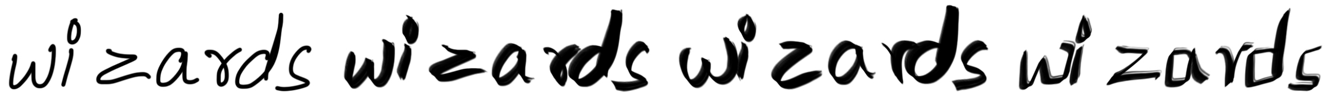
HelpingHand: Example-based Stroke Stylization

Jingwan Lu, Princeton University
Fisher Yu, Princeton University
Adam Finkelstein, Princeton University
Stephen DiVerdi, Adobe Systems Inc.



(a) Query (b) Pose synthesis (c) Trajectory & pose synthesis (d) Alternate style

Figure 1: *(a) Given one or more query strokes, we synthesize (b) hand pose data crucial for bush simulation, based on artist's examples. (c) We can also modify the trajectory to locally match the artist's style. (d) Trajectory and pose synthesis using different artist's style.*

Abstract

Digital painters commonly use a tablet and stylus to drive software like Adobe Photoshop. A high quality stylus with 6 degrees of freedom (DOFs: 2D position, pressure, 2D tilt, and 1D rotation) coupled to a virtual brush simulation engine allows skilled users to produce expressive strokes in their own style. However, such devices are difficult for novices to control, and many people draw with less expensive (lower DOF) input devices. This paper presents a data-driven approach for synthesizing the 6D hand gesture data for users of low-quality input devices. Offline, we collect a library of strokes with 6D data created by trained artists. Online, given a query stroke as a series of 2D positions, we synthesize the 4D hand pose data at each sample based on samples from the library that locally match the query. This framework optionally can also modify the stroke trajectory to match characteristic shapes in the style of the library. Our algorithm outputs a 6D trajectory that can be fed into any virtual brush stroke engine to make expressive strokes for novices or users of limited hardware.

CR Categories: I.3.4 [Computer Graphics]: Graphics Utilities

Keywords: stylus, stroke, trajectory, stylization, data-driven

Links: DL PDF WEB

1 Introduction

Artists wielding traditional drawing instruments such as a pencil, a stick of charcoal or a paint brush exercise fine control over the interplay between their gestures and the physical medium. The appearance of the strokes is governed not only by the path of the instrument but also the pressure and angle (see inset). Digital artists often work with high quality tablets that track a stylus with 6 DOFs (2D position, pressure, 2D tilt, and rotation), which painting software can use to simulate the interaction of digital brushes in various media with increasing fidelity [Baxter and Lin 2004].

However, most people drawing on digital devices today do not use 6-DOF tablets, for two reasons. First, high-quality tablets are relatively expensive as compared with other input devices like mice (2-DOF), multitouch screens (≥2-DOF), or mass-market tablets (3- or 5-DOF). Second, even if everyone owned a 6-DOF tablet, most people would not have the training and experience to be able to wield it effectively. To achieve high quality calligraphy, for example, takes many hours (perhaps years) of practice.

This paper describes a method whereby a non-expert draws a 2D query stroke (Figure 1a), perhaps using an inexpensive input device, and missing DOFs are synthesized based on a library of examples supplied by an artist (Figure 2a). The resulting marks follow the trajectory drawn by the non-expert but convey the gestural style of the artist (Figure 1b). We also show how the approach can be extended to *approximate* the input trajectory while adopting some of the character of the strokes in the artist's library (Figure 1c). This option is beneficial for users who are not confident in their own style, or when using a coarse input device like a mouse. One can easily change style by choosing a different library (Figure 1d).

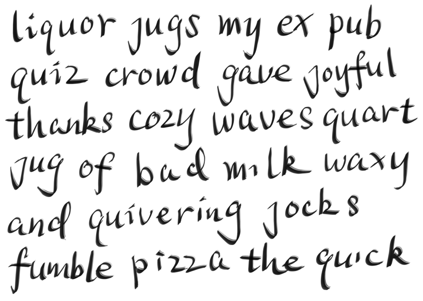
To synthesize hand pose, we address this problem: given a library of 6-DOF strokes (drawn by the artist) and a 2-DOF query stroke (drawn by the user), produce 4-DOFs to accompany the query so that the 6-DOF (2+4) combination *looks like* the strokes in the library. Trajectory stylization optionally replaces the 2-DOF trajectory as well. There is no single correct answer because the same path can be drawn differently, even by a single artist, so our goal is to synthesize *plausible* gestures.

Our method produces gestures that are plausibly in the style of the artist's library. It performs at interactive rates to provide immediate feedback to the user during drawing and to make it easy to choose amongst different styles. Finally, while we have designed the system for interactive use by non-experts or people without high-quality tablets, the same method can apply stylization to any source of line art, including the output of vector illustration software or computer-generated line drawings based on 3D models.

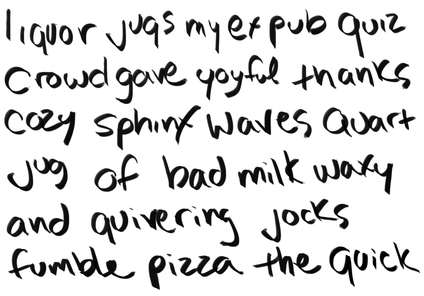
The main contributions of this paper include stroke stylization via example-based synthesis of hand pose—data needed to control a virtual brush—as well as optional trajectory modification. Through data collection and analysis we offer understanding of how hand pose relates to individual writing style. Finally, we explore the space of candidate algorithms and evaluate them quantitatively and qualitatively, concluding with a demonstration of practical results.

2 Related Work

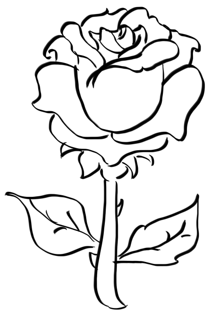
The fundamental idea of our method builds on recent advances in **non-parametric modeling** that have been shown to be effective for texture synthesis by example in a variety of domains—images, meshes, video, and animation [Wei et al. 2009]. These techniques work by selectively copying portions of a library to generate new


liquor jugs my ex pub quiz crowd gave joyful thanks cozy waves quart jug of bad milk waxy and quivering jocks fumble pizza the quick

(a) ArtA


liquor jugs my ex pub quiz crowd gave yoyful thanks cozy sphinx waves quart jug of bad milk waxy and quivering jocks fumble pizza the quick

(b) ArtB



(c) ArtC

(d) ArtD

Figure 2: *Example libraries. (a) and (b) show the same pangrams written by two different artists; (c) and (d) are line drawings.*

data that maintains a similar statistical distribution and aesthetic property, with careful tailoring to the particular requirements of each domain. Ashikhmin [2001] observed that copying swaths of pixels for image texture greatly improves performance over per-pixel methods, but leads to discontinuity artifacts at boundaries. Optimal boundary placement on the 2-D grid is intractable; however, we show that on our 1-D domain it can be solved via dynamic programming. Due to this optimization over the 1-D domain, our method is similar to the video textures of Schödl et al. [2000].

We also derive high-level inspiration from the approach for human motion synthesis of Kovar et al. [2002] and Hsu et al. [2004], both of which synthesize high dimensional output from a database of low dimensional control input. Several aspects of stroke gesture synthesis make our problem unique. For example, artist data collection must be fully automatic, as many strokes are made quickly. Similarly, user input must be processed at interactive rates and without additional user input, to allow for a fluid painting experience. Furthermore, we separate the concerns for pose stylization versus input trajectory stylization, and show how to apply our algorithm to either one individually or both simultaneously.

Stylization. There have been a number of versions of curve and stroke stylization in the graphics community. In the 2D domain, Elasticurves [Thiel et al. 2011] intelligently neaten input strokes based on velocity data, but are only able to produce a single style of smooth result. McCrae and Singh [2011] presented a different neatening technique that would piecewise fit and blend among french curve segments to sketched input, to create pleasing line shapes. They also demonstrated alternative stylizations by substituting different canonical strokes for french curves. The curve analogies framework of Hertzmann et al. [2002] could apply arbitrary stylizations to input curves, by learning variations from pre-existing input-output curve pairs. But its extension to more general brush parameters such as pose is not obvious. Moreover, our approach for trajectory modification relies on a library of output-only exemplars, rather than input-output pairs.

