Curve Synthesis from Learned Refinement Models

Saul Simhon†

Gregory Dudek‡

Centre for Intelligent Machines
McGill University
Quebec, Canada

Abstract

We present a method for generating refined 2D illustrations from hand drawn outlines consisting of only curve strokes. The system controllably synthesizes novel illustrations by augmenting the hand drawn curves' shape, thickness, color and surrounding texture. These refinements are learned from a training ensemble. Users can select several examples that depict the desired idealized look and train the system for that type of refinement. Further, given several types of refinements, our system automatically recognizes the curve and applies the appropriate refinement. Recognition is accomplished by evaluating the likelihood the curve belongs to a particular class based on both its shape and context in the illustration.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Line and Curve Generation

1. Introduction

We present a system to beautify curves: i.e. to take curves that roughly depict some property of interest and make them look more like what experts would draw. Our approach consists of learning properties from a database of ideal examples and transform a coarse input curve to make it *look* like those in the database. The key scientific issue is: *in what sense are these curves 'like' one another*? In our work, this likeness is expressed statistically. However, unlike other approaches in statistical modeling [2, 1, 5] which are typically based on single-layered Markovian methods, our method uses *Hidden* Markov Models (HMM). This two-layered approach (akin to and LDS system) allows us to learn not only the statistical properties of the desired look, but also to learn drawing habits and control schemes for that look. Specifically, this modeling formalism allows us to express the relationship between aspects of a system we can observe directly (the curve drawn by the user) and variables that cannot be observed, but which determine the output (in this case, the curve the user "really wants").

In additions to beautifying curves on an individual basis, when we are given outlines consisting of many strokes, we must also be able to automatically detect which beautifications model should be applied on what curve. Our HMM framework allows us to this by evaluating the likelihood that a curve *belongs* to an HMM based on its shape. This recognition step is further refined by also considering high-level semantic constraints on the viable sequences of refinements that can apply, including constraints on their associated position. For example, we should never apply a terrain refinement on a curve that is drawn above another curve that looks like a cloud.

Learning and synthesis in this fashion is accomplished using a *Hierarchical* Hidden Markov Model (HHMM) where the levels in the hierarchy consist of HMMs that can encode both low-level individual curve refinements and high-level constraints on the refinements themselves. We present a two level hierarchy in which the refinement process is applied at: the curve level and the scene level. At the curve level, each HMM learns from examples the desired behavior of a stroke by associating it's shape to a refined look. At the scene level, the HMM encodes semantic constraints on the sequence in which objects are drawn, including constraints on their rel-

† e-mail: saul@cim.mcgill.ca
‡ e-mail: dudek@cim.mcgill.ca

ative position. Finally, synthesis is performed by first propagating information from the user inputs up the hierarchy and then decoding the most likely path down the hierarchy to enforce the learned constraints of all levels.

2. Related Work

Recent advancements in texture synthesis methods [6, 1, 5] suggest that we can learn the regular properties of example images and generate new ones that exhibit the same statistics. Work by Hertzmann *et al.* [2] show how such methods can be applied to synthesizing stylized curves. Curve styles are learned from the statistics of example styles and new curves exhibiting those styles are generated following the shape of an input curve. Analogies between the inputs and outputs are computed by calculating a rigid transformation offset that best matches the input curve and the partial synthesis to a coupled training example (similar to a texture map).

In our work, we provide three key components that extend the approaches cited above. First, using a Hierarchy of Hidden Markov Models allows us to capture multiple stochastic functions and representing scene dynamics over various scales and context. Secondly, individual HMMs are themselves two-layered systems and allow us to model a *controllable* process. That is, the synthesis is driven by the input curve where the actual features generated are *directly* dependent on the shape of the input. This allows us to model examples with localized features that are tied to the shape about a given region (such as a roof ledge that extends only at the corner of the roof). Finally, most approaches to synthesis mainly consider greedy type strategies, choosing the best match at the current point. When we are given a sequence of inputs, the best immediate points may not provide the global optimum. Inputs further down the sequence often bias the likelihood of earlier points, for example, when drawing a vertical line we do not know whether to apply brick features for a wall or bark features for a tree until we see what will be drawn later. Our approach, based on the *Viterbi* algorithm, takes into account the entire sequence of inputs while also avoiding exponential run time complexity.

Hidden Markov Models are analogous to LDS methods which have been readily applied to motion synthesis techniques [4]. In [4], several example motion patterns are used as an ensemble to train an LDS system. Their synthesis procedure controllably generates novel motions as consistent *mixtures* of the original sets based on key-frame points. An alternative approach for mixture models [3] uses a graph to encode the allowable curve transitions and how those transitions should take place. Similar in spirit to these approaches, we generate novel multi-dimensional curve attributes (shape, color, pen-thickness, texture fill) as locally consistent mixtures of examples in an ensemble. Although, our synthesis procedure is controlled by an input of lower dimension (the shape of stroke).

