Biharmonic Diffusion Curve Images from Boundary Elements

Peter Ilbery*[1], Luke Kendall[1], Cyril Concolato[2], Michael McCosker[1]
[1]Canon Information Systems Research Australia (CiSRA), [2]Institut Mines-Télécom; Télécom ParisTech; CNRS LTCI

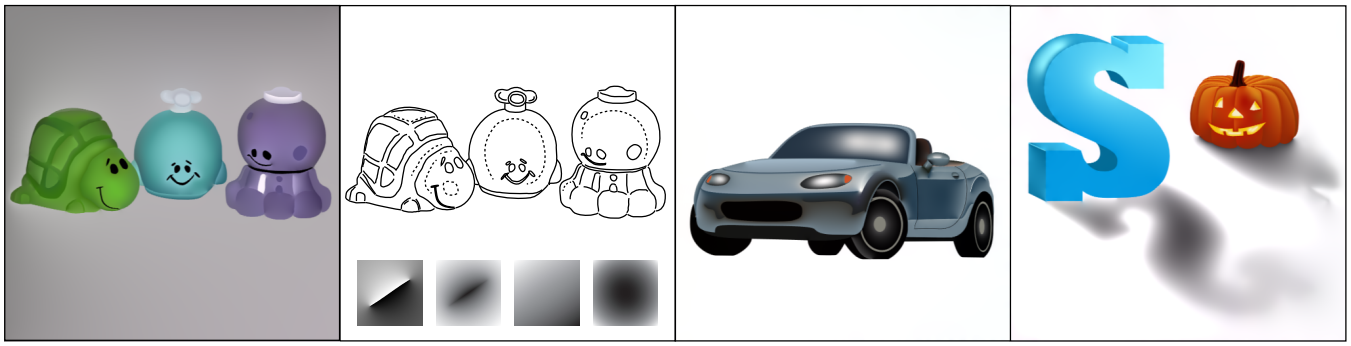

Figure 1: *Boundary element rendering of biharmonic diffusion curve images. From left to right: toys image; sharp-profile (solid) and smooth-profile (dotted) curves of the toys image, with thumbnail images below showing examples of the 4 building block segment fields; car image and pumpkin image. The above images are ©CiSRA; the toys image is a CiSRA re-creation of a photograph taken by C. Concolato.*

Abstract

There is currently significant interest in freeform, curve-based authoring of graphic images. In particular, "diffusion curves" facilitate graphic image creation by allowing an image designer to specify naturalistic images by drawing curves and setting colour values along either side of those curves. Recently, extensions to diffusion curves based on the biharmonic equation have been proposed which provide smooth interpolation through specified colour values and allow image designers to specify colour gradient constraints at curves. We present a Boundary Element Method (BEM) for rendering diffusion curve images with smooth interpolation and gradient constraints, which generates a solved boundary element image representation. The diffusion curve image can be evaluated from the solved representation using a novel and efficient line-by-line approach. We also describe "curve-aware" upsampling, in which a full resolution diffusion curve image can be upsampled from a lower resolution image using formula evaluated corrections near curves. The BEM solved image representation is compact. It therefore offers advantages in scenarios where solved image representations are transmitted to devices for rendering and where PDE solving at the device is undesirable due to time or processing constraints.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Display algorithms;

Keywords: vector graphics, diffusion curves, boundary elements

Links: DL PDF

*peter.ilbery@cisra.canon.com.au

1 Introduction

Vector graphic images are easily editable and scalable, with compact representations. A key issue for the creation of naturalistic vector graphic images is how an image designer specifies colour variation ("colour gradients") in the image.

Diffusion curves. [Orzan et al. 2008], [Orzan et al. 2013] propose "diffusion curves" for image designers to specify colour variation by drawing curves, setting colour values at sparse points along either side of the curves, and setting "degree of blur" values at sparse points along the curves. A diffusion curves image is generated by an interpolation process followed by a blur process. In the interpolation process, initial image colours are determined at remaining points along curves by 1D interpolation, and then at points away from curves by solving, for each colour component, a partial differential equation (PDE) with boundary constraints provided by the colour values along the curves. In the blur process, degree of blur values are determined throughout the image in a similar fashion to the initial image colours and are then used to apply space-variant blur filtering to transform the initial image colours to final image colours. [Jeschke et al. 2009] describes a fast solver for diffusion curves images; they also describe the interpolation process as solving the 2D Laplace equation with values specified at boundaries, i.e. Dirichlet boundary conditions.

In the absence of blur, diffusion curve images typically have a jump or a gradient discontinuity across the curves; that is, image values have a "sharp-profile" across curves. The diffusion curves blur process may leave the image value profile across a curve unchanged (zero blur) or it may perform some averaging of colours across the curve to form a "smooth-profile". However, the diffusion curves blur process has disadvantages. Typically, at a curve, the blur process does not preserve the average of the left and right colour values an image designer specifies along the curve, including when left and right colours are the same; that is, diffusion curves do not provide smooth interpolation through specified colour values. As well, the blur process has the problem that it can generate colours at a curve having non-zero blur by undesirably drawing colours from both sides, including the far side, of a nearby curve with no blur.

Biharmonic diffusion curves. [Finch et al. 2011] extends the dif-

fusion curves image model to provide smooth interpolation through colour constraints while omitting the diffusion curves blur operation. The smooth interpolation is achieved using biharmonic interpolation. [Boyé et al. 2012] describe rendering diffusion curves images with smooth interpolation and gradient constraints using a finite element method (FEM) biharmonic equation solver. They generalize and neatly classify previous types of constraint curves.

Previous boundary element methods. [van de Gronde 2010] describes rendering diffusion curves (without blur) using a boundary element method for solving the Laplace PDE. [Sun et al. 2012] represent diffusion curves images, without blur and without gradient constraints, using a compact representation of fundamental solutions and fitted (i.e. solved) weights, which they call "diffusion curve textures".

The main contributions of our paper are:

- a unified mathematical description of diffusion curves and biharmonic diffusion curves and their closed curve isolation properties;
- a compact (BEM) solved representation for biharmonic diffusion curve images;
- a computationally efficient line-by-line method for evaluating diffusion curve images or biharmonic diffusion curve images, from a solved boundary element representation;
- a curve-aware upsampling technique for increasing the resolution of a diffusion curves image or a biharmonic diffusion curves image using the BEM solved representation.

In section 2, we present an overview of diffusion curves (without blur) and biharmonic diffusion curves. Section 3 provides a description of our boundary element approach to rendering biharmonic diffusion curves, including line-by-line evaluation and curve-aware upsampling. Section 4 reports the performance of our implementations. Section 5 provides discussion of our results and section 6 is our conclusion.

2 Biharmonic Diffusion Curves

Mathematical description of image models In the diffusion curves image model, following [Jeschke et al. 2009], each of the R, G and B colour components of an image are solutions of the Laplace equation with value constraints at curves. In the biharmonic diffusion curves image model, following [Boyé et al. 2012], the colour component images are solutions of the homogeneous biharmonic equation with value and normal gradient constraints at curves. Table 1 provides a summary of features of these image models. The Laplace equation (1) is the governing partial differential equation (PDE) for diffusion curves; while the homogeneous biharmonic equation (2), in which the Laplace operator is applied twice, is the governing PDE for biharmonic diffusion curves. The fundamental solution (i.e. free-space Green's function) of the 2D Laplace equation is the logarithmic potential function of equation (3). The fundamental solution of the 2D homogeneous biharmonic equation is the thin plate spline function of equation (4).

Equations (5) and (6) give formulae for any solution of the 2D Laplace equation and the homogeneous biharmonic equation, for a bounded domain in the plane, in terms of the fundamental solutions (3), (4) and their directional derivatives. These equations are simplified version of equations given in [Mai-Duy and Tanner 2005]. In these equations, Γ represents the totality of curves bounding the domain, which includes an all-enclosing boundary curve and may include internal boundary curves, P is a point in the plane, Q is a point on a boundary curve, $C(P)$ is a constant which is 1 for points inside the domain and 0 for points outside the domain, n is the outward normal to the boundary and r is the distance between P and Q. [Sun et al. 2012] use equation (5) as the basis for their boundary element method of rendering diffusion curve images. We use equation (6) as the basis for the boundary element method we describe in this paper for rendering biharmonic diffusion curve images. [Weber et al. 2012] also derive equation (6) (for the interior of a region), referring to it as the boundary integral identity for the biharmonic equation. They use it to define coordinates within the region for controlling shape deformation. In contrast, for diffusion curve images we use equation (6) inside and outside the domain.

When there are nested closed curves and 2-sided open curves within the domain, versions of equations (5) and (6) apply in which the values of the solutions U, F and their normal derivatives along boundaries become 2-sided "jump" values. Consequently, any diffusion curves image and any biharmonic diffusion curves image can be viewed as a superposition of "point source" fields (the fundamental solutions and their directional derivatives) which are positioned with the field source points along the curves and weighted according to jumps in the value, normal gradient, Laplacian and normal gradient of Laplacian of the image at the curve.

Uniqueness properties. The Laplace equation and the homogeneous biharmonic equation satisfy uniqueness properties. Any solution of the Laplace equation satisfying certain continuity requirements within a region is uniquely determined by its values on its boundary [Kreyszig 1999]. Similarly, any solution of the homogeneous biharmonic equation satisfying certain continuity requirements within a region is uniquely determined by its values and its normal derivative values on its boundary [Jaswon and Symm 1977, p. 114]. In the context of diffusion curve images, these uniqueness properties are important for two reasons.