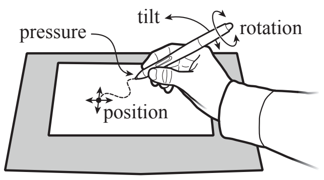
Some previous work address additional brush parameters. House and Singh [2007] propose a dynamic control process to generate varying width along smooth curves. Saito et al. [2007] describe heuristics to procedurally compute stroke width based on curvature. Both approaches depend on explicitly and narrowly defined relationships between stroke shape and width, and do not support additional pose DOFs as needed for a virtual brush model. The isophote distance of Goodwin et al. [2007] formulates heuristics about line thickness, but relies on a 3D scene as input.

Specifically targeted towards digital painting, Okabe et al. [2005] apply realistic brush strokes to curves, by first acquiring videos of the changing shape of real brushes during strokes, and then training an HMM to generate the appropriate footprint based on the user's curves. The IntuPaint system of Vandoren et al. [2008] creates artistic paintings using instrumented, physical brushes and workspaces and interactively acquiring the pigment deposition for compositing to a virtual document in realtime. It supports a variety of traditional media tools and techniques, but imposes heavy custom hardware requirements and thus is not suitable towards our goal of more generally applicable brush stroke rendering.

Handwriting. Our approach of learning and then reproducing the brush pose styles of real artists also bears similarities to a branch of biometrics research concerned with handwriting verification and synthesis. Much of the work records samples of individual character glyphs in advance, and then plays back those glyphs based on the desired target text, often with random or physically inspired perturbations [Varga et al. 2005]. However, these techniques only reproduce strokes they have recorded and labeled, making them inappropriate for freeform sketching.

More general approaches attempt to create dynamical systems that can reproduce 2D handwriting-like motion from training samples [Hinton and Nair 2005]. In the extreme, Plamondon [1995] developed a kinematic theory of rapid human movements that was used to synthesize stroke trajectories with properties similar to natural motions. Beyond 2D trajectories, Franke et al. [2005] created a handwriting robot that holds a pen at a static orientation but can vary pressure throughout a stroke, to create very convincing signature forgeries. Yu et al. [2004] consider hardware that acquire 5D input including pressure and 2D tilt, but only for the purposes of writer identification, and not synthesis. We are not aware of previous work that considers the six DOFs of a real brush.

3 Data Collection and Analysis



We collected two data sets from the artists: English pangrams and line drawings (Figure 2). Five artists drew strokes in their own natural drawing style using a Wacom Intuos4 tablet with an art pen (inset) and Adobe Photoshop CS5 for immediate visual feedback and recording, sampled at 140 Hz. During recording, the tablet orientation remained consistent for each particular artist and is treated as part of their drawing style. Each 6D sample contains 2D position $\mathbf{x} = \langle x, y \rangle$ and 4D pose $\mathbf{h} = \langle \rho, \theta, \phi, \omega \rangle$ comprised of pressure ρ, tilt $\langle \theta, \phi \rangle$ and rotation ω. We refer to *trajectory* $\mathbf{T}$ as a sequence of sample positions $\mathbf{T} = \{\mathbf{x}_i\}$, and *pose* $\mathbf{H}$ as the corresponding hand pose sequence $\mathbf{H} = \{\mathbf{h}_i\}$. After capture, we resample each stroke uniformly in arc length (roughly 1 pixel spacing in 2000x2000 images).

Our system takes those recordings as input and builds a collection of 6D strokes that we call library $\mathbb{L}$. Each library stroke $\mathbf{L} \in \mathbb{L}$

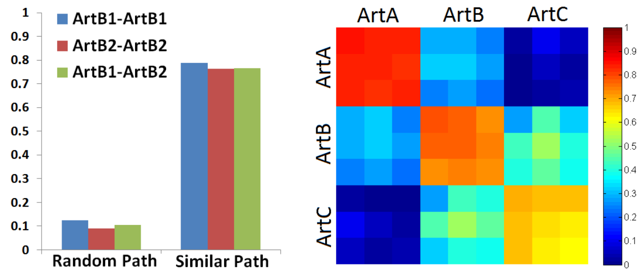


Figure 3: *Pose correlation.* Left: *Correlation is weak for random pairs and strong for similar shapes.* Right: *Libraries drawn by the same artist are highly correlated (diagonal blocks), relative to libraries from different artists.*

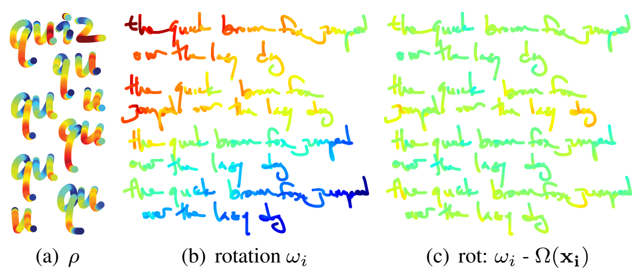


Figure 4: *Pose visualizations (color range: blue=low – red=high). (a) Pen pressure follows similar progressions for similar shapes drawn at different times. (b) Rotation angle depends strongly on pen position. (c) Positional influence removed via linear regression.*

consists of a set of position and pose samples, $\mathbf{L} = \{\mathbf{x}_i, \mathbf{h}_i\}$. Figure 2 shows Photoshop renderings from some examples of the data that we collected. While artists created multiple pages of input (e.g., Figure 1a), we treat each page as a separate library. Typical pangram libraries have between 100 and 200 strokes, each of which typically has between 40 and 80 samples, whereas drawings often contain some longer strokes.

3.1 Artist Distinctiveness

This project relies on several assumptions. The first is that shapes of strokes used in handwriting (for example) are part of the "style" of a given artist; different examples of similar shapes like loops or 'v' shapes will be more consistent from the same person as compared with examples from other artists. This assumption is widely accepted. For example, it forms the basis for forensic handwriting analysis. We rely on this assumption for the trajectory synthesis method presented in Section 4. But we also make a second assumption that is less well studied—that artists' hand poses are determined by their target trajectories, with some variance. An artist drawing the same shapes (with the same intention) multiple times tends to select poses from the same pool of possible gestures. This implies that given a stroke $\mathbf{L}$ with trajectory $\mathbf{T}$, if we find other strokes with similar trajectories their poses should have similar statistics as $\mathbf{T}$. This observation forms the underpinnings of the example-based pose synthesis methods presented in Section 4.

Therefore, to verify our pose assumption, we conduct a correlation-based analysis on the recorded library data, investigating how well the pose attributes are correlated between stroke patches with similar shapes. For each sample i in each stroke $\mathbf{L} \in \mathbb{L}$, we construct a patch $\mathbf{t}_i$ centered at sample i and composed of neighboring samples $\mathbf{t}_i = \{\mathbf{x}_j \mid n \geq |i - j|\}$. For a stroke with m samples, we have a total of $m - 2n$ overlapping patches, each of size $2n + 1$. The size n provides a notion of what we mean by "local" and through moderate trial and error we found $n = 12$ to work well; we use this value throughout our experiments.

For each patch $\mathbf{t}_i \in \mathbf{L}$ we construct a simple feature vector $\mathbf{F}_i$ as the vector of local tangents at every sample, concatenated with a pair of coordinates $\langle x_i^-, x_i^+ \rangle$ that indicates whether the sample is near the beginning or end of the stroke (described fully in Section 4.1). Next for each patch $\mathbf{t}_i$ we find the most similar patch $\mathbf{t}_c$ from a different stroke within the library, measured by the L_2 distance $\|\mathbf{F}_i - \mathbf{F}_c\|$. We reject all pairs of patches whose distance is larger than a threshold (the mean feature distance over all pairs of patches). For each remaining pair of patches $\langle \mathbf{t}_i, \mathbf{t}_c \rangle$ we calculate the Pearson correlation coefficient between their corresponding pose attributes, $r_{i,c} = \text{corr}(\mathbf{H}_i, \mathbf{H}_c)$. To aggregate across multiple patch pairs and multiple pose attributes, we convert Pearson's r coefficients into Fisher's z coefficients, compute the average, and then convert the result back into Pearson's r [Silver and Dunlap 1987]. The aggregated r gives us an indication of how well a library's pose is correlated among similar strokes. As a baseline comparison, we also find *random* pairs of patches of the same size $2n + 1$ within the library and calculate the correlation among them.

