On the Elaboration of Hand-Drawn Sketches

Saul Simhon and Gregory Dudek
Centre for Intelligent Machines
McGill University
3480 University St, Montreal, Canada H3A 2A7

June 15, 2001

Abstract

This work considers an approach for artificially enhancing the richness and level of detail of graphical scenes. In particular, we examine a method for automatically generating high-resolution novel curves from manually sketched drawings of those curves. The essential idea is to augment the hand-drawn curves using prior knowledge to produce a more elaborated picture. Our method uses multi-scale analysis of a class of training data to capture statistical properties of the set. These properties are then conditioned at a coarse scale by the hand-drawn curve to *steers* the synthesis according to the overall shape of the curve. Given an approximation sketch, the algorithm generates the most likely scene by propagating probabilities over a Markov Chain. Users without artistic capabilities can then describe scenes in a more natural way to build impressive graphics.

1 Introduction

In this paper we consider the manually guided synthesis of detailed curves from very rudimentary steering information. Our goal is to provide a quick and easy way for users to draw elaborate drawings. We present a method to automatically synthesize the elements of a drawing that are too difficult or time consuming to manually draw while maintaining the overall shape of the drawing. In most cases, users do not have difficulty outlining the overall shape of drawings at a course scale. There may be some deviation in the proportion and placements of elements but the main difficulties arises at low scales and high variation areas. Developing a method to help reconstruct those intricate details is an integral step towards more sophisticated systems for elaborating complex images from much simpler high-level specifications.

The approach we take consists of learning statistical properties of a class of *a priori* data over multiple scales. We calculate probabilities using a Non-Stationary Markov Model over the curves' arc-lengths. By learning the statistical relationships between consecutive elements and between a curve and a smoother hand-drawn facsimile, we can later exploit these relationships to synthesis the high-resolution details. Further, since a non-deterministic approach is taken, various instances of the same illustration can be generated simply by random sampling over the regions of maximum likelihood.

Preliminary application to this framework uses a class of parametric curves for coastline synthesis but the frame work can be applied to any data class where high-resolution detail is a function of overall shape. Providing the techniques we describe below can assist in many applications such as: 3-D modeling where curves are used to specify object boundaries and deformations; key-frame animation where curves specify trajectories of objects; pen-and-ink illustrations where curves are the main elements of the design. This work can also be extend to applications for surfaces and textures.

2 Background

One of the most widely used methods to draw curves and surfaces consists of specifying a set of vertices and using an interpolating function that defines the geometry between the vertices. Commonly, tensor product NURBS [5] are used since they provide local control of the curve segments with up to second degree parametric continuity. However, in order to represent surfaces of arbitrary topology, the model must be partitioned into a collection of patches and explicitly stitched together [3]. A large number of parameters are introduced to stitch adjacent patches using geometric continuity conditions. Subdivision curves and surfaces offer a better alternative by repeatedly refining an initial control mesh [1]. DeRose *et al.* present a general subdivision model for reconstructing piecewise smooth surface models from scattered control points [10]. Their work describes subdivision rules that model sharp features while maintaining the smooth areas by relaxing continuity conditions across labeled edges and modifying subdivision masks based on label type. Although subdivision surfaces are used often in state of the art computer graphics applications, they still require highly-skilled users, especially in scenes that are over-refined where editing becomes very cumbersome and details are very intricate.

Another difficulty in traditional curve and surface fitting techniques is that there is no attempt to preserve higher resolution detail when editing at a broader region. Work by Forsey and Bartels [6] addressed this problem by developing hierarchical B-splines. Large- or small-scale changes to the surface can be made by manipulating control points at the corresponding levels in the hierarchy. Some of the drawbacks in their work imposed users to manually design the hierarchical network. Further, the surface points and derivatives were no longer linear functions of the control points, introducing computational complexities and non-unique solutions. Later work [7] automated the process by recursively fitting surfaces at a coarse scale and refining areas with large residuals. In other work, Salesin *et al.* [12, 4] describe a multi-scale curve representation based on wavelets that produced unique solutions for given shapes. Curves may be modified at multiple levels of detail, such as changing the overall form of the curve while preserving its detail or vise versa.

