Volume 33 (2014), Number 2 pp. 1–13 COMPUTER GRAPHICS *forum*

Shading curves: vector-based drawing with explicit gradient control

Henrik Lieng, Flora Tasse, Jiří Kosinka, Neil A. Dodgson

The Computer Laboratory, University of Cambridge, UK

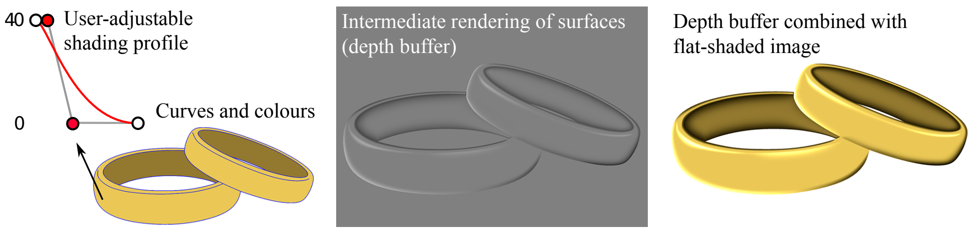

40
0
User-adjustable shading profile
Curves and colours
Intermediate rendering of surfaces (depth buffer)
Depth buffer combined with flat-shaded image

Figure 1: *Drawing with shading curves. Left: input curves, colours, and a shading profile at the given curve location. In this example, the shading profile represents difference in luminance to the colour of its related region. That is, the luminance at the curve in the resulting image is the luminance of the yellow-ish colour plus* 40*. Middle: intermediate image extracted from the depth buffer of rendered surfaces created with our framework. The shapes of the surfaces are dictated by the shading profiles. Right: combining the flat colour image with the luminance modification produces the final result.*

Abstract
A challenge in vector graphics is to define primitives that offer flexible manipulation of colour gradients. We propose a new primitive, called a shading curve, that supports explicit and local gradient control. This is achieved by associating shading profiles to each side of the curve. These shading profiles, which can be manually manipulated, represent the colour gradient out from their associated curves. Such explicit and local gradient control is challenging to achieve via the diffusion curve process, introduced in 2008, because it offers only implicit control of the colour gradient. We resolve this problem by using subdivision surfaces that are constructed from shading curves and their shading profiles.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Generation, Graphics Utilities

1. Introduction

Vector graphics provides a powerful framework for drawing compelling 2D imagery. An important aspect of such drawing is to be able to control colour gradients [OBW*08, FSH11]. Currently, there are three ways to manually manipulate colour gradients: diffusion curves (DCs) [OBB*13], the linear gradient tool, and the gradient mesh tool. The last two tools are found in vector drawing applications like Adobe Illustrator and CorelDRAW (gradient mesh is called 'mesh fill' in Corel DRAW).

Our approach is related to the DC primitive. A DC is a freeform curve (modelled as a B-spline curve) that is associated with colours on each side of the curve. These colours are smoothly propagated, or diffused, filling the entire image. Its advantage is that it is associated with a natural type of input whilst supporting smooth propagation of colours [OBB*13]. By contrast, the linear gradient tool does not support smooth propagation of colours (although it can be associated with freeform curves) and the gradient mesh tool is restricted to rectangular control meshes and does not support freeform curves as input (although it produces smooth propagation of

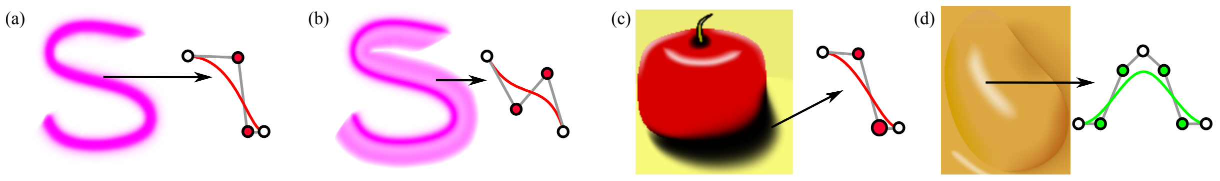


Figure 2: *Our shading curve supports manipulation of shading profiles, which dictate how the colour, associated with a shading curve, propagates out from the curve. Such shading profiles can give rise to a wide range of effects. In (a) and (b), the colouring related to an S-shaped curve is largely controlled via shading profiles. In (c), the extent of a cast shadow is controlled principally by moving the enlarged control point. In (d), a specular highlight is created with a bell-like profile. We also refer to the supplementary video, which demonstrates interactive manipulation of shading profiles.*

colours). However, a limitation of DCs is that they do not support explicit control of the colour gradient (Sections 3 and 6).

To achieve explicit control of the gradient of the colour that is related to the curve, instead of performing diffusion, we associate *shading profiles* to each side of the curve. Figure 1(top) shows how shading of two rings can be achieved with such shading profiles. First, the boundary curves are drawn, defining the shapes of the objects. A bounded region can be associated with a colour, resulting in a flat-shaded image with a colour defined in each region. Additionally, each side of a curve is associated with a luminance or colour adjustment value. This value specifies the modification in luminance (or colour) of the underlying colour of the flat-shaded image. Then, a shading profile, which can be manually adjusted, represents the resulting profile of the luminance (or colour) adjustment in the perpendicular direction to the curve. Figure 2 shows the influence of shading profiles on the resulting image. While our primitive can be employed in many artistic settings, it is particularly motivated by the problem of drawing shade and light (Section 3). We therefore refer to our primitive as a *shading curve*.

It is not clear how one could use diffusion to propagate colours in accordance with shading profiles (Section 6). First-order DCs [OBW*08] are restricted to (colour) value constraints and do not natively support manipulation of the colour gradient. By contrast, second-order DCs [FSH11], support first-derivative constraints to alter the colour gradient in a given direction. However, these first-derivative constraints are restricted to zero derivatives, which cannot represent a general shading profile. For that reason, we chose to use a method different from diffusion and employ Catmull-Clark subdivision surfaces constructed from shading curves and their associated profiles. Catmull-Clark subdivision is related to several attractive properties, such as the local convex hull property and local support [dB78, PR08], which have enabled us to demonstrate *explicit* and *local* control of the colour gradient. A challenge faced with this approach is to convert the shading curve primitive to control meshes for Catmull-Clark subdivision. A robust solution to this problem is presented in Section 4.2.

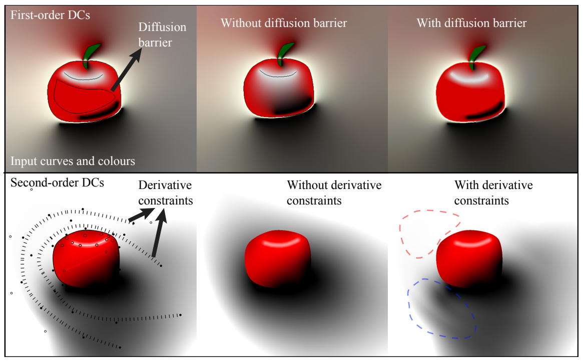


Figure 3: *Influence of DCs on colour propagation. The results with first-order DCs and second-order DCs were drawn with prototype software provided in OBW*08 and FSH11, respectively. The dashed regions (bottom right) illustrate that second-order DCs' derivative constraints give rise to different types of behaviour depending on the local configuration of value constraints.*

In summary, our contributions are a new way to define smooth gradient profiles for vector graphics and a robust solution for determining the locations of control points for subdivision surfaces from freeform curves.

