Diffusion Curves

Weixin Lu

May 15, 2022

Contents

1 Abstract

My project is based on the Diffusion Curves paper [1]. This method can render an detailed colored image from a few control points and color sources.

2 Data

2.1 Bezier Curve

As the example in the paper, I select cubic Bezier Curves as input data. A Bezier Curve is decided by 4 points P_0, P_1, P_2, P_3.

$$B(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t) t^2 P_2 + t^3 P_3, 0 \leq t \leq 1$$

The curve B(t) has two endpoints P_0 and P_3. It doesn't pass through P_1 and P_2.

$$B'(t) = 3(1-t)^2(P_1 - P_0) + 6(1-t)t(P_2 - P_1) + 3t^2(P_3 - P_2)$$

To get the normal vector of point $B(t)$, rotate the normalized tangent vector

$$\frac{B'(t)}{||B't||}$$

by 90 degrees clockwise.

The rotation matrix of θ is

$$n(t) = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix}$$

Thus, the normal vector of point $B(t)$ is

$$n(t) = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \frac{B'(t)}{||B't||}$$

2.2 Color Sources

Given some color constraint on both sides (left and right) of the curve.

$$Cl_1, Cl_2, ...Cl_n$$

$$Cr_1, Cr_2, ...Cr_m$$

Here $m, n \geq 2$. $Cl_j = (r_j, g_j, bj, tj)$, $0 \leq t_j \leq 1$.

Interpolate to get all the colors of sample points (both sides of the curve). These points is a small distance d in the normal direction away from the original curve.

2.3 Gradients

There is also a gradient constraint calculated from color constraints. The gradient on point $B(t)$:

$$w(t) = (cl(t) - cr(t)) \times n(t)$$

where $n(t)$ is a 2D normal vector, $cl(t) - cr(t)$ is a 3D vector(RGB), $\times$ is a Cartesian product, $w(t)$ is 6D.

2.4 Data Structure

Only need $\{P_i\}, \{Cl_i\}, \{Cr_i\}$. in this project, I created 4 P points, 2 Cl constraints, 3 Cr constraints. Very little memory to storage this image!

3 Triangles

Use the triangle library, given the constraint points and the upper limit of the triangle size, it will automatically generate more vertices and edges.

4 Diffusion

4.1 Divergence

To simplify this, only pick R value of RGB here, then w is 2D. $w = (w_x, w_y)$

$$\nabla w = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y}) \cdot (w_x, w_y) = \frac{\partial w_x}{\partial x} + \frac{\partial w_y}{\partial y}$$

Note that the divergence is "the volume density of the outward flux of a vector field from an infinitesimal volume around a given point" and $w = 0$ elsewhere except for on the curve. Thus, I changed the basis (x, y) to (τ, n):

$$\nabla w = \frac{\partial w_\tau}{\partial \tau} + \frac{\partial w_n}{\partial n} = \frac{\partial w_\tau}{\partial \tau}$$

where τ is a unit vector in the tangent direction, n is a unit vector in the normal direction.

Discretization:

$$\nabla w = \frac{dw \cdot \frac{dx}{||dx||}}{||dx||} = \frac{dw \cdot dx}{||dx||^2}$$

where $dw = w_{i+1} - w_i$, $dx = x_{i+1} - x_i$.

4.2 Poisson Equation

With divergence elsewhere equals 0, I have gotten a divergence matrix $D \in \mathbb{R}^{N \times 3}$. The Laplacian operator is

$$\triangle = L = M^{-1} L_{cot}$$

where M is the mass matrix, L_{cot} is the cotangent matrix.

The Poisson Equation:

$$\begin{cases} LI = D & (1) \\ I_c = C_c & (2) \end{cases}$$

4.3 Matrix Partitioning

$$I = \begin{bmatrix} I_x \\ I_c \end{bmatrix}$$

where $I \in \mathbb{R}^{N \times 3}$, I_x is unknown, I_c is constant (color constraints).

Then let

$$L = \begin{bmatrix} L_{xx} & L_{xc} \\ L_{xc} & L_{cc} \end{bmatrix}$$

Note it is symmetric.

