INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Controllable Diffusion Curves

Hedlena Bezerra — Elmar Eisemann — Doug DeCarlo — Joëlle Thollot

N° 7175

Janvier 2010

ISSN 0249-6399 ISRN INRIA/RR--7175--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
GRENOBLE - RHÔNE-ALPES

Controllable Diffusion Curves

Hedlena Bezerra*, Elmar Eisemann†, Doug DeCarlo ‡ , Joëlle Thollot *

Thème : Acquisition, Representation and Transformations for Image Synthesis
Équipes-Projets Artis

Rapport de recherche n° 7175 — Janvier 2010 — 16 pages

Abstract: This report introduces new drawing tools for controlling color diffusion in an image. It builds on the Diffusion-Curve algorithm, which computes images by diffusing colors from user-defined constraint curves. However, control of the diffusion process was limited in significant ways. In this report, we adapt the representation of Diffusion Curves by allowing artists to specify color strength, diffusion directions, and non-diffusing barriers. We also describe the algorithmic changes necessary for their efficient implementation. We demonstrate how these extensions give artists more control, and streamline the editing process in common situations.

Key-words: Poisson equation, color diffusion, expressive rendering.

* Grenoble Université - LJK - INRIA Rhône-Alpes
† Télécom ParisTech / CNRS-LTCI
‡ Rutgers University

Centre de recherche INRIA Grenoble – Rhône-Alpes
655, avenue de l'Europe, 38334 Montbonnot Saint Ismier
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

Courbes de diffusion contrôlables

Résumé : Dans cet rapport nous présentons plusieurs outils de controle de la diffusion de couleurs dans une image. Ces outils sont mis en oeuvre sous forme d'extensions de l'approche "courbes de diffusions" qui consiste à calculer une image à partir de contraintes placées le long de courbes.

Nous proposons ici d'étendre la représentation par courbes de diffusion pour permettre à un artiste de définir localement la puissance de la diffusion, sa direction ainsi que des courbes barrières qui peuvent stopper la diffusion. Nous décrivons de plus les modifications algorithimiques nécessaires pour une implémentation efficace.

Mots-clés : Équation de Poisson, diffusion de couleur, rendu expressif.

1 Introduction

The process of producing traditional 2D freehand drawings can be roughly divided into two major stages: the drawing process and the painting process. Once drawings are traced onto the canvas, colors are applied in the resulting areas of the image. Although filling regions with solid colors can be very expressive, complex color gradients more effectively convey the illusion of volume and illumination, enabling the depiction of a broad range of effects.

Raster images are a very suitable means to encode images depicting color gradients since all the color information is explicitly stored per pixel. Although they are very flexible, they require a large amount of storage and considerable effort to be edited. On the other hand, vector graphics images perform better in terms of manipulation and storage. In particular, the recent introduction of Diffusion-Curve images by Orzan et al [OBW*] allows for the creation of expressive color gradients. This work extends the Diffusion Curve approach with richer color gradients and streamlined color constraints.

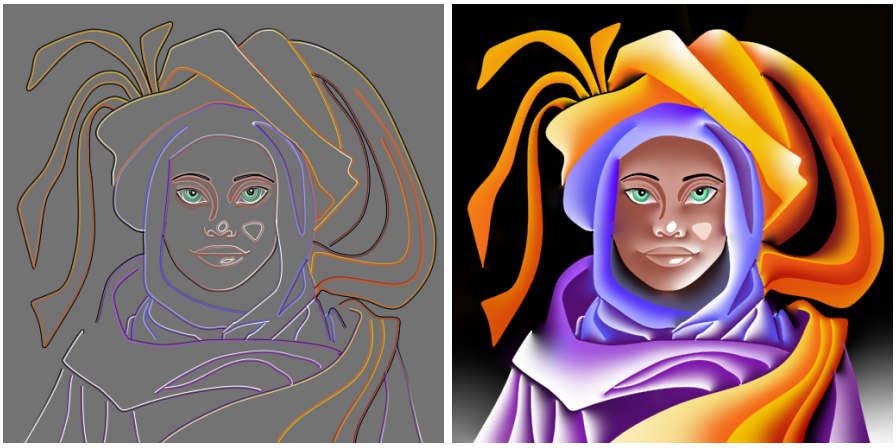

Figure 1: Left: Image encoded by curve primitives. Right: Image after diffusion process.

Technically, a Diffusion-Curve image is defined by a set of Bézier curves. For each curve, the user specifies colors along each of its sides (Figure 1 left). These colors are diffused outwards to define the final image (Figure 1 right). By specifying a blur value along the curve, an artist can also create smooth color transitions across curve boundaries. The diffusion curves themselves hold an intuitive meaning and are very compact since the colors and blur values are only specified at the control points of the curves.