2. Related work

As mentioned earlier, controlling colour gradients is currently achieved with three approaches: diffusion curves, the linear gradient tool, and the gradient mesh tool. In Section 6, we compare our approach with all of these three methods. In this section, we describe previous solutions proposed in the literature to alter the colour gradient with diffusion curves and alternative methods to achieve shading effects with vector graphics.

First-order DCs are rasterised via Laplacian diffusion [OBW*08] (thus solving the PDE $\Delta f = 0$). The Dirichlet boundary conditions associated with the PDE (specifying values of the unknown function) correspond to the colours associated with each side of the freeform curve drawn by the artist. That is, Laplacian diffusion is performed by diffusing colours specified at curves to the image domain, which produces a harmonic colour function.

Laplacian diffusion does not natively support manipulation of the colour gradient. However, several extensions have been suggested to achieve some degree of manipulation: Gaussian blurring can be used to smooth the sharp transition across the curve [OBW*08], weights for rational harmonic functions can be utilised to alter the relative influence of a boundary condition [BEDT10], and diffusion barriers have been proposed to limit the extent of the colour propagation [BEDT10] (Figure 3(top)).

Gradient control can be achieved by higher-order interpolation. To this end, the DC primitive has been extended to second-order interpolation with the bi-Laplacian operator [FSH11,BBG12] (thus solving the PDE $\Delta^2 f = 0$). Curves and points related to Neumann boundary conditions (derivatives of the unknown function) can be specified. Derivative curves constrain the first derivative to zero either along the curves or across the curves [BBG12].

While the zero-derivative constraints provide a way to manipulate the gradient of the colour function [FSH11, BBG12], they do not achieve *explicit* colour gradient control. We have found these constraints to have two types of behaviour on the colour function, as demonstrated in Figure 3(bottom). The first type of behaviour *suppresses* the propagation of colour. This can be seen in the top portion of the example, where the black colour is suppressed (red-dashed region). This is because there are only white point constraints located normal to the derivative curve; the other black point constraints are occluded by tear curves (the boundary of the apple). The colour of the white point constraint located normal to the curve is therefore propagated to the zero-derivative curves. In the bottom portion of the image, however, both black and white constraints are located normal to the zero-derivative curves, without being occluded by tear curves. In this scenario, the derivative curves give rise to a wavy-shaped colour profile (blue-dashed region). As a consequence of these two different types of behaviours, zero-derivative constraints can give rise to unpredictable behaviour because their influence on the colour gradient depends on the spatial configuration of neighbouring colour constraints. By contrast, our approach lets the user manipulate the colour gradient out from curve locations explicitly without being influenced by neighbouring curves, as demonstrated in the supplementary video and in Figure 2.

An alternative way to shade vector-based drawings is to convert the image into a pseudo-3D representation (that is, a normal vector is associated with each pixel) or a full 3D representation (with depth coordinates and normal vectors). Standard 3D shading techniques can then be used to shade the image. If normal vectors have been extracted, a local shading model, like Phong shading, is suitable [WTBS07, WOBT09, SBSS12] and if depth information has been extracted, global rendering methods, like path tracing, can be used [SKČ*14].

We decided not to pursue this line of research. The rationale for this decision is that estimating normal vectors and depth information in the general setting of 2D images is challenging and therefore prone to give a result other than that desired by the artist. A manual method, like ours, provides the user with a type of input that is directly associated with the resulting image: the input colours are associated with the colours of the objects and the shading profile represents the propagation of those colours. Note that we are not arguing that manual methods are 'better' than automatic methods. However, we do argue that a user of future drawing technology should be allowed to draw shading and abstract colourings manually. There is therefore incentive to improve such manual drawing technologies.

Alternatively, one can model full 3D representations, purely via the 2D domain, which can be later rendered and viewed in traditional 3D applications [Joh02, JC08, OSJ11, AJC11]. Such methods typically *inflate* bounded domains to surfaces, using input curves as boundary conditions. Note that several methods employ Laplacian diffusion to achieve such inflation (e.g. [JC08, AJC11], which is similar to how diffusion curves are used for colour diffusion. In this respect, Andrews et al. [AJC11] provide several mechanisms to adjust the gradient of the surface using internal curves, an approach akin to our slope curves and the zero-derivative constraints employed by Finch et al. In general, however, such methods are not directly suitable for colour interpolation because additional aspects, such as avoiding colour saturation, would have to be incorporated. Note that a reconfiguration of 2D-to-3D modelling methods that employ Laplacian diffusion would correspond to first- and second-order diffusion curves, which we compare against in Section 6.

Finally, Olsen et al. [OSJ11] use the distance transform for the inflation of their surfaces, which can seem similar to our usage of the distance transform (Section 4). However, the two methods employ the distance transform for different purposes. The method of Olsen et al. uses the distance transform to inflate their meshes (that is, defining z coordinates). By contrast, our solution uses the distance transform to place the boundary of the shading from a given curve (that is, defining (x,y) coordinates). Our use of the distance transform in this setting is therefore novel, as further described in Section 4.

3. Drawing with shading curves

The shading curve is inspired by chiaroscuro drawing [Cen54] (Figure 4). We suggest the following approach to drawing with shading curves:

1. Draw areas of constant tone with curves, including object boundaries and main tonal areas.

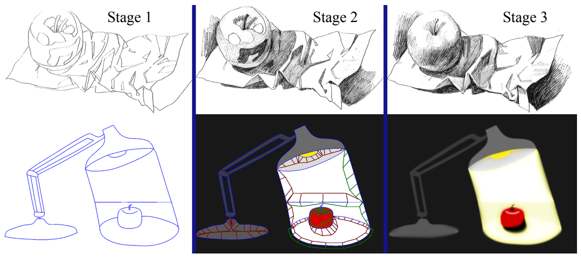


Figure 4: *Drawing chiaroscuro with a traditional method (top; images from Civ05) and with the shading curve (bottom). Such drawing can be performed in three stages. Stage 1: outline object boundaries and main tonal areas. Stage 2: fill in each area with a tone or colour, and with the shading curve: define the extent of the propagation of colours (green and red quads and triangles). Stage 3: smooth out the colours or tones, and with the shading curve: adjust shading profiles globally and locally.*

2. Fill in each individual area with constant colour and select the influence of that colour to adjacent areas.
3. Smooth out the colours with shading profiles. Refine colours and tones locally by adjusting the attributes of the shading curves.

In addition to the shading curve primitive, we have found it helpful to treat boundary curves and interior curves differently. Thus, we have implemented two types of shading curves with different behaviour. The first type of curve, the *boundary curve*, defines a sharp transition across the curve. It is therefore suitable for object boundaries since they are typically defined as hard edges (e.g., Figure 2(c)). By contrast, the second type of curve, the *slope curve*, defines a smooth transition across the curve. Slope curves are useful for shading highlights and transitions within the object interior, such as specular highlights and cast shadows from other objects (e.g., Figure 2(d)).

4. Manipulation and rasterisation of shading curves

In this section, we present the framework for defining and rendering shading curves. Figure 5 shows the pipeline of the framework and the supplementary video shows interactive manipulation of shading curves. Our framework takes, as input, a set of cubic B-spline curves in 2D. A set of attributes is associated with either side of each curve. These attributes are stored along with the curve control points. A user can chose to globally manipulate attributes, by selecting curves, or to perform local manipulations, by selecting curve control points. From this set of inputs, our framework creates 3D control meshes, which define Catmull-Clark subdivision surfaces. The depth buffer of an OpenGL rendering is then used to create the desired effect, implemented as either a luminance alteration or a colour profile.

