Diffusion Constraints for Vector Graphics

Hedlena Bezerra, Elmar Eisemann, Doug Decarlo, Joëlle Thollot

To appear in NPAR 2010

Diffusion Constraints for Vector Graphics

Hedlena Bezerra* INRIA - Grenoble Univ.
Elmar Eisemann Telecom ParisTech/CNRS LTCI
Doug DeCarlo Rutgers University
Joëlle Thollot* INRIA - Grenoble Univ.

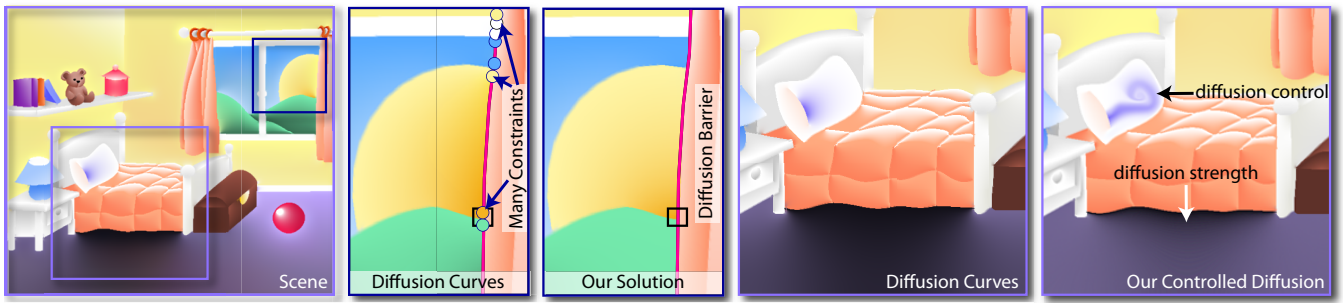

Figure 1: *Diffusion Curves allow us to draw vectorial images with a rich set of color gradients (left). It is based on a diffusion process that propagates color information from curves in the scene. While the colors could be chosen arbitrarily, the diffusion itself is not controllable by the user. Our work introduces ways to alter diffusion behavior. This allows us to reduce the number of color definitions for an equivalent output (middle), to control the diffusion strength of certain colors (right, floor), or even influence diffusion directions (right, cushion).*

Abstract

The formulation of Diffusion Curves [Orzan et al. 2008] allows for the flexible creation of vector graphics images from a set of curves and colors: a diffusion process fills out the parts of the image that are away from curves. However, this model has limitations in certain situations and does not always seem to agree with how an artist wants to use the software. First, the diffusion itself cannot be controlled, only the colors. Further, the fact that color needs to be defined everywhere along the curve can lead to tedious and nonintuitive interactions. In this paper, we present a number of adaptations to diffusion curves that constrain how color is spread across the image. Specifically, we argue for the utility of controlling the speed and direction of the color diffusion, and the ability to have barriers that can be defined without the need to specify a particular color along these curves. We also describe how this can be implemented by solving a linear system, and demonstrate the effectiveness of our solution on a number of examples.

1 Introduction

Today, we rely on many different devices to display information that range from small hand-held devices to large projector screens. Consequently, producing scale-independent content, such as vectorial images, becomes important. Further, for artists it is often beneficial to work with vector-based applications because objects can be described directly via primitives that represent the shape instead of having to work with pixels. Consequently, manipulations such as deformations are less prone to artifacts and do not degrade the object representation as is often the case for raster illustrations.

While vectorial images can adapt to the screen resolution and produce an adequate image, some challenges remain. Traditional vector graphics tend to look less detailed than their pixel-counterparts. The advent of more advanced color fills helped to close this gap. Initially distinguishing only between linear or radial color gradients, new solutions like gradient meshes (Adobe Illustrator©, Corel CorelDraw©) are capable of representing almost photo-realistic images by interpolating colors on a quad mesh. Unfortunately, the introduction of such a grid adds more complexity to the vector illustration, trading off some of its editing advantages against a richer representation, and making automatic conversion methods the preferred method of creation [Sun et al. 2007; Lai et al. 2009].

Diffusion curves [Orzan et al. 2008] combine curve primitives with a diffusion method that smoothly spreads color from the curves across the image. The fact that the representation relies on a small number of simple entities makes it particularly well-suited for artists. The diffusion process allows for highly expressive results [McCann and Pollard 2008]. This observation was also made by Johnston [2002] who successfully applied similar techniques in the context of cel animation.

