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 V. Depending on the nature of the desired blending, we would like to set appropriate linear combinations of $v \in V$ 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:

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

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 Another Gradient-Domain Problem

We first work through a slightly different problem: 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

$$a(I_{i,j+1}) - a(I_{i,j}) = g_x(I_{i,j}).$$

Similarly, the y-gradients provide $n(m-1)$ relations in the form

$$a(I_{i+1,j}) - a(I_{i,j}) = g_y(I_{i,j}).$$

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), v the final intensities (identical to $a(I)$), and A the coefficients of v in the above relations. However, attempting to solve this system will yield infinitely many solutions; given any one solution, what happens when we add an arbitrary constant to each intensity? This is where the lone pixel intensity comes in, giving us the additional relation $a(I_{1,1}) = p$ for some provided value p. 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:

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

III Poisson Blending

We return now to our original problem: given a desired "foreground object" image[1] $F \subset T$ and "background" image B, how do we output an image V that cleanly incorporates F into a specified region within B? It turns out that this problem statement is still very general due to the diversity of images that humans perceive as "clean." This section outlines some of the many reasonable interpretations of our original problem and merely touches the enormous set of ways to solve each of them. It is heavily inspired by Section 3 of the Pérez paper.

A Seamless Blending

The sharp seam associated with directly pasting images on top of one another corresponds to an "unnatural" gradient value. This phenomenon is to be expected everywhere along the border of the newly introduced foreground region. One intuitive solution, then, is to *maintain the original background gradient along the border location* and keep all other gradients the same. Formally, we have four cases that depend on pixel location p in the final image and each of its four cardinal neighbors $n \in N$:

1. If $p \in B$ and $n \in B$, set the appropriate final directional gradient at p to equal $a(B_n) - a(B_p)$, where $a(B_p)$ and $a(B_n)$ are the background image intensies at p and n, respectively.
2. If $p \in F$ and $n \in F$, set the final gradient at p to equal $a(F_n) - a(F_p)$.
3. If $p \in B$ and $n \in F$, generate two equations, setting the final gradient at p to equal both $a(B_n) - a(B_p)$ and $a(F_n) - a(T_p)$. This will, in general, yield an unsolvable system, but the least-squares solution works remarkably well.
4. Likewise, if $p \in F$ and $n \in B$, set the final gradient at p to equal both $a(B_n) - a(B_p)$ and $a(T_n) - a(F_p)$.

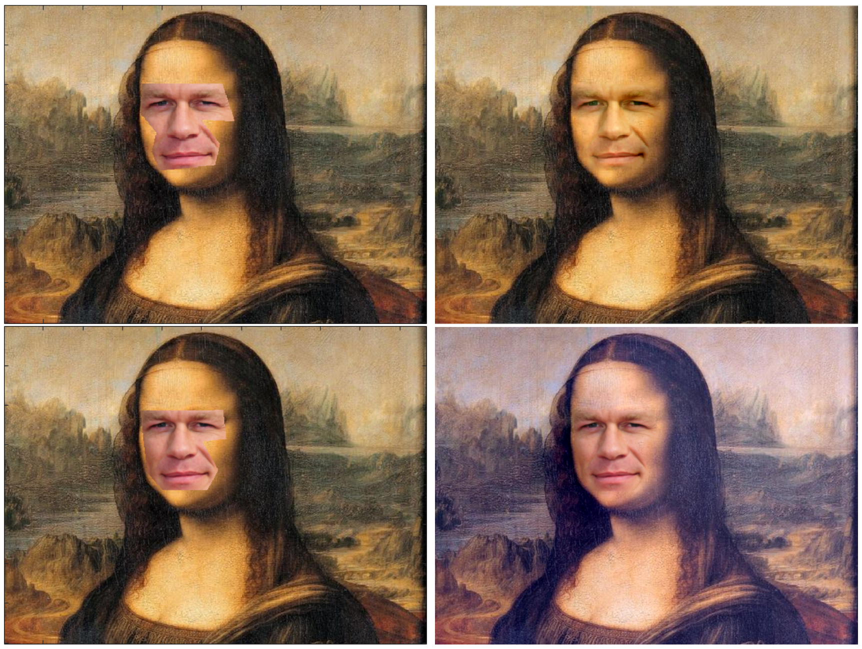

Figure 1: **Seamless blending**. The post-processing seam is equally unnoticeable in both cases; this behavior is expected because our code uses identical gradient-domain conditions. However, different *initial* conditions were used. The top example retains background pixels, while the bottom retains foreground pixels.

[1]We assume that F and B are aligned as they would be in the final result. T is the entire foreground image, while F is only the foreground region that will show up in the final image. Consequently, if a pixel p exists in F, then $F_p = T_p$. We will always use F instead of T to identify a pixel when F is defined for that pixel.

We then simply reconstruct the image from these gradient values. However, as you may have suspected, an infinite number of solutions again exist to the above system without any initial conditions. There is no universal solution; a user may desire to strictly retain the colors of the background sometimes and to utilize some sort of weighted average of foreground and background at other times. Figure 1 displays the results of using two different sets of initial conditions.

Depending on the initial conditions, the above system may be intelligently revised. For example, suppose we wanted to edit the background region as little as possible, i.e. enforce that any final background pixel maintains its original intensity. The above construction could then be equivalently replaced with:

1. If $p \in B$ and $n \in B$, set $a(f_p) = a(f_b)$.
2. If $p \in F$ and $n \in F$, set the final gradient at p to equal $a(f_n) - a(f_p)$.
3. If $p \in B$ and $n \in F$, set $a(f_n) = a(b_p) + a(f_n) - a(f_p)$.
4. If $p \in F$ and $n \in B$, set $a(f_p) = a(b_n) + a(f_p) - a(f_n)$.

B Mixed Blending

TODO