Firstly, they allow an image designer to isolate or preserve parts of their artwork as they continue to edit other parts of their artwork. Secondly, they allow image rendering methods to achieve significant efficiencies by independently processing different regions of the image. For diffusion curve images, curves inside a closed curve can be ignored when rendering outside the curve, and curves outside the closed curve can be ignored when rendering inside the curve. [Sun et al. 2012] use this region independence to perform curve "culling" in their diffusion curve image rendering and they report high speed gains when it is applied in rendering particular test images. Similarly, the uniqueness property of the homogeneous biharmonic equation allows optimisations in rendering biharmonic diffusion curve images.

Biharmonic diffusion curve images. Defining diffusion curve images as solutions of the homogeneous biharmonic equation brings the benefits of a long history of study of the biharmonic equation, a variety of solving methods, and it gives well-defined images which facilitates interoperability with other image models. We note that any image of the diffusion curves (without blur) image model is also an image of the biharmonic diffusion curves image model.

3 BEM biharmonic diffusion curves rendering

3.1 Solved boundary element representation

The following formula, corresponding to equation (6), gives a discrete approximation of the values of a colour channel of a biharmonic diffusion curves image.

$$im(x,y) = \sum_i d_i D_i(x,y) + \sum_i l_i L_i(x,y) + \sum_i s_i S_i(x,y) + \sum_i t_i T_i(x,y) \tag{7}$$

Table 1: *Summary of features of the diffusion curves and biharmonic diffusion curves image models.*

	Diffusion curves	**Biharmonic diffusion curves**
PDE	$\nabla^2 U = 0$ (1)	$\nabla^4 F = \nabla^2(\nabla^2 F(x,y)) = 0$ (2)
fundamental solution	$G^H(x,y) = \frac{1}{2\pi}\ln\left(\frac{1}{r}\right)$ (3)	$G^B(x,y) = \frac{1}{8\pi} r^2 \left[\ln\left(\frac{1}{r}\right)+1\right]$ (4)
boundary constraints	values	values and normal gradient values
solution inside a bounded domain	$C(P)U(P) = \int_\Gamma G^H(P,Q)\frac{\partial U(Q)}{\partial n} d\Gamma - \int_\Gamma \frac{\partial G^H(P,Q)}{\partial n} U(Q) d\Gamma$ (5)	$C(P)F(P) = \int_\Gamma G^H(P,Q)\frac{\partial F(Q)}{\partial n} d\Gamma - \int_\Gamma \frac{\partial G^H(P,Q)}{\partial n} F(Q) d\Gamma + \int_\Gamma G^B(P,Q)\frac{\partial U(Q)}{\partial n} d\Gamma - \int_\Gamma \frac{\partial G^B(P,Q)}{\partial n} U(Q) d\Gamma$ (6) where $U(x,y) = \nabla^2 F(x,y)$
uniqueness property	Inside a region bounded by piecewise smooth curves, the solution is uniquely determined by values on the curves.	Inside a region bounded by piecewise smooth curves, the solution is uniquely determined by values and normal gradients on the curves.

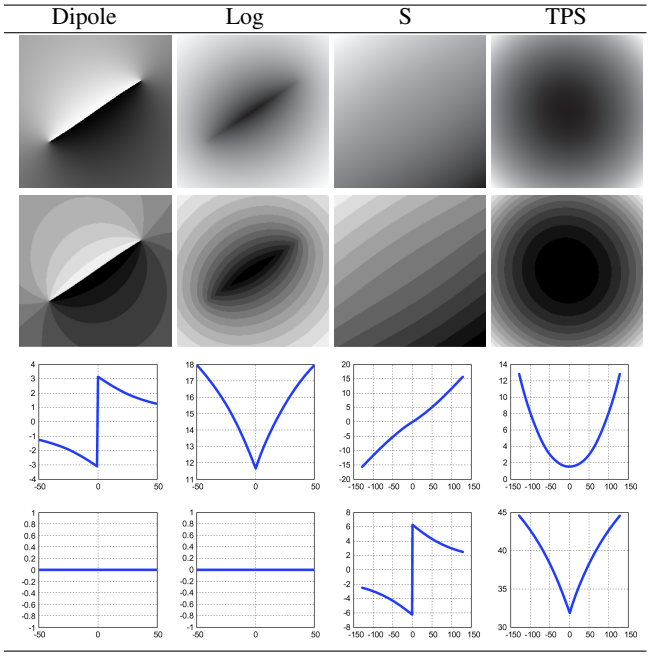

Figure 2: *The four line segment field types of the biharmonic diffusion curves image model, shown for an example line segment. Top (1st) row from left to right: dipole (D), logarithmic (L), derivative of thin plate spline (S), thin plate spline (T) line segment fields. 2nd row: quantized fields to show contours. 3rd row: plot of image values along the straight line which is normal to the line segment and passes through the centre of the line segment. 4th row: plot of image Laplacian values along the same straight line normal to the line segment.*

In this formula, the four summations are performed over straight line segments which approximate the curves of the image, including the outer curve enclosing the image region. The four summations correspond to the four integrals of equation (6). Each subscripted lower case value is a line segment weight; and each subscripted upper case function is a line segment field function. Each line segment field is the integral along a line segment of a unit-weighted point field function positioned along the segment. The four point field functions, which correspond to the fundamental solutions of equations (3), (4) and their directional derivatives, are:

1. the 2D dipole potential field (derivative of the 2D logarithmic field), oriented normal to the curve;
2. the 2D logarithmic potential field;
3. the derivative of the 2D thin plate spline, oriented normal to the curve;
4. the 2D thin plate spline.

Figure 2 shows images of the four types of line segment fields: dipole (D), logarithmic (L), derivative of thin plate spline (S) and thin plate spline (T). The dipole and logarithmic line segment fields are harmonic at points away from the line segment. Equation (7) with only the first two terms, i.e. with only dipole and logarithmic line segment fields, is a boundary element representation of diffusion curves without blur, similar to the solved representation of [Sun et al. 2012]. The S and thin plate spline segment fields are biharmonic. In equation (7), the presence of the thin plate spline segment field enables smooth interpolation of colour constraints, while the presence of the S segment field enables use of gradient constraints normal to curves.

Table 2 gives formulae for the four types of line segment fields and the point fields from which they are derived. For a directed line segment running from endpoint (x_1, y_1) to endpoint (x_2, y_2) with angle of inclination $\lambda = atan2(y_2 - y_1, x_2 - x_1)$, the line segment field formulae give values at (x, y) using variables u_1, u_2, v dependent on x, y according to equations (8). The variables u_1, u_2, v are the components of the two vectors from the line segment endpoints to a field point (x, y) in a (u, v) co-ordinate system rotated so that the u axis is oriented in the same direction as the line segment. This rotation aligns the branch cut of atan2(v, u) with the segment, so that the angular quantity atan2(v, u_2)−atan2(v, u_1) in the segment field formulae is a continuous function of the field point except across the line segment.

$$\begin{aligned} u_1 &= \cos(\lambda)(x - x_1) + \sin(\lambda)(y - y_1) \\ u_2 &= \cos(\lambda)(x - x_2) + \sin(\lambda)(y - y_2) \\ v &= -\sin(\lambda)(x - x_2) + \cos(\lambda)(y - y_2) \\ &= -\sin(\lambda)(x - x_1) + \cos(\lambda)(y - y_1) \end{aligned} \quad (8)$$

Table 2: *Formulae for the four line segment fields of biharmonic diffusion curves.*

	Point field	Line segment field
Dipole (D)	$d(x,y,\lambda)=\frac{\cos(\lambda)y-\sin(\lambda)x}{x^2+y^2}$	$D(x,y)=\text{atan2}(v,u_2)-\text{atan2}(v,u_1)$
Logarithmic (L)	$l(x,y)=\frac{1}{2}\log(x^2+y^2)$	$L(x,y)=-\left(u_2\left[\frac{1}{2}\log({u_2}^2+v^2)-1\right]-v\text{atan2}(v,u_2)\right)$ $+\left(u_1\left[\frac{1}{2}\log({u_1}^2+v^2)-1\right]-v\text{atan2}(v,u_1)\right)$
Derivative of thin plate spline (S)	$s(x,y,\lambda)=(\cos(\lambda)y-\sin(\lambda)x)$ $\left[\frac{1}{2}\log(x^2+y^2)-\frac{1}{2}\right]$	$S(x,y)=-\left(u_2v\left[\frac{1}{2}\log({u_2}^2+v^2)-\frac{3}{2}\right]-v^2\text{atan2}(v,u_2)\right)$ $+\left(u_1v\left[\frac{1}{2}\log({u_1}^2+v^2)-\frac{3}{2}\right]-v^2\text{atan2}(v,u_1)\right)$
Thin plate spline (T)	$t(x,y)=\frac{x^2+y^2}{2}$ $\left[\frac{1}{2}\log(x^2+y^2)-1\right]$	$T(x,y)=-\frac{1}{6}\left(({u_2}^3+3u_2v^2)\left[\frac{1}{2}\log({u_2}^2+v^2)-1\right]-2v^3\text{atan2}(v,u_2)-\frac{1}{3}{u_2}^3-2u_2v^2\right)$ $+\frac{1}{6}\left(({u_1}^3+3u_1v^2)\left[\frac{1}{2}\log({u_1}^2+v^2)-1\right]-2v^3\text{atan2}(v,u_1)-\frac{1}{3}{u_1}^3-2u_1v^2\right)$

The third row of Figure 2 shows for each type of line segment a plot of image values along the straight line oriented 90 degrees anti-clockwise to the segment and passing through the centre of the segment. The dipole and logarithmic segment fields have sharp profiles, while the S and TPS segment fields have smooth profiles. The fourth row of Figure 2 shows a plot of image Laplacian values along the same straight line. The profiles of S and TPS segment fields for Laplacian values have the same shape as the profiles of dipole and logarithmic segment field values. The dipole, logarithmic, S and TPS segment fields respectively contribute value, normal gradient, Laplacian and normal gradient of Laplacian discontinuities to a biharmonic diffusion curves image. Because each type of segment field contributes one type of discontinuity but no other, the four line segment field types can be considered to form an orthogonal set.

