Eric Kearney
CS 390S - Digital Image Processing
Professor Feng Jiang
20, November 2018

Project 2 -- Visual Saliency report

Since I didn't purchase MatLab, I couldn't use the provided tool, however, I was able to quickly find another post by Adrian Rosebrock, of pyimagesearch to help me with this problem. I had to make some slight modifications, but I got this tool working fairly quickly. I'm not sure how well it compares to the provided MatLab tool, but I'm guessing it's an unfavorable comparison. This Python version has the unfortunate tendency to see blacked-out areas of an image as particularly interesting, which you'll be able to see for yourself soon enough.

I named the image quality metric I invented Thresholded Saliency Similarity, or TSS. A saliency map is generated for both the original and the distorted image, and each saliency map is thresholded to create a binary image. Finally, a bitwise XOR is computed over both the images. In other words, we're finding where the two binary images are different. The final step is to sum all the 'hits' of the bitwise XOR and divide by the original image size; the result will be 0 <= TSS <= 1, a value of 0 indicates the two images are identical, a value of 1 indicates they're completely different.

In practice, I don't think TSS works very well, it will likely claim two images are much more similar than they actually are, if the two images have a 'most interesting region' in a similar place, there will be a lot of 'hits' from the XOR, just because the pixels will naturally overlap with one another.

To run my program, use: **python 2program.py**. By default, it will look for a file called "lucy1_resized.jpg" to use as the original image and a file called "dog_destroyed.jpg" to use as the distorted image. You can pass in your own original image and distorted image using --image and --distorted, respectively.

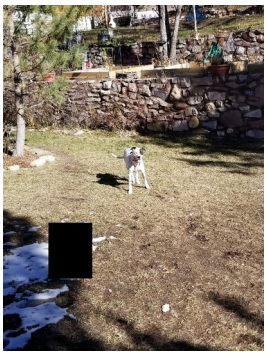
Results

Original

Saliency

Thresholded

Distortion 1

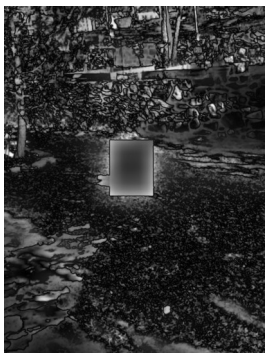


Saliency

Thresholded

Distortion 2

Saliency



Thresholded