# Poisson Blending

Rohit Bose

## I Motivation

Without powerful and/or specific softwares, making clean image insertions into a background is often difficult. The usual approach to this problem is to first tightly crop foreground regions before pasting into the background; however, this is an arduous process that often still keeps an undeseriable border.

My roommate recently showed me a Wikipedia page, which was my first introduction to image processing using intensity gradients. A bit more research led me to the Pérez paper, which, while admittedly far more general and technical than my solution, provided all of the insight behind the relevant boundary conditions I was looking for.

The overall premise of my solution is simple. Let the set of foreground pixels be denoted by $F$, background ones by $B$, and resulting by $X$. Depending on the nature of the desired blending, we would like to set appropriate linear combinations of $x \in X$ to equal linear combinations of $f \in F$ and $b \in B$. To make this approach work in *gradient* domain, we need a definition of gradient that is such a linear combination. Currently, we use the following trivial definitions:

$$g_x(I_{i,j}) \equiv a(I_{i,j}) - a(I_{i,j+1})$$
$$g_y(I_{i,j}) \equiv a(I_{i,j}) - a(I_{i+1,j})$$

with $i$ being a pixel's row index, $j$ a column index, $I$ some image, and $a$ the intensity function. Currently, we also ignore "difficult" gradients on image boundaries (e.g. $g_x$ of pixel on the image's right edge). We then solve the resulting overdetermined linear system; as an exact solution will almost surely not exist, we make do with the least squares solution.

## II A Linear System Approach

### A A Purely Gradient-Domain Problem

We first work through a slightly different problem where the process is similar but far easier to explain: given the "non-difficult" gradient of an image and $a(I_{1,1})$, i.e. the intensity of the top-left pixel of image $I$, how do we reconstruct the rest of the original image's intensities?

For an $m \times n$ image, the $x$-gradients alone provide $m(n-1)$ relations in the form $a(I_{i,j}) - a(I_{i,j+1}) = g_x(I_{i,j})$. Similarly, the $y$-graidents provide $n(m-1)$ relations in the form $a(I_{i,j}) - a(I_{i+1,j}) = g_y(I_{i,j})$. Finally, the provided intensity value gives us the relation $a(I_{1,1}) = p$ for some given value $p$.

This information is sufficient to produce a linear system $Av = b$, where $b$ is the list of provided values (in this case the gradient values plus the lone intensity), $v$ the final intensities (identical to $a(I)$, and $A$ the coefficients of $v$ in the above relations. Solving this linear system for $v$ reconstructs our image.

For further clarity, here is what our system looks like for a general 2×2 image:

$$\begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_{1,1} \\ v_{2,1} \\ v_{1,2} \\ v_{2,2} \end{bmatrix} = \begin{bmatrix} g_x(I_{1,1}) \\ g_x(I_{2,1}) \\ g_y(I_{1,1}) \\ g_y(I_{1,2}) \\ p \end{bmatrix}$$

## B Seamless Blending

TODO: finish. Hopefully this isn't too confusing for now. We say a pixel in the final, blended image is "outside" the foreground region if it lies outside the boundaries of where we are pasting images.

1. Outside foreground regions, set the final pixel intensities to equal background intensities (generated equations look like the bottom equation in the reconstruction system, i.e. relevant A row has a bunch of 0s and single 1).

2. Whenever pixel $d$ in the FOREGROUND has a cardinal neighbor $e$ also inside the FOREGROUND region, generate the equation $v(d) - v(e) = f(d) - f(e)$, where $f(d), f(e)$ are foreground intensities.

3. Whenever pixel $d$ in the FOREGROUND has a cardinal neighbor $e$ in the BACKGROUND, generate the equation $v(d) = b(e) + f(d) - f(e)$. $b(e)$ is the BACKGROUND pixel intensity at $e$. Can't really think of a clean explanation as to why at the moment, but for now you can try examining the more intuitively rearranged $v(d) - b(e) = f(d) - f(e)$ and trying to understand what this implies.

## C Mixed Blending

TODO: finish. You don't need this section anyway until you understand seamless anyway.

# III Desired Additions

1. What to do on the borders instead of ignoring altogether? Perhaps give the user options–replicate, zero-pad, etc?
2. Current gradient is not a good indicator of the true image rates of change. Ideally would want to enable Prewitt/Sobel, but this would certainly lead to more complicated equations.
3. Due to the nature of mixed blending, should allow for the user to opt for a rectangular selection rather than strictly limit to free-form.
4. For color images, we have to do the process three times. Is there a better algorithm? Can we parallelize?
5. See localized color changes in the Perez paper.
6. REALLY ambitious: do this in a language that isn't MATLAB.