Image Blending in Gradient Domain

A Project Report

submitted by

Pavan C M (13990)

as part of

Advanced Image Processing (E9 246)

undertaken during

February 2017

DEPARTMENT OF ELECTRICAL COMMUNICATION ENGINEERING

INDIAN INSTITUTE OF SCIENCE BANGALORE - 560012

March 3, 2017

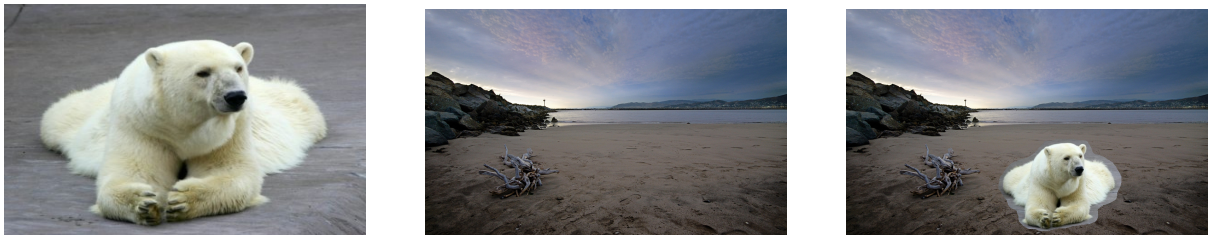
TABLE OF CONTENTS

CHAPTER 1
Introduction

1.1 Problem definition

This project investigates the problem of fusing multiple images to form a single composite image. Image blending has applications in image editing, panorama stitching, image morphing etc. Human eyes are sensitive to color and lighting differences within images. Aim of image blending is to provide smooth transitions between image parts which may be obtained from different sources. The simplest image blending method is naive blending which essentially performs cut and paste operation. However this method performs poorly when images to be combined differ in exposure levels, lighting conditions, background colors etc. As seen from figure 1.1, the transition from one image to another in naive blending is not smooth and result in undesirable visible seams.

Image blending is a well studied topic. Various algorithms have been proposed to obtain seamless composite images. In spatial domain most widely used methods are multi-band blending [1] and feathering [2]. In this project an attempt has been made to study blending methods in gradient domain. The motivation comes from the fact that slow gradients of image intensity can be superimposed on other images with barely noticeable difference. Gradient based blending techniques result in cost functions whose solution involves solving Poisson partial differential equation with Dirichlet boundary conditions [3].



(a) Source Image (b) Target Image (c) Naive Blended Image

Figure 1.1: Figures illustrating Naive Blending

CHAPTER 2
Algorithm Description

2.1 Poisson Blending

Poisson Blending algorithm description has been derived from [3]. Figure 2.1 illustrates the notations. Let S, a closed subset of R^2, be the image domain and let Ω be a closed subset of S with boundary $\partial\Omega$. Let f be the unknown scalar function defined over Ω and f^* be the known function defined over S minus the interior of Ω. Let v be the guidance vector field.

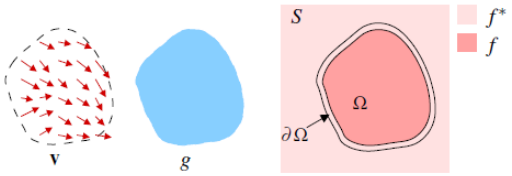


Figure 2.1: Guided Interpolation Notation. Image taken from [3]

The interpolant f of f^* over Ω is the membrane interpolant defined as the solution of the minimization problem

$$\min_f \int\int_\Omega |\nabla f - v|^2 \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega} \tag{2.1}$$

whose solution is the unique solution of Poisson equation with Dirichlet boundary conditions

$$\Delta f = \text{div} v \text{ over } \Omega \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega} \tag{2.2}$$

where $\text{div} v = \frac{\partial v_1}{\partial x} + \frac{\partial v_2}{\partial x}$ is the divergence of $v = (v_1, v_2)$. Equation 2.2 is independently solved for three channels of the image in RGB color space to obtain the interpolant f.

