Poisson Image Solver

C. Albert Thompson

Department of Computer Science
The University of British Columbia

Abstract

This paper presents the Jacobi method to solve Poisson's equation and regenerate images from the divergence of an original image. There has been a lot of related work that implement Poisson solvers, which explains the steps necessary to generate an image using a linear solver. Poisson's equation can be simplified to be solved by a linear equation, and this paper uses the Jacobi method approach. The program calculates divergence of the original image and the Jacobi method is able to approximate an image similar to the original. Regenerated images have an error of 0.01% with the iteration count of 50000 . The project also implements a basic watermark feature that adds watermarks to an image. Future work of the project can further implement different types of image manipulation, such as copy paste, image restoration and other linear solvers. This paper demonstrates one approach to solve Poisson's equation for image regeneration.

1 Introduction

The goal of this project is to understand how image reconstruction is done using Poisson's equation, and then implement a simple reconstruction algorithm on an image.

Restoring dated or printed images becomes more prominent as older images become affected by time. Using chemicals to print photos is a good way to have them last a long time. A long time is not enough, photos have been around for two hundred years, and photos will inevitably get damaged either by light, or any other of the many ways to damage printed photos. Figure 1 shows a photo that is about twenty years old and it is facing noticeable damage 1. As you can see here are vertical lines running across the photo and there is noticeable fading of the whole photo.

Restoration of images has been around for many years and early work used Fourier-transformation to restore images [Richardson 1972]. His work was later build upon by future authors such as [Bertalmio et al. 2000], and [Pérez et al. 2003]. They were able to use the old well known Poisson's equation to solve for images and repair damage or images with missing pieces. These authors used Poisson's equation to restore images in more discrete methods.

Our project takes a further look at solving Poisson's equation to regenerate images, and leaves restoration of images for future work. Initial investigation of related work shows

Figure 1: An old picture of C. Albert's family.

that most authors do manipulations on gradient fields and then restore the image using divergence. The reason Poisson's equation is so useful is that it is a partial differential equation, and allows computation an image from it's divergence.

When Poisson's equation is setup properly a linear solver may be used to retrieve an image from the divergence. The Jacobi method is the linear solver we use to solve Poisson's equation. Our implementation of the Jacobi method reconstructs an image with an error of only 0.01% in 50000 number of iterations. With these initial results show the importance of linear solvers and Poisson's equation.

This paper shows how to calculate the gradient of an image. The first step is to change the image into a scalar function. The scalar function is used to calculate gradient field of an image. The gradient of the image allows us to make minor changes to an image, and even make estimations where images contain artifacts. After the gradient manipulations occur the divergence of the image is restored using Poisson's equation.

This process shows how to add images together to get a completely new image. This paper shows the results of using a Jacobi solver to regenerate an image from its divergence. We also manipulate an image divergence by merging two images together to add a simple watermark to an image. The results of this paper are promising and give us many paths to take to improve our methods.

The goal of this project is to understand how image reconstruction is done using Poisson's equation, and then implement a simple algorithm to regenerate an image from the divergence a vector.

The sections are broken down as follows. Section 2 discusses current research in Poisson image restoration. Section 3 is dedicated to introducing explaining Poisson's equation, gradients and a way to solve Poisson's equation for image restoration. Section 4 describes artifacts and proposed methods to remove them. In section 5 talks about the future work. Section 6 is the conclusion.

2 Related work

Using Poisson's equation to reconstruct and repair images is a popular, becausle allows images to be transformed into vectors and manipulted. The resulting vectors can be solved with Poisson's equation retrieve a smooth image. An initial analysis shows that recently [Bertalmio et al. 2000] Poisson's equation has been used to do image reconstruction. The current work includes many different ways of doing image reconstruction. One field looks at taking current images that are damaged or missing parts and using gradient and Poisson to reconstruct the image. Another area looks at dragging and dropping images on top of each other then treat the resulting image as a damaged image and do the repair using Poisson.