Diffusion is the key component that enables a rich, yet simple definition of resolution-independent illustrations. Nevertheless, previous drawing systems did not allow the user to control the diffusion process, which limited expressivity and could make some operations cumbersome. In this work, we address this limitation by enabling more control over the diffusion process itself. To illustrate the importance of this control, we will look at some examples.

Johnston [2002] pointed out that not all curves should diffuse values to both sides. In particular, along occlusion boundaries, diffusion should typically only occur on the occluding part. To handle these exceptions, a blending mask is manually created that limits the extent of diffusion. On the other hand, diffusion curves [Orzan et al. 2008] do not offer this kind of control, and many illustrations exhibit unwanted halos or modifications of the color gradients, because the artist is forced to specify diffused colors on both sides of each curve, even if there is no need for them. An example is illustrated in Figure 1 (middle). The curtain needs color constraints even on the occluding boundary, leading to a large number of additional color constraints whose presence affected the radial appearance of the sun's gradient. Instead, our system allows the definition of *diffusion barriers* that block diffusion without emitting colors. No color

*ARTIS (INRIA Rhône-Alpes / LJK Laboratoire Jean Kuntzmann)

constraints need to be defined on the blocked side, which avoids the inconsistencies, as illustrated in the inset, that can arise from such unnecessary color constraints that do not perfectly match up with the geometry. Diffusion barriers can also be used to define shapes that can simply be filled with colors via diffusion curves, that are applied similarly to a paint-bucket tool.

Another example is a gradient where colors from different places intervene with differing strengths. Figure 1 (right) illustrates how the shadow underneath the bed also strongly influences the surrounding floor. Controlling the color strength directly provides influence over the impact of a color. Previously, the artist was obliged to place additional curves to simulate a non-linear diffusion behavior. Not only is this tedious, but any color change also implied that all these curves would need to be adapted. Influencing the diffusion via color strength is independent of the associated colors.

We also introduce a control over diffusion orientation which helps to guide color locally (almost like a smearing tool). It allows complex shapes and color variations that cannot be achieved with the uniform diffusion available from previous work. In Figure 1 (right) this technique was used to create a complex pattern on the cushion.

Finally, in some situations, one needs to use a color resulting from the diffusion process at a different location of the scene; e.g. if a shape represents a thin occluder and one wants to guarantee a smooth color transition on the occludee. Figure 2 shows an example and Figure 3 shows the entire teaser image after applying our diffusion extensions.

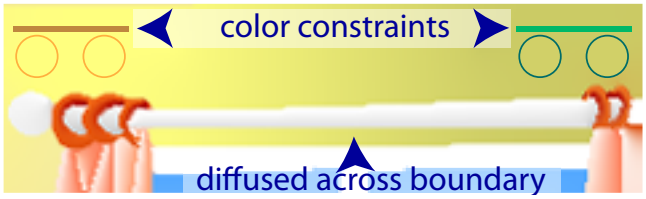

Figure 2: *The artist decided to add a color gradient to the wall. Usually it would be blocked by the curtain rod, but by virtually connecting the two regions, colors are transferred from one side to the other. Further, the color gradient was conveniently defined in the interior region of the wall, similar to a paint-bucket fill. This was enabled by setting some boundaries to diffusion barriers.*

Our technique avoids such problems by enabling control over the diffusion process and related diffusion constraints while maintaining the simplicity of the original Diffusion Curves. The new degrees of artistic freedom allow for further expressivity and enable an intuitive design of complex illustrations and color gradients. Precisely, our contributions are:

- Diffusion barriers that block diffusion (but do not emit color)
- Control over diffusion anisotropy and orientation
- Control over diffusion strength (speed)
- A generalized solution method for non-local diffusion

2 Previous Work

Diffusion processes often underlie methods for solving the Poisson equation, as its solution minimizes the deviation from a given gradient field. Its 2D formulation has found applications in many contexts, including image compression [Elder and Zucker 1998], manipulation of photographs [Elder and Zucker 1996; Orzan et al. 2007], seamless cut-and-paste operations [Pérez et al. 2003], or alpha matting [Sun et al. 2004].

Figure 3: *The figure shows the teaser image that was modified using the diffusion curve extensions presented in this paper. Our solution offers more control over the diffusion process and makes several tasks simpler.*

The Poisson equation is also the basis of Diffusion Curves [Orzan et al. 2008] and real-time gradient domain painting [McCann and Pollard 2008], where colors are sparsely defined along curves in the image and interpolated everywhere else. Jeschke et al. [2009a] introduced a faster and more accurate solver by exploiting the fact that the constraints are very sparse. Also, triangulation-based solutions [Farbman et al. 2009] are interesting alternatives to pixel-based diffusions. Although all these solutions are efficient, the diffusion is always uniform. Our work is strongly inspired by the aforementioned approaches, but we aim for a more flexible tool that provides the user with more control over the diffusion process.