2.2 Discrete Poisson Solver

Solution developed in section 2.1 applies to continuous case of functions. However in real life images encountered are in discrete domain, so the solution of (2.2) has to be modified to suit discrete images. Without loss of generality let S and Ω be discrete grids with pixels. Let x and y denote the co-ordinates of the 2D grid. The condition in (2.2) reduces to

$$f(x,y) = f^*(x,y) \qquad \forall (x,y) \in \partial\Omega \tag{2.3}$$

Let p be a pixel in S such that $p = (x,y)$, let N_p be the set of its 4-connected neighbours which are in S, and let (p,q) denote a pixel pair such that $q \in N_p$ with $q = (x_1, y_1)$. Let f_p be the value of f at p. The minimization problem of (2.1) in discrete domain reduces to

$$\min_f \sum_{(p,q)\in\Omega} (f_p - f_q - v_{pq})^2, \qquad \text{with} f_p = f_p^*, \forall p \in \partial\Omega \tag{2.4}$$

The solution of (2.4) for all pixels p interior to Ω satisfies

$$|N_p|\, f_p - \sum_{q\in N_p} f_q = \sum_{q\in N_p} v_{pq} \tag{2.5}$$

In equation (2.5), $|N_p| = 4$. There can be cases in which $|N_p| < 4$ near the border of S, in such cases

$$|N_p|\, f_p - \sum_{q\in N_p} f_q = \sum_{q\in N_p\cap\partial\Omega} f_q^* + \sum_{q\in N_p} v_{pq} \tag{2.6}$$

Equations (2.5) and (2.6) form a sparse, symmetric, positive-definite system. The linear system has a size $N \times N$ where N is the number of pixels in the image. The solution to (2.5) can be solved in an iterative manner or using an exact closed form solution. In this project two methods, one using Gauss Seidel iteration which solves in iterative manner and one using sparse LU, which gives closed form solution have been discussed.
The basic choice for the guidance field v is a gradient field taken directly from a source image. Let g denote the source image, then

$$\begin{aligned} v &= \nabla g \\ v_{pq} &= g_p - g_q \end{aligned} \tag{2.7}$$

Under these conditions (2.2) reduces to

$$\Delta f = \Delta g \qquad \text{over } \Omega \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega} \tag{2.8}$$

In equation (2.8) Δ represents laplacian operator. Approximating laplacian operator using the matrix $\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$ equation (2.8) can be written as

$$\begin{aligned} 4f(x,y) - f(x+1,y) - f(x-1,y) - f(x,y+1) - f(x,y-1) &= b(x,y) \\ \text{Where} \quad b(x,y) = 4g(x,y) - g(x+1,y) - g(x-1,y) - g(x,y+1) - g(x,y-1) & \end{aligned} \tag{2.9}$$

Since g is known image, $b(x,y)$ can be calculated for all x, y. Equation (2.9) results in a linear system of equations with size $N \times N$, where N is the total number of pixels

contained in f. For an image f of size 4×4,$N = 16$ equation (2.9) can be written as

$$\begin{pmatrix} 4 & -1 & 0 & 0 & -1 & .. & .. \\ -1 & 4 & -1 & 0 & 0 & -1 & .. \\ 0 & -1 & 4 & -1 & 0 & 0 & -1 \quad .. \\ .. & 0 & -1 & 4 & -1 & 0 & .. \\ .. & .. & .. & .. & .. & .. & .. \end{pmatrix} \begin{pmatrix} f(1,1) \\ f(2,1) \\ f(3,1) \\ \vdots \\ f(4,4) \end{pmatrix} = \begin{pmatrix} b(1,1) \\ b(2,1) \\ b(3,1) \\ \vdots \\ b(4,4) \end{pmatrix}.$$

which can be written in the form

$$Ax = b \tag{2.10}$$

where A is a sparse matrix of dimensions $N \times N$. The solution to (2.10) can be obtained using various algorithms with varying degree of accuracy.

2.3 Implementation Details

The solution to equation (2.10) has been attempted using two different algorithms in this project. Gauss Seidel method and sparse LU decomposition are the two algorithms. Gauss-Seidel is an iterative algorithm whereas sparse LU decomposition has a closed form solution. A brief description of Gauss-Seidel is presented here.
Gauss-Seidel method is applicable to any matrix with non-zero diagonal elements, however convergence is guaranteed only for symmetric positive definite matrices. A is decomposed into lower triangular component L and strictly upper triangular U.

