Image Processing Pipeline Model Integrating Steganographic Algorithms for Mobile Cameras

Prabhat Dahal
Department of Electrical and Computer Engineering
University of Nebraska-Lincoln, USA
prabhat.dahal@huskers.unl.edu

Dongming Peng
Department of Electrical and Computer Engineering
University of Nebraska-Lincoln, USA
dpeng2@unl.edu

Hamid Sharif
Department of Electrical and Computer Engineering
University of Nebraska-Lincoln, USA
hsharif@unl.edu

ABSTRACT

Mobile phones these days are extensively used to capture and share multimedia right from the device without the basic security processing like watermarking. This necessitates the need for multimedia to be watermarked right within the mobile hardware for copyright protection and authentication. In this paper, we propose an image processing pipelining model for mobile cameras that integrates watermarking algorithm with the image capturing processes in cameras. Differently from existing works reported in literature, the unique contribution in this paper is to have image watermarked while in the early capturing process. In other words, the processes for digital image capturing and watermarking have become as a single non-separable procedure in the camera hardware systems. The paper proposes watermarking raw image data before it becomes display ready within the image capturing process. Also, the watermarking algorithm used to develop the model, without losing generality, caters to the computational efficiency requirement for mobile devices while being robust against image coding at the same time. Simulations results demonstrate that the proposed approach is robust against JPEG compression.

CCS Concepts

• **Security and privacy→Security services→Authentication.**

Keywords

Watermarking; Steganography; Mobile Camera; Color Filter Array; Information Hiding.

1. INTRODUCTION

There have been numerous efforts in securing digital images before they are redistributed [16], [17] by watermarking or steganography. Different mechanisms have developed over the years with new ones being more robust than the previous ones. However, majority of such efforts are concentrated on processing the images after they have been captured and transferred to the computer just before they are re-distributed. But this leaves enough room for a user to secure copies of the image before watermarking and sharing it. In addition, easy availability of internet in mobile devices itself doesn't provide enough leverage to wait for the images to be transferred to the computer before they are shared. Reasons as such call for mechanisms that can watermark images right during the capturing phase so that all images coming out of the camera include authenticity information by default and no copy of un-watermarked exists at all.

The idea here is to remove the gap between the image acquisition and processing before re-distribution. There have been limited works that utilize this gap and bring watermarking close to the acquisition stage. Efforts have been made to overlap the process of watermarking with acquisition phase and obtain a real-time watermarking solution embedded within the camera hardware itself. Literature reveals plethora of methods for watermarking an image [1]-[5], [18]-[28]. Watermarks in multimedia can be inserted in spatial or frequency domains utilizing different signal processing techniques and can be either visible or invisible. Some tend to be more robust against different attacks while others are focused on maintaining desirable image quality. Each has it pros and cons. However the problem lies in the fact that these methods cannot be readily implemented within the existing image acquisition phase.

There are specific set of basic steps in a digital camera called an Image Sensor Pipeline (ISP) that the sensor data go through until an image that is ready for human perception is created. Watermarking within the camera hardware is nothing but integration of watermarking algorithm somewhere in the ISP. Each stage within the ISP modifies image data from the previous stage in order to create the final image. Simple insertion of an existing watermarking algorithm within the pipeline might cause the algorithm and ISP processes to interfere with each other. Hence, we aim to understand these processes and create an algorithm customized to fit into the existing ISP to generate a watermarked image as the final result of the image acquisition phase.

The unique contribution in this paper is the watermarked image processing pipelining model for mobile cameras that integrates watermarking algorithm with the generic ISP-based image capturing processes in generic digital cameras. This will avoid any post processing requirement before the image is shared and will ensure that each shared image is automatically and inherently watermarked. In other words, the processes for capturing and watermarking digital images have become as a single non-separable procedure in the camera hardware systems. The proceedings are the records of the conference. ACM hopes to give these conference by-products a single, high-quality appearance. To do this, we ask that authors follow some simple guidelines. In essence, we ask you to make your paper look exactly like this

WTMC'16, May 30 2016, Xi'an, China.
© 2016 ACM. ISBN 978-1-4503-4284-1/16/05 …$15.00.
DOI: http://dx.doi.org/10.1145/2903185.2903191

document. The easiest way to do this is simply to download a template from [2], and replace the content with your own material.