Diffusion processes also could be beneficial to interpolate other values such as normals [Johnston 2002] (to enable relighting), or even general surface details [Jeschke et al. 2009b] (for real-time contexts). These approaches can benefit from the extensions proposed in this paper as well.

There are a number of techniques that achieve resolution independence, of which reconstruction using diffusion is just one possibility. The Ardeco system [Lecot and Levy 2006] simulates complex shading via local approximations using linear or quadratic gradients. Since it is an image conversion process, the results may contain a very large number of regions. A different approach known as gradient meshes is an artistic tool available in commercial software. It enables a user to specify colors at the vertices of a (planar) quadrilateral mesh; these colors are then interpolated. The creation can be tedious and Sun et al. [2007] and Lai et al. [2009] propose to assist the user by automatically optimizing a gradient mesh according to a given input image. Nevertheless, no control is given over color interpolation and many vertices (and patches) are necessary to produce complex shading.

3 Mathematical Background

Before presenting our algorithm, we will discuss interpolations based on the Poisson equation, which is the principle that underlies Diffusion Curves (Section 3.1). We reformulate the relationship

into a constrained linear system that will provide the basis for our work (Section 3.2). We present how to modify the system to support diffusion barriers, curves that block the diffusion processes without emitting colors (Section 4.1). We show how to guide the diffusion process by orienting it according to a user-specified flow field (Section 4.2). We then add the possibility to control the strength of a color during the diffusion (Section 4.3) before addressing a non-local extension of the diffusion process to allow us, e.g., to transfer color from one part of the image to another (Section 5).

3.1 Diffusion Process

Our work builds upon the Poisson-equation framework previously applied in many contexts [Tumblin and Turk 1999; Pérez et al. 2003; Orzan et al. 2008]. 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 $I_{i,j}$). The goal is to derive an interpolant matching a set of constrained pixel colors $\{C_k \,|\, k \in \mathcal{I}\}$, where $\mathcal{I} \subseteq \{1 \dots n\}$ is an index set, and having a gradient close (L_2-norm) to a given vector field $w = \{w_k \,|\, k \in 1 \dots n\}$. The vector field values are also stored in pixels and addressed, just like for I with $w_k := (w_k^x, w_k^y)$.

The image I is defined implicitly using the Poisson equation:

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

where $\triangle$ is the Laplace operator, and div is the divergence operator. The solution is usually 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 multigrid solvers (as in [Orzan et al. 2008]) are an option. In the case of Gauss-Seidel iterations, a value $I_{i,j}$ needs to be updated by adding $(I_{i+1,j} + I_{i-1,j} + I_{i,j+1} + I_{i+1,j+1} + \operatorname{div} w_{i,j})/4$.

In the case of Diffusion Curves, colors are specified along each side of the curve and represent hard constraints. In addition, the vector field w is zero everywhere except across constraint curves. In other words, the solution will show a continuous, smooth change of colors except across hard constraints.

3.2 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:

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

where ∇ is the gradient operator, C_k are the color constraints at the pixel positions $\mathcal{I}$, and $w_i = (w_i^x, w_i^y)$ is the vector field w at pixel position i. The solution is 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. Instead we will set up a constraint system involving hard and soft constraints. 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. Hard constraints will be satisfied exactly. The soft constraints guide the diffusion in the image and implicitly define the color of the remaining pixels. Soft constraints might not be satisfied exactly, but the solution best satisfies our system in the least square sense.

To illustrate the use of such an equation system, we start by writing the Poisson-equation as a soft constraint system:

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

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. The matrix encodes the properties of the solution, which we are going to modify for our purposes.

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:

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

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. In the Poisson-equation example, 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.

4 Diffusion Control via Constraint Systems

We will now illustrate how to apply the previous formulation in order to improve upon the original standard definition.

