

Passive Digital Image Forgery Detection Techniques and Implementation

Mrugesha Lad¹, Naresh Patel²

Student, Electronics and Communication, Sarvajanic College of Engineering and Technology, Surat, India¹

Assistant Professor, Electronics and Communication, Sarvajanic College of Engineering and Technology, Surat, India²

Abstract: In today's world image authenticity is very important in many social areas. With the advent of low-cost and high-resolution digital cameras, and sophisticated photo editing software, such as Adobe Photoshop, GIMP, etc. digital images can be easily manipulated and tampered. When creating a digital forgery, it is often necessary to combine several images, for example, when compositing one person's head onto another person's body. If these images were originally of different JPEG compression quality, then the digital composite may contain a trace of the original compression qualities. We detect this type of tampering using JPEG ghost detection technique. We describe how resampling (e.g., scaling or rotating) introduces specific statistical correlations, and describe how these correlations can be automatically detected in any portion of an image using EM algorithm. Most digital cameras, for example, employ a single sensor in conjunction with a color filter array (CFA), and then interpolate the missing color samples to obtain a three channel color image. This interpolation introduces specific correlations which are likely to be destroyed when tampering with an image. We quantify the specific correlations introduced by CFA interpolation, and describe how these correlations, or lack thereof, can be automatically detected in any portion of an image. In this way, this paper describes some of the passive digital image forgeries like JPEG Ghost, resampling and CFA interpolation.

Keywords: Authenticity, Forgery, Types of forgery, Passive forgery detection, JPEG ghost, EM algorithm, CFA interpolation.

I. INTRODUCTION

Image authenticity is important in many social areas. For instance, the trustworthiness of photographs has an essential role in courtrooms, where they are used as evidence. Every day newspapers and magazines depend on digital images. In the medical field, physicians make critical decisions based on digital images. With the advent of low-cost and high-resolution digital cameras, and sophisticated photo editing software, such as Adobe Photoshop, GIMP, etc. digital images can be easily manipulated and altered. It is possible to change the information represented by an image and create forgeries, which are indistinguishable by naked eye from authentic photographs. This calls for a reliable forgery detection system for digital images[1]. Although good forgeries may leave no visual clues of having been tampered with, they may, nevertheless, alter the underlying statistics of an image making the forgery detection possible[2].

First, the possible forgeries are splicing, copy paste, image processing operations and false captioning. Existing digital forgery detection methods are divided into active and passive (or blind) approaches[1]. The active approaches are mainly based on digital watermarking and signatures. In contrast to active approaches, passive approaches do not rely on pre-registration or pre-embedded information and they have not been thoroughly researched. Passive techniques for image forensics operate in the absence of any watermark or signature.

In this paper, we mainly focus on developing passive techniques for detecting forgeries in digital images. Then different methods to detect those forgeries like Pixel based

(resampling like rotation and scaling), Format-based(JPEG-Ghost) technique, Camera-based(CFA interpolation) techniques are described in detail. Implementation results for all forgery detection techniques are also describe in detail.

II. FORGERY DETECTION TECHNIQUES

Here, we are going to deal with three types of forgery detection techniques. 1)Format based 2)Pixel based 3)Camera based. For format based forgery we used JPEG ghost detection.

A. Format Based Forgery Detection Technique(JPEG Ghost)

When creating a digital forgery, it is often necessary to combine several images, for example, when compositing one person's head onto another person's body. If these images were originally of different JPEG compression quality, then the digital composite may contain a trace of the original compression qualities. Here, describe a technique to detect whether the part of an image was initially compressed at a lower quality than the rest of the image. This approach is applicable to images of high and low quality as well as resolution.

In the standard JPEG compression scheme, a color image(RGB) is first converted in to luminance/ chrominance space (YCbCr). Each channel is then partitioned into 8 * 8 pixel blocks. These values are converted from unsigned to signed integers (e.g., from [0, 255] to [-128, 127]). Each block is converted to frequency

space by using a 2-D DCT. Each DCT coefficient c is then quantized by an amount q .

$$C = \text{round}(c/q) \quad (1)$$

where the quantization q depends on the spatial frequency and channel. Larger quantization values q yield better compression at the cost of image degradation. Quantization values are typically larger in the chrominance channels, and in the higher spatial frequencies, roughly modeling the sensitivity of the human visual system.

