Hierarchical Diffusion Curves for Accurate Automatic Image Vectorization

Guofu Xie*‡ Xin Sun† Xin Tong† Derek Nowrouzezahrai*
*University of Montreal †Microsoft Research ‡State Key Laboratory of Computer Science, ISCAS

Figure 1: *We are able to accurately vectorize raster images ("Input") by automatically tracing a sparse set of multi-scale diffusion curves to a broad class of images, including vector art (i.e., diffusion curve images), fluid simulations with turbulent flows, and natural images.*

Abstract

Diffusion curve primitives are a compact and powerful representation for vector images. While several vector image *authoring* tools leverage these representations, automatically and accurately vectorizing *arbitrary* raster images using diffusion curves remains a difficult problem. We automatically generate sparse diffusion curve vectorizations of raster images by fitting curves in the Laplacian domain. Our approach is fast, combines *Laplacian* and *bilaplacian* diffusion curve representations, and generates a hierarchical representation that accurately reconstructs both vector art and natural images. The key idea of our method is to trace curves in the Laplacian domain, which captures both sharp and smooth image features, across scales, more robustly than previous image- and gradient-domain fitting strategies. The sparse set of curves generated by our method accurately reconstructs images and often closely matches tediously hand-authored curve data. Also, our hierarchical curves are readily usable in all existing editing frameworks. We validate our method on a broad class of images, including natural images, synthesized images with turbulent multi-scale details, and traditional vector-art, as well as illustrating simple multi-scale abstraction and color editing results.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation;

Keywords: diffusion curves, vectorization, image reconstruction

Links: DL PDF

1 Introduction

Diffusion curve images (DCI) [Orzan et al. 2008], and their recent freeform vector graphics (FFG) extension [Finch et al. 2011], are an expressive tool for vector image authoring. Artists sketch curves and set color constraints, and a solver "diffuses" colors perpendicularly to the curves, respecting their boundaries, to create an image.

Image vectorization is a fundamental problem in content creation for computer graphics. As such, automatically extracting *feature-aware* diffusion curves (and their color constraints) from raster images is integral to an end-to-end vector authoring and editing pipeline that supports the DCI and FFG representations. This is challenging, in part, because many images break the fundamental assumption of current diffusion-based image representations: namely, that non-zero Laplacian and/or bilaplacian regions in images align along sparse, localized, and well-connected curves, while Laplacians/bilaplacians in other regions of images are zero.

Previous approaches use edge detection to place curves along sharp image features, relying on the assumption that regions with high Laplacian magnitude will overlap intensity discontinuities in the image domain. We show that, although this assumption is valid for sharp edges, it does not hold in general. Specifically, for blurred edges or bumps, the image edges do not overlap the Laplacian maxima (e.g, Fig. 3). We observe this behavior in both natural images and computer synthesized imagery resulting e.g. from a turbulent fluid simulation.

We formulate image vectorization as a Laplacian (and/or bilaplacian) reconstruction problem, and seek to fit *sparse* and *connected* curves to *directly approximate* the image Laplacian/bilaplacian. We avoid traditional edge detection in favor of extracting curves in the

ACM Reference Format
Xie, G., Sun, X., Tong, X., Nowrouzezahrai, D. 2014. Hierarchical Diffusion Curves for Accurate Automatic Image Vectorization. ACM Trans. Graph. 33, 6, Article 230 (November 2014), 11 pages.
DOI = 10.1145/2661229.2661275 http://doi.acm.org/10.1145/2661229.2661275.

DOI: http://doi.acm.org/10.1145/2661229.2661275

Laplacian and/or bilaplacian domains of an image. A trivial solution can easily be generated by scattering short curves with high density throughout the image Laplacian/bilaplacian; this solution is both computationally inefficient and generates an over-complete "curve soup" output that is not amenable to any after-the-fact editing. Instead, we fit curves and determine their color profiles hierarchically, where the coarse structure of an image is resolved using fewer longer curves. Fitting is *fully-automatic*, and the final curve output is organized in a progressive multi-scale level-of-detail scheme. Curves are associated with a scale index, allowing artists to edit the resulting vector image in a multi-resolution fashion.

Compared with previous diffusion curve vectorization methods, the output of our approach differs in the structure of the curves and the quality of the results it generates. Our curves align with important image features, resulting in an intuitive representation for artists. Our *automatically* generated curves are also strikingly similar in structure to manually-authored curves (see Fig. 8). To our knowledge, ours is the first fully-automatic curve fitting technique to combine Laplacian and bilaplacian diffusion curves.

2 Related Work

Vector representations have a long-standing history in graphics, and vectorization remains an important problem. For non-photographic images (i.e., vector art and cartoons) [Chang and Yan 1998; Zou and Yan 2001; Hilaire and Tombre 2006], where clear image boundaries are combined with smooth image gradients, vectorization using line and contour tracing approaches are typically sufficient. Vector representations for more complex images, such as imagery with turbulent details or digital photographs, typically demand a more expressive and flexible set of vector primitives to handle unstructured image regions and non-linear color variations. Recent diffusion curve representations, discussed in more detail below, combine flexibility in the types of image variations they can capture with compactness in their final form. We will focus primarily on recent work most relevant to our technique on mesh- and curve-based image vectorization primitives, where applications to vector art and photographic image content are considered.

Lecot and Levy [2006] decompose images into a set of regions delimited by cubic splines, each with their own precomputed color gradients. Swaminarayan and Prasad [2006] placed triangle primitives along edges in images and sampled the triangle colors from the original image, resulting in an artistic abstraction of the input. Demaret et al. [2006] combined an adaptive triangulation with a first-order spline in order to more accurately preserve input image details, specifically for image compression. Xia et al. [2009] generalized triangle-based approach with a patch-based image representation, allowing more flexible color variations. Liao et al. [2012] improved this method with piecewise smooth subdivision surfaces to avoid boundary discontinuities and support multiple levels of editing and abstraction. All these methods require a large number of primitives to accurately capture complex image structures which, in part, is why the earlier works favored stylized vector abstractions.

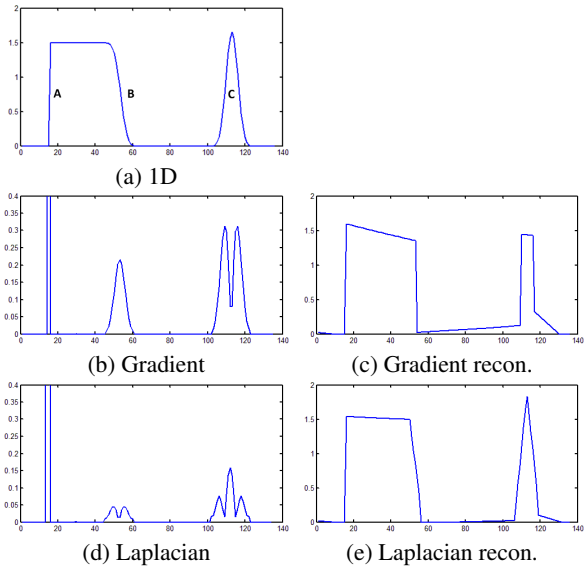
Moving from triangles and patches to connected mesh-like primitives, Sun et al. [2007] introduced the *gradient mesh* representation for vectorized images, which used a semi-automatic user-assisted procedure to fit the representation to raster images. Given sparse guide-strokes sketched by a user, their method generated the gradient mesh using an optimization procedure. Lai et al. [2009] proposed topology-preserving gradient meshes to represent arbitrary image regions, including those with binary alpha masks (i.e., holes), using a single mesh. While their output meshes are typically much more dense than the original gradient mesh results, the fitting strategy they propose is completely automatic. Our approach is also fully-automatic and can generate partially-connected (multi-scale) curves that tend to align with important image features. However, our method vectorizes the raster images as diffusion curves, which can capture non-linear image variation more compactly than gradient meshes.

Elder [1999] used edges as a near-complete and nature primitive to encode images with most color variations. Orzan et al. [2007] vectorized input image with a hierarchy of gradient-carrying edge structures. Later, Orzan et al. [2008] proposed *diffusion curves*, where colors along the curves are "diffused" outwards along the curve's normal direction in order to reconstruct an output image. They also proposed a method to automatically extract diffusion curves from an input bitmap by extracting edges in gradient space; once extracted, the curves' properties are determined by solving a constrained Poisson equation. Although this vectorization scheme works well for "vector-art-like" raster images, it is not capable of capturing all forms of color variation present in natural images. Jeschke et al. [2011] extended diffusion curves with more general color and texture functions, as well as leveraging an alternative formulation of the problem as a solution to 2D Laplace equations with Dirichlet boundary conditions. Finch et al. [2011] employed higher-order profile functions based on thin-plate splines with user-specifiable discontinuities. Sun et al. [2012] proposed Green's functions based representation for diffusion curves that support fast reconstruction. Ilbery et al. [2013] extend Green's functions to Finch et al.'s biharmonic curve formulation to broaden its flexibility and improve final image reconstruction performance. Sun et al. [2014] introduced a fast multipole representation for rendering diffusion curve images in real-time. Our method automatically extracts multi-scale diffusion curves in Laplacian domain, which is more robust and accurate for reconstructing a broad class of images, including vector-art/abstracted non-photographic images, natural images, and synthesized imagery with details across scales. We also use Green's functions based representation for diffusion curve fitting and reconstruction.

Many approaches perform edge detection in multi-scale. Perona and Malik [1990] used an anisotropic scale-space to preserve strong edges. Lindeberg [1998] proposed a mechanism for automatic selection of scale levels to precisely detect edges. While previous approaches apply edge detection at a Gaussian scale-space in the image or gradient domain in order to extract curves, we generate multi-scale image stack via structure-preserving multi-scale bilateral filters [Fattal et al. 2007] and then follow the aforementioned observation to extract curves in Laplacian and bilaplacian domain.

