CS 180: Project 1

Images of the Russian Empire: Colorizing the Prokudin-Gorskii photo collection

Meenakshi Mittal

Overview:

Sergei Mikhailovich Prokudin-Gorskii was an early pioneer in color photography. He traveled across the Russian Empire in the early 1900s, capturing thousands of images using his innovative technique of taking three separate exposures through red, green, and blue filters. He hoped that future technology could merge these exposures into vibrant color photographs, but he unfortunately never saw his dream realized.

Decades later, The Library of Congress digitized and colorized Sergei's glass plate negatives. In this project, we seek to recreate this work. Here, we digitally reconstruct 17 of the images in the collection by aligning the color channels from the glass plate negatives, producing high-quality color photographs with minimal distortion. Additionally, we experiment with automatic contrasting and white balancing methods.

Here is an example of the three color channel glass plate negatives:

Red Channel

Green Channel

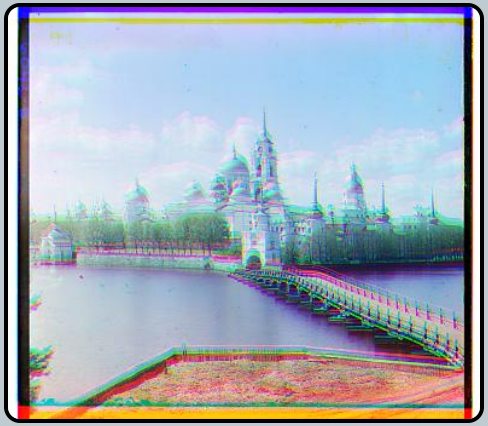
Blue Channel

Basic Alignment:

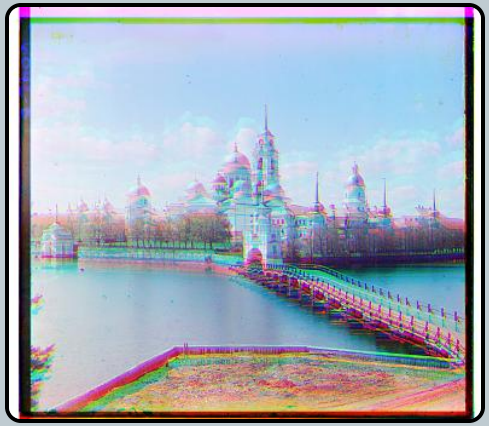
Overlaying the three color channels without any alignment results in a color shifted image almost 100% of the time. This can be seen below in the image labelled "Base".

My first naive attempt to align the color channels was to hold the blue channel steady, and then apply numpy.roll to the red and green channels over a range of [-20,20] pixels in both the x and y directions. numpy.roll circularly wraps the values of an array around to the other end, which essentially achieves the image "shifting" we are looking for. At each shifted position, we compute how well the current color channel matches the blue channel, and we choose the shift with the best match. I tried two matching metrics: minimizing the Euclidean distance and maximizing the Normalized Cross-Correlation between the two channels. They both gave similar results, so I arbitrarily selected the Euclidean distance approach for the rest of the project. The result of this attempt can be seen in "Naive Translation". The color channels still appear shifted, seemingly even more than we started with.

I eventually realized that the borders were affecting the alignment. I believe the solid-colored borders may have had near-perfect matching scores with one another, heavily influencing the total matching score. This is not ideal, because the borders of the channels are not necessarily aligned. My solution to this problem was to crop 20% off the sides of each color channel before computing the Euclidean distance between the channels. This seemed to solve the problem, as can be seen in the image labelled "Cropped Translation".



Base



Naive Translation

Cropped Translation

Pyramid Alignment:

Although the above approach works well for smaller images, we run into an issue with large images (i.e., higher resolution / pixel count). Firstly, searching over a 40x40 pixel range will not necessarily find the best alignment in a large image, as 40 pixels may only be a very small percentage of the image size. As we will see later, some of the larger images require close to a 200 pixel shift in one direction. Secondly, as the image size gets larger, computing the

Euclidean distance between two channels takes a longer time. Therefore, to use this approach for large images, we must use a large range of shifts, say [-200,200] in both directions, and each shift takes longer to compute. This gives us an unreasonably high compute time per image.