3.2 Specification of biharmonic diffusion curves

Sharp-profile / smooth-profile paradigm. In our implementation of biharmonic diffusion curves, we use a simple "sharp-profile" curve, "smooth-profile" curve paradigm for image designers to specify images. This is quite similar to the use of sharp-profile and smooth-profile curves in sketch-based surface modelling, such as in [Nealen et al. 2007], [Joshi and Carr 2008] and [Andrews et al. 2011]. Generally, a sharp-profile curve divides an image locally into independent left and right image regions, while a smooth-profile curve adjusts the colour variation of its (left and right) surrounding region. It is expected that an image designer would typically sketch outlines of regions with sharp-profile curves and adjust the colouring within regions using smooth-profile curves. However, the profile type of a curve is editable as are its associated constraint values. A sharp-profile curve has colour value constraints on both its left and right sides (the left and right colours may be the same). A sharp-profile curve is also given a Laplacian constraint: zero Laplacian at the curve. This means that colour surfaces near a sharp-profile curve have the same behaviour in the biharmonic diffusion curves image model as in the diffusion curves image model. As a consequence, a region bounded by sharp-profile closed curves, is independent of colours in the surrounding regions. This region independence is sensible default behaviour, because often sharp-profile curves occur as outlines of objects. A smooth-profile curve has constraints that apply to both sides of the curve: colour constraints along the curve, which are smoothly interpolated by surrounding colours; and optionally colour normal gradient constraints along the curve. We note that omitting normal gradient constraints gives a simpler image model, which still provides smooth interpolation of colour constraints.

The combination of diffusion curves with sharp-profiles, smooth-profiles, the option to set constraint values by sampling a work-in-progress rendered image, along with conventional image authoring schemes such as layering and transparency, provides a reasonably simple, but effective tool set for specifying colour variation. The solved boundary element representation of a biharmonic diffusion curve image of equation (7) is a general representation for solutions of the biharmonic equation with value and gradient constraints and as such is capable of representing images specified by all the curve constraint types described in [Boyé et al. 2012]. However, our specification of biharmonic diffusion curves images using sharp-profile and smooth-profile curves as above does not include the full range of these curve constraint types. Currently we do not implement normal gradient constraints at sharp-profile curves, smooth interpolation at isolated points and "barrier" curves (proposed by [Bezerra et al. 2010]), which would all be very useful for authoring. Nor do we implement contraint curves at which no colour value is specified, such as "tear", "crease" and "slope" curves. Implementation of additional curve constraint types is an area of future work for us.

3.3 Overview of rendering steps

There are two main steps in our diffusion curves image rendering.

1. Generate a solved boundary element image representation:
   (a) generate polyline segments from the curves;
   (b) determine dipole segment weights;
   (c) set up and solve a linear system of equations to obtain log, S and TPS segment weights.
2. Evaluate the output image from the solved representation (segments and segment weights).

The output image can be evaluated using direct evaluation of equation (7) or using line-by-line processing; in each case, curve-aware upsampling can be employed to reduce the computation.

3.4 Generating the solved BEM representation

Curves to polylines. Our current implementation renders a biharmonic diffusion curves image from a text file image specification, including curves, sharp-profile / smooth-profile curve attributes, and colour values and colour gradient values at positions along curves. We implement curves as cubic Bézier curves. We use recursive subdivision to convert cubic Bézier curves to line segments. The recursion terminates when a straightness threshold is reached. Subsequently we divide long segments so that segments do not exceed a maximum length threshold. Constraints for line segments, being colour values and colour normal gradient values at the centre of line segments, are derived from the curve constraints of the image specification by interpolating values along curves.

Dipole field weights. Weights for line segment dipole fields for a colour channel are equal to the difference of colour channel values on the left and right sides of the line segment, divided by 2π.

Linear system of equations. The weights for line segment logarithmic, S and thin plate spline fields for a colour channel are obtained by solving a linear system of equations consisting of three types of equations (9), (10), (11). In these equations the summations are performed over all line segments.

$$\sum_i (l_i L_i + s_i S_i + t_i T_i)(x,y) = c - \sum_i d_i D_i(x,y) \quad (9)$$

$$\sum_i \Big(s_i \nabla^2 S_i + t_i \nabla^2 T_i \Big)(x,y) = 0 \quad (10)$$

$$\sum_i \Big(l_i \frac{\partial L_i}{\partial n} + s_i \frac{\partial S_i}{\partial n} + t_i \frac{\partial T_i}{\partial n} \Big)(x,y) = \frac{\partial c}{\partial n} - \sum_i d_i \frac{\partial D_i}{\partial n}(x,y) \quad (11)$$

Let s be the number of sharp-profile segments, m be the number of smooth-profile segments and g be the number of smooth-profile segments having a normal gradient constraint. The linear system includes firstly $s+m$ equations of the form of equation (9) which constrain the colour channel value c at the centre of a line segment (x,y). For a sharp-profile line segment the colour channel value is the average of the segment left and right colour channel values. Secondly, there are s equations of the form of equation (10) constraining the Laplacian of colour channel values to be zero at the centre of line segments of sharp-profile curves. And thirdly, there are g equations of the form of equation (11) constraining the gradient of colour channel values at the centre of a line segment in the direction normal to the segment, for curves for which a normal gradient has been specified. This gives a total of $2s+m+g$ equations in the system. For smooth-profile curves, weights for logarithmic segment fields are set to zero (m weights); and for sharp-profile curves and smooth-profile curves with no normal gradient specified, weights for S segment fields are set to zero ($s+m-g$ weights). This leaves $3*(s+m)-m-(s+m-g) = 2s+m+g$ weights. In this way, the linear system has the same number of unknowns as constraint equations, and it is solved to provide, alongside the weights set to zero, a full set of log, S and TPS weights.

3.5 Line-by-line evaluation

Rendering a biharmonic diffusion curves image by direct evaluation of equation (7) requires calculating values for all segment fields at all image positions. Here we describe our line-by-line method for evaluating biharmonic diffusion curve images, which is a much more efficient alternative when rendering large images with many curves. The line-by-line evaluation method calculates a close approximation of a biharmonic diffusion curves image equal to the sum of approximate dipole, log, S and TPS segment fields, as in equation (12). Formulae for the approximate segment fields, as sums of periodic and polynomial functions, are provided in Table 4 in the appendix, along with a description of the derivation of the formulae.

$$\begin{aligned} i\tilde{m}(x,y) = &\sum_i d_i \tilde{D}_i(x,y) + \sum_i l_i \tilde{L}_i(x,y) + \\ &\sum_i s_i \tilde{S}_i(x,y) + \sum_i t_i \tilde{T}_i(x,y) \end{aligned} \quad (12)$$

Before describing in detail image evaluation according to equation (12), we introduce principles of line-by-line evaluation of biharmonic functions using repeated 1D convolution operations.

