Poisson Image reconstruction using Conjugate Gradient

C. Albert Thompson

Department of Computer Science
The University of British Columbia

Abstract

Abstract goes here.

1 Introduction

Restoring dated or printed images becomes more prominent as older images become effected 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. We have a photo that is only about twenty years old and it is facing noticeable damage **??**. 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].

Restoring this photo is possible and can be accomplished by solving Poisson's equation.

One of

For the course project of 514 we solve Poisson's equation: $\nabla^2\varphi = f$

to restore an image that has been changed to its gradient back into it's original image. Poisson's equation is interesting to solve because we can take

We plan to find out how Poisson's equation is used to remove, or blur, artifacts in images. Poisson's equation needs an image to be in the form of normal vectors. This means that we will need to write a program to take input of an image and convert it to a gradient field [Pérez et al. 2003]. We also will used conjugate gradient to simplify the problem. The goal this project is to understand how image reconstruction is done using Poisson's equation, and then implement a simple reconstruction algorithm on an image.

2 Related work

Using Poisson's equation to reconstruct and repair images is a popular because images can be transformed into vectors and completing Poisson's equation can be solved to smooth an image. From an initial analysis we have found that only recently [Bertalmio et al. 2000] has Poisson's equation used to do image 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.

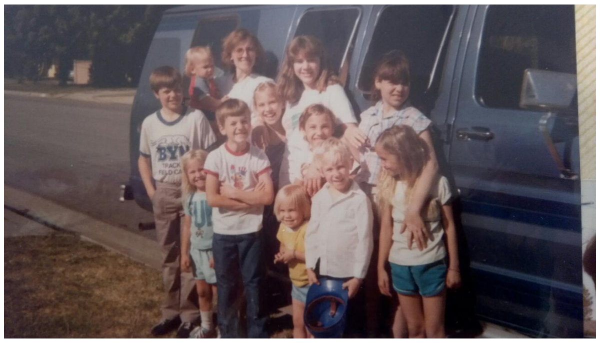

Figure 1: An old picture of C. Albert's family.

Bertalmio et al. introduces their paper by making a goal that they want to repair damaged photos. These photos may have inpainting, meaning words written on top of the picture and remove part of the image to add the text. 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 to determine the area that is defined to be filled in. 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 they introduce three steps in the process. First they introduce a gradient-patch filling to make up gradient that should be in the image. Then second they use an update strategy to gradient-patches by measure the distance between the source patch and a target in the gradient domain. Last to complete the image by using the Poisson equation on the gradient map generated by the tool.[Shen et al. 2005] [Ballester et al. 2001]

Image reconstruction has also been extended into restoring documents that have folds or other geometric distortions. [Sun et al. 2005] combines 3D images with restoration of documentation. The two initial challenges present themselves in using 3D images are shading, and geometric correction. The paper first solves this problem using previous work on the related subjects. 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 documents without touching them or that may be fragile.

Mixing images is a difficult task to accomplish because the two image have different gradients. The gradients are important because it is what allows an image to seemingly mesh together. [Pérez et al. 2003] offered a solution to fixing the problem of images not being able to be place together without causing major clashing. His idea was to mix the gradients of the images together to soft of weave 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. These tools that he added helped move image drag and drop along.

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 by making automatic edits to the image that is being dropped into a destination image. The boundary of the image that is being dropped does not fit within the gradient of an image. This method that they use changes the boundary and makes the image fit nicer into the destination image.

3 Poisson

We wanted to understand all of the related work better. We wanted to understand what Poisson's equation is and how to solve it. To accomplish this task we first studied to understand how Poisson's equation works. Next we took another step and tried to understand how gradients of an image work. Then we implemented a Poisson solver to restore the image from the divergence from the gradient field. By doing this project we have learned more about Poisson's equation and how it is used to restore in image from gradient vectors.

3.1 Poisson Equation

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

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

Where Δ is the Laplace operator. The Laplace operator is a differential operator, and $\varphi(x, y)$ and $f(x, y)$ are real functions. When Poisson's equation is used in this project we it is used in 2D images. We represent the Laplacian operator

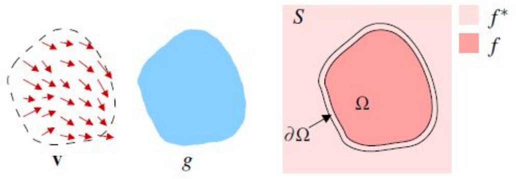

Figure 2: **Guided interpolation notations**. Unknown function f interpolates in domain Δ the destination function f^*, under guidance of vector field **v**, which might be or not the gradient field of a source function g

as ∇^2. This representation is to show that we are working with images in 2D space.

To further understand how this implementation works with our image editing 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 of S with the function f^*. **v** is the vector field of image Ω.

We Poisson's equation and solve the for Ω to get an image. 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 for our image.