2. RELATED WORKS

There has been limited but laudable research in implementing watermarking in the camera firmware. The authors in [6] propose a secure digital camera utilizing lossless watermarking. They embed biometric identifier (iris image) of the photographer combined with image's cryptographic hash and other forensic data to verify the image integrity. In order to keep the image appearance unchanged, they efficiently utilize the JPEG quantization steps and the Discrete Cosine Transform (DCT) coefficients to perform Least Significant Bit (LSB) embedding. However, the embedding done isn't essentially a part of the camera ISP as watermarking is done in the final image from the camera sensor and not on the intermediate raw data. Also, implementation of this method requires modification of existing camera viewfinder to get the iris image.

In [7], the authors propose combination of semi-fragile information along with more robust biometrics to create a watermark that can be used for integrity detection as well as ownership protection. The paper claims to secure images from a digital camera but does not show the integration of the proposed method with any digital camera at all.

The authors in [8] propose a detailed hardware implementation of robust watermarking of color image combined with encryption to provide a Secure Digital Camera (SDC) system. They extensively describe the details of implementation on Field Programmable Gate Array (FPGA) but prelude the integration of watermarking into the processes within the ISP. The work in [8] is important from the perspective of analyzing the encryption and embedding processes in hardware. The work in [29] focuses on the separable watermark implementation on embedded computing platforms.

In [9], Nelson *et al*. address the issue of lack of sensor level integration of watermarking. They propose a chip design that can be integrated with Complementary Metal-Oxide-Semiconductor (CMOS) sensor for hardware level watermarking. This is a significant effort in the field of watermarking in cameras but cannot be incorporated as a software into the existing ISP that the cameras follow.

In [10], the authors propose embedding a visible watermark in the camera sensor data and transferring the watermark to the final image using the demosaicing algorithm. The prime purpose in the paper was to verify image authenticity by simple inspection of the visible watermark.

The authors in [11] propose a method that differs from the ones described above provided that it is a software-only implementation. The authors compare the usage of different demosaicing algorithms and how they affect performance. However, the embedding algorithm uses a simple spread spectrum approach.

It is worth mentioning that several camera manufacturers like Epson and Kodak have manufactured cameras with watermarking abilities in the past [6]. Epson required that the users purchase a software called Image Authentication System (IAS) while Kodak had some inbuilt features in the camera that could insert visible watermark. However, the Kodak camera models that supported these features have been discontinued and are no more available in any Kodak consumer cameras.

Despite some of the landmark efforts as discussed above, they suffer from one or more of the shortcomings listed below:

(i) Requirement of additional hardware in the camera for implementation of proposed method.

(ii) Focused on making the watermark more secure by the use of encryption and hashing function and not on the process of watermarking.

(iii) Utilization of primitive watermarking technique such as LSB embedding that is no more considered robust as compared to many others.

(iv) Not implemented within the camera ISP and are only appended to the camera hardware after the final stage of image generation from the camera (which is no different from a normal image watermarking).

(v) Lack integrity analysis of the extracted watermark (primarily because they don't need to, as most of them are simply hardware realization of the existing general image watermarking algorithms).

In this paper, we address the aforementioned issues and our method is proposed to be a part of the camera ISP which sidelines the requirement of post processing any image after it has been captured.

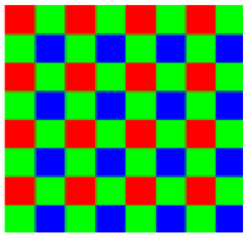
Figure 1. A Typical Color Filter Array

3. SYSTEM ARCHITECTURE

The first set of digital data that the camera produces is an array of single channel intensity, most commonly known as the raw image. This data contains the true information and the camera component that generates this is the photo sensor in conjunction with the Color Filter Array (CFA) [15]. The CFA has a specific pattern as shown in Figure (1). The CFA pattern is such that it allows for the interpolation of missing color information at any location. This process is called demosaicing [24]. For the sake of simplicity, we choose to work with the most common 'rggb' pattern where, in a 2 x 2 array of the Color Filter. The result of demosaicing is an m x n x 3 (Red Blue Green) RGB image that we expect out of any camera.