Bertalmio et al. introduces their paper by making a goal that they want to repair damaged photos. These photos may have inpainting, meaning adding text on top of a picture. The authors also include photos that are damaged or missing parts of the image. Their contribution is different from other approaches as they do not require user input to define the texture of the spot being restored. They create a tool that uses partial differential equations and a mask to determine the area to be filled in and reconstructed. Once they fill in the area they use a smoothness function to improve the image. [Bertalmio et al. 2000] made a major contribution by allowing restoration of images to be done automatically.

Image completion is bad in many cases because it lacks gradient of an image and thus cannot fill in missing parts of an image without having significant artifacts. To overcome this problem [Shen et al. 2005] introduces three steps in the process. First a gradient-patch is used to make up gradient that should be in the image. Then second gradient-patches use an update measure to find the distance between the source patch and a target in the gradient domain. Last the image is completed using Poisson's equation on the gradient map generated by the tool.

Image reconstruction has also been extended to restoring documents that have folds and other geometric distortions. [Sun et al. 2005] combines 3D imaging with Poisson image restoration. Two initial challenges present themselves in using 3D images first shading, and second geometric correction. The paper first solves this problem using previous work on the related subjects. [Ballester et al. 2001] Once the image has been restored to its 2D format then they solve Poisson equation and restore the image. This is work is useful for restoring fragile documents without touching them.

Mixing images is a difficult task to accomplish because the every image has a different gradient field. The gradients are important because it is what allows an image to seemingly mesh together. [Pérez et al. 2003] offered a solution to fix the problem of images not being able to be place together. They do this without causing major clashing of the two images. They mix the gradients of the images together by weaving the two images into one another. They also made a contribution selecting only part of an image that one might want to reconstruct and then running Poisson's partial differential equation on that section, this was accomplished by using a mask. These tools improved the field of image drag and drop.

Drag and drop images as previously mentioned simply selected an image and dropped it onto the destination image. By doing this the destination image may contain undesired artifacts. [Jia et al. 2006] offers a way to avoid having these artifacts. They make automatic edits to the image being dropped into a destination image. This method looks at when the boundary of the image that is being dropped is different than the gradient of the destination image. They change the boundary and make the image fit nicer into the destination image.

3 Poisson

The related work gives insight on using Poisson's equation to solve for an image. Understanding Poisson's equation is necessary learn how to reconstruction an image. The next step is to retrieve a gradient field of an image. With the gradients a linear method is used to solve Poisson's equation restore an image from the divergence from the gradient field. This project shows how to setup Poisson's equation, manipulate an image and restore the image from a gradient field.

3.1 Poisson Equation

Figure 2 taken from [Pérez et al. 2003] illustrates notations used in applying Poisson's equation to our image editing problem. S is a domain for the function f which is a closed subset of $\mathbb{R}^2$. Also in the image above Ω is a closed subset
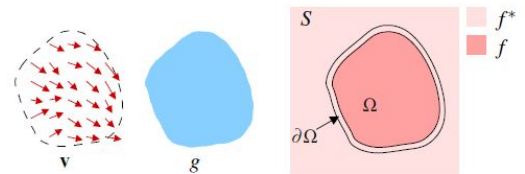

Figure 2: **Guided interpolation notations**. Unknown function f interpolates in domain Δ the destination function f^*, under guidance of vector field **v**, which is an example gradient field of a source function g

of S with the function f^*. $\mathbf{v}$ is the vector field of image Ω.

Poisson's equation is a partial differential equation that is expressed as follows:

$$\Delta \varphi = f \tag{1}$$

Δ is the Laplace operator, which is a differential operator, and $\varphi(x, y)$ and $f(x, y)$ are functions, in $\mathbb{R}^2$ domain. In this project Poisson's equation is used to regenerate 2D images.

Poisson's equation is used to solve the Ω to get an image f. This can be accomplished using satisfying the associated Euler-Lagrange equation $\Delta f = 0$ over Ω, Δ in our equation is represented as $\frac{\delta^2}{\delta x^2} + \frac{\delta^2}{\delta y^2}$. We now transform this problem into a minimization problem to solve the image f.