3 Overview

Problem Formulation. Given a raster color image $u[p]$ defined at discrete pixel locations p, our goal is to automatically extract a set of diffusion curves from $u[p]$ in order to accurately reconstruct a vectorized form of the image. A diffusion curve [Orzan et al. 2008] is defined using a curve primitive with colors set on both sides of it. These boundary conditions are equivalent to enforcing non-zero Laplacian values on the curve itself and zero Laplacian values elsewhere. We thus extract *Laplacian diffusion curves* directly from the Laplacian transform of the image $\Delta u[p]$, and we may extract additional *bilaplacian diffusion curves* using the bilaplacian of the image $\Delta^2 u[p]$ in order to capture higher-order variations in smooth images regions. To this end, we present a multi-scale curve extraction scheme to efficiently trace and chain curves together across image scales. Once extracted, we can reconstruct the final image by solving a partial differential equation (PDE) with boundary conditions defined by our curves. We will present a unified formulation (Eq. (2)), based on Green's functions [Sun et al. 2012; Ilbery et al. 2013], for reconstructing a continuous image signal $u(x)$ from both Laplacian and bilaplacian diffusion curves.



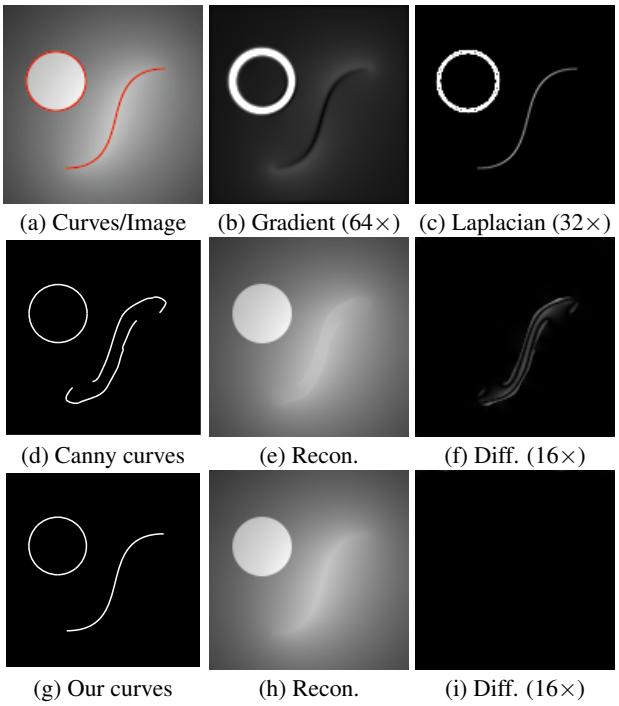
(a) 1D (b) Gradient (c) Gradient recon. (d) Laplacian (e) Laplacian recon.

Figure 2: *1D comparison of gradient and Laplacian fitting approaches.*

Key Observations and Motivation. We will motivate our choice for fitting curves in the Laplacian and bilaplacian domains, comparing against gradient domain alternatives using a simple but representative 1D example shown in Fig. 2 (a). Our fundamental observation is that many images not only include sharp edges (around A in Fig. 2 (a)), but also include blurred edges (around B in Fig. 2 (a)) and regions with variations (around C in Fig. 2 (a)). All of these image features can lead to large Laplacian magnitudes (A, B, and C in Fig. 2 (c)) where diffusion curves should be placed for accurate reconstruction.

Previous diffusion curve fitting approaches [Orzan et al. 2008] extract curves in the image domain using edge detection [Canny 1986], effectively fitting curve structures to local maxima in the gradient domain. As shown in Fig. 2 (b), for sharp edges, significant Laplacian magnitudes overlap large gradient magnitudes. Therefore, in this case, gradient domain methods accurately detect and place curves in meaningful regions. However, for blurred edges, maxima in the gradient domain do not overlap with regions of significant Laplacian magnitude. As a result, gradient domain methods end up placing curves far away from regions with significant Laplacian magnitudes. Without accurate curve placement in these regions, the resulting diffusion curve image reconstruction cannot properly capture these smooth image variations (shown in Fig. 2 (d)). Indeed, Orzan and colleagues note that accurately detecting regions of smooth image variation, or with very blurred edges, in the gradient domain is very difficult. Our Laplacian domain algorithm, on the other hand, directly detects these regions with significant Laplacian values, including those associated with both sharp edges and blurred edges, as well as the types of variations illustrated in Fig. 2 (c). With accurately placed curves, our method faithfully reconstructs the input image (see Fig. 2 (e)).

Fig. 3 illustrates another simple, representative example of the aforementioned limitations of gradient-domain curve fitting, this time in 2D. The source image Fig. 3 (a) contains both a sharp circular edge and a blurred edge defined along an "S"-shaped fall-off profile. While the circular edge is clearly identified by a gradient domain edge detector Fig. 3 (b), only the Laplacian transform is capable of identifying the large Laplacian magnitudes Fig. 3 (c) of both features. Our Laplacian domain approach successfully extracts these features and sets the appropriate diffusion curve profile function values in order to achieve a more accurate reconstruction Fig. 3 (h,i) than the results Fig. 3 (e,f) generated by the gradient



(a) Curves/Image (b) Gradient (64×) (c) Laplacian (32×) (d) Canny curves (e) Recon. (f) Diff. (16×) (g) Our curves (h) Recon. (i) Diff. (16×)

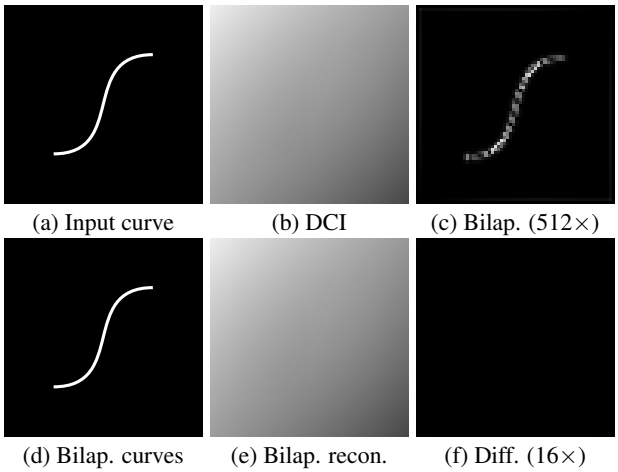
Figure 3: *Comparing curve extraction and reconstruction from gradient (Canny) and Laplacian (our) domains; (e) is reconstructed using curves (d) from our approach.*

domain approach.

While these two examples are (purposefully) simple, we will show that the same principles extend to general vector art and more complex natural images, including those with turbulent details (see e.g., Fig. 11 and Fig. 1 in the supplemental material). One noteworthy concern is that natural images may be stored in bitmap formats with compression artifacts, leading to artificial fine-scale image structures that would skew our Laplacian domain fits and result e.g. in the creation of many shorter superfluous curves; to address this issue, we apply an structure-preserving multi-scale bilateral filter [Fattal et al. 2007] to natural images in compressed formats in order to reduce the impact of these artifacts while preserving important small- and large-scale features. Moreover, our hierarchical curve fitting (Section 4) extracts and links curves across scales of the filtered images. An added benefit of such a multi-scale curve representation is that it allows us to perform simple editing operations, such as automatic vectorized image abstraction and scale-dependent color and structure editing.

Image Vectorization. We will outline a procedure to fit diffusion curves that are fully described by their curve-geometry and a compact set of endpoint weights, and our approach comprises two main steps. First, we determine the position and shape of the curves with hierarchical extraction in the *Laplacian domain* (Section 4). Next, for these extracted curves, we efficiently solve for the Laplacian diffusion endpoint weights using a Green's function formulation (Section 5).

Earlier, we motivated the use of the Laplacian of an image as a powerful image feature identifier, since regions of significant Laplacian magnitude correlate to local intensity variations, and the distribution of the Laplacian's local maxima almost fully describes important image regions such as hard edges and localized smooth color variation. Images with many fine-scale features, as well as smoother images, are often sparse in the Laplacian domain. In Section 4, we detail how to extract multi-scale curves that align to the



(a) Input curve (b) DCI (c) Bilap. (512×)

(d) Bilap. curves (e) Bilap. recon. (f) Diff. (16×)

Figure 4: *Curves extracted from the image Bilaplacian domain.*

spatially-coherent local maxima in the Laplacian domain. These regions correspond to important (and sometimes low-contrast) features in the image that are difficult to robustly identify in either the image or gradient domains (see Fig. 3).

We extract curves in a hierarchical fashion, according to the scale of image features, and our final curve representation is also explicitly hierarchical in scale-space. Our multi-scale representation is both amenable to image reconstruction with user-controllable error thresholds, as well as after-the-fact manual editing, like multi-scale abstraction. We extract long connected curves to reconstruct large-scale image features, and shorter curves to resolve smaller-scale features. Our hierarchical extraction allows us, among other things, to carefully control the distribution of large and short curves.

After extracting (multi-scale) curves, we must solve for the endpoint values that define the influence of the curves on the final (reconstructed) image. Section 5 will detail this process, highlighting three fundamental differences between our representation and previous DCI fitting schemes [Orzan et al. 2008]. First, we use a Green's function formulation to represent Laplacian diffusion curves, which has two terms (see Eq. (2)): a color and an additional normal derivative of color. The normal derivative term permits a additional powerful degree of expressiveness and allows us to better capture smooth variations without any post-processing (i.e., blurring). This flexibility is especially useful for smooth images. Second, we *progressively* fit all curves *up to a given hierarchy level* to the image based Green's functions, exploiting the hierarchical nature of our curves. Third and lastly, we can dynamically reconstruct variations of the image by editing the color term, without needing to refit the curve weights.