4.1 Diffusion Barriers

An important issue addressed in this paper relates to the inflexibility of the relation between curves and colors. 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 5. 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,

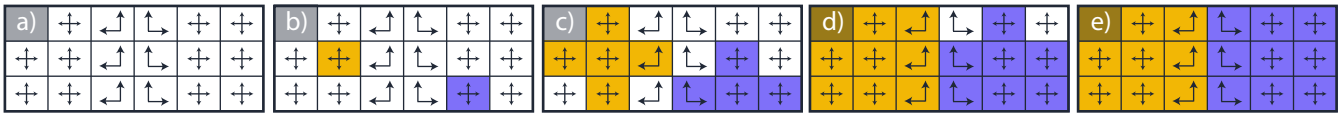

Figure 4: *a) Soft constraints indicating the diffusion behavior. b) Color is attributed to some pixels. c-d) Color diffusion illustration. e) Result after minimization. The missing soft constraints placed on the pixels in the middle of the image prevent colors from mixing.*

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.

Figure 5: *For Diffusion Curves colors have to be defined along the curve. Here, the background was supposed to be dark and this color is dragged into the interior of the hat (red circle).*

Being able to specify our constraint system allows us to define locally controllable diffusion behavior. Omitting constraints that relate two pixels breaks the connectivity between them and, therefore, blocks the diffusion process at that location. For a better understanding, Figure 4 illustrates such a situation.

Each pixel represented in the image stores a soft constraint that indicates the direction in which color information is diffused (a). Thus, every pixel diffuses color to its 4-connected neighbors, except those pixels located along the two columns in the middle of the grid. In order to illustrate the impact of locally manipulating a soft constraint, such pixels relate to only two of their neighbors, therefore breaking the connectivity between pixels in these columns. When color information is defined at some pixel positions (b), the result is that it will be diffused following the connectivity defined by the soft constraint (c-d). Because pixels in the middle of the grid do not relate, color information cannot cross them, and is therefore prevented from mixing (e).

This simple operation enables us to define *diffusion barriers*: curves that do not actively emit colors but, instead, are responsible for blocking the diffusion process from crossing the pixels underneath it. This kind of curve is useful when the user desires to restrain the diffusion from reaching a certain region without having to actually define any color at that location.

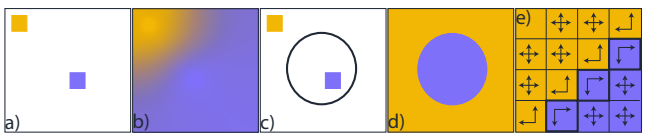

Figure 6: *a) Blue and orange strokes. b) Diffusion of blue and orange strokes. c) The circle defines a diffusion barrier. d) Diffusion blocked in the interior and exterior of the circle (diffusion barrier). e) Soft constraints breaking connectivity across the curve.*

Figure 6 shows a practical example of the use of such a curve. Blue and a orange color pixels are placed on the image (a). If no other curve is added, the blue and orange pixels will be diffused and mix at certain locations (b). Nevertheless, if we place a circle curve as a barrier (c), the diffusion of the blue pixels will be restrained to its interior; analogously, the orange to its exterior (d). In practice, diffusion barriers are obtained by breaking the connectivity between pixels underneath the curve and those located on its left side (e).

It is also possible to only set a different behavior to 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 5.

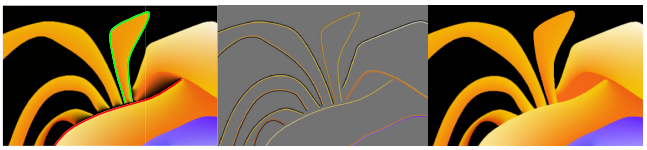

Figure 7: *Left: Lines along which problems occur. Center: Red and green line curves are transformed into barrier curves emitting colors from only one of its sides. Right: Result of the diffusion.*

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 prevent the right side of the red line from emitting colors to the interior of the hat, but on its left side, we remove the constraints connecting it to the pixels underneath the curve. Analogously, 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 in Figure 7 right. Notice that colors diffused from the ribbons are now nicely expanded to the boundaries of the hat without discontinuity artifacts.

4.2 Anisotropic Diffusion

We have seen in Section 3.2 that the diffusion process is guided by soft constraints on the derivative. Minimizing derivatives in all directions ensures the uniformity of the results. While this is of interest in many situations, it can be useful to give more control to this process. When drawing motion-blur-like streaks, flames, paint strokes and other phenomena, the color interpolation often has a privileged direction. In other words, continuity is enforced more strongly along one direction than another. Such behavior can be achieved via directional smoothness constraints, resulting in an anisotropic diffusion.

Let's look at a simple example. 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 ∇_y would lead to a diffusion along the x-axis. This is a direct consequence of the fact that differences along the y-axis are no longer penalized. Figure 8 (left) shows the influence of this process and illustrates the resulting motion-blur-like streaks obtained with this solution.

