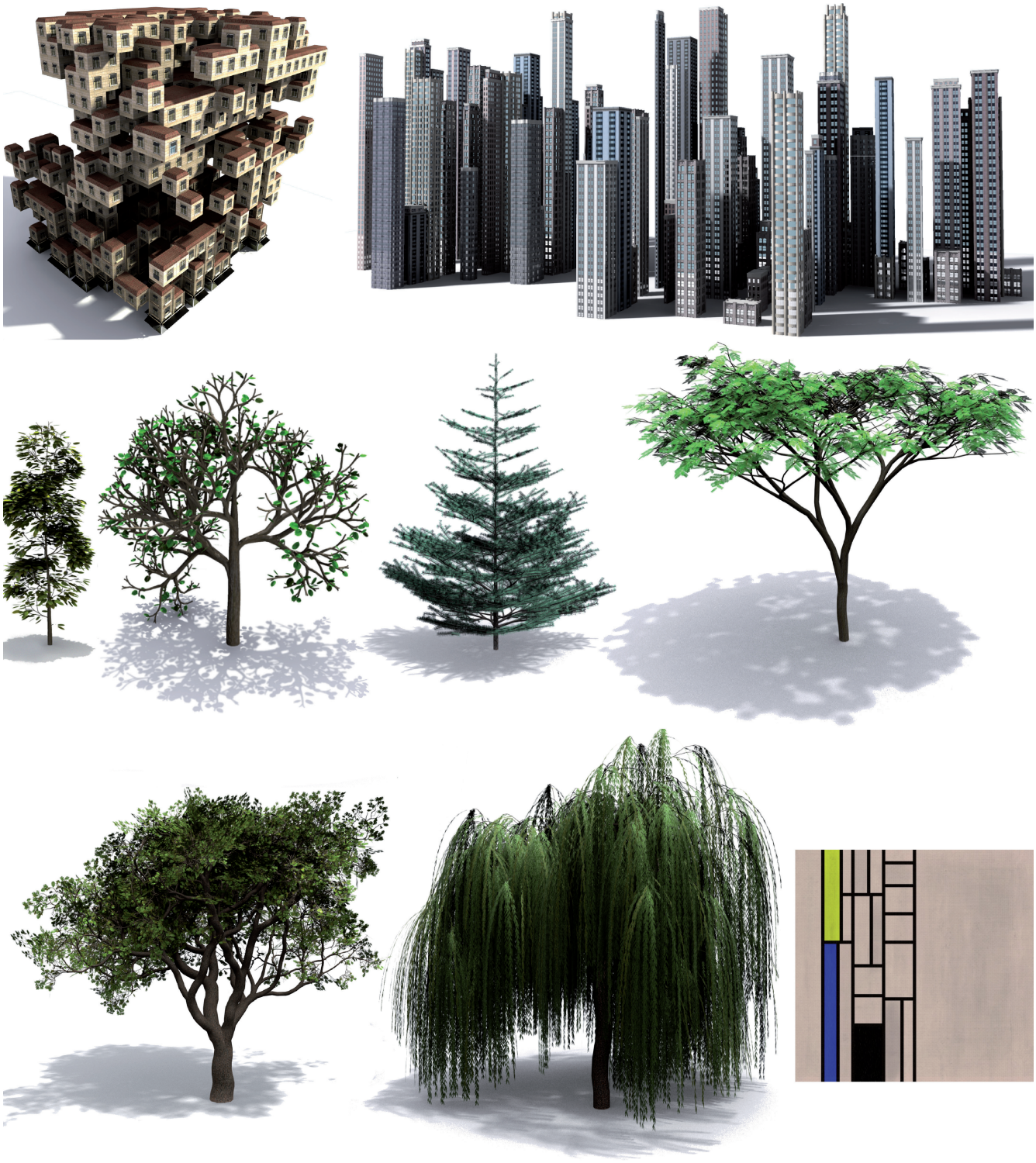


Fig. 5. Illustrations of the models used in this article. From top left to lower right: random derivations from the building, city, poplar, young oak, conifer, acacia, old oak, willow, and Mondrian grammars. Zoom in for detail.

