Curve Stylization by Example using Histograms of Curvature

Edy Garfinkiel[*] University of Toronto
Pierre Bénard[†] University of Toronto
Aaron Hertzmann[‡] University of Toronto

Abstract

We present an approach for transferring the style of an exemplar curve to a target curve by analyzing and matching their distribution of curvatures at multiple scales. The user inputs two curves, an exemplar curve possessing a desired style and a guide curve determining the overall path of the output curve. We hypothesize that curvature is one of the fundamental features describing the style and use statistical analysis of the curvatures to transfer elements of the style to the guide curve. Histograms of curvatures of both curves are matched progressively by an iterative re-weighting scheme that minimizes an energy function balancing closeness of the curvature distributions and distance from the guide curve to the output curve. Our algorithm is able to match histograms well and produces interesting outputs in the single and multi-scale approach. Our results show that curves of similar style have similarly matching histograms at multiple scales, which leads to the transfer of elements of the style.

Keywords: line drawings, non-photorealistic rendering, style transfer, curvature.

Links: DL PDF

1 Introduction

Style is an inherent part of line drawings, paintings, fonts and virtually all forms of art as an expression of the artist conveying a message to the viewer. In this paper, we target curves from line drawings and draw motivation from the different ways they can be portrayed. Style gives a deeper level of meaning to a drawing varying from artist to artist and depends on the application it is intended for. It isn't completely understood how style is conceived by an artist, however it is hypothesized that the style of a drawing can be separated from its underlying content, implying that it could be stored and applied to other content. The cartoon bear of Figure 1 will serve as our main motivating example. Depending on the context, the cartoon can be drawn in different styles providing different looks and feels while the underlying content remains the same, namely the bear. The intent in the illustrations of Figure 1 is to present the same bear, only with a different style. Fonts are another similar example where the font itself is the style with the underlying content across fonts, namely the character, remaining the same.

Talented artists carry the ability to draw graceful strokes and curves

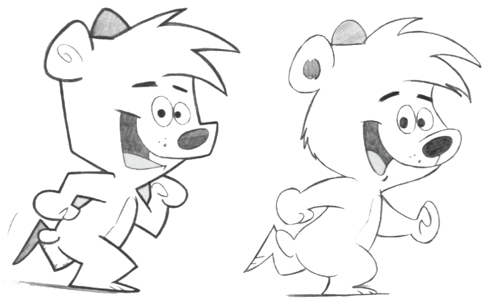


Figure 1: *"Retro-Style" left, "Classic Cartoon-Style" right. source: The Cartoonist's Big Book of Drawing Animals, Christopher Hart, page 21.*

that make drawings highly expressive. Most casual users, however, do not have these skills and wish to be able to compose artistic-looking drawings. As a motivation to provide novice users tools to achieve this as well as to enhance the work of professional artists, we wish to learn more about the style of drawings. In doing so we hope to discover novel ways of transferring style by better understanding what its best descriptors are while building on previous techniques that have been researched. Although this has been studied [Tenenbaum and Freeman 2000], it remains a difficult and unsolved problem as we do not completely know how humans perceive and decouple style from content.

A notable feature of curves that give a drawing style is curvature. We believe this to be an important feature that holds stylistic information and believe it to have the potential to form the basis for a novel algorithm. Line drawings exhibit curves featuring high curvatures, smooth curves with low curvatures and sharp corners with infinite curvature, among others. A set of hand drawn curves that can be described with curvature is shown in Figure 3. Wiggles form a style that exhibits varying curvature adding a certain touch to curves. For example, Figure 2 displays a cartoon given a meaningful jelly-like look using this style. Smooth round curves can give another perspective to a line drawing as in the right of Figure 1. Clothoids are mathematical curves that possess a uniform curvature profile. That is curvature varies linearly along its arc length and such curves can be described concisely by means of this feature. They appear naturally in some art forms and give pleasing gracious strokes. Improving curves by fitting [McCrae and Singh 2009] piecewise clothoids can greatly beautify hand drawn strokes. Curvature may vary as well with the scale in which the drawing is observed. For example, an illustration drawn with many wiggles has high fine-scale curvatures as can be seen in the left of Figure 6. Regardless of the scale in which it is observed, curvature is an obvious feature that describe curves in how they are shaped on paper.

We approach the problem of transferring curvature statistics from an exemplar curve with a desired style to a target curve. We believe that curvature is a good descriptor of style and thus curves of the same style are likely to have similar curvature statistics at different scales. Following this hypothesis, matching the statistics of curvatures in a curve to those of a curve with a desired style will transfer the style. For instance, it should be possible to transfer the wiggles of an exemplar curve to a smooth target path. While

[*]egarfink@dgp.toronto.edu
[†]pierre.benard@laposte.net
[‡]hertzman@adobe.com

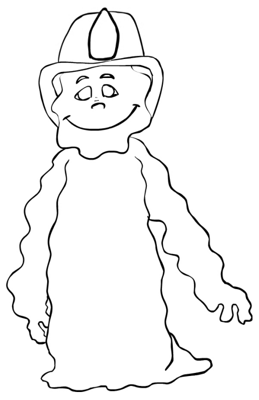
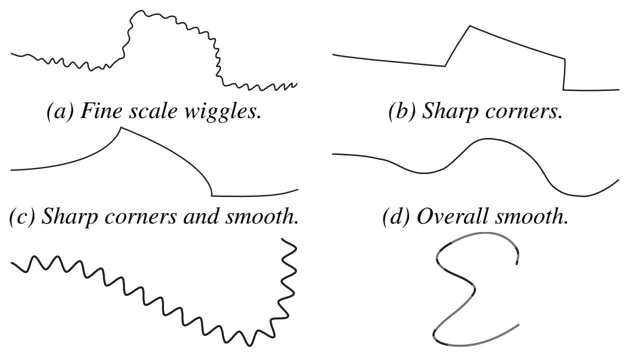


Figure 2: *The wiggly body has high curvature in the low-scale but low curvature in the high-scale. In other words, the overall path is relatively straight but the finer details are not.*



(a) Fine scale wiggles. (b) Sharp corners. (c) Sharp corners and smooth. (d) Overall smooth. (e) Uniformly wiggly. (f) Piecewise clothoids.