Although these and other similar methods provide multi-scale editing, they lack in representing the connection between overall shape and high-resolution detail. The goal is to disassociate the scales, providing independent control for each level of detail. While this is desirable for certain applications, it does not allow users to fully describe a curve directly from coarse data. Our method, analogous to wavelet transform, captures statistical relationships over varying scales, recognizing that a change in the overall shape should affect the high-resolution features. The main premise is that a good way to interpret an incomplete specification is based on previously seen data.

The approach we take in synthesizing curves is similar to and inspired by stochastic methods of texture synthesis [14, 11, 2]. Work by Efros and Leung [2] describe a method of learning statistical properties of a texture sample and extrapolating these properties to generate the pattern. A texture is modeled as a Markov Random Field where the probability distribution for the value of a pixel is dependent on only its neighboring pixels. A new pixel is synthesized by choosing the value that is most probable based on a neighborhood match with the sample texture. Wei and Levoy [13] combined this method with a pyramid-based filter [9] to maintain large-scale structure. They also apply their method in the temporal domain, producing movies such as ocean waves and fire. A strong assumption in these approaches is that of a local and stationary random process. Such assumptions are not necessarily valid when dealing with the syntheses of arbitrary scenes. Further, there is no way of controlling and steering the growth of textures. In our work, presented in the next section, we use a non-stationary transition probabilities and adopt a controlling component to steer the synthesis process.

3 Curve Synthesis: Markov Steering

We treat the user as a stochastic process, generating a sequence of random measurements over some time interval (a random experiment). Let us suppose that at each point in time we observe a measurement x. This measurement corresponds to a sample point described by the entire function $X(t)$. Let $X_s(t)$ represent a stochastic process for a class of sample curves s over a time interval T. A single sequence of observations for a particular sample set s is called the realization of the stochastic process, corresponding to an instance of the user's curve description. We assume the sequence has an n^{th} order Markov property, i.e. a Markov Process:

$$p\{x_s(t+1)|x_s(t), x_s(t-1) \;..\; x_s(t-n+1)\} = p\{x_s(t+1)|x_s(t), x_s(t-1) \;..\; x_s(0)\}$$

Let α represent a closed curve (the method will still work for curves that are not closed but the Markov chain is not homogeneous). The curve has a parametric representation $(x(t), y(t))$ where t is the arc-length of the curve from $0 <= t <= T$. We denote the absolute angle of the tangent of α at point t by $\theta(t)$. We denote $\phi(t)$ as the steering component of the curve $\theta(t)$ corresponding to a level of abstraction:

$$\phi(t) = \psi(\theta(t)) \tag{1}$$

where ψ is some function that maps the high-resolution curve to one that the operator is more apt to sketch. That is, $\phi(t)$ is considered as the input curve used to maneuver the Markov Process. In principle, this controlling component can consist of any function of the high-resolution curve; there is no geometric requirement as it can even be a mapping to a language symbol space. However, to maintain the Markov property and real time execution, we transform the state space using a time-invariant causal linear filter [8]. We consider a linear transformation such that Equation 1 can be represented in matrix form:

$$\Phi = \Psi\Theta \tag{2}$$

where Θ is a state vector consisting of n sample points (discrete version of $\theta(t)$), Ψ is a $n \times m$ low-pass filter matrix and Φ is a state vector with m sample points (discrete version of $\psi(t)$). By removing the high-frequency components, we provide the user with a control curve that is easier to hand-draw and requires less sampling points to model. Such a transformation loses information and is not invertible.

Multi-scale analysis is performed by repeatedly filtering the training curves using a low-pass filter. This results a multi-scale curves where the controlling component lies near one of the coarse levels. We then define a state space as an $OxSxA$ dimensional space Γ, where O is the order (history) and S is the scale and A is the dimension of the curve attributes that are used (in our case, $A = 1$, the tangent angle θ).

