ECES 435

Professor Matthew Stamm

Assignment 4

Wanyu Li and John R Seitz IV

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

What are the camera manufacturer and camera model reported in the metadata tags (Exif) for each image?

Image 1: Canon PowerShot A75
Image 2: Minolta Co., Ltd. DiMAGE S304
Image 3: Canon PowerShot SD400
Image 4: Minolta Co., Ltd. DiMAGE S304
Image 5: SONY DSC-V1
Image 6: Not listed in the EXIF, but matched the Sony Cybershot U.

Does each image have metadata tags specifying the camera manufacturer and model?

Image 1: Yes
Image 2: Yes
Image 3: Yes
Image 4: Yes
Image 5: Yes
Image 6: No, but matched when searching compression signatures.

What are the luminance and chrominance quantization tables for each image?

Image 1:

(Luminance)

DQT, Row #0: 1 1 1 2 3 6 8 10
DQT, Row #1: 1 1 2 3 4 8 9 8
DQT, Row #2: 2 2 2 3 6 8 10 8
DQT, Row #3: 2 2 3 4 7 12 11 9
DQT, Row #4: 3 3 8 11 10 16 15 11
DQT, Row #5: 3 5 8 10 12 15 16 13
DQT, Row #6: 7 10 11 12 15 17 17 14
DQT, Row #7: 14 13 13 15 15 14 14 14
Approx quality factor = 92.96 (scaling=14.08 variance=5.28)

(Chrominance)

DQT, Row #0: 4 4 5 9 15 26 26 26
DQT, Row #1: 4 4 5 10 19 26 26 26
DQT, Row #2: 5 5 8 9 26 26 26 26
DQT, Row #3: 9 10 9 13 26 26 26 26
DQT, Row #4: 15 19 26 26 26 26 26 26
DQT, Row #5: 26 26 26 26 26 26 26 26
DQT, Row #6: 26 26 26 26 26 26 26 26
DQT, Row #7: 26 26 26 26 26 26 26 26
Approx quality factor = 88.24 (scaling=23.52 variance=21.42)

Image 2:

(Luminance)

DQT, Row #0: 1 1 4 5 6 5 2 5
DQT, Row #1: 1 2 6 5 1 2 8 5
DQT, Row #2: 1 1 2 1 1 8 3 6
DQT, Row #3: 1 1 1 1 6 2 8 12
DQT, Row #4: 1 2 5 1 7 10 10 12
DQT, Row #5: 4 6 2 10 11 8 10 11
DQT, Row #6: 5 3 10 9 7 7 9 10
DQT, Row #7: 5 6 4 6 9 9 10 9
Approx quality factor = 94.65 (scaling=10.71 variance=62.00)

(Chrominance)

DQT, Row #0: 1 1 9 9 9 9 9 9
DQT, Row #1: 2 9 9 9 2 9 9 9
DQT, Row #2: 4 1 9 2 6 9 9 9
DQT, Row #3: 2 6 5 4 9 9 9 9
DQT, Row #4: 2 9 9 9 9 9 9 9
DQT, Row #5: 9 9 9 9 9 9 9 9
DQT, Row #6: 9 9 9 9 9 9 9 9
DQT, Row #7: 9 9 9 9 9 9 9 9
Approx quality factor = 94.92 (scaling=10.16 variance=47.37)

Image 3:

(Luminance)

DQT, Row #0: 1 1 2 2 3 3 7 14
DQT, Row #1: 1 1 2 2 3 5 10 13
DQT, Row #2: 1 2 2 3 8 8 11 13
DQT, Row #3: 2 3 3 4 11 10 12 15
DQT, Row #4: 3 4 6 7 10 12 15 15
DQT, Row #5: 6 8 8 12 16 15 17 14
DQT, Row #6: 8 9 10 11 15 16 17 14
DQT, Row #7: 10 8 8 9 11 13 14 14
Approx quality factor = 92.73 (scaling=14.53 variance=19.39)

(Chrominance)

