Image Simplification and Vectorization

Sven Olsen*
University of Victoria

Bruce Gooch†
University of Victoria

Abstract

We present an unsupervised system which takes digital photographs as input, and generates simplified, stylized vector data as output. The three component parts of our system are image-space stylization, edge tracing, and edge-based image reconstruction. The design of each of these components is specialized, relative to their state of the art equivalents, in order to improve their effectiveness when used in such a combined stylization / vectorization pipeline. We demonstrate that the vector data generated by our system is often both an effective visual simplification of the input photographs, and an effective simplification in the sense of memory efficiency, as judged relative to state of the art lossy image compression formats.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Graphics Utilities

Keywords: image stylistization, vectorization, image reconstruction

1 Introduction

In this paper, we approach image stylization, edge tracing, and edge-based image reconstruction with the assumption that the three tasks are synergistic. We describe an unsupervised system that takes digital photographs as input and uses them to create stylized vector art, resulting in a simplification of the source data in terms of bit encoding costs, as well as visual complexity.

The algorithms that comprise our system are modified relative to the current state of the art in order to take better advantage of the complementary nature of the component tasks. Our primary technical contributions are:

1) We show that the *edge modeling problem*, previously identified as one of the fundamental challenges facing edge-only image representations, has a relatively simple and robust solution, in the special case of images that have been stylized using aggressive smoothing followed by soft quantization. (See Section 3.4.)

2) We introduce a novel edge-based image reconstruction method, which differs from prior work in that anisotropic regularization is used in place of a varying width Gaussian blur. While previous vector formats have successfully used varying width blurring to model soft edges, we found that the technique leads to artifacts given the unusually large widths required by our traced vector data. Our regularization approach avoids these artifacts, while maintaining a high degree of reconstruction accuracy. It also avoids the expense of calculating a wide varying width blur—considerably reducing the cost of luminance reconstruction. (See Section 6.1 and Figure 3.)

*e-mail:sven2718@gmail.com
†e-mail:brucegooch@gmail.com

3) We demonstrate that the vector data generated by our system is, in the sense of memory efficiency, significantly simpler than the input photographs. Specifically, we compare our vector output with state of the art lossy image compression results. While our vector encodings are in no sense accurate reproductions of the input photographs, they do maintain a sharp, stylized look, while preserving most visually important elements. The results of general purpose compression codecs suffer from significant visual artifacts at similar file sizes. (See Sections 5 and 7.1.)

2 Background

The idea that image stylization, simplification, and edge tracing should be approached as complementary tasks has a long history in both computer graphics and computer vision.

In their seminal work on the theory of image segmentation, Mumford and Shah [1989] cited the ability of artists to capture most important image information in simple cartoon drawings as evidence that it should be possible to create image segmentations that contain most of the semantic content present in natural images. In his contemporaneous work, Leclerc [1989] went further, and hypothesized that the most efficient possible representation of natural images would be as the sum of a piecewise smooth image and noise data.

Elder [1999] similarly hypothesized that a sparse, edge-only image representation could be used to store all the visually important content of most natural images. Elder developed an image format that contained only edge locations and edge gradient samples, and demonstrated that it was possible to reconstruct high quality grayscale images from that data. While Elder's format did not show competitive memory efficiency when compared with more conventional lossy image encodings, the similar but more recent edge-only format proposed by Mainberger et al. [2010] has been able to outperform conventional lossy encodings in the case where the inputs are limited to cartoon-like images.

An interesting variation on Elder's edge only format was developed by Orzan et al. [2008], who introduced *diffusion curves*. The primary purpose of the diffusion curve format is to enable artists to more easily construct soft-shaded vector art. The underlying edge data is thus parameterized by splines, in contrast to Elder's use of point samples. The image reconstruction method remains closely related to Elder's, although modifications have been made to support the presence of color. Diffusion curve vector data can also be generated automatically from source photographs, though the consequences of this process for either memory efficiency or visual fidelity relative to the source photograph have not been studied.

DeCarlo and Santella [2002] described a system for converting input photographs to cartoon-like images. The goal of this system was to simultaneously simplify and clarify the contents of an image. It operated by combining eye tracking data with mean shift segmentations and a b-spline wavelet analysis of edge lines. The resulting images were qualitatively simpler than the source data, but also appealing when considered as works of digital art.

Lecot and Lévy [2006] developed Ardeco, a combined image stylization / vectorization system. Ardeco operates by combining a Mumford-Shah energy minimization with a sequence of increas-