Line-by-line evaluation of biharmonic functions. The following Poisson (or Schwarz) integral formula [Brown and Churchill 1996] describes the harmonic extension of a function from its values on the x axis to the upper half plane, $y > 0$.

$$U(x,y) = U(x,0) * \frac{1}{\pi} \frac{y}{x^2+y^2} \quad (13)$$

According to this formula, values of a harmonic function on a horizontal line above the x axis are obtained from values of the function on the x axis by convolution with the Cauchy distribution (i.e. function) with scale parameter y. The Cauchy distribution is a bell-shaped curve. It has the property, verified by considering its Fourier Transform, that it is closed under self-convolution. In particular, the Cauchy distribution with an integer scale parameter n, is equal to the result of convolving together n copies of the standard Cauchy distribution (having scale parameter 1) as expressed by the following equation.

$$\frac{1}{\pi} \frac{n}{n^2+x^2} = \left(\frac{1}{\pi} \frac{1}{1+x^2} \right)^{*n} \equiv \frac{1}{\pi} \frac{1}{1+x^2} \overbrace{\left(* \frac{1}{\pi} \frac{1}{1+x^2} \right)}^{n-1 \text{ times}} \quad (14)$$

Intuitively, in a convolution operation, a Cauchy distribution acts as a low pass or "spreading" filter. Repeated convolution with a Cauchy distribution progressively smooths the (1D) function on which it operates. Combining equations (13) and (14) to give equation (15), shows that values of a harmonic function on a horizontal line can be used to successively obtain values of the harmonic function on uniformly spaced horizontal lines with greater y intercept, by repeated convolution with the same filter function[1]. Also, any 2D biharmonic function can be expressed according to equation (16), as a sum of a first harmonic function and y times a second harmonic function [Poritsky 1946, p. 253].

$$U(x,y+n) = U(x,y) \left(* \frac{1}{\pi} \frac{1}{1+x^2} \right)^{*n} \quad (15)$$

$$F(x,y) = U_1(x,y) + yU_2(x,y) \quad (16)$$

Equations (15) and (16) indicate the possibility for biharmonic functions, and hence biharmonic diffusion curve images, of taking values on one line parallel to the x axis and performing 1D filtering operations to calculate values on the next line, and then repeating this process to calculate values on subsequent lines. Specifically,

[1] This processing scheme has been used by the first author in algorithms for digital halftoning and mixed content image compression [Ilbery 2008].

consider the initialization expressed by equations (17) of two univariate functions, $line_1$ and $line_2$ using values of the bivariate harmonic functions U_1 and U_2 along a horizontal line, and the set of operations on the two univariate functions expressed by equations (18). After repeating the set of operations n times we have the result given by equation (19). That is, values of the biharmonic function, $F(x, y)$ are obtained on subsequent lines.

$$\begin{aligned} line_1(x) &= U_1(x, y) + yU_2(x, y) \\ line_2(x) &= U_2(x, y) \end{aligned} \tag{17}$$

$$\begin{aligned} line_1(x) &:= line_1(x) * \frac{1}{\pi}\frac{1}{1+x^2} \\ line_2(x) &:= line_2(x) * \frac{1}{\pi}\frac{1}{1+x^2} \\ line_1(x) &:= line_1(x) + line_2(x) \end{aligned} \tag{18}$$

$$line_1(x) = U_1(x, y+n) + (y+n)U_2(x, y+n) \tag{19}$$

We use this processing scheme in our line-by-line method of evaluating biharmonic diffusion curve images. However, rather than working with continuous functions we operate on discrete functions and use cyclic (i.e. periodic) convolution. Cyclic convolution is desirable because non-cyclic convolution spreads a signal wider so that repeated non-cyclic convolution on finite length signals requires special processing to prevent signal loss, which is liable to distort the signal. We next discuss formulae expressing approximate line segment fields using harmonic and biharmonic 1-D periodic functions, suitable for line-by-line evaluation.

Line segment fields as sums of 1D-periodic functions and polynomial functions. In Table 4 in the appendix, the top formula for each approximate segment field is a summary formula. E.g.the summary formula for the approximate dipole segment field is:

$$\begin{aligned} \tilde{D}(x, y) = {}& {}_X D(x - x_2, y - y_2) - {}_X D(x - x_1, y - y_1) \\ & + {}_y D(y) + {}_{xy} \tilde{D}(x, y) \end{aligned} \tag{20}$$

The summary formulae for approximate log, S and TPS segment fields have the same structure. As seen in Table 4, the first two terms of each approximate segment field formula (indicated by the prefix "X") are composed of functions which are periodic in the x dimension. The third term is a line adjustment term (prefix "y") composed of polynomials in y, and the fourth term is a bivariate polynomial term (prefix "xy") which is composed of bivariate polynomials in x and y. In calculating a diffusion curves image according to equation (12), we evaluate the periodic terms line-by-line using cyclic convolution, while the remaining polynomial terms are evaluated by conventional means.

The periodic terms of the approximate segment fields are composed from real and imaginary parts of polylogarithms of complex exponentials. The polylogarithm function of order n is defined by equation (21). For $y \geq 0$, real and imaginary components of the polylogarithms of the complex exponential e^{i2z} are given by equation (22). Now, as well as being periodic in x, the polylogarithm of complex exponential functions extend harmonically to the upper half plane [Lerma 2002] as described by equation (13). Consequently, values of the periodic terms of the dipole and logarithmic segment fields are harmonic, and the values of the periodic terms of the S and TPS segment fields, which include polylogarithms multiplied by y, are biharmonic.

$$Li_n(z) = \sum_{m=1}^{\infty} \frac{z^m}{m^n} \tag{21}$$

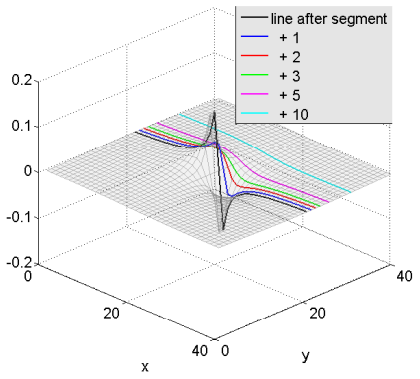

Figure 3: *Line plots of a dipole segment field, showing the shape of the plots smoothing from line to line.*

$$\begin{aligned} \text{Re}\left[Li_n\left(e^{i2z}\right)\right] &= \sum_{m=1}^{\infty} \frac{\cos(2mx)}{m^n} e^{-2my} \\ \text{Im}\left[Li_n\left(e^{i2z}\right)\right] &= \sum_{m=1}^{\infty} \frac{\sin(2mx)}{m^n} e^{-2my} \end{aligned} \tag{22}$$

In our line-by-line processing, we perform cyclic convolution with the discrete periodized Cauchy distribution given in the appendix. Figure 3 is a graph of an approximate dipole segment field for lines of constant y value. This segment field is calculated using the imaginary part of the order 1 polylogarithm of complex exponential, which at $y = 0$ has a "saw tooth" or periodic triangular shape. In the line-by-line method, the smoothed values of the approximate dipole segment field and other types of segment fields along lines of constant y value are successively obtained using field values along the preceding line closer to the segment by convolution with the periodized Cauchy distribution acting as a spreading filter. We now discuss evaluation of approximate line segment field formulae using an upward and downward sweep over image, before describing our implementation of line-by-line evaluation.

Sub-segments and upward and downward passes. The formulae of Table 4 are valid when $y \geq y_1, y_2$ (the y co-ordinates of the segment end points). That is, the formulae are valid in an upper half-plane above the segment. The formulae are also valid when $y \leq y_1, y_2$, i.e. in a lower half-plane below the segment. In our line-by-line method, each segment is split into sub-segments, such that each sub-segment lies in a region bounded by adjacent rows of pixel centres. Values for sub-segment fields can then be evaluated using the formulae on *all* rows of pixel centres and the values of a segment field are obtained as the sum of the values of the constituent sub-segment fields.

From inspection of the approximate segment field formulae of Table 4 it can be seen that the periodic and line adjustment terms contain a sign term s, but the bivariate polynomial terms do not. The sign term is ± 1 depending on whether y is above or below the segment. Because bivariate polynomial terms are the same above or below a segment, their contribution to image values in equation (12) can be calculated simply by summing the contribution of each segment to the coefficient of each polynomial $x^i y^j$ and then evaluating the polynomials at image positions using the summed coefficients.

In distinction, because the formulae for the periodic and line adjustment terms are different above and below a line segment, we calculate contributions from these terms in two parts - in an upward pass over image scanlines and a downward pass over image scan-

```
im = zeros(Yim,Xim);        % Xim = width, Yim = height
[im,coeffs] = do_pass(im,Xim,1,   1,Yim,segs);  % up
[im]        = do_pass(im,Xim,Yim,-1,  1,segs);  % down
for y=1:Yim                 % add polynomial values
  [c0,c1,c2,c3,c4] = x_coeffs_from_coeffs(y,coeffs);
  for x=1:Xim
    im(y,x) = im(y,x)+c0+c1*x+c2*x^2+c3*x^3+c4*x^4;
  end
end

function [im,coeffs] =
         do_pass(im,Xim,y_start,y_step,y_end,segs)
line1 = zeros(1,X);         % X >= 2*Xim; line1, line2
line2 = zeros(1,X);         % are scanline data stores
sys_vars = init_dsys();     % init. discrete systems
coeffs = init_coeffs();     % init. polynomial coeffs

for y=y_start:y_step:y_end
  contribs1 = zeros(1,X);
  contribs2 = zeros(1,X);
  for each sub-segment in image region (y+y_step,y]
    contribs1 = add_seg_contribs1(contribs1,sub_seg);
    contribs2 = add_seg_contribs2(contribs2,sub_seg);
    sys_vars = update_dsys_with_seg(sys_vars,sub_seg);
    if y_step==1
      coeffs = add_seg_coeffs(coeffs,sub_seg); end
  end
  line1 = line1 + contribs1;
  line2 = line2 + contribs2;

  line_adjust = adj_from_dsys(sys_vars);
  for x=1:Xim               % add pass image values
     im(y,x) = im(y,x) + line1(x) + line_adjust;
  end

  % updates for step in y
  line1 = cconv(line1,h,X); % cyclic convolution
  line2 = cconv(line2,h,X); % with spreading filter, h
  line1 = line1 + line2;
  sys_vars = time_step_dsys(sys_vars);
end
```

Figure 4: *Pseudo-code (pseudo-Matlab) for line-by-line evaluation of a single colour plane biharmonic diffusion curves image.*

lines. In the upward pass, upper half-plane values for the periodic terms are calculated in a similar fashion to the line-by-line processing scheme of equations (17) and (18). However, rather than calculating values for a single biharmonic function, sums of values of biharmonic functions are calculated, being upper half-plane values for all the periodic terms of the fields of all sub-segments preceding the current scanline. Lower half-plane values are calculated in a similar way in the downward pass. Values for line adjustment terms are also calculated using the upward pass and downward pass over image scanlines but using discrete time systems described in the appendix.