Then

$$\begin{bmatrix} L_{xx} & L_{xc} \\ L_{xc} & L_{cc} \end{bmatrix} \begin{bmatrix} I_x \\ I_c \end{bmatrix} = \begin{bmatrix} D_x \\ D_c \end{bmatrix}$$

$$\begin{bmatrix} L_{xx} I_x + L_{xc} I_c \\ L_{xc} I_x + L_{cc} I_c \end{bmatrix} = \begin{bmatrix} D_x \\ D_c \end{bmatrix}$$

Because $D_c = 0$ and the divergence of color constraints is not precise (it is displaced manually from the original curve), I discard the second direction.

Solve

$$L_{xx} I_x = D_x - L_{xc} I_c$$

Then concatenate I for the mesh point colors (Fig 1).

5 Improvement

Previously, I used a loop to calculate the divergence of each curve points. Inspired by Assignment 5, I implemented an alternative way using np.einsum to calculate divergence without loops.

6 Thoughts

The Laplacian operator in this project is very similar to the one in the lecture. The Laplacian in class is $Laplacian(x(u,v), y(u,v), z(u,v))$ on the surface parameter(u,v). Here it is $Laplacian(r(x,y), g(x,y), b(x,y))$ on the surface parameter(x,y)

References

[1] Alexandrina Orzan, Adrien Bousseau, Holger Winnemöller, Pascal Barla, Joëlle Thollot, and David Salesin. Diffusion curves: A vector representation for smooth-shaded images, 2008.

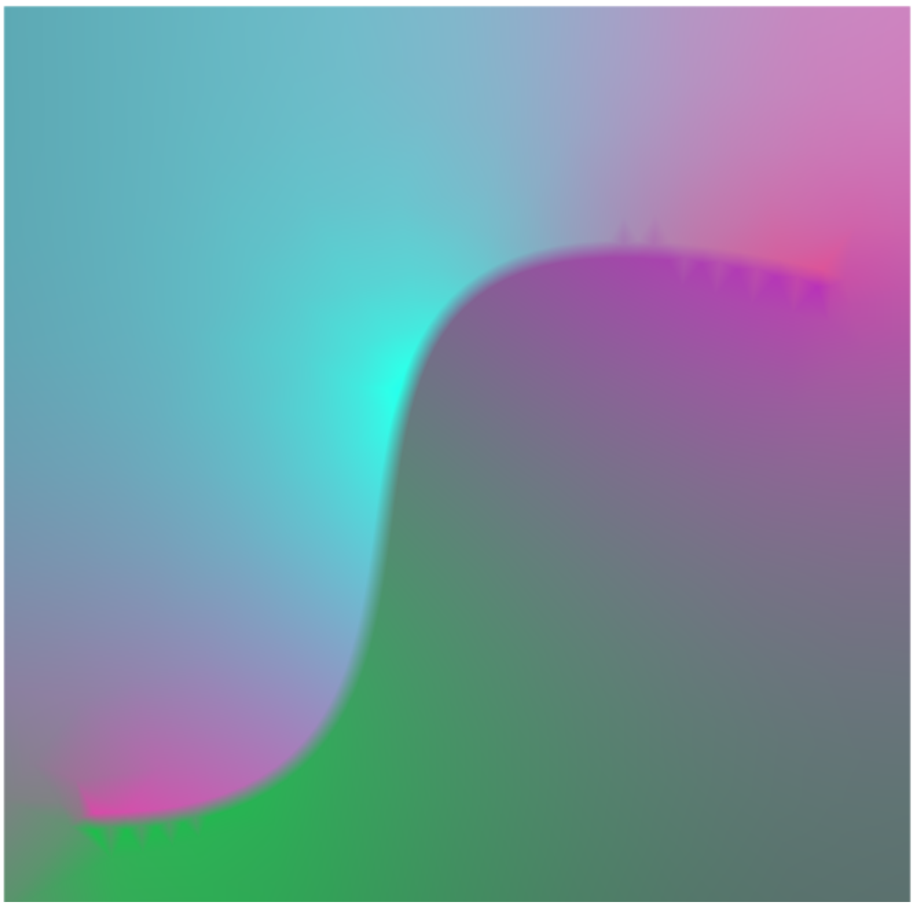

Figure 1: Image with Diffused Color