Despite many advantages, Diffusion Curves also have some drawbacks. One problem relates to the diffusion process itself. Inherently, Diffusion Curves are bound to the standard Poisson equation, producing linear color gradients that arise from the diffusion process. In many situations, perfect symmetry and linearity are not desired, such as for the sunset in Figure 2. In this example, the dark brown color at the bottom of the image is diffused through nearly half of the image until it blends with the bright beige line under the mountains. Being able to express this and many other color gradients requires an approach

that provides the user with control over how strongly a given color must be spread. In the same spirit, it is very common in illustrations to define an anisotropic diffusion along some preferred direction, whereas Diffusion Curves always propagate colors in all directions equally. In our work, we offer this possibility, which allows designers to achieve images that otherwise would have been very laborious to produce with standard diffusion.

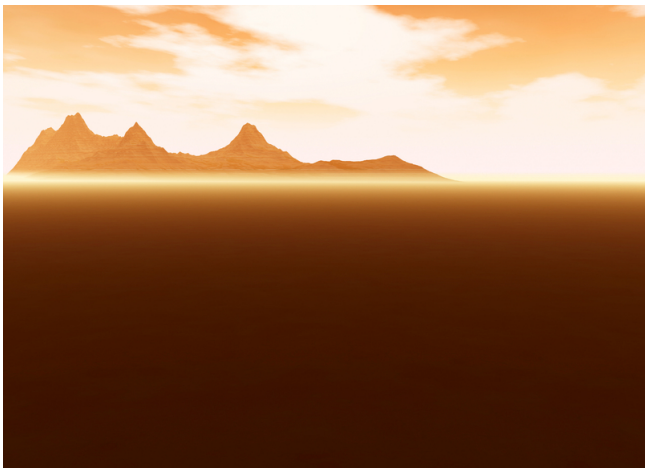

Figure 2: Complex color gradient in a sunset image. ©Bobbigmac | Dreamstime.com

Another important issue addressed in this report relates to the inflexibility of the relation between curves and colors. As explained above, for each Diffusion Curve, one needs to define color values for *both* sides of the curve. These double-sided constraints often oblige users to select colors at awkward locations, which produces unwanted side-effects as shown in Figure 3. In this example, the black line defined on the exterior side of the hat impacts the color of the ribbons at that location. In order to avoid such artifacts, the user must select colors along the intersection between hat and ribbons. Furthermore, the choice of the right colors is difficult, since they need to match the diffused region accordingly. A better solution would enable the diffusion process to determine some of the colors directly. In other words, the curve in question would, on one side, define colors in the interior of the hat, but, on the other side, simply be used as a barrier to prevent the ribbon and hat colors from mixing. In this way, the application of color would approach again the traditional method of coloring, and give the appearance that one part of an object is in front of another.

In this report, we propose several important extensions to the Diffusion Curves approach in which we address the aforementioned problems. Our solution provides the user with a more flexible and expressive tool. More precisely, we introduce:

- **color strength** which translates into a dominance during the diffusion process and enables colors to expand more strongly across the image;
- **diffusion barrier curves** block diffusion across the curve without defining the color—this provides the user with more flexibility;

Figure 3: Side-effects of double-sided constraints: the black line defined in the exterior side of the hat has a clear impact on the color of the ribbons.

- **diffusion directions** to effect anisotropic diffusion.

This report is organized as follows: we start by reviewing related work in Section 2. Our algorithm, and our extensions to Diffusion Curves, will be detailed in Section 3. Implementation details and results are presented in Section 4. Finally, we conclude and point out directions for future improvements in Section 5.

2 Previous Work

In an image context, lines are often important features and encode much of the initial scene information, which is exploited by many vision-related algorithms [Eld99]. Further, it is possible to compress images [EZ98] and manipulate photographs [EZ, OBBT]. These task often underly the principle of the Poisson equation which has found applications in many contexts of computer graphics, e.g., image editing, where it enables seamless cut-and-paste operations [PGB].

The Poisson equation also laid the groundwork for Diffusion Curves [OBW*] and real-time gradient domain painting [MP], where colors are sparsely defined over the image and interpolated everywhere else. Although both implementations were very efficient, few control over the diffusion is giver to the user. In addition to that, the final convergence of the solution was not always guaranteed and spurious artifacts could remain for an insufficient number of iterations. A better convergence behavior is achieved with the approach by Jeschke et al. [JCWa] which introduces a faster and more accurate solver by exploiting the fact that the constraints are very sparse. Our work is strongly inspired by the aforementioned approaches. Nevertheless, we aim for a more flexible tool to provide the user with useful controlling over the content creation.