Line-by-line evaluation implementation. We now describe an overview of our line-by-line method of evaluating biharmonic diffusion curves for a single colour plane image, referring to the pseudo-code of Figure 4 (a colour image is evaluated by separately evaluating each colour plane). Please note that our implementation of line-by-line evaluation includes optimisations not described in the pseudo-code but discussed later. The input for the evaluation is a set of segments, each with weights for dipole, log, S and TPS segment fields. The image values are calculated according to the periodic, line adjustment and bivariate polynomial terms of the summary segment field formulae of Table 4, using sub-segments. At the start of the pseudo-code, the image is calculated by first setting image values to zero, then adding pass image values generated by an upward pass over image scanlines, then adding pass image values generated by a downward pass over image scanlines and finally adding polynomial values corresponding to the bivariate polynomial terms of Table 4.

In the upward pass over image scanlines, the pass image values are the sum of biharmonic function values corresponding to the periodic terms of Table 4, and line adjustment values corresponding to the line adjustment terms of Table 4. The upward pass image values are calculated for successive lines of the image by proceeding from the lowest y coordinate value to the highest; the values correspond to all sub-segments preceding (i.e below) the line. Similarly in the downward pass, downward pass image values are calculated for successive lines of the image, proceeding from the highest y coordinate value to the lowest, as sums of values corresponding to all sub-segments preceding (i.e. above) the line.

The upward pass and downward pass over image scanlines are implemented as calls to the pseudo-code function `do_pass`, with the parameter y_step set respectively to 1 and -1. At the start of the pass, the two line stores, line1 and line2 (of length X greater than or equal to twice the image width), are zero initialised; and the discrete time system variables and the bivariate polynomial coefficients are zero initialised using calls to pseudo-code functions `init_dsys` and `init_coeffs`. Within each pass, for each scanline, each sub-segment lying in the region between the pixel centres of the current scanline and the pixel centres of the preceding scanline is processed as follows. Firstly, contributions from periodic functions for the sub-segment are accumulated in line stores contribs1 and contribs2. The sub-segment contributions added to contribs1 (`add_seg_contribs1`) are values along the scanline for each of the polylogarithm terms of Table 4, multiplied by the appropriate segment weight. The sub-segment contributions added to contribs2 (`add_seg_contribs2`) are segment weighted values along the scanline for those polylogarithm terms of Table 4 which are multiplied by y (occurring in the S and TPS fields), except that rather than being multiplied by y, they are multiplied by the distance between scanlines. Secondly, the discrete system variables are updated for the sub-segment (`update_dsys_with_seg`). Also, if the pass is upward, the sums of bivariate polynomial coefficients are updated for the sub-segment (`add_seg_coeffs`).

Once all sub-segments for the scanline have been processed, the line stores, line1 and line2, are updated with the periodic function contributions, contribs1 and contribs2 similar to equations (17). At this point, line1 stores sums of biharmonic function values for all sub-segments preceding the current scanline. These values plus the line adjustment value calculated from the discrete time system variables (`adj_from_dsys`) constitute the pass values of the current scanline and are added to the image. Then the line stores are prepared for the next scanline by performing cyclic convolution with the spreading filter (the periodic Cauchy kernel) and adding line2 values to line1 values, similar to equations (18), and the discrete system variables are updated, corresponding to a time step of 1 (`time_step_dsys`), to prepare for the next scanline.

Following the upward and downward passes, polynomial values corresponding to the bivariate polynomial coefficients obtained in the upward pass, are added to the image scanline by scanline, as polynomials in x, using pseudo-code function `x_coeffs_from_coeffs` to determine the coefficients of the polynomials in x for the current scanline. In this way, the weighted values of all sub-segment fields from the periodic, line adjustment and bivariate polynomial terms are added to the image

Line-by-line evaluation optimisations and details. In our implementation we avoid calculation of polylogarithm values by using pre-calculated values stored in look up tables. Each sub-segment is approximated as a weighted sum of segments with end points having pixel centre co-ordinates. In this way, the look up tables only need to store polylogarithm values evaluated at uniformly sam-

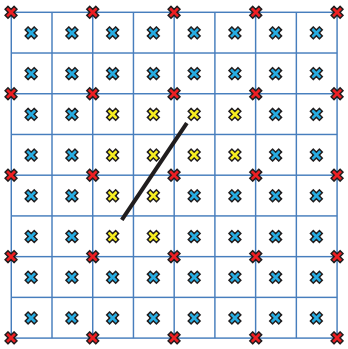

Figure 5: *Curve-aware upsampling. A line segment, shown in black, lies on a coarse grid of low resolution image pixels. Image values at a higher resolution are calculated from the coarse grid image values by upsampling. Correction image values are calculated for blocks of higher resolution pixels close to the line segment.*

Figure 6: *Diffusion curve images rendered using our implementation. Top row images are re-creations of images in [Orzan et al. 2008]. The duck and cat images are re-creations of images in [Finch et al. 2011] and [Boyé et al. 2012] respectively.*

pled positions in the x dimension along two horizontal lines, with y position equal to 0 and y position equal to the distance between scanlines. Instead of performing circular convolution operations in the spatial domain, we perform the convolutions using multiplications of frequency values. We transform to and from frequency values using power-of-2 Fast Fourier Transforms (FFTs). In using the approximate field formulae of Table 4, co-ordinate values are first scaled by π/X, where X is the polylogarithm period which is set greater than or equal to twice the image width.

3.6 Curve-aware upsampling

Line segment fields are increasingly smooth as the distance from the line segment increases. Consequently, the value of a line segment field at an image position can be calculated to a close approximation by interpolating line segment values at surrounding image positions, provided the region covered by the surrounding positions is not close to the line segment. With curve-aware upsampling, the image is calculated by upsampling a lower resolution image and adding correction values near each line segment.

Figure 5 illustrates one level of 2x2 symmetric upsampling, showing a coarse grid of low resolution pixel centres in red and a fine grid of blue or yellow high resolution pixel centres. A square image region with corners at the centres of four nearest neighbour lower resolution pixels (red crosses), is the same as the image region occupied by a 2x2 block of higher resolution pixels (blue or yellow crosses). For each line segment, correction values are calculated for blocks of higher resolution pixels that are close to the line segment (yellow crosses). For a line segment, the correction value at a higher resolution pixel is the sum of:

1. an upsampling compensation value which is the negative of the contribution by upsampling to the higher resolution pixel value from parts of lower resolution pixel values due to the line segment, and
2. a replacement value being the sum of line segment field values at the high resolution pixel due to the line segment, calculated according to the line segment field formulae of Table 2.

In this way, when lower resolution pixel values are exact, for an image value at a higher resolution pixel close to a line segment, the total contribution to the image value due to the line segment (i.e. the part of the upsampled value due to the line segment plus the correction value) is equal to the exact contribution to the image value from the line segment according to equation (7). This distinguishes curve-aware upsampling from "discontinuity-aware upsampling" described in [Finch et al. 2011] in which the effect of a close curve is estimated. Curve-aware upsampling can be performed multi-scale by successively taking a low resolution image to higher resolutions. However, all the curve-aware upsampling results reported in this paper are for one level of 4x4 symmetric upsampling using bicubic interpolation.

Table 3: *Execution times. All images are 512x512 RGB.*

image	no. of segments	execution times (secs)		
		solve	evaluate (direct +1)	evaluate (line-by-line)
pepper	1487	0.245	0.634	0.078
flower	3536	0.946	1.222	0.076
dolphin	3659	1.072	1.212	0.084
parrot	1012	0.279	0.501	0.119
pumpkin	1710	0.792	0.719	0.151
toys	1912	0.797	0.789	0.134
duck	2192	1.338	0.891	0.140
car	2611	2.200	0.999	0.126
cat	2755	2.020	1.028	0.161
lines	4241	8.052	1.457	0.147

4 Performance

Speed. Table 3 gives execution times for rendering the example images of Figure 1, Figure 6 and Figure 8. The top row of Figure 6 shows our rendering of diffusion curve images appearing in [Orzan et al. 2008]. These images are rendered using vector data from the authors of [Orzan et al. 2008], but without applying blur; with vector data for the dolphin image being cropped. Figure 6 also shows our rendering of the cat image appearing in [Boyé et al. 2012] and the duck image appearing in [Finch et al. 2011]. For our rendering of the cat image we have modified original data by adding colour values for curves at which colour was not specified and we have excluded or slightly moved curves to avoid having smooth-profile curves very close to other curves. The "lines" texture image of Figure 6 was prepared programmatically by placing randomly oriented straight lines separated from each other.