We extend our diffusion curve representation to support bilaplacian diffusion curves [Ilbery et al. 2013]. Bilaplacian (or biharmonic) diffusion curves have been shown to be a very powerful representation, capable of capturing sharp boundaries and higher-order smooth image variations, such as those caused by a smooth interpolation of color constraints along a curve. These types of image features often appear in natural images and turbulent phenomena such as fluid simulations. However, no existing approach can automatically extract bilaplacian diffusion curves from a raster color image, for the same reasons that automatic Laplacian curve extraction remains an open problem: since bilaplacian curves represent smooth image variation without pronounced boundaries, typical gradient-domain fitting strategies will always fail to correctly extract them. As such, we will extend our Laplacian curve fitting technique to bilaplacian curve extraction by similarly tracing local maxima in the image bilaplacian domain.

Although the representational capability of bilaplacian curves subsume that of Laplacian curves, we have found that Laplacian curves are often sufficient to represent many of the sharp boundaries and smooth image variations that we are interested in. From their Green's function formulation (Eq. (2)), we note that Laplacian curves require only two unknown terms to be resolved during weight computation, instead of the four terms needed for bilaplacian curves, and so we are able to more quickly compute the curve weights, and reconstruct the final image, with a Laplacian-only representation compared with a bilaplacian representation. And so, for sharp boundaries, we prefer Laplacian curves. After extracting curves, however, we use a voting scheme to optionally classify each curve as a Laplacian or bilaplacian curve, depending on the type and scale of the features we want to represent. Performing this two-step fitting and classification allows us to significantly improve the fitting and reconstruction performance of our approach, without any loss in image quality when compared with exclusive fitting (and reconstruction) of the more costly bilaplacian curves.

4 Hierarchical Curve Construction

We will introduce our multi-scale Laplacian diffusion curves fitting approach in this section before discussing its extension to bilaplacian diffusion curves in the next section. Prior to computing the image Laplacian, we first create a multi-scale detail-preserving filtered image stack to filter out features that will not be reconstructed by the curves of a specific scale, for each scale (Section 4.1). Curves are extracted, at each scale, by first identifying and then connecting candidate (Laplacian) pixels (Section 4.2). Finally, in order to preserve large-scale features, a coarse-to-fine pass ensures that the features captured at coarser scales are properly included at the finer scales (Section 4.3).

4.1 Structure-preserving Image Prefiltering

When extracting curves at a given scale, it is important to filter out all image features that will not be reconstructed by curves at that scale, eliminating their influence on curve placement. To do so, we apply a structure-preserving bilateral filter with a blur kernel set according to the scale. Specifically, we adapt a multi-scale bilateral decomposition [Fattal et al. 2007]: we blur the input image u into a stack of filtered images u^j, substituting the standard grayscale luminance distance with a CIE-Lab distance metric that preserves sharp color variation across iso-luminance contours in the image, and using a modified range filter better suited to our feature isolation. We do not fit separate curves per color channel. Our robust weight solving solution (Section 5) is powerful enough to capture all color variation using a single set of curves per image.

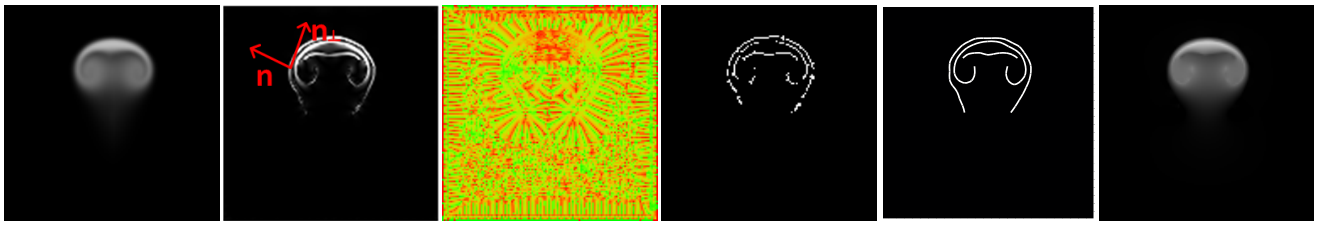
By defining the finest scale image $u^0 = u$, we can express the bilateral filtering operation iteratively as:

$$u^{j+1}[p] = {}^1\!/\!_k \sum_{q \in \Omega_j} g_{\sigma_{s,j}}(\|q\|)\ g_{\sigma_{r,j}}\Big(u^j[p+q] - u^j[p]\Big)\ u^j[p+q]$$

where $g_\sigma(x) = exp(-\|x\|^2/\sigma^2)$, $\|\cdot\|$ denotes Euclidean distance and color distance is computed in the CIE-Lab color space, $\sigma_{s,j}$ and $\sigma_{r,j}$ are the spatial and range widths of the filter, q iterates over surrounding pixels in the spatial filter's support Ω_j, and k is a normalization term:

$$k = \sum_{q \in \Omega_j} g_{\sigma_{s,j}}(\|q\|)\ g_{\sigma_{r,j}}\Big(u^j[p+q] - u^j[p]\Big)\ .$$

We double the spatial filter width $\sigma_{s,j}$ at each scale, so $\sigma_{s,j+1} = 2 \times \sigma_{s,j}$ and $\sigma_{s,j} = 2^j \times \sigma_{s,0}$, where we define the finest scale's spatial width as $\sigma_{s,0} = \sigma_s$. In the case of the range filter width, we *halve* its size across scales, and so we similarly arrive at $\sigma_{r,j} =$



(a) Input image (b) Laplacian (8×) (c) $\mathbf{n}_\perp$ direction (d) Curve pixels (e) Extracted curves (f) Our reconstruction

Figure 5: *Curve extraction (Section 4): we locate curve pixels (d) and trace curves (e) along the local maxima of the Laplacian (b).*

$\sigma_r/2^j$ and define the finest scale $\sigma_{r,0} = \sigma_r$. All the results in our paper use $\sigma_s = 2$, $\sigma_r = R/10$, where R is the intensity range of the image.

4.2 Curve Extraction in the Laplacian Domain

Once our filtered image stack is prepared, we proceed to extract curves in the image Laplacian domain Δu by first identifying pixels that should lie on curves, and then connecting them to form discretized curves (which we later vectorize).

Identifying candidate curve pixels in the Laplacian image Δu is a challenging problem for two reasons: first, the Laplacian is not sparse to begin with (unlike our final Laplacian curves, which serve to "sparsify" the Laplacian); secondly, the dynamic range of the Laplacian values can be very large, yet we are interested in placing curves in both low- and high-intensity Laplacian regions in order to capture large- and fine-scale details. Given these constraints, we choose to identify curve pixels according to the *local Laplacian contrast* and not the global Laplacian value. Specifically, we will extract (discrete) curve pixels by locating, and tracing along, the local maxima of the image Laplacian. We later convert these discrete "curves" into a continuous vector representation (Section 5).

We will extract curves, for each filtered image u^j, in its (filtered) image Laplacian Δu^j using a simple two-step approach. We begin by setting the absolute value of the Laplacian as the image Laplacian and identifying curve pixels along the local maxima in the image Laplacian using a *non-maximal suppression* detector [Canny 1986]. We first compute an estimate of curve directions locally at each image pixel by approximating the second partial derivatives $v_{x,x}$, $v_{y,y}$, and $v_{x,y}$ of the image Laplacian. To obtain these second-order approximations, we convolve the image with a discrete two dimensional second partial derivative kernel of a Gaussian. The estimated direction $\mathbf{n}$, in which the second directional derivative takes on its maximum absolute value, can be determined by calculating the eigenvalues and eigenvectors of the resulting Hessian matrix:

$$H[x,y] = \begin{pmatrix} v_{x,x} & v_{x,y} \\ v_{x,y} & v_{y,y} \end{pmatrix} . \tag{1}$$

Then, by applying the non-maximal suppression kernel to this direction field, we can compute the local maxima of the Laplacian magnitude. The resulting maxima form our *candidate curve pixels* set. Pixels from this set that satisfy the following two conditions are marked as curve points: first, the Laplacian value at the pixel should be the maxima in its neighborhood *along the direction* $\mathbf{n}$ (Fig. 5b); and, second, the value should be larger than a user defined threshold. The threshold is usually set as 0.01. If the Laplacian value is smaller than the threshold, it cannot be a candidate point to form curves.

Once we identify the curve pixels (Fig. 5d), we can proceed to link them into discrete curves along the direction perpendicular to $\mathbf{n}$, $\mathbf{n}_\perp$ (Fig. 5c).

We first select a random curve pixel p and compare its perpendicular direction $\mathbf{n}_\perp[p]$ with the perpendicular direction $\mathbf{n}_\perp[q]$ of the pixels q in its 1-ring (8-pixel) neighborhood Ω_p; of these neighbors, we select the pixel q^* as a connecting point (for the curve starting at p) if its direction aligns most with p's direction and if it is within a user-defined threshold ϵ_c (usually set $\sqrt{2}/2$):

$$\mathbf{n}_\perp[p] \cdot \mathbf{n}_\perp[q^*] > \mathbf{n}_\perp[p] \cdot \mathbf{n}_\perp[q],\ \forall q \in \Omega_p \text{ and } \mathbf{n}_\perp[p] \cdot \mathbf{n}_\perp[q^*] > \epsilon_c\ .$$

If no suitable connecting curve point q^* is identified, we increase the search region to a 2-ring (16-pixel) neighborhood Ω_{p^+} and repeat the procedure. If, when searching in Ω_{p^+} we find a valid connecting curve point q^*_+, we additionally mark the pixel (in Ω_p) between p and q^*_+ as a connecting curve point. Then, we find valid connecting curve points from the opposite perpendicular direction $-\mathbf{n}_\perp[p]$ to form a longer curve. Meanwhile, we remove curves whose length is less than 3 pixels as they are likely caused by residual noise in the image.