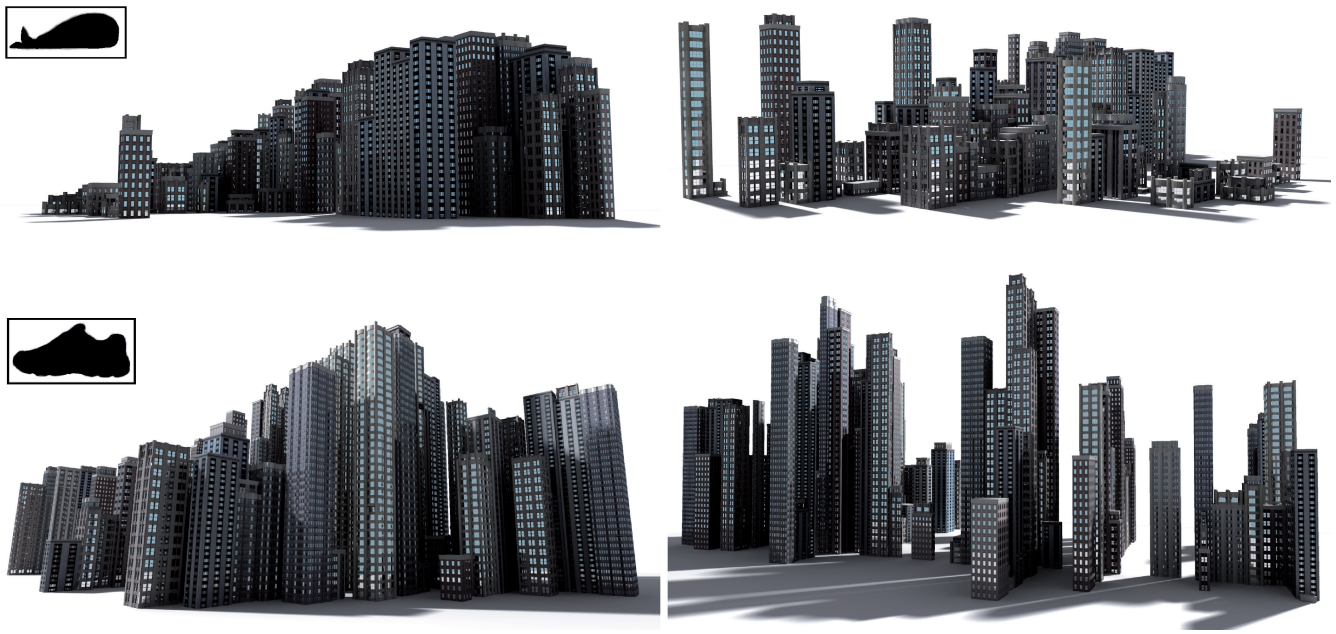


Fig. 6. (left) Derivations from the city grammar targeted to skylines of a whale (top) and shoe (bottom). (right) The same derivations from different viewpoints, highlighting the urban structure enforced by the grammar.

where the weights w_s are set manually to approximate the relative importance of the statistics.

12. RESULTS AND DISCUSSION

Figure 8 shows eight young oak trees targeted to the word *SIGGRAPH* using the image-based likelihood formulation. Figure 9 shows two different grammars (acacia and willow), targeted to identical specifications. Figure 10 shows conifer, old oak, and poplar trees targeted to a set of sketches.

These examples highlight several advantages of the presented technique over prior work. Since our method functions on arbitrary grammars (which are among the most faithful and general representations for botanical structures), it can be used to control tree models with complex, realistic growth patterns, accounting for physical variation in apical dominance and photo- and gravitropism, and rigidly enforcing monopodial and sympodial branching structures. This obviates the need for special-purpose algorithms that only loosely approximate the growth of real trees, such as those described by Neubert et al. [2007] and Chen et al. [2008].

Furthermore, the presented technique is not restricted to grammars based on biological growth. Figure 6 demonstrates sketch-based modeling applied to the city grammar. Figure 11 illustrates volume-based targeting applied to the building grammar. Figure 15 shows several modes of the Mondrian likelihood function.

The success of any grammar-based inference procedure necessarily hinges on the precise form of the chosen grammar. If a user provides a specification that the grammar simply cannot achieve, the results can be unsatisfactory, as shown in Figure 7. Moreover, it can be difficult to predict a priori if a particular grammar is capable of producing a good match for a provided specification: some experimentation may be required.

On the other hand, the presented technique affords users increased flexibility in writing and developing grammar-based procedural models. Properties that are difficult to impose via generative rules are often easier to specify as terms in a likelihood function. As a result, the presented technique alleviates the need to painstakingly construct grammars for which every derivation possesses all of the desired characteristics. Contrast, for instance, the random derivations from the building and Mondrian grammars shown in Figure 5, and their targeted counterparts in Figures 11 and 15. In these examples, desired characteristics were obtained partly through the grammar specification and partly by optimization.

Fig. 7. The grammar in these examples supports only bipodial branching. Thus, it produces a good fit for one sketch (left), but not the other (right).