Figure 3: *All curves are hand drawn and traced. Curve (a) has natural variation in curvatures; (b) & (c) have sharp corners; (d) has a relatively smooth curvature profile; (e) is overall uniform in its curvature variation at the fine scale; (f) is a curve composed of piecewise clothoid splines with a smooth curvature profile (fit to a hand drawn polyline using the approach of [McCrae and Singh 2009]).*

transferring statistics, curves are analyzed at multiple scales due to the fact that the curvature can be described differently according to scale. Curves may have low-scale variations, high-scale variations and a varying mixture in the spectrum of scales.

Taking inspiration from a gradient distribution matching technique for image restoration [Cho et al. 2012], we develop a similar algorithm for curvature distribution matching. By globally matching histograms of curvature, we are able to introduce elements of the style onto a target curve while retaining its original content.

2 Related Work

2.1 By-Example Line Drawing Stylization

There have been several works in the area of style transfer using various techniques for capturing style. Early notable works include Skeletal Strokes [Hsu et al. 1993], which deal with texture mapping onto a known base path. To add expressiveness to the path, a "skeleton" with a reference x-axis and thickness-axis is drawn onto an image or rastered texture, defining a stroke. Using this stroke definition, the image can be deformed by stretching, compressing or bending over arbitrary user-drawn paths to create stylized strokes and abstract deformations of images. This simple and effective approach made it possible for users to draw vector graphics with styles at interactive rates, greatly enhancing style in computer graphic art. This method however is limited in that it stretches an image to give a curve an apparent style, which can look distorted depending on the degree of stretching.

Texture synthesis, much related to style synthesis, saw great strides in advancements by Portilla and Simoncelli [2000] who presented a parametric statistical approach to visual textures. By sampling under their model, new textures can be synthesized producing many images in the same style. Using this statistical approach, sampling can generate new images avoiding severe distortions when stretching the texture or unnatural repetitions when tiling. The style of a curve can be abstracted away from its base path so as to decouple the two pieces of information. In this case, the style itself is defined by the details on the underlying base path, however a preexisting decomposition between the two is not known ahead of time. The approach of Bénard *et al.* [2012] exploit similar geometry and self-similarity in styles using offset synthesis to achieve a parametric, style-by-example approach at varying levels of detail. Non-parametric methods to style transfer have been approached by Kalnins *et al.* [2002] by taking a style's brush geometry and treating it as a texture. By formulating it as displacement mappings of the vertices of a base, synthesized styles over new paths are formed.

Novel ways of building a decomposition to automatically extract displacement offsets from curves can be done through the use of multiresolution wavelet decomposition [Finkelstein and Salesin 1994], whereby a curve is decomposed into multiple scales of lower resolutions with diminishing detail. The lost details at each coarser resolution are captured as a separate signal that can be used independently. With the path isolated from the style, it can be modified into a new path with the style later reapplied to it. As well, the style information can be extracted and transferred to a different curve altogether [Brunn et al. 2004]. With this approach, varying styles can be reproduced extremely well.

Extracting the decomposition of curves is a challenging approach that mostly captures the detail of the style. The following work examines a broader description of style, considering the curve as a whole. A novel non-parametric approach for synthesizing style by "analogy" was conceived by Hertzmann *et al.* [2002]. Given an "unstyled" curve A along with a styled version A', the transformation between them is learned for application to a new curve B. Local, well-matching pieces of the styled curves are used to synthesize the style along the path of the input curve B. This is similar to constrained texture synthesis applied on curves instead of pixels. This paper considers the transfer of statistics at multiple scales of curves inspiring some of the work we undertake in this paper. Learning-based approaches are a different perspective on style transfer whereby a training set of curves of different styles are used to modify the content of another drawing to take on the style of the training set [Freeman et al. 2003]. Using a linear combination algorithm to form a mixture of the *k* nearest neighbour curves, line drawings take on varied styles from the set of supplied curves. A major drawback of this approach is the requirement of an adequately sized dataset to capture enough sample curves of a style.

A key goal in style transfer is for the synthesis of strokes for novice users. By drawing a rough outline of their intention, strokes can be improved via an interactive system. Synthesizing strokes for users of lower-end tablets in HelpingHand [Lu et al. 2012] is done through the use of data generated from artists using higher-end tablets that allow for higher degrees of input. Using higher-dimensional curve descriptors to engineer feature vectors for training strokes, new strokes are synthesized for the user in real time using optimal matching. Simhon and Dudek's [2004] developed an interactive system for automatically generating illustrations from coarse curve strokes. They make use of Hidden Markov Models to synthesize 2D curves through the use of a training set of stylized examples with the ability to mix instances of the training set. Using their statistical model, they achieve some similar results to Hertzmann *et al.*[2002].

2.2 Histogram Matching

Cho *et al.* [2012] developed a novel algorithm for the restoration of images such that their reconstructions have similar gradient distribution to a reference distribution. By imposing a global constraint on the gradients of the image, an iterative penalizing scheme is used to modify its statistics in order to bring the gradient distribution closer to that of a desired distribution. By reassessing the penalty function iteratively using the KL divergence of the distributions, the penalty function is progressively fine-tuned to bring the distribution of the image gradients closer to the desired distribution.

This statistical approach leads to pleasing restorations in images and inspires the underlying idea for use in curves. We develop a similar penalty function using a curvature measure for polylines and use a desired distribution of an exemplar curve as our "reference" distribution. We perform a similar iterative scheme that updates the penalty function as the distribution of curvatures changes, in order to modify the statistics of a curve to have distribution similar to that of a desired distribution. This is done at multiple scales of the curve in order to introduce details of the style present over the spectrum of scales.

Our approach in the transfer of a style by means of example curves and capturing their statistics is very much complementary to the work of Hertzmann *et al.* [2002] in which local neighbourhoods are matched non-parametrically. Our approach however is parametric, using curvature to globally match the statistics of a supplied curve with those of a desired style. Following our hypothesis on curvature as a meaningful descriptor of style, we apply a similar statistical matching algorithm for images as presented in the work of Cho *et al.* [2012].