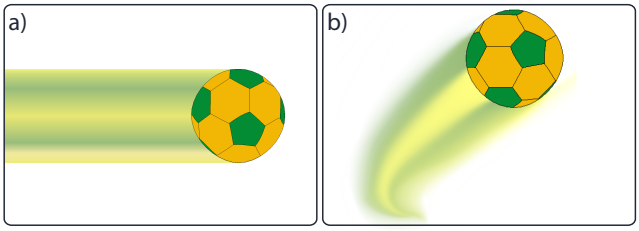

Figure 8: *Left: Horizontal diffusion; Right: Path-guided diffusion*

For an arbitrary diffusion direction $\vec{d} := (\cos\theta, \sin\theta)^T$, the matrix row needs to be changed. The directional derivative along $\vec{d}$ is $(\nabla_x, \nabla_y)\vec{d}$. Correspondingly, the discretized constraint reads:

$$\cos\theta(I_{i+1,j} - I_{i,j}) + \sin\theta(I_{i,j+1} - I_{i,j}) = 0. \quad (5)$$

Replacing the original full-derivative constraint leads to a diffusion process only along $\vec{d}$. Our goal is to use differing directional constraints to globally guide the diffusion (Figure 8, right).

In general, diffusion is rarely just following a single direction. Usually, a tradeoff between a privileged direction and its orthogonal counterpart is wanted. This implies the need for a similar smoothness constraint involving $\vec{d}^{\perp} := (-\sin\theta, \cos\theta)^T$. Adding both equations to the system would result again in a uniform diffusion process (it merely reflects a rotation of the basis vectors, meaning that $\nabla_{\vec{d}}$ takes the role of ∇_x and $\nabla_{\vec{d}^{\perp}}$, ∇_y in Eq. 3). Nonetheless, an anisotropic result can be obtained by scaling the constraints differently, as this influences the least-square result of the equation system. We will show how to use this observation to control how strongly a given direction is respected during the color diffusion.

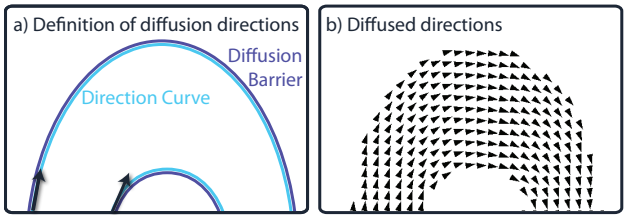

Figure 9: *To define per-pixel diffusion directions, the directions are themselves diffused.*

In order to specify a direction $\vec{d}$ and a tradeoff between standard and anisotropic diffusion, we suggest that the user draws *direction curves*. These curves contain a 2D vector, whose direction defines an orientation $\vec{d}$ and whose length results in a scaling factor to perform the tradeoff. The vectors defined by the directional curves are spread via a uniform diffusion, leading to a value in each pixel. In our interface, we let the user define vectors of length smaller than one, because a global scale does not affect the solution.

In the example of Figure 9 b, diffusion barriers were used to refrain the diffusion to the right (outer arc) and left (inner arc) sides of the curves. Adding color constraints, our diffusion process, according to Equation 5, leads to the result depicted in Figure 10. Here, color information defined along a curve (a) is dragged by the flow field resulting in an arc-shaped diffusion of colors creating a rainbow effect (b). To define a diffusion direction, the user only defined a single direction per curve, although more would have been possible. Just like colors, directions are interpolated along the curve, but follow the curve's tangent direction.

While in the previous example the directions were basically of constant length, directional constraints can vanish when opposing directions are merged during diffusion. In these areas, a privileged direction does not exist and a uniform diffusion should be applied. This need is compatible with our idea to use the length of $\vec{d}$ to define the tradeoff between standard and anisotropic diffusion. One possibility would be to use $1 - ||\vec{d}||$ as a scaling factor on the equation according to $\vec{d}^{\perp}$. For $||\vec{d}|| = 1$, only the diffusion along $\vec{d}$ is applied and for $\vec{d} = 0$ the uniform diffusion is reestablished.

In practice, we use a different solution with a threshold τ. If the vector is longer than the threshold, we renormalize it. Only if it is below τ, we use the length $\vec{d}/\tau$ as before. This threshold could also be diffused, but we found that a global value of 0.5 is usually a good choice.

