Poisson Image Editing - Implementation

Mahmoud Afifi - York University, Canada

March 7, 2017

Abstract

Gradient domain is used instead of intensity of pixels in image cloning to blend two images by solving Poisson equations with a predefined boundary condition [1].

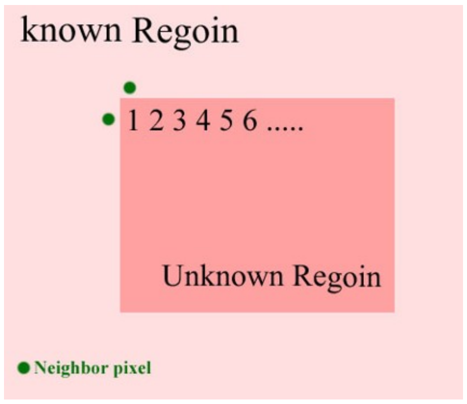

Figure 1: Simple representation of how we drive the matrix representation of the problem.

0.0.1 Introduction

Blending pixels in images is achieved by solving the minimization problem below

$$\min_{f} \iint_{\Omega} \mid \nabla f - v \mid^{2}, \text{with } f \mid_{\partial\Omega} = f^{*} \mid_{\partial\Omega} . \tag{1}$$

Where f is the unknown scalar function of the final composited image, ∇f is the first-order derivative of f with respect to each spatial coordinates (x, y), f^{*} is the scalar function of the target image, Ω is the unknown region, and $\partial\Omega$ is the boundary condition. The minimization of Eq. 1 reduces the variation between the target image and the final composited image over the boundary condition.
For each pixel p, let N be a set of its connected-four neighbors, f_p be the intensity of this pixel, f_q be the value of the pixel q where $q \in N$ and v be the gradient of the source image. Interpolation pixels over boundary can be achieved by finding the minimization of the discrete function in Eq. (1) as indicated in the following equation

$$\min_{f|_{\Omega}} \sum_{\langle p,q \rangle \cap \Omega \neq 0} (f_p - f_q - v_{pq})^2, \text{with } f_p = f^{*}{}_{p}, \text{ for all } p \in \partial\Omega. \tag{2}$$

To minimize Eq. (2) with Dirichlet boundary conditions, the Laplacian of the unknown region $\triangle f$ should be equal to the divergence of v over Ω with f equals to f^{*} over $\partial\Omega$; so, the Eq. (2) can be represented in simple linear equations as following

$$\text{for all } p \in \Omega, \mid N_p \mid f_p - \sum_{q \in N_p \cap \Omega} f_q = \sum_{q \in N_p \cap \partial\Omega} f^{*}{}_{q} + \sum_{q \in N_p} v_{pq}. \tag{3}$$

For seamless cloning, $v = \nabla g$ such that g is a known scalar function of source image; so, ∇v_{pq} can be obtained by using a convolution of source region with Laplace mask. The same for mixing gradient case, but $v = \nabla f^{*}$ in the case of that the gradient of target image is grater than the gradient of the source image.
To simplify the problem, Fig. shows a simple case of the problem. For the first

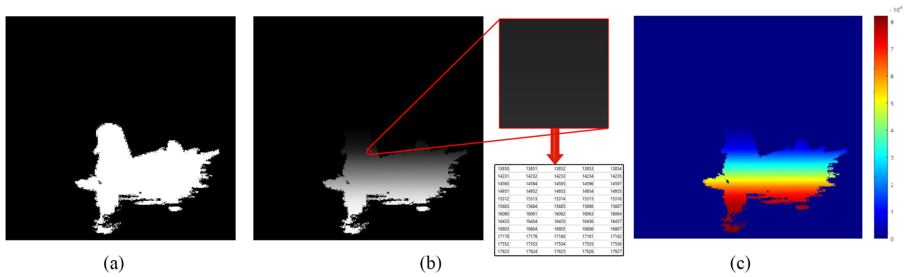

Figure 2: Mapping the pixels' locations from 2D to 1D. (a) Original mask. (b) A visualization of the mask's map that contains the 1D index of each pixel at (x,y). (c) The density representation of the map.