Figure 3 summarizes our findings. The bar chart shows that within libraries written by the same person, pose sequences of random pairs of patches are not well correlated whereas pose from strokes with similar trajectories have high correlation. The blue bars report aggregate correlation within the library shown in Figure 2b. Red bars are for a different library from the same artist, and green shows correlation between those two libraries. While correlation is low for random pairs it is still positive; many strokes have *some* correlation because pressure always starts and ends near zero, for example. As evidence that different people have different writing styles, we observe that their hand pose data also carry different statistics. The matrix on the right shows the same correlation analysis described above, comparing all pairs from a set of nine libraries (three by each of three artists). The block diagonal (same artist) exhibits obviously higher values than the off-diagonal (different artists).

Of course even within the same library, correlation is not perfect ($r < 1$) for at least three reasons. (1) During the recording process, which lasts about an hour, the volunteer may introduce hand pose variation setting down the stylus and picking up again with a different grip. (2) Pairs of similar patches do not have exactly the same trajectories. (3) The same artist exhibits some variation in hand pose, even when following the same path multiple times. The degree of variance depends on the individual and to some extent the artist's skill. Pose data from novice users tends to exhibit less consistency. Three of our five volunteers have more than two years of experience with a stylus (Artists A-C in Figure 3) while the other two are less experienced and have correlation statistics around 0.5—still higher than typical correlation between different artists (off-diagonal). Figure 4a supports these claims with a visualization of the pressure magnitude of example characters drawn by an artist (the word "quiz" above with other examples of letters 'q' and 'u' below). Observe that the same shapes drawn at different moments on different parts of the tablet exhibit similar pressure progressions, although there is variation.

3.2 Spatial Variation

There are other influences related to pose, beyond shape and artist. One of the prominent factors is the position of the stroke on the tablet. This is to be expected based on body kinematics, because the pen pose changes when the person moves his hand from one position to another. This factor can have an observable effect. As an extreme example, Figure 4b shows that the rotation of the pose is strongly related to the position of the stroke, at least in this library where the range from blue to red is roughly $50°$. It is possible to remove the strong spatial component prior to pose synthesis, as follows. We model the pose attributes at sample i, for example rotation ω_i, as a sum of two components: (1) local variation intended by the artist ω'_i, and (2) a positional component $\Omega(\mathbf{x_i})$. We observe that the the local variations are fairly uniformly distributed and may be viewed as noise relative to the more global positional signal. Therefore we approximate $\Omega(\mathbf{x_i})$ by a quadratic polynomial over the coordinates of $\mathbf{x_i}$, and perform a simple least squares fit to find the coefficients of the polynomial. Then we can remove the approximate positional influence simply by subtracting the polynomial from ω_i, but adding in its place the average value. Figure 4c shows the result, with a range of roughly 20°. We find that tilt tends to vary by position the way rotation does, but pressure is more consistent. Positional influence varies from library to library. Nevertheless, we have not found these effects to have a strong impact on the visual quality of the synthesis results. Therefore, our libraries have not been modified using the regression fit for any of the results that follow.

3.3 General Observations

Here we note a few general observations we found when studying the data in the example libraries supplied by our artists:

- **Continuity.** Pose tends to change smoothly along strokes.
- **Directionality.** Strokes drawn from left to right are not identical to strokes drawn right to left, for example.
- **Gravity.** Statistics of stroke paths are not rotationally symmetric; up, down, left and right are not interchangeable.
- **Endpoints.** Special care must be taken for the beginnings and endings of strokes because the physical act of drawing makes these parts look different from the middle of the path.

4 Algorithm

Data analysis in Section 3.1 tells us strokes with similar trajectory are drawn with similar hand pose. Given a query stroke trajectory, we can “hallucinate” plausible hand pose from segments of library strokes that have locally similar trajectory. The synthesis algorithm operates on a stroke by stroke basis, processing each stroke in isolation, so inter-stroke effects are not considered. In an interactive painting program, on mouse-up after a stroke is drawn, the synthesis algorithm is run and the new stroke replaces what the user drew (in a fraction of a second for typical strokes).

In both handwriting and line drawings, the hand pose attributes themselves do not fully represent an artist’s style. We observe that one of the most important cues is the trajectory itself. Given a query trajectory that might be drawn with the shaky hand of an amateur user, we can find a new trajectory that follows the user’s overall intention but has the style and expertise of the artist who drew the library. This application relies on the same algorithm developed for pose hallucination. We replace the query trajectory with similar, but not identical, patches from the library (striking a balance between following query intent and preserving library style).

The algorithm operates in a series of stages, where for each stage, we consider different implementations to support various data and performance tradeoffs. These stages are: computing feature vectors (Section 4.1), selecting approximate nearest neighbors (Section 4.2), and post processing (Section 4.3). Figure 5 offers a concise overview of these stages and implementation options.

4.1 Feature Vectors

Our process begins with an offline step, where for each sample in the library we compute a feature vector that describes the local shape, using either *shape contexts* (Section 4.1.1) or *filtered velocities* (Section 4.1.2). Then online, for a given query stroke, synthesis

Preprocess:
- • Compute feature vectors for library samples: (§4.1)
  - choice { ∘ shape contexts (§4.1.1)
  - ∘ filtered velocities (§4.1.2)

Online:
- • Compute feature vectors for queries, as above (§4.1)
- • Find k nearest neighbors (k-NN) of each sample (§4.2)
- • Select neighbors from k-NN for synthesis:
  - choice { ∘ closest neighbor (§4.2.1)
  - ∘ weighted average (§4.2.2)
  - ∘ optimal sequence (§4.2.3)
- • Post process:
  - ∘ for optimal: transition blending (§4.3.1)
  - ∘ for trajectory: shape optimization (§4.3.2)

Figure 5: *Algorithm overview.*

begins by computing feature vectors for the query samples, and then searching for similar features in the library.

One goal of the feature vector is to characterize the local trajectory of the query (or library) strokes. Therefore, our feature vector takes into account samples in the “recent history” and “near future” as in the patch construction from Section 3.1. The pose attributes at the start or end of a stroke have different distributions than those in the middle (Section 3.3). Therefore, a second goal for the feature vector is to encode closeness to the endpoints, where appropriate. We define the scalar $x_i^- = \min(1, \sqrt{d/d_h})$ where d is the arc length distance to the start of the stroke and d_h is the size of our recent history window. Likewise we have x_i^+ measuring distance to the end of the stroke relative to the future window. These values are clamped such that the interiors of the strokes are undifferentiated. Putting this all together we have feature vector $\mathbf{F}_i$:

$$\mathbf{F}_i = \{x_i^-, x_i^+, W\{\mathbf{f}_j\} \mid j\} \qquad (1)$$

where $\{\mathbf{f}_j\}$ is the set of shape features and W is a weight that balances between the relative importance of shape features versus end features. The next two subsections consider alternatives for the shape features.

4.1.1 Shape Contexts

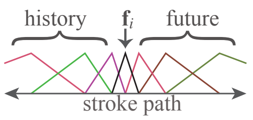
Introduced by Belongie et al. [2001], a *shape context* is a log-polar histogram of sample positions on the curve, relative to the current sample. We have adapted a variant that performs well for our application, with two modifications. First, we consider *local* shape by using only a small number of history and future samples. Second, we further divide these into two separate histograms, one for history and one for future; this division allows us to distinguish, for example, a stroke drawn left to right versus right to left.

In our experiments we have used from 25 to 40 history and future samples each, depending on the data set. The shape features $\mathbf{f}_j$ are then the histogram bins, of which we use 6 angular bins and 5 radial bins for a total of $2 \times 6 \times 5 + 2 = 62$ dimensions for $\mathbf{F}_i$.