Diffusion processes also can be beneficial to interpolate other values such as normals [Joh] (to enable relighting), or even general surface details [JCWb] (for real-time contexts). These approaches can benefit from the extensions proposed in this report to achieve a stronger control over the interpolation.

Finally, reconstructing images by diffusion can be seen as a way to opt for resolution independence. This aspect is shared by Gradient Meshes [SLWS07, LHM09] and Ardeco [LL], which derive vector graphics from a 2D image, but often prove less user-friendly than Diffusion Curves.

3 Our Algorithm

In the following, we address the identified Diffusion-Curve shortcomings of Section 1. We familiarize the reader with the mathematical background in Section 3.1. Section 3.2 introduces our first contribution, which is an efficient mechanism to allow colors to be expanded according to a strength parameter. We then show the well-known formulation of the Poisson equation as a minimization problem (Section 3.3). This facilitates understanding the diffusion process interactions, diffusion barriers and anisotropic diffusion, in Sections 3.4 and 3.5 respectively.

3.1 Diffusion Process

Our work builds upon the Poisson equation framework previously applied in many contexts [TT, PGB, OBW*]. Consider an image I with n pixels, $\{I_k \,|\, k \in 1 \dots n\}$ (colors are addressed individually as I_k or as a grid simply as $I_{i,j}$). For Diffusion Curves, colors are specified along each side of the curve serve as hard constraints. This yields a set of constrained pixel colors $\{C_k \,|\, k \in \mathcal{I}\}$, where $\mathcal{I} \subseteq \{1 \dots n\}$ are the indices of pixels with constrained colors. The remaining parts of the image are filled by solving the Poisson equation resulting in the final image I, where

$$\begin{aligned} \triangle I &= \operatorname{div} w, \\ \text{and } I_k &= C_k, \ \forall k \in \mathcal{I}, \end{aligned} \tag{1}$$

where $\triangle$ is the Laplace operator, div is the divergence operator, and $w = \{w_k \,|\, k \in 1 \dots n\}$ a vector field (its values are also stored in pixels and addressed, just like for I with $w_k := (w_k^x, w_k^y)$). In the case of Diffusion Curves, the vector field w is zero everywhere except across constraint curves [OBW*]. In other words, the solution will show a continuous, smooth change of colors except across the curves. The solution is found by solving a discretized version of Equation 1 for each color channel separately. A Gauss-Seidel solver could be used, but more efficient conjugate gradient or multi-grid solvers (as used for Diffusion Curves) are an option. In the case of Gauss-Seidel iterations, a value $x_{i,j}$ needs to be updated by adding $(I_{i+1,j}+I_{i-1,j}+I_{i,j+1}+I_{i+1,j+1}+\operatorname{div} w_{i,j})/4$.

3.2 Color Strength

Our first extension to Diffusion Curves is the control of the strength with which a color dominates the diffusion process. Figure 4 depicts a simple example with two color constraints, yellow on the top and purple at the bottom. The result of a standard Poisson diffusion leads, as expected, to a linear gradient that connects both color constraints (Figure 4, left). By manipulating the color strength, the artist can make the yellow color become more dominant over the purple, thus pushing the diffusion further in this direction (Figure 4, center). A variety of effects can be obtained if different values of color strength are defined along lines as, for example, the diagonal diffusion effect in Figure 4, right.

One way of controlling the strength of colors during diffusion is to formulate this problem as an interpolation process. Intuitively, if we have two colors c_1, c_2 with respective strengths a_1 and a_2, then we would like the interpolation T of

Figure 4: Left: Standard Poisson diffusion (equal strengths, effectively). Center: Yellow has a greater strength than purple. Right: Varying the strength along the curve.

the two to yield:

$$T\left((c_1, a_1), (c_2, a_2)\right) = \frac{a_1 c_1 + a_2 c_2}{a_1 + a_2}.$$

As we can see, the equation results in c_i for $a_i \to \infty$, and if $a_1 = 0$, the equation simplifies to a_2. Therefore, the values a_1, a_2 can be used to control the dominance of one color over the other. The result is a linear combination of the initial color values that naturally favors colors with a higher alpha value. This generalizes

$$T\left((c_1, a_1), \ldots, (c_k, a_k)\right) = \frac{\sum (r_i, g_i, b_i) a_i}{\sum a_i},$$

In some sense, when thinking of a blending process, the strength value will indicate the mixing coefficients. Due to the normalization, such a weighted blending is non-linear and, hence, would not fit into the diffusion framework. Nevertheless, it is possible to linearize the computations using *homogenous colors*.