Consider now a set of coefficients c_1 quantized by an amount q_1 , which are subsequently quantized a second time by an amount q_2 to yield coefficients c_2 . With the exception of $q_2 = 1$ (i.e., no quantization), the difference between c_1 and c_2 will be minimal when $q_2 = q_1$. It is obvious that the difference between c_1 and c_2 increases for quantization value $q_2 > q_1$ since the coefficients become increasingly more sparse as q_2 increases. For values of $q_2 < q_1$, the difference between c_1 and c_2 also increases because although the second quantization does not affect the granularity of the coefficients, it does cause a shift in their values. Shown in Fig. 1, for example, is the sum of squared differences between c_1 and c_2 as a function of the second quantization q_2 , where $q_1 = 17$, and where the coefficients c_1 are drawn from a normal zero-mean distribution. Note that this difference increases as a function of increasing q_2 , with the exception of $q_2 = q_1$, where the difference is minimal. If q_1 is not prime, as in our example, then multiple minima may appear at quality values q_2 that are integer multiples of q_1 . As will be seen below, this issue can be circumvented by averaging over all of the JPEG DCT coefficients.

III. PAGE STYLE

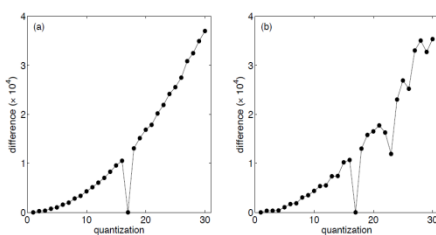


Fig. 1 Sum squared difference between coefficients quantized by different quality[4]

Figure Shown in panel (a) is the sum of squared differences between coefficients quantized by an amount $q_1 = 17$, followed by a second quantization in the range $q_2 \in [1, 30]$ (horizontal axis) – this difference reaches a minimum at $q_2 = q_1 = 17$. Shown in panel (b) is the sum of squared differences between coefficients quantized initially by an amount $q_0 = 23$ followed by $q_1 = 17$, followed by quantization in the range $q_2 \in [1, 30]$ (horizontal axis) – this difference reaches a minimum at $q_2 = q_1 = 17$ and a local minimum at $q_2 = q_0 = 23$, revealing the original quantization. Consider now a set of coefficients c_0 quantized by an amount q_0 , followed by quantization by an amount $q_1 < q_0$ to yield c_1 . Further

quantizing c_1 by q_2 yields the coefficients c_2 . As before, the difference between c_1 and c_2 will be minimal when $q_2 = q_1$. But, since the coefficients were initially quantized by q_0 , where $q_0 > q_1$, we expect to find a second minimum when $q_2 = q_0$. Shown in Fig.(b) is the sum of squared differences between c_1 and c_2 , as a function of q_2 , where $q_0 = 23$ and $q_1 = 17$. As before, this difference increases as a function of increasing q_2 , reaches a minimum at $q_2 = q_1 = 17$, and most interestingly has a second local minimum at $q_2 = q_0 = 23$. We refer to this second minimum as a JPEG ghost, as it reveals that the coefficients were previously quantized (compressed) with a larger quantization (lower quality).

Recall that the JPEG compression scheme separately quantizes each spatial frequency within a 8×8 pixel block. One approach to detecting JPEG ghosts would be to separately consider each spatial frequency in each of the three luminance/color channels. However, recall that multiple minima are possible when comparing integer multiple quantization values. If, on the other hand, we consider the cumulative effect of quantization on the underlying pixel values, then this issue is far less likely to arise (unless all 192 quantization values at different JPEG qualities are integer multiples of one another – an unlikely scenario!). Therefore, instead of computing the difference between the quantized DCT coefficients, we consider the difference computed directly from the pixel values, as follows

$$d(x, y, q) = \frac{1}{3} \sum_{i=1}^3 [f(x, y, i) - fq(x, y, i)]^2 \quad (2)$$

where $f(x, y, i)$, $i = 1, 2, 3$, denotes each of three RGB color channels and $fq(\cdot)$ is the result of compressing $f(\cdot)$ at quality q .

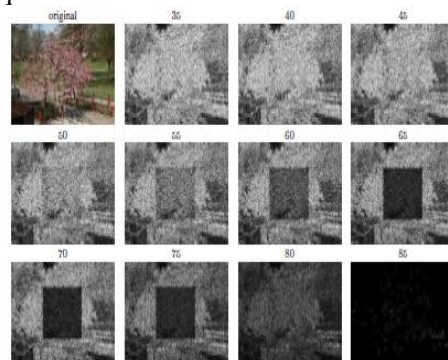


Fig. 2 Difference between original and resaved version of different quality image[4]