We define the stochastic model as a first order homogeneous Markov Process for the multi-dimensional states $\gamma(t)$ over the state space Γ. Note that although

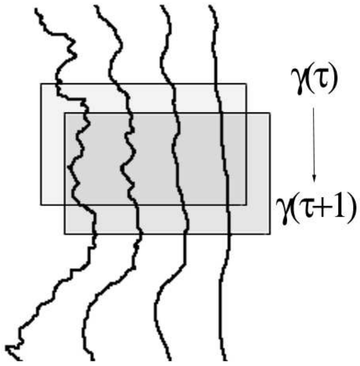

Figure 1: Extracting the next state given the previous ones. Each state is a multi-scale/multi-order representation of the curve.

we use a first order Markov Model, higher order history of a curve is encoded within a single state (Figure 1). Statistical properties of *a priori* data are learned by calculating the transition probabilities $p(\gamma(t+1)|\gamma(t))$.

A Markov Chain with non-stationary transition probabilities is used to reconstruct the curve:

$$P_\gamma(t+1) = M_\gamma(t) P_\gamma(t) \tag{3}$$

Where $P_\gamma(t)$ is the probability state vector at sample t and $M_\gamma(t)$ is the transition matrix at sample t. Similar to texture synthesis techniques, given a seed ($\phi(0)$, $\theta(0)$), we can extrapolate a curve by statistically sampling over $P_\gamma(t)$. This generates a new curve that is based on the statistical properties of the training data. We don't explicitly construct the transition matrix since the data is very sparse and the state-space it too large. Instead we store the probabilities in a linked list, which requires searching at each iteration (space-time tradeoffs).

Transition probabilities for a multi-scale/multi-order state $\gamma(t)$ are calculated by first comparing $\gamma(t)$ with all states $\gamma(t)'$ in the training data using a similarity measure. We then associate the transition probability to the next state $\gamma(t+1)$ by summing up values and normalizing. The similarity measure is calculated as follows:

$$D(\gamma(t), \gamma(t)') = \frac{\sum_\tau \sum_s w(s,\tau) G(\gamma(t,s,\tau) - \gamma(t,s,\tau)')}{\sum_\tau \sum_S w(s,\tau)} \tag{4}$$

where G is a Gaussian function and W is a weight matrix that associates the importance of matches over scale and history. We integrate over the region defined by the state space Γ, (i.e. $\tau : 0 -> O, s : 0 -> S$).

We configure the weight matrix to maintain consistency over both high resolution details early in the history and overall shape. Further, for training sets that exhibit some stationarity, the search space does not have to be restricted by

t but can consist of any location in the training data that has similar statistical response. In fact, one might want to search over a range Δt about t to provide additional tolerance for the hand-drawn curve.

The probability of the next state is calculated by first propagating the probabilities of the previous states using the transition probabilities calculated above and then conditioning the result by the user input ϕ_{in}. The conditioned probability for the i'th state is given by:

$$p_i^{conditioned} = p_i^{predicted} G(\phi_{in} - \theta_{s,i}) \tag{5}$$

where $\theta_{s,i}$ is the current angle extracted from the i^{th} state at a pre-determined scale s. The variance of the Gaussian G sets the influence the input has over the prediction. Large variances depict low user confidence while small variances such as a delta function only pick close/exact matches. A sample point is instantiated using the value of maximum probability, providing an interactive environment where users can see the resulting deblurred curve. For efficiency, a threshold is set to maintaining only the top few candidates.

A summary of the algorithm is outlined as follows:

- Initialize the probability vector by assigning equal probabilities to each $\gamma(0)$ in the training example
- Condition the probability vector using the hand-drawn sketch point $\phi(0)$.
- Threshold the probability vector
- For each sample point from the hand-drawn sketch:
  - Compute the transition probabilities from time t to $t+1$.
  - Given the probability vector at time t, predict the next one using the computed transition probabilities.
  - Condition and threshold the probability vector

3.1 Experimental Results

We have implemented the algorithm and provided a graphical user interface where users can draw curves and see the results. The interface also allows to set parameters, such as scales, orders, weights and variances, and to load and save curves and training data. This is advantageous when users with to add newly synthesized curve to the training set. For our experiments, we used 40 curves of world country borders for training, a 5 scale 20'th order state space and a uniform average filter. Figure 2 shows some samples of the training set. Experiments were done on a 1 GHz P3 processor with real time response.

Figure 3 and 4 show some sample runs with both the hand-drawn curves and the resulting synthesized curves. Most of the curves generated are *novel* ones, never seen in the training set. They maintain the statistical properties of the training set and the overall shape of the hand-drawn curve. The resulting curves *look* like coastlines. There are some issues with curve closures and intersections which are primarily due to accumulated errors and the stochastic nature of the algorithm. A possible extension to this work can consist of imposing additional constraints to maintain properties such as closure.