3 Overview

We now describe the general setup of how curves are represented and the intuitive goal of our algorithm. We experiment with both curvature approximation on polylines as well as turning angles, each having their own tradeoffs, however for simplicity, we refer to our main feature as "curvature" when overviewing the algorithm. Turning angles have a few advantages such as being bounded to $[0, \pi)$ and independence to the distance between the sample points. A drawback of turning angles is their dependence on the sampling rate. A dense sampling rate will produce turning angles nearly all close to 0 while a sparse sampling may increase the turning angles for the same sample point. The curvature approximation measure takes into account the distance to neighbouring points attempting to remain invariant to sampling.

Let the curve describing the path of the final output be referred to as the *guide* curve denoted by g and the curve possessing the desired style be referred to as the *exemplar* curve denoted by e. The exemplar curve is used to extract statistical information of curvature to build a *desired histogram* H_d. Though the input curves are in analytic form, we focus on discrete polyline approximations of curves using N points sufficing at capturing elements of the curve's style. The terms "polyline" and "curve" are used interchangeably. We define an energy function over the sampled points, perturbing the points to find an energy minimum.

The polyline $p \in \mathbb{R}^{2N}$ approximates a curve with the points $P_i = (x_i, y_i), i = 1, ..., N$. The interior points of the curve have a turning angle θ_i associated with them where θ_i denotes the turning angle at the point P_i for $i = 2, ..., N-1$, defined in terms of the dot product:

$$\theta_i = \cos^{-1} \frac{\vec{v}_i \cdot \vec{v}_{i+1}}{\|\vec{v}_i\| \|\vec{v}_{i+1}\|} \tag{1}$$

where $\vec{v_i} = \overrightarrow{P_i P_{i+1}}, i = 1, ..., N-1$. The curvature estimator [Farin et al. 2002] experimented with is defined as:

$$\kappa_i = \frac{\theta_i}{(\|\vec{v}_i\| + \|\vec{v}_{i+1}\|)/2} \tag{2}$$

where κ_i is the curvature measure at the point P_i.

The problem statement is the following: given an exemplar curve e and a guide curve g, we would like the output of our algorithm to be a target curve t that captures the style of e while remaining close to g. To this end, we begin with the target curve as the guide curve and attempt to match the target histogram H_t (the histogram of curvatures of the target curve t) with the desired histogram H_d of e, while maintaining a close distance to g. Over a set number of iterations we transform t so that H_t and H_d are matching under a histogram distance metric. With the goal of transferring statistics from e to t, we wish for elements of the style of e to appear in t. We illustrate in Figure 4 the two required inputs and the desired output.

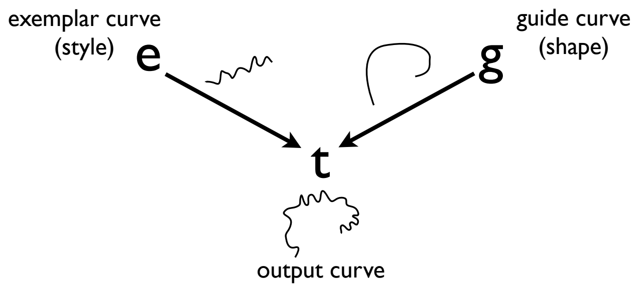


Figure 4: *The exemplar curve e gives style information while the guide curve g gives path information in order to create an output curve t having closely matching histogram with e while remaining close to the guide curve.*

4 Energy-based model

Our goal is to minimize a distance between histograms. This however is not easy to perform directly as finding the gradients of such an energy function is challenging. Gradient-free optimization methods such as Covariance Matrix Adaptation (CMA) can be used, however they may be very slow. Therefore, we design a curvature penalty function as in the Cho *et al.* paper [Cho et al. 2012]. It is used in the general formulation of our energy function and is modified over successive iterations to minimize indirectly the distance between histograms.

We express this problem as the minimization of the sequence of energy functions $(E^{(n)})$, each of them being formulated as:

$$E(t; g, H_d) = E_{guide}(t; g) + E_{hist}(t; H_d), \tag{3}$$

where t is the curve being optimized. It is important to note that this energy function is dynamic and changes over the course of iterations. This is the general formula and not a static energy function but rather the energy for a particular step of our minimization.

The energy function consists of two terms. The guide term E_{guide} encourages the target curve to stay close to the guide curve while the histogram term E_{hist} enforces the target distribution of curvatures H_t to match the desired one, H_d. The guide term E_{guide} is a distance metric between the argument curve t being minimized and the guide curve g which we set to the sum of squared distances:

$$E_{guide}(t; g) = d^2(t, g) = \sum_{i=1}^{N} \|T_i - G_i\|^2 \tag{4}$$

where T_i, G_i are the points on the target and guide curve respectively. The target curve t is initially set to the guide curve g and the sample points are perturbed while the energy is evaluated. As such, both the guide curve and the target curve have the same number of sample points.

The histogram term E_{hist} is based on the definition of a penalty function ρ, which seeks to deter or encourage curvatures in a desired range. It is defined as follows:

$$E_{hist}(t; H_d) = \sum_{i=2}^{N-1} \rho(\kappa_i^{(t)}) \tag{5}$$

where $\rho(\kappa_i^{(t)})$ is the penalty for the curvature κ_i at the point T_i on the polyline t. Each $E^{(n)}$ has its respective penalty function $\rho^{(n)}$. The penalty function ρ is defined in terms of the desired histogram H_d and the target histogram H_t. Intuitively, it attempts to deter curvatures that are too great in abundance by penalizing them (giving them a higher energy value) and encourage desired curvatures that are not present enough by rewarding them (giving them a lower energy value). With this approach in mind, we build a penalty function that assigns penalty values to curvature ranges and assesses the overall energy of the curve being optimized to find one with low energy.

