CS172 Computer Vision I: Poisson Image Editing

Ziqi Gao
2018533193
gaozq@shanghaitech.edu.cn

Abstract

Poisson Image Editing [4] introduced a generic machinery for seamless cloning based on two theories: one is a theory of psychology, indicating that second-order variations extracted by the Laplacian operator are the most significant perceptually; the other is a theory of mathematics, the core of which is that a scalar function on a bounded domain is uniquely defined by its values on the boundary and its Laplacian in the interior. Solving the Poisson equation also has an alternative interpretation as a minimization problem: it computes the function whose gradient is the closest, in the L2-norm, to the guidance vector field under given boundary conditions, which is discussed and implemented in this homework.

1. Introduction

Image editing tasks concern about global changes or local changes that are defined to a section. In this homework, the famous Poisson Image Editing algorithm is implemented, which achieves beautiful seamless cloning by doing local changes on a region restricted by a rough mask. After seamless cloning, color intensity of the masked area of the target image is different from that of the source image; but they are almost the same in gradient and color intensity of edges of the masked region.

In this homework, principle of this algorithm is explained and three approaches of seamless cloning mentioned in the paper[4] are implemented in Python: normal seamless cloning, seamless cloning and destination averaged, and mixed seamless cloning.

2. Mathematical Basics

2.1. Laplace Filter

Laplacian operator is a 2-D isotropic measure of the second spatial derivative of an image, highlighting regions of rapid intensity change and is thus widely used for edge detection.[3] Laplacian operator L(x,y) of a single-channel image I(x,y) is given by

$$L(x,y) = \frac{\partial^2}{\partial x^2} I(x,y) + \frac{\partial^2}{\partial y^2} I(x,y) \tag{1}$$

which can be computed by a Laplacian filter that acts as a convolutional kernel on the image.

0	-1	0
-1	4	-1
0	-1	0

Table 1. Laplacian Filter

2.2. Poisson's Equation

Poisson equation is in the form of

$$-\Delta\varphi = f \tag{2}$$

, where Δ is the Laplace operator and f and φ are real or complex-valued functions on a manifold[5]. Poisson partial differential equation with Dirichlet boundary conditions which specifies the Laplacian of an unknown function over the domain of interest can be denoted in the form of

$$\begin{cases} -\Delta\varphi(x,y) = f(x,y) & x,y \in \Omega \\ \varphi(x,y) = g(x,y) & x,y \in \partial\Omega \end{cases} \tag{3}$$

, where $\Omega \in R^n$.

3. Principle of Poisson Image Editing

3.1. Psychological Theory

As noted in abstract, Poisson Image Editing is based on two theories, the first of which is psychological theory. It is well known to psychologists that slow gradients of intensity, which are suppressed by the Laplacian operator, can be superimposed on an image with barely notice-able effect. Conversely, the second-order variations extracted by the Laplacian operator are the most significant perceptually.

3.2. Mathematical Theory and Modeling

3.2.1 Notation

Symbol	Notation
S	a closed subset of R^2
Ω	a closed subset of S with boundary $\partial\Omega$
$\partial\Omega$	boundary of Ω
f^*	a known scalar function defined over $S-\Omega+\partial\Omega$
f	an unknown scalar function defined over $\Omega-\partial\Omega$
$\mathbf{v}$	a vector field defined over Ω
p	a pixel in S
N_p	a set of p's 4-connected neighbors which are in S
f_p	the value of f at p
$< p, q >$	a pixel pair such that $q \in N_p$
v_{pq}	the projection of $\mathbf{v}(\frac{p+q}{2})$ on the oriented edge [p,q]
g	a source function

3.2.2 Guided Interpolation

Guided interpolation is basically using guidance vector field to do image interpolation. The simplest interpolant f of f^* over Ω is the membrane interpolant defined as the solution of the minimization problem:

$$min_f \iint_\Omega |\nabla f|^2 \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega} \tag{4}$$

where $\nabla = [\frac{\partial}{\partial x}, \frac{\partial}{\partial y}]$ is the gradient operator. The minimizer must satisfy the associated Euler-Lagrange equation:

$$\Delta f = 0 \text{ over } \Omega \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega} \tag{5}$$

The solution to this problem is the unique solution of Poisson partial differential equation with Dirichlet boundary conditions.