Table 3 shows times for generating the solved BEM representation using a GPU, and for two methods of evaluation from the solved representation. The first of these methods is direct evaluation with one level of curve-aware upsampling ("direct +1") using equations (7), (8) and the line segment field equations of Table 2, performed

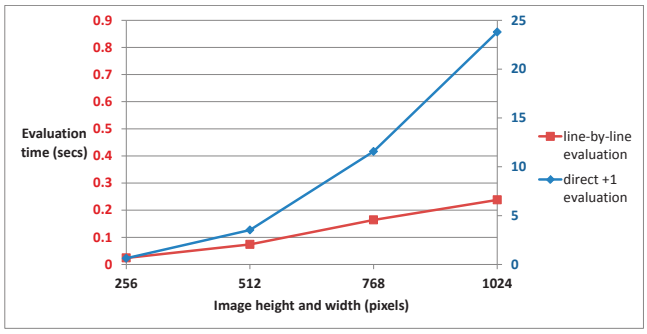

Figure 7: *Evaluation times by image size.*

using a GPU. The second method ("line-by-line") is line-by-line evaluation *on a CPU*. Times for the top row diffusion curve images of Figure 6 are listed separately to times for the other images, because the top row images, which conform to the diffusion curves without blur image model, do not include smooth-profile curves (i.e. no S and TPS segment fields) and as a consequence have comparatively smaller solve and line-by-line evaluation times. Our performance figures are for execution on a PC with an Intel Xeon E5630 2.53 GHz processor having 4 cores and with an NVIDIA Quadro 2000 graphics card with 1GB of video memory.

The computational complexity of direct evaluation of diffusion curve images according to equation (7) is $O(nm)$, while the computational complexity of the line-by-line evaluation algorithm is $O(nlogn) + O(m)$, where n is the number of pixels and m is the number of segments. Figure 7 shows times for direct evaluation with curve-aware upsampling on a CPU and line-by-line evaluation times on a CPU for a simple image as the image height and width are changed together. The direct evaluation with upsampling times increase approximately in proportion to the number of pixels times the number of segments. The increase in the line-by-line evaluation times is much more subdued, which is consistent with the computational load not jointly depending on the image size and the number of segments.

Accuracy. The top row of Figure 8 shows on the left a plot of a TPS segment field, and on the right, the difference between the field and its line-by-line evaluation using the formulae of Table 4. The approximation error, which in this case is at most approximately $5.1\text{x}10^{-4}$ of the maximum image field value, is primarily due to truncating polynomial series in the derivation of the approximate field formulae. The bottom row of Figure 8 shows on the left a parrot image rendered using line-by-line evaluation. On its right is a difference image obtained by subtracting it from the same image rendered using direct evaluation (with differences multiplied by 20 to enhance visibility). The direct evaluation is not anti-aliased, so there are significant differences along the sharp-profile curves of the image. Elsewhere in this image, differences of 8 bit colour values are at most one before multiplying. For each test image shown in this paper it is difficult to observe, except along sharp-profile curves, any difference between the image rendered line-by-line and the image rendered by direct evaluation with curve-aware upsampling.

In the line-by-line evaluation method, image approximation errors are primarily due to TPS segment fields. This is because TPS segment field weights can become large when smooth profile segments are close to each other or to other curves. This is understandable, because when smooth-profile curves are close to other curves, the value and gradient constraints specified along curves can cause high curvature, i.e. bending of the colour surfaces. In an animation

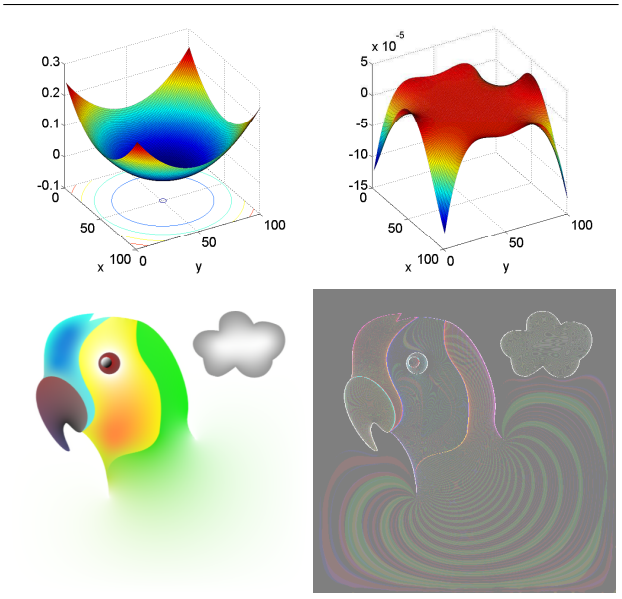

Figure 8: *Difference images. Top-left: TPS field for a segment from endpoint (50,50) to endpoint (52,51); top-right: TPS segment field minus the approximate field obtained using our line-by-line evaluation implementation; bottom-left: parrot biharmonic diffusion curves image ©CiSRA; bottom-right: parrot image rendered by direct evaluation minus image rendered line-by-line, with differences multiplied by 20.*

composed of rendered frames, the requirement for rendering accuracy is much higher than for individual images, because slight changes from one frame to the next can produce annoying flicker artefacts. For all the video we have produced using direct evaluation with curve-aware upsampling, flicker artefacts have been absent. In the video provided in the supplemental materials, rendered using our line-by-line evaluation method, image approximation errors are small enough so that the animations are smooth without flicker. However, it is easy to specify images with smooth-profile curves close to other curves, which give line-by-line evaluated video with significant flicker. In such cases it is expected that flicker should be alleviated by using approximate segment field formulae with less truncation of polynomial series. Another strategy to avoid flicker is to restrict image specification.

5 Discussion

Extreme constraints. Extreme constraints present difficulties for biharmonic diffusion curve images. Smoothly interpolated colour values very close to other constraints and arbitrarily large colour gradients can define colour surfaces which move rapidly out of range. With boundary element rendering of biharmonic diffusion curve images, extreme constraints can lead to very large TPS weights and to solving failure. For the image content presented in this paper, we have excluded smooth-profile curves which are very close to other curves as a way of side-stepping this issue.

Line-by-line evaluation. For the previously published boundary element methods of rendering diffusion curve images, [van de Gronde 2010] and [Sun et al. 2012], the computational load for evaluation from the solved representation is in general dependent on the product of the number of evaluation points (e.g. pixels) and the number of boundary elements. In our line-by-line evalu-

ation method, the computational load does not jointly depend on the image size and the number of boundary elements. The line-by-line evaluation method may be useful in efficiently evaluating solutions of PDEs for other applications. We expect to improve the performance of line-by-line-evaluation by reducing the use of FFTs through calculating segment contributions to line stores using infinite impulse response (IIR) filtering along scanlines. However we have not included this in our implementation yet.

Direct evaluation. Direct evaluation of equation (7) is suited to evaluation of small image regions; however, the computation is typically heavy for large image regions. Direct evaluation is suited to parallel execution and implementation on a GPU; and the computational load is significantly reduced by curve-aware upsampling. The ability to directly evaluate image values at arbitrary positions can be very useful, for colouring surfaces as in [Sun et al. 2012] and [Boyé et al. 2012], and for setting new curve constraint values during image authoring, discussed in section 3.2. Our implementations of direct evaluation and curve-aware upsampling are currently not antialiased. However, adding antialiasing should be straightforward.

Solving. The boundary element method requires solving a linear system of equations with a dense system matrix. This solving step is the bottleneck in our rendering implementation, being comparatively slow and $O(n^3)$. We hope to be able to address this. Solving performance can be significantly improved for particular images by separately solving for independent image regions. We have not implemented this yet. FEM solving times reported in [Boyé et al. 2012] are well below one second. [Boyé 2012, p. 119] indicates that this FEM solving exploits separate solving of independent regions. However, for some types of images, including the flower and dolphin images of Figure 6 which are obtained by automatic vectorization of photographs described in [Orzan et al. 2008], the number of curves per independent region can be high, in which case it is expected that separate solving of independent regions will be less effective in reducing the solving load. The solving times reported in [Finch et al. 2011] are also below our solving times, although that solving includes coarse-to-fine processing with acknowledged accuracy limitations.

Accuracy. The accuracy of a biharmonic diffusion curves image evaluated according to equation (7) using our BEM solved representation, depends only on how well curves are approximated by line segments and how well the image discontinuities at curves are modelled by line segment weights. The BEM solved representation, like the FEM solved representation of [Boyé et al. 2012], is vectorial; so there is no dependence on a pixel grid as in the finite difference method of [Finch et al. 2011]. However, the BEM solved representation has no dependence on the density or quality of a 2D mesh as in [Boyé et al. 2012].

Compactness. The FEM solved representation for biharmonic diffusion curve images of [Boyé et al. 2012] has solved values at mesh points covering the 2D image space. Similarly to the previously published boundary element methods of rendering diffusion curve images, our solved representation contains solved values only along the 1D boundary curves. Table 4.1 of [Boyé 2012] reports the cat image of Figure 6 being rendered by the FEM solver using 49,731 nodes and 12,439 triangles. This indicates that the FEM solved representation would be about 5 times the size of the BEM solved representation for this RGB image. Considering the dimensionality of the solved representations, it is expected that the relative compactness of BEM over FEM would increase with image size.

Rendering from a compact solved representation. In our rendering, the highly compact BEM solved diffusion curves image representation can be pre-calculated, and times for our line-by-line evaluation from the solved representation are sub-second while only increasing slowly and linearly with the number of image curve segments. Thus line-by-line evaluation from the BEM solved representation may be attractive for rendering complex diffusion curve images.

