CSE 468/568 Lab 3: Colorizing the Prokudin-Gorskii photo collection

Sergei Mikhailovich Prokudin-Gorskii (1863-1944) was a photographer who, between the years 1909-1915, traveled the Russian empire and took thousands of photos of everything he saw. He used an early color technology that involved recording three exposures of every scene onto a glass plate using a red, green, and blue filter. Back then, there was no way to print such photos, and they had to be displayed using a special projector. Prokudin-Gorskii left Russia in 1918, following the Russian revolution. His glass plate negatives survived and were purchased by the Library of Congress in 1948. Today, a digitized version of the Prokudin-Gorskii collection is available online at: http://www.loc.gov/exhibits/empire/gorskii.html

The goal of this assignment is to learn to work with images in Python by taking the digitized Prokudin-Gorskii glass plate images and automatically producing a color image with as few visual artifacts as possible. In order to do this, you will need to extract the three color channel images, place them on top of each other, and align them so that they form a single RGB color image. The assignment file (`lab3.tar.gz`) contains six images (image*.jpg) that you should run your algorithm on.

Your program should take a glass plate image as input and produce a single color image as output. The program should divide the image into three equal parts and align the second and the third parts (G and R) to the first (B). For each image, you will also need to print the (x,y) displacement vector that was used to align the parts. Detailed description is below. Example images are shown in Figure below.

Figure 1: Sample image given to you as three separate images, one each for each color channel (left); Unaligned color image (center); Aligned color image (right)

Simple Color Images

There should be six images (`image*.jpg`) in lab3.tar.gz. Each image is a concatenation of three separate glass plate images, one for each color channel (B, G and R). Your first task is to take each of the six images, align the three plate images as three color channel images and save the resultant color image. These images will look blurred as they are not aligned correctly. This is OK. Save all six images with the names `image*-color.jpg` i.e., save `image1.jpg` as `image1-color.jpg` and so on.

Create a file `main.py` that iterates through each image, creates a color image and can write the color image into a file.

Aligning Images

As described above, just overlaying the individual channel images creates blurred images. In this step, we will find the correct alignment. The easiest way to align the parts is to exhaustively search over a window of possible displacements (say [-15,15] pixels), score each one using some image matching metric, and take the displacement with the best score. There is a number of possible metrics that one could use to score how well the images match. The simplest one is just the L2 norm distance, also known as the Sum of Squared Differences (SSD), which is simply:

$$\sum_{x=1}^{n}\sum_{y=1}^{m}(image1[x][y] - image2[x][y])^2$$

Another is normalized cross-correlation (NCC), which is simply a dot product between two normalized vectors.

Write two methods that are called inside `main.py` that take an input image and, using SSD or NCC described above, calculate the alignment and print out the results. The result is in the form of two (x,y) vectors: one for SSD and one for NCC. Finally, you should create a color image based on the calculated alignments for each algorithm, and write them as `image*-ssd.jpg` and `image*-ncc.jpg`.

Some tips:

- Note that the filter order from top to bottom is RGB!
- Your task is to align the G and B channels with the R channel
- Depending on how you shift the images, the border pixels will mess with the SSD calculation, giving false optima. Test different methods for shifting and/or drop the border pixels in the SSD calculation.

Acknowledgements

This assignment is adapted from Radhakrishna Dasari with permission, though originally designed by Aloysha Efros at CMU.

Submission Instructions

You will submit `lab3.tar.gz`, a compressed archive file containing the lab3 folder. The folder should contain the original files, along with the main.py file. Please take care to follow the instructions carefully so we can script our tests. Problems in running will result in loss of points.

Please use CSE autolab for submission

The assignment is due Friday, Nov 13 before midnight.