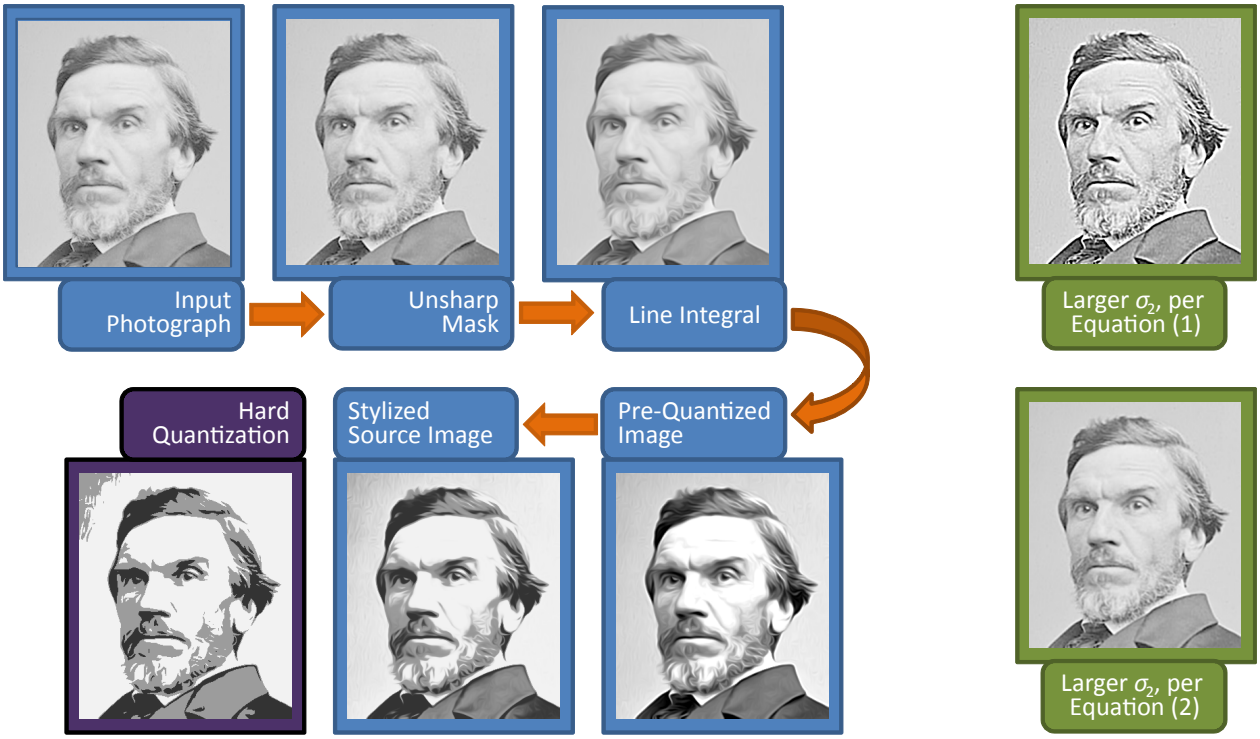


Figure 1: *Overview of image space simplification. The green boxes on the right show the degree to which the variance-based reparameterization given in equation* (2) *removes the influence of* σ_2*. The unsharp mask result shown in blue is calculated with* $\sigma_2 = 1.1\sigma_1$*, while both alternate results are shown with* $\sigma_2 = 1.6\sigma_1$*.*

ingly simplified triangle meshes, which are converted to spline boundary curves at the end of the process. In an approach similar to Elder's edge image reconstruction, adaptive blurring is used to model soft edges between regions. A study of the memory efficiency of Ardeco's vector output showed that the system could, under some circumstances, outperform JPEG encoding, but the compression results were less competitive relative to the more modern JPEG2000 standard [Stoiber 2007].

Gradient meshes are a vector format capable of representing smooth shaded images. A gradient mesh uses a collection of spline patches to parameterize smoothly varying colors over an image. Gradient meshes are supported by vector editing programs such as Adobe Illustrator, but have traditionally required a large amount of user guidance to create. However, in 2007, Sun et al. showed that the task of finding an optimal gradient mesh representation of an input image could be productively approached using a nonlinear least squares solver [Sun et al. 2007]. Subsequent work by Xia et al. demonstrated that arbitrary input images could be represented by relatively simple gradient meshes at a very high level of accuracy [Xia et al. 2009]. The primary motivation of gradient mesh generation algorithms has been to simplify graphic design tasks [Price and Barrett 2006]; however, Sun et al. [2007] were able to show that for simple images, their optimized gradient mesh results led to more compact files than JPEG compression.

The image-space stylization filter which we present here benefits from the many recent computer graphics papers that have advanced the art and science of image space stylization. In particular, we make use of the ability of difference of Gaussians filtering to effectively simplify and abstract facial features, something first noted by Gooch et al. [2004]. Variations on difference of Gaussian filtering that allow a wider range of artistic effects and higher quality results have since been developed by Winnemöller et al. [2006], Kang et al. [2007], and Kyprianidis and Döllner [2008]. The flow-guided filters introduced by Kang et al., in particular, have proven very useful in creating high quality stylizations for use as input to our vector tracing and reconstruction algorithms.

3 Image Space Simplification and Stylization

The image space simplification is composed of three steps. First, the photograph is converted to grayscale, and a combination of blurring and unsharp masking is used to remove details and exaggerate edges. Next, an edge orientation field is generated, and used to guide a line integral convolution, thus simplifying object boundary lines. The operation finishes by applying a non-uniform soft quantization filter, setting most of pixels in the image to one of three main tones. Figure 1 demonstrates the effects of each step.

3.1 Blurring and Unsharp Masking

Let I_σ denote the Gaussian blur of image I using a kernel of standard deviation σ. Blurring the input image using σ_1, then performing an unsharp mask of strength p using a second blur result yields an image I_m, which will have reduced details but stronger edges. If σ_2 is defined to be the sum of the widths of the unsharp mask blur and σ_1, then result of these first two steps can be expressed as follows,

$$I_m := I_{\sigma_1} + p(I_{\sigma_1} - I_{\sigma_2}), \qquad \text{where } \sigma_2 > \sigma_1. \tag{1}$$

The closer σ_2 is to σ_1, the larger p must be to create a noticeable

edge enhancement effect. This makes experimenting with different parameter values tedious, as small changes to either σ value can dramatically change the effect of different p values. In practice, it is helpful to reparametrize equation (1) in terms of the variance of the Difference of Gaussians image $E := I_{\sigma_1} - I_{\sigma_2}$.

$$I_m := I_{\sigma_1} + p\sqrt{\frac{Var(I_{\sigma_1})}{Var(E)}}E. \tag{2}$$

As shown in Figure 1, under this parameterization the choice of σ_2 proves to have very little impact on the resulting image I_m. Thus most features of the image I_m can be controlled by adjusting two relatively intuitive parameters—the base blur strength σ_1, and the unsharp mask strength p. All images shown in this paper are generated using $\sigma_2 := 1.1\sigma_1$, while p is typically set at .16, and σ_1 is typically set to 0.2 percent of the input image width.

3.2 Line Integral Convolution

The second step simplifies object boundaries and eliminates most of the noise introduced by unsharp masking. This is achieved by using a smoothed edge orientation field to guide a line integral convolution of I_m. The edge orientation field is calculated using a structure tensor constructed from blurred Sobel gradient terms, as in Kyprianidis and Döllner [2008]. We typically use a structure tensor blurring parameter, σ_f, equal to 0.64 percent of the image width.

The edge orientation field is then used to guide line integral convolution, with a Gaussian kernel of standard deviation σ_c, using the method of Cabral et al. [1993]. As noted by Kang et al. [2007], Cabral's method requires a slight modification in the case of edge orientation fields, as at each sample point there are two displacements consistent with the edge orientation. This ambiguity is best resolved by choosing the displacement that minimizes the bending of the line integral's center line.

Our system defaults to a line integral convolution strength σ_c equal to 1.6 percent of the image width, but, this can lead to over blurring in more detailed images. In such cases we frequently use a σ_c value of 0.3.

3.3 Linear Transformation and Soft Quantization

The image is next modified in such a way that shadows and highlights will be exaggerated. We achieve this effect by applying a piecewise linear transformation chosen to map the values $(.45, .75, .85)$ to $(.2, .61, .95)$. We refer to the result of this transformation step as the *pre-quantized* image.