DQT, Row #0: 4 4 5 9 15 26 26 26
DQT, Row #1: 4 4 5 10 19 26 26 26
DQT, Row #2: 5 5 8 9 26 26 26 26
DQT, Row #3: 9 10 9 13 26 26 26 26
DQT, Row #4: 15 19 26 26 26 26 26 26
DQT, Row #5: 26 26 26 26 26 26 26 26
DQT, Row #6: 26 26 26 26 26 26 26 26
DQT, Row #7: 26 26 26 26 26 26 26 26
Approx quality factor = 88.24 (scaling=23.52 variance=21.42)

Image 4:

(Luminance)

DQT, Row #0: 4 3 11 14 17 15 8 14
DQT, Row #1: 2 6 17 16 4 6 24 15
DQT, Row #2: 4 3 7 3 4 22 10 18
DQT, Row #3: 3 5 4 4 17 6 23 34
DQT, Row #4: 4 6 16 5 22 29 29 34
DQT, Row #5: 11 19 6 29 32 24 28 32
DQT, Row #6: 16 10 31 26 22 20 28 28
DQT, Row #7: 16 19 14 18 26 27 29 28
Approx quality factor = 83.81 (scaling=32.39 variance=477.61)

(Chrominance)

DQT, Row #0: 4 5 28 28 28 28 28 28
DQT, Row #1: 6 28 28 28 6 28 28 28
DQT, Row #2: 13 5 28 7 18 28 28 28
DQT, Row #3: 6 18 16 13 28 28 28 28
DQT, Row #4: 7 28 28 28 28 28 28 28
DQT, Row #5: 28 28 28 28 28 28 28 28
DQT, Row #6: 28 28 28 28 28 28 28 28
DQT, Row #7: 28 28 28 28 28 28 28 28
Approx quality factor = 84.00 (scaling=32.00 variance=450.51)

Image 5:

(Luminance)

DQT, Row #0: 1 1 1 1 2 3 4 5
DQT, Row #1: 1 1 1 2 2 5 5 4
DQT, Row #2: 1 1 1 2 3 5 6 4
DQT, Row #3: 1 1 2 2 4 7 6 5
DQT, Row #4: 1 2 3 4 5 9 8 6
DQT, Row #5: 2 3 4 5 6 8 9 7
DQT, Row #6: 4 5 6 7 8 10 10 8
DQT, Row #7: 6 7 8 8 9 8 8 8
Approx quality factor = 96.06 (scaling=7.87 variance=0.69)

(Chrominance)

DQT, Row #0: 1 1 2 4 8 8 8 8
DQT, Row #1: 1 2 2 5 8 8 8 8
DQT, Row #2: 2 2 4 8 8 8 8 8
DQT, Row #3: 4 5 8 8 8 8 8 8
DQT, Row #4: 8 8 8 8 8 8 8 8
DQT, Row #5: 8 8 8 8 8 8 8 8
DQT, Row #6: 8 8 8 8 8 8 8 8
DQT, Row #7: 8 8 8 8 8 8 8 8
Approx quality factor = 96.02 (scaling=7.97 variance=0.33)

Image 6:

(Luminance)

DQT, Row #0: 8 6 5 8 12 20 26 31
DQT, Row #1: 6 6 7 10 13 29 30 28
DQT, Row #2: 7 7 8 12 20 29 35 28
DQT, Row #3: 7 9 11 15 26 44 40 31
DQT, Row #4: 9 11 19 28 34 55 52 39
DQT, Row #5: 12 18 28 32 41 52 57 46
DQT, Row #6: 25 32 39 44 52 61 60 51
DQT, Row #7: 36 46 48 49 56 50 52 50
Approx quality factor = 74.75 (scaling=50.51 variance=0.81)

(Chrominance)

DQT, Row #0: 9 9 12 24 50 50 50 50
DQT, Row #1: 9 11 13 33 50 50 50 50
DQT, Row #2: 12 13 28 50 50 50 50 50
DQT, Row #3: 24 33 50 50 50 50 50 50
DQT, Row #4: 50 50 50 50 50 50 50 50
DQT, Row #5: 50 50 50 50 50 50 50 50
DQT, Row #6: 50 50 50 50 50 50 50 50
DQT, Row #7: 50 50 50 50 50 50 50 50
Approx quality factor = 74.74 (scaling=50.52 variance=0.19)

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?