3. Hidden Markov Model

A Hidden Markov Model encodes the dependencies of successive elements of a set of *hidden* states along with their relationship to *observable* states. It is typically used in cases where a set of states, that exhibit the Markov property, are not directly measurable but only their effect is visible through other observable states. Formally, a Hidden Markov Model Λ is defined as follows:

$$\Lambda = \{M, B, \pi\} \tag{1}$$

where M is the transition matrix with transition probabilities of the hidden states, $p\{h_i(t) \mid h_j(t-1)\}$, B is the confusion matrix containing the probability that a hidden state h_j generates an observation o_i, $p\{o_i(t) \mid h_j(t)\}$, and π is the initial distribution of the hidden states. In this work the user curve plays the role of the observations while the hidden states express what we can think of as the Platonic ideal the user might have had in mind at that point along the curve (the refined shape, color etc.).

There is an abundance of literature on Hidden Markov Models and the domain is frequently decomposed into three basic problems of interest:

- **Evaluation**: Given a model Λ and a sequence of observations $o_1, o_2, ..., o_T$, what is the probability that those observations are generated by that model?
- **Decoding**: Given a model Λ and a sequence of observations $o_1, o_2, ..., o_T$, what is the most likely hidden state sequence $h_1, h_2, ..., h_T$ that produces those observations?
- **Learning**: Given an observed set of examples, what model Λ best represents that observed set.

Solutions to the above three problems are key to our work. Learning allows us to model various illustration styles by simply providing the examples. Decoding allows us to synthesize new curves based on a coarse user input. Evaluation allows us to detect the appropriate class of illustration types that an input stroke belongs to, determining the likelihood that the input curve would be generated by the model in question.

4. Two-Level Hierarchical Hidden Markov Model

At the first level of the hierarchy (curve level), we construct a set S consisting of HMMs where each individual HMM is trained using a particular training ensemble:

$$S = \{\Lambda_0^0, \Lambda_1^0, ..., \Lambda_N^0\} \tag{2}$$

The training ensemble consists of a collection of refines curves (the idealized look) associate to a a set of control curve (what the user would draw). In most of our experiments, the control curve is simply a blurred version of the refined curve. Figure 1 shows an example set. The state space of the HMM's corresponds to the values that the sample points of the curve can take on. We sample a curve uniformly over the arc-length and represent it by a sequence of

Figure 1: *Some samples from a training set used for leaf refinement. The left shows the control curves while the right shows the detailed curves.*

tangent angles. We quantize the values to one degree (360 values). Additionally, we extend the state space dimensionality to allow for supplementary attributes such as thickness or color (these can take on any value).

At the second level (scene level), we encode constraints on the models in the first level. Thus, the state space represents the possible models in S. Figure 2 shows an example graph used to represent the allowable sequences of HMMs in the curve level (i.e. users tend to draw curves in particular orders) and include constraints on where they can be applied (as a supplementary attribute). Several such graphs are used to train HMMs at the scene level of the hierarchy:

$$G = \{\Lambda_0^1, \Lambda_1^1, ..., \Lambda_M^1\} \tag{3}$$

Each model in G depicts different kinds of scenes. For example, you can have face scenes that suggest the sequence $forehead \rightarrow nose \rightarrow mouth \rightarrow chin$ or landscape scenes that suggest $grass \rightarrow (flower, above), cloud \rightarrow (tree, below)$. When we do not wish to have constraints on the order in which curves are drawn, a graph can suggest that every model can be followed by any other model, with only their relative positioning as a constraint.

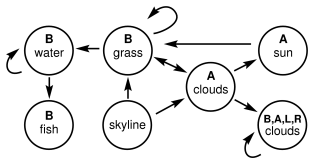

Figure 2: *Example relationships in a scene. The labels correspond to HMMs in the curve level and the letters above correspond to the allowable relative position (i.e. A: above, B: below, L:left R:right).*

5. Synthesis Overview

Given the two-level hierarchy $H = (S, G)$, we first propagate information from the input curve strokes (at the curve level) up the hierarchy and then propagate the model constraints at the scene level down the hierarchy. Let us assume that we are given the desired scene model Λ_d^1 in G, S contains N refinement models and there are M user drawn curve strokes $O_0^0, O_1^0, ..., O_M^0$. For each HMM in S, using the *forward-backward* algorithm, we can *evaluate* its likelihood of generating the sample point sequence for the current curve O_k^0:

$$o_{ki}^1 = p\{O_k^0 \mid \Lambda_i^0\} \tag{4}$$

This identifies how well the input stroke's shape can be represented by the model in question. Calculating this for all N models gives rise to a classification vector O_k^1 where the elements of the vector are the N probabilities $o_{k0}^1, o_{k1}^1 ... o_{kN}^1$ corresponding to the likelihoods of each refinement model in S. Indeed we can stop here and choose the most likely refinement model to use (in a greedy fashion). But we must also consider the higher level constraints in the model Λ_d^1, how the current curve relates to the likelihood of models on the previously drawn curves. Using the current observation vector along with the previous ones (generated in the same fashion) $O_0^1, O_1^1, ..., O_k^1$, we apply the *Viterbi* algorithm to *decode* Λ_d^1 and generate the best sequence of models in S. (This sequence is consistent with both the shape of the curves and the high-level constraints.) We then further decode each individual model in this maximum likelihood sequence of models $\Lambda_0^0, \Lambda_1^0, ..., \Lambda_k^0$, using the associated drawn curve stroke in $O_0^0, O_1^0, ..., O_k^0$. This is performed up to the $M'th$ input stroke to synthesize the refined set of curves.