It should be noted that there is a series of steps that the sensor data has to go through to make the color image more meaningful and realistic. These processes can be specific to different camera models and might not be made public, but the basics are pretty much the same and common to all models and manufacturers. A generic Image Processing Pipeline for any digital camera would include the processes as shown in Figure 2 and discussed later in this section. The major processes are linearization, white balancing, demosaicing, color space conversion and brightness and gamma control before the final eye-ready image is created. In this paper, we propose to interject a watermark embedding algorithm right after the color space conversion phase, before the image goes through the final brightness and gamma correction. The reason behind the choice of this point of embedding is to ensure that the watermark, which is already something meaningful, doesn't undergo any irreversible changes which the raw data might have to go through. The final meaningful image won't be ready until the last process in Figure 2, but the watermark might be easily corrupted then, provided that changes that the data goes through is not properly monitored or understood.

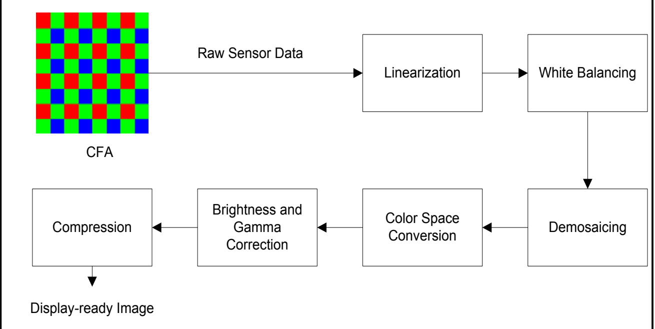

Figure 2. A General Camera Image Processing Pipeline

The proposed system model is shown in Figure 3 and is fundamentally a camera ISP integrated with a watermark embedding block that can be realized in software. The basic embedding and extraction algorithms are adapted from [13] with modifications in order to make it computationally efficient as explained in the next section. The watermark to be embedded is a pseudo-random set of binary bits which can be customized to represent a camera footprint for more practical uses.

4. PROPOSED IMAGE PIPELINING ALGORITHM

Since our method proposes a generic camera ISP integrated with watermarking algorithm, the image processing for our algorithm starts right after the digital set of 2D raw array is ready from the color filter in a camera. The algorithm contains the following steps in series:

Step 1: Acquiring the raw data

Acquire the raw data from the camera sensor. If $f_S(x, y)$ be the raw pixel value at (x, y) location of the image, then it can be represented as,

$$f_S(x,y) = I_s(x,y) \qquad (1)$$

where s represents the color domain and can be R, G or B and $I_s(x, y)$ represents the single channel intensity captured by the color pixel at each pixel location corresponding to the color channel represented by s.

Step 2: Neutralizing offset and scaling

Depending on how specific camera works, the raw image $f_S(x, y)$ that we can get from a camera might have some offset and random scaling applied to it. The camera stores that information and may be retrieved in order to convert neutralize the offset and scaling applied to the raw data. Let the neutralized image be $f_{S\_N}(x, y)$, then,

$$f_{S\_N}(x,y) = \frac{[f_S(x,y)-off]}{(sf-off)} \qquad (2)$$

where off represents the offset factor and sf represents the saturation factor that the camera applied to the raw image.

After this is done, the pixel values that might have been thrown off the limit (i.e. above theoretical limit and below the black level or 0) due to unwanted sensor noise needs to be clipped off, as they are not part of the original image data. It is important to note that since

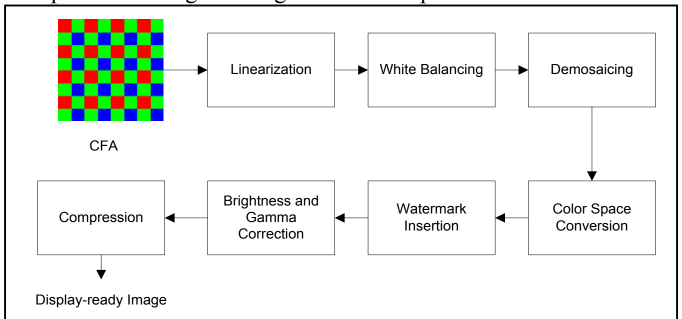

Figure 3. Proposed Camera Image Processing Pipeline with Watermarking

this is an irreversible process, watermark embedding is not done before this stage. The clipped data is represented as f_{S_C} and expressed as,

$$f_{S\_C} = \begin{cases} 0; & \text{if } f_{S\_N}(x,y) < 0 \\ 1; & \text{if } f_{S\_N}(x,y) > 1 \\ f_{S\_N}(x,y); & otherwise \end{cases} \qquad (3)$$