As a final step of the stylization, we apply a non-uniform soft-quantization function. The function we use is similar to the soft quantization operator in Winnemöller et al. [2006], but it allows arbitrary bin centers, and lacks the discontinuities present in the original operation.

Given an ordered list of characteristic values, $\mathbf{b} = (b_1, ..., b_n)$, the *hard quantization* of a value v is given by,

$$q(v, \mathbf{b}) := b_i \quad \text{where } b_i \text{ is the characteristic value closest to } v.$$

To create an analogous soft quantization function, we split the interval $[b_1, b_n]$ into $n-1$ regions, each of which will map to a different sigmoid curve. For $v \in [b_1, b_n]$, let b_i be the closest characteristic value to v. Now define the sigmoid curve index j as $j := i - [b_i \geq v] + [v = b_1]$ (here the square brackets are Iverson notation [Graham et al. 1994]). Finally, define the width of sigmoid j as $w_j := \frac{1}{2}(b_{j+1} - b_j)$, and its vertical shift as $c_j := \frac{1}{2}(b_{j+1} + b_j)$. Using these variables, we create the following non-uniform, continuous soft quantization function,

$$p(v, s) := w_j \frac{\mathrm{sig}(\frac{s}{w_j}(v - c_j))}{\mathrm{sig}(s)} + c_j. \tag{3}$$

Note that the the division by $\mathrm{sig}(s)$ ensures C_0 continuity. Also note that that the derivative at the border between two quantization regions is independent of the spacing between bins. Specifically, $p'(c_j, s) = \frac{s\,\mathrm{sig}'(0)}{\mathrm{sig}(s)}$. Thus the sharpness of the soft quantization is controlled exclusively by the sharpness parameter s; it is independent of the spacing of the characteristic values. For sigmoid functions with $\mathrm{sig}'(0) = 1$, equation (3) converges towards $p(v, s) = v$ as $s \to 0$. In other words, the maximally smooth soft quantization function is simply the identity transform.

Given arbitrary $v \in \mathbb{R}$, we clamp v to the interval $[b_1, b_n]$, before applying equation (3), this allows us to extend the domain of the soft quantization function to all $\mathbb{R}$.

In our stylization system, the non-uniform soft quantization function is applied using $\mathbf{b} = (.2, .61, .95)$, and $s = 2$.

We also choose to apply soft quantization using the exponential sigmoid $\mathrm{sig}(x)$, rather than more commonly seen sigmoidal curves such as $\tanh$ or erf.

$$\mathrm{sig}(x) := \begin{cases} 1 - e^{-x} & \text{if } x > 0, \\ e^{x} - 1 & \text{otherwise.} \end{cases} \tag{4}$$

The choice of sigmoid function impacts the edge model used when vectorizing and reconstructing the stylized image, and using the exponential sigmoid in the soft quantization step implies that the differential equations in Section 6.1 have simple solutions.

After applying soft quantization, our image stylization is complete. We thus refer to the soft quantization result as the *stylized source image*.

The boundaries between quantization regions will form the starting lines input to our tracing algorithms. As the image shadows and highlights are exaggerated by the linear transformation, and then quantized using bin values chosen to divide the image into extremes of light and dark regions, there is a sense in which our approach to combined stylization and vectorization follows the advice given by Edwards [1979] in her famously effective instructional book, *Drawing on the Right Side of the Brain*. Rather than attempting to trace the outlines of individual objects, we simply focus on tracing patterns of light and dark tones.

3.4 A Solution to the Edge Modeling Problem

When Elder [1999] initially proposed his edge-only image format, he identified *edge modeling* as a central challenge facing any edge-only format, and one that might limit the utility of edge only images in practice. To solve the edge modeling problem, we must be able to characterize how image data should behave on either side of an edge. For soft edges, simply sampling the gradient at the edge crossing is not sufficient, it is necessary to have some model that specifies how luminance will vary as the distance from the edge increases. Elder's solution to the edge modeling problem was to assume that all luminance variations across edges could be modeled by the error function $\mathrm{erf}(x)$. The edge model parameter k was then found by fitting this function to the source image data near the edge.

Elder's edge fitting method sampled only four pixel values, but, even so, it lead to good results in the context of his own edge-based