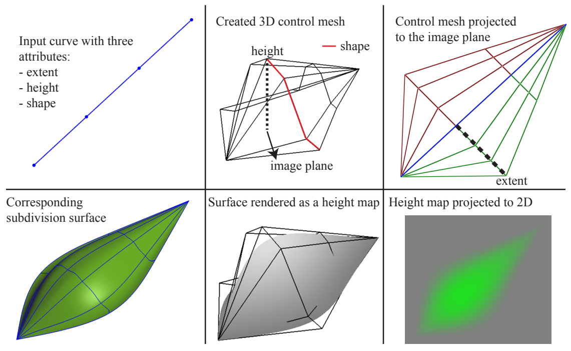

Input curve with three attributes:
- extent
- height
- shape
Created 3D control mesh
height
shape
image plane
Control mesh projected to the image plane
extent
Corresponding subdivision surface
Surface rendered as a height map
Height map projected to 2D

Figure 5: *Creating a 3D control mesh from a curve. Attributes used to control the shape of the corresponding surface are associated with the curve: an extent attribute is associated with the extent of the mesh in the image plane perpendicular to the curve, a height attribute is associated with the height of the mesh perpendicular to the image plane, and a shape attribute defines the shape of the surface normal to the curve. The resulting subdivision surface can be used for various effects. In this example, a green colouring is created from the depth buffer of the surface rendering.*

There are two computational steps involved in our pipeline: creating 3D control meshes from curves (Section 4.2) and rendering the surfaces (Section 4.3). The framework is computationally efficient and the user is able to refine the input interactively. Note that both computational steps must be performed if shading curves are moved in the image plane. In contrast, it is not necessary to re-create the 3D control meshes if curves are not moved (e.g., control meshes do not have to be re-created if shading profiles are manipulated).

Before the computational steps are presented, we need formal definitions of the input curves and the output control meshes. These definitions are presented next.

4.1. Definitions

An input B-spline curve is defined by a sequence of n control points $Q_i = (x_i, y_i)$, $i = 1, \ldots, n$ (Figure 6(left)). The 2D unit normal vector of the curve at the position related to Q_i, computed from the curve's first derivative [dB78], is denoted N_i. It is assumed that the normal vectors are consistently oriented. Three attributes, extent, height, and shape, are attached to each control point:

- extent, $e_i \in \mathbb{R}_0^+$: defines the extent of the mesh in the direction N_i from Q_i.
- height, $h_i \in \mathbb{R}$: defines the height of the mesh in the perpendicular direction to the image plane at Q_i.
- shape, $((\alpha_{i,2}, \beta_{i,2}), (\alpha_{i,3}, \beta_{i,3})); \alpha_{i,j}, \beta_{i,j} \in [0,1] \subset \mathbb{R}$: defines the shape of the surface profile from Q_i towards N_i.

Figure 7 illustrates how these shading profiles can be edited in a user interface by manipulating curves (rendered as cubic

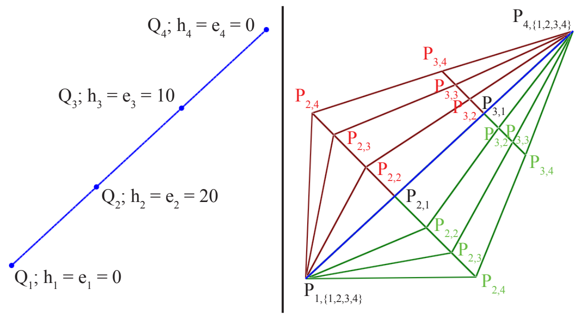
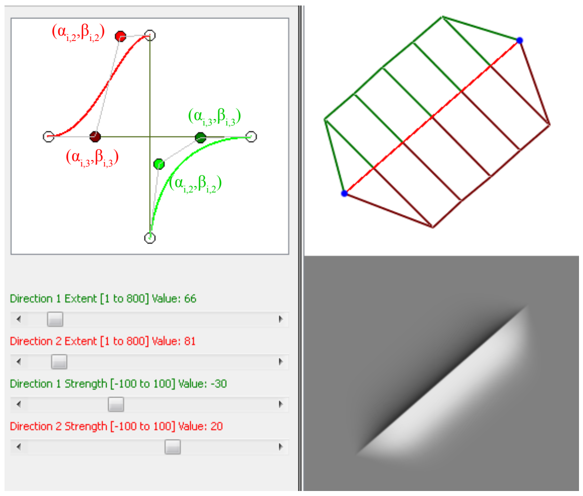


Figure 6: *Notational labels added to the example given in Figure 5. There is no notational difference between the two control meshes associated with the curve. In this illustration, they are marked in red and green colours and the black control points are shared.*



$(\alpha_{i,2},\beta_{i,2})$
$(\alpha_{i,3},\beta_{i,3})$
$(\alpha_{i,3},\beta_{i,3})$
$(\alpha_{i,2},\beta_{i,2})$

Direction 1 Extent [1 to 800] Value: 66
Direction 2 Extent [1 to 800] Value: 81
Direction 1 Strength [-100 to 100] Value: -30
Direction 2 Strength [-100 to 100] Value: 20

Figure 7: *The three attributes of a shading curve manipulated in our prototype user interface: shape – the shading profile – is manipulated by the inner control points of a cubic Bézier curve. The extent and height (strength) attributes are edited with sliders. The red/green colourings are used to separate the two sides of the curve.*

Bézier curves) defined in a normalised coordinate system. The extent and height attributes are, in our prototype system, edited via sliders.

With the input curves and attributes defined, the output 3D control meshes are now described. A control mesh is created on the side of the curve related to the direction of N_i. A mesh in the opposite direction $-N_i$ can also be created. Note that there is no requirement that both meshes are created and they can be treated as completely separate. In our prototype system, the user can manually enable or disable the meshes on either side of the curve. Additionally, control meshes are created separately for each curve. Given a set of disjoint curves, our framework therefore creates a set of disjoint control meshes. If multiple curves are joined at a junction point, the related control meshes can be merged (see the supplementary document, Section 1, for implementation details).

The 3D control points derived from and associated with Q_i are defined as $P_{i,j} = (x_{i,j}, y_{i,j}, z_{i,j})$, $i = 1,\dots,n; j = 1,\dots,m$, where n is the number of control points associated with Q_i and m is the number of control points associated with the shading profile (Figure 6(right)). We have found $m = 4$ sufficient to demonstrate our results in Section 5. A rectangular grid of size $n \times m$ is initially assumed to be created out from the curve. However, some quads will later be deliberately degenerated to triangles.

The coordinates $(x_{i,4}, y_{i,4})$ represent the location of the 'outermost' control points, $P_{i,4}$, of the mesh. A naïve definition of this point is:

$$(x_{i,4}, y_{i,4}) = Q_i + N_i e_i. \tag{1}$$

This solution, however, can give rise to folding artefacts when the surface is projected to 2D. Note that this problem has been encountered previously in vector graphics (see the supplementary document, Section 2, for a brief review). However, previous solutions are not robust to the geometrical layout of the input curves. Placing $(x_{i,4}, y_{i,4})$ is therefore the principal problem of our framework. We present a robust solution to this problem in Section 4.2.