13. PERFORMANCE

Performance characteristics for the examples in this article are summarized in Table I. For most procedural modeling tasks, producing a good visual fit is more important than guaranteeing a globally optimal solution. Therefore, rather than employing an automated convergence test such as coupling from the past [Propp and Wilson 1996], we observe each simulation as it evolves and terminate it manually. Figure 12 plots the number of iterations against model complexity for the seven image-based examples. Figure 13 shows the evolution of the MAP estimate over time for each of these simulations. Overall, the jump formulation described in Section 7.2 results in a global acceptance rate of about 30%.

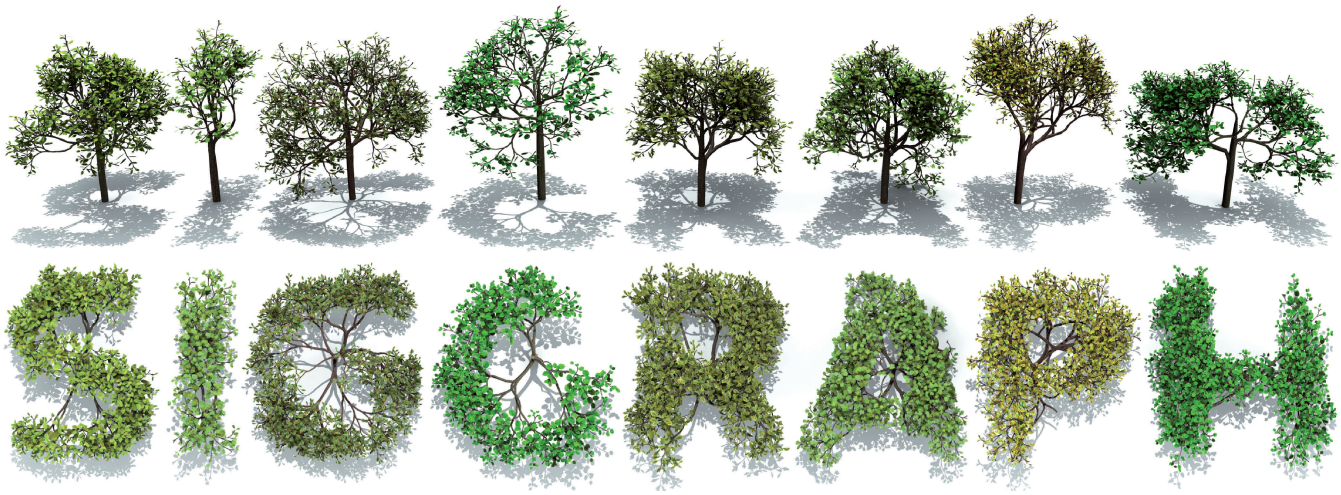


Fig. 8. A set of young oak trees from a single grammar, produced with the algorithm described in this article. The trees exhibit natural, realistic structure (top) and spell "SIGGRAPH" when viewed from above (bottom). Zoom in for detail.

Fig. 9. Acacia trees (left) and willows (right) targeted to the SIGGRAPH logo. Zoom in to see the small branches supporting the thin edges of the crescent shapes.