Step 3: White balancing

Since the raw data includes independent R, G and B values at each pixel that can represent any color depending upon the illumination, it needs to be converted to a data that represents actual color. In order to do this we choose a reference point that can be considered to represent a certain color and adjust the R, G and B values until that chosen color is represented. This is essentially what white balancing signifies. In our algorithm, we choose green as the reference and scale red and blue values with respect to this. When the camera shoots an image, it stores corresponding scaling values, often termed as white balancing multipliers in the EXIF information. We simply extract those multipliers and multiply each channel with the corresponding multiplier. So, if w_s be the multiplier matrix with three multiplier values for R, G and B channels, expressed as

$$w_s = (w_r, w_g, w_b) \qquad (4)$$

where w_r, w_g and w_b are the multipliers for R, G and B elements of f_{S_C}, then w_s has to be scaled with respect to wg before applying the multipliers for white balancing. Therefore, w_m is the new multiplier matrix that can be obtained as shown below,

$$w_m = \frac{w_s}{w_g} = [w_{mr}, w_{mg}, w_{mb}] \qquad (5)$$

where $w_{mg} = 1$ so that the reference G value in f_{S_C} remains unchanged. A white balanced 2D image, f_{WB} is now obtained from f_{S_C} by multiplying the values corresponding to R, G and B in f_{S_C} by w_{mr}, w_{mg} and w_{mb} respectively. This can be mathematically expressed as,

$$f_{WB}(x,y) = f_{S\_C}(x,y) * w_{mS} \qquad (6)$$

where *s* can be R, G or B. The equation (above) represents an element wise multiplication such that, for instance, if *s*=R, the pixel values corresponding to red channel in f_{S_C} will be scaled by red multiplier w_{mr}. Same applies for the green and blue channel values as well.

Step 4: Demosaicing

The 2D data with one color per pixel is now converted to a 3D color array by interpolation. There are different interpolation techniques in literature, but we choose to use gradient corrected bi-linear interpolation. Here, each missing color value at each pixel location is calculated from the adjacent pixels with that color. For example, in order to obtain missing G value at R position represented with a question mark sign (?) as seen in Figure 4, we first interpolate the adjacent G values as shown below,

$$\hat{G}_R(x,y) = \frac{\sum_{i,j} G(x+i,y+j)}{4} \tag{7}$$

where (i,j) = (-1,0), (1,0), (0,-1), (0,1) represents offsets to adjacent red pixel locations with reference to (x, y); and $\hat{G}_R(x, y)$ represents the interpolated green value at the red location (x, y). Blue value at the same location can be calculated similarly with a different set of (i, j) values to represent the adjacent blue pixel locations. Same rule applies to calculating missing color values at green and blue locations as well. Here, we only take $\hat{G}_R$ as an example for further discussions.

Since, $\hat{G}_R$ is not the true green value at (x, y), in order to make it as close as possible to the original value from the same location, the algorithm utilizes *R* at (x, y) to make adjustments to $\hat{G}_R$ by the process of gradient correction. If $G_{corr}(x, y)$ is the corrected green value at (x, y) then,

$$G_{corr}(x,y) = \hat{G}_R(x,y) + \alpha\, \Delta_R \tag{8}$$

and

$$\Delta_R = R(x,y) - R_{avg} \tag{9}$$

where R_{avg} is the average of adjacent red values with reference to (x, y) and α is a gain factor that can vary. We can now represent the 3D demosaiced color image array as $f_{color}(x, y, s)$ with *s* representing either R, G or B color domain.

Step 5: Converting to RGB color space

Now the demosaiced image's corordinates in the camera has to be converted to a correct RGB space that a computer can accept for viewing. In order to achieve this, we multiply each pixel of the image from step 4 with a 3x3 color space conversion matrix $C_{camtoRGB}$ as shown below,

$$C_{camtoRGB} = M1_{camtoxyz} * M2_{xyztoRGB} \tag{10}$$

where *M*1 can be obtained from the image's metadata in the camera and *M*2 is a standard value expressed as shown in [12] as,

$$M2^{-1}_{xyztoRGB} = \begin{bmatrix} 0.4124564 & 0.3575761 & 0.1804375 \\ 0.2126729 & 0.7151522 & 0.0721750 \\ 0.0193339 & 0.1191920 & 0.9503041 \end{bmatrix} \tag{11}$$

where $M2^{-1}_{\text{xyztoRGB}}$ is simply $M2_{\text{RGBtoxyz}}$, a RGB to XYZ matrix obtained from [12].

