Assignment 3

This assignment is due by 11:59 p.m. on Monday, March 7. All reports should be submitted as PDFs. Only one report must be submitted per team, but all team member's names must appear on the report.

Part 1: Verifying an Image's Source Using JPEG Quantization Tables

In many scenarios, it is important to determine or verify the source of a digital image. One way to do this is by examining the metadata tags associated with an image. By default, most digital cameras append metadata tags indicating the camera model and manufacturer to images that they capture. Unfortunately, these metadata tags can be easily edited. As a result, a forger can falsify the source of a digital image by changing its metadata tags to indicate that a different digital camera was used to capture the image.

In class, we discussed a method to verify the source of a digital image using JPEG quantization tables. By default, most digital cameras store the images that they capture as JPEGs in order to save space. However, most digital cameras and editing software do not use the standard JPEG quantization tables when compressing an image. Instead, they use proprietary quantization tables that have been designed to better balance the trade-off between file size and image quality.

Since each camera or software manufacturer uses their own set of quantization tables, a forensic investigator can use this information to trace the source of a digital image. This is done by first reading the quantization tables used to encode and decode a JPEG from its header information. These tables can then be compared with a list of quantization tables used by different digital camera models and image editing software packages.

- In class, we discussed a software tool called JPEGsnoop that can be used to read the quantization tables from a JPEG's header. JPEGsnoop also compares these quantization tables to a list of known quantization tables used by different cameras and image editing programs.

  Download JPEGsnoop using the following link:

  www.impulseadventure.com/dl.php?file=JPEGsnoop_v1_6_1.zip

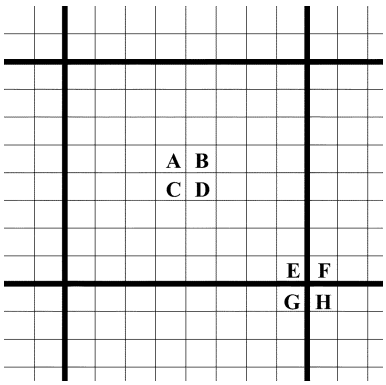
  Note: JPEGsnoop is only designed to run on computers using a Windows operating system. Linux users can reportedly run JPEGsnoop under wine and Mac users can run JPEGsnoop using CrossOver Mac.

- Use JPEGsnoop to analyze the following images: **imageOrigin1.jpg**, **imageOrigin2.jpg**, **imageOrigin3.jpg**, **imageOrigin4.jpg**, **imageOrigin5.jpg**, and **imageOrigin6.jpg**. What are the camera manufacturer and camera model reported in the metadata tags (Exif) for each image? Does each image have metadata tags specifying the camera manufacturer and model? What are the luminance and chrominance quantization tables for each image? Do these quantization tables match the camera reported in each image's metadata? If not, does JPEGsnoop report that these quantization tables match those used by any image editing software? Furthermore, if JPEGsnoop doesn't find a match between the quantization tables and the metadata tags, does this mean that the image's origin has been falsified? Why or why not?

- How could a forger fool falsify the origin of a digital image (i.e. pass the image off as having been captured by a different camera) and fool a program like JPEGsnoop?

Part 2: Detecting JPEG Compression Using Blocking Artifacts

Because an image's JPEG compression history can reveal a great deal of forensic information about an image, a forger may attempt to pass off a JPEG compressed image as an image that was never compressed. One way that a forger can attempt to do this is by decompressing a JPEG, then re-saving the resulting image in an uncompressed format such as bitmap or tiff.



In class, we discussed how JPEG compression leaves behind blocking fingerprints in a digital image. These blocking fingerprints can be detected using the algorithm proposed by Fan and de Quieroz in their paper "Identification of Bitmap Compression History: JPEG Detection and Quantizer Estimation". This algorithm operates by measuring the difference between pixel values at the center of each 8×8 block of pixels and comparing it to the difference between pixel values located at the corners of 4 separate 8×8 pixel blocks.

A detailed description of the algorithm is provided below:

1. For each 8×8 pixel block in the image, calculate the values

$$Z'(i,j) = |A - B - C + D| \tag{1}$$
$$Z''(i,j) = |E - F - G + H| \tag{2}$$

   where(i,j) denotes the block indices (i.e. the i^{th} block in the row direction and the j^{th} block in the column direction), and A, B, C, D, E, F, G and H are the values of the pixels in the positions shown in the figure above.

2. Calculate the normalized histogram $H_I(n)$ of Z' values and the normalized histogram $H_{II}(n)$ of the Z'' values.

   Note: A normalized histogram is calculated by dividing each histogram entry by the sum of all of the histogram entries. The Matlab function *hist* can be used to calculate the histograms of K' and K'' values.

3. Measure the strength K of the blocking fingerprints using the equation

$$K = \sum_n |H_I(n) - H_{II}(n)|. \tag{3}$$

4. Determine if the image was previously JPEG compressed by comparing the blocking fingerprint strength to a detection threshold η. The algorithm detects evidence of JPEG compression if $K > \eta$ and classifies the image as never compressed if $K \leq \eta$.

- Briefly explain why the histograms of K' and K'' values should be different if an image has been JPEG compressed.

- Write a Matlab function that calculates the implements Fan and de Quieroz's JPEG blocking artifact detection algorithm. This function should accept the image to be examined as an input and return the value K as an output. Additionally, this function should display the histograms H_I and H_{II} in the same plot. Please fully comment your code and append it to your report.

- Use the function you wrote to examine the images **blockArtifacts1.tif**, **blockArtifacts2.tif**, and **blockArtifacts3.tif** for blocking artifacts. Use $\eta = 0.25$ as your detection threshold. Include the K value that you measure for each image as well as plots of the histograms H_I and H_{II} in your report.

Part 3: Detecting JPEG Compression Using DCT Coefficient Quantization Fingerprints

In class, we discussed how JPEG compression leaves behind fingerprints in an image's DCT coefficients. These fingerprints are visible when examining the histogram of a single subband of an image's DCT coefficients. To calculate a DCT coefficient histogram, first, an image is first divided into 8×8 pixel blocks. The DCT of each block is computed and the appropriate coefficient is then added to a vector of containg the same DCT coefficient for each block in the image. The histogram (with a bin size of 1) of this vector is then computed.

- Write a Matlab function that calculates and displays a histogram of a single subband of an image's DCT coefficients. This function should accept as inputs both the image to be examined along with two variables (valued between 1 and 8) specifying the row and column index of the DCT subband to be examined. Please fully comment your code and append it to your report.

  Note: It may be helpful to create this function by modifying the code you wrote to simulate JPEG compression in Assignment 2.

- Use the function you wrote to examine the images **DCTfprints1.tif**, **DCTfprints2.tif**, and **DCTfprints3.tif** for DCT coefficient compression artifacts. Examine the (2,2) subband of DCT coefficients for each image and include a plot of the corresponding histogram for each image in your report.

- Which of these images have been JPEG compressed at least once? For images that have been JPEG compressed at least once, determine the quantization interval used when quantizing the (2,2) subband of DCT coefficients by visually inspecting the DCT coefficient histogram. If the image has been JPEG compressed twice, determine the quantization interval used during the last compression.

- Which of these images have been JPEG compressed twice? For images that have been compressed twice, determine whether the quantization interval used during the second compression was larger or smaller than the quantization interval used during the first compression.