A homogenous color is defined by a RGB-tuple (r, g, b) and an alpha value a. Algebraically they resemble homogeneous coordinates, widely used in projective geometry calculations [Wil]. Two homogenous colors (r_1, g_1, b_1, a_1) and (r_2, g_2, b_2, a_2) describe the same actual color when $a_2(r_1, g_1, b_1) = a_1(r_2, g_2, b_2)$. If a_i is not zero, the actual color is obtained via a projection mapping $P(r, g, b, a) = (r/a, g/a, b/a)$. It is easy to verify that the projection of the sum of homogenous coordinates correspond to the weighted sum above. When a_i equals zero it corresponds to an absence of color, and equivalence is irrelevant.

The key idea of our extension is that the alpha value of a color will define the color strength. In the interface, the user only specifies a standard color $c = (r, g, b)$ and a color strength a. This input is then transformed into a homogenous color by mapping it to (ar, ag, ab, a). Each channel (including the alpha channel) is thus diffused separately, exactly as in the Diffusion-Curves approach. At the end of the diffusion process, we perform the projection to obtain the final result. The correctness of this solution becomes clear when inspecting the way that the Gauss-Seidel iterations would update the values in the solver, where an average is computed in each step. The final projection then transforms the result into a weighted sum and the diffusion will reflect the weight.

Figure 5 shows a direct application of the color strength extension. In this example, complex color gradients can be achieved by manipulating the strength of the colors in the interior and exterior of the eye, which are expanded through nearby regions (Figure 5, right). Compared with the standard result (Figure 5,

left), we show that the color strength extension can lead to interesting results with no additional curves.

Figure 5: Left: Standard Poisson Diffusion. Right: Controlled diffusion

3.3 Reformulating the Diffusion Process

To facilitate the understanding of how to influence the diffusion process, we need to look a little closer at its properties. After solving Equation 1, the resulting image I is the solution to the following constrained minimization:

$$\begin{aligned} I &= \underset{J}{\operatorname{argmin}} \sum_{i=0}^{n} |\nabla J_i - w_i|^2, \\ &\quad \text{subject to } J_k = C_k, \quad \forall k \in \mathcal{I}, \end{aligned} \tag{2}$$

where C_k are the color constraints at the pixel positions $\mathcal{I}$, $w_i = (w_i^x, w_i^y)$ is the vector field w at pixel position i. Once again, it should be pointed out that w equals zero almost everywhere. In fact, for Diffusion Curves, the result is the same for $w = 0$ everywhere. For a better convergence though, it is of interest to initialize the vector field at neighboring color constraints by storing a divergence.

The solution is, thus, the result of a minimization process that searches for the image whose gradient is the best fit to a given vector field, while respecting the color constraints. In our work, we want to guide the diffusion process in various ways. Consequently, we cannot always rely on the original Poisson equation, but have to set up our own constraint system. To give a better intuition of how we will incorporate changes, we will rely on a formulation that describes our solution via a system of hard and soft constraints. The hard constraints ($I_k = C_k$) are the colors stored at pixel positions $\mathcal{I}$ and defined by the initial color curves chosen by the user. The soft constraints will enforce a smoothness property on the image, that will implicitly define the color of the remaining pixels.

The Poisson-equation soft constraints can be written as follows:

$$\begin{pmatrix}\nabla_x\\ \nabla_y\end{pmatrix}\begin{pmatrix}I_1\\ \vdots\\ I_n\end{pmatrix} = \begin{pmatrix}w_1^x\\ \vdots\\ w_n^x\\ w_1^y\\ \vdots\\ w_n^y\end{pmatrix}, \tag{3}$$

where ∇_x is a simple matrix that encodes the derivative along the axis x, (w_k^x, w_k^y) is the vector field w's value at pixel $k \in 1 \dots n$. Each line of ∇_x is of the form $(0, \dots, 0, -1, 1, 0, \dots, 0)$. ∇_y is defined accordingly. This matrix is the one that allows us to choose what properties we want our solution to have.

In general, the Equation system 3 is over-constrained. To find the least-squares fit, we use the pseudo-inverse. For this, the equation needs to be multiplied by (∇_x^t, ∇_y^t). The result of doing so is:

$$(\nabla_x^t, \nabla_y^t)\begin{pmatrix}\nabla_x\\ \nabla_y\end{pmatrix}\begin{pmatrix}I_1\\ \vdots\\ I_n\end{pmatrix} = (\nabla_x^t, \nabla_y^t)\begin{pmatrix}w_1^x\\ \vdots\\ w_n^x\\ w_1^y\\ \vdots\\ w_n^y\end{pmatrix}$$

$$(\nabla_x^t\nabla_x + \nabla_y^t\nabla_y)\begin{pmatrix}I_1\\ \vdots\\ I_n\end{pmatrix} = \nabla_x^t\begin{pmatrix}w_1^x\\ \vdots\\ w_n^x\end{pmatrix} + \nabla_y^t\begin{pmatrix}w_1^y\\ \vdots\\ w_n^y\end{pmatrix}, \tag{4}$$

where n is the number of pixels in the image.