unknown pixel $f_{(1,1)}$, we can represent the Laplacian of the first unknown pixel as

$$\triangle f_{(1,1)} = f_{(2,1)} + L + f_{(1,2)} + U - 4f_{(1,1)} = \nabla v, \tag{4}$$

where, L is the left known pixel and U is the upper known pixel. We can rewrite it as

$$-f_{(2,1)} - f_{(1,2)} + 4f_{(1,1)} = -\nabla v + K, \tag{5}$$

where, K refers to all known boundary pixels. So, the Laplacian of any unknown pixel as

$$\triangle f_{(x,y)} = 4f(x,y) - \sum_{i}^{M} u_i = B, \tag{6}$$

where, u_i is the i^{th} unknown neighbor pixel out of M unknown neighbors, and B is $\sum_i^M k_i$ -∇v, i.e. the gradient of v minus the sum of any known pixel k in the boundary.

By stacking all equations, we can rewrite the problem in the form $A\vec{x} = \vec{b}$, where A is a coefficients matrix that represents the coefficients of $f_{(i,j)}$, $\vec{x}$ is a column vector that represents the corresponding $f_{(i,j)}$ for each coefficient in A, and $\vec{b}$ is a column vector that represents the scalar in the right hand side of the equations. Where i and j represents the coordinates of each pixel, i.e. the horizontal and vertical location of it, respectively. The diagonal values of A represent the coefficient of the pixel represented by each row.

It is clear and simple to deal with a rectangular mask. However, we should find a way to represent an arbitrary mask. So, we reshape the unknown region of the composited image to be represented as a 1D vector. Consequently, f_l is used instead of $f_{(i,j)}$ using a map function that is responsible for mapping f_l to the corresponding $f_{(i,j)}$ in the original space in both directions, such that $\Phi(f_{(i,j)}) \rightarrow f_l$ and $\Phi^{-1}(f_{(l)}) \rightarrow f_{(i,j)}$. So, each unknown pixel is indexed sequentially using a single scalar number, as shown in Fig. 0.0.1. Thus, after solving this linear system of equations, we can get the intensity of the unknown pixels. Eventually, the final composited image is constructed after remapping the 1D vector $\vec{x}$ into the 2D space using the map function.

0.0.2 Implementation

Algorithm 1 shows the pseudocode of the main function that has been used to obtain the results in the next section.

0.0.3 Results

Fig. 0.0.3 shows some experimental results obtained by the presented implementation of Poisson Image Editing.