Image 1: Yes
Image 2: Yes
Image 3: Yes
Image 4: No, there are no compression signatures in the current database that were found matching this make/model.
Image 5: Yes
Image 6: Since no camera was reported in the EXIF, the image is not known to be original or not. After searching compression signatures, it matched 1 camera and 15 different softwares. Based on analysis of compression characteristics and EXIF metadata, the image is processed or edited.

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?

This could mean that the image has been falsified, but does not guarantee it. For example, it could match a camera's characteristics that is just not in JPEGsnoop's database. Though, the database is extensive, so it is likely that the image is processed or editted.

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?

In order for this to work, not only would the EXIF data have to be modified to display the false camera, but also process the image with the same quantization tables of that false camera, resulting in a match of compression characteristics and EXIF metadata.

Part 2: Detecting JPEG Compression Using Blocking Artifacts

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

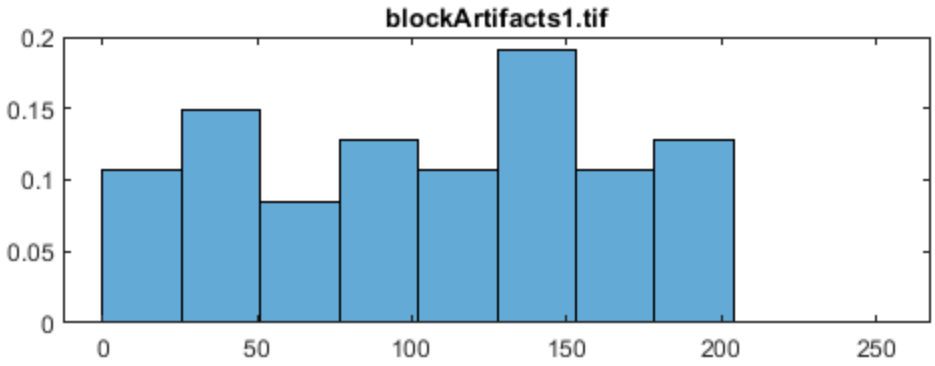
When an image is compressed, it leaves small but consistent discontinuities across block boundaries. This is taken advantage of by assuming that if there is no compression, then the pixel differences across blocks should be similar to those within blocks. In a compressed image, these pixel differences across adjacent blocks are not similar to those in the middle. By obtaining the K value, it can be compared to a threshold to determine if it was compressed or not. Since K is related to the area under the curves of the histogram, an original image and the compressed image's absolute histograms will not be the same.

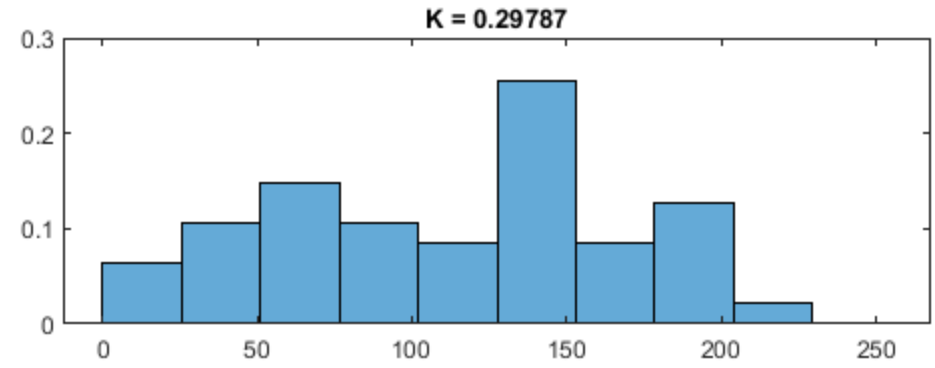
Use the function you wrote to examine the images blockArtifacts1.tif, blockArtifacts2.tif, and blockArtifacts3.tif for blocking artifacts. Use N = 0.25 as your detection threshold. Include the K value that you measure for each image as well as plots of the histograms H' and H'' in your report.

