CSE 590: Computational Photography
Homework #2: Image Blending

Marina von Steinkirch, steinkirch@gmail.com
State University of New York at Stony Brook

March 14, 2013

1 Introduction

This project explores the gradient-domain processing, a technique with many applications including:

- blending,
- tone-mapping,
- and non-photorealistic rendering.

The goal of this assignment is to *seamlessly blend* an object from a source image into a target image. If we naively tried a simple cut and paste, we would see noticeable *seams*. However, using the *Poisson blending technique* [2], we are able to achieve better results than just cutting and pasting. The technique consists on finding values for the target pixels that *maximally preserve the gradient* of the source region, without changing any of the background pixels. In other words, we preserve the integrity of the gradient at the seams.

The Poisson blending technique is solved as a *least squares problem*, *i.e.,* given the pixel intensities of the source image, s, and of the target image, t, we solve for *new intensity values*, v, within the source region S:

$$v = \text{argmin} \sum_{i \in S, j \in N_i, S} \Big(\big(v_i - v_j\big) - \big(s_i - s_j\big) \Big)^2 + \sum_{i \in S, j \in N_i, -S} \Big(\big(v_i - t_j\big) - \big(s_i - s_j\big) \Big)^2 . \tag{1.1}$$

The idea is to recover an image that is best reflected by this edited gradient. Since the gradient provides us with linear constraints for every pixel in every color channel,

$$I_{i,j} - I_{i+1,j} = \frac{\partial}{\partial x},$$

and

$$I_{i,j} - I_{i,j+1} = \frac{\partial}{\partial y},$$

we can formulate a *overdetermined system of linear equations*. Moreover, we set the desired *gradient field* Ax equals to the constraints defined by the vector field b of the two images. The matrix A can be seen as the representation of the gradient as a *linear transformation* of the original image x. The result is a *large linear system* for each color channel with *one variable per pixel*, $Ax = b$.

2 Results

Toy Model: Reconstruction of an Image from its Gradient

We start by showing that an image can be reconstructed by its gradient values, where we:

- preserve x-y gradients, and
- preserve intensity of one pixel.

For this objective, we denote the *intensity of the source image* at (x, y) as $s(x, y)$ and the value to solve for as $v(x, y)$ and, for each pixel, we had three objectives:

- minimize $\Big(\big(v(x+1,y) - v(x,y)\big) - \big(s(x+1,y) - s(x,y)\big)\Big)^2$,
- minimize $\Big(\big(v(x,y+1) - v(x,y)\big) - \big(s(x,y+1) - s(x,y)\big)\Big)^2$,

- minimize $\Big(v(1,1) - s(1,1)\Big)^2$, *i.e.,* adding any constant value to v.

Processing the sample image with this recipe return an identical image, with *root square* equal to 2.3631×10^{-5}.

Poisson Blending

To solve the Poisson blending for two images, we implement the following steps:

1. we select source and target regions (Fig. 1-1 and 1-2),
2. we get a mask for the source image, and align the two images and the mask (Fig. 1-3 and 1-4),
3. we solve the blending constraints described in the Eq. 1.1,
4. we copy the solved values into the target image, where for RGB images each channel is processed separately. The cut and past and the final blending results can be seen in the Fig. 1-5 and 1-6.

In the Figs. 1, we see that the Poisson blending worked nicely (no seams). This is due the fact that the penguin is surrounded by snow in the source image, matching the snow in the background of the target image. We also notice that the overall intensity of the penguin is darker with the Poisson blending technique. This is due the fact that the intensity of the surrounding snow is slightly darker in the target area.

The same successful result can be see in the Fig. 2. The figures also illustrate the noticeable seam in the naive cut and paste cropping. In the other hand, in the Figs. 3 and 4 we see some failure examples of the Poisson blending. This is due the fact that the background of the two chosen images do not match even when taking the gradients.