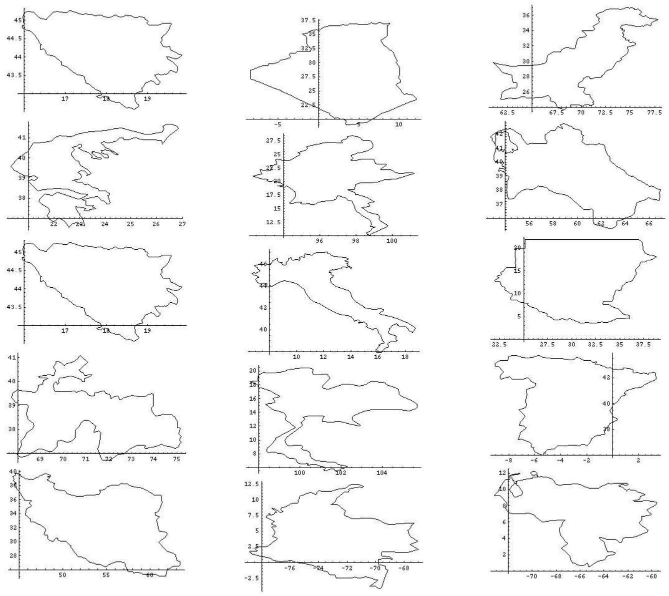

Figure 2: Some examples of curves used in training.

The algorithm produces good results for data classes that sustain structure over local regions. This forms a good framework for applications such as texture mixing but limits the application to data types where global structure prevails. Further, data types with large scale structure usually have features that are well localized in time. As much as the filters blur this time localization, a single offset in the hand-drawn curve might steer the synthesis in a drastically different path. Such mis-synchronization can be seen in figure 6. Figure 5 show some samples of a training set we used that exhibits large-scale properties. The set consisted of 14 leaf outlines. Figure 6 shows the results using this set. One example produced an exact reconstruction for the training set, another example shows a mixture of the training data that is locally consistent but not globally and a third example shows a case where there is no close match found in the training data due to mis-synchronization of global features.

4 Future Work

This paper describes a new approach in generating graphics. The idea can be extended for various applications. Primarily, we will investigate methods