4.1.2 Filtered Velocities

Here we describe a feature vector that is based on velocity rather than shape. We use the sample velocities $\{\mathbf{v}_i\}$, where $\mathbf{v}_i = \mathbf{x}_i - \mathbf{x}_{i-1}, \mathbf{v}_0 = \mathbf{0}$. A naïve implementation of the feature vector would have each tap correspond directly to a velocity sample from the stroke: $\mathbf{f}_j = \mathbf{v}_j$. Because of the high sampling rate, this approach would require many taps to capture a significant portion of the trajectory, and would apply equal weight to samples regardless of how near or far they are to the current sample.

Instead, each filter tap is obtained by applying a discrete triangle filter to the velocity samples, $\mathbf{f}_j = \sum w_k \mathbf{v}_k$, where w_k are the triangle filter weights, for $(i-N) \leq j \leq (i+M)$ taps. N and M are the number of history and future taps, respectively. The triangle filters overlap so that the start index and the end index of a triangle are the center indices of the adjacent triangles, which ensures that all the weights applied to a single velocity sample in a feature vector sum to one. The filter width increases exponentially with distance from sample i, which means individual samples that are distant contribute relatively less to the feature vector. When synthesizing the query samples near the start or the end of the query stroke, some of the feature taps will cover hypothetical samples that are outside of the query stroke. In that case, we only average the velocity of the samples that are valid. If all the samples under a triangle centered at j are located outside of the valid range, then we use $\mathbf{f}_j = 0$. Using triangle filters reduces the dimensionality of the feature vector needed to represent a large local neighborhood and filters out noise present in the raw samples.



Results in this paper use triangle filters with equal sized history and future windows $M = N = 7$. Thus, considering 2D velocity and two extra "endpoint" dimensions, the overall dimension of the feature vector $\mathbf{F}_i$ is $2 \times (2 \times 7 + 1) + 2 = 32$. We also use a weight $W = 0.3$ balancing shape features against endpoint features. In our experiments we have found the method to be relatively insensitive to these parameters, and robust through a broad range. Section 6 discusses the relative merits of the two feature vectors we investigated.

4.2 Selecting Among Nearest Neighbors

Once we have computed a feature vector for each sample in a query stroke, we find its k nearest samples in the library as potential samples for synthesis. Brute force search is not practical because the library is potentially large and the search must be performed for many samples. Acceleration structures like K-D trees are known to lose efficiency in high dimensional searches, as in the case of our feature vectors. Therefore, we resort to algorithms for finding *approximate* k nearest neighbors, the output of which we have found acceptable in the stages that follow. We experimented with two algorithms, the FLANN implementation of Muja and Lowe [2009], and an adaptation of the PatchMatch algorithm of Barnes et al. [2011]. The PatchMatch approach requires a straightforward modification to search over the 1-D domain of samples, and we found that with our data the algorithm converges to a reasonable approximation in ~5 passes (as Barnes et al. found for image patches). Overall, the two approaches offer roughly equivalent performance for equivalent quality of output. The major tradeoff between the methods is that FLANN expends time and memory to build auxiliary structures during the offline library construction phase; on the other hand PatchMatch is only efficient when processing the entire query stroke at once and therefore is not suitable for on-the-fly synthesis during stroke input.

Once we have the k neighbors (k-NN) for each query sample i, the next step is to synthesize a new sample $\{\mathbf{x}'_i, \mathbf{h}'_i\}$ from them. The synthesized attributes should look like those of similar shapes in the library, especially at the start and end of strokes, and vary smoothly in time. We consider three approaches—*closest neighbor, weighted average,* and *optimal sequence*—discussed in the following subsections.

4.2.1 Closest Neighbor

The most straightforward approach is to select the one neighbor with the smallest distance. That is, just use the closest neighbor to the query sample. While simple, this method does not lead to good coherence properties along the length of the stroke. Suppose library sample $\mathbf{L}_j$ is the 1-NN of query sample $\mathbf{Q}_i$. For strong coherence we want $\mathbf{L}_{j+1}$ to be the 1-NN of $\mathbf{Q}_{i+1}$, which is only sometimes true (and it becomes more rare as the library size grows). However, $\mathbf{L}_{j+1}$ is *usually* among the k-NN, which means there is an opportunity to do better.

Nevertheless, we retain this case for comparison with the other methods in Section 5. Moreover, this case may be thought of as a stand-in for the general non-parametric texture synthesis approaches following that of Efros and Leung [1999]. In particular, we also experimented with choosing randomly among the k-NN, and also the variant proposed by Ashikhmin [2001] (which probabilistically chooses $\mathbf{L}_{j+1}$ in the scenario above). However, in our experiments, none of these led to results as coherent as those of the methods that follow.

4.2.2 Weighted Average

Building on the intuition of the closest neighbor approach, interpolation is a way to provide smooth transitions. Given k neighbors $\mathbf{n}_j$, with distances d_j from the query sample in feature space, we synthesize a sample as their weighted average using weights $1/d_j$. This produces consecutive samples that vary smoothly, creating a continuous output stroke. However, the library typically contains many similar strokes (say multiple versions of the letter 'a'), each with slightly different pose and trajectory. The weighted average method therefore generates a stroke that is a compromise among the distinctive features of the library strokes, not quite reproducing any of them well. The result is that this method produces strokes that tend to be smoother than the artist's style.

4.2.3 Optimal Sequence

Both closest neighbor and weighted average are local solutions that have difficulty reproducing the smoothness and distinctiveness of the artists' style. To achieve both of these qualities, we propose a (per-stroke) global optimizing solution, which we solve using dynamic programming for efficient computation. One goal of the optimization is to select a few long *segments* of library strokes (sequences of near-consecutive samples) to be matched to segments of the query (Figure 6), avoiding many potential discontinuities.

We optimize for the sequence of transition indices $\{\mathbf{c}_i = \langle a_i, b_i \rangle \mid i = 1, 2, \cdots, s\}$ by minimizing a total synthesis cost over the s samples in the query stroke. For the i-th query sample: a_i is the index of the selected library stroke, and b_i is the index of the selected sample on stroke a_i. We seek the optimal sequence $\{\mathbf{C}_i\}$ that minimizes the sum of four error terms, e_f, e_t, e_s, and e_m. These terms address, respectively, matching features, choosing good transitions between segments, avoiding short segments, and matching stroke endpoints—discussed below.

Matching Features. The term e_f simply sums (over all query samples i) the distance in feature space to the selected neighbor $\mathbf{c}_i$.

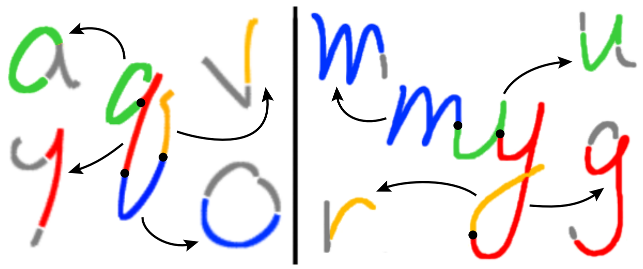


Figure 6: *Optimal matching visualization. Query strokes 'q' and 'my' are broken into parts that map to segments of library strokes.*

If all other terms in the optimization had no impact then this term would drive the solution to the closest neighbor selection method (Section 4.2.1). However there is typically a different neighbor among the k-NN that is *almost* as good as the closest neighbor and that has other desirable properties that would cause it to be selected over the closest neighbor, based on the terms that follow.