Mixed Blending

To improve cases where we see failure in the Poisson bending, we try to implement an adaptation of the Eq. 1.1,

$$v = \text{ argmin} \sum_{i \in S, j \in N_i, S} \left(\Big(v_i - v_j \Big) - d_{i,j} \right)^2 + \sum_{i \in S, j \in N_i, -S} \left(\Big(v_i - t_j \Big) - d_{i,j} \right)^2, \tag{2.1}$$

where $d_{i,j}$ is the value of the gradient from the source or the target image with *larger magnitude*. However this technique did not show any improvement in the above examples.

References

[1] *Tamara Berg's Class*, *http://www.tamaraberg.com/teaching/Spring13/compphotog/3*

[2] *Poisson Editing*, Patrick Perez, Michel Gangnet, & Andrew Blacke, 2003

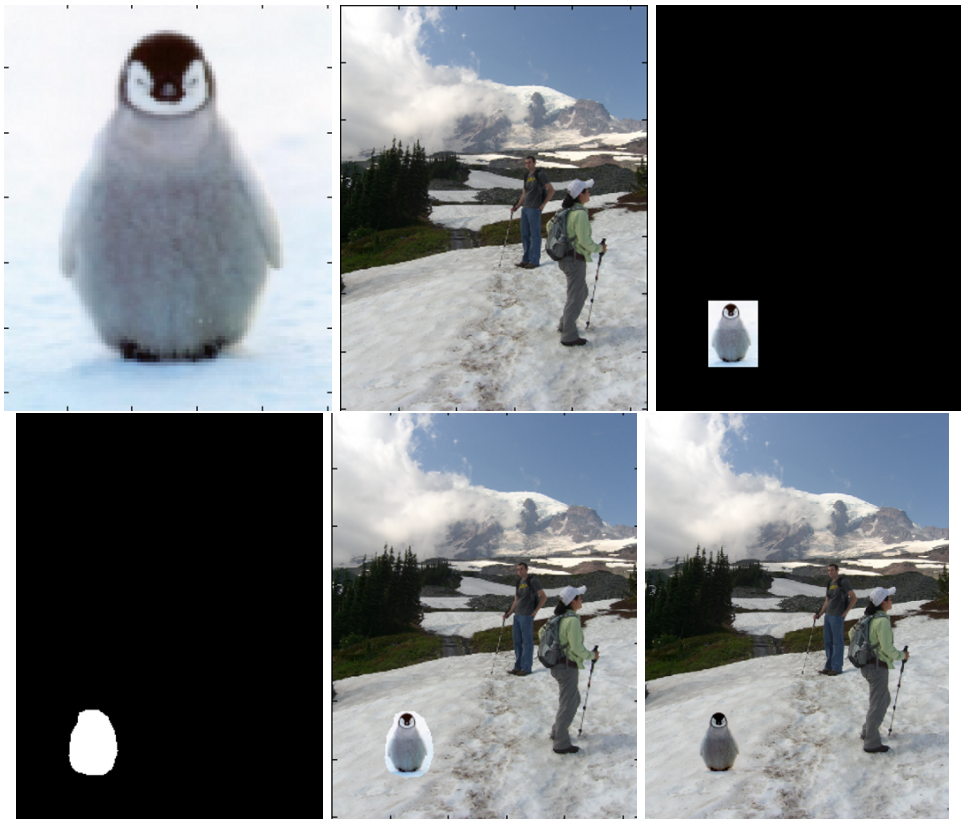

Figure 1: Sample example of Poisson blending. (left, top) Source object, (middle, top) background image, (right, top) aligned source, (left, bottom) user input mask, (middle, bottom) cut and paste result, (right, bottom) Poisson blending final result.

Figure 2: My favorite blending result. (left, top) Source object, (right, top) background image, (left, bottom) cut and paste result, (right, bottom) Poisson blending final result.

Figure 3: An failure example of Poisson example: the texture of the background of the two images are too distant for a simple gradient mixing. (left, top) Source object, (right, top) background image, (left, bottom) cut and paste result, (right, bottom) Poisson blending final result.

Figure 4: Another failure example of Poisson example, here the gradient literally changes the color of the Sun, however in the object are not corrected.(left, top) Source object, (right, top) background image, (left, bottom) cut and paste result, (right, bottom) Poisson blending final result.