Merging Double Curves. In some cases, nearly identical curves can appear right next to each other (in a 1-pixel neighborhood). While retaining these "double curves" does not affect the final image quality, they remain redundant and incur an unnecessary reconstruction cost. As such, we detect and merge double curves into a single curve as follows: after extracting all the curves, we sort them according to the length of curves and, starting with the shortest curves, trace along each curve and test to see if another curve lies within a 1-pixel distance along the curve's $\mathbf{n}$ direction. As we step along the curve, we flag the neighboring curve index until we either reach the end of the curve we are tracing along. At this point, we clip the neighboring curve at the endpoint of the traced curve and merge them into a single curve, averaging the positions of the curve points.

Our overall tracing procedure has several favorable side-effects: it not only (locally) ignores non-maximal points, but also aligns the connected curve points along coherent local structures in the image Laplacian (and, thus, coherent color variation in the image domain).

4.3 Hierarchical Curve Consistency

When extracting curves *across* image scales (Section 4.2), we exploit the curve location and connectivity information gathered at coarser scales when extracting curves at finer scales. If we did not use this information and, for instance, directly extracted curves from each image scale independently, curves extracted at finer scales would become increasingly disconnected and less sparse (more and more short curves), and would risk overlapping existing (coarser) curve (sub-)segments [Lindeberg 1994]. We will detail how we leverage the multi-scale image stack created in Section 4.1 to impose a level-of-detail structure over the curves extracted across scales, so as to avoid the aforementioned problems. The final multi-scale structure will resolve large-scale image features with longer curves (that are extracted at coarser scales), and only then add smaller details with (fewer) shorter curves extracted at finer scales (see Fig. 6 (d,e,f)).

To preserve large-scale image features, curves are extracted from coarser scales first. When extracting curves at a finer level, we proceed as described in Section 4.2. However, before accepting

a newly extracted curve, we perform an additional test to verify whether a connected curve (extracted at the finer scale) overlaps with a curve extracted at *any* of the coarser scales. The main difficulty when performing this additional test is that, due to the nature of the bilateral filtering, similar curve structures may not *perfectly* overlap across scales. As such, we first define a 5×5 "overlap region" $\Omega_\cap$ and extract *all possible* curves (instead of just the optimal p-to-q^*-connected curve) within this region; we remove *every one* of these curves that overlaps with a curve from *any* of the coarser scales. The motivation for the size of $\Omega_\cap$, and the rejection-sampling that we apply to curves extracted within it, is that while the bilateral filter preserves edges, finer details may slightly shift when moving from u^j to u^{j+1}; the 5×5 coverage roughly corresponds to the increased footprint size between $\sigma_{s,j}$ and $\sigma_{s,j+1}$. Only the finer scale curves, extracted in $\Omega_\cap$, that were not rejected due to overlap in coarser scales are used when extracting curves in the finer scale. This allows finer scale curves to exclusively resolve finer-scale details.

At the coarsest scale, we additionally attempt to link the connected curves into longer (ideally closed) curves. Given two endpoints of a connected curve, q_a and q_b, we search for other connected curve endpoints within a user-defined search length d_{link} (all our results use $d_{\text{link}} = 5$ pixels), along the perpendicular directions $\mathbf{n}_\perp[q_a]$ and $\mathbf{n}_\perp[q_b]$. If another connected curve's endpoint q_c is located within the region (without loss of generality, when searching from q_b), then the pixels between q_b and q_c are set as connecting curve points, forming a longer linked curve. It is not uncommon for closed structures to naturally form when applying this procedure from both endpoints.

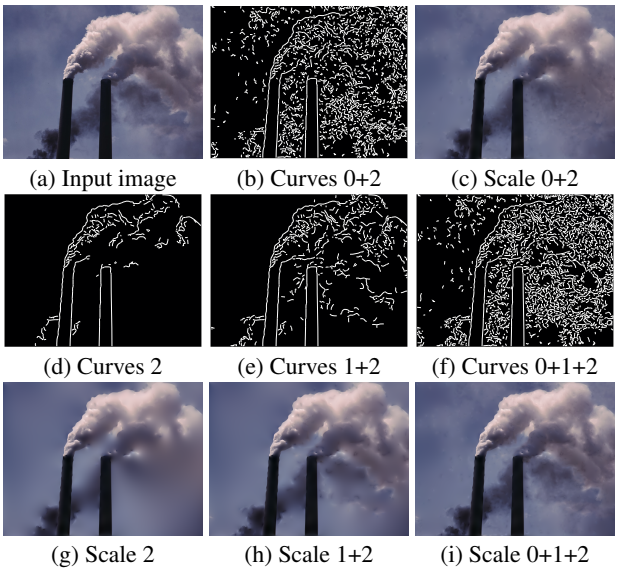
5 Solving for Multi-scale Curve Weights

We can also extract bilaplacian diffusion curves in the image bilaplacian domain by using the extraction algorithm in Section 4. Given the multi-scale extracted curves, we need to solve for endpoint weights in order to complete our diffusion curve representation. These weights vary depending on the type of diffusion curve we employ. We consider both Laplacian and bilaplacian diffusion curves and will discuss them in more detail below. At a high-level, during final image reconstruction, each pixel value is reconstructed as the sum of weighted Green's functions along all the extracted curves. Given an input image, and the curves we have extracted, we must first solve for these unknown weights (see Eq. (2)).

5.1 Laplacian and Bilaplacian Diffusion Curves

Our representation combines Laplacian and bilaplacian diffusion curves in a manner that leverages the advantages of each representation: Laplacian diffusion curves are more effective at resolving sharp and low order smooth regions, whereas bilaplacian diffusion curves are able to better resolve high order smooth regions in the images. Ilbery et al. [2013] show that Laplacian diffusion curves can be interpreted as a subset of bilaplacian diffusion curves, requiring only two weights at each curve endpoint (that are interpolated along *continuous* positions $\mathbf{x}$ in the vector image along the curve): the color along the curve boundary, $u(\mathbf{x})$, and the normal derivative along the curve boundary, $\partial u(\mathbf{x})/\partial \mathbf{n}(\mathbf{x})$. In contrast, bilaplacian diffusion curves require two additional (continuously interpolated) weights to be solved for: the Laplacian value along the curve boundary, $v(\mathbf{x})$, and the normal derivative of the Laplacian along the curve boundary, $\partial v(\mathbf{x})/\partial \mathbf{n}(\mathbf{x})$. Note the graduation from a discrete image notation $u[p]$ to a continuous one $u(\mathbf{x})$.

Since bilaplacian diffusion curves subsume Laplacian diffusion curves, a naïve approach to determine whether one of our extracted curves in the image bilaplacian domain should be classified as a

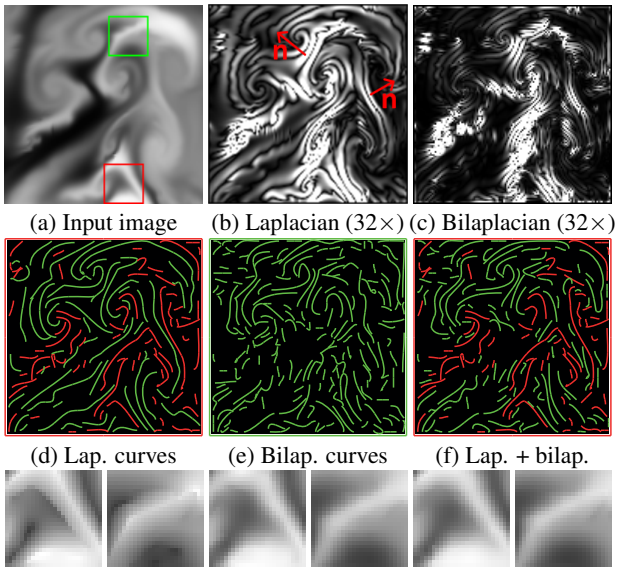


(a) Input image (b) Curves 0+2 (c) Scale 0+2
(d) Curves 2 (e) Curves 1+2 (f) Curves 0+1+2
(g) Scale 2 (h) Scale 1+2 (i) Scale 0+1+2

Figure 6: *Hierarchical curve extraction and reconstruction. Longer curves (d) reconstruct larger-scale image features (g) than their shorter counterparts (f). Curves at finer scales encompass all coarser-scale curves, providing a level-of-detail.*

Laplacian or bilaplacian diffusion curve (for the purposes of reconstruction) would begin by solving for its four unknowns, and then testing for whether the Laplacian and normal derivative of the Laplacian along the curve have negligible magnitude; if so, the curve can reconstruct the same color profile as a Laplacian diffusion curve, without requiring the additional reconstruction (and fitting) cost of a bilaplacian representation. Unfortunately, this naïve classification scheme demands that the more costly bilaplacian diffusion curve solver be applied to *every* curve. We instead show how to accurately classify curves *without solving for the bilaplacian weights*. Afterwards, we can apply the appropriate solver depending on our classification, reducing computation cost.

There is no guarantee that curves extracted exclusively in the Laplacian domain will be less sparse than those extracted exclusively from the bilaplacian domain, and vice-versa: indeed, there may be regions in an image where fewer curves would be traced from the Laplacian domain than from the bilaplacian domain, and regions where the opposite holds. We observe that regions in the Laplacian domain that exhibit sharper (Laplacian) intensity fall-off can be very well represented using Laplacian curves, whereas regions in the Laplacian domain that have smoother fall-off behavior are better represented using bilaplacian curves. As such, after extracting curves in the Laplacian domain, we use a voting scheme to classify each curve as a Laplacian or bilaplacian curve: for each point of each curve, we step along Laplacian values in the direction $\mathbf{n}$ on the left side of the curve until we reach a minimum Laplacian magnitude threshold (we use a threshold of 0.002); if the traced width is less than a classification threshold (3 pixels for all our examples), we increment the curves Laplacian classification vote by 1, otherwise we increment its bilaplacian classification vote by 1. We repeat the process by tracing along the right side of the curve and, after we have traced through the entire curve, we only classify it as a Laplacian curve if the ratio between the Laplacian and bilaplacian votes is greater than 0.5 (Fig. 7 (d)). Note that we only classify Laplacian curves with this strategy; to identify bilaplacian curves, we first extract curves in the bilaplacian domain (Fig. 7 (e)) and cull all bilaplacian curves that overlap with our Laplacian curves (using the algorithm in Section 4.3). The remaining bilaplacian curves are combined with our previously classified Laplacian curves to form our final set of diffusion curves (Fig. 7 (f)).



