CS 283 Final Project
Poisson Image Editing

George Wu

Fall 2014

1 Introduction

What if you could seamlessly blend parts of images to make it seem as if those parts actually transpired in the same setting? As photography has shifted to the digital world, photo editing has become more accessible to the average consumer, with software such as Adobe Photoshop [2] and Lightroom [1] allowing such capabilities. Traditional methods involve directly copying and pasting portions of a source image's contents onto a background and blending the edges, but unless the source and background are already similar, these edges will still be visible. Adobe Photoshop [2] also has a Healing Brush, which has a similar result to poisson image editing [4]. We will explore ways to improve this process, borrowing heavily from the Poisson equation. Using a discrete version of the Poisson solver, we will be able to perform operations such as seamless cloning.

2 Methods

2.1 Discrete Poisson Solver

To understand how the discrete poisson solver works, we will define some notation. Let Ω be the region to be solved for and its boundary $\partial\Omega$. Let f be known values outside the interior of Ω, and let h be values to be solved for within Ω. Let v be a vector field over Ω that we will define based on the operation we are performing. For discrete images, we then want to perform the following minimization:

$$\min_{h|_\Omega} \sum_{\langle p,q\rangle \cap \Omega \neq \varnothing} (h_p - h_q - v_{pq})^2,\ h_p = f_p, \forall p \in \partial\Omega \tag{1}$$

We can reformulate this problem as solving a system of linear equations

$$\forall p \in \Omega, |N_p| h_p - \sum_{q \in N_p \cap \Omega} h_q = \sum_{q \in N_p \cap \partial\Omega} f_q + \sum_{q \in N_p} v_{pq} \tag{2}$$

where N_p is the set of horizontal and vertical neighbors of pixel p. In MATLAB, after forming the matrices that represent the system of equations, we can use the mldivide operator to determine the points.

2.2 Seamless Cloning

In the context of seamless cloning, f represents our "destination image" or our "background image", which is the image we are pasting onto. We define g to be the known values of the "source image" or the image we are pasting. h is the values that we want to compute based on f and g. To build the seamless cloner, we use the gradient of the source image as our value for v. With this we define

$$v_{pq} = g_p - g_q \tag{3}$$

and we plug this value into equation (2) and solve the system of equations. This allows us to retain the differences between pixels in the source image while keeping the boundaries consistent. This is implemented in `src/SeamlessCloning.m`.

2.3 Mixed Seamless Cloning

With the seamless cloning method, within the region Ω, we discarded all the information relating to the background or destination image. We can see that because for any pixel $p \in \Omega$, or p is strictly within the region Ω, equation (2) becomes

$$\forall p \in \Omega, |N_p| h_p - \sum_{q \in N_p \cap \Omega} h_q = \sum_{q \in N_p} v_{pq} \tag{4}$$

and we can see that this has no reliance on f, the background image. However, there are some scenarios where we would like to preserve the information in the background image. This is especially true when dealing with transparent objects, objects with holes, or cloning a region from the source image onto a region that overlaps with subjects in the background image. In this situation, we would like to keep the important parts of both images. In [Perez et al. 2003], the method described is to take the stronger of the two variations in the source and destination images. Therefore, we define

$$v_{pq} = \begin{cases} f_p - f_q & \text{if } |f_p - f_q| > |g_p - g_q| \\ g_p - g_q & \text{otherwise} \end{cases} \tag{5}$$

and substitute this into equation (2).

2.4 Texture Flattening

In addition to performing cloning on two different images, we can use the poisson solver to edit portions of one image. Texture flattening is one of the many edits we can do, and [Perez

et al. 2003] suggests using an edge detector. In this case, we will have

$$v_{pq} = \begin{cases} f_p - f_q & \text{there is an edge between } p \text{ and } q \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

and we again substitute this into equation (2) to solve.