The z, or 'height', coordinates are now defined. The control points along the original curve, $P_{i,1}$, are set according to the corresponding height attribute. We can assume that the effect of the surface, being adjustment in luminance or colour, should fair out to have zero effect at the 'other' end of the control mesh at $P_{i,4}$. That is, surface values of $z = 0$ do not alter the image. Given the related curve control points $Q_i = (x_i, y_i, 0)$, the following coordinates can now be defined:

$$P_{i,1} = (x_i, y_i, h_i);$$
$$P_{i,\{2,3\}} = \alpha_{i,\{2,3\}} \left(P_{i,1} - Q_i\right) + \beta_{i,\{2,3\}} \left(P_{i,4} - Q_i\right);$$
$$z_{i,4} = 0.$$

The control points $P_{i,\{2,3\}}$ related to the shape attribute are therefore placed on the plane defined by $P_{i,1}$, $(x_i, y_i, 0)$, and $P_{i,4}$, ensuring that the profile is indeed modelled in the direction towards $P_{i,4}$.

By default, the height attribute defines changes in luminance using the LAB colour space. To support coloured profiles, the RGB colour space can optionally be used. Each curve therefore has an optional colour attribute C on each side.

We have so far treated the control meshes on each side of the curves as separate meshes. Thus, the transition across a curve will typically be discontinuous. However, one might wish to model a smooth transition across the curve. To this

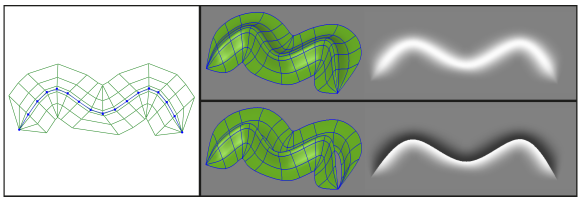


Figure 8: *Slope (top) and boundary (bottom) curves. Slope curves compose single smooth surfaces and boundary curves compose two separate surfaces.*

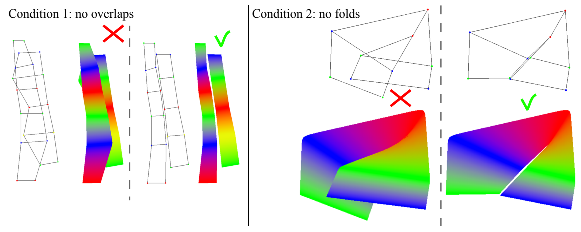


Figure 9: *When the 3D control meshes are projected to the (x,y)-plane, overlaps between meshes and folds are unwanted. The rows of the meshes have been coloured with a constant colour to highlight the folding artefacts.*

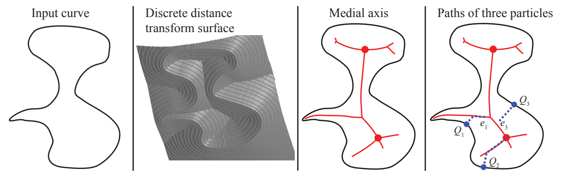


Figure 10: *Our solution traces particles on the DT surface. The particles follow the gradient of the surface. Consequently, they are implicitly traced along the MA.*

end, we have implemented two types of curves: boundary curves for discontinuous transitions and slope curves for continuous transitions (Figure 8). The only technical difference between these two types of curves is that, for each i, the $P_{i,1}$ on both sides of a slope curve are merged; that is, they appear only once in the control mesh. The slope curve, now only associated with a single (merged) control mesh, therefore gives rise to a single smooth surface defined on both sides of the curve.

All coordinates of the 3D control mesh have now been introduced. In the supplementary document, Section 1, we describe additional aspects related to these meshes, including design decisions at junctions, high curvature points, and curve-end points.

4.2. Defining the coordinates related to extent

In this section, we describe our solution to the placement of the 'outermost' mesh control points related to the extent attribute. This placement captures the extent of the shading profile out from the curve. Before presenting our solution (Section 4.2.2), we describe this problem in more detail.

4.2.1. Problem description

To discuss this problem, we need some idea of what we mean by a 'good' solution. Informally speaking, the control meshes should behave naturally and should not give rise to visual artefacts when applied to images. Such visual artefacts appear when the following conditions are violated. When the control meshes are projected to the 2D image, they should:

- **Condition 1**: not overlap each other;
- **Condition 2**: not fold.

Figure 9 illustrates the visual artefacts related to the two conditions. Mathematically, when either of these conditions is violated, the resulting function defined by the projected surfaces gives rise to discontinuities. Such discontinuities should be avoided since they represent sharp jumps in the image, like image edges. Instead, the resulting function should be smooth across the image, unless colour jumps are specifically specified by, for example, drawing two curves in separate layers.

Note that we are only concerned with the extents of the control meshes projected to the 2D plane. This projection is defined by simply neglecting the z coordinate. To this end, we will, in the remainder of this section, refer to the mesh control points $P_{i,j}$ as 2D points with the coordinates $(x_{i,j}, y_{i,j})$. Additionally, let the line $P_{i,1}$–$P_{i,4}$ be L_i and the curve defined by $P_{i,4}$ be the approximate *offset curve* of the input curve.

4.2.2. Solution

Our solution performs a single step of tracing on the distance transform [RP66] (DT) surface given by the input curves. The 3D coordinates of a DT surface point are defined as an (x,y) position in image plane and z, the distance to the closest curve point at (x,y) (Figure 10). This surface has many useful properties:

- The gradient at any point not positioned at an input curve or at the medial axis (MA) [Blu67] of the set of input curves points in the direction normal to the curve;
- The slope of this gradient is 45 degrees;
- C^0 creases in the surface relate to the input curves and the MA;
- The slope along the MA is less than or equal to 45 degrees;
- Stationary points (not related to curve points) and local extrema on the surface lie on the medial axis transform.

A conceptual solution is now described. Imagine a particle dropped on the DT surface for each Q_i along N_i. Such a particle is dropped infinitely close to the curve point associated with Q_i in the direction of N_i (that is, it is dropped on the correct side of the curve). These particles then follow the gradient field on the DT surface.

According to the properties of the DT surface, the particles behave as follows (Figure 10). As time increases, they move 'upwards' along the direction of the gradient, towards the MA. When the MA is reached, they will continue to move along the MA until they reach a local extremum and then remain stationary. Eventually, all particles will reach a local extremum or the edge of the domain (i.e., the image border). A particle also turns stationary if its z coordinate (i.e., its DT value) equals the extent attribute.

A control point $P_{i,4}$ is placed on the location of its related stationary particle. Quads related to Q_i and Q_{i+1} (that is: the quads defined by $P_{i,1}$, $P_{i+1,1}$, $P_{i+1,4}$, and $P_{i,4}$) are degenerated to triangles if their related particles coincide (that is: if $P_{i,4} = P_{i+1,4}$, they merge into a single control point). Our implementation of this solution traces pixels on the discrete distance transform of the input curves (see the supplementary document, Section 3, for implementation details).

4.2.3. Discussion

Our solution satisfies Conditions 1 and 2, as long as the input curves do not intersect (in such cases, the curves can be trivially split): Control meshes of neighbouring curves do not overlap (Condition 1) as the meshes are constrained by the MA. Meshes do not fold (Condition 2) as the paths created by the particles only merge; they do not cross.

