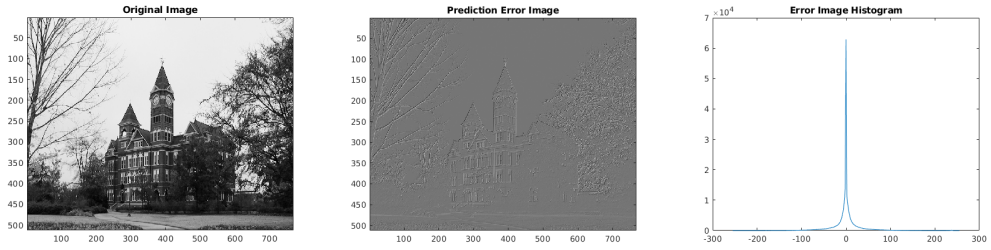
Matt Boler
ELEC7450 – Project 7

Using the image predictor function included at the end of this memo, an error image is generated from the original image of Samford Hall. It turns out that the values of surrounding pixels are a pretty good predictor of pixel values, as the vast majority of the error image is at or near zero. High error values occur at edges and corners. This makes sense as edges are large changes in value over small distances, which should be decorrelated from surrounding values. The original image, error image, and a histogram of error values are shown below.

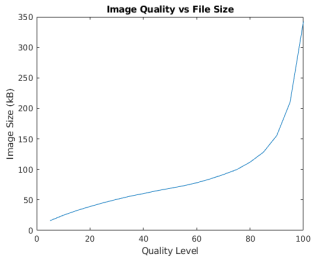


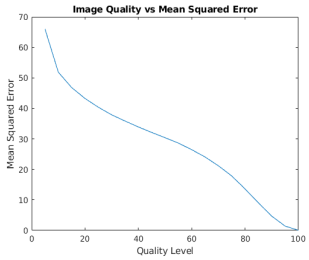
We can calculate the entropy of this error image by summing the product of the probability of each gray value and the base-2 log of this probability and then taking the negative of this sum. The original image gives an entropy of 7.26 while the error image gives an entropy of 5.96. While the default values for this predictor are a 50/50 split between the two sampled pixels, a split of 55/45 results in a lower entropy (5.95).

On a related note, we can evaluate the effectiveness of the compression implemented in the JPEG format. We can vary the coding quality parameter from 100 to 0 to effectively choose how much information to lose when coding the image. While high quality levels are almost indistinguishable from the original, levels below around 65 start to look unnatural. Below this level, texture loss and ringing around edges start to appear. At extremely low levels, such as quality level 20, there is noticeable blurring in high-texture regions (such as the grass), ringing around edges (such as the outline of the building against the sky), and mild staircasing of curves (such as around the archways). An image at level 20 is shown below.

When comparing the quality parameter to the resulting file size, we can see that there is an initial jump from 5 to 25, then little marginal cost to increasing the quality until you get to about 80, then file sizes skyrocket. Similarly, comparing quality vs mean squared error (MSE) reveals a brief rapid decrease in error until around quality 25, some leveling off until around quality 80, then another sharp decrease after. Both of these demonstrate lightly cubic behavior. The behavior moving from low quality to middling quality is representative of the facts of data storage, as adding a bit of resolution to a low resolution image represents both a high relative increase in information to store and a high relative increase in color accuracy. Moving from middling to high quality represents what I assume is a design

choice to maintain a similar apparent “quality setting increase to image quality increase” rate, as a larger increase in filesize is needed to overcome the diminishing returns of increased resolution.






```
function [errorImage, predictedImage] = predictImage(image, split)
%   split is [weight1, weight2]

predictedImage = zeros(size(image), 'double');
a = split(1);
b = split(2);

% NOTE: Function is undefined for m = 1 or n = 1 b/c (out of range)

for m = 2:size(image, 1)
    for n = 2:size(image,2)
        predictedImage(m, n) = a * image(m, n-1) + b * image(m-1, n);
    end
end

errorImage = image - predictedImage;

end
```