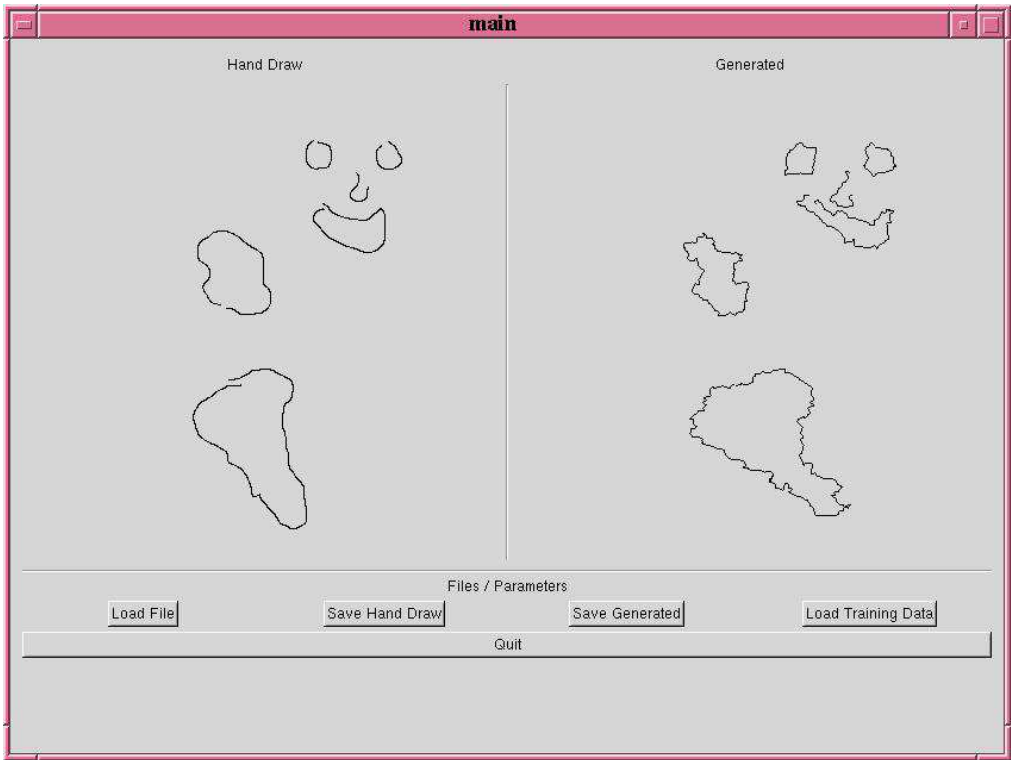

Figure 3: Hand drawn sketches (left) and the synthesized coastline (right).

to extend this work to data that requires consistency over large scales. This may include imposing global function over the curve such as an energy based regularization method. Such methods can also refine the curve by locking on to the right segment or minimizing mixtures. Future direction also includes dealing with multiple curves, automatic classification and segmentation for a better synthesis. We also plan on extending the application to textures and motion signals.

5 Conclusion

This work presents a step towards a method for elaborating hand-drawn curves in real time. The method consists of using *a-priori* data to automatically generate the details for a coarse hand-drawn curve. We model the curves as multi-scale stochastic signals over time. We then propagate uncertainties over a Markov Chain and condition the resulting probability vector by the hand-drawn curve. It was show that the method generates curves well for coastlines which posses only local properties while for data with large scale structure the

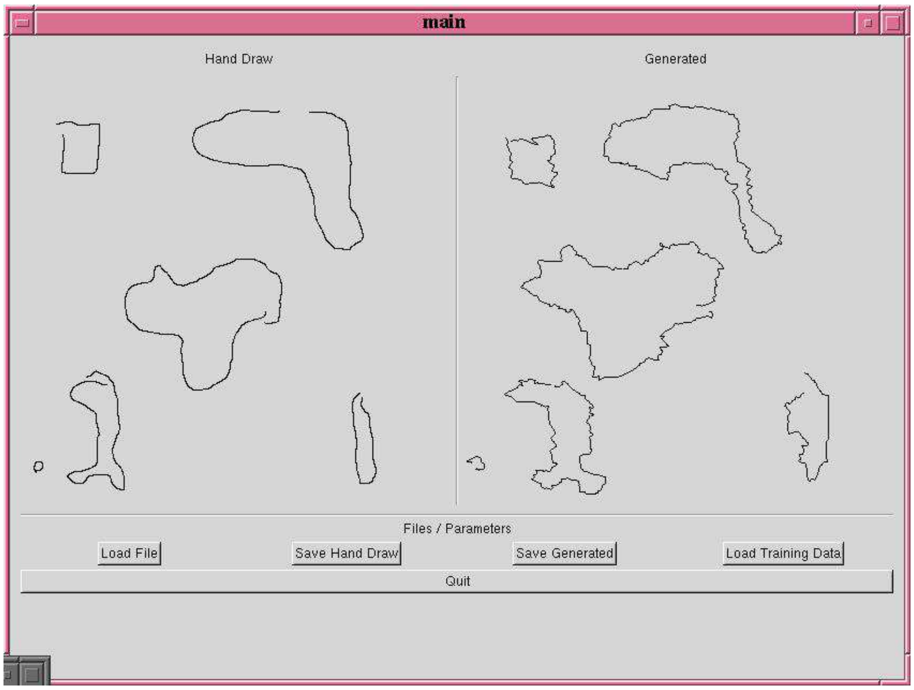

Figure 4: Hand drawn sketches (left) and the synthesized coastline (right).

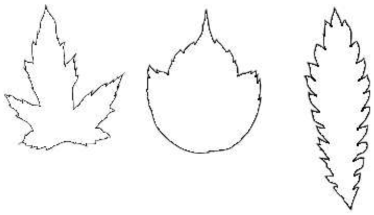

Figure 5: Samples taken from the training set.

method did not maintain those properties. The method allows users to construct libraries and generate new curves that are similar to the class of libraries used.