(a) Input image (b) Laplacian (32×) (c) Bilaplacian (32×)

(d) Lap. curves (e) Bilap. curves (f) Lap. + bilap.

(g) Lap. only (h) Bilap. only (i) Lap. + bilap.

Figure 7: *Determining Laplacian and bilaplacian diffusion curves and reconstruction comparison among Laplacian curves, bilaplacian-only curves and Laplacian + bilaplacian curves.*

We can apply this classification and merging routine to *automatically* choose Laplacian and bilaplacian diffusion curves during vectorization, however we observe the following behavior: synthesized images with multi-scale details, such as fluid simulations of smoke or fire, often have large smooth regions and require a mix of Laplacian and bilaplacian diffusion curves; however, DCIs and natural images can typically be reconstructed with high-accuracy using *only* Laplacian diffusion curves.

We compare reconstruction results with Laplacian diffusion curves, bilaplacian-only diffusion curves, and their combination, in Fig. 7. Note that, in Fig. 7g, the color discontinuity along the curves presents a very high color variation; if we apply bilaplacian-only diffusion curves, instead of Laplacian curves, the discontinuity disappears. In general, we found that automatically combining both types of diffusion curves is a conservative solution: Fig. 7h,i show similar reconstruction results using bilaplacian-only and combined Laplacian and bilaplacian diffusion curves.

5.2 Solving for Laplacian and Bilaplacian Weights

After extracting our *discrete* curves (Section 4), we generate Bezier curves (similar to Orzan et al. [2008]) from the pixel chains. Meanwhile, we subdivide long curves to shorter curves, making sure the length of each Bezier curve is not larger than a threshold (usually set to 25 pixels). We connect Bezier curve segments with preserving C2 continuity. Then, inspired by Sun et al. [2012] and Ilbery et al. [2013], we employ a Green's function formulation to express both the final reconstructed image, and the weight-fitting equation. Our choice here is motivated by the fact that Green's functions combine boundary values and boundary normal derivatives along the curves. This scheme is more accurate than Orzan et al.'s [2008] approach where only the boundary values are directly solved for. We will demonstrate that our reconstruction results are more accurate than DCI [Orzan et al. 2008].

For our Laplacian or bilaplacian diffusion curves, we propose a unified formulation for the contribution of a single curve to the final (continuous) image color $u(\mathbf{x})$ as:

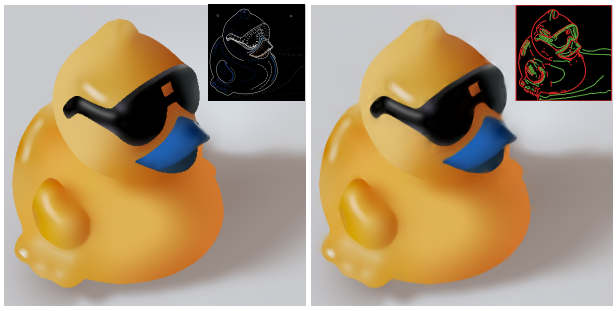
$$u(\mathbf{x}) = \oint_{\partial \mathbf{D}} \left(\frac{\partial u(\mathbf{x}')}{\partial \mathbf{n}(\mathbf{x}')} G^L(\mathbf{x}, \mathbf{x}') - u(\mathbf{x}') \frac{\partial G^L(\mathbf{x}, \mathbf{x}')}{\partial \mathbf{n}(\mathbf{x}')} \right) \mathrm{d}\mathbf{x}' + \oint_{\partial \mathbf{D}} \left(\frac{\partial v(\mathbf{x}')}{\partial \mathbf{n}(\mathbf{x}')} G^B(\mathbf{x}, \mathbf{x}') - v(\mathbf{x}') \frac{\partial G^B(\mathbf{x}, \mathbf{x}')}{\partial \mathbf{n}(\mathbf{x}')} \right) \mathrm{d}\mathbf{x}' \quad (2)$$

where, $G^L(\mathbf{x}, \mathbf{y})$ is the Laplacian Green's function and $G^B(\mathbf{x}, \mathbf{y})$ the bilaplacian Green's function, and we employ the same notation as Ilbery et al. [2013].

As discussed earlier, for those curves classified as Laplacian diffusion curves, only the first line in Eq. (2) is relevant. In order to reconstruct the original image accurately, we solve for the weights (color $u(\mathbf{x}')$, normal derivatives of color $\partial u(\mathbf{x}')/\partial \mathbf{n}(\mathbf{x}')$, the image Laplacian $v(\mathbf{x}')$, and the normal derivative of the image Laplacian $\partial v(\mathbf{x}')/\partial \mathbf{n}(\mathbf{x}')$) of the Green's functions Eq. (2) by sampling colors $u(\mathbf{x})$ throughout the image and applying a least-squares solver to our fitting equation Eq. (3). We modify the solver proposed by Sun et al. [2012] and uniformly discretize C diffusion curves at N points $\{\mathbf{x}'_\mathbf{i}\}$ and solve for the weights in the following linear equation (for a single color channel):

$$\begin{bmatrix} -l_1 \frac{\partial G^L(\mathbf{x_1}, \mathbf{x}'_\mathbf{1})}{\partial \mathbf{n}(\mathbf{x}'_\mathbf{1})} & \dots & -l_1 \frac{\partial G^L(\mathbf{x_M}, \mathbf{x}'_\mathbf{1})}{\partial \mathbf{n}(\mathbf{x}'_\mathbf{1})} \\ l_1 G^L(\mathbf{x_1}, \mathbf{x}'_\mathbf{1}) & \ddots & l_1 G^L(\mathbf{x_M}, \mathbf{x}'_\mathbf{1}) \\ -l_1 \frac{\partial G^B(\mathbf{x_1}, \mathbf{x}'_\mathbf{1})}{\partial \mathbf{n}(\mathbf{x}'_\mathbf{1})} & \ddots & -l_1 \frac{\partial G^B(\mathbf{x_M}, \mathbf{x}'_\mathbf{1})}{\partial \mathbf{n}(\mathbf{x}'_\mathbf{1})} \\ l_1 G^B(\mathbf{x_1}, \mathbf{x}'_\mathbf{1}) & \ddots & l_1 G^B(\mathbf{x_M}, \mathbf{x}'_\mathbf{1}) \\ \vdots & \ddots & \vdots \\ -l_N \frac{\partial G^L(\mathbf{x_1}, \mathbf{x}'_\mathbf{N})}{\partial \mathbf{n}(\mathbf{x}'_\mathbf{N})} & \ddots & -l_N \frac{\partial G^L(\mathbf{x_M}, \mathbf{x}'_\mathbf{N})}{\partial \mathbf{n}(\mathbf{x}'_\mathbf{N})} \\ l_N G^L(\mathbf{x_1}, \mathbf{x}'_\mathbf{N}) & \ddots & l_N G^L(\mathbf{x_M}, \mathbf{x}'_\mathbf{N}) \\ -l_N \frac{\partial G^B(\mathbf{x_1}, \mathbf{x}'_\mathbf{N})}{\partial \mathbf{n}(\mathbf{x}'_\mathbf{N})} & \ddots & -l_N \frac{\partial G^B(\mathbf{x_M}, \mathbf{x}'_\mathbf{N})}{\partial \mathbf{n}(\mathbf{x}'_\mathbf{N})} \\ l_N G^B(\mathbf{x_1}, \mathbf{x}'_\mathbf{N}) & \dots & l_N G^B(\mathbf{x_M}, \mathbf{x}'_\mathbf{N}) \end{bmatrix}^\top \begin{bmatrix} \vdots \\ u(\mathbf{x}'_\mathbf{i}) \\ \frac{\partial u(\mathbf{x}'_\mathbf{i})}{\partial \mathbf{n}(\mathbf{x}'_\mathbf{i})} \\ v(\mathbf{x}'_\mathbf{i}) \\ \frac{\partial v(\mathbf{x}'_\mathbf{i})}{\partial \mathbf{n}(\mathbf{x}'_\mathbf{i})} \\ \vdots \end{bmatrix} = \begin{bmatrix} u(\mathbf{x_0}) \\ \vdots \\ u(\mathbf{x_j}) \\ \vdots \\ u(\mathbf{x_M}) \end{bmatrix} \quad (3)$$

where l_i is the arc length of a curve between sampling points $\mathbf{x}'_\mathbf{i}$ and $\mathbf{x}'_{\mathbf{i-1}}$, the right hand side of the equation is the vector of input image $u(\mathbf{x})$ with M pixels, and each row of the matrix is the contributions of the kernels (evaluated at the sampling points) to a pixel $u(\mathbf{x_j})$. The equation (for three color channels, $\mathbf{Ax} = \mathbf{b}$) has dimensions $[M \times 4N] \times [4N \times 3] = [M \times 3]$. Directly solving this linear system for all sampling points and all image pixels is heavily time consuming, even impossible for large image sizes. We employ four strategies to reduce the size of the problem: first, we assume that the weights vary smoothly along curves and can be linearly interpolated by the weights defined at the endpoints. So we replace $w(\mathbf{x}') = t\, w_s(\mathbf{x}') + (1-t)\, w_e(\mathbf{x}')$, where $w_s(\mathbf{x})$ and $w_e(\mathbf{x})$ are kernel weights (all four terms in Eq. (2)) at curve endpoints and t is the parametric variable of the Bezier curve. We replace all $w(\mathbf{x}')$ values in the linear system with these interpolated values. Thus the $[M \times 4N]$ matrix reduces to $[M \times 8C]$; secondly, since $C << M$, we multiply both sides of the equation by the transpose of the $[M \times 8C]$ matrix, resulting in a final matrix equation (that scales with C instead of M) with dimension $[8C \times 8C] \times [8C \times 3] = [8C \times 3]$ (similar to Jeschke et al. [2011]); thirdly, if we only use Laplacian curves, all the $8C$'s above further reduce to $4C$; lastly, due to connected Bezier curves having shared endpoints, we use shared endpoints to interpolate two connected Bezier curves and furthermore reduce the matrix dimension. Through these strategies, we reduce the number of unknowns and highly accelerate the computation.



