Assignment: Automatically Aligning the Plates of Prokudin-Gorsky

October 17, 2014

As I discussed in class, the Prokudin-Gorsky plates are the result of 3 photographs of the same scene, each taken with a different color filter. This assignment uses the Prokudin-Gorsky photos and asks the question, "Can we develop an algorithm to automatically align the 3 color plates for each photo?" In an attempt to do this, you will explore alignment by maximization of mutual information. To produce good looking images, you will also have to figure out which third of the templates correspond to the red, blue, and green channels (they are not necessarily in order).

Please follow these steps.

1. As a warm-up exercise, write a Matlab ".m" file containing a function called entropy that computes and returns the entropy of a discrete probability distribution. You should pass the distribution as a vector. Obviously, since the vector represents a probability distribution, each element should be greater than or equal to zero and the elements should sum to 1. You should calculate the entropy in *bits*. Don't forget to handle the case in which some probablities are 0.

2. Once you've done this, write another function, called mutInfo, which computes and returns the mutual information of two variables given the joint distribution. You should pass the joint distribution in as matrix each of whose entries corresponds to a joint probability. You may want to have support functions (in other .m files) that return a marginal distribution given a joint distribution. *Hint: producing a marginal distribution from a joint distribution in Matlab can be done in one very short command. Look up the sum command.*

3. The next function you are going to write is another warm up. It will be called distributionFromImage.m. It will take as an argument one layer of an image (one third of a Prokudin-Gorsky photo), and another number saying how many different bins you want to describe the brightness values of the image. Since each filtered image has 256 values per pixel, you can choose to bin these values in up to 256 bins, or as few as 2 bins. The number of bins should be a power of 2. The function should simply return

the relative proportion of pixels from the image that fell into each bin. For example, if you choose 2 bins, then the function should return a vector of length 2. The first number in the vector would describe the proportion of values that fell in the range 0-127. The second number would describe the proportion of image values that fell in the range 128-255.

4. Now write a function called jointDistFromImages.m. The function takes as arguments two images of the same size, and a number of bins B from 2 to 256 (for each image), again a power of 2. It returns a joint distribution of brightness values in the two images in the form of a BxB matrix. In this function, you consider pairs of pixels (p_1, p_2) from the same location in the two different images. You should return a BxB matrix which gives the proportion of pixel pairs that fall in each bin, where a bin defines one range for the pixel from image 1, and another range for the pixel from image 2. For example, if the number of bins was for each image was 128, then one matrix element of the matrix should contain the proportional of pixel pairs (p_1, p_2) in which p_1 is 0 or 1 and p_2 is 12 or 13.

5. You are now ready to do automatic alignment of images. First load the full plates into matlab, and split them up into thirds, with each third corresponding to one of the photographs. Pick one of the three filtered images as a "base" that you will not change. Your first task will be to align the second filtered image to the first. To do this, write a double loop to shift the second image up to 15 pixels in each direction (+-15 in x, +-15 in y). For each new position, compute the mutual information between the first filtered image and the second shifted filtered image. Use the final position with the *maximum* mutual information between the two images. Then do this with the third image with respect to the first image.

6. You should perform this task with at least 3 different values of B, which defines the number of bins, and for each of the 8 images from problem set 2. Record how many of the automatic alignments worked well (out of 8) for each value of B. Comment on the different success rates for different values of B.

7. To visualize the full, aligned images, put each aligned image into one layer of a color image array which is width by height by 3 layers, and then use the Matlab command `imagesc` to display the color image. You should experiment with which layer is red, which is green, and which is blue until the final image looks as realistic as possible.

You should turn in all of the Matlab functions discussed above, and your composited color images (8 of them).