CSC320: Assignment #1

Due on Saturday, January 24, 2016

Maxwell Huang-Hobbs

January 25, 2015

Part 0

Dataset description
The dataset consists of groups of of inverted black and white photo negatives taken through different colored lenses and from similar angles. The lenses used to take these photos are red, green, and blue lenses, in order to isolate the different primary colors of light.
These groups of photos are arranged vertically in the same image file, surrounded by a near-black border from the film negative, and a perfect white (RGB 255, 255, 255) border around that.

Figure 1: Examples of images from the dataset

The absolute positions of the images vary from image to image, but they are always arranged in the order (blue, green, red), from top to bottom.
Lastly, the large png images in the dataset have much greater color depth than the jpg images, and each value corresponds roughly to 256 times the equivalent value in one of the jpg encoded images.

Part 1

Small scale image composition

Image Cropping Algorithm

The image cropping algorithm attempts to find the inverted negatives within the image by looking for blocks of known colors. It depends on the following assumptions:

1. The areas between and around the subimages are entirely, or nearly, black.
2. There is little to no horizontal displacement between the three subimages.
3. There are only three subimages, divided at approximately $\frac{1}{3}$ and $\frac{2}{3}$ of the height of the image.
4. There is a significant value difference between the end of the border and the beginning of the image.

The algorithm works as follows:

1. The white border is cropped out of the image by scanning inward from the edges of the image, looking for the point when a sampled color value is not within a specified threshold of pure white. those values are averaged on each side and used to determine the cropping rectangle of the internal black image.
2. the external black border is removes by scanning inward in a manner similar to the one used in step 1. However, positions where no significant color difference is encountered within a specified threshold are recorded and put aside, to be referred to as "cuts".
3. The horizontal cuts generated by scanning the left and right sides of the image are then consolidated by grouping together like values within a given range, and the resulting groups are considered as candidates for a strip through the image. Two cuts are then selected based on their closeness to $\frac{1}{3}$ and $\frac{2}{3}$, and are considered the borders between the subimages.
4. The coordinates of the black border bounding box and the cuts are then used to crop out each of the subimages.

This algorithm performs decently when each of the specified assumptions is met. However, on images where one or more of the assumptions was violated, it leads to undesirable cropping. In the future, this could be improved by adding sanity checks for image height, and by defaulting to a $\frac{1}{3}, \frac{2}{3}$ crop in those situations. Moreover, this system of cuts and borders is somewhat unweidly, and could perhaps be replaced by averaging the values of various rows and columns of the array.

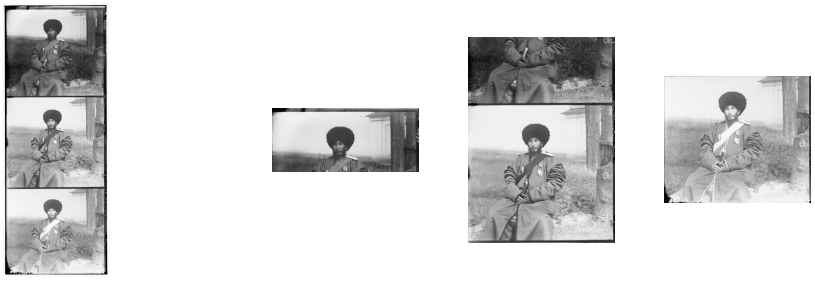


Figure 2: Example of a poor crop. The large dark patch accross the blue channel causes it to register as a cut.

Comparison of Results (SSD versus NCC)

Both the Sum of Squared Differences (SSD) and Normalized Cross Correlation (NCC) performed well on images with little variation between the intensities of their color channels, as is visible in figures 3 and 5.

(a) The unprocessed image

(b) The output of SSD

(c) The output of NCC

Figure 3: Comparison of NCC and SSD on the same image with similar results

(a) The unprocessed image

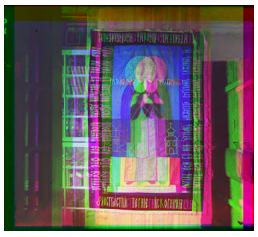
(b) The output of SSD

(c) The output of NCC

Figure 4: Another comparison of NCC and SSD and SSD on the same image with similar results

However, NCC outperformed SSD when the average value of one channel was significantly different than the value of another channel.

(a) The unprocessed image



(b) The output of SSD

(c) The output of NCC

Figure 5: An example of where SSD is outperformed by NCC. The mainly blue hues of the painting mean that the red and green channels are significantly different in value than the blue channel of the image, and so the Sum of Squared Differences becomes a less accurate measure of likeness in image positioning

Part 2

Resizing large images
The image resizing algorithm is relatively simple, though not very robust:

1. If an image is larger than 400 pixels across, a downscaled copy is created at 400px wide.
2. The downscaled copy is scored with NCC.
3. The original imgage is then scored again, over a range of values corresponding to the optimal displacements of the downscaled image.

Figure 6: Examples of large images processed in this manner

The runtime of this algorithm could be improved dramatically if it were made to use a Gaussian Pyramid (i.e. downsize by factors of two multiple times up to a threshold). Be that as it may, it is already a large step up from the naive approach, in which no scaling whatsoever is done.

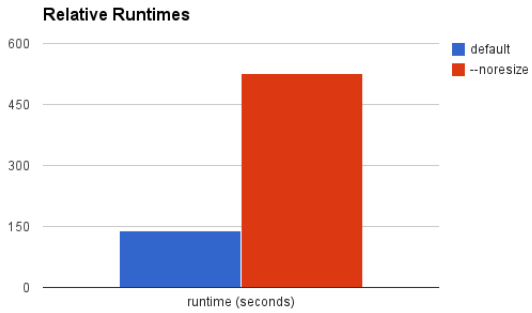


Figure 7: Runtime of processing a large image, with and without scaling.

Acknowledgements

Alexander Biggs, for pointing me onto a cleaner and more efficient way to crop the images. If only we head talked about it earlier in the assignment period, these runtimes could have been improved significantly.