3 Results

3.1 Seamless Cloning

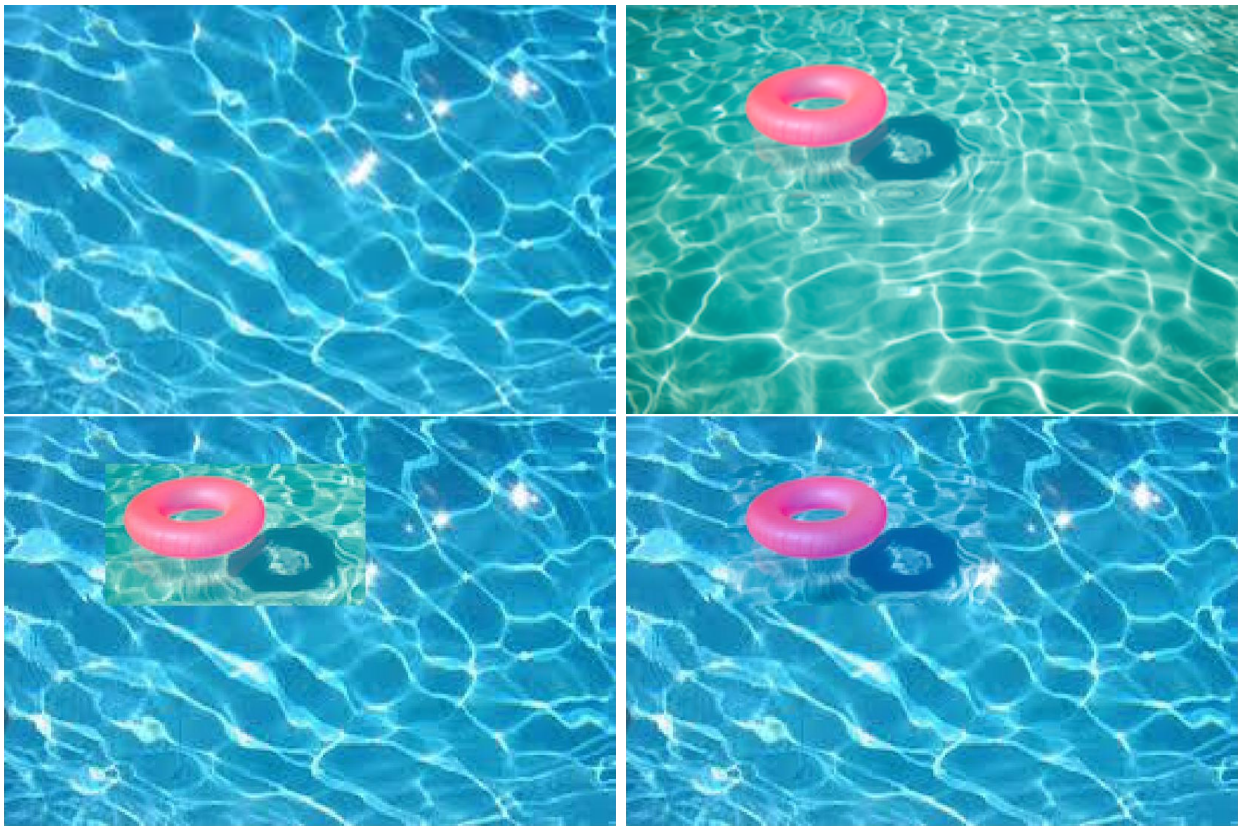

Figure 1: Top left: Our destination or background image. Top right: Our source image. Bottom left: Image after directly pasting the source onto the destination. Bottom right: Image after applying Poisson Seamless Cloning. Seamless cloning achieves smoothness at the borders that regular image editing cannot match. At the same time, the color of the pool tube is more blue compared to the color in the original image.

We first tested our algorithm on two images of water, the source having an object clone over and the destination without any objects as shown in Figure 1. Because the textures of

the two images are relatively similar, the resulting cloned image looks believable. One thing to notice is that along with the water surrounding the tube, the tube itself is now a deeper shade of blue, which could be an undesired effect.