The offset curve defined by $P_{i,4}$ is restricted by the MA. However, one might wish to extend the shading profile further. A future investigation could be to verify whether a weighted distance transform [KKB96] can be used for this purpose. This extension would increase the complexity of the curve primitive since a weight needs to be attached to it.

4.3. Rendering

The process of creating the final image from the control meshes is straightforward: subdivide the control meshes given by $P_{i,j}$, render the surfaces off screen, and then apply these surfaces to the underlying image.

In our prototype system, we allow users to associate a colour with each image region bounded by the input curves. This approach produces a flat-shaded image and represents the 'underlying' image. This image is *supplemented* with a rendering of the surfaces to produce the resulting image.

We render our surfaces as Catmull-Clark subdivision surfaces. Since a surface must interpolate its boundary curve (defined by Q_i) in the image plane, boundary subdivision rules are used [DKT98]. We have used two subdivision steps to produce our results. In some cases, where the surface is stretched over a large area, three subdivision steps can produce a visually smoother result. Note that increasing to three subdivision levels still provides an interactive system in our setup.

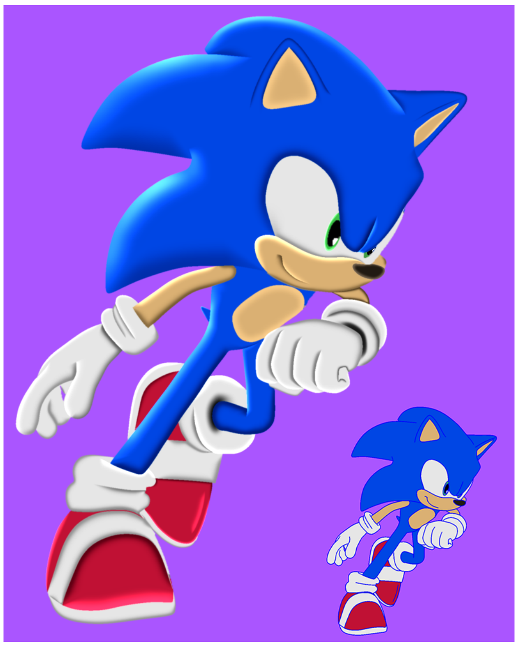


Figure 11: *Image shaded with our tool by a user with no artistic training in less than 10 minutes. We provided the user with the input curves (bottom right) which we traced from an image of Sonic ©SEGA.*

To create a complete per-pixel shading profile for the image, we render the Catmull-Clark surfaces using OpenGL off-screen rendering. The shading image, S, is then extracted from the depth buffer. The camera parameters are set so that depth values directly correspond to the height attributes; thus $S \in [-100, 100]$ for luminance and $S \in [0, 1]$ for colour.

Finally, this shading image is applied to the underlying image I producing the resulting image R. Luminance adjustment in the LAB luminance channel L is performed by:

$$R_L = I_L + S.$$

The two other colour channels are set by the corresponding channels in I.

In the setting of colour adjustment, the result is created by linearly interpolating between the original image and an image, I_C, defined with the piece-wise constant colour C:

$$R = S \times I_C + (1 - S)I.$$

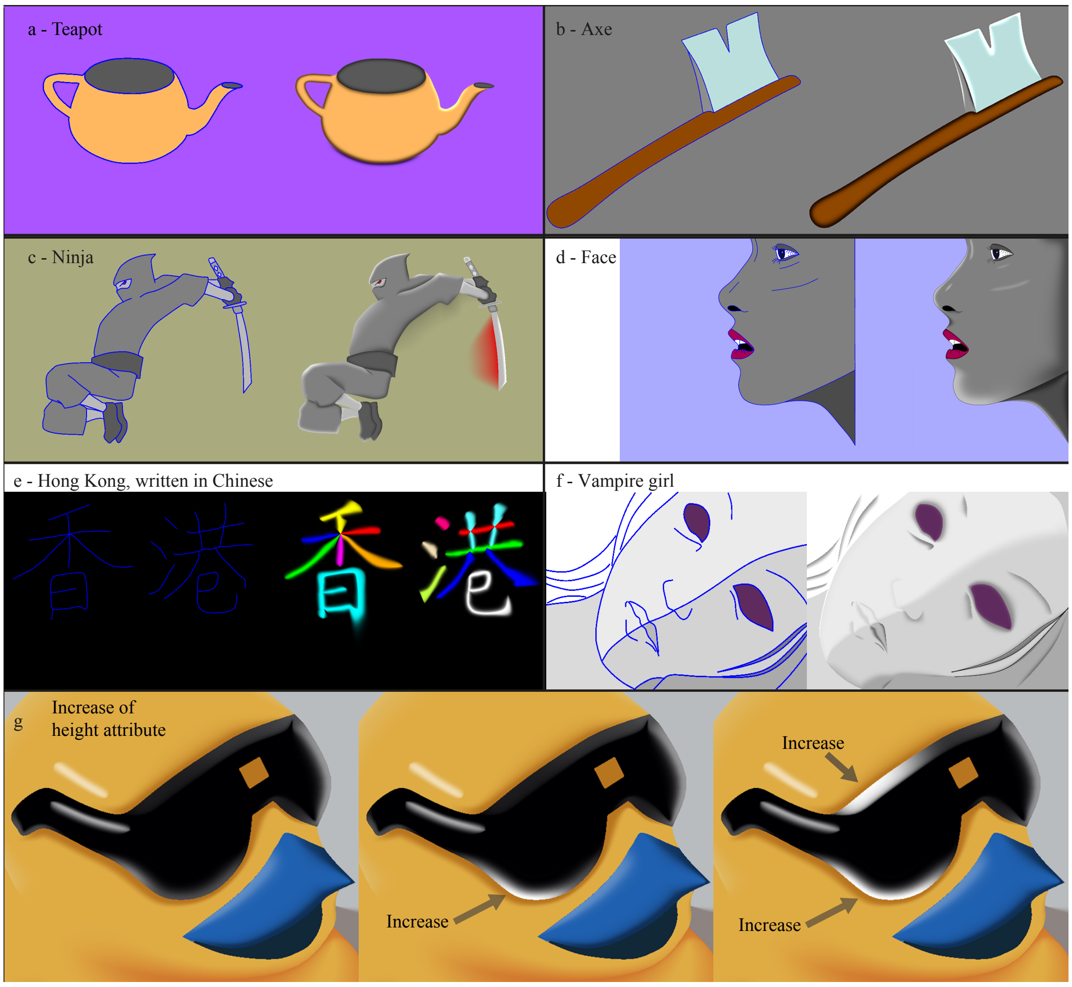


Figure 12: *Images drawn with our system.*

5. Results

The shading curve is adaptable to a wide range of visualisations. In this section, we discuss various types of visualisations that can be drawn using shading curves and we describe ways in which this primitive can be used. The prototype system was informally tested with novice users (Figure 11), which indicated that the primitive is easy to learn and use. Furthermore, we achieved interactive editing with our implementation. The Sonic image (Figure 11) has most surfaces (172) in a single layer. On a system running Ubuntu 12.04 on an Intel Core i7 CPU (870, 2.93GHz x8), timing for the DT tracing was 18 milliseconds and 70 milliseconds for the rendering of the subdivision surfaces. The performance for subdivision could be further improved with a GPU implementation [NLMD12], where we should expect a performance boost of at least one order of magnitude, according to timings on such implementations.