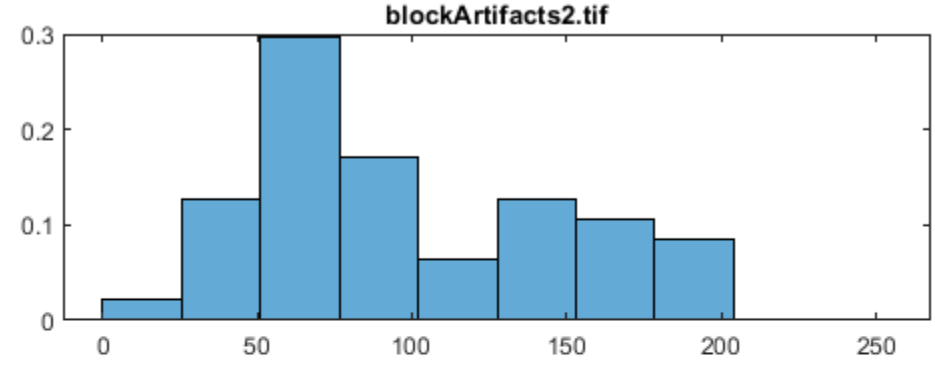
If N = 0.25, then each of the three images are compressed, because they all have a K value greater than N.

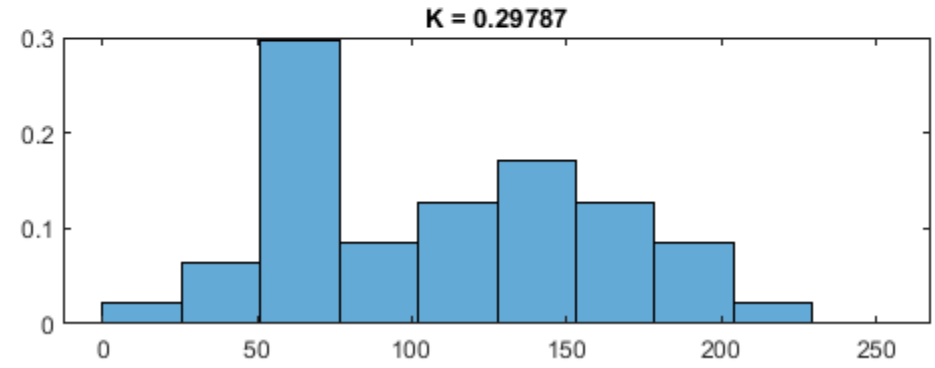
Image	blockArtifacts1.tif	blockArtifacts2.tif	blockArtifacts3.tif
K value	0.29787	0.29787	0.34043