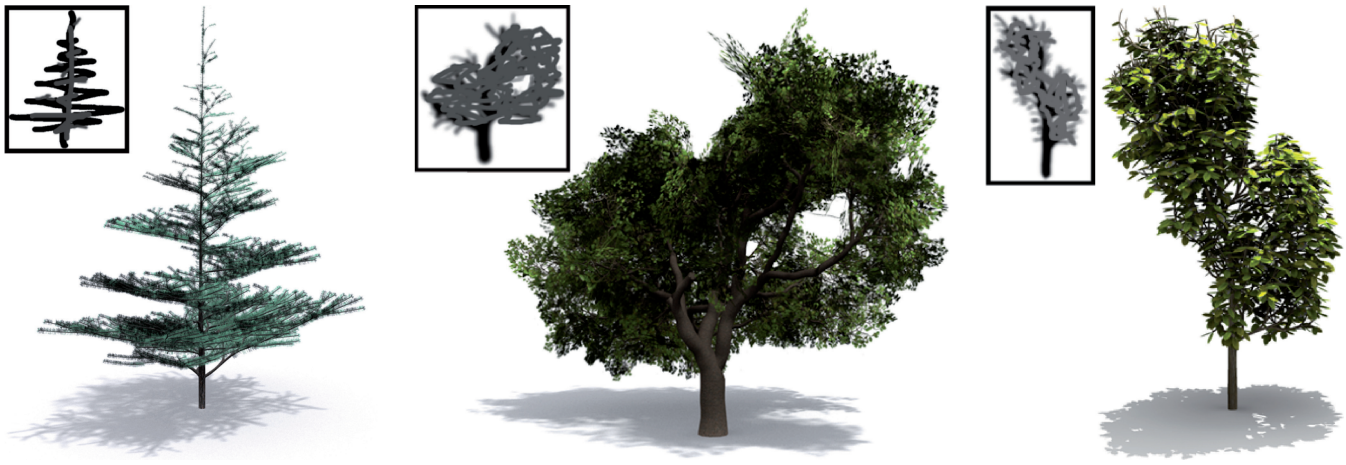
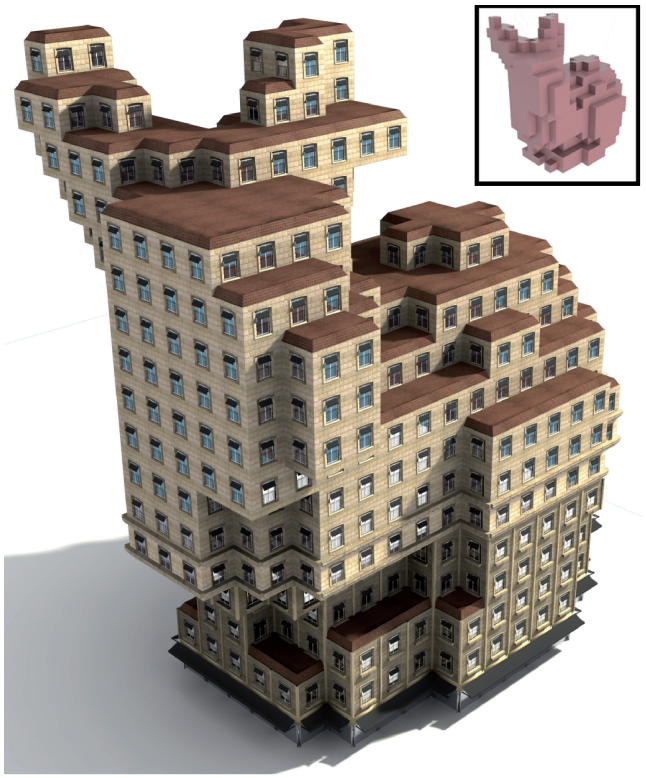


Fig. 10. Conifer, old oak, and poplar grammars targeted to sketches. The inference technique successfully models details in the sketches, such as silhouettes and the low-density hole in the oak.



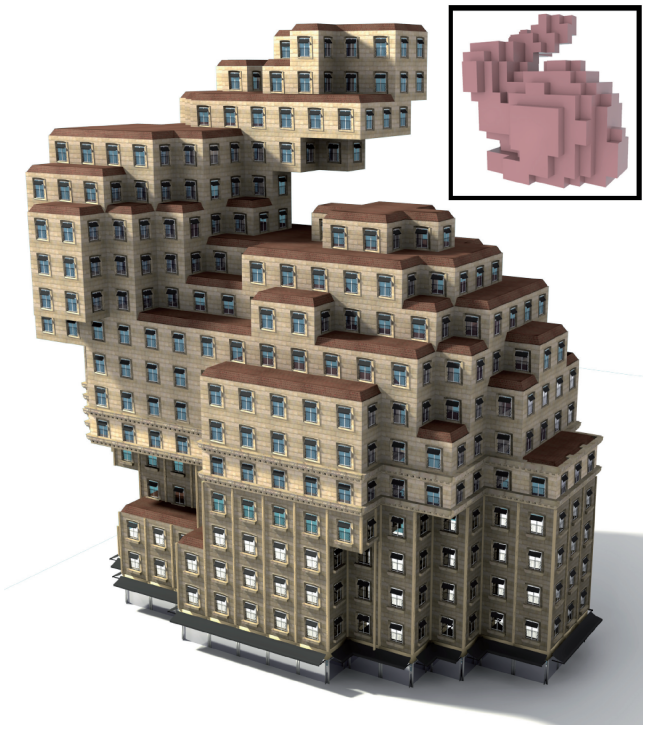


Fig. 11. Two views of the building grammar targeted to a voxelization of the Stanford bunny (inset).