Because the multi-dimensional state space is extremely large, in practice we can not explicitly store the probability matrices or the complete distribution for all states. Instead, we only maintain a list of the top candidate states. Further rather than having the random access of a probability matrix, at each iteration we must search the training set for matches and compute the likelihoods. Additionally, to avoid quantization issues in matching and provide some control and flexibility over the *mixing tendency*, we do not constrain our system to exact matches. Instead, we update the probabilities based on the goodness of the matches using a Gaussian blur.

6. Results

All of our experiments were executed on a Linux PC with a 1GHz Pentium IV processor and 1GB of RAM. The results were generated in real time.

Figure 3: *Samples from a training set for simple shapes. Curves on the left show the control curves while curves on the right show the associated refined ones that include color.*

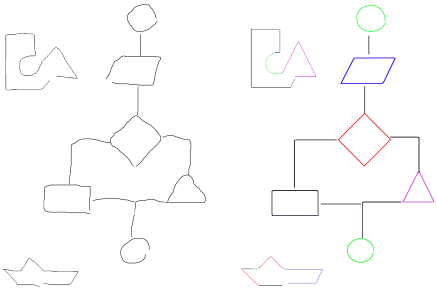

Figure 4: *Example of synthesis using training set in Figure 3. The left shows the inputs and right shows the result.*

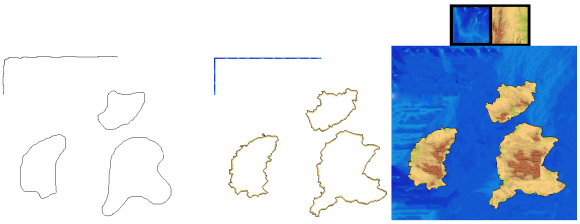

Figure 5: *Generating coastlines with texture fill seeds. Training examples consisted of 25 coastlines. Left shows input, middle shows novel coastlines generated with texture seeds, right shows results using a Markov texture filler (akin to [5]) with the texture image shown above.*

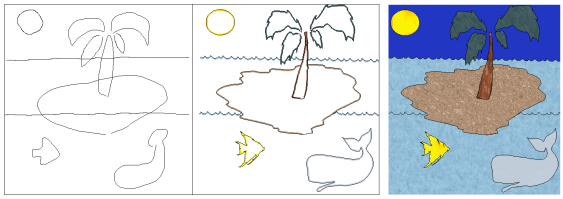

Figure 6: *Synthesis of a beach scene.*

7. Conclusion

We have described an approach for the synthesis of stylized drawings from coarse outlines. This process is based on the representation of coarse to fine refinements as a Hierarchical Hidden Markov Model. The desired refinements are learned by example sets and the semantic constraints on those refinements are learned by a semantic graph. Novel full colored illustrations are generated from noisy curves based on this hierarchy of constraints, including scene level, curve level and, as a post processing step, pixel level constraints.

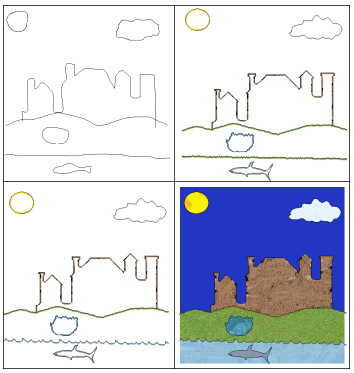

Figure 7: *Top left shows the input sketch, top right shows the output using a greedy method in the scene-level HMM, bottom left shows the output using Viterbi and bottom right shows the result using the Markovian texture filler.*

Figure 8: *Training data used for the skyline model in Figure 7. The two left shapes show the control curves and the two right shapes show the refined ones. It can be seen that from a simple set such as this, a novel skyline can be automatically classified and generated.*

References

1. Alexei A. Efros and Thomas K. Leung. Texture synthesis by non-parametric sampling. *International Conference on Computer Vision*, pages 1033–1038, September 1999.
2. Aaron Hertzmann, Nuria Oliver, Brian Curless, and Steven M. Seitz. Curve analogies. In *13th Eurographics Workshop on Rendering*, June 2002.
3. L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. In *Proceedings of ACM SIGGRAPH*, 2002.
4. Yan Li, Tianshu Wang, and Heung-Yeung Shum. Motion texturing: A two-level statistical model for character motion synthesis. In *Proceedings of ACM SIGGRAPH*, 2002.
5. Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of ACM SIGGRAPH*, pages 479–488, 2000.
6. 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.