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

where A can be written as

$$A = L + U \quad \text{where} \quad L = \begin{pmatrix} a_{11} & 0 & \dots & 0 \\ a_{21} & a_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}, \quad U = \begin{pmatrix} 0 & a_{12} & \dots & a1n \\ 0 & 0 & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix}$$

The system of equations can be rewritten as

$$Lx = b - Ux \tag{2.11}$$

Gauss-Seidel method solves left hand side expression of x iteratively using previous value of x

$$x^{(k+1)} = L^{-1}(b - Ux^{(k)}) \tag{2.12}$$

where $x^{(k+1)}$ represents value of x at $(k+1)^{th}$ iteration. Using triangular form of L, equation (2.12) can be written for elements of x as

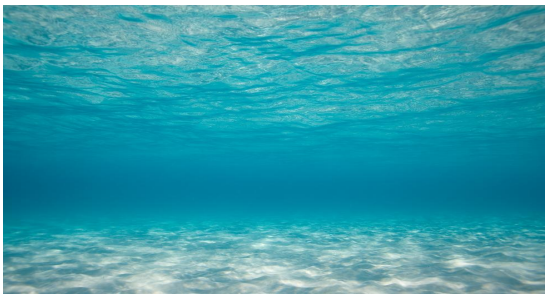
$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right), \quad i = 1, 2, ..., n \tag{2.13}$$

The method is continued until the difference between successive values becomes less significant. Sparse LU decomposition uses unsymmetric multifrontal method, which is used by default by Matlab for taking inverse of a sparse matrix.

2.4 Results

The above blending methods were applied to 20 different images collected from internet. Poisson blending code was provided by the authors of [4].

(a) Source Image



(b) Target Image

(c) Naive Blended Image

(d) Poisson Blended Image

Figure 2.2: Figures comparing Naive Blending and Poisson Blending

(a) Naive Blending

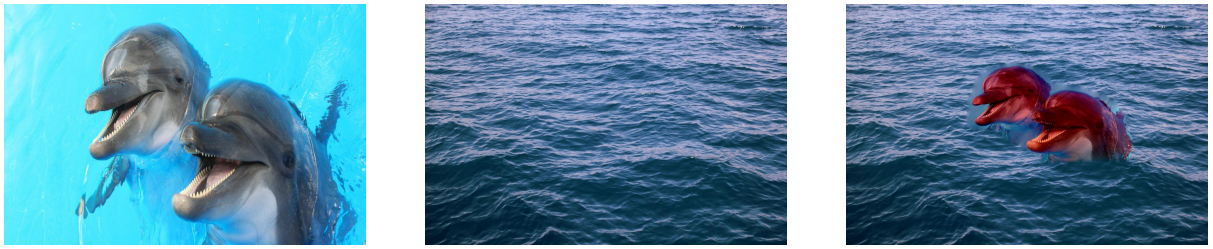
(b) Poisson Blending

Figure 2.3: Poisson Blending example

2.4.1 Limitations

Poisson blending produces best results when the source and target images have similar background color as observed in figures 2.2 and 2.3. Poisson blending incorporates in-

tensity changes only in source image, leaving target image untouched. Drawback of this method is that during optimization of color of source image, color consistency is not necessarily maintained always as Poisson tries to maintain same color contrast in blended image as that of original source image. This results in unnatural colors in some cases when source and image have significant color differences in their background which is shown in 2.4. In figure 2.4 dolphins which are originally gray colored appear to have a reddish brown color in blended image.



(a) Source Image (b) Target Image (c) Poisson Blended Image

Figure 2.4: Illustration of color distortion

Poisson problem solves color discontinuity in order to overcome visible seams. However it doesn't consider texture discontinuity which can be observed in 2.5.Tiger in the blended image of figure 2.5 appears greenish and also the grass color in source image and target image is different which leads to visible seams.

(a) Source Image (b) Target Image (c) Poisson Blended Image

Figure 2.5: Illustration of color distortion

2.4.2 Gauss Seidel vs Sparse LU Decomposition

Poisson method was tried with two different algorithms, Gauss-Seidel and sparse LU decomposition. The advantages and disadvantages are discussed here.

	Gauss-Seidel	Sparse LU
algorithm type	iterative	closed form solution
Computational complexity	$O(N^2)$	$O(N^{3/2})$
Storage Requirement	Low ($O(N)$)	High ($O(NlogN)$)

Table 2.1: Comparison of algorithms

Table 2.1 shows comparison of algorithms. Figure 2.6 compares the results obtained by different algorithms. In case of Gauss-Seidel, as iterations increase it progressively becomes better as illustrated. However the number of iterations needed for convergence is dependent on the type of image. Therefore for fixed number of iterations it might not always give best results. On the other hand sparse LU solves using closed form, however with higher complexity.