FFG curves & result Our curves & result

Figure 8: *We automatically generate curves similar to manually-authored curves, matching the quality of [Finch et al. 2011].*

We need to solve for the weights in a manner that permits the level-of-detail reconstruction scheme detailed earlier: namely, for all L scales, that image reconstruction at a fixed scale l should employ all curves at that scale l and all those coarser than it, $\{l+1,\ldots,L-1\}$. In other words, scale L includes only curves at the coarsest scale, while scale 0 includes all curves at all scales. In order to reconstruct the image at a given scale, we solve a simultaneous set of functions Eq. (2) to fit all the required scales to the original image using Eq. (4), generating the same weights across scales for each curve by enforcing a constraint in the solver. If, instead, we fit the weights independently at each scale, a curve's Green's function weights would differ across scales. Solving for the weights across scales results in a representation that is suitable for multi-scale editing (e.g., adding/removing features at a specific scale) without re-fitting the weights of curves (see Fig. 6)

$$\{\mathbf{A}_{L-1}\mathbf{x}_{L-1}=\mathbf{b},\ \ldots,\ \mathbf{A}_0\mathbf{x}_0=\mathbf{b}\}\ \text{subject to: } \mathbf{x}_{l_i}(k)=\mathbf{x}_{l_j}(k) \tag{4}$$

As with the closed-form integral of the Laplacian over a rectangular domain presented in Sun et al. [2012], we show that a closed-form solution exists for the integral of the Bilaplacian $G^B(\mathbf{x},\mathbf{x}')$ (and its normal derivatives $G^B_n(\mathbf{x},\mathbf{x}')=\partial G^B(\mathbf{x},\mathbf{x}')/\partial\mathbf{n}(\mathbf{x}')$) over a rectangular domain: the closed-form integrals F_{G^B} and $F_{G^B_n}$ are:

$$F_{G^B}(\mathbf{R},\mathbf{x}')=\iint_{\mathbf{R}} G^B(\mathbf{x},\mathbf{x}')\,\mathrm{d}\mathbf{x}=\sum_{i,j\in\{0,1\}}(-1)^{i+j}H_{G^B}(\hat{x},\hat{y}) \tag{5}$$

where

$$H_{G^B}(\hat{x},\hat{y})=\frac{1}{48\pi}\left(\hat{x}^4-\hat{y}^4\right)\operatorname{atan}\left(\frac{\hat{x}}{\hat{y}}\right)+\frac{1}{144\pi}\hat{x}\hat{y}\left(\hat{x}^2+\hat{y}^2\right)\left(11-3\ln\left(\hat{x}^2+\hat{y}^2\right)\right) \tag{6}$$

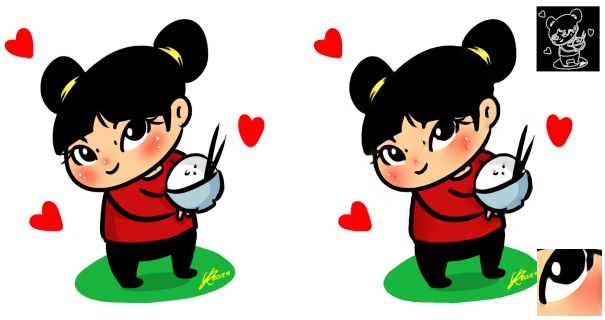
and

$$F_{G^B_n}(\mathbf{R},\mathbf{x}')=\iint_{\mathbf{R}} G^B_n(\mathbf{x},\mathbf{x}')\,\mathrm{d}\mathbf{x}=\sum_{i,j\in\{0,1\}}(-1)^{i+j}H_{G^B_n}(\hat{x},\hat{y},n_x,n_y) \tag{7}$$

where

$$\begin{aligned}&H_{G^B_n}(\hat{x},\hat{y},n_x,n_y)\\&=\frac{1}{48\pi}\left(10\hat{x}\hat{y}\left(\hat{x}n_x+\hat{y}n_y\right)-4\hat{y}^3n_y\operatorname{atan}\left(\frac{\hat{x}}{\hat{y}}\right)-4\hat{x}^3n_x\operatorname{atan}\left(\frac{\hat{y}}{\hat{x}}\right)\right)\\&\quad-\frac{1}{48\pi}\left(\hat{x}^3n_y+3\hat{x}^2\hat{y}n_x+3\hat{x}\hat{y}^2n_y+\hat{y}^3n_x\right)\ln\left(\hat{x}^2+\hat{y}^2\right)\end{aligned} \tag{8}$$

where $\mathbf{R}=\{x\in(x_0,x_1),y\in(y_0,y_1)\}$, $\mathbf{x}=(x,y)$, $\mathbf{x}'=(x',y')$, $\mathbf{n}(\mathbf{x}')=(n_x,n_y)$, and $(\hat{x},\hat{y})=\mathbf{x}-\mathbf{x}'$. Detailed derivations are provided in the supplemental material and we us our analytic integration to perform efficient pixel anti-aliasing. Our integral is robust to the singularity present when a boundary point falls exactly atop a pixel center.



Input image Our curves & result

Figure 9: *We automatically generate curves from cartoon images. The eye part of right image is close-up view.*

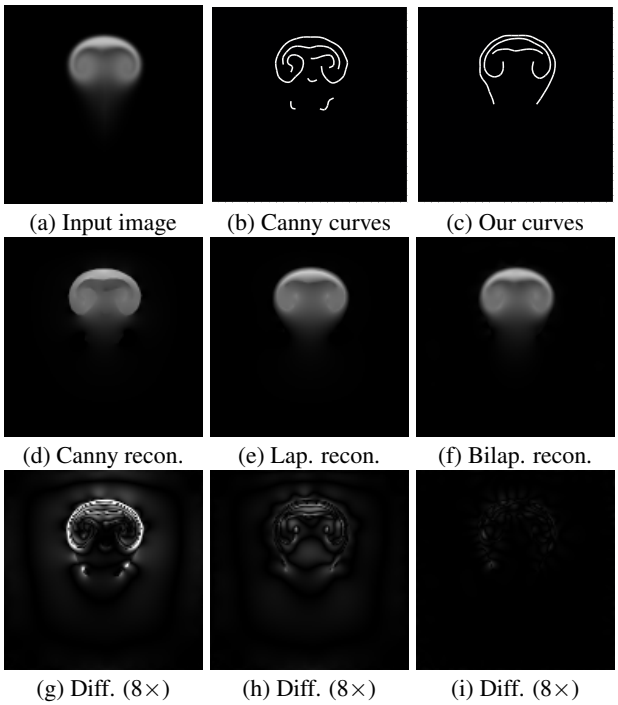
Editing. We can readily perform multi-scale color editing based on our Green's function formulation by simply fixing the values of the curve color profile $u(\mathbf{x}')=-u_l(\mathbf{x}')+u_r(\mathbf{x}')$ in Eq. (4), where u_l and u_r are the left-side and right-side color constraints for a curve. Interestingly, any editing constraints we impose improve the curve weight fitting procedure since we reduce the number of unknown terms during the weight solving process.

6 Results and Discussion

Performance. Results were generated on an Intel Core i7-3770K 3.50 GHz CPU and an Nvidia Quadro 6000 graphics card, with fitting executed on the CPU with 8 threads and reconstruction executed on the GPU. Table 1 lists the statistics and performance for all images. As shown in the table, multi-scale curve pixel identification and extraction takes between **2ms** and **123ms**. Then, using four scales (the setting used for most results), the remainder of the algorithm depends on the resolution of input images and the distribution of local maxima in the image Laplacian and/or bilaplacian. The fitting and reconstruction cost grows linearly with the number of curves. Fitting time is between **0.25s** and **11 mins**. After additionally adapting the curve culling and adaptive sampling schemes in [Sun et al. 2012] based on the parallel characteristics over pixels and curves, all of our reconstruction results can be interactively computed in **1.9ms** and **623ms** by using the integral form of Green's functions for anti-aliasing. Evaluating Green's function kernels forms our computation bottleneck.

Figure	Image resolution	Number of curves	Storage (KB)	Extract (ms)	Fitting (s)	Recon. (ms)
Fig. 1a	256×256	1183	95	8	47.6	92.3
Fig. 1b	600×600	4526	363	99	581.6	591.2
Fig. 1c	440×440	700	56	12	37.6	103.8
Fig. 1d	512×512	1999	160	35	47.2	162.6
Fig. 6	462×358	2446	320	33	348.1	401.4
Fig. 7	128×128	42	4	2	0.25	1.9
Fig. 9	880×907	446	36	41	28.9	93.5
Fig. 8	440×440	708	86	13	48.1	172.8
Fig. 11a	484×369	3285	263	47	301.2	392.5
Fig. 11j	946×633	3722	298	123	660.2	622.7
Fig. 12	800×800	1636	131	40	122.6	349.2
Fig. 14	268×400	1488	119	19	21.9	59.2
Fig. 15	553×707	643	51	10	39.1	138.1
Fig. 16a	600×450	3567	286	71	360.2	405.9
Fig. 16b	640×480	4327	347	82	451.8	491.5
Fig. 16c	578×566	2983	239	51	254.1	379.2
Fig. 16d	222×222	970	78	7	22.8	40.8
Fig. 16e	420×315	1996	160	24	42.9	148.2

Table 1: *Statistics and performance for the images shown in this paper.*



(a) Input image (b) Canny curves (c) Our curves

(d) Canny recon. (e) Lap. recon. (f) Bilap. recon.