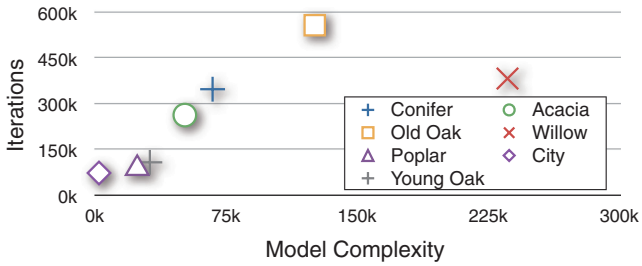


Fig. 12. Number of iterations plotted against model complexity (symbols plus parameters) for the seven image-based examples. The lone outlier (the willow) was terminated after ten hours due to its high per-iteration cost.

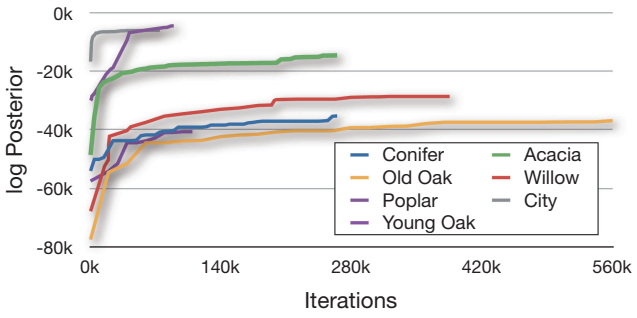


Fig. 13. Evolution of the MAP estimate over time for each of the seven image-based examples.

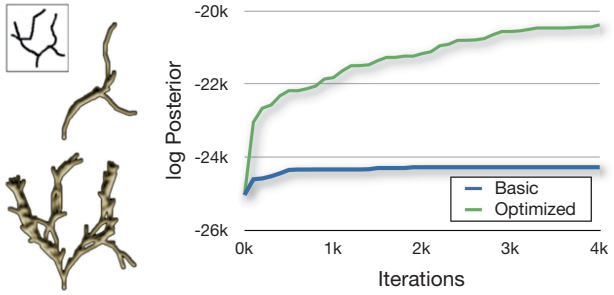


Fig. 14. (left) MAP estimates produced for a simple branching grammar with (bottom) and without (top) the optimizations described in Section 8. (right) The evolution of the MAP estimates for these simulations.

Table I. Performance Figures for the Examples

Example	Symbols	Params	Depth	Vertices	N	Time
City (6)	850	1,154	120	1,432	75k	14m
Young Oak (8)	12,152	18,937	15	176,520	110k	2h
Acacia (9)	21,176	29,909	29	50,862	265k	4h
Willow (9)	70,754	164,823	34	166,328	385k	10h
Conifer (10)	31,319	35,639	27	221,941	350k	9h
Old Oak (10)	53,624	71,738	30	66,191	560k	6h
Poplar (10)	11,019	12,836	32	193,016	90k	2h
Building (11)	9,673	16,805	28	1,660	850k	30m
Mondrian (15)	74	134	21	38	6k	5s

Shown: the number of symbols and descriptive parameters in each derived model, the depth of the derivation tree, the number of iterations N used to produce the result, and the total simulation time on a 2GHz Intel Core 2.