5 Iterative re-weighting minimization

We first describe the penalty function ρ and its initialization that paves the way for the single-scale approach, explained in section 5.1. The basic overview of the algorithm is to minimize the sequence of energy functions $(E^{(n)})$, effectively perturbing the set of points in t until a good balance between both E_{guide} and E_{hist} is found. Once achieved, the output curve t should have closely matching histograms with e while following the overall path of the guide curve, as outlined in Figure 4. The minimization of E is done by an iterative scheme that penalizes and rewards certain ranges of curvature in order to match the histograms H_t and H_d. We measure the distance between histograms using the Quadratic-Chi (QC) [Pele and Werman 2010] distance. To construct H_d, a suitable number of bins n must be chosen so as to give a good representation of the underlying distribution of the exemplar curve. There are various heuristics to find a good number (e.g. Freedman-Diaconis', Scott's and Sturges' among others). Once H_d has been computed, the penalty function ρ is initialized to:

$$\rho^{(0)}(\kappa) = \text{-}w_1 \ln(H_d(\kappa)) \tag{6}$$

where w_1 is an initialization weight parameter responsible for setting the initial penalties to drive the curvatures toward the range of those found in the desired histogram. In other words, curvatures lying far out where frequencies in H_d are low receive a high penalty. This approach follows closely the work of Cho *et al.* [Cho et al. 2012]. With this initialization, ρ is set to a reasonable starting point encouraging the desired curvatures in the exemplar curve to be introduced in the target curve. Once initialized, E is minimized with respect to t by gradient descent using finite differences for gradient vector approximations.

We note that since dealing with discrete probability distributions, the result of ρ at any stage is a step function and penalties are assigned to respective bins. However when computing penalties, rather than assigning curvatures within the same bin the same penalty, we linearly interpolate between the penalties of neighbouring bins and assign curvatures this value for a smoother overall penalty function.

Following this minimization step, the penalty function is updated and penalties for particular ranges are reassessed such that curvatures too high in abundance are penalized further while curvatures not present enough are rewarded to produce a new updated penalty function. The update rule after each iteration is:

$$\rho^{(n+1)}(\kappa) \leftarrow \rho^{(n)}(\kappa) + w_2 \ln(\frac{H_t(\kappa)}{H_d(\kappa)}) \tag{7}$$

where w_2 is a correction parameter responsible for adjusting the penalty function as it is reassessed. The update rule uses the histogram H_t of the current curve t and compares it with H_d. By using the natural logarithm, frequencies in curvature ranges that surpass that of H_d cause the curvatures in those bins to be penalized and frequencies in curvature ranges that are below those of H_d to be rewarded. Moreover, frequencies that match those of H_d are left unchanged, noticing that the correction sum in equation (7) is 0. This is illustrated with an example in Figure 5. This iterative reweighting scheme encourages desired curvatures to approach the frequencies present in H_d and fine-tunes the penalty function to reach a solution curve that has low energy with H_t close to H_d.

Using the updated function we minimize the new energy function and repeat the procedure. This is done for a set number of iterations while storing the curve t between iterations that produces the closest histogram H_t to H_d.

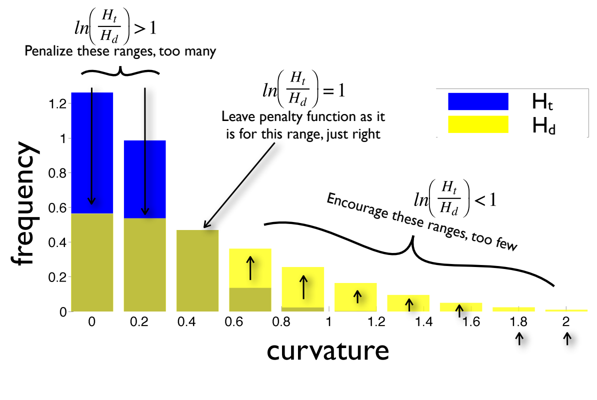


Figure 5: *Desired histogram superimposed with the histogram of the target curve as it is being optimized. Penalties for the curvature ranges are updated according to equation* (7).

5.1 Matching at one scale

Here we describe the general framework of Algorithm 1 used to match histograms at a single-scale that becomes the basis for the multi-scale approach. The idea behind the algorithm is to iteratively reassess the penalty function with the goal of introducing the desired curvatures in the target curve until the histograms H_t and H_d are relatively close, based on the QC distance. Beginning with inputs e and g sampled by arc length, we initialize $t = g$ and compute their histograms H_d and H_t respectively. Using equation (6), ρ is set to its initial value.

We set a maximum number of iterations for the "outer loop" in line 8 of the algorithm. Between these iterations, ρ is updated using equation (7) and the curve t having the lowest QC distance is stored. If the QC distance in subsequent iterations rises and continues to do so, we stop the optimization and use the best matching curve as our final output. We presume that there exists a close curve with a matching histogram, however this is affected by a number of variables. Our iterative re-weighting scheme seeks to find this curve of low energy though it does not guarantee convergence to the desired histogram. As such, we continue to minimize until the lowest attainable energy of a curve is reached, always storing the one with best histogram distance and using it as our final output.

Algorithm 1 Histogram Matching at One Scale

```
 1: procedure HISTMATCH($e, g$)
 2:     $maxIterations = C$
 3:     $n = 1$
 4:     compute $H_d$          ▷ Histogram of curve $e$ with desired style
 5:     $\rho(\kappa) \leftarrow -w_1 \ln(H_d(\kappa))$
 6:     $t \leftarrow g$
 7:     compute $H_t$          ▷ Histogram of curve $t$ being optimized
 8:     while $n \leq maxIterations$ do
 9:         $\rho(\kappa) \leftarrow \rho(\kappa) + w_2 \ln(\frac{H_t(\kappa)}{H_d(\kappa)})$
10:         $t \leftarrow \underset{t}{\operatorname{argmin}}\, E(t)$ ▷ Gradient descent using Finite Diff
11:         $n \leftarrow n + 1$
12:     end while
13:     return $t$   ▷ Curve with closely matching histogram to $H_d$
14: end procedure
```

5.2 Matching at multiple scales

Drawn curves have detail at multiple scales which together compose the overall style of the curve. There are various finer to coarser details that must be analyzed over the spectrum of scales. For example in Figure 6, the hand drawn curve has an overall relatively smooth path as seen on the right scale, however its finer details following this path are seen on the left scale. Such wiggles are an example of fine-scale detail, which can only be captured by analyzing the various scales. Therefore to apply our statistical approach of style transfer, we analyze multiple scales and match histograms in this spectrum in order to capture the fine wiggles.