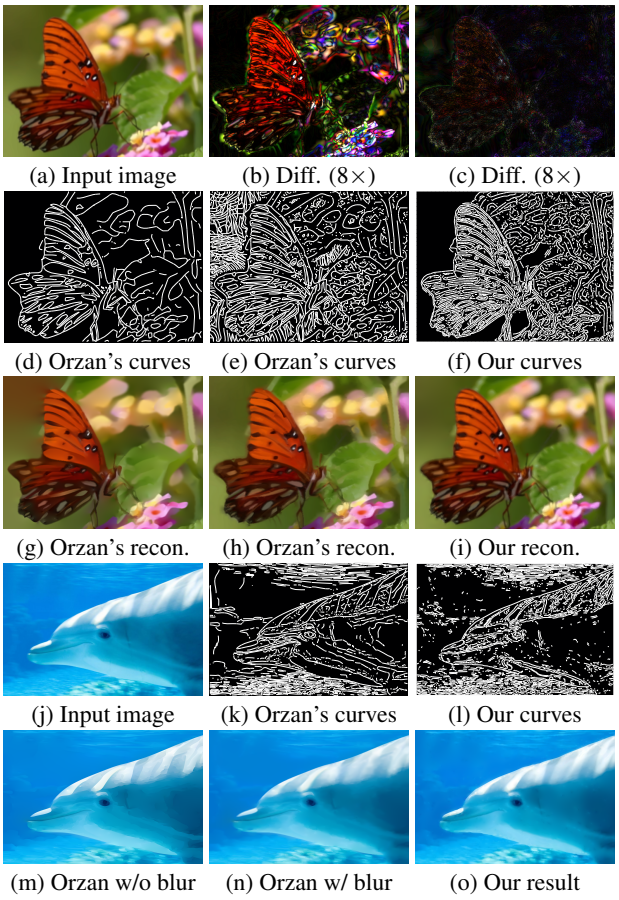
(g) Diff. (8×) (h) Diff. (8×) (i) Diff. (8×)

Figure 10: *Comparing reconstruction with curves extracted from the gradient (Canny), Laplacian and bilaplacian domains.*

Comparison. We validate our automatically generated curves (and reconstruction) with manually authored DCIs [Orzan et al. 2008] and thin-plate splines [Finch et al. 2011] by fitting our curves directly to their authored results. In Fig. 1 (c), our curves are almost identical to the given DCI curves in [Orzan et al. 2008]. In Fig. 8, we automatically fit curves to a (manually authored) FFG reconstruction [Finch et al. 2011]. Our curves bear a striking resemblance to Finch et al.'s sketched curves and our reconstruction retains both the sharp image edges and the smooth color variations by using Laplacian and bilaplacian diffusion curves. We can also extract curves from cartoon images in Fig. 9. Fig. 10 compares curve extraction and reconstruction in the gradient, Laplacian and bilaplacian domains.

Unlike the automatic extraction method of Orzan et al. [2008], we can resolve intricate image features (Fig. 11 (i)). We note our improved reconstruction of the detailed texture on the butterfly's wings and pink flower, as well as on the (out-of-focus) leaf. Our reconstruction error is lower in Fig. 11 (c), while more dense curves (Fig. 11 (e)) do not reduce the error significantly (Fig. 11b) for [Orzan et al. 2008]. Meanwhile, we compare the extracted curves and reconstruction results in Fig. 11 (o). Here, the input natural image (Fig. 11 (j)) is taken from [Orzan et al. 2008]. We reconstruct a smoother result that is closer to the original image, while the method in [Orzan et al. 2008] reconstructs sharp boundaries (Fig. 11 (m)) without blurring. In order to generate smoother results, the method in [Orzan et al. 2008] needs to use apply a costly expensive post-processing blur.

Given a set of existing Bezier curves, Jeschke et al. [2011] also use a global optimization to solve the diffusion curve coloring problem, including positioning color points and fitting color values, which can well reconstruct image features of given curves. We use another different fitting algorithm. Given the same curves, we can reconstruct a similar result (Fig. 12 (e)). We only compare the representation ability of curves without post-processing (adding blur and textures). However, our method proposes a unified framework, including extracting curves and fitting/reconstructing images. Curves



(a) Input image (b) Diff. (8×) (c) Diff. (8×)

(d) Orzan's curves (e) Orzan's curves (f) Our curves

(g) Orzan's recon. (h) Orzan's recon. (i) Our recon.

(j) Input image (k) Orzan's curves (l) Our curves

(m) Orzan w/o blur (n) Orzan w/ blur (o) Our result

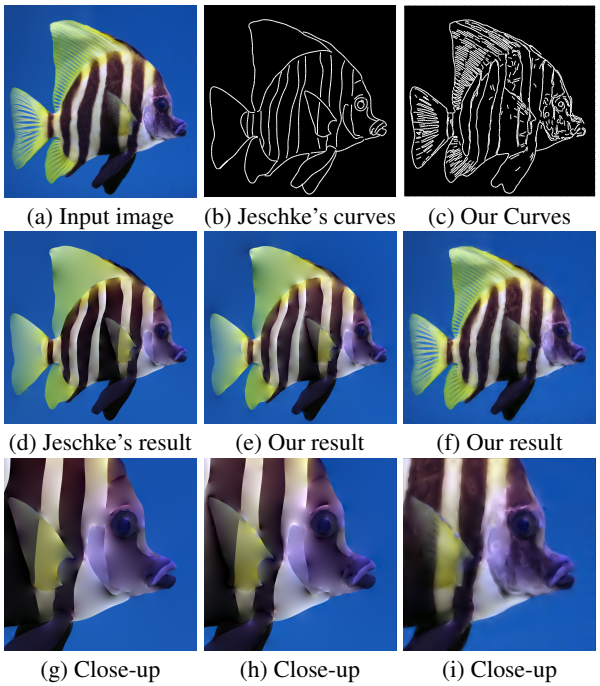
Figure 11: *Comparing our results with automatically extracted curves and reconstruction results from [Orzan et al. 2008]. (d) and (e) are extracted with different initial Gaussian scales, (m) is reconstructed without any post-processing (blurring). And (g), (h) and (n) are reconstructed after applying a post-processing blur; (i) and (o) are reconstructed by our method with our curves (f) and (l), and (b) and (c) are the differences of (h) and (i) with (a).*

of some features which are difficult to manually extract can be automatically extracted by our method, like the tail and dorsal fin of the fish in Fig. 12 (f).

We also compare our method with gradient meshes methods: both techniques capture image features with smooth variations and sharp edges, and we illustrate results generated by both methods in Fig. 13.

Results. While image editing is not an explicit goal of our work, we can immediately leverage our hierarchical representation to perform simple but powerful image manipulations, including detail removal and multi-scale shape abstraction. Fig. 6 (g,h,i) illustrates different levels of detail for an image, allowing a simple multi-scale abstraction. By removing mid-scale curves (Scale 1) in Fig. 6 (b), we can generate a different result without refitting the weights of curves. Fig. 14 illustrates how our multi-scale curves can readily be used for (vectorized) image abstraction: the coarsest scale reconstruction preserves large-scale features (e.g., the silhouette of the head and jacket), while finer scales progressively re-introduce the small-scale details.

Fig. 15 illustrates a simple editing result: given an input image (Fig. 15 (a)), we first extract Laplacian diffusion curves and color profiles to reconstruct the original image (Fig. 15 (c)) before modifying the normal derivate weights in order to generate the edited

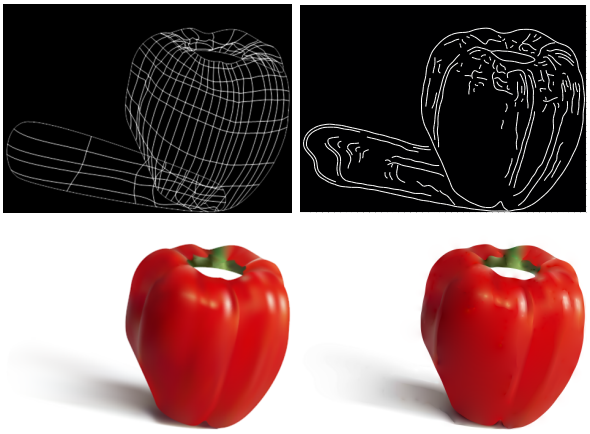


(a) Input image (b) Jeschke's curves (c) Our Curves

(d) Jeschke's result (e) Our result (f) Our result

(g) Close-up (h) Close-up (i) Close-up

Figure 12: *Comparing our results with the method in [Jeschke et al. 2011]: (e) is reconstructed using our method with the curves from Jeschke et al.'s [2011] curves (b); (g), (h), and (i) are close-up views of (d), (e) and (f).*



Gradient meshes & result Our curves & result

Figure 13: *Comparing reconstruction with optimized gradient meshes [Sun et al. 2007] and our method.*

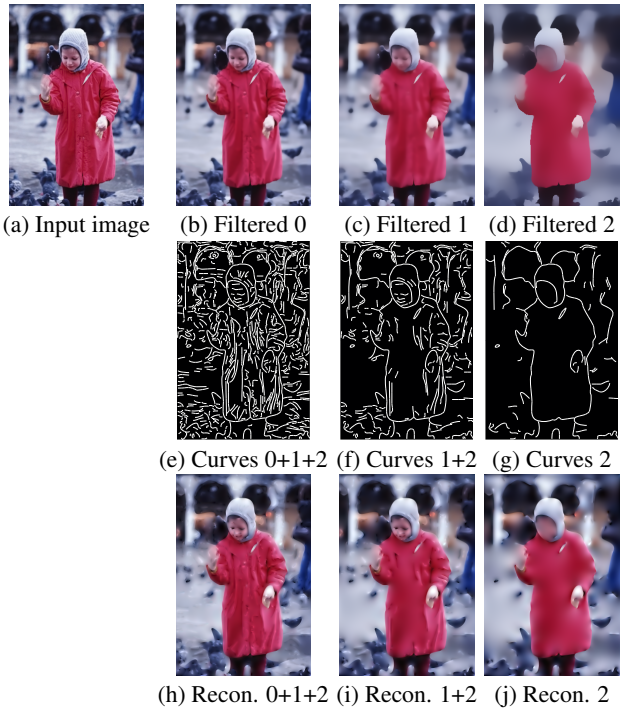
result in Fig. 15 (d).