Transition Penalty. The term e_t encourages using fewer segments. For consecutive query samples, we encourage selecting samples that are consecutive on a library stroke, since they have natural coherence in the hand pose attributes and capture the local style. If that is not possible for other reasons, we still encourage repeating the same library sample once, or skipping exactly one library sample—policies that allow the library segment to be stretched or shrunk to match the query. Otherwise we call the transition a *jump*, and assign a penalty that includes both a large constant cost and a cost based on pose discontinuity at the jump location. We compute the transition penalty e_t by summing over the query e_t^i, the transition cost of going from $\mathbf{c}_i$ to $\mathbf{c}_{i+1}$. According to the policies above there are four cases:

1. *Consecutive:* If $a_i = a_{i+1}$ and $b_{i+1} = 1 + b_i$, then $e_t^i = 0$.
2. *One repeat:* If $a_i = a_{i+1}$ and $b_{i+1} = b_i$ and $b_i \neq b_{i-1}$, then $e_t^i = C_r$, the cost of stretching.
3. *One skip:* If $a_i = a_{i+1}$ and $b_{i+1} = 2 + b_i$, then $e_t^i = C_s$, the cost of shrinking.
4. *Jump:* Otherwise, $e_t^i = C_t + C_p \left\| \mathbf{h}_{c_i} - \mathbf{h}_{c_{i+1}} \right\|$, the cost of a jump with deeper penalty for large pose discontinuities.

We use $C_r = 3$, $C_s = 3$, $C_t = 50$, and $C_p = 10$. To synthesize trajectory instead of pose, we modify the transition cost e_t slightly. Rather than penalizing pose difference at the jumps, we penalize direction changes. The term e_t^i becomes:

4b. *Trajectory jump:* $e_t^i = C_t + C_p \left(1 - \mathbf{v}'_{\mathbf{c}_i} \cdot \mathbf{v}'_{\mathbf{c}_{i+1}}\right)$ where $\mathbf{v}'_{\mathbf{c}_i}$ is the normalized velocity of sample b_i in library stroke a_i.

Short Segment Penalty. We find that short segments often lead to visual discontinuities in the synthesized result. To avoid them we impose a penalty e_s taken as the sum of $(C_l + (L_{\min} - l_i))^2$ over all segments of length $l_i < L_{\min}$. In our experiments, we use $C_l = 3$ and $L_{\min} = 12$ samples.

Endpoint Penalty. As noted in Section 3.3, ends of strokes have unique characteristics. Therefore we prefer where possible to synthesize the beginning of a query stroke with the beginning of of a library stroke, and the same for ends. We impose a penalty term $e_m = C_e((b_1 - 1) + (s' - b_s))$ where b_1 is index of the neighbor selected for the first sample, b_s is the index for the last query, and s' is the number of samples in the library stroke a_s. This term essentially measures how far the neighbors selected for the ends of the query are from the ends of their respective strokes. $\mathbf{C}_e$ is the cost of not matching endpoints, which we set to $C_e = 3$.

Optimization. We solve the optimization with dynamic programming over a transition table with k rows for the k-NN by n columns for the stroke samples. To fill each entry in column i, we consider all the entries in $i - 1$. Once full, we recover the optimal sequence by selecting the lowest cost path through the table.

4.3 Post Processing

After selecting a sequence of neighbor samples, we post-process them to create the final stroke data. For the case of optimal neighbor selection we apply *transition blending*, and for the case of trajectory synthesis we perform *shape optimization*, both discussed below.

4.3.1 Pose Transition Blending

The optimal sequence method (Section 4.2.3) can still have discontinuities at the jump locations, despite the penalty term e_t^i that attempts to minimize them. Therefore, we employ blending to remove remaining artifacts. We gather as many as 6 samples (if they exist) on either side of the jump from both library strokes that abut the jump, and perform a simple cross fade between their attributes (pose and/or trajectory) in this overlap region.

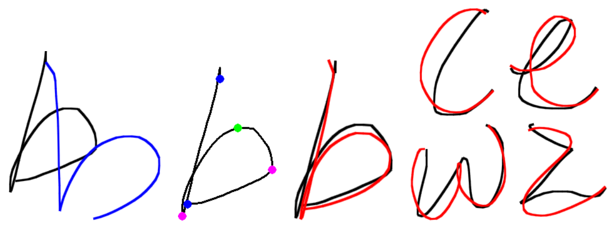


Figure 7: *Trajectory stylization of tentatively-drawn input curves, in black. (a) Integration of the output velocities results in drift, shown in blue. (b) Anchor points define tangent and velocity constraints. (c) Optimizing to satisfy the constraints results in the final stylized trajectory, in red. (d) More examples.*

4.3.2 Trajectory Shape Optimization

When synthesizing trajectories, the result can "drift" away from the query shape, so we correct it using shape optimization. Suppose $\mathbf{T} = \{\mathbf{x}_i\}$ is the input query trajectory. If we simply integrate the synthesized velocity $\mathbf{V} = \{\mathbf{v}_i\}$ starting from the position of the first sample $\mathbf{x}_1$, we get a new trajectory that looks like the library, but deviates from the query (Figure 7a black vs. blue).

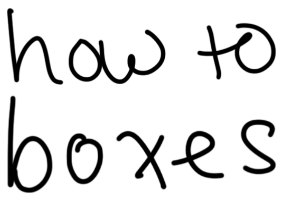
We therefore design a simple linear optimization that attempts to simultaneously preserve the synthesized shape while matching a few key features of the query. We identify a set of *anchor points* on the query, of which we have three types. We place the first type of anchor point a few samples from the start and end locations (blue dots in Figure 7b). Second, we add locations with very rapid change in orientation, as follows. For each query sample, we calculate the stroke turning angle in a local window. We find all local maxima above a threshold angle (we use 60°) and call them *angle* points (magenta dots). Finally, between successive pairs of end points and angle points we approximate the path as a polyline using a dynamic programming optimization similar to that of McCrae and Singh [2009]. The added vertices of the polylines are the third kind of anchor point (green dot).

Next we set up a system of equations composed of three constraints, each of which we wish to satisfy in a least-squares sense, in order to construct the final curve (red in Figure 7c-d). The first constraint attempts to match the final velocities $\bar{\mathbf{v}}_i$ to the synthesized velocities $\mathbf{v}_i$. The second constraint attempts to match the positions of the anchor points. To avoid second order discontinuities induced by optimization at the anchor points, the third constraint attempts to match the local curvature of the result to that of the synthesized velocities, near the anchors. With known synthesis velocities $\mathbf{v}_i$ and query positions $\mathbf{x}_i$ we seek the unknown positions $\bar{\mathbf{x}}_i$ via a system of equations:

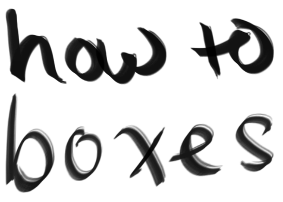
$$0 = \sum_{i=1}^{s} (\bar{\mathbf{v}}_i - \mathbf{v}_i)^2 \tag{2}$$

$$0 = C_a \sum_{i \in \mathbf{A}} (\bar{\mathbf{x}}_i - \mathbf{x}_i)^2 \tag{3}$$

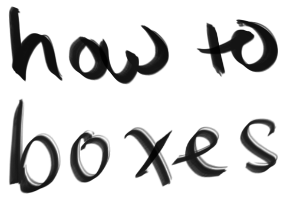
$$0 = C_k \sum_{i \in \mathbf{A}} \sum_{j=i-\varepsilon}^{j=i+\varepsilon} (\kappa(\bar{\mathbf{x}}_j) - \kappa(\mathbf{x}_j))^2 \tag{4}$$



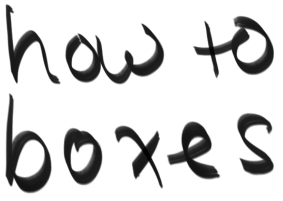
(a) *Query*



(b) *Ground Truth*



(c) *Synthesis*



(d) *Alternate Style*

Figure 8: *Handwriting pose synthesis. Query strokes (a) are ground truth strokes (b) without the pose data. (c) Our synthesis results are visually indistinguishable. (d) Applying another artist's style yields characteristically different visual appearance.*

where $\bar{\mathbf{v}}_i = \bar{\mathbf{x}}_i - \bar{\mathbf{x}}_{i-1}$, and $\kappa(\mathbf{x}_i) = \mathbf{x}_{i+1} + \mathbf{x}_{i-1} - 2\mathbf{x}_i$. The set $\mathbf{A}$ contains the indices of the anchor points. ε is the number of surrounding samples involved in the second order constraints. In our experiments, we use $\varepsilon = 2$, $C_a = 0.25$, and $C_k = 2$.