6 Conclusion

We have presented a unified mathematical description of diffusion curves without blur and biharmonic diffusion curves, and a boundary element approach to rendering biharmonic diffusion curve images. Like the finite element approach, the boundary element approach has an image rendering advantage over the finite difference approach in producing a vectorial solved image representation independent of a pixel grid. However the boundary element solved representation is more compact than the finite element solved representation. We have also presented an efficient line by line calculation method and a curve-aware upsampling method for calculating diffusion curve images from the solved boundary element representation. The new approach to rendering diffusion curve images offers the possibility of transmitting compact solved image representations to devices for evaluation at the device, where the devices, due to time or processing constraints, are not suited to PDE solving.

7 Acknowledgements

We thank A. Orzan and co-authors, and S. Boyé and M. Finch for permission to use images from their papers. We also thank N. Andronikos and A. Chang for assistance in preparing images.

References

ANDREWS, J., JOSHI, P., AND CARR, N. 2011. A linear variational system for modeling from curves. *Computer Graphics Forum 30*, 6, 1850–1861.

BEZERRA, H., EISEMANN, E., DECARLO, D., AND THOLLOT, J. 2010. Diffusion constraints for vector graphics. *NPAR 2010: Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*.

BOYÉ, S., BARLA, P., AND GUENNEBAUD, G. 2012. A vectorial solver for free-form vector gradients. *Transaction on Graphics* (Nov.).

BOYÉ, S. 2012. *Représentation hybride pour la modélisation géométrique interactive*. PhD thesis, Université Sciences et Technologies-Bordeaux 1.

BROWN, J., AND CHURCHILL, R. 1996. *Complex variables and applications*. McGraw-Hill Inc.

FINCH, M., SNYDER, J., AND HOPPE, H. 2011. Freeform vector graphics with controlled thin-plate splines. *ACM Trans. Graph. 30*, 6 (Dec.), 166:1–166:10.

ILBERY, P. 2008. *Scanline calculation of radial influence for image processing*. PhD thesis, University of NSW.

JASWON, M. A., AND SYMM, G. 1977. *Integral equation methods in potential theory and elastostatics*. Academic Press Inc.

JESCHKE, S., CLINE, D., AND WONKA, P. 2009. A GPU Laplacian solver for diffusion curves and Poisson image editing. *Transaction on Graphics (Siggraph Asia 2009) 28*, 5 (Dec.), 1–8.

JOSHI, P., AND CARR, N. A. 2008. Repoussé: automatic inflation of 2D artwork. In *Proceedings of the Fifth Eurographics con-*

ference on Sketch-Based Interfaces and Modeling, Eurographics Association, SBM'08, 49–55.

KREYSZIG, E. 1999. *Advanced engineering mathematics (8th ed.)*. John Wiley.

LERMA, M., 2002. The Bernoulli periodic functions. `http://www.math.northwestern.edu/~mlerma/papers/bern_period_func.pdf`.

MAI-DUY, N., AND TANNER, R. 2005. An effective high order interpolation scheme in BIEM for biharmonic boundary value problems. *Engineering Analysis with Boundary Elements 29*, 3, 210 – 223.

NEALEN, A., IGARASHI, T., SORKINE, O., AND ALEXA, M. 2007. Fibermesh: designing freeform surfaces with 3D curves. *ACM Trans. Graph. 26*, 3 (July).

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*, 3 (Aug.), 92:1–92:8.

ORZAN, A., BOUSSEAU, A., WINNEMÖLLER, H., BARLA, P., THOLLOT, J., AND SALESIN, D. 2013. Diffusion curves: a vector representation for smooth-shaded images. *Commun. ACM 56*, 7 (Jul.), 101–108.

PORITSKY, H. 1946. Application of analytic functions to two-dimensional biharmonic analysis. *Transactions of the American Mathematical Society 59*, 2 (Mar.), 248–279.

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*, 4 (July), 74:1–74:9.

VAN DE GRONDE, J. 2010. *A High Quality Solver for Diffusion Curves*. Master's thesis, University of Groningen.

WEBER, O., PORANNE, R., AND GOTSMAN, C. 2012. Biharmonic coordinates. In *Computer Graphics Forum*, vol. 31, Wiley Online Library, 2409–2422.

Appendix

Discrete periodized Cauchy distribution

The spreading filter in our line-by-line evaluation method is the following periodic Cauchy convolution kernel, evaluated at $y = 1/X$ and $x \in \mathbb{Z}$. It is given with its Fourier Transform in the x dimension. In this formula X is the period of the periodic functions.

$$h(x) = \frac{1}{X} \frac{\sinh(2\pi y) - 2e^{-\pi y X} \begin{pmatrix} \sinh(\pi y)\cos(\pi x)\cos(\pi x/X) \\ -\cosh(\pi y)\sin(\pi x)\sin(\pi x/X) \end{pmatrix}}{\cosh(2\pi y) - \cos(2\pi x/X)}$$

$$\overset{x}{\longleftrightarrow} \qquad e^{-y\,|\,\omega\,|} \;\; \frac{2\pi}{X} \sum_{m\,=\,-\frac{X-1}{2}}^{m\,=\,\frac{X-1}{2}} \delta\left(\omega - m\tfrac{2\pi}{X}\right)$$

Derivation of approximate segment field formulae

Table 1 in the Supplemental Appendix gives formulae for each line segment field using a corresponding real function of the complex variable z. Each of the functions of z involves real or imaginary components of integrals of the complex function $1/z$.

From the Laurent series for cot(z), valid for $|z| < \pi$:

$$\frac{1}{z} = cot(z) + \frac{1}{3}z + \frac{1}{45}z^3 + \frac{1}{945}z^5 + \frac{1}{4725}z^7 + ...$$

From [Lerma 2002], for $y \geq 0$:

$$cot(\pi z) = -i[1 + 2Li_0(e^{i2\pi z})]$$

So we have

$$\frac{1}{z} = -i[1 + 2Li_0(e^{i2z})] + \frac{1}{3}z + \frac{1}{45}z^3 + \frac{1}{945}z^5 + \frac{1}{4725}z^7 + ...$$

Repeatedly integrating this analytic function and adding appropriate constants gives the formulae in Table 2 of the Supplemental Appendix, which express the integrals of $1/z$ as the sum of higher order polylogarithms of a complex exponential and sums of polynomials, including polynomial series. The approximate segment field formulae of Table 4 are then obtained using the following three steps. Firstly, the polynomial series of the integrals of $1/z$ being $log(z)$, $z[log(z)-1]$, $(z^2/2)[log(z)-3/2]$, $(z^3/6)[log(z)-11/6]$ are truncated to remove powers of z greater than 2, 3, 4 and 5 respectively. Secondly, these truncated formulae are used to replace terms which are integrals of $1/z$ in the segment field formulae of Table 1 in the Supplemental Appendix. Thirdly, the resulting segment field formulae are simplified after expanding the polynomials about the segment mid-point. The polynomial series truncation has the result that the bivariate polynomial terms (prefix "xy") in Table 4 are approximations as indicated by use of the tilde symbol.

Discrete-time linear systems

Discrete time linear systems are used to calculate values of line adjustment terms in Table 4. Consider the upward pass. Let $d = \lceil y_i \rceil - y_i$, where y_i is the y co-ordinate of the centre of the i'th sub-segment and $\lceil\,\rceil$ denotes the ceiling function. Then d satisfies $0 \leq d < 1$, and is the distance from the sub-segment centre to the nearest horizontal image line at or above the sub-segment centre. To calculate values of the line adjustment terms of Table 4, discrete-time linear systems are used which have impulse responses which match sampled continuous-time polynomial impulse responses.

For the input operations below, performed at integer time k corresponding to an impulse at time $k-d$, followed by the accumulating operations below,

input operations	accumulating operations
$in_0[k] \mathrel{+}= 1$	$A_0[k] = A_0[k-1] + in_0[k]$
$in_1[k] \mathrel{+}= d$	$A_{10}[k] = A_{10}[k-1] + in_1[k]$
$in_1[k+1] \mathrel{+}= 1-d$	$A_{11}[k] = A_{11}[k-1] + A_{10}[k]$
	$A_{12}[k] = A_{12}[k-1] + A_{11}[k]$
	$A_{13}[k] = A_{13}[k-1] + A_{12}[k]$
$in_2[k] \mathrel{+}= d^2 - d$	$A_2[k] = A_2[k-1] + in_2[k]$
$in_{31}[k] \mathrel{+}= d^3 - d$	$A_{31}[k] = A_{31}[k-1] + in_{31}[k]$
$in_{32}[k+1] \mathrel{+}= d^2 - d$	$A_{320}[k] = A_{320}[k-1] + in_{32}[k]$
	$A_{321}[k] = A_{321}[k-1] + A_{320}[k]$

the outputs below have the desired impulse responses.

output	impulse response
$A_0[k]$	$u[k]$
$A_{11}[k]$	$(k+d)u[k]$
$2A_{12}[k] - A_{11}[k] + A_2[k]$	$(k+d)^2u[k]$
$6A_{13}[k] - 6A_{12}[k] + A_{11}[k] + A_{31}[k] + 3A_{32}[k-1]$	$(k+d)^3u[k]$

Here $u[k]$ is the unit step sequence, satisfying $u[k] = 1$ for $k = 0, 1, 2, ..$ and $u[k] = 0$ otherwise.