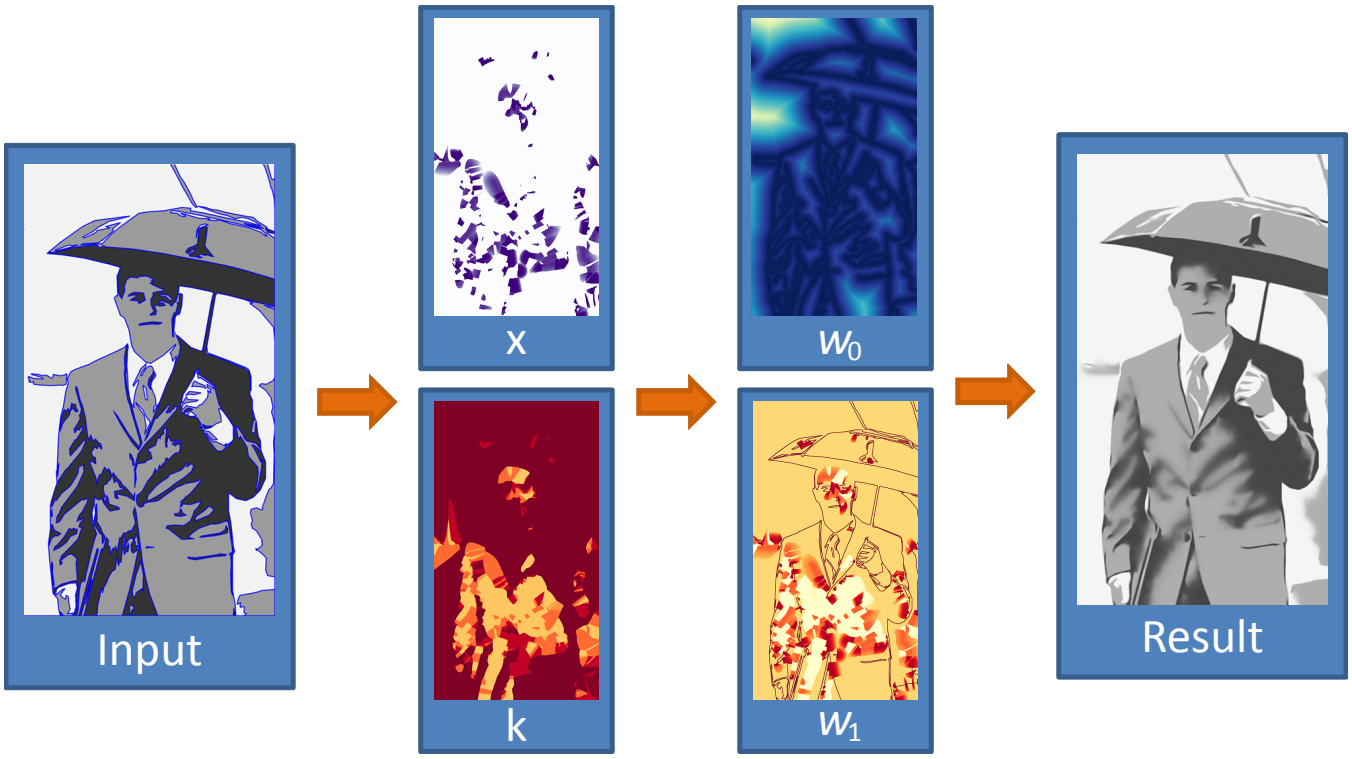


Figure 2: *The first image reconstruction step is to calculate a set of edge distance values x and sharpness values k based on the nearest edge points. With x and k thus defined, we can use the definitions in Section 6.1 to calculate data weights w_0 and edge weights w_1. In these visualizations, brighter colors correspond to larger values.*

system. However, performing a high accuracy fit of an edge model function to the very smooth edges that appear in our own stylizations would require much larger sets of pixel samples, making edge modeling both difficult and computationally expensive.

Yet, in the context of our own system, it is not necessary to perform such a function fit, as we have additional information that can be used to derive parameters for the model functions.

Quantizing the image implicitly partitions it into a set of regions having shared quantization values. Some of the boundaries between regions will occur in areas having low difference of Gaussian response, and thus, they will tend to have smoothly varying, locally linear luminance values. It is such smooth region boundaries that are responsible for most of the apparent "smooth shading" in our stylized images. We now show that, by assuming local linearity near region boundaries, it is straightforward to derive a single sharpness parameter, k, which models the behavior of the stylized image across such a region edge.

Let $I : \Omega \to [0, 1]$ be a grey scale image, having domain $\Omega \subset \mathbb{R}^2$. Let $S : \Omega \to \mathbb{R}^+$ be an image that defines the desired sharpness of the soft quantization at any point in the image. The soft quantization of I using sharpness image S is thus $p(I, S)$.

Consider the case in which the boundary between quantization regions occurs in an area where I is approximately linear. More specifically, let $\mathbf{x}_0$ be a point on the boundary between two adjacent regions, and Ω_0 be a subset of Ω for which,

$$I(\mathbf{x}) \approx I(\mathbf{x}_0) + \nabla I(\mathbf{x}_0) \cdot (\mathbf{x} - \mathbf{x}_0), \text{ and } \nabla I(\mathbf{x}) \approx \nabla I(\mathbf{x}_0).$$

Now consider a line $\mathbf{x}(\delta)$ normal to the boundary, where δ denotes the distance from the boundary point $\mathbf{x}_0$. By local linearity, the image value at a point $\mathbf{x}(\delta) \in \Omega_0$ can be approximated using,

$$I(\mathbf{x}) \approx I(\mathbf{x}_0) + \delta ||\nabla I||. \tag{5}$$

Substituting equation (5) into equation (3) yields the result that, for points inside the region Ω_0, the soft quantization $p(I, S)$ can be approximated by

$$p(\mathbf{x}, s) \approx w_j \frac{\text{sig}(\frac{s}{w_j}(\delta ||\nabla I(\mathbf{x})||))}{\text{sig}(s)} + c_j. \tag{6}$$

Where the sigmoid index j is chosen to correspond to the sigmoid centered at the boundary value, such that $c_j = I(x_0)$.

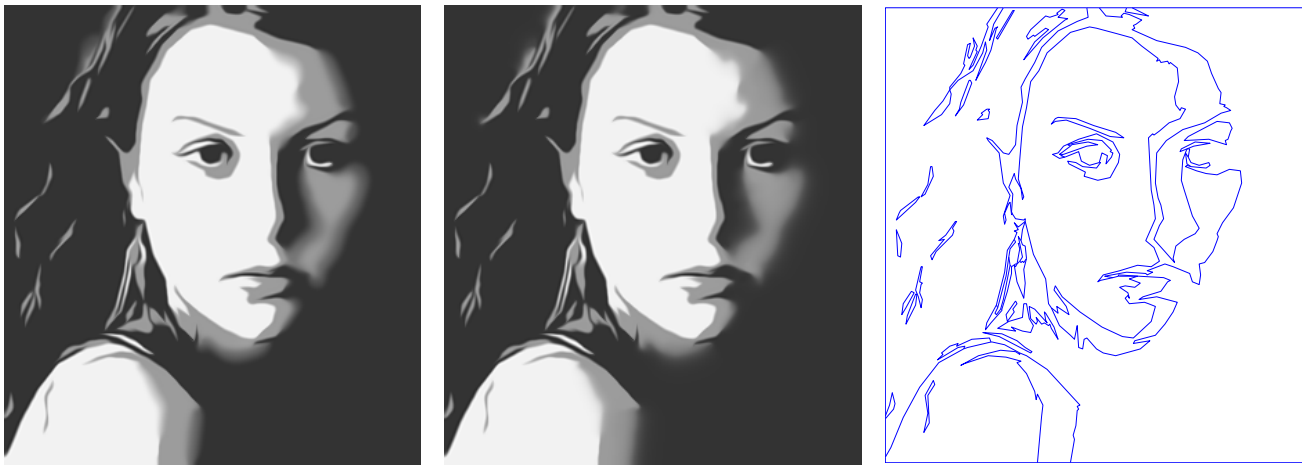
In general, if an image has been aggressively smoothed, then luminance variations throughout the image are likely to be both small and locally linear. However, if an image has been sharpened, then luminance variations near edges are likely to be large and discontinuous.

In areas with high edge response, we thus model edges as sharp step functions, represented in our regularization model as edges with infinitely large sharpness parameter, k. For low edge response cases, however, we use the edge model function $w_j \, \text{sig}(kx)$, with $k = \frac{s}{w_j} ||\nabla I(\mathbf{x})||$, which, as demonstrated by equation (6), will be a highly accurate model in the case of linearly varying image data.