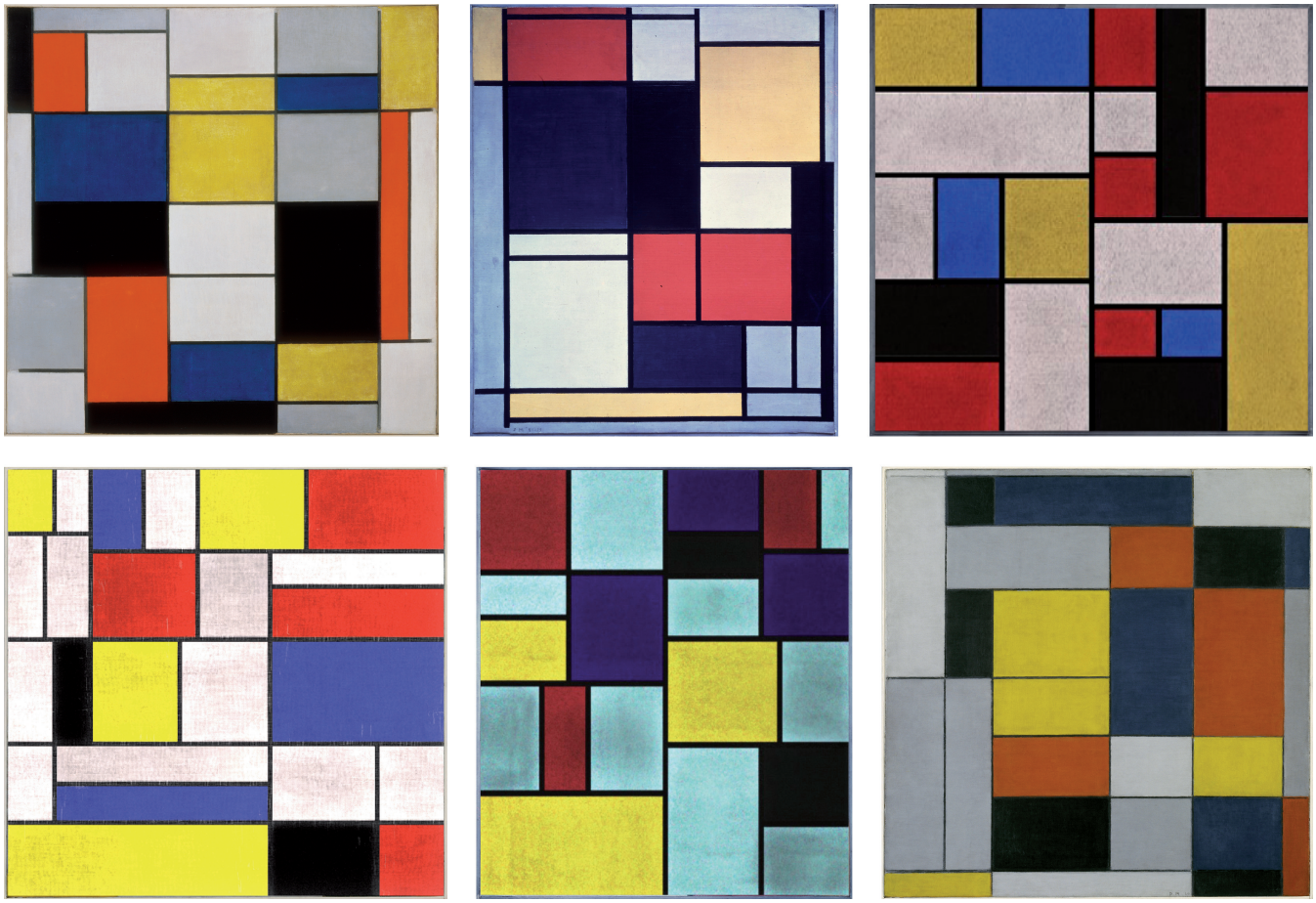


Fig. 15. Three images of original Mondrian paintings, and three random samples from our targeting function. Clockwise from top left, the first, second, and fourth images are Mondrians. The paintings are "Composition A." 1920; "Tableau No. II with Red, Blue, Black, Yellow and Grey" 1921-25; and "No. VI / Composition No. II" 1920, all © 2011 Mondrian/Holtzman Trust c/o HCR—International Virginia.

These results compare favorably with related work on MCMC inference on procedural models. Schlecht et al. [2007] report using 20,000 iterations to fit a simple two-dimensional branching structure (with fewer than a hundred symbols and parameters) to fungal microscopy data via a hand-tuned RJMCMC formulation. In contrast, our models, which are roughly 100× more complex, require only about 10× more computation. Moreover, the optimizations described in Section 8 are quite effective: Figure 14 illustrates the speed and quality enhancements afforded by optimized nonterminal selection, parallel tempering, delayed rejection, and annealing.

14. CONCLUSION

We presented an algorithm for controlling grammar-based procedural models. The algorithm formulates procedural modeling tasks as probabilistic inference problems and solves these problems efficiently with Markov chain Monte Carlo techniques. We believe that this work takes a step towards making grammar-based procedural modeling more useful and accessible.

One clear opportunity for future work is improving the convergence of the optimization. In the current framework, the rate of improvement decreases as a dominant mode of the posterior is found and explored (see the long elbows of the curves shown in Figure 13). It seems likely that the use of data-driven proposal distributions in this context could result in as much as an order of magnitude improvement in performance.

Another promising research avenue is extending the methods described in this article to more powerful procedural representations such as open or context-sensitive L-systems [Prusinkiewicz and Lindenmayer 1990]. Similarly, our requirement that descriptive parameters live only on terminal symbols is currently necessary to provide a clean separation between dimension-altering and dimension-preserving moves, but also limits the types of models that can be described.

Lastly, this work highlights the need for more effective tools for creating grammar-based procedural models. The nine grammars employed in this article contain between 50 and 400 lines of code each, and took several hundred man-hours to develop and refine. We are eager to see new techniques aimed at removing this final stumbling block from the procedural modeling pipeline. Early steps in this important direction have recently been reported [Bokeloh et al. 2010; Stava et al. 2010].