The operator $(\nabla_x^t\nabla_x + \nabla_y^t\nabla_y)$ is the discrete version of the Laplace operator and, similarly, $(\nabla_x^t w^x + \nabla_y^t w^y)$ is the divergence. The lines in $\nabla_x^t\nabla_x$ have the form $(0, \dots, 0, 1, -2, 1, 0, \dots, 0)$ which corresponds to a discrete second derivative. The similar structure of $\nabla_y^t\nabla_y$ implies that the lines in matrix $(\nabla_x^t\nabla_x + \nabla_y^t\nabla_y)$ have the form:

$$4I_{i,j} - I_{i+1,j} - I_{i-1,j} - I_{i,j+1} - I_{i+1,j+1} = \operatorname{div} w_{i,j},$$

which readily corresponds to the discrete Poisson equation.

3.4 Diffusion Barriers

When we remove all constraints along a curve from the pixels underneath (hard, as well as soft constraints), the effect is that the diffusion is blocked at that location. Because no soft constraint is defined along these pixels, no value can be computed at these locations and also, no other pixel in the image will refer to their values. Therefore, no color can cross these pixels during the diffusion and, no color is actively emitted. This kind of curve is useful when the user desires to restrain the diffusion to reach a certain region without having to actually define any color at that location.

Figure 6 shows an example of the use of such a curve. Yellow and a pink color strokes are placed on the image (a). If no other curve is added, the pink and yellow strokes will be diffused and mix at certain locations (b). Nevertheless, if we place the circle curve as barrier curve (c), the diffusion of the yellow stroke will be restrained to its interior; analogous the pink to its exterior (d).

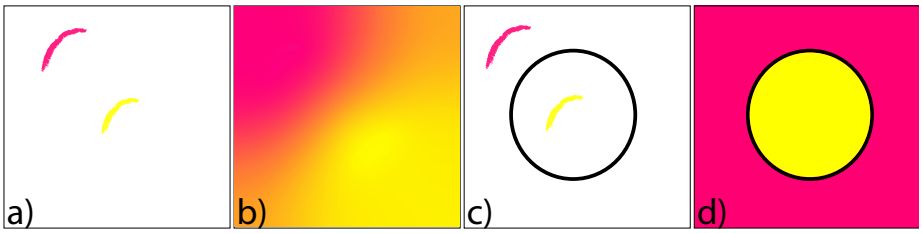

Figure 6: a) Pink and yellow strokes. b) Diffusion of pink and yellow strokes. d) Circle curve defines a diffusion barrier. d) Diffusion blocked in the interior and exterior of the circle (diffusion barrier).

It is also possible to only remove the constraints for one side of the curve. In this case, one side emits colors, whereas the other serves as a barrier to prevent colors from crossing the curve at that location. This is an important tool that lets the user avoid defining colors at awkward locations, as previously shown on Figure 3.

The proper way to realize such diffusion barriers is to rely on the linear system in Section 3.1. As explained before, we can simply omit all affected constraints along the curve. This will break the continuity between left and right sides of the curve allowing the system to compute each side independently.

Figure 7 shows an example of how diffusion barriers can be used in order to improve the result of standard Diffusion Curves presented on Figure 3. Figure 7 left indicates the lines where placing double-sided color constrained curves is problematic. To solve this problem, we transform these lines into diffusion barriers. For this, we keep the right side of the red line emitting colors to the interior of the hat but, on its left side, we remove all constraints thus preventing colors to cross the curve. Analogous, the interior side of the green line will emit its original colors, while its exterior side will work as a barrier. The result of the diffusion is depicted on Figure 7 right. Notice that colors diffused from the ribbons are now nicely expanded to the boundaries of the hat without discontinuity artifacts.

There is a also simpler mechanism that is compatible with the well-established and optimized algorithms for Diffusion Curves. We now demonstrate how to use the color strength described in Section 3.2 to imitate a similar behavior by setting it to zero. Indeed, a zero value indicates that the color is not participating in the final weighted sum, thus not appearing in the output image. Such a constraint curve still ensures that colors from both sides will not mix, breaking the connectivity between pixels on either side.