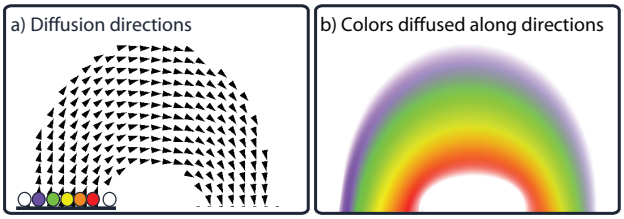

Figure 10: *The directions define constraints that ensure that the color is diffused accordingly.*

Figure 11 illustrates the influence of anisotropic diffusion and the threshold. Standard diffusion curves lead to a very smooth result that is missing many of the vivid characteristics one would expect in the case of a fire illustration. With our solution, the contrast is improved because color follows the flow of the lines and the final result looks more detailed, although we only relied on the same color curves. Modifying the threshold allows us to obtain a more uniform body of the fire.

Figure 11: *A complex anisotropic diffusion defined with a small set of curves. The two examples use different threshold settings to tradeoff uniform and anisotropic diffusion.*

4.3 Color Strength

Figure 12: *Left: Standard diffusion (equal strength), Center: Orange stronger than blue. Right: Varying strength along curve.*

The previous section presented a way to control diffusion directions, but one limitation is that it does not allow us to influence the diffusion speed. In other words, independent of the direction, the diffusion between two colors will weight both colors in the same way. In this section, we will present a solution to attribute a strength to a color in order to define its dominance in the diffusion process.

Figure 12 depicts a simple example with two color constraints, orange on the top and blue at the bottom. As expected, the diffusion process spreads these colors uniformly over the remaining image connecting both color constraints. In order to achieve fine-grained results, we introduce *color strengths*, a mechanism to control the region of influence of colors during the diffusion process. By manipulating the color strength, the artist can make the orange color become more dominant over the blue, thus pushing the diffusion in this direction (Figure 12, 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 12, right.

Figure 13 shows an example of complex color gradients achieved by manipulating the color strength in the interior and exterior of the eye. Compared with the standard result, we show that the color strength extension can lead to interesting results with no additional curves.

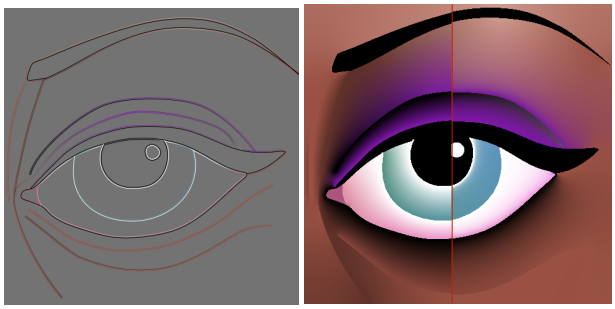

Figure 13: *The curves (left) define both parts of the image (right). The left part uses uniform weights, the right has varying weights.*

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 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 c_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 strength. This generalizes:

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

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 $a \neq 0$. Algebraically they resemble homogeneous coordinates, widely used in projective geometry calculations [Willis 2006]. 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 colors corresponds to the weighted sum of the actual colors, as defined above.

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, but all following the same diffusion behavior. At the end of the diffusion process, we perform the projection $(r/a, g/a, b/a)$ 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.

We explicitly excluded the case when a_i equals zero. It makes the color's contribution to the weighted sum be zero as well. Therefore, it would be possible to use this special case to define diffusion barriers, but in practice, this can lead to small artifacts along the boundaries and care has to be taken to correct them. Such difficulties do not arise with the solution presented in Section 4.1.

5 Beyond Local Constraints

The final problem we will address relates to the fact that the diffusion is usually locally defined. In other words, a differently colored region will always block the diffusion on its boundary. In this section, we will present a solution to connect different areas of the image to ensure a continuous diffusion between them. To some extent this can serve as a color picking of colors that are only implicitly defined by the diffusion process.

Our solution to ensure the same color on two locations is to simply link them in the diffusion process via soft constraints via a similar condition as the one that usually exists between neighboring pixels. For two pixels I_k, I_j, this translates to a soft constraint of the form $I_k - I_j = 0$. Again, the importance of this similarity can be steered by multiplying the equation with a factor. It is, hence, possible to ensure that both pixels will receive similar values at the end of the diffusion. The definition of such a constraint is simple. We allow the user to link two curves and then define points of correspondence between them, by default, we match both curves via their parametrization uniformly. It is also possible to match one curve with several others if needed.