(a) Gauss Seidel - 10 iterations

(b) Gauss Seidel - 100 iterations

(c) Gauss Seidel - 350 iterations

(d) Sparse LU

Figure 2.6: Poisson method with different implementations

CHAPTER 3
Modified Poisson Problem

Poisson blending has a limitation, only the source of image is edited to obtain a seamless blend. However this might not be effective in certain situations where there exists a significant color difference between the source and target images. Color consistency is not always maintained in Poisson blend. To overcome this disadvantage, a modification to minimization problem of (2.1) was proposed by [4]

$$\min_f \int\int_\Omega |\nabla f - \nabla g|^2 + \int\int_T \epsilon \left(f^* - r\right)^2 \tag{3.1}$$

where r represents naively blended image (source is simply copied to target), ϵ is a constraint parameter used for color consistency, f is unknown function which is found by minimization, g is the source image, T represents entire image region and f^* is the known target image. ϵ can be varied to obtain suitable color consistency in the blended image.The solution to (3.1) is given by Euler-Lagrange differential equation

$$\frac{\partial f}{\partial f^*} - \frac{d}{dx}\left(\frac{\partial f}{\partial \nabla f^*}\right) = 0 \tag{3.2}$$

Equation (3.2) has a closed form solution and is obtained in [5]. The solution is given by

$$\Delta f^* - \epsilon f^* = \frac{dg}{dx} - \epsilon r \tag{3.3}$$

Taking DCT of (3.3) will give closed form solution for DCT of f^*. f^* is obtained by taking inverse DCT.

3.1 Results

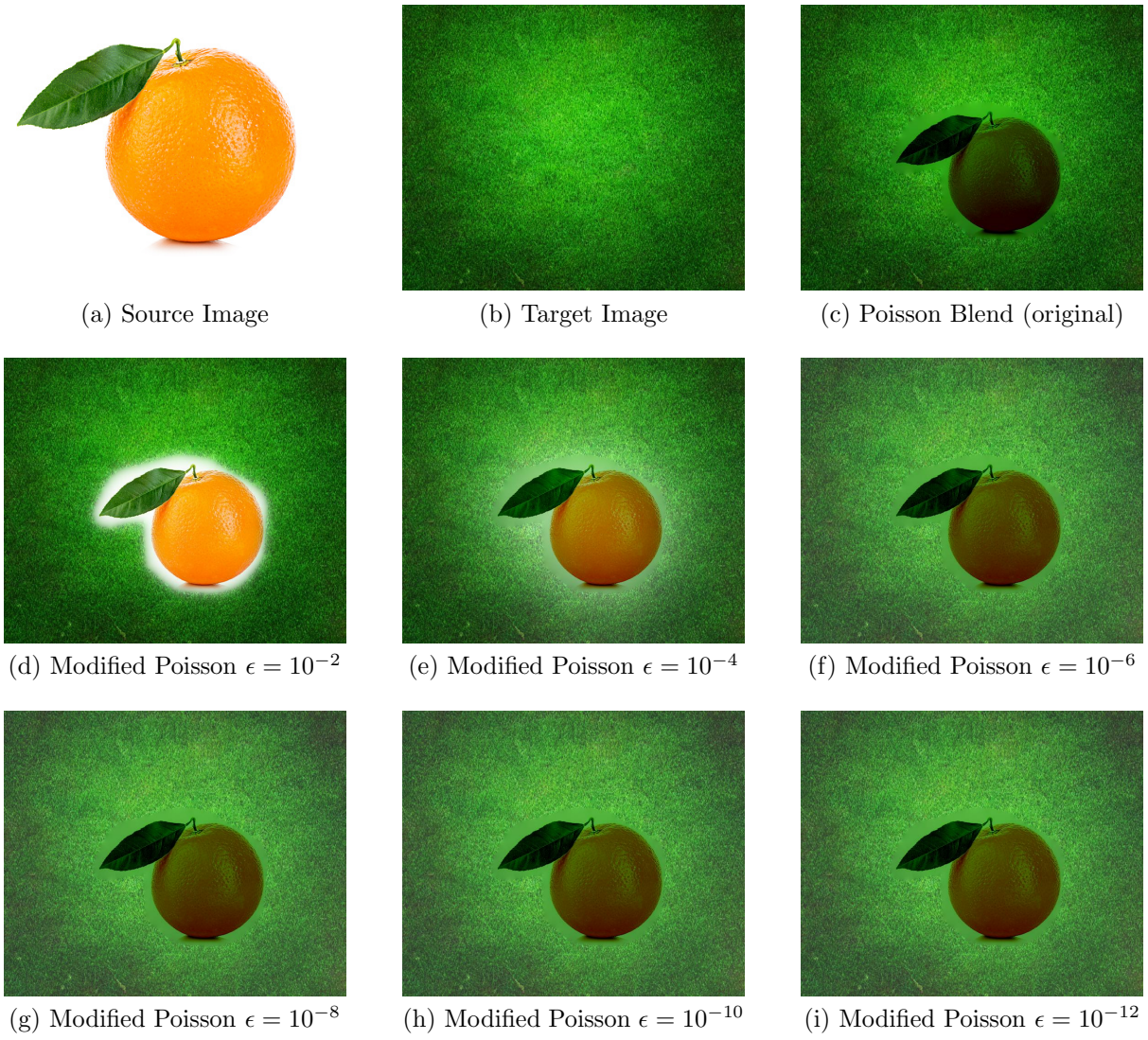
Modified Poisson method was analyzed using the code provided by authors of [4]. Original Poisson method and it's modified version are compared in figure 3.1. Baby in figure 3.1 appears darker in poisson blended image, upon application of modified version, it appears brighter. Modified Poisson reduces the intensity of surrounding pixels, thereby increasing contrast as opposed to Poisson which doesn't change any intensity values in target image.