References

[1] E. Cohen, T. Lyche, and R. Riesenfeld. Discrete b-splines and subdivision techniques in computer-aided geometric design and computer graphics.

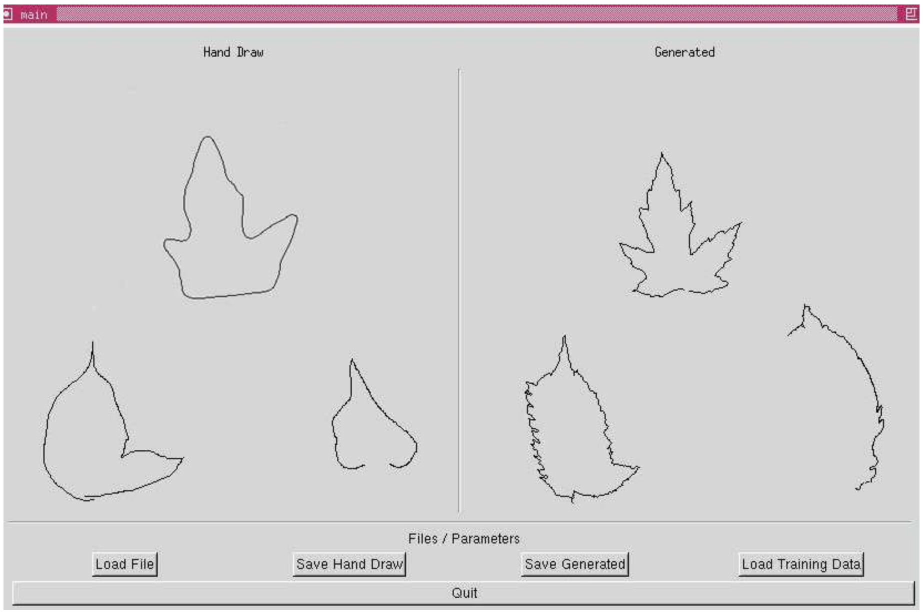

Figure 6: Hand drawn sketches (left) and the synthesized leaves (right). The top center shows an exact reconstruction; bottom left shows a reconstruction consisting of several samples from training that is globally inconsistent; bottom right shows a reconstruction where the control curve reaches the top of the leaf too early and the training data is not rich enough to properly adjust the synthetic curve.

CGIP, 14(2):87–111, October 1980.

[2] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. *International Conference on Computer Vision*, September 1999.

[3] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, 1992.

[4] A. Finkelstein and D. H. Salesin. Multiresolution curves. In *SIGGRAPH '94 Proceedings*, July 1994.

[5] A. R. Forrest. The twisted cubic curve: A computer-aided geometric design approach. *Computer Aided Design*, 12(4):165–172, July 1980.

[6] D. R. Forsey and R. H. Bartels. Hierarchical b-spline refinement. *Computer Graphics*, 22(4):205–212, August 1988.

[7] D. R. Forsey and R. H. Bartels. Tensor products and hierarchical fitting. *Curves and Surfaces in Computer Vision and Graphics II, SPIE proceedings*, 1610:86–96, 1991.

[8] W. Gardner. *Introduction to Random Processes*. McGraw-Hill, 1989.

[9] D. J. Heeger and J. R. Bergen. Pyramid-based texture analysis/synthesis. In *SIGGRAPH '95 Proceedings*, 1995.

[10] H. Hoppe, T. DeRose, T. Duchanm, and M. Halstead. Piecewise smooth surface reconstruction. In *SIGGRAPH '94 Proceedings*, July 1994.

[11] R. Paget and I. Longstaff. Texture synthesis via a non-causal non-parametric multiscale markov random field. In *IEEE Transactions on Image Processing*, volume 7, pages 925–931, June 1997.

[12] E. J. Stollnitz, T. D. DeRose, and D. H. Salesin. *Wavelets for Computer Graphics: a Primer*. Number Technical Report 94-09-11. University of Washington, Computer Science and Engineering, September 1994.

[13] Li-Yi Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. In *SIGGRAPH 2000 Proceedings*, 2000.

[14] S. Zhu, Y. Wu, and D. Mumford. Filters, random fields and maximum entropy, towards a unified theory for texture modeling. *International Journal of Computer Vision*, 27(2):107–126, 1998.