For the minimization problem we define it as the simplest interpolant f of f^* over Ω and represent it by the following equation:

$$\mathbf{min}_f \iint_\Omega |\Delta f|^2 \tag{2}$$

Include the vector field $\mathbf{v}$ to the minimization equation (2). We now apply the directly to the vector field $\mathbf{v}$.:

$$\mathbf{min}_f \iint_\Omega |\Delta f - \mathbf{v}|^2 \tag{3}$$

there problem has a unique solution with Dirichlet boundary conditions and is expressed:

$$\Delta f = div\mathbf{v} \text{ over } \Omega \tag{4}$$

We use this equation for each of the three colors in the RGB space to solve for the colored image f. All the results in this paper are reported with solving equation (3) each color red blue and green of an image independently. In the next section we discuss how to calculate the gradient of an image.

3.2 Gradient and Divergence

The gradient field is necessary to solve equation (4), Poisson's equation and retrieve an image. The divergence in equation 4 is derived from the gradient field. The following preliminary steps are necessary to use Poisson's equation 1) represent the image as a scalar, 2) have an equation to calculate the gradient, 3) take care of boundary conditions, 4) calculate the divergence and 5) normalize results to represent as an image.

In order to calculate a gradient we need a scalar function.

$$f(x, y) \tag{5}$$

This scalar function can be used to take the partial derivative of x and y to get a gradient field. The gradient field is the rate of change in the x and y direction and is represented by the following equation:

$$\nabla f = (\frac{\delta f}{\delta x}, \frac{\delta f}{\delta y}) \tag{6}$$

To get a gradient field an image be represented as a scalar function. An easy way to represent the image as a scalar is to represent each pixel as a red green blue (RGB) value. This project uses that method to represent an image f with the variables x, and y are the x y coordinates of the pixel in the image. The return value of function f is an integer representation of the RGB value for each pair (x, y). This is how we represent our image as a scalar function.

The gradient field of our image can be shown with the scalar function previously explained. This is calculated by taking the difference between the current pixel and its neighbors. This is calculated with the follow equation, where f_x is the is the gradient in the x direction.

$$f_x(i, j) = f(i)(j + 1) - f(i)(j - 1) \tag{7}$$

$$f_y(i, j) = f(i + 1)(j) - f(i - 1)(j) \tag{8}$$

Using these equations we get the rate of change at every pixel.

Using these equations to calculate the gradient is not possible if the border of the picture is included. The problem that occurs when using equations 8 and 7 is that these equations cannot be used to calculate the boundary conditions. If 7 comes to a boundary pixel it cannot use the equation calculate the gradient because there is no $j + 1$ pixel. To overcome this boundary problem every boundary pixel is duplicated in both x and y axis. The duplicated pixels allow us to keep the original image size and still get the gradient of the image.

The divergence, which is the right hand side of Poisson's equation (4), can be calculated with the resulting gradient field of an image. An example of the divergence of figures 3 and 4 is seen in figure 5. The divergence is the second derivative of equations 8 and 7 added together:

$$divf = (\frac{\delta^2 f}{\delta x^2} + \frac{\delta^2 f}{\delta y^2}) = f_{xx}(i, j) + f_{yy}(i, j) \tag{9}$$

The same boarder conditions apply to calculating divergence that happened with calculating the gradient. solve the boarder problem the same way we did for gradients and duplicate each border pixel.

Calculation of the gradient fields gives can result in values -255 up to 255. But negative values cannot be represented in an image that has regular RGB values 0 to 255. We normalize the gradient field array in order to display these values as an image in this paper. Following function is used to normalize the array:

$$I(x, y) = \lfloor \frac{f(i, j) - f_{min}}{f_{max} - f_{min}} \times 255 \rfloor \tag{10}$$

3.3 Poisson Solver

In Poisson's equation discussed in section 3.1 there is one matrix Δ and two vectors f, and div $\mathbf{v}$. The divergence vector $\mathbf{v}$ is known by calculations from equation (5), but there