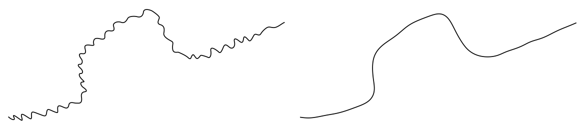


Figure 6: *Hand drawn curve in two scales: left scale captures wiggles, right scale captures overall path.*

Our multi-scale approach works iteratively matching coarser to finer scale histograms of the desired and target curves using the single-scale approach at each step. A curve t at scale s is denoted as t^s and increasing values of s represent coarser scales of the curve (where $t^0 = t$). There are more sophisticated ways to decompose a curve into scales [Hahmann et al. 2007; Finkelstein and Salesin 1994], however for the purposes of our experiments, a curve at scale s is constructed by only considering every $k^{th} = 2^s$ sample point, dependent on the total number of sample points in the curve. This is done in line 6 of Algorithm 2.

We initialize the target curve t to g, which is an overall "coarse" representation of the guide curve's path when input to the algorithm. We wish to match histograms of the exemplar curve e with coarser to finer representations of the input curve t in an iterative fashion, beginning with the coarsest scale of e. Setting a chosen number of scales $S = s$, we set the variable $step\_size$ to 2^s, keeping track of the samples to skip on the curve e. We begin by using this sample step to give us exemplar curve e^s and use the single-scale matching with the initial t. After each iteration of matching, we subdivide the output curve t into t' in line 7 to double its sample points. As well, we halve $step\_size$ to produce the finer scale version e^{s-1}. At this point the process is repeated with the matching of e^{s-1} and t'. This process continues until we have reached the finest scale of exemplar curve e (i.e. e^0 or e^1).

Algorithm 2 Histogram Matching at Multiple Scales

```
 1: procedure MULTISCALEMATCH($e, g$)
 2:     $num\_scales \leftarrow s$          ▷ Number of scales to consider
 3:     $step\_size \leftarrow 2^s$           ▷ Number of samples to skip
 4:     $t \leftarrow g$
 5:     for $i = 1$ to $num\_scales$ do
 6:         $e\_step \leftarrow everyKthSample(e, step\_size)$
 7:         $t \leftarrow HistMatch(e\_step, t)$
 8:         $t \leftarrow subdivideLine(t)$
 9:         $step\_size \leftarrow \frac{step\_size}{2}$
10:     end for
11:     return $t$
12: end procedure
```

6 Experiments

6.1 Technical details

In this section we evaluate the potential of our histogram matching algorithm with some experiments at one and multiple scales. For the single-scale experiments we use a simple "style" curve. For the multi-scale experiments we use a more stylish hand drawn curve shown on the left of Figure 6. We assess the accuracy of our algorithm by comparing the histogram of the output curve with that of the input curve. We also visually assess the meaningfulness of the resulting output curve. Our focus is mainly on smoothing and introducing fine-scale wiggles alongside simpler synthetic experiments to test some special interesting outcomes. In section 7 we explain the limitations of our implementation. We leave the details of a more sophisticated energy function as an interesting open research problem.

We divide our experiments into two sets: single-scale and multi-scale. In the single-scale we minimize the energy function until a sufficiently close histogram to the desired histogram is achieved while in the multi-scale we match histograms one scale at a time using the iterative approach outlined in 5.2. To this end, and by using carefully chosen parameter settings of w_1 and w_2, our algorithm is able to produce a curve with a histogram close to the desired histogram. In some experiments of the multi-scale approach, we are able to output a curve with a close histogram and with visual elements of the desired style serving as a feasible basis for evaluation. Introducing wiggles is more challenging, which we were able to do in two of our experiments.

Beginning with a small setting for the main parameters w_1 and w_2 and a suitable number of bins for the desired histogram, we run the algorithm repeatedly with increasing w_1 and w_2 values until histograms are suitably matched at which point we assess the visual effect of style transfer of the output curve. The parameters w_1 and w_2 are increased in unison where w_1 is set to a low starting value and held for a number of iterations while the correction parameter w_2 is progressively increased to find a good balance between it and w_1. If no decent histogram matching is performed or if the output curve does not behave as expected, w_1 is increased and the process is repeated. If the output curve behaves “wildly” (i.e. the result is a scribble) then likely w_1 has been set to too large a value. To achieve decent results, it is important to choose these parameters sensibly to find the right balance of the initialization parameter w_1 and the correction parameter w_2. Otherwise the penalty function will be over-corrected resulting in rubbish or under-corrected resulting in slow to no results.

The histograms are computed using a modified version of the *hist* function in MATLAB (*histnorm*) that ensures histograms are normalized with area summing to one. This way histograms can be compared on an equal footing. It is important that both histograms use the same bin width and range so that the bins line up and can be compared for the construction of the penalty function. The desired histogram is computed first by creating a row vector x_{hist} of n (the number of bins chosen using the Freedman-Diaconis rule) linearly spaced points that is bounded at each extreme by the floor (ceiling) of the smallest (largest) value rounded to a number reasonable for that range. Then, using x_{hist} and the curvature data of t as input, *histnorm* returns the target histogram. When computing H_t, if any data values fall beyond the range of x_{hist}, *histnorm* bins them at the very last bin which is not an issue given the following workaround: any values κ_i falling outside of this range have their penalty assigned as

$$\max(\rho(endpoint), \text{-}w_1 ln(\epsilon_{hist})) \tag{8}$$

which will be a high number as a result of the binning procedure. Since in these cases H_t's data range lies outside of H_d, the frequency of data values of H_t at the end points of x_{hist} is greater than H_d and data values in that bin are already penalized. We assign out-of-bounds values the penalty that is maximal between these two numbers to deter them in the output curve.

Histograms can be smoothed out by adding to the frequency of a bar a fraction of the distance between it and the average of its neighbours' frequencies. This has the effect of smoothing out the histogram without affecting the overall area and can be used in order to deal with 0 values posing an issue for the quotient in equation (6). However, for the purposes of our experiments, a small ϵ_{hist} value was added to both histograms to mitigate this issue.