To solve this issue, I used a pyramid scaling approach. The algorithm first repeatedly scales the image down by a factor of 2 until it falls within a predetermined size. Then we find the best (x,y) shift over a range of [-20,20] pixels in both directions (very similar to the naive approach). This happens quickly because the image is small. Next, the algorithm "scales" the image up a level (i.e., it rescales the original image down to the level above the current one, as opposed to scaling the current one up). We apply the best shift found in the previous step, multiplied by 2 to account for the scaling up. Now, the algorithm again finds the best (x,y) shift, but this time only over a range of [-2,2] pixels in both directions. We add this shift to the doubled shift from the level below, and this becomes the new best (x,y) shift. The algorithm scales the image up another level, and we repeat the process until the image returns to its original scale.

I stopped scaling down once the height or width became less than 500 pixels. This value was chosen after experimenting with a few different ones like 50, 200, and 1000. Scaling the image down too much runs the risk of losing too much information, to the point that the best alignment is no longer obvious. On the other hand, not scaling it down enough increases the runtime.

The algorithm checks over the range of [-2,2] pixels in the scaled up images because the best shift from the level below ensures that we are no more than a pixel away from the correct alignment in the level above. I checked over 2 pixels instead of 1 to give the algorithm a margin of error and allow for self-correction.

Here is an example of how an image may look at each stage of scaling. (The scaling was exaggerated here for visual effect-- the image here is scaled far below the 500 pixel threshold):

1/16 1/8 1/4 1/2 1/1

Edge Alignment:

The approach described above-- pyramid scaling with Euclidean distance scoring-- worked for the majority of images. However, the color channels of an image are not always similar enough that Euclidean distance scoring works. Take the following image, where the bright blue clothing appears black in the red channel and white in the blue channel:


Red Channel

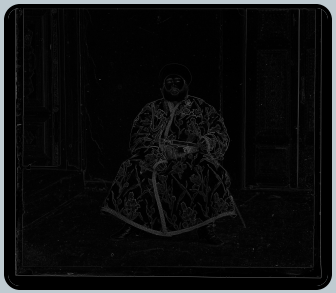

Green Channel


Blue Channel

→


Failed Alignment

To fix this problem, I performed edge extraction on each of the three color channels using the skimage.filters.prewitt function. Then I applied the previous approach to the edge images to find the best shift. All three channels usually have very similar edges, and this approach successfully aligned all the images I tested. (Please turn up brightness if edges are not visible):


Red Channel Edges


Green Channel Edges


Blue Channel Edges

→


Successful Alignment

Bells and Whistles:

After alignment, I tried a few methods of automatic image contrasting and white balancing.

- **0-1 Contrast:**
  - Rescaling pixel values such that the brightest pixel is 1 and the darkest pixel is 0
  - This had little to no effect on any of the images
- **Histogram Equalization Contrast:**
  - Rescaling pixel values so that the image's histogram is more evenly distributed across all of its values
  - Calculated using skimage.exposure.equalize_hist function
  - This tends to increase saturation and contrast-- I think it looks great on 'train', but very harsh on 'church'
- **Adaptive Histogram Equalization Contrast:**
  - Dividing image into smaller tiles and applying histogram equalization to each one

    - Calculated using skimage.exposure.equalize_adapthist function
    - This tends to significantly increase image definition and somewhat increase contrast
    - This effect is more subjective in my opinion-- I think it looks great on 'melons' and 'church', but not so good on 'emir'
- **Gray World White Balancing:**
    - Rescaling all color channels so that the average color is gray (R:128,G:128,B:128)
    - I manually implemented this-- gray is 128 in each channel, so we multiply each color channel by 128/color_channel_average
    - This works quite well for the purpose of white balancing-- it is most noticeable in 'icon' and 'arch'.
    - It also gives an almost dreamy look to some images, specifically 'church'
    - It makes some photos very gray, for lack of a better term, such as 'monastery' and 'tobolsk'
- **Average World White Balancing:**
    - Rescaling all color channels so that the average of each channel is the average of the image
    - I manually implemented this-- we take the average of the entire image, then multiply each color channel by image_average/color_channel_average
    - This has a similar effect to the gray world white balancing, except the brightness of the image tends to match its original brightness.
    - I like it more than gray world white balancing because it doesn't make the images 'grayer'

Here's how all 6 edits look on the 'cathedral' image:

Base

Histogram Equalization

Gray World White Balance

0-1 Contrast

Adaptive Histogram Equalization

Average World White Balance

Results for Provided Images:

Hover your mouse over the buttons below each image to display the edited image. The base images were removed due to their extreme similarity to the 0-1 contrast images.

self_portrait: [R: (176, 37), G: (78, 29)]