Figure 8 shows the use of color strength to imitate barrier diffusion curves and solve the same problem shown in Figure 3. Again, we refer to the red and green lines depicted on Figure 7. Giving a strength value of zero to left side of the red line, results that the ribbon colors touch the hat's boundary, but no colors cross the curve (see Figure 8 right). Notice that no color is emitted from

Figure 7: Left: Lines along which problems occurs. Center: Red and line curves are transformed into barrier curves emitting colors from only one of its sides. Right: Result of the diffusion.

this side of the red line (we can no longer see the black color in the diffusion). In the way, the exterior side of the green line received a strength value of zero preventing its black color from mixing at its intersection with the hat.

Although this is a simple solution, there is one subtlety. Using the diffusion strength, the solution is undefined on the curve itself, as a division by alpha would lead to a division by zero. Instead, we perform a simple pretest. The final alpha value corresponds to the sum of the weights of surrounding pixels. The larger the weight, the more confidence we can have that numerical issues will be avoided. Hence, our solution is to use a weighted sum in a window around such a pixel whenever its actual weight is lower than 1/255, which ensures a smooth transition.

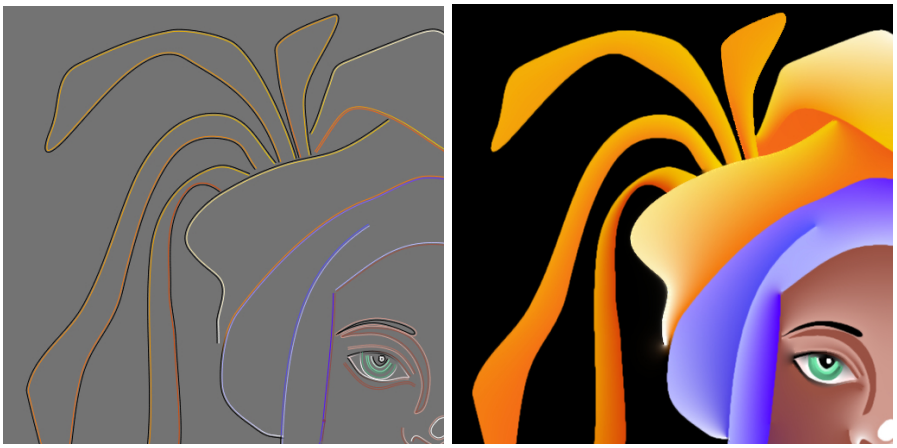

Figure 8: Left: Original color constraints from Diffusion-Curves approach. Right: Diffusion using zero values for weights.

3.5 Anisotropic Diffusion

We have seen in Section 3.3 that the diffusion process is the least squares solution to a linear system. Each row of the equation matrix corresponds to a constraint.

In contrast to a solvable system, a least squares fit is influenced by the norm of each row. Scaling a line with a small number, will decrease its influence on the final solution. In fact, it will participate less. This allows us to steer the influence of the constraints in certain regions of the image.

Let's look at a simple example. The soft constraints ensured the smoothness of the resulting diffusion surface. We have seen that each pixel has a row in the matrices ∇_x and ∇_y which enforces a smoothness along the corresponding axes. Leaving out the row in ∇_x would lead to a diffusion along the y-axis. This is a direct consequence of the fact that x-axis differences are no longer penalized.

In practice, to define a general anisotropic diffusion, we enable the artist to define two orthogonal directions in which the diffusion should take place, as well as the corresponding weight. Consequently, we can simply weight the corresponding equations.

For a given direction $\vec{d} := (\cos\theta, \sin\theta)$, the gradient is defined as $\cos\theta\frac{\partial}{\partial x} + \sin\theta\frac{\partial}{\partial y}$. Correspondingly, the discretized version would read:

$$\cos\theta(x_{i+1,j} - x_{i,j}) - \sin\theta(x_{i,j+1} - x_{i,j})$$

In the same way, we can define the orthogonal direction. The discretized version shows us how to setup the rows in the matrix. The row is then scaled by the user-defined weight to indicate the strength of the diffusion along the corresponding axis.

The direction itself can be diffused over the image plane in order to define in each pixel its final direction. In practice, we diffuse the cosine and sine values of the direction, as well as the weights along the principal axes. Other schemes could exist, but because the feedback is immediate, potential problems, such as gradient reversals or poles can be fixed.

Figure 9 shows a simple example. The top of the image has been initialized to the standard diffusion, whereas the bottom makes use of a directional diffusion along the y-axis. In comparison to standard diffusion, the directional diffusion clearly lets the color constraint at the bottom affect the scene much more. For standard diffusion the relatively small mountain tops directly propagate their color between each side giving it a flat appearance, whereas for directed diffusion, the light ground propagates its values to the mountain tips, hence achieving a richer appearance.