We use a combination of hand drawn (Figure 6) or computer generated curves (Figures 8 and 9). The curves are either open (e.g. a curve as in Figure 6), or closed (i.e. a curve with equal starting and ending points), the latter being dealt with by using an input option to our algorithm making the small trivial modification. The hand drawn paths were scanned and manually traced using the Pen Tool in Adobe Illustrator and the resulting curves saved using the SVG

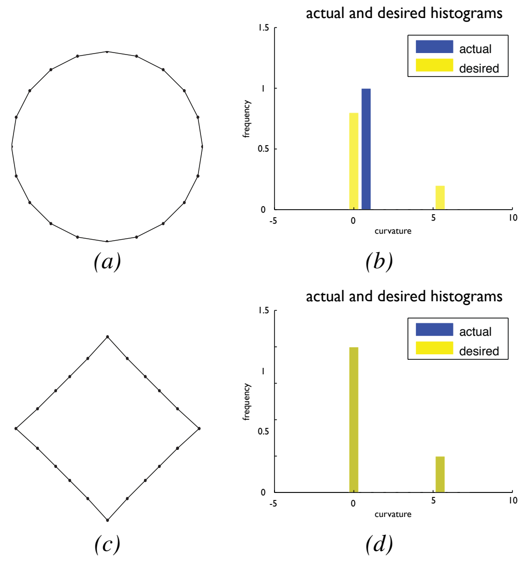


Figure 8: *(a) Input circle. (b) Desired histogram in yellow superimposed with histogram of circle in blue. (c) Output curve (square-like) after histogram matching. (d) Desired histogram in yellow superimposed with histogram of target curve; both same in this case.*

image format. A custom interface for extracting control point information from each separate curve in the SVG file was written in JavaScript allowing for the user to select which curve information to extract. Using this tool, the control points are output to a text file used to define the Bézier curves which are then rendered and sampled in MATLAB for processing.

6.2 Single-scale results

6.2.1 Smoothing a noisy curve

The simplest test to perform is to smooth a curve by using as target the histogram of a smooth curve. To begin with, we use as input a curve that has been altered by adding random noise giving it variation in curvature making for a non-smooth curve. The intent behind smoothing is to perturb the points such that the output curve has relatively close path to the input curve with the majority of curvatures close to 0.

In the top row of Figure 7 we show the input noisy curve along with its histogram, in blue. It is contrasted with the histogram of the desired style in yellow to show their initial disparity. The output of the algorithm, shown in the bottom row, displays the output curve along with its histogram, contrasted with the desired histogram. We see a smooth curve following the general path of the input curve with most of the noise dissolved, as expected. The minimization of the energy function found a new solution able to push inward the higher curvature frequencies towards 0 and matching the frequencies of curvatures. This experiment was run with several parameter settings giving good results for $0.025 \leq w_1 \leq 1.6635$ and $0.0025 \leq w_2 \leq 0.1875$ and with $n = 10$ bins. We found that smoothing was not as sensitive to parameter settings as in other experiments.

6.2.2 Square as desired

This is a synthetic experiment carried out to test the histogram matching performance of our algorithm. We used as input a unit

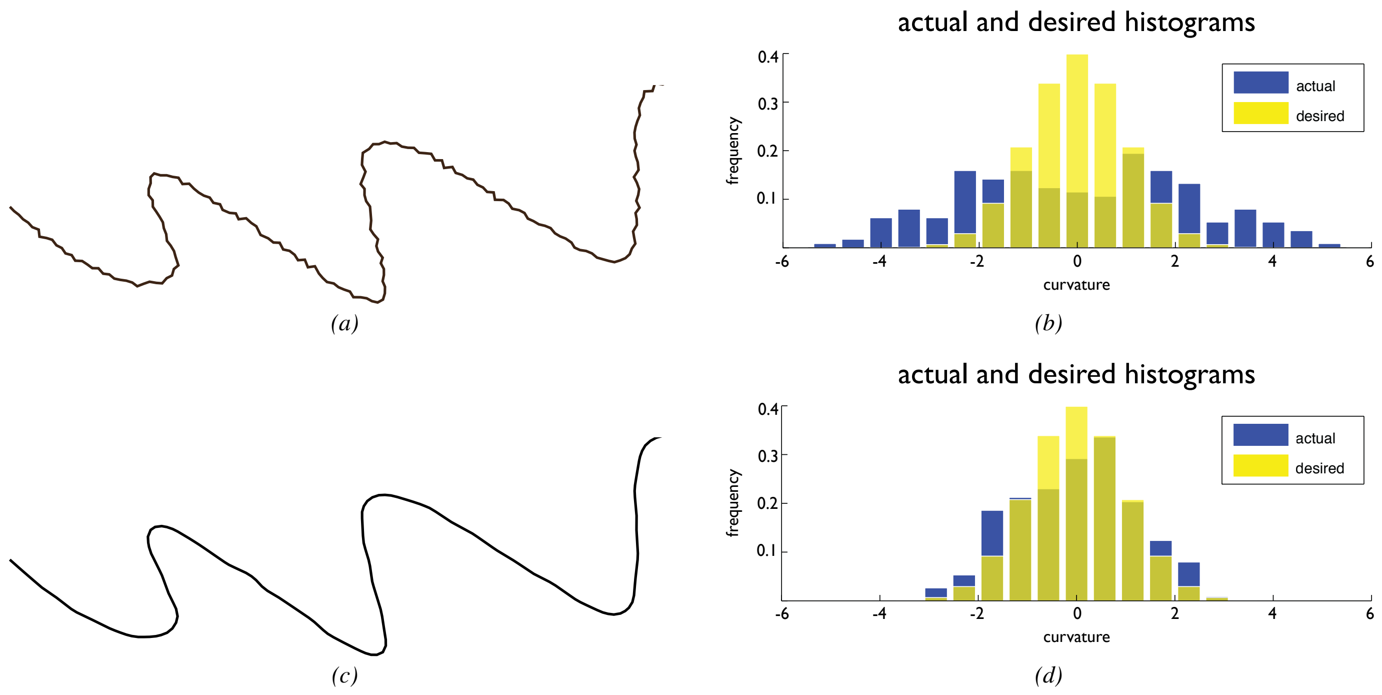


Figure 7: *Input noisy curve with desired distribution of a smooth curve: (a) Input noisy curve. (b) Desired histogram in yellow superimposed with histogram of the input noisy curve in blue. (c) Output smooth curve after histogram matching. (d) Desired histogram in yellow superimposed with histogram of target curve, both nearly the same.*