There are several applications that arise from this possibility. It is possible to address smaller occlusions without resorting to layers, by *channeling* colors from one side of an object to another, as illustrated in Figure 2. Here, color gradients are continued across boundaries and small gaps. Another application is the creation of seamless textures. This is usually a complicated process because colors have to be matched correctly across boundaries. Solutions exist to create such textures in a postprocess [Pérez et al. 2003], but in this case the artist has little control over the appearance. By matching boundary pixels via non-local constraints, the diffusion can be wrapped around the domain and during the design process the result can already be visualized. It is also possible to move and repeat constraints, so that their displacement effectively drags the texture over the screen. An example can be found in Figure 14.

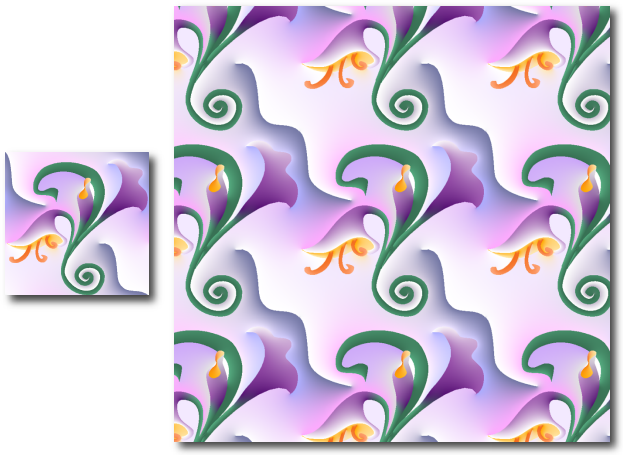

Figure 14: *Seamless diffusion textures via non-local constraints.*

Figure 15: *Left: Image outlines. Center: Shading using color diffusion. Right: Result of relighting using a normal map created by normal vector diffusion.*

6 Results

Our method was implemented using OpenGL on a NVIDIA GT285 graphics card. Besides the non-local constraints, all other extensions can be handled locally and thus can be integrated directly into the diffusion curve algorithm. Color strength comes at an added cost of about 30%, because the diffusion uses four instead of three channels. Directional diffusion comes at roughly twice the cost, because we first diffuse directions, but it still leads to a real-time solution. Unfortunately, the quality of the final result suffers from the multi-grid solver, and, for the non-local constraints, a local diffusion model is no longer very efficient. Instead, we rely on a general global linear solver implemented in CUDA which, as a side benefit leads to more precise images. Unfortunately, due to its generality the solver is slower and our system then does no longer reach real-time performance. For the teaser image, the computation took approximately $4-5$ seconds for a 512×512 image. Although the feedback is not instant, it is sufficiently fast to create convincing drawings in a small amount of time. If needed, one could imagine caching non-local constraint results to give an approximate, but faster feedback. In the same spirit, we could also maintain an inverted matrix to allow interactive color adaptation, but we keep such investigations for future work.

The advantage of our solution is that it is more flexible. Artifacts such as halos can be avoided if the artist desires. Diffusion barriers also make the color fill more intuitive and can be useful for diffusing normals. The folds on the skirts in Figure 15 and on the lion's mane in Figure 16 benefits from this solution and makes the lighting look more realistic. Color strength is a simple way of making adjustments to the illustration without increasing its complexity. The same holds for anisotropic diffusion that allows us to increase the richness in the illustrations drastically. A few directional curves can intuitively define a complex color diffusion. Finally, non-local diffusion allows easy color transfer and occlusion treatment.

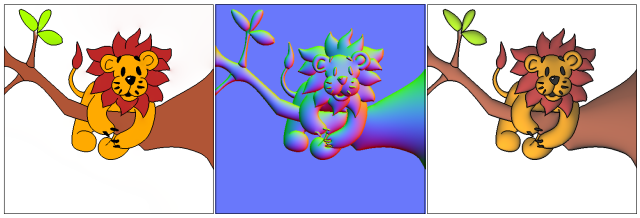

Figure 16: *Our approach can diffuse not only colors (left), but also normals (middle) for relighting purposes (right).*

7 Conclusion and Future Work

In this paper, we presented methods to increase the flexibility of diffusion-based tools for artists. We illustrated several scenarios in which our solution can be of strong benefit for the user. Our work also enabled new designs that were previously not easily realizable with existing solutions.