Fig. 16 illustrates our multi-scale curves and reconstruction results for complex natural images and a turbulent computer synthesized fire simulation result. Our results clearly reconstruct both the coarse- and fine-scale details of these challenging images.

Limitations. Although our method works well across a large class of images, it still has several limitations: the multi-scale bilateral filter may remove image features with low contrast and, when vectorizing noisy natural images with small initial spatial filter widths, our algorithm can sometimes generate many short curves.

7 Conclusion

We present a robust and fully-automatic method to vectorize images using a hybrid, sparse diffusion curve representation based on

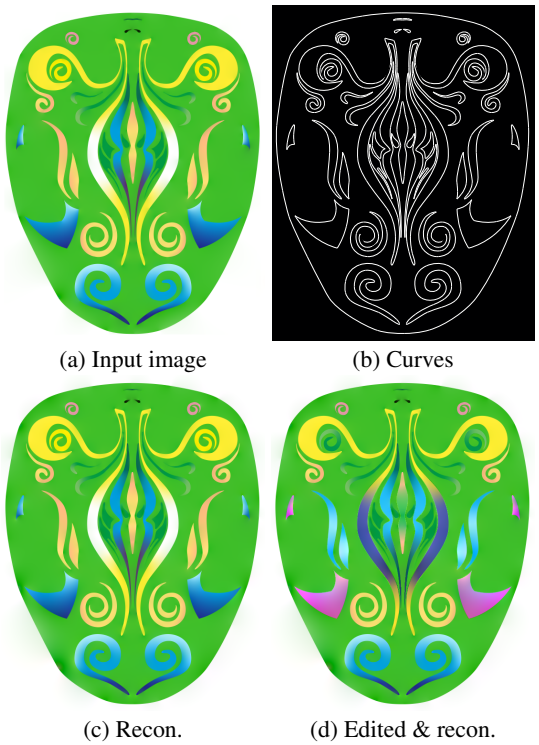


(a) Input image (b) Filtered 0 (c) Filtered 1 (d) Filtered 2

(e) Curves 0+1+2 (f) Curves 1+2 (g) Curves 2

(h) Recon. 0+1+2 (i) Recon. 1+2 (j) Recon. 2

Figure 14: *Hierarchical curve extraction and reconstruction.*



(a) Input image (b) Curves

(c) Recon. (d) Edited & recon.

Figure 15: *Color editing: (d) is generated without any refitting.*

DCIs and FFG. We leverage a fundamental observation that extracting curves in the Laplacian domain, along local maxima, serves to *directly* sample these primitives where they will have the most impact on image reconstruction. Our approach is robust to a broad class of input, as illustrated in our results, and we compare favorably with previous manual-authoring approaches. We classify our extracted curves as Laplacian or bilaplacian diffusion curves using a simple but robust scheme, yielding improved reconstruction quality

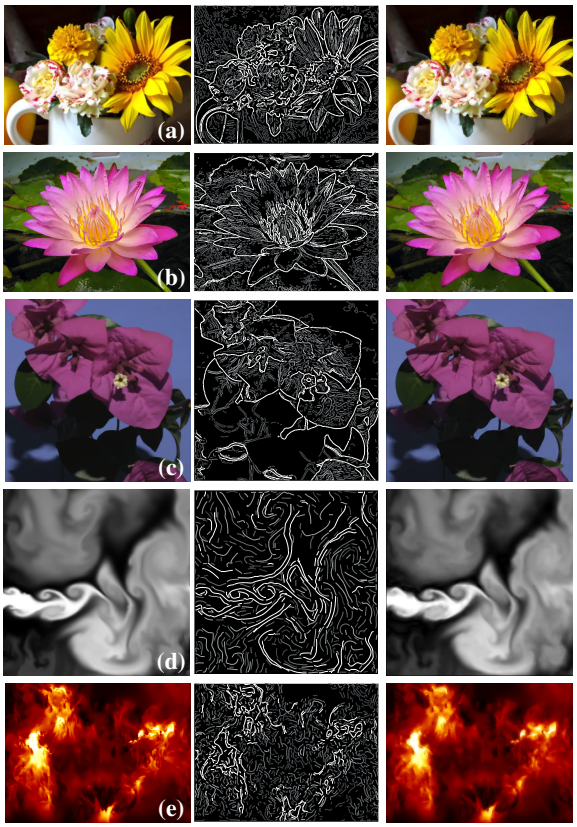


Figure 16: *Auto-vectorization of natural and simulated images.*

with a measured increase in computation. Our hierarchical fitting and multi-scale representation present a level-of-detail that permits both vectorized abstractions and intuitive curve structures.

Two interesting related avenues of future work include an extension to 3D volume/density vectorization and vectorization of animation sequences, where we hope to exploit inter-frame coherence by extracting "curves" in a higher-dimensional space (e.g., space-time).

Acknowledgements

We thank the anonymous reviewers for their valuable comments and suggestions. The authors also thank Adrien Bousseau for sharing his implementation of the approach in [Orzan et al. 2008], as well as providing technical support required to perform the comparison in Fig. 11. The authors also thank Pierre Poulin and Stephen Lin for proofreading the paper. Guofu Xie was supported by the Natural Sciences and Engineering Research Council of Canada, a MITACS Accelerate Cluster in collaboration with Microsoft Game Studios, and NSF of China (61379087). This work was completed while Guofu Xie was a Post-Doc at the University of Montreal.

References

CANNY, J. 1986. A computational approach to edge detection. *IEEE Trans. PAMI 8* (June), 679–698.

CHANG, H.-H., AND YAN, H. 1998. Vectorization of hand-drawn image using piecewise cubic bézier curves fitting. *Pattern Recognition 31*, 11, 1747–1755.

DEMARET, L., DYN, N., AND ISKE, A. 2006. Image compression by linear splines over adaptive triangulations. *Signal Process. 86*, 7 (July), 1604–1616.

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

FATTAL, R., AGRAWALA, M., AND RUSINKIEWICZ, S. 2007. Multiscale shape and detail enhancement from multi-light image collections. *ACM Trans. Graph. 26* (July), 51:1–51:9.

FINCH, M., SNYDER, J., AND HOPPE, H. 2011. Freeform vector graphics with controlled thin-plate splines. *ACM Trans. Graph. 30* (Dec.), 166:1–166:10.

HILAIRE, X., AND TOMBRE, K. 2006. Robust and Accurate Vectorization of Line Drawings. *IEEE Trans. PAMI 28*, 6, 890–904.

ILBERY, P., KENDALL, L., CONCOLATO, C., AND MCCOSKER, M. 2013. Biharmonic diffusion curve images from boundary elements. *ACM Trans. Graph. 32* (Nov.), 219:1–219:12.

JESCHKE, S., CLINE, D., AND WONKA, P. 2011. Estimating color and texture parameters for vector graphics. *Computer Graphics Forum 30*, 2 (Apr.), 523–532.

LAI, Y.-K., HU, S.-M., AND MARTIN, R. R. 2009. Automatic and topology-preserving gradient mesh generation for image vectorization. *ACM Trans. Graph. 28* (July), 85:1–85:8.

LECOT, G., AND LÉVY, B. 2006. Ardeco: Automatic region detection and conversion. In *EGSR*, 349–360.

LIAO, Z., HOPPE, H., FORSYTH, D., AND YU, Y. 2012. A subdivision-based representation for vector image editing. *IEEE TVCG 18*, 11, 1858–1867.

LINDEBERG, T. 1994. *Scale-Space Theory in Computer Vision*. Kluwer Academic Publishers.

LINDEBERG, T. 1998. Edge detection and ridge detection with automatic scale selection. *IJCV 30*, 2, 117–154.

ORZAN, A., BOUSSEAU, A., BARLA, P., AND THOLLOT, J. 2007. Structure-preserving manipulation of photographs. In *NPAR*, 103–110.

ORZAN, A., BOUSSEAU, A., WINNEMÖLLER, H., BARLA, P., THOLLOT, J., AND SALESIN, D. 2008. Diffusion curves: a vector representation for smooth-shaded images. *ACM Trans. Graph. 27* (July), 92:1–92:8.

PERONA, P., AND MALIK, J. 1990. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. PAMI 12*, 7, 629–639.

SUN, J., LIANG, L., WEN, F., AND SHUM, H.-Y. 2007. Image vectorization using optimized gradient meshes. *ACM Trans. Graph. 26* (July), 11:1–11:7.

SUN, X., XIE, G., DONG, Y., LIN, S., XU, W., WANG, W., TONG, X., AND GUO, B. 2012. Diffusion curve textures for resolution independent texture mapping. *ACM Trans. Graph. 31* (July), 74:1–74:9.

SUN, T., THAMJAROENPORN, P., AND ZHENG, C. 2014. Fast multipole representation of diffusion curves and points. *ACM Trans. Graph. 33* (Aug.).

SWAMINARAYAN, S., AND PRASAD, L. 2006. Rapid automated polygonal image decomposition. In *AIPR*, IEEE Computer Society, 28.

XIA, T., LIAO, B., AND YU, Y. 2009. Patch-based image vectorization with automatic curvilinear feature alignment. *ACM Trans. Graph. 28* (December), 115:1–115:10.

ZOU, J. J., AND YAN, H. 2001. Cartoon image vectorization based on shape subdivision. In *Computer Graphics International*, IEEE Computer Society, 225–231.