circle with 20 sample points and the histogram of a square (i.e. the desired curve), with the same number of points (though this was done only for simplicity). In Figure 8, the matching results are successfully shown. The top row shows the input circle along with the disparate histograms while the bottom shows the resulting output curve and the matched histograms. Using $n = 10$ bins for the histogram and parameter settings $w_1 = 0.05972$, $w_2 = 0.025678$, the algorithm is able to perturb the point on the circle into a new curve with matching histograms. We notice that in this particular test case, the output curve with matching histogram has been somewhat morphed into the square. In more sophisticated experiments, we seek a solution curve that isn't morphed into the input curve as this would be a trivial solution. However, this experiment is included to show the success in matching histograms as well as to contrast it with the next experiment.

6.2.3 Circle as desired

Here we perform a similar experiment with the desired distribution as that of a circle's. The input curve used is a square, hence we are performing the inverse of the previous experiment. We use the same number of sample points as in the previous experiment, namely 20. The results in Figure 9 show the successful matching of the histograms. The top row shows the input square with the disparate histograms while the bottom row shows the output curve and the matched histograms. The parameter settings are $n = 5$, $w_1 = 0.025$ and $w_2 = 0.0025$. In this particular test case, the output curve with matching histogram is not simply a morphing into the input curve but rather another shape. It is more inline with what we wish for: a curve close to the original (the square) with matching histograms.

6.3 Multi-scale results

In the following two experiments, we show results using our multi-scale approach with a hand drawn curve possessing fine-scale wiggles where we are able to transfer elements of the style.

6.3.1 Introducing fine-scale wiggles to a smooth curve

In this experiment, we wish to introduce fine-scale wiggles to a relatively smooth curve. We do this by inputing the guide curve g at the right of Figure 10 along with the hand drawn exemplar curve e with the desired style on the left. We match at three scales beginning with a sampling of 25 points on the input guide curve. We experimented with $n = 5$ bins, $w_1 = 0.025$ and $w_2 = 0.0045$. We optimized with a cap of 5 outer loop iterations using the multi-scale approach described in section 5.2. We evaluated the experiment based on the result of the histogram matching and visually to find transferred elements of the desired style

6.3.2 Input circle with desired hand drawn curve

In this experiment we evaluate the effect of using the same exemplar curve as in 6.3.1 along with a unit circle as the input guide curve. We hope to achieve similar results as in 6.3.1, introducing fine scale wiggles to the circle. The output curve t in Figure 11 shows the result of running our algorithm on four scales using $n = 10$ bins, $w_1 = 0.075$ and $w_2 = 0.0085$. We achieved these results with a cap of 30 outer loop iterations.

7 Conclusions and Future Work

Our algorithm is able to perform histogram matching with the right parameter settings for several cases of our experiments. It serves

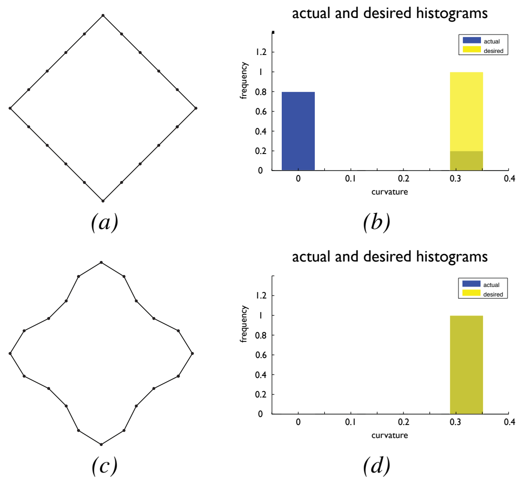


Figure 9: *(a) Input square. (b) Desired histogram in yellow superimposed with histogram of the input square in blue. (c) Output curve after histogram matching. (d) Desired histogram in yellow superimposed with histogram of target curve; both same in this case.*

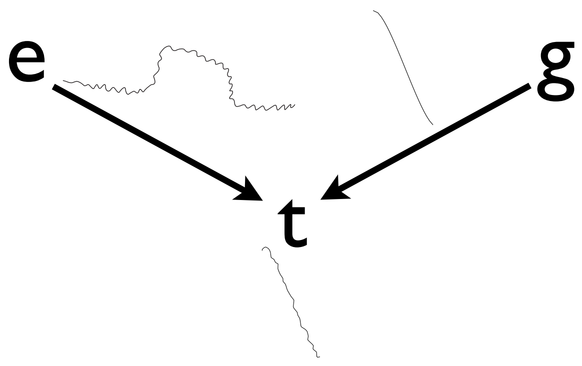


Figure 10: *Input smooth guide curve g with input exemplar curve e with desired style. Output of algorithm is target curve t possessing visual elements of the style.*

as a proof of concept that our approach at minimizing a distance between histograms can be done by using the iterative re-weighing scheme. Though we were able to produce some interesting results with some transfer of style, we ran into difficulties by using discrete polyline approximations of curves and having to resort to curvature approximations.

A drawback of this measure is its sensitivity to the sampling density of the curve, which varies as we optimize. Furthermore while optimizing the points of the polyline, curve information is lost and cannot be recovered in order to resample evenly. Another drawback of this approach is that when moving in the gradient directions, perturbing one point in one direction and computing the discrete curvature at this new point, it does not produce an accurate description of how a real curve behaves when a local part is modified. The movement of one point changes slightly the sampling density on the curve and what began as a uniformly sampled curve is no longer the case. This leads to sharp corners being formed while optimizing as a result of moving points on a polyline and does not guarantee for them to move in unison. Ideally we seek to perturb various points in unison so that the area local to this perturbation experiences a smooth change.