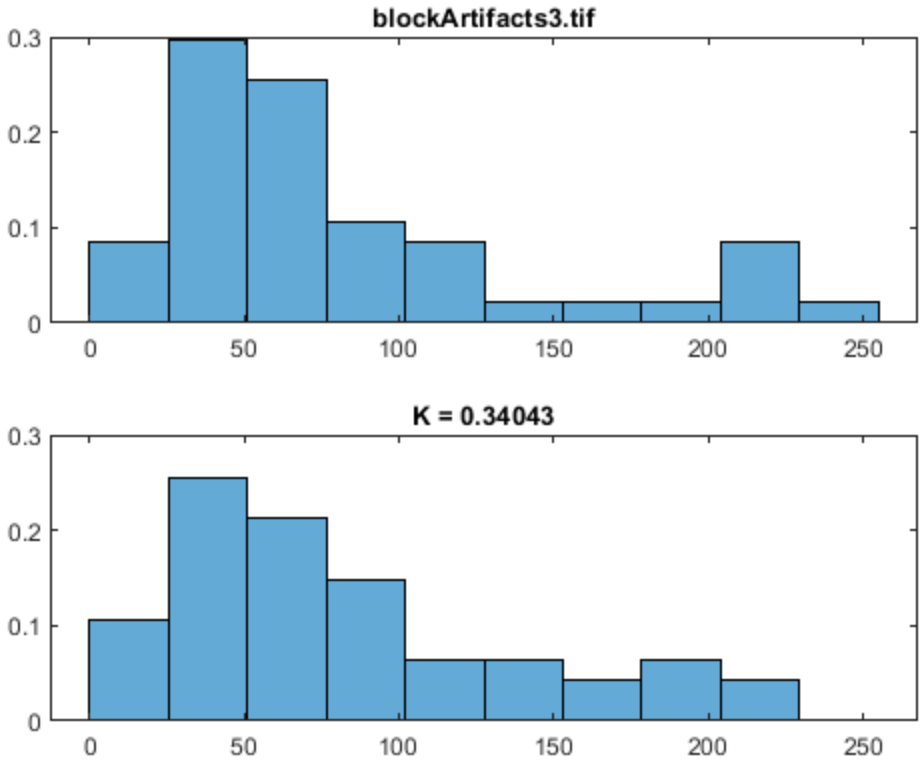
```
%ECES435 Assignment 4 Part 2 - By Wanyu Li and John Seitz
close all; clear all; clc;

% Examing 3 block artifact images using the created function to
 implement
% Fan and de Quieroz's JPEG blocking artifact detection algorithm

Imgs =
 {'blockArtifacts1.tif','blockArtifacts2.tif','blockArtifacts3.tif'};

for i=1:length(Imgs)
    figure(i)
    K = ArtifactDetector(Imgs{i});

    if K > 0.25 % Detect evidence of JPEG compression if K>n and
 classifies image as never compressed if K<n
    sprintf('Evidence of JPEG Compression')
else
    sprintf('Never Compressed')

    end
end

ans =
    'Evidence of JPEG Compression'
ans =
    'Evidence of JPEG Compression'
ans =
    'Evidence of JPEG Compression'
```


blockArtifacts1.tif
0.2
0.15
0.1
0.05
0
0
50
100
150
200
250


K = 0.29787
0.3
0.2
0.1
0
0
50
100
150
200
250


blockArtifacts2.tif
0.3
0.2
0.1
0
0
50
100
150
200
250


K = 0.29787
0.3
0.2
0.1
0
0
50
100
150
200
250



Function of Artifact Detector is below

```
type ArtifactDetector.m
```

```
function [K_prime] = ArtifactDetector(Imgs)

%This function is to implement the Fan and de Quieroz's JPEG blocking
%artifact detection algorithm following the steps outlined in the
%assignment PDF.

Img = imread(Imgs); % Read in the image

[x,y] = size(Img); % Set variables x and y to the size of the image
x = floor(x/8)-1; % Round the dimenions of the image, leaving out the
 right and bottom sides
y = floor(y/8)-1; % Round the dimenions of the image
Z_prime = zeros(x,y); % Create matrix of zeros of matching size
Z_2prime = zeros(x,y); % Create matrix of zeros of matching size

for i = 1:x % Repeat for each
    for j = 1:y
        new_1 = 8*(i-1)+4; % Where Z_prime is in the center of the
 block
        Z_prime(i,j) = abs(Img(new_1,new_1)-Img(new_1,new_1+1)-
Img(new_1+1,new_1)+Img(new_1+1,new_1+1));
```

```
        new_2 = 8*i; % Where Z_2prime is at the corner of blocks
        Z_2prime(i,j) = abs(Img(new_2,new_2)-Img(new_2,new_2+1)-
Img(new_2+1,new_2)+Img(new_2+1,new_2+1));
    end
end

bound = linspace(0,255,11); % Create boundaries for both histograms to
 ensure the same x axis

subplot(2,1,1) % plot the first histogram
H_1 =
 histogram(Z_prime,'BinEdges',bound,'Normalization','probability');
title(string(Imgs)) % Title is the name of the image file
K_prime = H_1.Values;

subplot(2,1,2) % plot the second histogram
H_11 =
 histogram(Z_2prime,'BinEdges',bound,'Normalization','probability');
K_2prime = H_11.Values;
K_prime = sum(abs(K_prime-K_2prime)); % Get the K value from the sum
 of the absolute value of H_11 minus H_11
title(['K = ',num2str(K_prime)]); % Title is the calculated K value

end
```

Published with MATLAB® R2019b