ACKNOWLEDGMENTS

We are grateful to P. Merrell for creating the geometric components and materials for the city and building grammars, C. Platz for creating the materials and setting up the rendering for the tree examples, and V. Jojic for helpful discussions. We would also like to thank two anonymous reviewers for their helpful comments and suggestions.

REFERENCES

AGARWAL, M. AND CAGAN, J. 1998. A blend of different tastes: the language of coffee makers. *Environ. Plan. B: Plan. Des. 25,* 2, 205–226.

ALEGRE, F. AND DELLAERT, F. 2004. A probabilistic approach to the semantic interpretation of building façades. In *Proceedings of the International Workshop on Vision Techniques Applied to the Rehabilitation of City Centres*.

ANDRIEU, C., DE FREITAS, N., DOUCET, A., AND JORDAN, M. I. 2003. An introduction to MCMC for machine learning. *Mach. Learn. 50,* 1, 5–43.

ANDRZEJEWSKI, D., STORK, D. G., ZHU, X., AND SPRONK, R. 2010. Inferring compositional style in the neo-plastic paintings of Piet Mondrian by machine learning. *Comput. Vis. Image Anal. Art 7531,* 1.

BOKELOH, M., WAND, M., AND SEIDEL, H.-P. 2010. A connection between partial symmetry and inverse procedural modeling. In *Proceedings of the SIGGRAPH Conference*. ACM.

BOX, G. E. P. AND MULLER, M. E. 1958. A note on the generation of random normal deviates. *Ann. Math. Statis. 29*, 610–611.

BUELINCKX, H. 1993. Wren's language of City church designs: a formal generative classification. *Environ. Plan. B: Plan. Des. 20,* 6, 645–676.

CAGAN, J. AND MITCHELL, W. J. 1993. Optimally directed shape generation by shape annealing. *Environ. Plan. B: Plan. Des. 20,* 1, 5–12.

CHEN, X., NEUBERT, B., XU, Y.-Q., DEUSSEN, O., AND KANG, S. B. 2008. Sketch-Based tree modeling using Markov random field. In *Proceedings of the SIGGRAPH Asia Conference*. ACM.

CHENNEY, S. AND FORSYTH, D. A. 2000. Sampling plausible solutions to multi-body constraint problems. In *Proceedings of the SIGGRAPH Conference*. ACM, 219–228.

DEUSSEN, O., HANRAHAN, P., LINTERMANN, B., MĚCH, R., PHARR, M., AND PRUSINKIEWICZ, P. 1998. Realistic modeling and rendering of plant ecosystems. In *Proceedings of the SIGGRAPH Conference*. ACM, 275–286.

DUARTE, J. P., ROCHA, J. M., AND SOARES, G. D. 2007. Unveiling the structure of the Marrakech Medina: A shape grammar and an interpreter for generating urban form. *Artif. Intell. Engin. Des. Anal. Manufact. 21,* 4, 317–349.

EBERT, D. S., MUSGRAVE, F. K., PEACHEY, D., PERLIN, K., AND WORLEY, S. 2002. *Texturing and Modeling: A Procedural Approach*, 3rd ed. Morgan Kaufmann.

GEYER, C. 1991. Markov chain Monte Carlo maximum likelihood. In *Proceedings of the 23rd Symposium on the Interface: Computing Science and Statistics*. 156–163.

GREEN, P. J. 1995. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika 82*, 711–732.

GREEN, P. J. AND MIRA, A. 1999. Delayed rejection in reversible jump Metropolis-Hastings. *Biometrika 88*, 1035–1053.

HASTIE, D. AND GREEN, P. J. 2009. Reversible jump MCMC. Tech. rep., University of Bristol.

HASTINGS, W. K. 1970. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika 57,* 1, 97–109.

KNIGHT, T. W. 1980. The generation of Hepplewhite-style chair-back designs. *Environ. Plan. B: Plan. Des. 7,* 2, 227–238.

KONING, H. AND EIZENBERG, J. 1981. The language of the prairie: Frank Lloyd Wright's prairie houses. *Environ. Plan. B: Plan. Des. 8,* 3, 295–323.

LINDENMAYER, A. 1968. Mathematical models for cellular interactions in development ii. Simple and branching filaments with two-sided inputs. *J. Theor. Biol. 18,* 3, 300–315.

MARINARI, E. AND PARISI, G. 1992. Simulated tempering: A new Monte Carlo scheme. *Europhys. Lett. 19,* 6, 451–458.

