CS 129 Final Project Writeup

Greg Yauney (gyauney)
21 December 2012

I implemented a version of Image Analogies from the paper *Image Analogies* by Hertzmann et al. The basic premise of the algorithm is thus: given a pair of images, A and A', and an image B, synthesize an image B' that is to B as A' is to A. Or more simply: `A:A'::B:B'`

the algorithm

Some preprocessing happens first: each of the three input images is first converted from RGB to the YIQ luminance color space.

The core of the algorithm is finding a best match pixel from A for each pixel in B. There are two competing ways to find this best match.

The first is to find the pixel in A that is the nearest neighbor to the pixel in B in question. This is a relatively simple operation: extract a 5x5 patch around

the pixel in B, and compare it with every 5x5 patch in A. That is, for each pixel in B, its neighborhood is compared with the neighborhood around every pixel in A. A distance measurement is produced via gaussian-weighted sum of squared differences of these patches, and the pixel with the smallest distance is returned.

The second is found by considering the coherence of the images, and this is a little trickier than the nearest neighbor approach. For each already synthesized pixel within the current pixel's neighborhood in B': we look up that pixel's best match in A, and then we find the gaussian-weighted SSD of the neighborhood centered around that pixel with the neighborhood around the current pixel in B. The pixel with the smallest distance is returned as the best match.

These two distances are then compared, and the nearest neighbor distance measurement is multiplied by an arbitrary weight because even though the nearest neighbor's distance may be closer, the coherence approach usually picks a pixel that looks

better. Increasing this weight makes B' look more like A', and decreasing it makes B' closer to the original B. The images below were computed with a weight of 3. For the pixel in A that corresponds to the lesser of these two distances, the Y value of its corresponding pixel in A' is transferred to the pixel in question in B'. The color information, in the I and Q channels, is transferred over to B' from the equivalent pixel in B.

Once this process has been repeated for every pixel in B, the resulting image B' is converted back to RGB color space.

One of the key points of this approach is that even though the patches around the pixels in A and B are compared, each pixel in B' is updated individually—patches are not transferred from A' to B'. This is markedly different from the entirely patch-based approach to Image Analogies that was possible with the framework from project 4.

results

Two of the simpler examples I tried were a gaussian filter and a laplacian filter.

A

A'

B

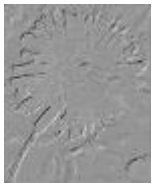
B'

A

A'

B

B'



I also tested my program with some of the artistically filtered images from the paper.

A

A'

B

B'

A

A'

B

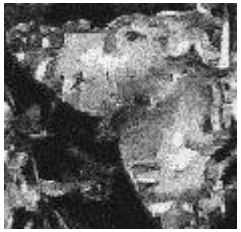
B'

A

A'

B

B'



There are, however, some limitations of my implementation. The most notable is that it is very, very slow with large images, hence the sheer number of really small images in my examples, and even then the larger images took upwards of half an hour. Instead of finding the exact nearest neighbor, I should've used an approximate nearest neighbor algorithm, as Hertzmann does in the paper.

And while not necessarily a limitation exactly, one of the main drawbacks of this approach is that there are

rather arbitrary parameters—the nearest neighbor weight, the neighborhood size—that need to be tweaked. When the neighborhood is larger, B' generally tends to be smoother. As I mentioned above, when the weight is increased, B' looks more like A', and when it is decreased, B' is more similar to the original B.

neighborhood size

3x3

5x5

nearest neighbor weight

weight = 3

weight = 10