Ray-traced diffusion points

Henrik Lieng
Oslo and Akershus University College of Applied Sciences, Norway

Figure 1: *Ray-traced diffusion curves and points. Left: input data for selected layers. In addition to colours, the proposed diffusion points support optional colour gradient constraints (black thin lines), which can be used to control the local spread of the specified colour. Dashed lines are diffusion barriers and do not have colours associated with them. Right: demonstration of complex colour gradients drawn with the proposed framework.*

Keywords: vector graphics, diffusion curves, ray tracing

Concepts: •**Computing methodologies** → **Image processing; Graphics file formats;**

1 Introduction

Diffusion curves [Orzan et al. 2008] (DCs) has risen as an attractive vector primitive for representing complex colour gradients. Its flexible mathematical definition, taking curves with colour values as input, can be easily adopted by artists and designers because curves represent an intuitive approach to 2D drawing and design. However, the (Laplacian) diffusion process is computationally expensive and naive DCs (solving the large sparse PDE naively) are consequently unattractive as a practical vector graphics primitive. Lots of work has therefore been undertaken to identify a practical framework for defining and rendering DCs.

A class of such frameworks aims to eliminate the need for the diffusion process and instead explore alternative approaches to evaluating DCs, such as ray tracing [Bowers et al. 2011]. The hypothesis is that by eliminating the diffusion process, the computational complexity is reduced and that the flexibility of DCs remain by using an alternative evaluation method. Recently, [Prévost et al. 2015] demonstrated that acceptable performance can be achieved using an intermediate triangular representation for evaluating ray-trace-based DCs [Bowers et al. 2011]. While their framework resolves the performance aspect of the hypothesis, it is less flexible compared to state-of-the-art diffusion-based methods, which employ bi-Laplacian diffusion. More specifically, the framework of Prévost et al. does not support diffusion points and colour gradient control. See [Finch et al. 2011] for a full discussion on the advantages of diffusion points in the setting of the DC primitive.

SIGGRAPH '16, July 24-28, 2016, Anaheim, CA,
ISBN: 978-1-4503-4371-8/16/07
DOI: http://dx.doi.org/10.1145/2945078.2945085

In this poster, I present a ray-trace-based DC framework, similar to that of Prévost et al., with the additional support for diffusion points (Figure 1). Further, the possibility of using spline surfaces to represent colour gradients in the local neighbourhood of the diffusion points is explored.

2 Ray-traced diffusion points

In the following, I present the following components: formulation, rendering, and local control.

2.1 Formulation

The formulation, defining the proposed vector primitive, includes diffusion points into the ray-trace-based formulation of Bowers et al. [2011]. The proposed formulation defines the colour Z at an arbitrary point p in the image plane as follows:

$$Z(p) = \frac{1}{N(p)} \int_{\Omega} w\,(p, p^*)\, C\,(p^*)\, d\theta + \frac{1}{N(p)} \sum_{i=1}^{n} w\,(p, L_i)\, C\,(L_i)\, V\,(p, L_i)\,; \tag{1}$$

where $N(p)$ is the normalisation term:

$$N(p) = \int_{\Omega} w\,(p, p^*)\, d\theta + \sum_{i=1}^{n} w\,(p, L_i)\, V(p, L_i);$$

and L_i represents the geometric coordinates of the i^{th} diffusion point; there are n such diffusion points. The term $V(p, L_i)$ denotes the visibility between the point p and the diffusion point L_i. That is, if there are no curve points occluding the diffusion point, $V(p, L_i) = 1$ and the contribution from it is taken into account. Visibility $V(p, L_i)$ is determined via ray tracing.

The point p^* is the closest point in the intersection between the ray $R(t) = p + \begin{pmatrix} t\cos\theta \\ t\sin\theta \end{pmatrix}$ and the set of curves D (the input DCs). That is, $p^* \equiv R(a) \cap D$, where a is the lowest positive value possible (including 0). The angle, θ, is in the interval $\Omega = [0, 2\pi]$.

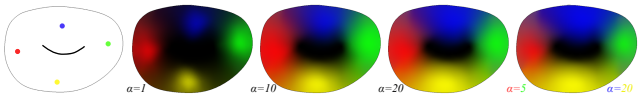

Figure 2: *The weight α in Eq. 2 is used to alter the relative influence of the input data. In this example, there are four diffusion points and one diffusion curve, where the weights for the diffusion points are changed; the weight for the curve is 1. The rightmost case demonstrates that different weights can be set for the diffusion points, where the blue and yellow points are associated with high weights.*