METROPOLIS, N., ROSENBLUTH, A. W., ROSENBLUTH, M. N., TELLER, A. H., AND TELLER, E. 1953. Equation of state calculations by fast computing machines. *J. Chem. Phys. 21,* 6, 1087–1092.

MÜLLER, P., WONKA, P., HAEGLER, S., ULMER, A., AND VAN GOOL, L. 2006. Procedural modeling of buildings. In *Proceedings of the SIGGRAPH Conference*. ACM, 614–623.

MĚCH, R. AND PRUSINKIEWICZ, P. 1996. Visual models of plants interacting with their environment. In *Proceedings of the SIGGRAPH Conference*. ACM, 397–410.

NEAL, R. 1994. Sampling from multimodal distributions using tempered transitions. *Statis. Comput. 6*, 353–366.

NEUBERT, B., FRANKEN, T., AND DEUSSEN, O. 2007. Approximate image-based tree-modeling using particle flows. In *Proceedings of the SIGGRAPH Conference*. ACM.

PALUBICKI, W., HOREL, K., LONGAY, S., RUNIONS, A., LANE, B., MĚCH, R., AND PRUSINKIEWICZ, P. 2009. Self-Organizing tree models for image synthesis. In *Proceedings of the SIGGRAPH Conference*. ACM.

PARISH, Y. I. H. AND MÜLLER, P. 2001. Procedural modeling of cities. In *Proceedings of the SIGGRAPH Conference*. ACM, 301–308.

PROPP, J. G. AND WILSON, D. B. 1996. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Struct. Algor. 9,* 1&2.

PRUSINKIEWICZ, P. 1986. Graphical applications of L-systems. In *Proceedings on Graphics Interface*. Canadian Information Processing Society, 247–253.

PRUSINKIEWICZ, P. AND HANAN, J. 1990. Visualization of botanical structures and processes using parametric L-systems. In *Scientific Visualization and Graphics Simulation*, 183–201.

PRUSINKIEWICZ, P., JAMES, M., AND MĚCH, R. 1994. Synthetic topiary. In *Proceedings of the SIGGRAPH Conference*. ACM, 351–358.

PRUSINKIEWICZ, P. AND LINDENMAYER, A. 1990. *The Algorithmic Beauty of Plants*. Springer, New York.

PUGLIESE, M. J. AND CAGAN, J. 2002. Capturing a rebel: Modeling the Harley-Davidson brand through a motorcycle shape grammar. *Res. Engin. Des. 13*, 139–156.

RIPPERDA, N. AND BRENNER, C. 2006. Reconstruction of façade structures using a formal grammar and RjMCMC. In *Proceedings of the DAGM Symposium on Pattern Recognition*. 750–759.

RIPPERDA, N. AND BRENNER, C. 2009. Evaluation of structure recognition using labelled façade images. In *Proceedings of the DAGM Symposium on Pattern Recognition*. 532–541.

SCHLECHT, J., BARNARD, K., SPRIGGS, E., AND PRYOR, B. 2007. Inferring grammar-based structure models from 3D microscopy data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Conference*. 17–22.

STAVA, O., BENEŠ, B., MĚCH, R., ALIAGA, D., AND KRISTOF, P. 2010. Inverse procedural modeling by automatic generation of L-systems. *Comput. Graph. Forum 29,* 2.

STINY, G. AND GIPS, J. 1971. Shape grammars and the generative specification of painting and sculpture. In *Proceedings of IFIP Congress 71*. 1460–1465.

STINY, G. AND MITCHELL, W. J. 1978. The palladian grammar. *Environ. Plan. B: Plan. Des. 5,* 1, 5–18.

SZELISKI, R. AND TERZOPOULOS, D. 1989. From splines to fractals. In *Proceedings of the SIGGRAPH Conference*. ACM, 51–60.

TIERNEY, L. 1994. Markov chains for exploring posterior distributions. *Ann. Statist. 22*, 1701–1762.

TIERNEY, L. AND MIRA, A. 1999. Some adaptive Monte Carlo methods for Bayesian inference. *Statist. Med. 18*, 2507–2515.

VEACH, E. AND GUIBAS, L. J. 1997. Metropolis light transport. In *Proceedings of the SIGGRAPH Conference*. ACM, 65–76.

WHITE, S. R. 1984. Concepts of scale in simulated annealing. *AIP Conf. Proc. 122,* 1, 261–270.

WONG, M. T., ZONGKER, D. E., AND SALESIN, D. H. 1998. Computer-Generated floral ornament. In *Proceedings of the SIGGRAPH Conference*. ACM, 423–434.

Received July 2010; revised November 2010; accepted January 2011