Table 4: *Formulae for dipole, log, S and TPS segment fields as the sum of functions periodic in the x dimension, univariate line adjusment functions and bivariate polynomial functions. Each periodic function is expressed using real and imaginary components of the polylogarithm function of order n, $Li_n(z)$, with $n = 1, 2, 3, 4$. In these formulae, the line segment start and end points are (x_1, y_1) and (x_2, y_2), λ is the angle of inclination of the line segment and len is the length of the line segment. The formulae are valid for $y \geq y_1$, y_2 and for $y \leq y_1$, y_2 with $s = \text{sign}(y - \frac{y_1+y_2}{2})$. In the formula for ${}_{xy}\tilde{T}(x, y)$, $\zeta[3]$ is an evaluation of the Riemann zeta function.*

$$\tilde{\boldsymbol{D}}(\boldsymbol{x},\boldsymbol{y}) = {}_{\boldsymbol{X}}\boldsymbol{D}(\boldsymbol{x}-\boldsymbol{x_2},\boldsymbol{y}-\boldsymbol{y_2}) - {}_{\boldsymbol{X}}\boldsymbol{D}(\boldsymbol{x}-\boldsymbol{x_1},\boldsymbol{y}-\boldsymbol{y_1}) + {}_{\boldsymbol{y}}\boldsymbol{D}(\boldsymbol{y}) + {}_{\boldsymbol{xy}}\tilde{\boldsymbol{D}}(\boldsymbol{x},\boldsymbol{y})$$

$${}_{X}D(x,y) = -s\ \text{Im}\left[Li_1(e^{-2|y|+i2x})\right]$$

$${}_{y}D(y) = s\ len\ \cos(\lambda)$$

$${}_{xy}\tilde{D}(x,y) = len\ (-\frac{1}{3})\left(\sin(\lambda)\left(x-\frac{x_1+x_2}{2}\right)+\cos(\lambda)\left(y-\frac{y_1+y_2}{2}\right)\right)$$

$$\tilde{\boldsymbol{L}}(\boldsymbol{x},\boldsymbol{y}) = {}_{\boldsymbol{X}}\boldsymbol{L}(\boldsymbol{x}-\boldsymbol{x_2},\boldsymbol{y}-\boldsymbol{y_2}) - {}_{\boldsymbol{X}}\boldsymbol{L}(\boldsymbol{x}-\boldsymbol{x_1},\boldsymbol{y}-\boldsymbol{y_1}) + {}_{\boldsymbol{y}}\boldsymbol{L}(\boldsymbol{y}) + {}_{\boldsymbol{xy}}\tilde{\boldsymbol{L}}(\boldsymbol{x},\boldsymbol{y})$$

$${}_{X}L(x,y) = \cos(\lambda)\frac{1}{2}\text{Im}\left[Li_2(e^{-2|y|+i2x})\right] - s\ \sin(\lambda)\frac{1}{2}\text{Re}\left[Li_2(e^{-2|y|+i2x})\right]$$

$${}_{y}L(y) = s\ len\ \left(y-\frac{y_1+y_2}{2}\right)$$

$${}_{xy}\tilde{L}(x,y) = len\left(-\log(2)+\frac{1}{6}\left(\left(x-\frac{x_1+x_2}{2}\right)^2+\frac{(x_2-x_1)^2}{12}-\left(y-\frac{y_1+y_2}{2}\right)^2-\frac{(y_2-y_1)^2}{12}\right)\right)$$

$$\tilde{\boldsymbol{S}}(\boldsymbol{x},\boldsymbol{y}) = {}_{\boldsymbol{X}}\boldsymbol{S}(\boldsymbol{x}-\boldsymbol{x_2},\boldsymbol{y}-\boldsymbol{y_2}) - {}_{\boldsymbol{X}}\boldsymbol{S}(\boldsymbol{x}-\boldsymbol{x_1},\boldsymbol{y}-\boldsymbol{y_1}) + {}_{\boldsymbol{y}}\boldsymbol{S}(\boldsymbol{y}) + {}_{\boldsymbol{xy}}\tilde{\boldsymbol{S}}(\boldsymbol{x},\boldsymbol{y})$$

$${}_{X}S(x,y) = \sin(\lambda)\cos(\lambda)\frac{1}{2}\text{Re}\left[Li_3(e^{-2|y|+i2x})\right] + s\ \sin^2(\lambda)\frac{1}{2}\text{Im}\left[Li_3(e^{-2|y|+i2x})\right] + y\frac{1}{2}\text{Re}\left[Li_2(e^{-2|y|+i2x})\right]$$

$${}_{y}S(y) = s\ len\ \cos(\lambda)\left(\left(y-\frac{y_1+y_2}{2}\right)^2+\frac{(y_2-y_1)^2}{12}\right)$$

$$\begin{aligned}{}_{xy}\tilde{S}(x,y) = len\Bigg(&-\cos(\lambda)(\log(2)+\frac{1}{2})\left(y-\frac{y_1+y_2}{2}\right)+\sin(\lambda)(\log(2)-\frac{1}{2})\left(x-\frac{x_1+x_2}{2}\right)\\ &+\frac{1}{72}\Bigg(-4\sin(\lambda)\left(x-\frac{x_1+x_2}{2}\right)^3+12\cos(\lambda)\left(x-\frac{x_1+x_2}{2}\right)^2\left(y-\frac{y_1+y_2}{2}\right)\\ &-12\sin(\lambda)\left(x-\frac{x_1+x_2}{2}\right)\left(y-\frac{y_1+y_2}{2}\right)^2-12\cos(\lambda)\left(y-\frac{y_1+y_2}{2}\right)^3\\ &+\sin(\lambda)\left(x-\frac{x_1+x_2}{2}\right)\left((x_2-x_1)^2-(y_2-y_1)^2\right)+\cos(\lambda)\left(y-\frac{y_1+y_2}{2}\right)\left((x_2-x_1)^2-5(y_2-y_1)^2\right)\Bigg)\Bigg)\end{aligned}$$

$$\tilde{\boldsymbol{T}}(\boldsymbol{x},\boldsymbol{y}) = {}_{\boldsymbol{X}}\boldsymbol{T}(\boldsymbol{x}-\boldsymbol{x_2},\boldsymbol{y}-\boldsymbol{y_2}) - {}_{\boldsymbol{X}}\boldsymbol{T}(\boldsymbol{x}-\boldsymbol{x_1},\boldsymbol{y}-\boldsymbol{y_1}) + {}_{\boldsymbol{y}}\boldsymbol{T}(\boldsymbol{y}) + {}_{\boldsymbol{xy}}\tilde{\boldsymbol{T}}(\boldsymbol{x},\boldsymbol{y})$$

$$\begin{aligned}{}_{X}T(x,y) = {}& s\ 2\sin^3(\lambda)\frac{1}{8}\text{Re}\left[Li_4(e^{-2|y|+i2x})\right]-\cos(\lambda)\left(1+2\sin^2(\lambda)\right)\frac{1}{8}\text{Im}\left[Li_4(e^{-2|y|+i2x})\right]\\ &+\sin(\lambda)y\frac{1}{4}\text{Re}\left[Li_3(e^{-2|y|+i2x})\right]-s\ \cos(\lambda)y\frac{1}{4}\text{Im}\left[Li_3(e^{-2|y|+i2x})\right]\end{aligned}$$

$${}_{y}T(y) = s\ len\ \frac{1}{3}\left(\left(y-\frac{y_1+y_2}{2}\right)^3+\left(y-\frac{y_1+y_2}{2}\right)\frac{(y_2-y_1)^2}{4}\right)$$

$$\begin{aligned}{}_{xy}\tilde{T}(x,y) = len\Bigg(&-\frac{1}{4}\zeta[3]-\frac{1}{2}\left(\log(2)-\frac{1}{2}\right)\left(\left(x-\frac{x_1+x_2}{2}\right)^2+\frac{(x_2-x_1)^2}{12}\right)-\frac{1}{2}\left(\log(2)+\frac{1}{2}\right)\left(\left(y-\frac{y_1+y_2}{2}\right)^2+\frac{(y_2-y_1)^2}{12}\right)\\ &+\frac{1}{72}\Bigg(\left(x-\frac{x_1+x_2}{2}\right)^4+6\left(x-\frac{x_1+x_2}{2}\right)^2\left(y-\frac{y_1+y_2}{2}\right)^2-3\left(y-\frac{y_1+y_2}{2}\right)^4\\ &+\left(x-\frac{x_1+x_2}{2}\right)^2\left(\frac{(x_2-x_1)^2}{2}+\frac{(y_2-y_1)^2}{2}\right)+\left(y-\frac{y_1+y_2}{2}\right)^2\left(\frac{(x_2-x_1)^2}{2}-3\frac{(y_2-y_1)^2}{2}\right)\\ &+2\left(x-\frac{x_1+x_2}{2}\right)\left(y-\frac{y_1+y_2}{2}\right)(x_2-x_1)(y_2-y_1)+\frac{1}{80}\left((x_2-x_1)^4+6(x_2-x_1)^2(y_2-y_1)^2-3(y_2-y_1)^4\right)\Bigg)\Bigg)\end{aligned}$$