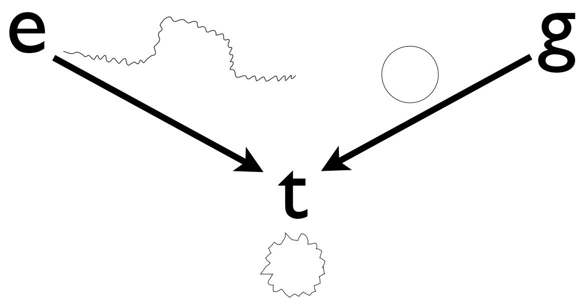


Figure 11: *Input circle g with input exemplar curve e with desired style. Output of algorithm is target curve t possessing visual elements of the style.*

The use of polylines and thus the curvature estimator or turning angles leads to another issue in setting the number of bins in the histogram. Ideally we would like to have a large number of samples that will give a good underlying representation of the curvature distribution. Due to sensitivity in sampling, histograms are not always accurately representative of the distribution.

These drawbacks point us to using parametric curves, setting an energy function over the control points of the curves instead. Using the control points, the analytic curve can be used during optimization and resampled as necessary while using the exact curvature to compute histograms. This will remove doubts on the curvature measure and its dependence on sampling. The original control points are available when storing curves in the initial steps of the pipeline and can be used directly.

Using the analytic curvature we could sample as many points as required to generate a good distribution. Moreover, when optimizing over control points, the movement of one control point would affect the curve much more gracefully than the movement of one sample point on a polyline, creating a break in its shape. A control point has the potential to affect the locality of the curve in conjunction with other control points, to create shapely wiggles with the constraint of keeping control points between splines symmetric. This would ensure C^1 continuity at the join point. The energy could then be computed by sampling the curve defined by those control points. A natural extension to this would be to introduce sharp corners as in Figure 3 (b, c) by relaxing the C^1 continuity constraint and using a separate bin in the histogram for “corners” (i.e. infinite curvature).

Finally, our multiscale approach is iterative, matching histograms progressively from coarser to finer scales. Though this approach is able to produce some interesting results, it does not in any way guarantee that by the time the final scale is matched that previous matchings will survive the process. Ideally we would like the final output curve to have matching histograms at all scales with the exemplar curve. To achieve this, histograms would need to be matched at multiple scales simultaneously by perturbing control points, computing the energy of the curve and comparing its multi-scale histograms with the corresponding histograms of the exemplar curve. A pyramid of all scales would be precomputed and stored to be accessed during optimization. We would then seek to find a curve with similar histograms at all scales while remaining close to the guide curve.

References

BÉNARD, P., JINGWAN, L., COLE, F., FINKELSTEIN, A., AND THOLLOT, J. 2012. Active Strokes: Coherent Line Stylization for Animated 3D Models. In *NPAR 2012 - 10th International Symposium on Non-photorealistic Animation and Rendering*, ACM, Annecy, France, P. Asente and C. Grimm, Eds., 37–46. Paper session 8: Lines, strokes and textures in 3D.

BRUNN, M., SOUSA, M. C., AND SAMAVATI, F. F., 2004. Capturing and re-using artistic styles with reverse subdivision-based multiresolution methods.

CHO, T. S., ZITNICK, C. L., JOSHI, N., KANG, S. B., SZELISKI, R., AND FREEMAN, W. T. 2012. Image restoration by matching gradient distributions. *IEEE Trans. Pattern Anal. Mach. Intell. 34*, 4, 683–694.

FARIN, G. E., HEGE, H.-C., HOFFMAN, D., JOHNSON, C. R., AND POLTHIER, K. 2002. *Mathematics and Visualization*. Springer.

FINKELSTEIN, A., AND SALESIN, D. H. 1994. Multiresolution curves. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, SIGGRAPH '94, 261–268.

FREEMAN, W. T., TENENBAUM, J. B., AND PASZTOR, E. C. 2003. Learning style translation for the lines of a drawing. *ACM Trans. Graph. 22*, 1 (Jan.), 33–46.

HAHMANN, S., BONNEAU, G.-P., CARAMIAUX, B., AND CORNILLAC, M. 2007. Multiresolution morphing for planar curves. *Computing 79*, 2 (Apr.), 197–209.

HERTZMANN, A., OLIVER, N., CURLESS, B., AND SEITZ, S. M. 2002. Curve analogies. In *Proceedings of the 13th Eurographics workshop on Rendering*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, EGRW '02, 233–246.

HSU, S. C., LEE, I. H. H., AND WISEMAN, N. E. 1993. Skeletal strokes. In *Proceedings of the 6th annual ACM symposium on User interface software and technology*, ACM, New York, NY, USA, UIST '93, 197–206.

KALNINS, R. D., MARKOSIAN, L., MEIER, B. J., KOWALSKI, M. A., LEE, J. C., DAVIDSON, P. L., WEBB, M., HUGHES, J. F., AND FINKELSTEIN, A. 2002. WYSIWYG NPR: Drawing strokes directly on 3D models. *ACM Transactions on Graphics (Proc. SIGGRAPH) 21*, 3 (July), 755–762.

LU, J., YU, F., FINKELSTEIN, A., AND DIVERDI, S. 2012. HelpingHand: Example-based stroke stylization. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*.

MCCRAE, J., AND SINGH, K. 2009. Sketching piecewise clothoid curves. *Computers & Graphics 33*, 4, 452–461.

PELE, O., AND WERMAN, M. 2010. The quadratic-chi histogram distance family. In *Computer Vision–ECCV 2010*. Springer, 749–762.

PORTILLA, J., AND SIMONCELLI, E. P. 2000. A parametric texture model based on joint statistics of complex wavelet coefficients. *Int. J. Comput. Vision 40*, 1 (Oct.), 49–70.

SIMHON, S., AND DUDEK, G. 2004. Sketch interpretation and refinement using statistical models. In *Proceedings of the Fifteenth Eurographics conference on Rendering Techniques*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, EGSR'04, 23–32.

TENENBAUM, J. B., AND FREEMAN, W. T. 2000. Separating style and content with bilinear models. *Neural Comput. 12*, 6 (June), 1247–1283.