Fig.2 Shown in the top left panel is the original image from which a central 200×200 region was extracted, saved at JPEG quality 65, and re-inserted into the image whose original quality was 85. Shown in each subsequent panel is the difference between this image and a re-saved version compressed at different JPEG qualities in the range [35, 85]. At the originally saved quality of 65, the central region has a lower difference than the remaining image. Shown in the top left panel of Fig. 2 is an image whose central 200×200 pixel region was extracted, compressed at a JPEG quality of 65/100, and re-inserted

into the image whose original quality was 85. Shown in each subsequent panel is the sum of squared differences, Equation (2), between this manipulated image, and a re-saved version compressed at different JPEG qualities. Note that the central region is clearly visible when the image is re-saved at the quality of the tampered region (65). Also note that the overall error reaches a minimum at the saved quality of 85. The image difference is computed across all spatial frequencies, a region with small amounts of high spatial frequency content (e.g., a mostly uniform sky) will have a lower difference as compared to a highly textured region (e.g., grass). In order to compensate for these differences, we consider a spatially averaged and normalized difference measure. The difference image is first averaged across a $b \times b$ pixel region.

$$\delta(x, y, q) = \frac{1}{3} \sum_{i=1}^3 \frac{1}{b^2} \sum_{bx=0}^{b-1} \sum_{by=0}^{b-1} [(f(x + bx, y + by, i) - fq(x + bx, y + by, i))]^2 \tag{3}$$

and then normalized so that the averaged difference at each location (x, y) is scaled into the range $[0, 1]$

$$d(x, y, q) = \frac{\delta(x, y, q) - \min_q[\delta(x, y, q)]}{\max_q[\delta(x, y, q) - \min_q[\delta(x, y, q)]]} \tag{4}$$

Although the JPEG ghosts are often visually highly salient, it is still useful to quantify if a specified region is statistically distinct from the rest of the image. To this end, the two-sample Kolmogorov-Smirnov statistic is employed to determine if the distribution of differences, Equation(4), in two regions are similar or distinct. The K-S statistic is defined

$$k = \max\{C1(u) - C2(u)\} \tag{5}$$

where $C1(u)$ and $C2(u)$ are the cumulative probability distributions of two specified regions in the computed difference $d(x, y, q)$, where each value of q is considered separately.

1. Flow chart of detecting JPEG ghost :

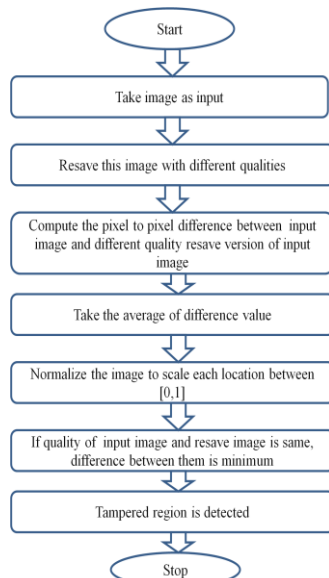


Fig. 3 Flow chart for JPEG Ghost detection

B. Pixel Based forgery Detection (Resampling)

In order to create a convincing match, it is often necessary to resize, rotate, or stretch portions of the images. This process requires resampling the original image onto a new sampling lattice. Although this resampling is often imperceptible, it introduces specific correlations into the image, which when detected can be used as evidence of digital tampering. If a person face is larger in one image it should be resized to the extent that faces size are similar in the composite image. This needs re-sampling the image that is to be composed on a new sampling and adding periodic correlations amongst the pixels in the neighborhood. Resampling is done by interpolation methods like nearest neighbor, bicubic and bilinear interpolation.

1. Detecting Resampling using EM algorithm^[7]

Consider resampling a signal by an arbitrary amount p/q . The i th sample of a resampled signal equal to a linear combination of its $2N$ neighbors,

$$y_i = \sum_{k=-N}^N \alpha_k * (y_i + k) \tag{6}$$

In order to determine if a signal has been resampled, we employ the expectation/maximization algorithm (EM) to simultaneously estimate a set of periodic samples that are correlated to their neighbors and the specific form of these correlations.

First assuming that each sample belongs to one of two models. The first model, M1, corresponds to those samples that are correlated to their neighbors, and the second model M2 corresponds to those samples that are not.

The EM algorithm is a two-step iterative algorithm:

- 1) in the E-step, the probability that each sample belongs to each model is estimated and
- 2) in the M-step, the specific form of the correlations between samples is estimated.