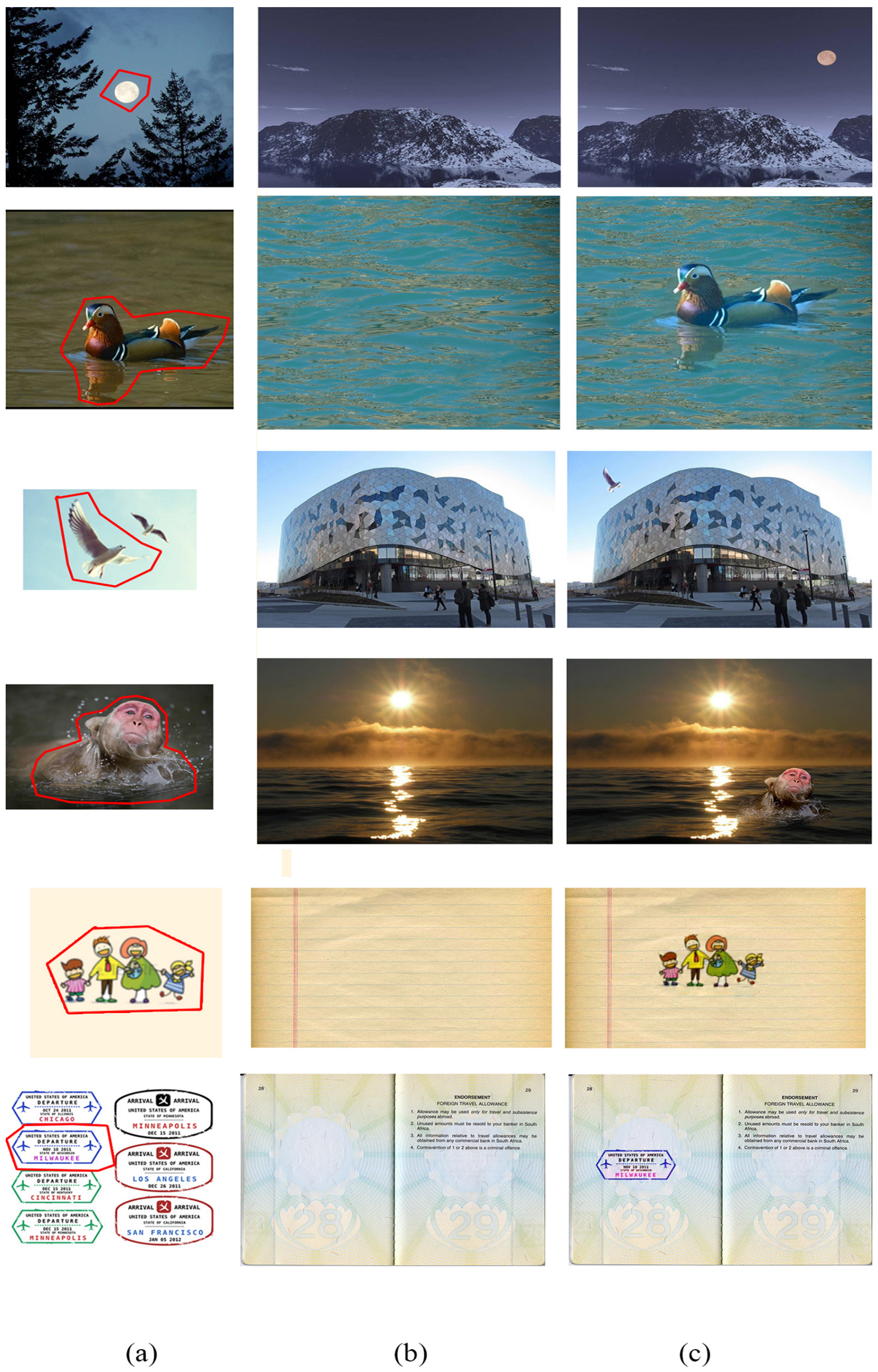

Figure 3: Results of the presented implementation of Poisson image editing. The first four rows show examples using seamless cloning, and the last two rows show example using a mixing gradient. (a) Source images. (b) Target images. (c) Results.

```
Result: Compositited image imOut
Input: imTarget, imSource, imMask, method, colorChannel;
Initialization:
n=length(imMask==1);
counter=0;
while x<=height(imMask) do
    while y<= width(imMask) do
        if imMask(x,y)==1 then
            counter=counter+1;
            map(x,y)=counter;
        else
            continue;
        end
    end
end
while c<=colorChannel do
    A=sparseMatrix(n,n);
    B=vector(n,1);
    Calculate v (the Laplacian)
    if method = 'seamless cloning' then
        lap=laplacian(imSource(:,:,c));
    else
        [gX,gY]=MixGradient(imSource(:,:,c),imTarget(:,:,c));
        lap =(gradientX(gX)+gradientY(gY));
    end
    Build coefficient matrix and the solution vector
    counter=0;
    while x<=height(imMask) do
        while y<= width(imMask) do
            if imMask(x,y)==1 then
                counter=counter+1;
                A(counter,counter)=4;
                knownPixel=getKnownNeighbors(imMask,x,y);
                while i<= length(knownPixel) do
                    (xn,yn)=getCoordinate(knownPixel);
                    B(counter)=B(counter)+imTarget(xn,yn,c);
                end
                unknownPixel=getUnknownNeighbors(imMask,x,y);
                while i<= length(unknownPixel) do
                    (xun,yun)=getCoordinate(unknownPixel);
                    A(counter, map(xun,yun))=-1;
                end
                if method = 'seamless cloning' then
                    B(counter)=B(counter)-lap(x,y);
                else
                    B(counter)=B(counter)-lap(x,y);
                end
            else
                continue;
            end
        end
    end
    Solve the linear system of equation:
    X=A\ B;
    Reallocate estimated pixels:
    imOut=reconstruct(imTarget,map,X);
end
return imOut;
```

Algorithm 1: The pseudocode of the PIE function

Bibliography

[1] Pérez, Patrick, Michel Gangnet, and Andrew Blake. "Poisson image editing." ACM Transactions on Graphics (TOG). Vol. 22. No. 3. ACM, 2003.