Recall from Section 3 the three-stage approach to drawing with the shading curve: (1) outline objects and main tonal areas, (2) fill in colours and define the influence of those colours, and (3) smooth out colours with shading profiles. In the following, we present ways in which the final stage can be performed efficiently.

Figure 12(a–b) shows cartoon-like images created using the default settings ($(\alpha_{i,2}, \beta_{i,2}) = (0.15, 1.0); (\alpha_{i,3}, \beta_{i,3}) = (0.4, 0.0); h_i = \pm 20; e_i = 3\%$ of the image diagonal); height is manually set negative for dark shadings. Adding slope curves to such images, as in Figure 12(d), can induce more glossy looks. Simulated 3D effects can be achieved by varying the attributes along the curves (Figure 12(c–d)). Increasing or decreasing the height at selective locations along boundary curves adds a sense of location of the main light sources in the scene, as well as their strength (Figure 12(g)). The two other parameters, extent and shape, are then varied

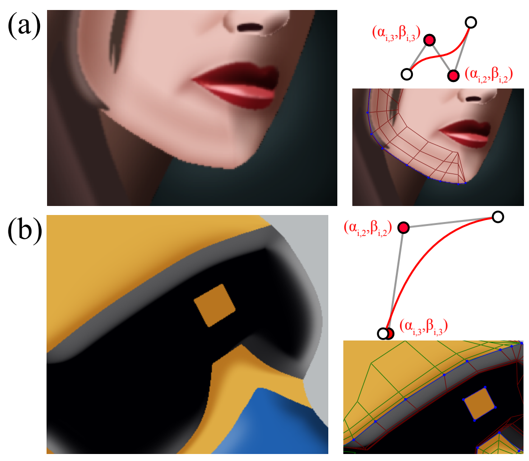

(a)
$(\alpha_{i,3},\beta_{i,3})$
$(\alpha_{i,2},\beta_{i,2})$
(b)
$(\alpha_{i,2},\beta_{i,2})$
$(\alpha_{i,3},\beta_{i,3})$

Figure 13: *Shading profiles for various effects. Image at left, image with overlaid control mesh at right. (a) The chin is emphasised with a wavy profile. (b) The 3D shape of the glasses is conveyed by the use of strong light and sharp fall off of the profiles.*

to give the shading form. Extent is typically set equal for all control points along a boundary at the preferred offset. Note that if the extent is set too large, the surfaces will extend to local maxima in the DT. See Figure 2(a) for an example where the extent is set further than the MA, forcing many 'outer' mesh points to be placed on a local maximum. This can be dealt with by the artist by decreasing the extents. Such editing is analogous to editing using the width tool in Adobe Illustrator.

The shading profile is the most important factor for creating various types of shadings (Figures 2 and 13). Varying the first shape point related to $P_{i,2}$ between $(0,0)-(0,1)$ ($\beta_{i,2}=0$) in the normalised coordinate system creates a steep shading fall-off, and is suitable for more diffuse conditions. On the other hand, varying this shape point between $(0,1)-(1,1)$ ($\alpha_{i,2}=1$) creates a stronger profile out from the boundary and is suitable for depicting cast shadows and shading highlights. The second shape point, defining $P_{i,3}$, should be placed along $(0,0)-(1,0)$ ($\alpha_{i,3}=0$) for a smooth fall-off to zero at $P_{i,4}$. The location of this fall-off in the image is then controlled by this shape point. The offset curve defined by $P_{i,3}$ can also be shown in the main editing window so that the artist can directly visualise this fall-off.

Figures 14 and 15 show multi-layered images shaded with the chiaroscuro-inspired drawing technique. Each area is defined as single layer and is blended with the underlying flat-shaded image with either screen or multiply blending modes.

6. Comparisons with related work

Recall from Section 1 that there are three previous methods to manipulate colour gradients in vector graphics: DCs, the linear gradient tool, and the gradient mesh tool. In this section, we discuss the differences between these methods and our approach.

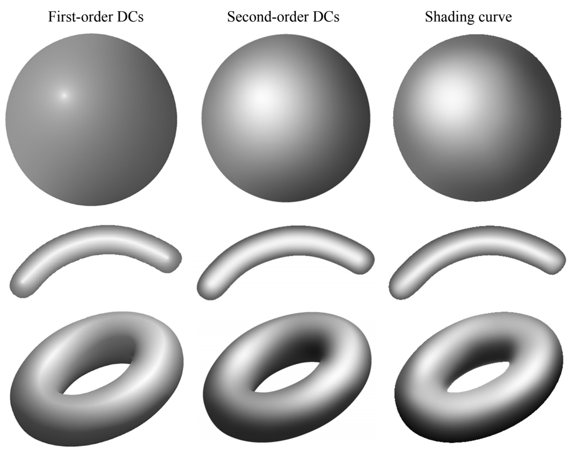


Figure 16: *Shading a sphere, a tube, and a doughnut with three different methods. The comparison between the two diffusion curve methods is fair and accurate, as the inputs are the same. However, the comparison to shading curves is inaccurate, as the inputs are fundamentally different from DCs and they are defined separately in different user interfaces. The results for the diffusion curve methods are from FSH11.*

6.1. Comparisons with diffusion curves

Recall that there are two types of DCs: first-order DCs and second-order DCs. In the following, we compare both of these two primitives with our shading curve.

First-order diffusion curves

The naïve first-order DCs primitive (that is, not assuming additional attributes like blur or weights) is simpler than our primitive. The DC primitive can be associated with a single attribute on each side of the curve: colour. By contrast, our method requires the additional width attribute (assuming that height and shape are fixed). In certain types of scenarios, where colour images produced by Laplacian diffusion provide acceptable results, the DC primitive can be preferable as less input is required.

The disadvantage of first-order DCs, compared to the shading curve, is that they are limited to harmonic solutions. The artist is therefore constrained to first-order diffusion conditions. Figure 16 demonstrates this point: the solution of first-order DCs is only smooth away from constraints. Thus, this solution produces creases along curves, which can only be smoothed by post-processing operations such as image blurring. By contrast, methods with control over derivatives

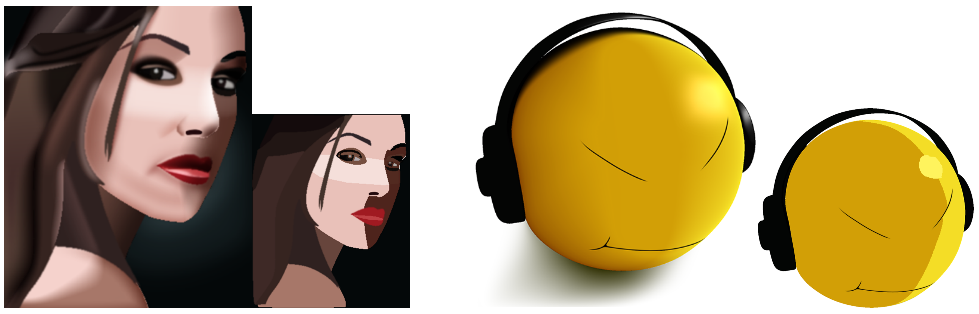


Figure 14: *Drawing with the chiaroscuro-inspired drawing technique. The smaller images, in the centre of the figure, comprise regions that are each shaded with a single flat colour. The larger images at left and right are these same images shaded off using the proposed shading curve framework. Curves were traced from images from ©Evie Moore, switchplane.com (left), and ©sha-x-dow.deviantart.com (right).*