Figure 3: f_x
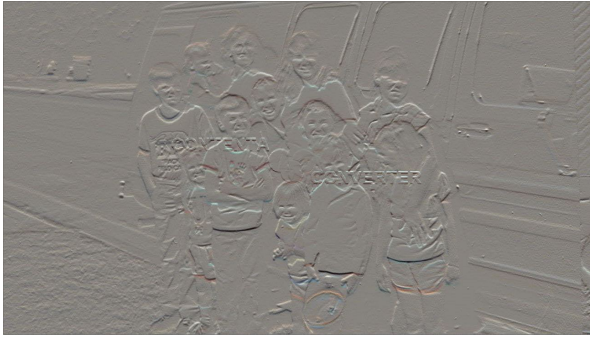
Figure 4: f_y
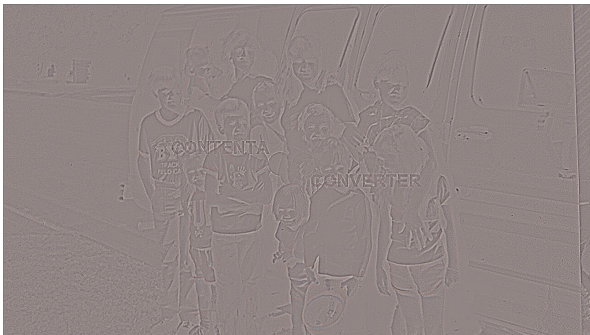
Figure 5: $(f_{xx} + f_{yy})$ Which is the divergence of f_x and f_y normalized to display in an image.

Figure 6: A 4-by-4 image setup as a Discrete Poisson Problem, image from Berkeley CS 267 class notes

are two unknowns Δ and f, which are respectivly the Laplacian, the image to be reconstructed. f can be computed in a discrete way by expressing the Laplacian and divergence as discrete filters. But this representation is difficult to solve for large vectors so an iterative Jacobi Poisson solver is used to approximate the solution. Using the Iterative Jacobi solver an image is regenerated from the divergence of that same image.

To solve Poisson's equation 4 xpress Δ and the divergence as discrete filters. An example of a discretized Laplacian for a single pixel grid would look as follow:

$$\Delta \rightarrow \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \qquad (11)$$

The divergence of the image needs to be setup to be a single column matrix. To do this we stack the columns of the image on top of each other. The image to be reconstructed is also a vector of the same dimension as the divergence. The final representation of this equation for an image that is 4-by-4 pixels would looks like Figure 6. The resulting problem is reduced to a linear system but does not scale well as the image size increases, the size of the matrix also increases by factor N^2.

Jacobi Method A Jacobi solver is used to approximate an image from the divergence, because direct solving of Poisson's equation (4) using the discretized Laplacian and divergence is not possible for larger images. The algorithm for the Jacobi method is as follows:

```
initialize all of f'(x,y) to 0
do
    for i to n
            for j to m
            f(i,j) = (f'(i-1,j)+ f'(i+1,j)+ f'(i,j-1)+ f'(i,j+1)
                    + div(i,j))/4
f'(i,j) = f(i,j)
while (image has not converged)
return f(i,j)
```

This algorithm needs only one input which is the divergence **v**. The algorithm initializes the approximation of the image to zero. Then the program goes to every pixel and takes it's four neighbors and adds them and the **div** of the pixel and then divides by four. This algorithm loops until the image converges or for a defined number of iterations. Our Jacobi approximation method that minimizes the differences between ∇f and $div\mathbf{v}$.

Applying Mask A Poisson solver create a transparent watermark on an image by adding two image divergences together. To do this we use two images, one that the original image and other hase the watermark to be added. Both images are the same size and the watermark has a homogenouse background, except for the location of the watermark. The homogenouse background makes the divergence of the 0 every where except for the location of the watermark. Both images are added together and produce an image with a watermark.

4 Artifacts

We use the Jacobi method on a number of images and compare them to the original image. There are a few artifacts in the resulting image. The watermark functionality also can be improved. Our final images had an average of 0.01%error when compared to the original image, when doing 50000 iterations.

Regenerating image After running Jacobi method figure 7 is an example of the resulting image. This image compared to the the original fig 1 it has good coloring in the middle of the picture. One of the problems in the picture are that the boarders are white and do not have the same darker coloring of the original image. The artifacts occur due to our method of dealing with the boarder of the image, using the Jacobi method.