(a) Poisson Image

(b) Modified Poisson Image

Figure 3.1: Comparison Poisson and Modified Poisson

In modified version, the color consistency can be varied by changing ϵ value. The effect of this is illustrated in 3.2. As illustrated in the figure, as ϵ increases, the seam becomes more visible. A very high ϵ value is as good as naive blending. As ϵ reduces, seam becomes less visible and after certain range there is no visible change in the images.



(a) Source Image (b) Target Image (c) Poisson Blend (original)

(d) Modified Poisson $\epsilon = 10^{-2}$ (e) Modified Poisson $\epsilon = 10^{-4}$ (f) Modified Poisson $\epsilon = 10^{-6}$

(g) Modified Poisson $\epsilon = 10^{-8}$ (h) Modified Poisson $\epsilon = 10^{-10}$ (i) Modified Poisson $\epsilon = 10^{-12}$

Figure 3.2: Variation of color consistency

3.2 Comparison and Conclusion

In this project two versions of gradient domain blending algorithms were analyzed. Poisson method performed well when the images to be blended were of similar color in nature. Significant variations in colors of the images resulted in color distortions in the blended images which in some cases resulted in visible seams, thereby defeating the aim of achieving seamless cloning of images. This drawback was mainly attributed to the fact that Poisson blending only altered image intensity values only in the source region, therefore any contrast change due to blending was applied only through the source region which resulted in reduced contrast. To overcome limitations of Poisson method, a modification

to minimization objective of Poisson was proposed. It's advantages were

- It effectively captured variations in source as well as target regions, which resulted in reduced color distortions.
- Color consistency could be varied by choosing suitable values to ϵ parameter.
- Complexity wise modified version is faster than original Poisson method.
- The method proposed by [4] for solving modified Poisson has a closed form solution. Poisson method has both closed form and iterative solution, depending on the implementation used.

REFERENCES

[1] Burt, Peter, and Edward Adelson. "The Laplacian pyramid as a compact image code." IEEE Transactions on communications 31.4 (1983): 532-540.

[2] M. Uyttendaele, A. Eden, and R. Szeliski. Eliminating ghosting and exposure artifacts in image mosaics. In Conf. on Computer Vision and Pattern Recognition, pages II:509516, 2001.

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

[4] Tanaka, Masayuki, Ryo Kamio, and Masatoshi Okutomi. "Seamless image cloning by a closed form solution of a modified poisson problem." SIGGRAPH Asia 2012 Posters. ACM, 2012.

[5] Frankot, Robert T., and Rama Chellappa. "A method for enforcing integrability in shape from shading algorithms." IEEE Transactions on pattern analysis and machine intelligence 10.4 (1988): 439-451.