Figure 2: Top left: Our source image. Top right: Our destination or background image. Bottom: Image after applying Poisson Seamless Cloning. We notice that the seamless cloning adds a very slight halo around aircraft, probably due to the mask being larger than the actual aircraft.

As shown in Figure 2, the Poisson solver does quite well with these images. The color of the aircraft has changed to match the environment it is being pasted into.

In general, seamless cloning is good for directly masking over a background. If there is a big difference in texture between the source and the destination, then seamless cloning will result in noticeable changes in textures. In addition, if there large differences in color, then the color of the pasted region will change dramatically as well. Seamless cloning does not seem to do well with transparent objects or when adding an object in the source close to another object in the destination.

3.2 Mixed Seamless Cloning

Here we show an example where the regular seamless cloning leads to undesirable effects.

Figure 3: Left: Our source image. Second from left: Our destination or background image. Second from right: Image after applying Poisson Seamless Cloning. Right: Image after applying Mixed Seamless Cloning. In this case, the source image is mostly hollow and produces an undesirable blur in the final image, which is taken out in the rightmost image.

As we can see in Figure 3, all the detail from the destination image in the pasted region is lost. By applying the mixed variant, we can achieve a better result. However, using the mixed variant is not always the best decision.

Figure 4: Left: Our source image. Second from left: Our destination or background image. Second from right: Image after applying Poisson Seamless Cloning. Right: Image after applying mixed seamless cloning. Here we see that the original cloning is better than the mixed cloning, mostly because we do not want to preserve the details of the underlying faces.

In Figure 4, we would like to swap the faces of the two siblings in the picture. Inherently, this means that we do not want to preserve the original face detail, and so mixed seamless cloning is not the best method to approach this. It turns out that regular seamless cloning will do the job just fine.

3.3 Texture Flattening

Figure 5: Left: Original image. Right: Flattened image. Only the most significant edges are left visible, and the rest of the face has a flat structure.

Applying the flattening over the child's face, we see that we can still make out boundaries such as the eyes, nose, and lips of the child.

Figure 6: Left: Original image. Right: Flattened image. The food in the center has a much flatter aspect, especially the broccoli and seaweed strips. The food also has lost much of its saturation.

4 Conclusion

Poisson image editing is a powerful tool that has a variety of uses. Perhaps its strongest features are that exact selection of an object is not required, and almost all of it is automated. In this paper, we explored many applications of poisson image editing such as seamless cloning and texture flattening. For the future, we would like to find ways to automatically correct for some side effects that these edits may bring about.

Seamless cloning changes the color of the entire region of the source image to match well with the destination image. While this is desirable in that it creates seamless boundaries, it also tends to change the subjects in the source images we are cloning. In some cases, such as in Figure 2, it is not a noticeable problem, but if the source background and the destination background were significantly different, then we could see undesirable color changes in the subjects that we are cloning. One thing we would like to do is to detect what the subject in the source image is and try to maintain its color values so that the final image do not change.

Another interesting thing to automate would be to smartly detect whether regular seamless cloning or mixed seamless cloning would result in a better image and perform the better of the two. We hope that there are even more extensions that use the Poisson equation to edit photos.

Image Credits Child in Figure 6 taken from [Perez et al. 2003].

5 Appendix

5.1 Source code

To check out the source code, please visit `https://github.com/georgewu2/cs283-project`. You will need to have MATLAB installed, and directions are in the `README.md` file.

References

[1] Adobe Lightroom. Adobe, Inc. <`https://www.adobe.com/products/photoshop-lightroom.html?promoid=KLXLX`>

[2] Adobe Photoshop. Adobe, Inc. <`http://www.adobe.com/products/photoshop.html`>.

[3] MATLAB documentation. <`http://www.mathworks.com/help/matlab/ref/mldivide.html`>.

[4] P. Perez, M. Ganget, and A. Blake, "Poisson Image Editing." ACM Transactions on Graphics. 2003.