4 Tracing

The algorithm we use to extract edge lines from the stylized image has the same general form as Potrace or the extraction method used by Diffusion Curves [Selinger 2003; Orzan et al. 2008]. What makes our tracing method unusual is that, even though we are constructing a representation of the stylized source image, we never reference the stylized image data directly. Instead, the information collected in the tracing step comes exclusively from intermediate images generated during the stylization process, specifically, from the *pre-quantized* and *hard-quantized* images (see Figure 1).

Source Photograph Licensed from iStockPhoto®



Reconstruction using Blurring | Reconstruction using Regularization | Edge Lines

Figure 3: *Comparison of varying width blurring and anisotropic regularization. Elder's method for smooth edge reconstruction operates by first solving a Laplace equation to generate a blur width image constrained by the sharpness values stored at edge boundaries. That data then guides a varying width Gaussian blur. The blur kernels cause banding artifacts near soft edges—notice that where the girl's cheek fades to shadow, the blur reconstruction contains two strong edge lines, though only a single edge exists in the vector data.*

Small Region Elimination We begin by interpreting the hard quantization result as an initial segmentation of the image. We perform a quick check to eliminate any small regions that may have resulted from the quantization. This check is similar to the small region elimination phase used in Selinger's Potrace vectorization system [Selinger 2003].

Generating Boundary Curves An initial set of boundary curves is found using marching squares. However, the boundaries found by marching squares take the form of closed loops around each region of the simplified quantization. We convert the closed loops returned by marching squares into a series of curve segments, such that every boundary point appears only once. We call any point on the boundary curve a *corner* if it lies between 3 or 4 different regions. We then convert our border curves into a set of curve segments, such that each segment connects two corner points. In the special case that a matched pair of closed loops contains no corner points, we convert them into a single closed boundary curve.

Boundary Curve Simplification The curves returned by marching squares take the form of a list of vertical and horizontal line segments, where each segment is exactly 1 pixel width long. We convert these segments into a more efficient format by applying a *straightness constraint*, similar to that used in Potrace's polygon extraction phase [Selinger 2003].

Edge Sharpness Sampling While performing the boundary simplification step, we sample the gradient magnitude of the underlying *pre-quantized* image points. Gradient sampling is performed using the results of a max filtered Sobel convolution, with the result that the largest gradient magnitude within r_g of the edge will be recorded. That sample is then used to imply an edge sharpness parameter k, as explained in Section 3.4.

5 Encoding

There are several parameters of the tracing process that can affect the memory efficiency of our vector data. Specifically, the small region elimination and line simplification parameters, and the gradient max filter width r_g.

In order to generate memory efficient vector data, we further simplify the traced data, as described below.

Sharpness Data First, we resample the edge sharpness samples k using sampling frequency f_k, where f_k is defined relative to the image space distance between two consecutive sharpness samples. Then, we define k_{max}, a maximum bound on the edge sharpness parameter k, and threshold all samples using this upper bound. Next, we lower the resolution of the k values, by limiting each k sample to one of n_k values, which are found by applying k-means clustering to the set of sharpness samples for which $k < k_{max}$. We default to $k_{max} = .6$, $f_k = 1.7$, $n_k = 5$. After these simplifications, edge samples of the form (x_i, k) can be eliminated, if both adjacent edge samples store the same k value, because such samples correspond to redundant knots in the linear spline that defines the sharpness values at any point.

After those simplifications have been applied, we store the k samples using Huffman coding. This requires storing frequency information for both each unique k value, and each unique sample position, x_i. To improve on the efficiency of this coding, we reexpress the position data x_i in terms of a sequence of offsets, $d_i = x_{i+1} - x_i$.

Luminance data Luminance information makes up a very small part of our vector format. We encode the luminance samples by simply recording a 2 bit id for each of the connected components implied by the edge data.

Edge Data After the edge model data simplifications have been applied, by far the largest portion of our vector data is the edge lines themselves. These are stored using Huffman encoding, after reexpressing each edge line in terms of its start point, and a sequence of delta values that define each successive point in the edge.

Source Image | JPEG2000 at 5.8kB | JPEG2000 at 5.8kB

DLI at 5.3kB | JPEG at 5.76kB | Vector at 5.8kB

Figure 4: *In the case of color photographs, we generate comparisons relative to the source both with and without the color channels. In this example, it appears that the luminance channel accounts for the majority of the information stored by the JPEG2000 encoding, as discarding the chrominance channels leads to only slight improvements in the compressed result. In this example, all compression results lose visual features that are preserved and sometimes even clarified by the vector stylization. Note in particular the shading of the students' coats, and the details of their faces.*

6 Reconstruction

Our boundary curve data implies a segmentation of the image pixels into connected components. The characteristic luminance values stored for each of these components can be used to construct a piecewise constant image. Our image reconstruction problem is thus to modify this piecewise constant image to create a piecewise smooth image that reflects, as accurately as possible, the edge sharpness values stored along the boundary lines.

We create such a piecewise smooth image by solving an anisotropic regularization problem [Weickert 1996]. Unlike the related image reconstruction methods of Elder and Orzan et al., we do not generate an intermediate blur width image, thus, the computational costs of our reconstruction should generally be less than the computational costs of image reconstruction in either of those systems, as our method requires only one linear system solve, while the varying blur method requires both a linear solve and a convolution with a large blur kernel. Also, as shown in Figure 3, given the vector data generated by our system, regularization appears to result in higher quality reconstructions of very smooth edges.

6.1 Regularization

Consider the very simple case in which our piecewise constant image can be described as a step function,

$$v(x) := \begin{cases} 1 & \text{if } x > 0, \\ -1 & \text{otherwise.} \end{cases}$$

Here we are assuming that, though the image is two dimensional, the value at any point is determined solely by the x-position.

There is only one edge boundary line in this image, which occurs at $x = 0$. Assume that the edge sharpness parameter at this edge is k, so the correct smooth image is defined by the exponential sigmoid curve, $\text{sig}(kx)$.

Now assume that our smooth image $f(x)$ must be reconstructed from $v(x)$ using an energy minimization process, in which $f(x)$ is defined as the minimum of the following regularization functional,

$$\min_f \int_{-\infty}^{\infty} w_1(x)|f'(x)|^2 + w_0(x)(f(x) - v(x))^2 dx. \quad (7)$$

Here $w_0(x)$ is a weighting function that determines how strongly $f(x)$ is attracted to $v(x)$, while $w_1(x)$ is a weighting that causes $f(x)$ to tend towards smooth functions.