4 Implementation and Results

We implemented our solution in OpenGL 3.0 and executed the program on various machines, ranging from laptops with a Geforce 8400M to a desktop with a GTX260. Our solution achieves interactivity on all platforms for most tasks (except higher-order constraints). For example, the use of diffusion strength only adds a penalty of $< 30\%$ with respect to Diffusion Curves. This facilitates the use of our software as a tool to produce convincing imagery.

To recapture our main contributions and illustrate the suitability of our solution, we illustrate its application on an example illustrated on Figure 10. Figure (a) was obtained with standard Diffusion Curves. The given color constraints are used to render the image of a cat (b), where each line of the drawing requires the definition of colors on each side of the curves. The double-sided constraint

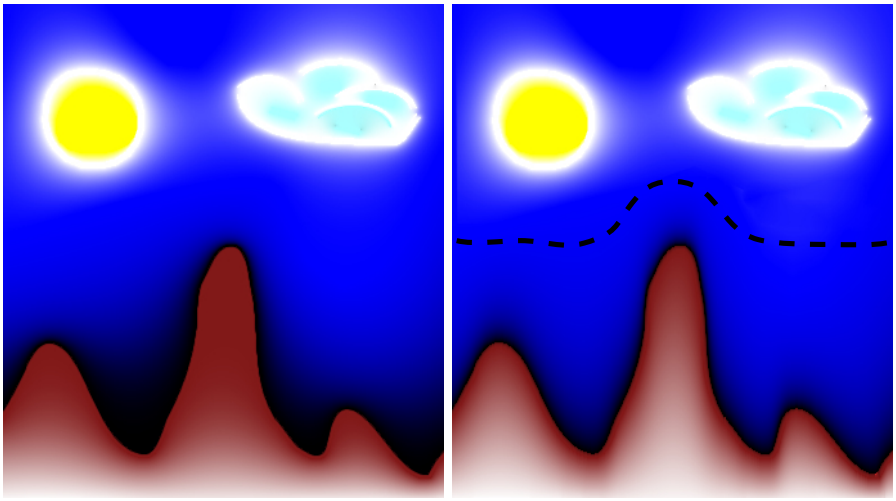

Figure 9: Left: Result with standard diffusion. Right: The image on the the top of the dashed line makes use of standard diffusion, the bottom uses a vertical direction.

curves required by this approach are due to the Diffusion Curve process. Indeed, just leaving out the color constraint on one side, as shown on most lines of (c), results in a blurred and artifact-filled result (d). Defining all these curves and adjusting the colors can be very tedious for an artist.

In our solution, only very few color constraints need to be defined. Most curves can be kept as a diffusion barrier, implying less effort for the artist. With diffusion barriers, colors simply fuse at their boundary (e). To be obliged to predict the fused colors and then adjust the boundary color itself is a difficult task. Further, one loses flexibility because the colors on the boundary are then set and will not change, even if the actual color sources are moved. The result using our solution looks very similar, even though only half the color constraints had to be defined.

Controlling the diffusion strength is a key component and was used in almost all images of this report. In the case of (f) the control allows us to produce a much more delicate look, where the fine details no longer blur out. This also underlines one of our claims, which was that control over diffusion strength, in combination with diffusion barriers, facilitates the creation of shaded looks.

5 Conclusions and Future Works

We presented several extensions to the original Diffusion-Curves approach. We introduced a simple solution to control the diffusion strength that remains compatible with the original implementation and thus benefits from efficient implementations. We introduced diffusion barriers that bring Diffusion Curves closer to the traditional drawing framework and makes the interaction more intuitive and more efficient. We presented several results that would have been difficult to reproduce with Diffusion Curves. We illustrated that it is possible to control the diffusion directions to achieve more control over the final result.