For the minimization problem we define it as the simplest interpolant of f of f^2 over Ω and represent it by the following equation:

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

If we include the vector field **v** we have a version of the the minimization problem 2 but this now applies directly to the vector field **v**.:

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

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 of an image is important when solving Poisson's equation. The divergence of the vector field describe in equation 4 for our image solver is the calculation of the divergence of the gradient field. To calculate the gradient we

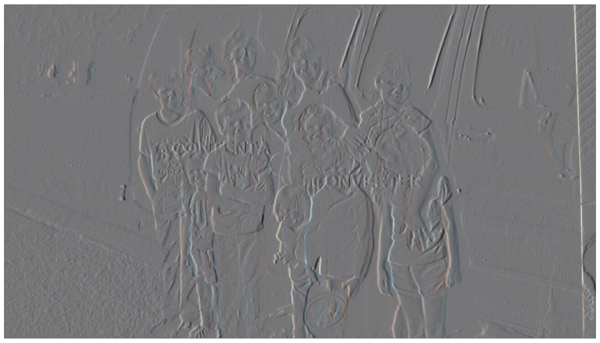

Figure 3: f_x

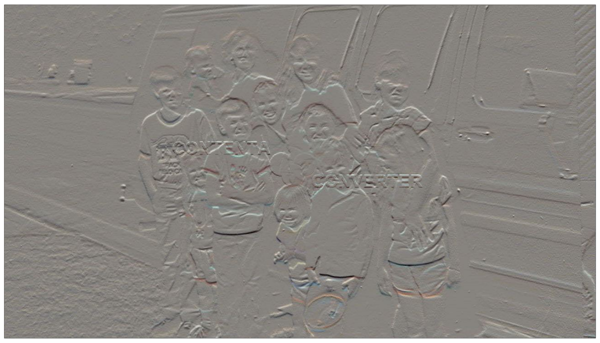

Figure 4: f_y

need 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 numbers to represent results as an image.

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

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

From this scalar function we can then 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 an image gradient field we need to be able to represent the image 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. The way we represent that in this paper is to have image f and the variables x, and y are the x y coordinates of the pixel in the image. The return value of this function is RGB value for each pair (x, y). This is how we represent our image as a scalar function.

With the scalar function we can find the gradient field of our image. This can be calculated by taking the difference between the current 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 that happens at every pixel.

If we were use only these equations to calculate the gradient, we end up with a picture that is one pixel smaller than the original image. One of the problems that occur when using equations 8 and 7 is we cannot use these equations to calculate the boundary conditions. If 7 come to a boundary pixel it cannot use the equation calculate the gradient because there is no $j+1$ pixel. We over come this boundary problem by duplicating every boundary pixel in both the x and y axis. Duplicating the pixels allows us to keep the original image size and still get the gradient of the image.

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

With the gradient field of the image we can now get the divergence which is the right hand side of the Poisson equation 4. An example of the divergence is seen in figure 5, this image demonstrates the divergence of figures 4 and 3. The divergence is the second derivative of equations 8 and 7 added together that gives us:

$$div f = (\frac{\delta^2 f}{\delta x^2} + \frac{\delta^2 f}{\delta y^2}) \tag{9}$$

When calculating divergence we also need to take into count the border conditions that happened with calculating the gradient. We solve the boarder problem the same way we did for gradients and duplicate each border pixel.

The calculation of the gradient gives us an array that can have the values -255 up to 255. But negative values cannot be represented in an image that has regular RGB values. In order to make the correction to display these values in a file and on this paper we normalize the array. To normalize the array we use the following function:

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

Normalizing the array we are able to display the gradient in the X direction with figure 3 and the gradient in the y

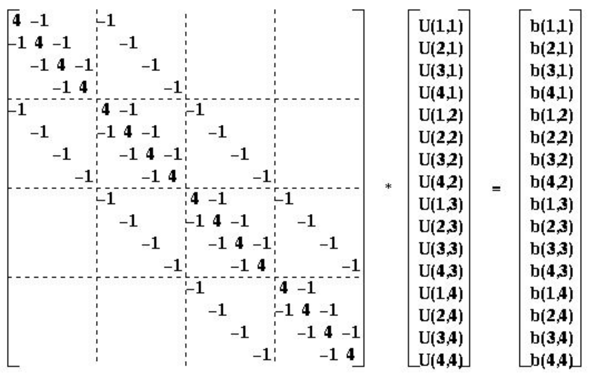

Figure 6: I_y

direction with figure 4.

3.3 Poisson Solver

In the Poisson's equation discussed in section 3.1 we deal with one matrix Δ and two vectors f, and div $\mathbf{v}$. We know the divergence vector, but we still have two unknowns the Laplacian Δ and f, which is the image we are trying to compute. We can compute f in a discrete way by expressing the Laplacian and divergence as discrete filters. But this expression is difficult to solve for large vectors so we use an iterative Jacobi Poisson solver to approximate the solution. Using the Iterative Jacobi solver we are able to regenerate an image from the divergence of that same image.