Applying the Euler-Lagrange equations to equation (7) shows that $f(x)$ must satisfy the following ordinary differential equation [Weinstock 1974],

$$w_0(x)(f(x) - v(x)) - f'(x)w_1'(x) - w_1(x)f''(x) = 0. \quad (8)$$

We can reinterpret equation (8) as a constraint on the possible w functions by setting $f(x) = \text{sig}(kx)$, i.e., by forcing the regularization result $f(x)$ to be a perfect reconstruction of the correct image $\text{sig}(kx)$.

There are many weighting functions that satisfy equation (8), given $f(x) = \text{sig}(kx)$. And if we were only interested in reconstructing images that contained a single straight edge line, any of these possible w definitions would work equally well. However, while it is

Uncompressed Stylization | Vector at 2.1kB | Stylization at 2.1kB

Figure 5: *This is a case in which the vector stylization noticeably loses several visually important features that are present in the image-based stylization of the source, such as a portion of the outline of the left eye. But as the rightmost image shows, a JPEG2000 encoding of the stylized image with equivalent memory efficiency exhibits significant compression artifacts. At such extremely small file sizes, the loss of some visual information may be inevitable.*

important that the reconstructed image be correct in the single edge case, it is also important that our weighting function definitions lead to reasonably good reconstructions given much more complex edge sets. We have found that some weighting function definitions lead to better reconstructions of complex edge images than others, even if both functions produce perfect results in the single edge case.

We now provide the w definitions used in our own reconstruction algorithm. These definitions have proven to yield good reconstructions even given complex edge information.

We use a different set of weighting function definitions depending on whether or not the edge sharpness parameter k is less than an average sharpness value, given by K. When $k = K$, we use the simplest possible solution to equation (8),

$$w_1(x) = 1 \quad \text{and} \quad w_0(x) = K^2.$$

For points far away from the edge boundary we clamp the weighting functions to this simple solution, setting $w_1(x) = 1$ and $w_0(x) = K^2$ for all $|x| > x_1$. In order to avoid introducing discontinuities as a result of this clamping, we will parameterize additional solutions to equation (8) in terms of the free parameter ϵ, chosen such that,

$$w_0(x_1) = K^2,\ w_1(x_1) = 1,\ \text{for } x_1 \text{ such that } 1 - \text{sig}(kx_1) = \epsilon.$$

Reconstruction errors due to clamping become increasingly likely when k is either much greater or much smaller than K, thus it is beneficial to set K to be a value that represents an average level of edge sharpness. We use $K = 0.3$.

For the case of a low sharpness edge, in which $k \leq K$, we use the constant data weighting function $w_0(x) := K^2$ for all values of x. This reduces equation (8) to an ordinary differential equation in w_1, the solution to which is [Potter et al. 1987],

$$w_1(x) = \frac{K^2}{k^2} + e^{k|x|}\epsilon(1 - \frac{K^2}{k^2}).$$

The main drawback of these weighting function definitions is that $w_1(x)$ is negative for very large x. However, clamping w_1 to have a minimum value of 1 causes minimal reconstruction errors, provided that we use an ϵ value which is sufficiently small. We use $\epsilon = .05$.

A different pair of w definitions is needed to handle the high edge sharpness case, in which k is larger than K. We derive weighting functions for this case by enforcing the constraint that $w_0(x)K^2 = w_1(x)$. Solving equation (8) given this constraint yields,

$$w_1(x) = e^{(1-\frac{K^2}{k^2})|kx|}\epsilon^{1-\frac{K^2}{k^2}}, \quad w_0(x) = K^2 w_1(x).$$

Like the low sharpness solutions, these weighting functions have undesirable behaviors far from $x = 0$. In this case, the problem is that the exponential increase in the two functions leads to unreasonably large weighting terms. However, as in the low sharpness case, reconstruction errors are minimal given a small sufficiently small ϵ value. Note that, after clamping is applied, these equations imply that for very large k, $w_1(x)$ approaches a constant function having a point discontinuity, specifically,

$$w_1(x) \approx \begin{cases} \epsilon & \text{if } x = 0, \\ 1 & \text{otherwise.} \end{cases}$$

Thus, very sharp edges may be modeled using weighting functions defined as in the average $k = K$ case, but ignoring pixels in neighboring regions whenever calculating the gradient terms associated with the smoothness weight.

In the general case, our reconstructed image $f(\mathbf{x})$, defined over image domain $\Omega \subset \mathbb{R}^2$, is found by solving the 2D regularization problem,

$$\min_f \iint_\Omega w_1(\mathbf{x})||\nabla f(\mathbf{x})||^2 + w_0(\mathbf{x})(f(\mathbf{x}) - v(\mathbf{x}))^2 d\mathbf{x}. \quad (9)$$

The 2D weighting functions $w_1(\mathbf{x})$ and $w_0(\mathbf{x})$ are defined by choosing the weighting function definition implied by the edge sharpness value k of the closest edge point $\mathbf{x}_e$, and evaluating those functions with $x = ||\mathbf{x} - \mathbf{x}_e||$. The partial differential equation implied by equation (9) is then solved using finite difference methods. We solve the implied sparse linear system using successive over relaxation [Saad 2003]. As shown in Figure 2, the distance and sharpness data needed to define the weighting functions is complex, but, well known distance image and Voronoi tessellation algorithms can be used to generate this data efficiently [Rosenfeld and Pfaltz 1966].

6.2 Post Smoothing

The image reconstructions generated by the regularization solve tend to accurately reproduce any smooth shading in the source stylized image. However, hard edges often suffer from "jaggies", aliasing artifacts that result from the conversion of the boundary data to a discrete image grid. One simple way to eliminate such artifacts is to reconstruct images at a much higher resolution than they will be displayed, and then downsample the results, thus taking advantage of the resolution independent nature of the data. However, we have found it both more computationally efficient and more effective to apply a slight smoothing effect to the result. We use an anisotropic smoother for this purpose, specifically, we apply coherence enhancing anisotropic diffusion, as defined by Weickert [1996], using the coherence biasing parameter $\alpha = .001$, and scalespace parameter $t = 3.6$.

7 Results

Our system is implemented in a mixture of Matlab and C. The dominant cost of image stylization and tracing is the line integral convolution step, which can can require more than a second of processing time per image. Studies performed by Kyprianidis and Döllner, however, demonstrate that similar convolutions can be performed at much greater speeds using GPU processing [Kyprianidis and Döllner 2008].