4.4 Synthesizing Both Pose and Trajectory

The output of our synthesis process so far is a sequence of poses and/or trajectories that can drive a virtual brush model. However, our pose synthesis framework uses trajectory as input. If both pose and trajectory are being synthesized, the algorithm can be run in either one or two passes. In the one pass version, the query trajectory is used to directly synthesize both the pose and new trajectory and thus the combined output does not have ideal correlation. In the two pass version, first the input stroke is used to synthesize a new trajectory, and then that output trajectory is used to synthesize pose data, which creates a data dependency but produces better correlation. Another advantage to the two-pass version is that it allows us to use separate parameters for synthesizing trajectory and pose. For example we generally tune for fewer segments in trajectory synthesis than in pose synthesis in the two-pass case.

5 Results and Evaluation

The running time of our synthesis algorithm is sublinear with the number of library samples, due to the approximate k-NN search. We found through experimentation that 150 strokes is sufficient for generating plausible results, and confirmed this through correlation analysis similar to that of Section 3. For such a library, the average running time for both pose and trajectory synthesis is 0.08 seconds per stroke. We experimented with "enriching" the library up to about 1000 strokes by including the same strokes scaled to different sizes ($\pm 50\%$), to increase matching scale-invariance, but did not see a significant improvement, and performance was reduced. We also tried using very small (fewer than 10) and very large (more than 500) libraries. With small libraries, feature matching cannot find similar enough trajectories and therefore picks random strokes to copy. Since the optimal sequence algorithm encourages long segments, the pose results still look plausible, but for trajectories, poor matches may result in complete loss of semantics in the results. Large libraries do not affect pose synthesis, whereas trajectory modification is more likely to find very similar patches to the query, reducing the effect of style transfer.

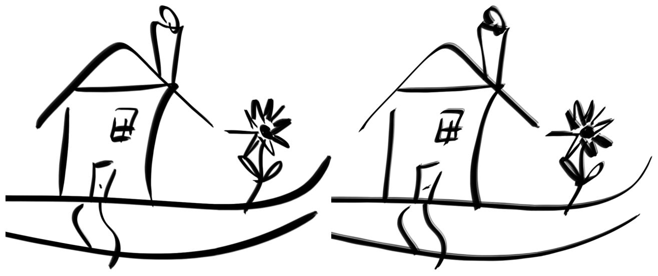


Figure 9: *Line drawing pose synthesis with two different styles.*

All the results shown here are rendered using Adobe Photoshop's bristle brush tool [DiVerdi et al. 2010] with the same brush settings. The differences in the synthesized pose data induce the visibly different shape and texture of the strokes.

Figure 8 shows the plausibility of pose synthesis on handwriting. An artist's 6-DOF strokes are stripped of their pose data and the trajectories are used as queries with the same artist's library applied. The synthesized output is visually indistinguishable from the ground truth, whereas applying another artist's library creates a visually distinct result. Pose synthesis on line drawings is demonstrated in Figure 9.

Examples of trajectory stylization are in Figure 15. We show robustness to noisy input and the utility of trajectory stylization by demonstrating how it can neaten mouse-written text, which has characteristic shakiness. Similarly, a line drawing made by an artist with an unsure hand can be made to have more confident strokes. Our algorithm can also apply stylistic flourishes such as serifs or block lettering. Figures 1, 14, and 15 show the two-pass algorithm stylizing both pose and trajectory.

The remainder of this section describes the studies we conducted to evaluate our results. As a quantitative measure of the pose synthesis quality, we computed the correlation between the synthesized results and ground truth. However, there is no equivalent measure for trajectory synthesis and ultimately we care most about how users judge our results. Therefore, we also conducted user studies on both pose and trajectory synthesis.

5.1 Quantitative Analysis

Section 3.1 shows that an artist exhibits some natural variation in hand pose even when drawing the same path multiple times. Therefore, there is no "gold standard" that we can compare our synthesis results against—L_2 measurement of reconstruction error against the ground truth is not a good indication of success. Correlation on the other hand is a better evaluation metric in this case. For pose synthesis, we can apply analysis similar to Section 3.1 to evaluate our results. If the style of the library is successfully transferred onto the query strokes, every local patch of the output should have a sim-

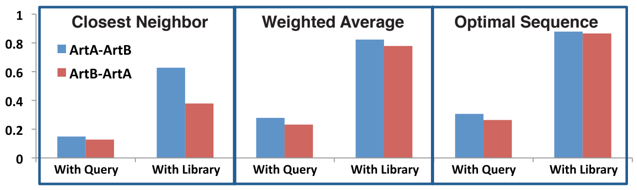


Figure 10: *Pose synthesis correlation. We apply ArtA to ArtB, and vice-versa, and see that the output is well correlated with the library and not the query, and optimal sequence performs best.*

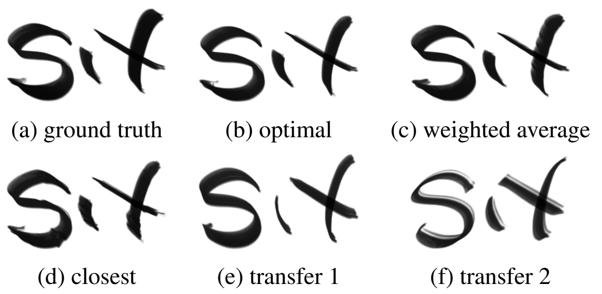


Figure 11: *Example images from pose study. (b-c) are algorithms being tested, and (e-f) have pose transferred from other artists.*

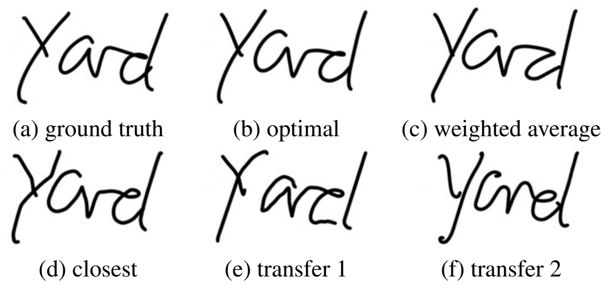


Figure 12: *Example images from trajectory study, organized as in Figure 11.*

ilar pose profile to similar patches in the library. We compared the closest neighbor, weighted average, and optimal sequence variants of our algorithm (Section 4.2). Figure 10 contains the correlation results. It shows that the optimal algorithm achieves the highest pose correlation in both synthesis cases. Furthermore, since we have ground truth pose data for the query strokes, we compare them to the synthesis and find they are not well correlated, as expected. Note that we have no analogous quality measure for trajectory synthesis because the goal is a blend of local features with global shape, and a fair objective function is more difficult to formulate.

5.2 User Studies

We conducted two user studies with the goal of evaluating whether a user can distinguish between our results and ground truth. Pose and trajectory are studied separately because users are overwhelmingly sensitive to trajectory over pose, which would make pose evaluation difficult if combined. Both studies use the same methodology; only the images are different.

Study Design. We show the subject three lines of English text from an artist's library—the *exemplar* (we used two libraries from each of the artists shown in Figure 2a-b). Below the exemplar appear two test images containing the same (3-9 letter) phrase—one *ground truth* originally written by the artist, and one *forgery* synthesized by one of our algorithms. The subject is instructed that one image is an "original" and the other is a "forgery," and is asked to identify the original by comparing both with the exemplar. In the pose study, the ground truth and forgery have the same trajectory (Figure 11). In the trajectory study, the ground truth trajectory is used for synthesis, and pose data is omitted from all images (Figure 12).