To solve Poisson's equation 4 we need to express Δ and the divergence as discrete filters. If we were to solve for on a single pixel grid the discretized Laplacian would look as follow:

$$\Delta \rightarrow \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{11}$$

The divergence of the image needs to be setup to be a vector. To do this we stack the columns of the image on top of each other. The image that we need to reconstruct is also a vector of the same dimension as the divergence vector. The final representation of this equation for an image that is 4-by-4 pixels would looks like Figure **??**. This problem is now reduced to a linear system but does not scale very well as the image increases in size the size of the matrix increases by N^2.

Jacobi Method We use a Jacobi solver to approximate the image from the divergence, because direct solving of Poisson's equation 4 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 vector **div**. It 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 Using a Poisson solver we can add divergences of different images together to create a transparent watermark in our resulting image. To do this we use two images, one that is our original image and another that is our watermark. Both images are the same size and the watermark has a complete white background, except for the location of the watermark. The white background allows us to locate the watermark and add together only the divergence the watermark and the picture.

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.

Regenerating image After running Jacobi method figure 7 is an example of the result. This image compared to the the original 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 my 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 way we dealt 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. When we were at a boarder pixel we duplicated the pixel and used the duplicated pixel instead of adjacent non existing pixels. The result of that image is show in figure **??**, it produced an image that was too dark. We stayed with our original method of calculating boarder pixels in the Jacobi method and left all boarder pixels at zero.

Watermark An example of the watermark feature of our application is show in figure 8. The image produced by the watermark has more artifacts than only regenerating the image. We found that simply adding divergence of the watermark and the image sometimes caused the image to not have equal levels over the entire image. One of the ways that we can fix this may be to use only a small ratio of the divergence of the watermark. We did when the divergence of the water-

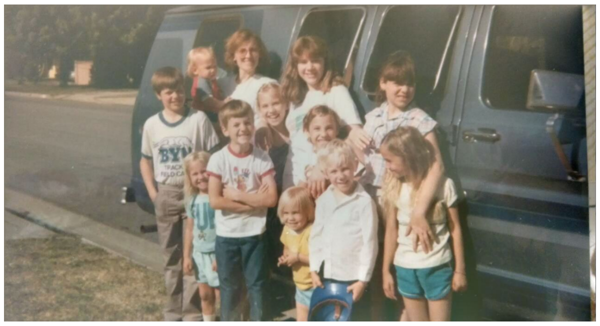

Figure 7: An approximation using Jacobi method to solve for the original image.

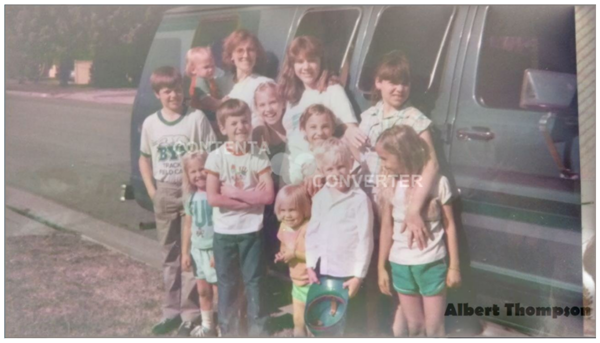

Figure 8: An example of creating a watermark on an image.

mark was small it would allow the image to produce a result much closer to that seen in when only regenerating an image.

5 Future Work

Generating images using Poisson's equation allows us to do things like change gradients of images, drag and drop images as discussed in future work. We were able to produce initial results on drag and dropping images. Adding a watermark to an image is very similar to how drag and dropping discussed by [Jia et al. 2006] works. Further work would be to add in calculations to improve the boarder of the image that is being pasted on. Also we could do more work in allowing a cut image to be transparent on the destination image.

We also would like to explore how to estimate divergences and restore images that are missing sections. This work is discussed by [Pérez et al. 2003]. To implement this work we need to find how to make estimations of divergence that have one two or three neighbors. After we can make the estimations then we may be able to reproduce an image that is missing significant portions.

Other work can be done on changing what Poisson solver we use. 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 works in N^2 space. Other methods like fast fourier transform, multigrid and conjugate gradient can be used to solve the same equations.

6 Conclusion

In this project we were able to make significant strides to better understanding how Poisson's equation. We learned how to calculate gradients of an image and also take in to account the boarder conditions. We also found how to solve Poisson's equation using the Jacobi method. After we did this work we were able to conduct a test with our program and see if we could return an image from our original image. In the end we were able to obtain images that looked similar to our original image.

Using the Jacobi method provided us with good experience to understand how gradients can be manipulated to produce different results in an image. In addition to reproducing our original image we implemented a method to add a watermark to our images. Implementing our watermark method allowed us to better understand how images react when two divergences are added together. We can continue this work and implement further gradient manipulations and refine our methods of calculating 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.