Reconstructing images at megapixel resolution takes several seconds, the overwhelming majority of which is spent performing the linear solve. We use a combination of supersampling and post smoothing to reduce aliasing artifacts in our result images, therefore, all results are rendered at at least 1.5x the desired resolution. Again, the costs of image reconstruction could likely be dramatically reduced by using an iterative linear solver implemented in CUDA or OpenCL.

There are several cases in which our algorithms fail to create simple, curve-based representations of an initial image. If the stylization filter does not create a useful abstraction of the initial image, then our vectorization will also fail to be useful. Such failure cases often appear to be over-blurred or over simplified. Additionally, applying aggressive line simplification and region elimination settings can lead to shape distortions in the vectorized results. While errors of this last form can be fixed by using more conservative line simplification settings, doing so will increase curve complexity, leading to results which fail in our goal of creating a vector graphic that represents a simplification of the input photograph, in the sense of improved memory efficiency.

7.1 Memory Efficiency

Our goal is to create simple, stylized images in which the key visual content of a source image has been clarified, rather than discarded. Given an input photograph and the resulting vector image, the degree to which we are successful in this goal is difficult to quantify.

For example, consider a system in which, for any input photograph, the "vector stylization" returned is always an arrangement of two black boxes on a white background. From a memory efficiency standpoint, it is clear that a dramatic simplification has been achieved. However, such a system could not reasonably claim to "preserve and clarify" the key visual content of the photograph. But, while quantifying the degree to which the stylizations are "good abstractions" is difficult, and, perhaps necessarily subjective, identifying cases in which the vectorizations fail to improve memory efficiency is straightforward.

Source Image at 1.28kB | Vector at 1.28kB

Figure 6: *Simplification relative to a lossy compression of the input photograph, as generated by the Kadaku JPEG2000 encoder.*

7.1.1 Simplification Relative to the Input Photograph

If the vector data, when stored to disk, has a higher encoding cost than a high fidelity compression of the input photograph, then the claim that the vector stylization is an effective *simplification* is suspect. For example, even if the vector image appears visually simpler than the source photograph, if the initial photograph can be encoded with a high degree of visual fidelity using 7kB, then any vector result that requires more than 7kB of storage space has actually made the source data more complex than it was in its original form.

The resolution independent nature of our vector format makes direct comparisons of encoding efficiency impossible. As our vector data can be used to reconstruct smooth images at arbitrarily large sizes, storage efficiency per-pixel of output data is undefined.

Thus, for a collection of example vector stylizations, we have created compressed versions of the input photographs having similar filesizes. Cases in which the compressed images suffer from minimal visual distortions must be considered failure cases for our algorithm, in the sense that the generated vector data does not appear to represent a true simplification of the input photograph.

For the purposes of these tests, we have used the Kakadu JPEG2000 encoder to create lossy compression results at various file sizes [Taubman 2010]. The Kakadu encoder is arguably the most efficient JPEG2000 encoder available. It performed very well in the 2005 codec comparison studies performed at MSU [Vatolin et al. 2005]. More recently, it proved to be the best of the various JPEG2000 codecs tested by Lee in his DLI performance studies [Lee 2010].

We have also generated a few results relative to the older JPEG image compression standard. In these cases our vector images typically pass the data simplification test easily, as JPEG compression shows strong blocking artifacts at filesizes close to those of our vector encodings. The JPEG result shown in Figure 4 was generated using XnView's optimized Huffman table encoder.

Using JPEG2000 compression results in less extreme visual errors, but noticeable artifacts frequently still result when images are compressed into file sizes that match those achieved by our vector format.

Finally, in some of the Figures in this paper, we have included comparisons relative to the DLI research codec. As of 2009, DLI was the record holder for minimum MSE compression of image data at very small file sizes [Lee 2010]. But the DLI research codec does not allow files to be created as a specified bitrate, so generating comparison images matching specific file sizes is difficult.

Comparing the JPEG result shown in Figure 4 to the DLI or

Vector at 1.62kB Stylization at 1.62kB

Vector at 1.82kB Stylization at 1.82kB

Figure 7: *Vector results compared to JPEG2000 encodings of the source image space stylization.*

JPEG2000 encodings should make it clear how dramatically image compression technology has improved over the last twenty years. In comparison to the sophisticated techniques used by the Kadaku encoder, the vector encoding method presented here is quite immature, and there is likely significant room for improvement. Even so, the encoding costs for our simplified vector images are often less than 3kB per image. And at such extremely small file sizes, even state of the art codecs frequently result in noticeable artifacts; thus demonstrating that it is reasonable to consider our vector results effective simplifications of the input photographs. Example results are shown in Figures 4 and 6.

7.1.2 Simplification Relative to the Image Space Stylization

While the image space stylization step is designed to anticipate subsequent vectorization, the translation to a vector format will typically cause noticeable changes in the image content. Given our goals, such changes are not necessarily problematic. Our aim is to create effective abstractions of the input photograph, and depending on what sorts of changes exist between the image space stylization and the vector data, either one might be considered more successful in that aim. For example, in the case of the Ardeco system the "artifacts" of vectorization typically create attractive artistic effects [Lecot and Lévy 2006].

However, in many of the cases shown in this paper, the vector abstractions lose some important visual content relative to the source stylization. For example, the outline of an iris may disappear, or a small but important highlight may be eliminated. Figure 5 contains a good example of the iris simplification problem. In such cases, if conventional image compression can be used to store the *stylized image* at the same filesize as the vector data, we must conclude that the vectorization step of our pipeline is sacrificing visually important information unnecessarily.

Thus, we compare our vector renderings not only to lossy compressions of the source data, but, also, to lossy compressions of the image space stylizations from which the vector data is derived. As with the comparisons against compressions of the input photographs, the results of these studies often demonstrate that the vector encodings do occupy a level of memory efficiency at which the source stylization cannot be stored with good visual fidelity. Several examples that pass this test are shown in Figures 7 and 5, while Figure 8 shows a failure case.

