Project 1 Writeup

Instructions

- I decided to only do sum of squared differences on a small window in the middle of the image instead of trying to match the entire image.
- The cropped window method speeds up my computation by a significant amount.
- I realized that constructing and finding the shift for the Gaussian Pyramid was a conducive to recursion, so it ended up being 5 lines of code!
- I implemented a contrast increasing function, but the results aren't what I want them to be.

Implementation

The algorithm without a Gaussian Pyramid is to fix one of the plates, and circular shift the other plate until my scoring metric, sum of squared differences, is at a minimum. I keep track of the shift corresponding to that minimum and return that value. To implement the Gaussian Pyramid, I wrote a recursive function that would in the base case return a call to the single-shift function. In the recursive case, the two plate inputs to the function are blurred (to reduce aliasing) and resized, and the resulting shift is multiplied by 2. The shift is multiplied by 2 when the image is reduced in size because in my resizing method, I remove every other row and column. This means that for every row shifted in the blurred image, two must be shifted in the larger size image.

I chose to implement a stationary window and a moving window. This essentially crops the image to 40% of its original size so that my scoring function (sum of squared differences) would be operating on a much smaller window.

```
1 windowStationary = a(floor(aheight/30):aheight-floor(
     aheight/30),floor(awidth/30):awidth-floor(awidth/30));
      % cropping image to set a middle window
2 windowMoving =     b(floor(bheight/30):bheight-floor(
     bheight/30),floor(bwidth/30):bwidth-floor(bwidth/30));
      % cropping image to set a middle window
3 ...
4 score = ssd(shifted,windowStationary);
```

Here is how I tried to implement contrast increasing:

```
1 function result = contrast(a)
2 highmask = a>0.9;
3 lowmask = a<(1-0.9);
4
```

```
a(highmask) = 1;
a(lowmask) = 0;
result = a;
end
```

Results

1. Result 1 (Figure 1) Here is the difference between my unaligned and aligned photo. The alignment vectors for this image is: green shift = 59, 21. red shift = 130, 27.
2. Result 2 (Figure 2) Here I tried to increase contrast, but the blacks in the photo had too much color to be properly crushed. I ended up with just a weird blown out look in the black where the cat's hind legs are.

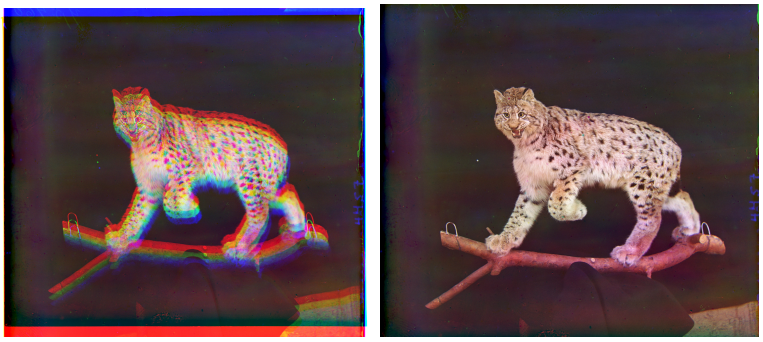

Figure 1: *Left:* Unaligned cat *Right:* Well aligned cat. 64MB photo took 11 seconds.

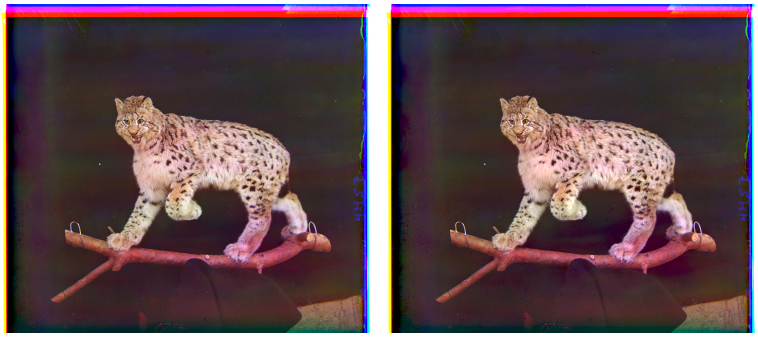

Figure 2: *Left:* No additional contrast added. *Right:* Added contrast by crushing top 10% of blacks and bottom 10% of whites.

The speedup for large images is much greater than speedup for small images. 1.

The speedup for small images is significantly smaller. All tests were done with a pyramid depth of 3. 2.

Condition	Time (seconds)
Average Brute force	114.5
Average Gaussian Pyramid	10.6

Table 1: Trials were done on roughly 3800x10000px images. Speedup is roughly 11 times for large images such as these.

Condition	Time (seconds)
Average Brute force	0.59
Average Gaussian Pyramid	0.14

Table 2: Trials were done on images roughly 400x1000px. Speedup is rouhgly 4.2 for small images such as these.