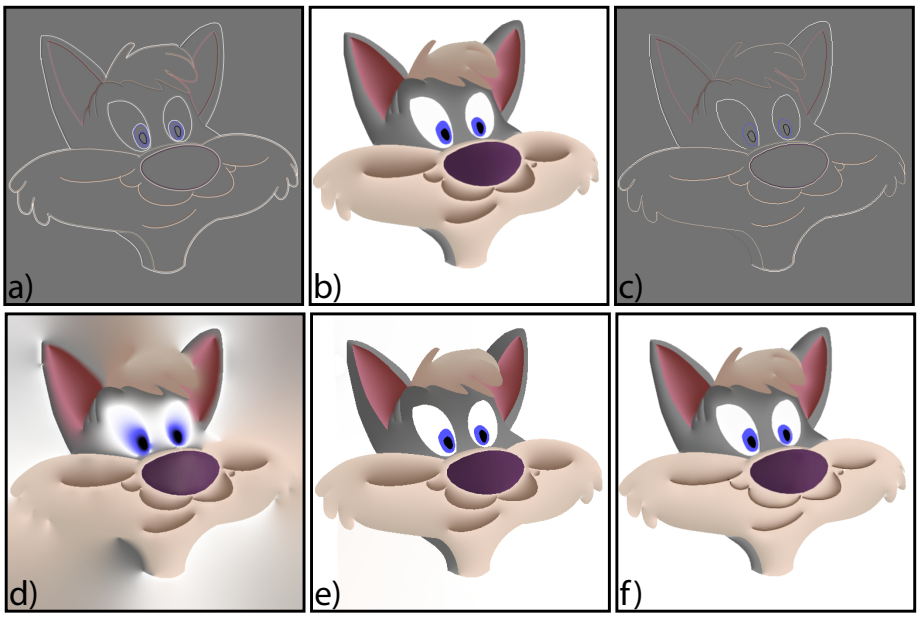

Figure 10: a): Diffusion Curves requires colors constraints to be defined at both sides of each curve. b): Diffusion performed by Diffusion Curves. When most curves define colors at only one of its sides (c), the result is a blurred and artifact-filled image (d). e) With diffusion barriers, colors simply fuse at their boundary producing a continuous color gradient. f): The color strength allows us to produce a much more delicate look, where the fine details no longer blur out.

The extensions presented in this report directly translate to a multigrid solver. Nevertheless, we believe that the general linear solver framework gives us more flexibility and might prove useful in the future. Currently, the computational cost is too high (Figure 10 (d) required nearly 55 seconds per color channel for a 512×512 image with a GPU linear solver implemented in CUDA on an NVIDIA 8400M graphics card). Obvious accelerations exist to improve the performance by specializing the solver for our purposes and we see this as one interesting avenue of future work.

References

[Eld99] Elder J. H.: Are edges incomplete? *International Journal of Computer Vision 34*, 2-3 (1999), 97–122.

[EZ] Elder J. H., Zucker S. W.: Space scale localization, blur, and contour-based image coding. In *Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR 1996)*, pp. 00–27.

[EZ98] Elder J. H., Zucker S. W.: Local scale control for edge detection and blur estimation. *IEEE Trans. Pattern Anal. Mach. Intell. 20*, 7 (1998), 699–716.

[JCWa] Jeschke S., Cline D., Wonka P.: A gpu laplacian solver for diffusion curves and poisson image editing. In *Proceedings of SIGGRAPH Asia 2009*, pp. 1–8.

[JCWb] Jeschke S., Cline D., Wonka P.: Rendering surface details with diffusion curves. In *Proceedings of SIGGRAPH Asia 2009*, pp. 1–8.

[Joh] Johnston S. F.: Lumo: Illumination for cel animation. In *International Symposium on Non-Photorealistic Animation and Rendering (NPAR 2002)*, pp. 45–ff.

[LHM09] Lai Y.-K., Hu S.-M., Martin R. R.: Automatic and topology-preserving gradient mesh generation for image vectorization. *ACM Transaction on Graphics 28*, 3 (2009), 1–8.

[LL] Lecot G., Levy B.: Ardeco: Automatic Region DEtection and COnversion. In *Proceedings of the 17th Eurographics Symposium on Rendering (EGSR 2006)*, pp. 349–360.

[MP] McCann J., Pollard N. S.: Real-time gradient-domain painting. In *Proceedings of SIGGRAPH 2008*.

[OBBT] Orzan A., Bousseau A., Barla P., Thollot J.: Structure-preserving manipulation of photographs. In *International Symposium on Non-Photorealistic Animation and Rendering (NPAR 2007)*.

[OBW*] Orzan A., Bousseau A., Winnemöller H., Barla P., Thollot J., Salesin D.: Diffusion curves: A vector representation for smooth-shaded images. In *Proceedings of SIGGRAPH 2008*, vol. 27.

[PGB] Pérez P., Gangnet M., Blake A.: Poisson image editing. In *Proceedings of SIGGRAPH 2003*, pp. 313–318.

[SLWS07] Sun J., Liang L., Wen F., Shum H.-Y.: Image vectorization using optimized gradient meshes. *ACM Transaction on Graphics 26*, 3 (2007), 11.

[TT] Tumblin J., Turk G.: Lcis: a boundary hierarchy for detail-preserving contrast reduction. In *Proceedings of SIGGRAPH 1999*, pp. 83–90.

[Wil] Willis P.: Projective alpha colour. In *Proceeding of Eurographics 2006*.

Contents

Centre de recherche INRIA Grenoble – Rhône-Alpes
655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399