More specifically, in the E-step, the probability of each sample y_i belonging to model M1 can be obtained using Bayes' rule

$$Pr\{y_i \in M1 | y_i\} = \frac{Pr\{y_i | y_i \in M1\} Pr\{y_i \in M1\}}{\sum_{k=1}^2 Pr\{y_i | y_i \in Mk\} Pr\{y_i \in Mk\}} \tag{7}$$

$$Pr\{y_i | y_i \in M1\} = \frac{1}{\sigma\sqrt{2\pi}} \times \exp\left[-\frac{(y_i - \sum_{u,v=-N}^N \alpha_k * y_i + k)^2}{2\sigma^2}\right] \tag{8}$$

The variance of the Gaussian distribution is estimated in the M-step. Note that the E-step requires an estimate of α , which on the first iteration is chosen randomly. In the M-step, a new estimate of α is computed using weighted least-squares, that is, minimizing the following quadratic error function :

$$E(\alpha') = \sum_{i,j} w(i) \left(y_i - \sum_{u,v=-N}^N \alpha_k * y_i + k \right)^2 \tag{9}$$

where the weights $w(i) = 0$ and $\alpha_0 = 0$. This error function is minimized by computing the gradient with respect to α , setting the result equal to zero, and solving for α , yielding

$$\alpha = \frac{1}{(Y'WY)} Y'Wy \tag{10}$$

W is a diagonal weighting matrix with w(i) along the diagonal. The E-step and the M-step are iteratively executed until a stable estimate of α is achieved.

2. Flow chart for detect Resampling

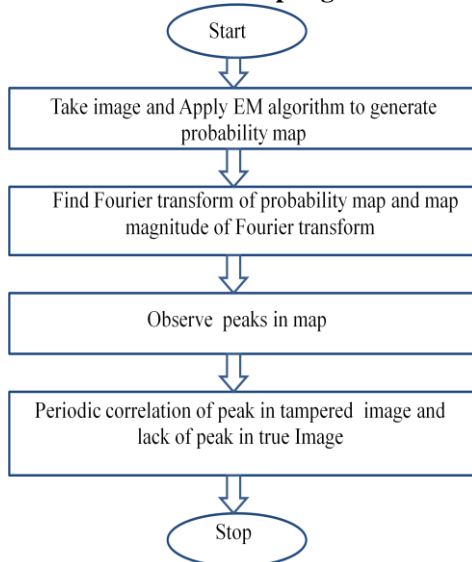


Fig. 4 Flow chart for detect Resampling

C. Camera Based Forgery Detection (Color filter array interpolation)

Now days, the digital cameras have single CCD or CMOS sensor and use color filter array (CFA). Generally the CFAs have RGB color filters placed at top of each sensor element, a single color sample is recorded at each pixel location, while the remaining two color samples are needed to be estimated from the neighboring samples. The process of estimating this missing color samples is called as CFA interpolation. In each color channel a correlation exists between subset of pixels which is induced due to color filter array interpolation. The color filters in a CFA are arranged in a periodic pattern, these correlations are periodic. As these correlations are periodic in nature they can be regarded as unique correlations can be used as a type of digital signature^[2]. Most digital cameras, for example, capture color images using a single sensor in conjunction with an array of color filters. As a result, only one third of the samples in a color image are captured by the camera and the other two thirds being interpolated. This interpolation introduces specific correlations between the samples of a color image. When creating a digital forgery, these correlations may be destroyed or altered.

Color filter array interpolation algorithm^[6] :

A digital color image consists of three channels containing samples from different bands of the color spectrum, e.g., red, green, and blue. Most digital cameras, however, are equipped with a single charge-coupled device (CCD) or complementary metal-oxide semiconductor (CMOS) sensor and capture color images using a color filter array (CFA). The most frequently used CFA, known as the

Bayer array, employs three color filters: red, green, and blue. The red and blue pixels are sampled on rectangular lattices, whereas the green pixels are sampled on a quincunx lattice. Since only a single color sample is recorded at each pixel location, the other two color samples must be estimated from the neighboring samples in order to obtain a three-channel color image.

$$R'(x,y) = S(x,y) \quad \text{If } S(x,y) = r \\ = 0 \quad \text{Otherwise} \tag{11}$$

$$G'(x,y) = S(x,y) \quad \text{If } S(x,y) = g(x,y) \\ = 0 \quad \text{Otherwise} \tag{12}$$

$$B'(x,y) = S(x,y) \quad \text{If } S(x,y) = b(x,y) \\ = 0 \quad \text{Otherwise} \tag{13}$$

Where (x,y) span an integer lattice. A complete color image, with channels R(x,y), G(x,y) and B(x,y), needs to be estimated. The estimation of the missing color samples is referred to as CFA interpolation or demosaicking. There are so many methods available for CFA interpolation^[6].