One trial consists of 26 such tasks (each with the same exemplar) comparing (a) a random ground truth phrase, and a forgery selected from one of five conditions: (b) optimal sequence, (c) weighted average, (d) closest neighbor, (e) style transfer, and (f) style transfer for validation. Conditions (b-d) are the algorithms being tested, whereas (e-f) are baselines that synthesize the forgery with another artist's library (so should be very easy to identify). Of the 26 tasks, four are from each of (b-e) and ten are from (f), randomly shuffled. For any particular subject, (e-f) are selected from two different library styles, randomly. We only retain data from a trial when 9 out of 10 of the validation conditions (f) are correct, so we know the user understands the task and is actually trying to succeed ($p = 0.01$). Then style transfer results are only reported for (e).

	correct / count (%)	
method	pose study	trajectory study
average	153 / 322 (48%)	96 / 302 (32%)
optimal	169 / 316 (53%)	172 / 305 (56%)
closest	269 / 304 (88%)	220 / 292 (75%)
transfer	293 / 301 (97%)	274 / 288 (95%)

Table 1: *Results for pose and trajectory user studies. Each entry is the number of times ground truth was correctly chosen out of a number of paired comparisons for each condition.*

Subjects were "workers" on the Amazon Mechanical Turk, a micro-job marketplace that has been used for a growing variety of such studies [Cole et al. 2009]. Our studies were restricted to US-only workers who were paid \$0.20 per task (a "HIT" containing the 26 comparisons, which they typically completed in a few minutes). Across the two studies, 70 workers completed a total of 214 HITs. Subjects were allowed to complete as many as 10 HITs per study, but most workers did just one. For cases where a particular pair was repeated by a particular worker due to the randomized selection, we only retained the first such response, for total of 4,170 responses.

Study Results. The results are reported in Table 1 as the frequency with which the ground truth was correctly identified. A Bonferroni corrected, randomized permutation test on the distributions for each consecutive pair of rows in the table shows that they are different with statistical significance ($p \ll 0.01$) except in the case of the average and optimal conditions in the pose study ($p = 0.34$).

The ideal forgery is indistinguishable from ground truth so subjects will choose randomly, resulting in a correct answer near 50% of the time. When the forgery is easy to identify (Figure 12e), we expect scores near 100%. The *transfer* condition is near 100% in both studies, which shows people can perform the task in easy cases. People tend to identify the *closest* method (88% and 75%), so it is generally implausible. In the pose study, *average* and *optimal* are both near 50%, and therefore plausibly synthesize pose data. *Optimal* performs well in the trajectory study, but *average* is significantly below 50% which means subjects tend to incorrectly identify the forgery as the "original." As mentioned in Section 4.2.2 the weighted average method produces a very smooth output trajectory, and we believe people have a bias towards smoother curves. In such cases people choose average over ground truth, even though it is actually smoother than the exemplar and thus fails to mimic the exemplar's style.

6 Discussion

Many of our results are handwriting samples, though we do not make special accommodations for them, because handwriting provides an easy to understand and readily available source of data. Handwriting is uniquely stylized by the writer's hand pose, the local trajectory variations, and the global features such as slant angle. Handwriting also provides readily-evaluated test sets; in contrast, it is more difficult to measure performance on strokes in a drawing. We consider that the pose and local trajectory are stylistic results of motor reflexes, whereas the global shape is intentional. Our algorithm attempts to transfer style but maintain intent (via the trajectory shape optimization). For handwriting, we could use auxiliary information such as the semantic content of the text to im-

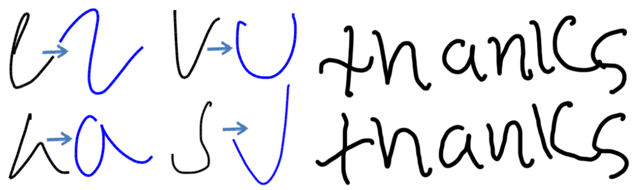


Figure 13: *Failures. (left) Filtered velocities can cause a query letter (black) to map to a different library letter (blue) with similar trajectory. (right) Filtered velocities (above) generally produce more visually pleasing results than shape contexts (below).*

prove stylization, but that would not be generalizable to other types of artistic strokes. Similarly, while an artist's style may consistently place higher weight on strokes in a drawing that are nearer to the viewpoint, we do not attempt to model these variations. In this way, we provide the most general result possible.

We examined both shape contexts and filtered velocities for feature vectors. Our results use filtered velocities, though it is difficult to say which is absolutely better. Figure 13 shows their limitations. Filtered velocities tend to perform very well, but can sometimes get "confused" and match a letter to a different but similar letter. This may be fixable with a handwriting specific algorithm, but we make no assumptions about the types of strokes being made. Shape contexts have fewer of these errors, but have difficulty balancing local style transfer with global shape, resulting in output that appears less plausible and noisy. Also it is generally more difficult to select parameters for shape contexts, whereas filtered velocities are more robust through a broader range of settings. Finally, to get high quality results, the shape contexts need to have more dimensions (see Section 4.1.2) and this impacts performance.

We also compared strategies for choosing among nearest neighbors: closest neighbor, weighted average, and optimal sequence. The result of our pose evaluations is that quantitatively, optimal sequence is best, but to the casual observer, weighted average is as effective. However, careful inspection can reveal subtle differences between the two (e.g. Figure 11b-c). For trajectory synthesis, weighted average is obviously more smooth, and in some cases too smooth. The advantage weighted average has over optimal sequence is that it does not require the query stroke be completed before it begins synthesizing, so it can be used in an "on-the-fly" implementation that synthesizes pose data while the user is still creating the stroke. Otherwise, optimal sequence results in the highest quality output.

Finally, our approach does not require hand-drawn input. The method applies to any kind of 2D path, such as Bézier curves (see Figure 14). The paths are first sampled uniformly in arc length and then treated as constant-velocity brush strokes by our unmodified algorithm. This approach would also work for lines extracted from 3D models or other automatic processes.

Limitations and Future Work. Our algorithm considers each stroke independently, but artistic styles also include interactions among nearby strokes, both temporally and spatially, which we hope to model in future work.

Our results are all rendered by Adobe Photoshop's bristle brush, which uses all 6-DOF of the stroke data. However the visual impact of each DOF may not be as obvious as it would be with a real physical brush. Because our algorithm only generates stroke data, it can be used with any digital paint engine, so other brush simulations may be able to use the data more effectively.

Figure 13 highlights some trajectory synthesis failures where the query shape was poorly matched by our algorithm. Our approach is local and has no higher-level notion of intent on the part of the

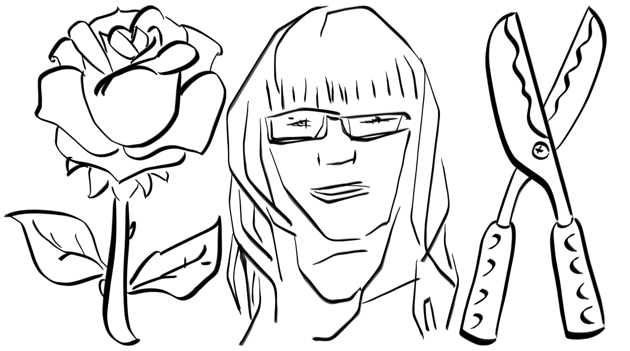


Figure 14: *(left, middle) Pose and trajectory stylization. (right) Pose synthesis for vector line art from Adobe Illustrator.*

artist. While the cognitive process is too complex to model there may be some intermediate levels that could help. Moreover, these failures could be addressed in our current framework by choosing different parameter settings. Our approach works well, especially for pose synthesis for a broad range of settings, but finding the *ideal* values for an arbitrary problem is difficult.

Finally, we treat the artist's library as a single monolithic style. In practice, artists may choose among multiple styles, even in a single drawing. A style is really more multi-modal than is characterized by our process. We can partially address this issue by offering the user the choice of different styles, but finding an effective interface for this remains an interesting problem. More challenging would be to try to characterize the multi-modal nature within a library.

Conclusion. This paper presents a data-driven approach for synthesizing plausible pose data that can be used to generate expressive handwriting and line drawings. The same framework can also be used to transfer stroke trajectory style. We collect a library of strokes drawn by artists on a 6-DOF tablet in Adobe Photoshop and analyze the pose data as insight for future research in this area.

Acknowledgements. We thank the anonymous volunteers who contributed both writing samples and drawings to this project. This research was sponsored in part by Adobe Corporation and conducted in conjunction with the Intel Science and Technology Center - Visual Computing.

References

ASHIKHMIN, M. 2001. Synthesizing natural textures. In *Symposium on Interactive 3D Graphics*, 217–226.