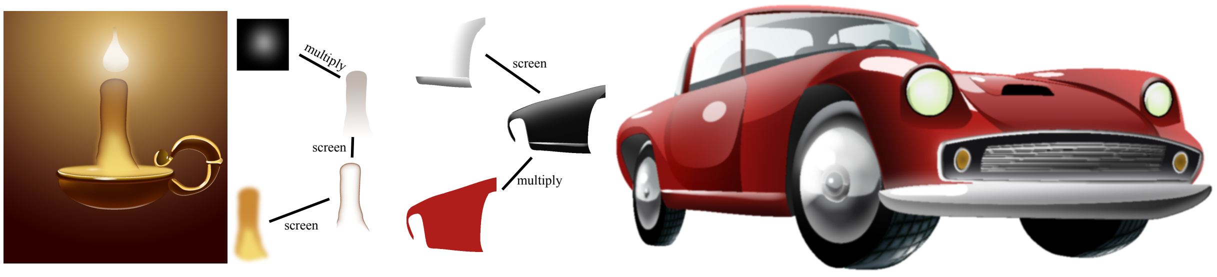


Figure 15: *Using inspiration from vector shade trees LMPB*13, our method, along with some artistic skill, can give rise to depictions of complex materials.*

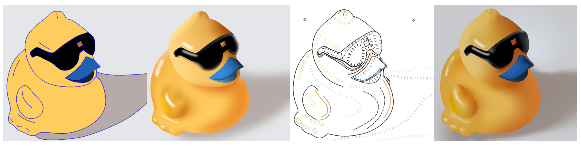


Figure 17: *Informal comparison between our method (left two images) and second-order diffusion (right two images) [FSH11].*

and gradients, such as the shading curve and second-order DCs, support smoothness across input curves.

Second-order diffusion curves

The limitations of first-order DCs motivated Finch et al. [FSH11] to propose second-order DCs. The PDE is now constrained by *RGB* colours and derivative constraints. In the framework proposed by Finch et al., a curve associated with derivative constraints forces the derivative of the colour function to zero either across the curve or along the curve. See Figure 17 for an informal visual comparison with our method.

The mathematical differences between our method (subdivision – Catmull-Clark spline surfaces) and second-order DCs (PDEs – thin-plate spline surfaces) give rise to practical differences between the two approaches (see the supplementary document, Section 4, for a brief introduction to these two types of surfaces). The advantage of DCs is related to vivid colourings. Our method uses surfaces as interpolants between an underlying image and colour or luminance associated with shading curves. Thus, colour propagation is controlled along a shading curve and can only be manipulated in its perpendicular direction with shading profiles. While DCs do not support the flexibility of shading profiles, they support points as constraints. A point constraint adds boundary constraints to its four nearest pixels. In this way, the colouring of an object can be separated from the curves defining the object's boundary by placing point constraints inside the object. This approach can be preferred over drawing 'special' curves, as required by shading curves, to achieve certain colourings (see the supplementary document, Section 5, for a practical example).

Aspects where the shading curve is preferable over DCs include local gradient control, layering, and computational efficiency:

- The shading profile, modelled with Catmull-Clark subdivision, provides a local way to model colour gradients.

The extents of the edits are local because the control meshes are directly created from the extent attribute of the shading curve. The shading profile is guaranteed to influence only locally the colour/luminance function owing to local behaviour of Catmull-Clark surfaces. By contrast, DCs do not provide a way to locally control the colour gradient. Instead of curved profiles, representing the gradient of the colour/luminance function in a given direction, DCs' derivative constraints are restricted to forcing the derivative to zero in a given direction (Section 2). Thus, DCs do not possess the same degree of freedom for manipulating the colour gradient. Additionally, the boundary conditions of DCs are global constraints, influencing the entire colour function. Providing a similar mechanism to the shading profile with DCs is therefore challenging (see the supplementary document, Section 4, for supporting information on this claim).

- DCs are not particularly suitable for layering. In image compositing, the spatial extent of a layer should be easily defined. This is ensured with the shading curve using the extent attribute. By contrast, the solution of DCs must be well defined inside the boundary specified by the DCs. This boundary must be explicitly defined by curves, meaning that influences of open curves and point constraints are not easily constrained. In practice, single objects, fully enclosed by curves or the image border, can be separated into layers. Local effects like specular highlights, however, would not typically be treated by layering and should be incorporated as a single layer to ensure that their effects are smoothly blended with each other.
- Interactive performance is easily achieved with our method. Even the naïve CPU implementation provides acceptable performance. On the other hand, a diffusion process solves a large, sparse, linear system which is naïvely time consuming. Approximate solutions with low-resolution CPU finite element methods [BBG12] or high-resolution (re-evaluation-only) boundary element methods [IKCM13] need to be employed in order to achieve interactive speeds with second-order DCs.

In the supplementary document, Section 4, we discuss additional practical aspects related to this comparison. Additionally, observations from drawing sessions with novice users indicated that a tool based on shading curves would be easier to learn compared to second-order DCs and drawing sessions with professional designers indicated that our primitive can complement the current suite of tools (see the supplementary document, Section 7, for details on these drawing sessions).

6.2. Comparisons with the linear gradient tool

Illustrator's brush tool supports linear and radial directions of colour gradients both along and across a brush stroke. The feature of applying the gradient across the stroke, named 'gradient on a stroke', was added in version CS6 (2012). A linear or radial gradient is defined as an editable univariate colour function (a curve in colour space) applied in a given (linear or radial) direction. The gradient-on-a-stroke feature is similar to the shading curve since it can be combined with Illustrator's width tool, introduced in version CS5 (2010), to define editable colour gradients out from curves in a given extent. While the extent can be varied along the curve, using the width tool, the colour gradient must remain constant for the entire stroke.

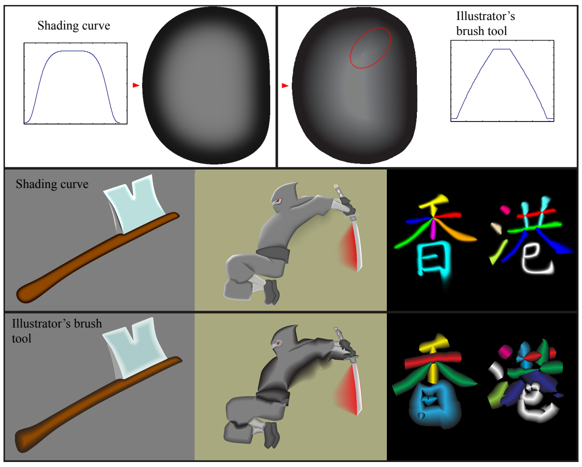


Figure 18: *Visual comparisons between the shading curve and Illustrator's brush tool. Top: Two types of artefacts produced with Illustrator's tool are highlighted: the transition between the brush colouring and the underlying layer is not smooth, as shown in the cross section, and folds give rise to C^0 crease artefacts, as shown inside the red ellipse. The comparisons are relatively fair as a similar type of input is defined for both methods.*

Mathematically speaking, the gradient-on-a-stroke tool and the shading curve are different. Although both are defined similarly, Illustrator's brush tool is associated with a univariate colour function and the shading curve is associated with a univariate shading profile, the rendering of the two primitives is fundamentally different. Illustrator's tool renders the brush primitive with reference to its associated colour profile. By contrast, our method creates a bivariate colour/luminance function (i.e., a surface), shaped according to the shading profile. The rendering of the shading curve is related to the surface instead of being directly produced from the original shading profile. With Illustrator's method, without any reference to a smooth surface, it is not clear how smooth colour profiles can be achieved.