1. Bilinear interpolation^[6]

Bilinear interpolation estimates the value of an empty pixel as the average of the values of its nonempty neighbours: more precisely, it performs a convolution of the matrix representing each channel with a bilinear interpolation matrix. Therefore,

$$R = R' * \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{14}$$

$$G = G' * \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{15}$$

$$B = B' * \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{16}$$

2. Smooth hue Transition^[6]

In bilinear interpolation, there are visible chrominance aliases in areas with detail, where the interpolation of colour information has caused the detail to be slightly shifted between colour channels. To attempt to eliminate such chrominance distortions, one can make the assumption that in natural images colour hue varies smoothly. If we define two hues as the ratios of each chrominance component to the luminance, and assuming we have interpolated the luminance already, we can interpolate these hue values rather than the chrominance values to reduce colour aliasing. In the smooth hue transition algorithm, we approximate luminance with the green channel, with the red and blue as chrominance components. This is good enough, as half the pixels captured are green, and green is the major component in luminance (approximately 0.6). The green channel is bilinearly interpolated, and then used in the hue interpolation. The red and blue channels have their ratios with the green channel interpolated, and are then pointwise multiplied by the green channel.

$$G = G' * \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$$R = G^{pt.wise} * \left(\left(R^{pt.wise} \div G \right) * \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \right)$$

$$B = G^{pt.wise} * \left(\left(B^{pt.wise} \div G \right) * \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \right)$$

3. Median Filtering^[6]

All of the above methods contain 'speckles', or noise, near edges. This is caused, as above, by shifted edge information, however discontinuities in hue mean that the smooth hue transition does not reduce these artefacts. Since this noise is caused not by the individual channels themselves, but by differences between them, it should be possible to decrease this noise by applying a despeckling filter to the pairwise differences of the colour channels, and incorporating this despeckled difference back into the original. This is what median filtering does. Firstly, all the channels are interpolated using another interpolation scheme, usually simply bilinear. Then, their pairwise differences are median filtered. The resulting colour channels are constructed pixel-by-pixel from the CFA, where for each pixel in the CFA image, the corresponding median filtered difference pixel is added or subtracted. Say M_{rg} , M_{rb} , M_{bg} are the median filtered pairwise differences. Then, we take the example of $S_{1;0}$, which is the green pixel $G'_{1;0}$. In this case:

$$R(1;0) = S(1;0) + (M_{rg})_{1;0} \tag{20}$$

$$G(1;0) = S(1;0) \tag{21}$$

$$B(1;0) = S(1;0) - (M_{gb})_{1;0} \tag{22}$$

The resulting image has fewer speckles on edges, and the image could be median filtered again to further remove speckles. Indeed, dcrw (an open-source program dealing with, among other things, CFA interpolation) has a separate option of how many times to median filter its interpolated result.

Flow chart for CFA interpolation detection :

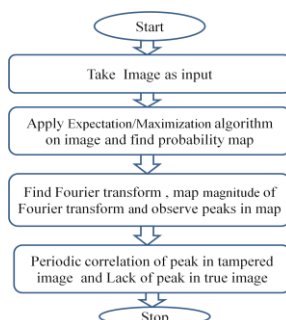


Fig. 5 Flow chart for CFA interpolation detection

III. IMPLEMENTATION AND SIMULATION RESULTS

1 . Experimental results of Tampering detection using JPEG Ghost

For implementing JPEG Ghost algorithm Image Data set taken from UCID (Uncompressed Color Image Database). All UCID color images are each of size 512 x 384. For creating tampered image, central portion from each image was removed, saved at specified JPEG quality q2, reinserted into image, and then the entire image was saved at the same or different quality of q1^[4]. Matlab function imwrite was used to save the image in the JPEG format. The size of the central region ranged from 150 x 150 to 300 x 300 pixels. The JPEG quality q1 was selected randomly in the range 40 to 90, and the difference between JPEG qualities q1 and q2 ranged from 0 to 25, where $q1 \leq q2$ ^[4].

A. Implementation of JPEG ghost algorithm

1) Input image : ucid00006.tif from UCID

Image size : 512 x 384

Altered region : first 200 x 200 pixel block
(20% tampering))

JPEG Quality of original image : 90

JPEG Quality of altered region : 30



Fig.6 Original Image

Fig. 7 Tampered Image

Matlab output for image ucid00006.tif

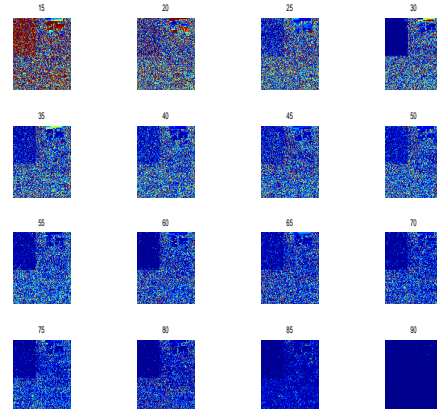


Fig. 8 Matlab output for JPEG ghost

In figure, each subsequent panel is the difference between this image and a resaved version compressed at different JPEG qualities in the range [15,90]. JPEG ghost is detected at quality 30.

