**Image Compression Techniques**

Authors:

*Christopher Zealand Miller*

*Carl Ostrenga*

**Summary:**

A large portion of today's internet traffic is images. With social media apps like Instagram and Snapchat only getting more popular, having good image compression is essential to not “clog the pipes”. Disregarding the internet having good image compression is good for local storage as well, saving space on your disc. There are two main types of image compression, lossy and lossless. Lossy is compressing an image while still maintaining a reasonable quality of the reconstructed image. Lossless is compressing an image and being able to reconstruct the original image exactly.

There are many ways to make an image take up less space on your hard drive. I will provide you with a summary of the potential ways to compress an image given the scope of the class.

* Change the color scheme from the standard RGB to another color scheme to see if it decreases storage size.
* Change the type of file that the image is stored as, this will allow the image to undergo further compression and through an algorithm can be stored in a different data structure than an image. This can later be changed back in the uncompressing process.
* After some pre compression preprocessing, encode the data of the image using Huffman encoding or some similar algorithm.
* Test various modifications/image transforms, scaling, blurring, edge detection, etc.

**Goals:**

Our goal for this project is to create a program that can compress and decompress and image file so that it is the same size or smaller than an image compressed using the LIB library. One point to note, the compression will most likely require the file being uncompressed before being viewed. With this in mind it allows for the storage of images in file types other than image types.

I have created a python program using LIB to use as a benchmark when comparing our programs compression ability. For the image I have selected the stock Lenna image that has been provided in previous projects. Below I have attached two screenshots showing the properties of the regular Lenna image and the compressed image

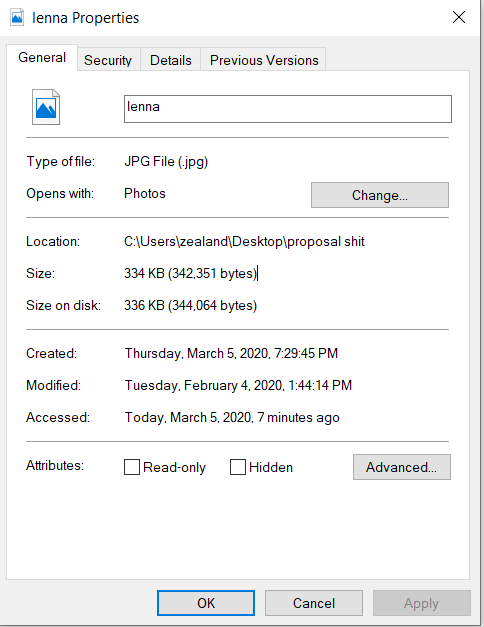
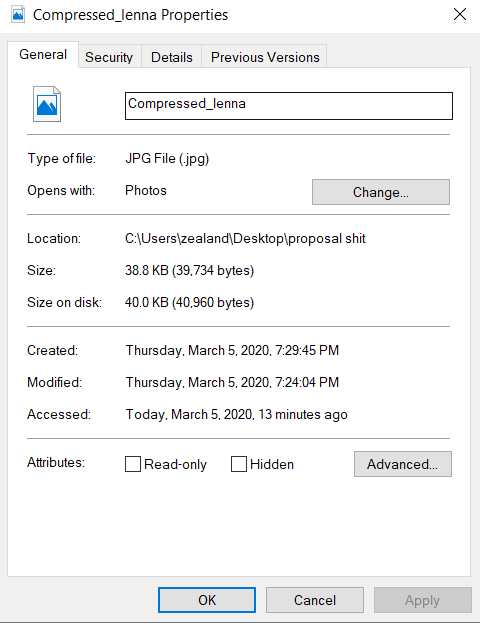
 

Figure 1: uncompressed lenna.jpg Figure 2: uncompressed lenna.jpg

Here you can see two images, the left image, Figure 1, is the uncompressed lenna.jpg it has a file size of (342,351 bytes). The image on the right, Figure 2, is the image that has been compressed by the program, it has a file size of (39,734 bytes). That is 11.6% of the original size.

These are some of the main goals that we would like to showcase through this project.

* Showcase the examples of the two different methods of image compression.
  + The first one will showcase a lossless image compression that keeps the integrity of the image as an image file.
    - (Meaning, that after it is compressed it can still be opened and viewed by an image viewer without any decompression.)
  + The second will attempt to showcase a lossless compression of an image that will remove the integrity of the image and will require uncompressing to the original image to be viewed.
    - (A more long term compression solution.)
* Compare the results between the two compressions and the baseline compression talked about above.
* Compare the original uncompressed images to the images post compression
  + Comparison points
    - The difference in file size in bits from pre to post compression
    - Verify the integrity of the image by uncompressing it (if needed) and comparing it to the original image to see if it is a lossy or lossless compression.
    - Contrast the various compression methods and suggest advantages and use cases each have

References:

Otemuyiwa, Prosper. “Python Image Optimization and Transformation.” *Cloudinary*, Cloudinary, 6 Feb. 2020, cloudinary.com/blog/image\_optimization\_in\_python.

Shantanujoshi, Joshi. “Compresssing Images in Python.” *Shantanujoshi.github.io*, 31 July 2016, shantanujoshi.github.io/python-image-compression/.