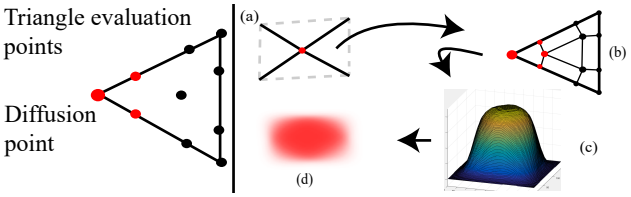

Figure 3: *Left: The red evaluation points are assigned with the colour of the related diffusion point, while the black evaluation points are evaluated according to Eq. 1. Right:* ***(a)*** *The user can add lines to diffusion points, representing the local propagation of the colour of that point. These lines act as constraints to the CDT; the dashed lines show the added lines in the triangulation.* ***(b)*** *The triangle mesh is subdivided to a denser mesh. Each triangle is subdivided to six quads surrounding a smaller triangle.* ***(c)*** *The subdivided mesh is rendered as a Catmull-Clark subdivision surface and* ***(d)*** *defines the local colour of the diffusion point.*

Note that p^* is determined via ray tracing in practice. Finally, the term C denotes the colour coordinates of points.

The weighting function w:

$$w(p_1, p_2) = \frac{1}{1 + \alpha_{p_2} \|p_1 - p_2\|^2}; \tag{2}$$

is used to control the influence of diffusion points relative to diffusion curves, as illustrated in Figure 2.

2.2 Rendering

The discretisation and rendering of the DC framework is similar to that of [Prévost et al. 2015]. The domain defined by the diffusion curves and points is discretised into a triangle mesh using the constrained Delaunay triangulation (CDT). Evaluation points are created for each triangle to sample Eq. 1. These points are used to render the final image on the GPU, according to the method of Prévost et al.

Due to the weight normalisation in Eq. 1, the colours of pixels close to a diffusion point will have colour values *close to* their specified colours, but are not guaranteed to be exactly identical. In my implementation, colour interpolation is enforced by assigning the evaluation points surrounding a diffusion point the colour of that point; see Figure 3(left) for an illustration.

2.3 Local control

A missing attribute for general DCs is colour gradient control. That is, local control of the colour propagation surrounding diffusion curves and points. Inspired by *local diffusion curves* proposed by Prévost et al., similar *local diffusion points* have been explored. A local diffusion curve does not influence the global formulation and is instead separated into an intermediate image layer. This layer is combined with the underlying image produced by the global formulation using blending.

A Catmull-Clark subdivision surface is constructed directly from the optional vector constraints associated with the diffusion points (see Figure 1 for an illustration). The process is illustrated in Figure 3(right). First, an initial triangular mesh is created with the CDT using the vectors as constraints. Each triangle is then split into a triangle and six quads. The placement of the new mesh points is determined using barycentric coordinates, with barycentric weights set to $1/3$ and 0, defining new patches of the same proportion. Finally, the subdivision surface is rendered using Pixar's OpenSubdiv library [Nießner et al. 2012] and is linearly blended with the global formulation, as local DCs in [Prévost et al. 2015].

3 Discussion and conclusion

In general, the performance of the procedure that calculates the contribution from the diffusion points is proportionally dependent on two factors: the number of diffusion points n and the number of curves that is ray traced to determine the visibility term V. In the special case when all curves are set as barrier curves, the formulation in Eq. 1 is reduced to the summation of the contribution from the diffusion points, since $w(p, p^*) = 0$. Thus, the integral is eliminated and there is no need to invoke the Monte Carlo integration procedure. This special case brings an obvious performance boost since the number of diffusion points is usually much lower than the number of samples used to approximate the integral. On my implementation, rendering times dropped one order of magnitude (from about 50 ms to 5 ms) when the integral was not computed.

Because the formulation is fundamentally different to bi-Laplacian diffusion, the proposed method possesses different behaviour compared to that of previous methods [see figure in poster]. Due to the normalisation term, my method possesses the convex hull property in colour space. Thus, the resulting colour gradient does not stray past specified colours. By contrast, bi-Laplacian diffusion does not possess this (maximum principle) property and can therefore stray outside the convex hull of specified colours.

References

BOWERS, J. C., LEAHEY, J., AND WANG, R. 2011. A ray tracing approach to diffusion curves. In *Proc. EGSR*, Eurographics Association, 1345–1352.

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

NIESSNER, M., LOOP, C., MEYER, M., AND DEROSE, T. 2012. Feature-adaptive GPU rendering of Catmull-Clark subdivision surfaces. *ACM TOG 31*, 1, 6:1–6:11.

ORZAN, A., BOUSSEAU, A., WINNEMÖLLER, H., BARLA, P., THOLLOT, J., AND SALESIN, D. 2008. Diffusion curves: A vector representation for smooth-shaded images. *ACM TOG 27*, 3 (Aug.), 92:1–92:8.

PRÉVOST, R., JAROSZ, W., AND SORKINE-HORNUNG, O. 2015. A vectorial framework for ray traced diffusion curves. *CGF 34*, 1, 253–264.