2) Input image : pic1.jpg (Image size : 413 x 531)

Altered region : Central 296 x 296 pixel
(40 % tampering)

JPEG Quality of original image : 90

JPEG Quality of altered region : 50



Fig. 9 Original Image



Fig. 10 Tampered Image

Matlab Output for pic1.jpg

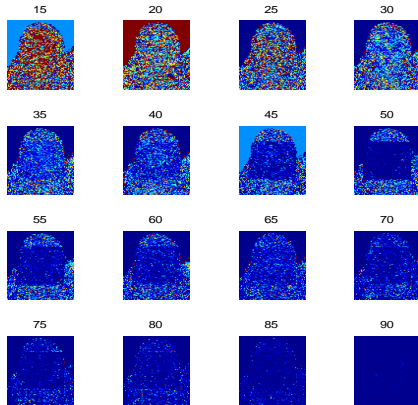


Fig. 11 Matlab Output for pic1.jpg

1)Input Image : ucid00006.tif

Table I: Comparison between original image and tampered image

Tampered Region in Percentage(%)	Tampered Region in Pixel	PSNR
10	140 x 140	31.1700
20	200 x 200	30.0828
30	242 x 242	29.5305
40	280 x 280	29.0721

2)Input Image : pic1.jpg

Table II: Comparison between original image and tampered image

Tampered Region in Percentage(%)	Tampered Region in Pixel	PSNR
10	148 x 148	45.6252
20	210 x 210	42.6183
30	255 x 255	40.8022
40	296 x 296	39.4960

Form observed output of table 1 and table 2 , we can say that if tampering is increase in image, PSNR is decrease.

B.Accuracy Measurement

Data set : UCID (uncompressed image dataset)

Image data : 100 Images from UCID

- 1)70 mages tampered (ucid00001.tif to ucid0070.tif)
- (ucid0001.tif to ucid0020.tif) 10 % tampering
- (ucid0021.tif to ucid0040.tif) 20 % tampering
- (ucid0041.tif to ucid0060.tif) 30 % tampering
- (ucid0061.tif to ucid0070.tif) 40 % tampering

2)30 Images are true (no tampering) (ucid0071.tif to ucid0100.tif). Accuracy is measured from TPR, TNR , FPR and FNR.

Table III
Accuracy measurement


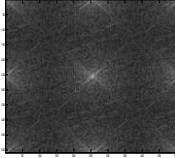
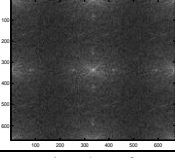
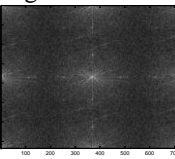
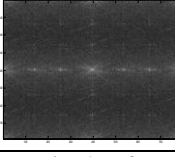
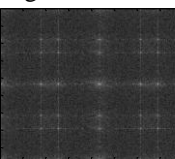
Tampered region with respect to original image region	Quality difference between tampered region and original image region	Accuracy (%)
40%	30	89
40%	40	91.14
40%	50	93.27
40%	60	95.45

If quality difference between original image region and tampered region is increase than Accuracy of detection algorithm is increase.

2. Experimental results of of resampling (tampering) detection using EM algorithm

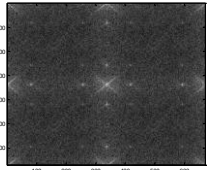
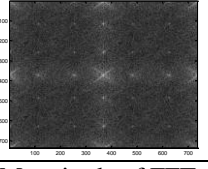
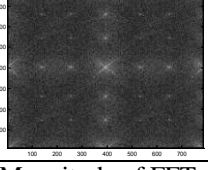
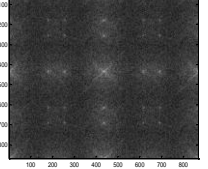
Here, we generate tampered image using adobe photoshop. We have done tampering like upsampling, down sampling, rotation (CW and CCW) using nearest neighbor interpolation, bilinear interpolation and bicubic interpolation.to detect resampling we have used EM algorithm.