We tried two different methods when dealing with boarder pixels using the Jacobi method. The first method to deal with the board pixel was to not calculate that pixel and leave it at zero. The result of that method is shown in figure 7. The other method we tried was similar to calculating the gradients. Boarder pixels were duplicated and instead of using adjacent non existing pixels the duplicated pixel was used. The result of that method is figure 8, it produced an image that was much darker than the origial image. The original method of calculating boarder pixels, which left boarder pixels at zero, while the Jacobi method was calculated is used for all images in this paper.

Watermark An example of the watermark feature of our application is figure 9. The image produced by the watermark has more artifacts than only regenerating the image. Simply adding divergence of the watermark and the image sometimes caused the image to not have equal levels over the entire image. One way to fix this may be to use only a small ratio of the divergence of the watermark. We did when the divergence of the watermark was small it would allow the

Figure 7: An approximation using Jacobi method to solve for the original image. With boarder pixels ignored.

Figure 8: An approximation using Jacobi method to solve for the original image. With boarder pixels duplicated.

image to produce a result much closer to that seen in when only regenerating an image.

5 Future Work

Poisson's equation allows regeneration of images that have changed gradient fields, or seamlessly integration of two images as discussed in section 2. Watermark feature of the project produces initial results on seamless integration of images, and is to how drag and dropping discussed by [Jia et al. 2006] works. Further work includes adding boundary calculations to improve the boarder of the image that is being

Figure 9: An example watermark on an image.

pasted on. Adding or removing the transparency of an integrated image can also help improve the final image.

Further exploration is necessary to implement an image restoration feature to this project. In order to accomplish this divergences need to be estimated for the mission sections of an image, and is discussed by [Pérez et al. 2003]. A mask may be used to identify what parts of the image needs restoration. With the mask estimations of missing sections are calculated and an linear solver can reproduce an image to restore missing, or damaged sections.

There are many different methods that can compute the linear equation (4) and minimize the difference in the original image and desired image. The Jacobi method used solve Poisson's equation is slow. A different linear solver can be implemented to improve the speed of the program. The Jacobi method in calculates linear problems in N^2 time. An improvement to the project would be to use the the conjugate gradient implementation which cuts the calculation time to $N^{(3/2)}$, or Fast Fourier Transform calculates in only N time. Further research allows us to change from using the Jacobi method and implement other linear solvers.

6 Conclusion

We show how Poisson's equation is used to regenerate an image from the divergence of gradient fields. We calculate gradients of an image and also take in to account the boarder conditions. An implementation of the Jacobi method solves Poisson's equation using only a divergence. With an error of 0.01%, our program obtains images that looked similar to our original image.

The Jacobi method provided us with an initial experience on methods of solving Poisson's equation. An implementation of a water mark feature demonstrates how image divergences can be layer to produce a new image. Implementing our watermark method allowed us to see how images react when two divergences are added together. Further work includes implementing further gradient manipulations and refining our calculation of boundary conditions.

References

BALLESTER, C., BERTALMIO, M., CASELLES, V., SAPIRO, G., AND VERDERA, J. 2001. Filling-in by joint interpolation of vector fields and gray levels. *Image Processing, IEEE Transactions on 10*, 8 (aug), 1200 –1211.

BERTALMIO, M., SAPIRO, G., CASELLES, V., AND BALLESTER, C. 2000. Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, SIGGRAPH '00, 417–424.

JIA, J., SUN, J., TANG, C.-K., AND SHUM, H.-Y. 2006. Drag-and-drop pasting. *ACM Transactions on Graphics (SIGGRAPH)*.

PÉREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. *ACM Trans. Graph. 22* (July), 313–318.

RICHARDSON, W. H. 1972. Bayesian-based iterative method of image restoration. *J. Opt. Soc. Am. 62*, 1 (Jan), 55–59.

SHEN, J., JIN, X., AND ZHOU, C. 2005. Gradient based image completion by solving poisson equation. In *Advances in Multimedia Information Processing - PCM 2005*, Y.-S. Ho and H. Kim, Eds., vol. 3767 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 257–268.

SUN, M., YANG, R., YUN, L., LANDON, G., SEALES, B., AND BROWN, M. 2005. Geometric and photometric restoration of distorted documents. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 2, 1117 –1123 Vol. 2.