BARNES, C., GOLDMAN, D., SHECHTMAN, E., AND FINKELSTEIN, A. 2011. The PatchMatch randomized matching algorithm for image manipulation. *Commun. ACM 54* (Nov.), 103–110.

BAXTER, W., AND LIN, M. 2004. A versatile interactive 3D brush model. In *Pacific Graphics*, 319–328.

BELONGIE, S., MALIK, J., AND PUZICHA, J. 2001. Shape matching and object recognition using shape contexts. *Trans. on Patt. Anal. and Mach. Int. 24*, 509–522.

COLE, F., SANIK, K., DECARLO, D., FINKELSTEIN, A., FUNKHOUSER, T., RUSINKIEWICZ, S., AND SINGH, M. 2009. How well do line drawings depict shape? In *SIGGRAPH*, vol. 28.

DIVERDI, S., KRISHNASWAMY, A., AND HADAP, S. 2010. Industrial-strength painting with a virtual bristle brush. In *Virtual Reality Software and Technology*, 119–126.

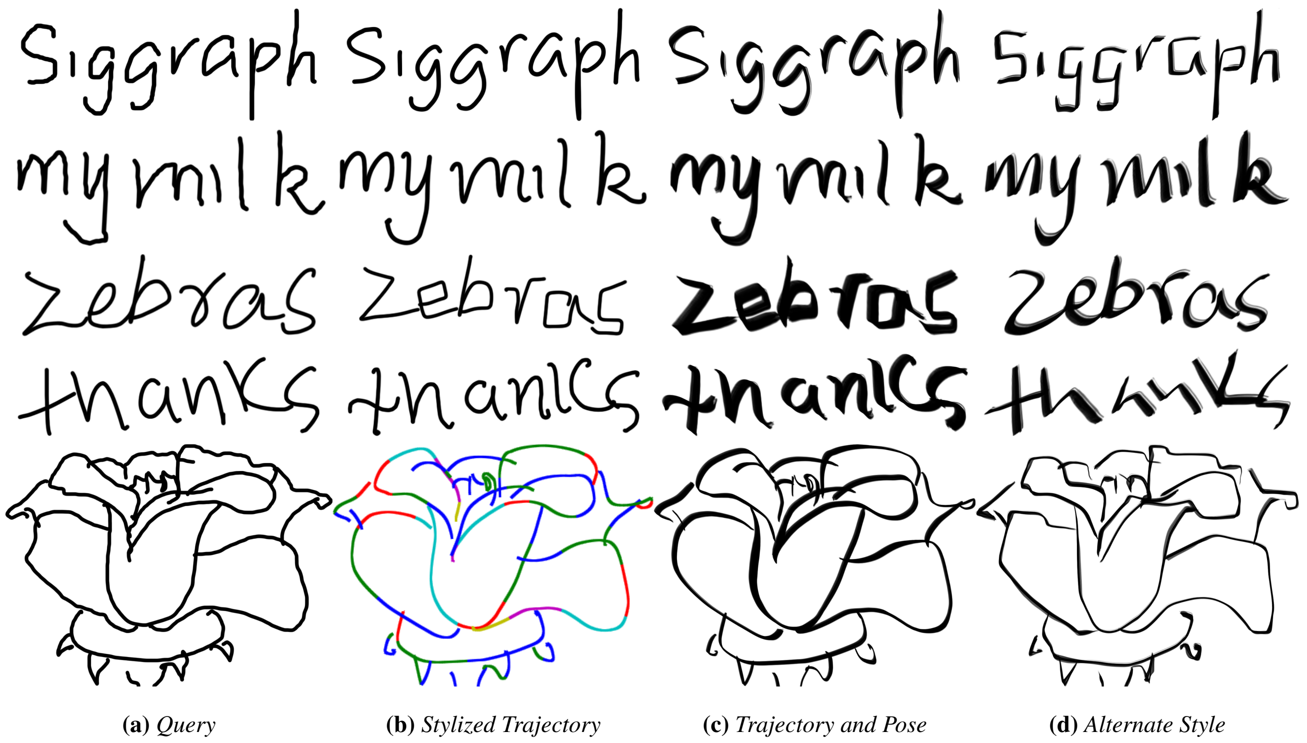


Figure 15: *Combined stylization. The words "Siggraph" and "my milk" are written with the mouse. The words "zebras" and "thanks" are written using a 2DOF stylus. The flower is drawn by an amateur user. The different colors represent different consecutive segments.*

EFROS, A., AND LEUNG, T. 1999. Texture synthesis by non-parametric sampling. In *International Conference on Computer Vision*, 1033–1038.

FRANKE, K., SCHOMAKER, L., AND KÖPPEN, M. 2005. Pen force emulating robotic writing device and its application. In *Workshop on Advanced Robotics and its Social Impacts*, 36–46.

GOODWIN, T., VOLLICK, I., AND HERTZMANN, A. 2007. Isophote distance: a shading approach to artistic stroke thickness. In *Non-Photorealistic Animation and Rendering*, 53–62.

HERTZMANN, A., OLIVER, N., CURLESS, B., AND SEITZ, S. 2002. Curve analogies. In *Eurographics Workshop on Rendering*, 233–246.

HINTON, G., AND NAIR, V. 2005. Inferring motor programs from images of handwritten digits. In *Advances in Neural Information Processing*, 515–522.

HOUSE, D., AND SINGH, M. 2007. Line drawing as a dynamic process. In *Pacific Graphics*, 351–360.

HSU, E., GENTRY, S., AND POPOVIĆ, J. 2004. Example-based control of human motion. In *Symposium on Computer Animation*, 69–77.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion graphs. In *SIGGRAPH*, 473–482.

MCCRAE, J., AND SINGH, K. 2009. Sketching piecewise clothoid curves. In *Sketch-Based Interfaces and Modeling*.

MCCRAE, J., AND SINGH, K. 2011. Neatening sketched strokes using piecewise french curves. In *Sketch-Based Interfaces and Modeling*, 141–148.

MUJA, M., AND LOWE, D. G. 2009. Fast approximate nearest neighbors with automatic algorithm configuration. In *Internat. Conf. on Comp. Vision Theory and Application*, 331–340.

OKABE, Y., SAITO, S., AND NAKAJIMA, M. 2005. Paintbrush rendering of lines using HMMs. In *GRAPHITE*, 91–98.

PLAMONDON, R. 1995. A kinematic theory of rapid human movements. *Biological Cybernetics 72*, 295–320.

SAITO, S., KANI, A., CHANG, Y., AND NAKAJIMA, M. 2007. Curvature-based stroke rendering. *Visual Computing 24* (November), 1–11.

SCHÖDL, A., SZELISKI, R., SALESIN, D. H., AND ESSA, I. 2000. Video textures. In *SIGGRAPH*, 489–498.

SILVER, N., AND DUNLAP, W. P. 1987. Averaging correlation coefficients: Should fisher's z transformation be used? *Journal of App. Psych. 72(1)*, 146–148.

THIEL, Y., SINGH, K., AND BALAKRISHNAN, R. 2011. Elasticurves: exploiting stroke dynamics and inertia for the real-time neatening of sketched 2D curves. In *User Interface Software and Technology*, 383–392.

VANDOREN, P., VAN LAERHOVEN, T., CLAESEN, L., TAELMAN, J., RAYMAEKERS, C., AND VAN REETH, F. 2008. IntuPaint: Bridging the gap between physical and digital painting. In *Horizontal Interactive Human Computer Systems*, 65–72.

VARGA, T., KILCHHOFER, D., AND BUNKE, H. 2005. Template-based synthetic handwriting generation for the training of recognition systems. In *Conference of the International Graphonomics Society*, 206–211.

WEI, L.-Y., LEFEBVRE, S., KWATRA, V., AND TURK, G. 2009. State of the art in example-based texture synthesis. In *Eurographics 2009, State of the Art Report*.

YU, K., WANG, Y., AND TAN, T. 2004. Writer identification using dynamic features. In *Biometric Authentication*, D. Zhang and A. Jain, Eds., vol. 3072 of *Lec. Notes in Comp. Sci.* 1–8.