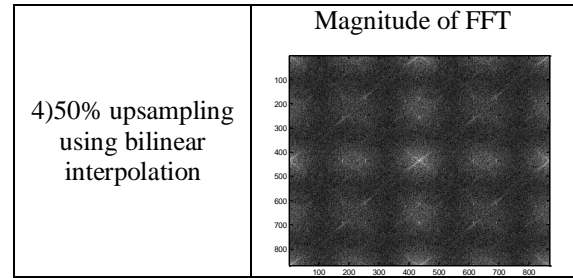
A.Upsampling using Nearest Neighbour interpolation :

Original Image: Cameraman.tif 	Magnitude of FFT 
1)10% upsampling using nearest neighbor interpolation	Magnitude of FFT 
2)20% upsampling using nearest neighbor interpolation	Magnitude of FFT 
3)30% upsampling using nearest neighbor interpolation	Magnitude of FF 
4)50% upsampling using nearest neighbor interpolation	Magnitude of FFT 

B. Upsampling using Bilinear interpolation :

Input Image : cameraman.tif Size : (256 x 256)

1)10% upsampling using bilinear interpolation	Magnitude of FFT 
2)20% upsampling using bilinear interpolation	Magnitude of FFT 
3)30% upsampling using bilinear interpolation	Magnitude of FFT 
3)50% upsampling using bilinear interpolation	Magnitude of FFT 



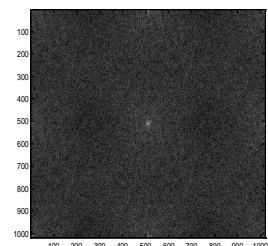
D. Clockwise Rotation using Bilinear interpolation :

Input Image : Baboon.png

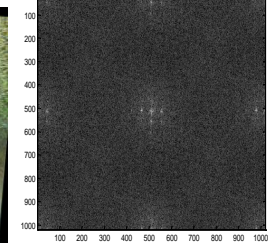
Original Image



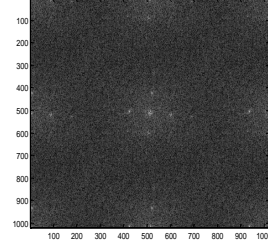
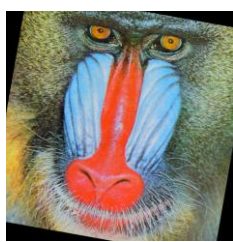
Magnitude of FFT



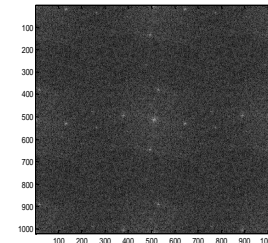
1)5° rotation : (Clockwise direction)



2)10° rotation : (Clockwise direction)



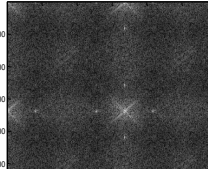
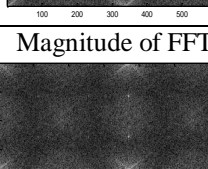
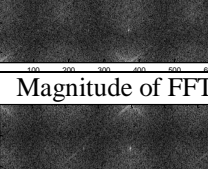
3) 15° rotation : (Clockwise direction)



C. Upsampling using Bicubic interpolation :

Input Image : cameraman.tif

Size : (256 x 256)

1)10% upsampling using bicubic interpolation	Magnitude of FFT 
2)20% upsampling using bicubic interpolation	Magnitude of FFT 
3)30% upsampling using bicubic interpolation	Magnitude of FFT 

For pixel based forgery, tampered image is detected using EM algorithm and they have localized peak in magnitude of FFT.

- True image have no localized peak.
- If we increase upsampling percentage the distance between peak also increase.

- Different interpolation have different peak pattern peak in magnitude of FFT.
- Peak pattern are periodic in all interpolation.
- If we rotate image than peak in magnitude of fft is also rotate as per degree of rotation(clock wise and counter clock wise)

3.Experimental results of CFA forgery detection using EM algorithm

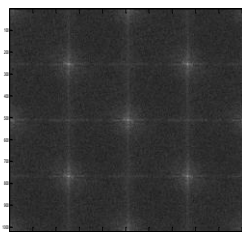
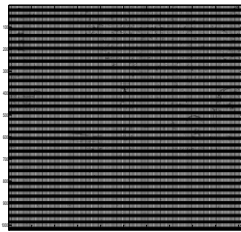
EM algorithm results for image sign.tif size (512 x512)
Original Image



1) Bilinear interpolated Image : sign_bl.tif

Magnitude of FFT

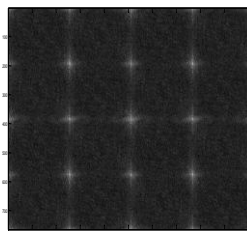
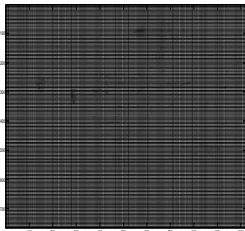
Probability Maps