0-1 Contrast

Histogram Equalization

Adaptive Histogram Equalization

sculpture: [R: (140, -26), G: (33, -11)]

0-1 Contrast

Histogram Equalization

Adaptive Histogram Equalization

lady: [R: (120, 13), G: (56, 9)]

0-1 Contrast

Histogram Equalization

Adaptive Histogram Equalization

Gray World Balance

Gray World Balance

Avg. World Balance

Gray World Balance

Avg. World Balance

Avg. World Balance

harvesters: [R: (124, 14), G: (60, 17)]

0-1 Contrast

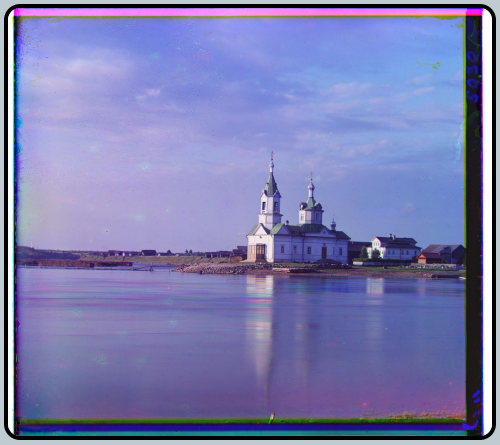
Histogram Equalization

Adaptive Histogram Equalization

Gray World Balance

Avg. World Balance

church: [R: (58, -4), G: (25, 4)]



0-1 Contrast

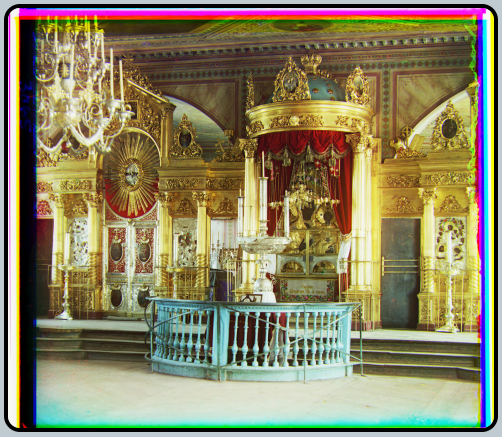
Histogram Equalization

Adaptive Histogram Equalization

Gray World Balance

Avg. World Balance

icon: [R: (90, 23), G: (42, 17)]



0-1 Contrast

Histogram Equalization

Adaptive Histogram Equalization

Gray World Balance

Avg. World Balance

onion_church: [R: (107, 36), G: (51, 26)]

melons: [R: (177, 13), G: (80, 10)]

monastery: [R: (3, 2), G: (-3, 2)]

0-1 Contrast

Histogram Equalization

Adaptive Histogram Equalization

Gray World Balance

Avg. World Balance

emir: [R: (107, 40), G: (49, 24)]

0-1 Contrast

Histogram Equalization

Adaptive Histogram Equalization

Gray World Balance

Avg. World Balance

cathedral: [R: (12, 3), G: (5, 2)]

0-1 Contrast

Histogram Equalization

Adaptive Histogram Equalization

Gray World Balance

Avg. World Balance

train: [R: (85, 29), G: (41, 2)]

0-1 Contrast

Histogram Equalization

Adaptive Histogram Equalization

Gray World Balance

Avg. World Balance

tobolsk: [R: (7, 3), G: (3, 3)]

0-1 Contrast

Histogram Equalization

Adaptive Histogram Equalization

Gray World Balance

Avg. World Balance

three_generations: [R: (111, 9), G: (54, 12)]

0-1 Contrast

Histogram Equalization

Adaptive Histogram Equalization

Gray World Balance

Avg. World Balance

0-1 Contrast

Histogram Equalization

Adaptive Histogram Equalization

Gray World Balance

Avg. World Balance

0-1 Contrast

Histogram Equalization

Adaptive Histogram Equalization

Gray World Balance

Avg. World Balance

Results for Additional Images:

cliff_house: [R: (76, -8), G: (-3, -2)]

0-1 Contrast

canoe: [R: (134, -14), G: (14, -9)]

0-1 Contrast

arch: [R: (154, 34), G: (73, 23)]

0-1 Contrast

Histogram Equalization

Adaptive Histogram Equalization

Gray World Balance

Avg. World Balance

Histogram Equalization

Adaptive Histogram Equalization

Gray World Balance

Avg. World Balance

Histogram Equalization

Adaptive Histogram Equalization

Gray World Balance

Avg. World Balance