Vector at 5.92kB Stylization at 5.92kB

Figure 8: *A case that fails our second evaluation test. Vectorizing the stylization discards many visual features, but, most of those visual features could have been faithfully preserved by simply creating a JPEG2000 encoding of the stylized source image.*

8 Discussion and Conclusions

The very high memory efficiency achieved by our vector format is interesting from an academic perspective, as it provides a case in which artistic abstraction is clearly linked to information efficiency. Such links have often been hypothesized [Leclerc 1989; McCloud 1994; DeCarlo and Santella 2002; Winnemöller et al. 2006]. But stylization systems that can be used as an alternative to lossy compression are relatively rare [Collomosse et al. 2005; Stoiber 2007; Sun et al. 2007; Qu et al. 2008]. And ours is arguably the first such system capable of showing clear benefits relative to JPEG 2000 encoding.

It is also interesting to note that the very low bitrate results created by cutting edge compression algorithms often contain artifacts not unlike the results of an artistic stylization. For example, compare the artifacts in the bottom left image of Figure 4 to the brush stroke stylizations created by Hertzmann [1998]. This further suggests that at extremely low bitrates the most effective compression methods may be those that approach increasingly stylized images.

The simplicity of our curve data suggests that our system could prove useful as a tool for artists. For example, one of the drawbacks of diffusion curves, as discussed in Orzan et al. [2008], is that the number and complexity of the edges returned by the tracing method makes the data difficult for artists to manipulate. For this reason, most of the diffusion curve results shown in that paper are generated from scratch by an artist, even in cases where the vector image is based on a reference photograph. The curves traced by our own joint stylization/vectorization framework are simpler and thus likely easier to manipulate; and as both our vector format and diffusion curves are variations on Elder's [1999] edge only format, converting our data to the diffusion curves format would be relatively straightforward.

References

CABRAL, B., AND LEEDOM, L. C. 1993. Imaging vector fields using line integral convolution. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 263–270.

COLLOMOSSE, J. P., ROWNTREE, D., AND HALL, P. M. 2005. Stroke surfaces: Temporally coherent artistic animations from video. *IEEE Transactions on Visualization and Computer Graphics 11*, 5, 540–549.

DECARLO, D., AND SANTELLA, A. 2002. Stylization and abstraction of photographs. In *Proceedings of SIGGRAPH '02*, 769–776.

EDWARDS, B. 1979. *Drawing on the right side of the brain*. J. P. Tarcher, Los Angeles, California, USA.

ELDER, J. H. 1999. Are edges incomplete? *Int. J. Comput. Vision 34*, 2-3, 97–122.

GOOCH, B., REINHARD, E., AND GOOCH, A. 2004. Human facial illustrations: Creation and psychophysical evaluation. *ACM Trans. Graph. 23*, 1, 27–44.

GRAHAM, R. L., KNUTH, D. E., AND PATASHNIK, O. 1994. *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

HERTZMANN, A. 1998. Painterly rendering with curved brush strokes of multiple sizes. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 453–460.

KANG, H., LEE, S., AND CHUI, C. K. 2007. Coherent line drawing. In *NPAR '07: Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, ACM, New York, NY, USA, 43–50.

KYPRIANIDIS, J. E., AND DÖLLNER, J. 2008. Image abstraction by structure adaptive filtering. In *Proc. EG UK Theory and Practice of Computer Graphics*, 51–58.

LECLERC, Y. G. 1989. Constructing simple stable descriptions for image partitioning. *International Journal of Computer Vision 3*, 1, 73–102.

LECOT, G., AND LÉVY, B. 2006. Ardeco: Automatic region detection and conversion. In *Eurographics Symposium on Rendering*.

LEE, D., 2010. DLI image compression . http://sites.google.com/site/dlimagecomp/software.

MAINBERGER, M., BRUHN, A., WEICKERT, J., AND FORCHHAMMER, S. 2010. Edge-based compression of cartoon-like images with homogeneous diffusion. *Pattern Recognition In Press, Corrected Proof*.

MCCLOUD, S. 1994. *Understanding Comics*. HarperCollins.

MUMFORD, D., AND SHAH, J. 1989. Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on Pure and Applied Mathematics 42*, 577–685.

ORZAN, A., BOUSSEAU, A., WINNEMÖLLER, H., BARLA, P., THOLLOT, J., AND SALESIN, D. 2008. Diffusion curves: a vector representation for smooth-shaded images. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, ACM, New York, NY, USA, 1–8.

POTTER, M. C., GOLDBERG, J. L., AND POTTER, M. C. 1987. *Mathematical methods / Merle C. Potter, Jack Goldberg*, 2nd ed. ed. Prentice-Hall, Englewood Cliffs, N.J. :.

PRICE, B., AND BARRETT, W. 2006. Object-based vectorization for interactive image editing. *Vis. Comput. 22*, 9, 661–670.

QU, Y., PANG, W.-M., WONG, T.-T., AND HENG, P.-A. 2008. Richness-preserving manga screening. *ACM Transactions on Graphics (SIGGRAPH Asia 2008 issue) 27*, 5 (December), 155:1–155:8.

ROSENFELD, A., AND PFALTZ, J. L. 1966. Sequential operations in digital picture processing. *J. ACM 13*, 4, 471–494.

SAAD, Y. 2003. *Iterative Methods for Sparse Linear Systems*, second ed. SIAM, Philadelphia.

SELINGER, P., 2003. Potrace: a polygon-based tracing algorithm. http://potrace.sourceforge.net/potrace.pdf, September.

STOIBER, N. 2007. *Compression of images and videos by geometrization*. Master's thesis, Technische Universitat Munchen.

SUN, J., LIANG, L., WEN, F., AND SHUM, H.-Y. 2007. Image vectorization using optimized gradient meshes. *ACM Trans. Graph. 26*, 3, 11.

TAUBMAN, D., 2010. Kakadu JPEG2000 Encoder. University of New South Wales. http://www.kakadusoftware.com/.

VATOLIN, D., MOSKVIN, A., PETROV, O., AND TITARENKO, A., 2005. JPEG 2000 image codecs comparison. Moscow State University. http://compression.ru/video/codec_comparison/jpeg2000_codecs_comparison_en.html.

WEICKERT, J. 1996. *Anisotropic diffusion in image processing*. PhD thesis, Dept. of Mathematics, University of Kaiserslautern.

WEINSTOCK, R. 1974. *Calculus of Variations: With Applications to Physics and Engineering*. Dover Pub. Inc., New York.

WINNEMÖLLER, H., OLSEN, S. C., AND GOOCH, B. 2006. Real-time video abstraction. *ACM Trans. Graph. 25*, 3, 1221–1226.

XIA, T., LIAO, B., AND YU, Y. 2009. Patch-based image vectorization with automatic curvilinear feature alignment. *ACM Trans. Graph. 28*, 5, 1–10.