2)Median filtered interpolated image : Sign_md.tif

Magnitude of FFT

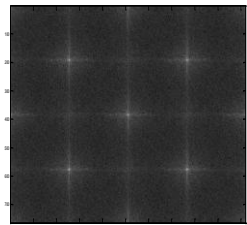
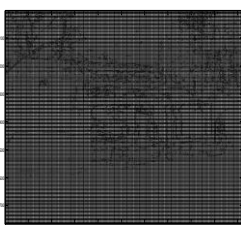
Probability Maps



3) Smooth hue Transition : Sign_sh.tif

Magnitude of FFT

Probability Maps



IV. CONCLUSION

Today's technology allows digital media to be altered and manipulated in ways that were impossible 20 years ago. Tomorrow's technology will certainly allow us to manipulate digital media in ways that today seem unimaginable. And as this technology continues to evolve,

it will become increasingly important for the science of digital forensics to try to keep pace. There is little doubt that as we continue to develop techniques for exposing photographic frauds, new techniques will be developed to make better frauds that are harder to detect. While some of the digital forensic tools may be easier to fool than others, some tools will be difficult for the average user to circumvent. Good forgery leave no clues of having been tampered so mainly focuses on developing passive techniques for detecting forgeries in digital images. Efficient and robust algorithms are described for the methods to detect forgeries. Format based forgery detection algorithm for JPEG ghost is implemented using matlab software and the Results shows that for higher quality difference between image region and tampered region gives higher accuracy. For pixel based forgery, tampered image is detected using EM algorithm and they have localized peak in magnitude of FFT. For camera based forgery, tampered image is detected using EM algorithm and have localized peak in magnitude of FFT. The digital image forgery detection has emerged to help return some trust to digital image.

REFERENCES

- [1] Farid, H., "Image forgery detection" in Signal Processing Magazine, Magazine of IEEE, vol.26, no.2, pp.16-25, March 2009.
- [2] Gajanan K. Birajdar, Vijay H. Mankar, "Digital image forgery detection using passive techniques: A survey", Digital Investigation, Journal of ELSEVIER, Volume 10, Issue 3, October 2013, Pages 226-245, ISSN 1742-2876.
- [3] Arun Anoop, M., "Image forgery and its detection: A survey," in Innovations in Information, Embedded and Communication Systems (ICIIECS), 2015 International IEEE Conference on, vol.,no.,pp.1-9,19-20, March 2015.
- [4] Farid, H., "Exposing Digital Forgeries From JPEG Ghosts," in Information Forensics and Security, IEEE Transactions on, vol.4, no.1, pp.154-160, March 2009.
- [5] J. Fridrich, D. Soukal, and J. Lukás, "Detection of copy move forgery in digital images," in Proc. Digital Forensic Research Workshop, Aug. 2003.
- [6] Popescu, A.C.; Farid, H., "Exposing digital forgeries in color filter array interpolated images," in Signal Processing, IEEE Transactions on, vol.53, no.10, pp.3948-3959, Oct. 2005.
- [7] Popescu, A.C.; Farid, H., "Exposing digital forgeries by detecting traces of resampling," in Signal Processing, IEEE Transactions on, vol.53, no.2, pp.758-767, Feb.2005.
- [8] Fangjun Huang; Jiwu Huang; Yun Qing Shi, "Detecting Double JPEG Compression With the Same Quantization Matrix," in Information Forensics and Security, IEEE Transactions on, vol.5, no.4, pp.848-856, Dec. 2010.
- [9] W. Luo, Z. Qu, J. Huang, and G. Qiu, "A novel method for detecting cropped and recompressed image block," in Proc. IEEE Conf. Acoustics, Speech and Signal Processing, Honolulu, HI, 2007, pp. 217-220.
- [10] Y.-F. Hsu and S.-F. Chang, "Image splicing detection using camera response function consistency and automatic segmentation," in Proc. Int. Conf. Multimedia and Expo, Beijing, China, 2007.
- [11] I. J. Cox, M. L. Miller, and J. A. Bloom, Digital Watermarking. San Francisco, CA: Morgan Kaufmann, 2002.
- [12] Z. Lin, R. Wang, X. Tang, and H-V Shum, "Detecting doctored images using camera response normality and consistency," in Proc. Computer Vision and Pattern Recognition, San Diego, CA, 2005.
- [13] Z. Fan and R. L. de Queiroz, "Identification of bitmap compression history: JPEG detection and quantizer estimation," IEEE Trans. Image Process., vol. 12, no. 2, pp. 230-235, 2003