A guidance field is a vector field v used in an extended version of the minimization problem (4),

$$min_f \iint_\Omega |\nabla f - \mathbf{v}|^2 \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega} \tag{6}$$

whose solution is the unique solution of the following Poisson equation with Dirichlet boundary conditions:

$$\Delta f = div\mathbf{v} \text{ over } \Omega \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega} \tag{7}$$

The result of this guided minimization problem leads to the solution of interpolation problem.

3.2.3 Discrete Poisson Solver

For Dirichlet boundary conditions defined on a boundary of arbitrary shape, it is better to discretize the variational problem (6) directly, rather than the Poisson equation (7). The finite difference discretization of (6) yields the following discrete, quadratic optimization problem, whose solution satisfies the following simultaneous linear equations:

$$|N_p| f_p - \sum_{q \in N_p \cap \Omega} f_q = \sum_{q \in N_p \cap \partial\Omega} f_q^* + \sum_{q \in N_p} v_{pq} \tag{8}$$

for all p $\in \Omega$. For pixel p that is interior to Ω, there is no boundary terms in the RHS of (8), which reads:

$$|N_p| f_p - \sum_{q \in N_p} f_q = \sum_{q \in N_p} v_{pq} \tag{9}$$

Equations (8) and (9) are the core of the algorithm, since they could be formed into the form of Ax = b, while x represents the pixels in the mask. Representing A and b respectively, we can solve the x.

3.2.4 Modeling: Contruct The Gradient Field

The basic choice of $\mathbf{v}$ is to let it be the gradient field of a source image g, that is,

$$\mathbf{v} = \nabla g \tag{10}$$

then for the discrete numerical implementation,

$$\text{for all } < p, q >, v_{pq} = g_p - g_q \tag{11}$$

will be pluged into (8).

Poisson methodology also allows non-conservative gradient field to be used, so a more effective $\mathbf{v}$ can be formed by retaining the stronger of the variations in f^* or in g. Then (10) can be modified to

$$v_{pq} = \begin{cases} f_p^* - f_q^* \text{ if } |f_p^* - f_q^*| > |g_p - g_q| \\ g_p - g_q \text{ otherwise} \end{cases} \tag{12}$$

for all $< p, q >$.

4. Implementation

4.1. Code Repository

- poisson.py: a Python file containing implementations of methods of seamless blending.
- figure: a repository containing input and output images.

4.2. How To Run

python poisson.py

4.3. Code Logic of poisson.py

The main body of the code is the Poisson class, which takes three images as NumPy arrays and one 'type' parameter to decide which method to implement. Specifically, three images are src, msk, and trgt: msk is an normalized NumPy array of a grey image 'mask'; src and trgt are array representations of two RGB images 'source', 'target' respectively. Those two RGB images are split into three channels and operated separately while indices of the points inside the mask are extracted and preserved as an instance variable self.g_pixels, whose name indicates that it records the number of vertices in "g".

Inside the class are some helper functions like neighbors(), is_mask(), is_edge(), laplacian(), whose return value and logic are all explained in poisson.py in comments.

Core algorithms are implemented in self.laplacian_mtrx(), self.div_vec(), self.blend(), which are computing A, computing b, computing x respectively.

- self.laplacian_mtrx(): For the cases where the mask is a rectangle of size m*n, A can be expressed in a formulated form of size mn*mn.[2]. However, masks are not rectangular usually, so we need to form A in a more tedious way. In this function, A has a size of self.g_pixels_num * self.g_pixels_num, which is the number of pixels in the mask; Every row of A is computed independently by concerning whether its correspouding pixel's neighbor is in mask. Note that A is a sparse matrix which should be computed using scipy.sparse.