Hence, there are three potential sources of visual artefacts in Illustrator's tool: the rendering procedure, the deformation procedure, and blending with underlying layers. Figure 18 shows images coloured with Illustrator's tool, where artefacts are visible. In comparison, our rendering procedure, using subdivision surfaces, produces smooth profiles without discontinuities.

6.3. Comparisons with the gradient mesh tool

Illustrator's gradient mesh tool allows the user to edit smooth (bivariate) colour functions with rectangular control meshes. Colours and first derivative constraints are associated with mesh control points. The control of the colour gradient is similar, but not equal, to our method. While Illustrator's tool does not support shading profiles, the derivative constraints directly correspond to the first derivative of the colour function out from given locations in the image. Additionally, colours can be edited in the object interior at points (and not curves), which is not supported with our method.

A major disadvantage of gradient meshes is that they are restricted to rectangular control meshes. They do not support freeform curves as input, which is a more natural type of input in drawing. This difference is identical to the difference between DCs and gradient meshes. Limitations related to rectangular control meshes for drawing are well established; see [OBW*08] for more discussion on this restriction and arguments to why control meshes are undesirable compared to curves in the setting of 2D drawing.

7. Conclusion

We have proposed shading curves as a new primitive for vector graphics. Shading curves are associated with shading profiles, defining the colour gradient out from curves. Our primitive is converted to 3D control meshes and rendered as Catmull-Clark subdivision surfaces. Catmull-Clark subdivision is related to several attractive properties, such as the local convex hull property and local support, which have enabled us to demonstrate explicit and local control of the colour gradient. Such local control is challenging to achieve with (bi-)Laplacian diffusion, which is used to render images created using diffusion curves.

Acknowledgements

Lieng was supported by a grant from the Norwegian Government. Tasse was supported by a Google European Doctoral Fellowship and an IDB Cambridge International Scholarship. Kosinka was supported by the Engineering and Physical Sciences Research Council [EP/H030115/1].

References

[AJC11] Andrews J., Joshi P., Carr N.: A linear variational system for modeling from curves. *Computer Graphics Forum 30*, 6 (Sept. 2011), 1850–1861. 3

[BBG12] Boyé S., Barla P., Guennebaud G.: A vectorial solver for free-form vector gradients. *ACM Trans. Graph. 31*, 6 (2012), 173:1–9. 3, 11

[BEDT10] Bezerra H., Eisemann E., DeCarlo D., Thollot J.: Diffusion constraints for vector graphics. In *Proc. NPAR* (2010), vol. 10, ACM, pp. 35–42. 3

[Blu67] Blum H.: A transformation for extracting new descriptors of shape. *Models for the Perception of Speech and Visual Form 1* (1967), 362–380. 6

[Cen54] Cennini C. D.: *The Craftsman's Handbook: "Il Libro dell' Arte"*. (D. V. Thompson, Trans.). Dover Publications, Mineola, USA, 1954. Original work published early 15th century. 3

[Civ05] Civardi G.: *Drawing Light & Shade: Understanding Chiaroscuro*. (L. Black, Trans.). Search Press Limited, Tunbridge Wells, UK, 2006 (Original work published 2005). 4

[dB78] de Boor C.: *A Practical Guide to Splines*. Springer, New York, USA, 1978. 2, 4

[DKT98] DeRose T., Kass M., Truong T.: Subdivision surfaces in character animation. In *Proc. SIGGRAPH* (1998), vol. 25, ACM, pp. 85–94. 7

[FSH11] Finch M., Snyder J., Hoppe H.: Freeform vector graphics with controlled thin-plate splines. *ACM Trans. Graph. 30*, 6 (2011), 166:1–10. 1, 2, 3, 9, 10

[IKCM13] Ilbery P., Kendall L., Concolato C., McCosker M.: Biharmonic diffusion curve images from boundary elements. *ACM Trans. Graph. 32*, 6 (2013), 219:1–12. 11

[JC08] Joshi P., Carr N. A.: Repoussé: Automatic inflation of 2D artwork. In *Proc. SBM* (Aire-la-Ville, Switzerland, Switzerland, 2008), Eurographics Association, pp. 49–55. 3

[Joh02] Johnston S. F.: Lumo: Illumination for cel animation. In *Proc. NPAR* (New York, NY, USA, 2002), ACM, pp. 45–ff. 3

[KKB96] Kimmel R., Kiryati N., Bruckstein A. M.: Sub-pixel distance maps and weighted distance transforms. *Journal of Mathematical Imaging and Vision 6*, 2-3 (1996), 223–233. 7

[LMPB*13] Lopez-Moreno J., Popov S., Bousseau A., Agrawala M., Drettakis G.: Depicting stylized materials with vector shade trees. *ACM Trans. Graph. 32*, 4 (2013), 118:1–10. 10

[NLMD12] Niessner M., Loop C., Meyer M., DeRose T.: Feature-adaptive GPU rendering of Catmull-Clark subdivision surfaces. *ACM Trans. Graph. 31*, 1 (2012), 6:1–11. 8

[OBB*13] Orzan A., Bousseau A., Barla P., Winnemöller H., Thollot J., Salesin D.: Diffusion curves: a vector representation for smooth-shaded images. *Commun. ACM 56*, 7 (2013), 101–108. 1

[OBW*08] Orzan A., Bousseau A., Winnemöller H., Barla P., Thollot J., Salesin D.: Diffusion curves: a vector representation for smooth-shaded images. *ACM Trans. Graph. 27*, 3 (2008), 92:1–8. 1, 2, 3, 12

[OSJ11] Olsen L., Samavati F., Jorge J.: Naturasketch: Modeling from images and natural sketches. *Computer Graphics and Applications 31*, 6 (Nov 2011), 24–34. 3

[PR08] Peters J., Reif U.: *Subdivision Surfaces*. Springer, New York, USA, 2008. 2

[RP66] Rosenfeld A., Pfaltz J. L.: Sequential operations in digital picture processing. *J. ACM 13*, 4 (1966), 471–494. 6

[SBSS12] Shao C., Bousseau A., Sheffer A., Singh K.: Crossshade: shading concept sketches using cross-section curves. *ACM Trans. Graph. 31*, 4 (2012), 45:1–11. 3

[SKČ*14] Sýkora D., Kavan L., Čadík M., Jamriška O., Jacobson A., Whited B., Simmons M., Sorkine-Hornung O.: Ink-and-ray: Bas-relief meshes for adding global illumination effects to hand-drawn characters. *ACM Trans. Graph. 33*, 2 (2014), 16:1–15. 3

[WOBT09] WINNEMÖLLER H., ORZAN A., BOISSIEUX L., THOLLOT J.: Texture design and draping in 2D images. *Computer Graphics Forum 28*, 4 (2009), 1091–1099. 3

[WTBS07] WU T.-P., TANG C.-K., BROWN M. S., SHUM H.-Y.: Shapepalettes: Interactive normal transfer via sketching. *ACM Trans. Graph. 26*, 3 (2007), 44:1–6. 3