In the future, we would like to investigate new interaction metaphors that could replace the curve-based input. For example, brush strokes could be a useful extension. We believe that our work also has other applications. Flow fields are very versatile and one recent example is street modeling [Chen et al. 2008]. Our work could be used in the design process and the possibility to guide the diffusion of information could by an interesting extension for city simulations. Another example concerns vectorial illustrations from 3D models. It is usually tedious, but an automatic transformation into layered vector graphics is possible [Eisemann et al. 2009]. The resulting document is usually refined by an artist. If diffusion curves are involved, care must be taken to ensure color consistency across cuts that were introduced to allow a layer decomposition. Our diffusion constraints can ensure smoothness in the final illustration across layers. In the same way, we could also connect different images at the same time in order to produce not a single, but various matching texture tiles, that can then be employed as Wang Tiles [Cohen et al. 2003].

Acknowledgements

We thank the reviewers for their helpful suggestions that improved the paper. Special thanks go to L. Boissieux for the lovely drawings. We also would like to thank M. Wand, T. Boubekeur and the ARTIS team for insightful comments. H. Bezerra is supported by CAPES Brazil. D. DeCarlo acknowledges support from the Alexander von Humboldt Foundation and the National Science Foundation under grant CCF-0541185.

References

CHEN, G., ESCH, G., WONKA, P., MÜLLER, P., AND ZHANG, E. 2008. Interactive procedural street modeling. *ACM Trans. Graph. 27*, 3.

COHEN, M. F., SHADE, J., HILLER, S., AND DEUSSEN, O. 2003. Wang tiles for image and texture generation. In *SIGGRAPH '03:*

ACM SIGGRAPH 2003 Papers, ACM, New York, NY, USA, 287–294.

EISEMANN, E., PARIS, S., AND DURAND, F. 2009. A visibility algorithm for converting 3D meshes into editable 2D vector graphics. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*, ACM, New York, NY, USA, 1–8.

ELDER, J. H., AND ZUCKER, S. W. 1996. Space scale localization, blur, and contour-based image coding. In *Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR 1996)*, 00–27.

ELDER, J. H., AND ZUCKER, S. W. 1998. Local scale control for edge detection and blur estimation. *IEEE Trans. Pattern Anal. Mach. Intell. 20*, 7, 699–716.

FARBMAN, Z., HOFFER, G., LIPMAN, Y., COHEN-OR, D., AND LISCHINSKI, D. 2009. Coordinates for instant image cloning. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*, ACM, New York, NY, USA, 1–9.

JESCHKE, S., CLINE, D., AND WONKA, P. 2009. A GPU laplacian solver for diffusion curves and poisson image editing. In *Proceedings of SIGGRAPH Asia 2009*, 1–8.

JESCHKE, S., CLINE, D., AND WONKA, P. 2009. Rendering surface details with diffusion curves. In *Proceedings of SIGGRAPH Asia 2009*, 1–8.

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

LAI, Y.-K., HU, S.-M., AND MARTIN, R. R. 2009. Automatic and topology-preserving gradient mesh generation for image vectorization. *ACM Transaction on Graphics 28*, 3, 1–8.

LECOT, G., AND LEVY, B. 2006. Ardeco: Automatic Region DEtection and COnversion. In *Proceedings of the 17th Eurographics Symposium on Rendering (EGSR 2006)*, 349–360.

MCCANN, J., AND POLLARD, N. S. 2008. Real-time gradient-domain painting. In *Proceedings of SIGGRAPH 2008*.

ORZAN, A., BOUSSEAU, A., BARLA, P., AND THOLLOT, J. 2007. Structure-preserving manipulation of photographs. In *International Symposium on Non-Photorealistic Animation and Rendering (NPAR 2007)*.

ORZAN, A., BOUSSEAU, A., WINNEMÖLLER, H., BARLA, P., THOLLOT, J., AND SALESIN, D. 2008. Diffusion curves: A vector representation for smooth-shaded images. In *Proceedings of SIGGRAPH 2008*, vol. 27.

PÉREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. In *Proceedings of SIGGRAPH 2003*, 313–318.

SUN, J., JIA, J., TANG, C.-K., AND SHUM, H.-Y. 2004. Poisson matting. *ACM Transactions on Graphics 23*, 3.

SUN, J., LIANG, L., WEN, F., AND SHUM, H.-Y. 2007. Image vectorization using optimized gradient meshes. *ACM Transaction on Graphics 26*, 3, 11.

TUMBLIN, J., AND TURK, G. 1999. Lcis: a boundary hierarchy for detail-preserving contrast reduction. In *Proceedings of SIGGRAPH 1999*, 83–90.

WILLIS, P. 2006. Projective alpha colour. In *Proceeding of Eurographics 2006*.