- self.div_vec(): b is different in different methods used. For the normal seamless cloning, each pixel of b is simply the result of applying Laplacian filter to the given pixel on source layer. For average one, each pixel of b is the average of the result of applying Laplacian filter to the given pixel on source layer and on target layer. For mixed one, each pixel of b is either the result of applying Laplacian filter to the given pixel on source layer or that on target layer, depending on whose gradient is larger.

- self.blend(): x's of different color channels are computed in this function using scipy.sparse.linalg.cg and merged into a RGB image, which is the result.

4.4. Result

This implementation gets images of two classic examples (jet plane and rainbow) from a course project of University of Brown.[1] The results are shown in Figure1 and Figure2, the left side of which are source, mask, target images respectively and the right side of which are results of seamless cloning, seamless cloning with destination-averaged, mixed seamless cloning respectively. The results of mixed seamless cloning are better than others from these two examples.

Figure 1. Jet Plane

5. Conclusion

The results of seamless cloning and destination-averaged seamless cloning are not better than that of mixed seamless cloning, which is reasonable since mixed seamless cloning is based on a loose selection process concerning the comparison of gradient of the target image and the source image.

Although the result are beautiful, there are still things to improve, mainly about the efficiency of the program. The most time-consuming part of the code is when computing A and b, some part of which are now implemented in list iteration instead of utilizing the features of NumPy array, which will be improved later.

References

[1] *Dataset of source, mask and target*. http://cs.brown.edu/courses/csci1950-g/results/proj2/pdoran/.

[2] *Discrete $_{p}oisson_{e}quation$*. https://en.wikipedia.org/wiki/Discrete_Poisson_equation.

[3] *Laplacian operator*. https://homepages.inf.ed.ac.uk/rbf/HIPR2/log.html.

[4] Patrick Pérez, Michel Gangnet, and Andrew Blake. "Poisson image editing". In: *ACM SIGGRAPH 2003 Papers*. 2003, pp. 313–318.

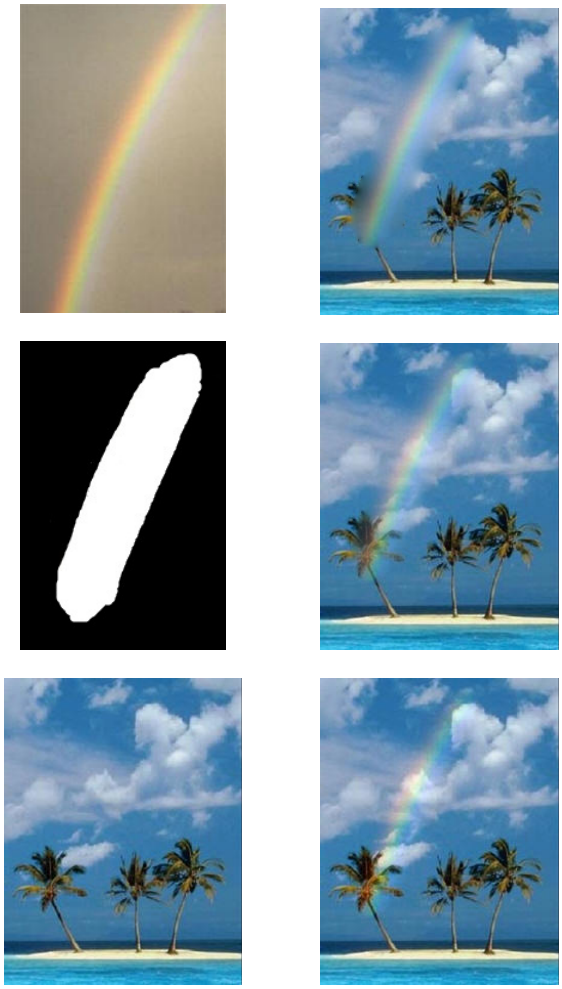

Figure 2. Rainbow

[5] *Poisson's equation*. https://en.wikipedia.org/wiki/Poisson%27s_equation.