So, if [*Ri Gi Bi*] be a pixel of the array $f_{color}(x, y, s)$ from step 5, where *i* represents a particular pixel position in the array, then,

$$[R_i'\ G_i'\ B_i'] = [R_i\ G_i\ B_i] * C_{camtoRGB} \tag{12}$$

where [*Ri' Gi' Bi'*] represents new RGB values for the pixel in the same position *i*. This operation is repeated for all the elements of $f_{color}(x, y, s)$ to obtain $f_{camtoRGB}$ in a display ready RGB color space.

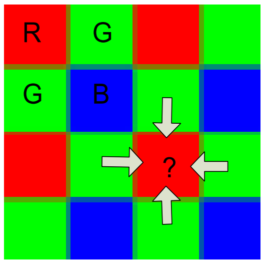

Figure 4. Interpolation for Demosaicing

Step 6: Embedding watermark in the cover media

The raw image is now ready to undergo watermark embedding. At this stage there are no more irreversible changes that the image is likely to suffer. A pseudo random watermark, in this case, can now be embedded into the image. Embedding is done in wavelet domain using 2 level Haar wavelet as the mother function, on a block by block basis. This means that the cover image is processed on a block by block basis, taking one at a time, instead of taking the complete image array for embedding.

This saves a great deal of computing time and memory requirement which is of special concern when it comes to mobile devices as opposed to computers. In addition, Smallest Univalue Segment Assimilating Nucleus (SUSAN) corner detection [14] is utilized to embed watermark bits at feature points and make it more robust. The choice of Discrete Wavelet Transform (DWT) for embedding in wavelet domain together with the corner detection algorithm makes the entire implementation localizable and positively aids our attempt of block level processing. This approach, in overall, avoids the unnecessary processing of further blocks once the bits to be embedded are over.

In order to achieve this, an image is fed to a corner detection algorithm on a block by block basis. Each block tested positive for corners is then passed through an embedding algorithm for watermarking until all the bits are embedded. For embedding, we adopt an embedding algorithm proposed by Nagham et al. in [13]. The image undergoes a DWT generating horizontal, $Hw(x,y)$ and vertical, $Vw(x,y)$ wavelet coefficient for each image pixel at location (x,y). Then, depending upon the threshold *Th* chosen to control message invisibility, an information bit *b* is hidden into location (x,y)as shown in equations (13) and (14).

According to the algorithm proposed in [13], if b=1 and $\Delta_1 = Hw(x,y) - Vw(x,y) < Th$,

$$\begin{cases} Hw'(x,y) = Hw(x,y) + \frac{Th-\Delta_1}{2} \\ Vw'(x,y) = Vw(x,y) - \frac{Th-\Delta_1}{2} \end{cases} \tag{13}$$

else if $\Delta_1 = Hw(x,y) - Vw(x,y) \geq Th$; no change.

If b=0 and $\Delta_2 = Hw(x,y) - Vw(x,y) < Th$

$$\begin{cases} Hw'(x,y) = Hw(x,y) - \frac{Th-\Delta_2}{2} \\ Vw'(x,y) = Vw(x,y) + \frac{Th-\Delta_2}{2} \end{cases} \tag{14}$$

else if $\Delta_2 = Hw(x,y) - Vw(x,y) \geq Th$; no change where Hw' and Vw' represent the modified coefficients.

The modified blocks are then treated with Inverse DWT (IDWT) function and combined with the remaining unmodified blocks to create a stego image.

Step 7: Gamma and brightness correction

Finally the watermarked cover image array that exists in the right color space is gamma and brightness corrected to produce a meaningful final watermarked image. The simplest way to brighten it up is to compute the mean luminance of image at this point and scale the image by a certain reasonable fraction of this mean. This is expressed as,

$$f_B(x,y,S) = f_{camtoRGB} * \frac{1}{k}[\mu(\sum I_{camtoRGB}(x,y))] \tag{15}$$

where f_B is the brightened image, μ refers to the mean, $I_{camtoRGB}(x, y)$ is the intensity or grayscale value at each (x, y) and $1/k$ represents a certain fraction of the mean intensity of $f_{camtoRGB}$.

Once this is done, f_B can be converted to nonlinear by applying a gamma correction power function for a more realistic looking image. The gamma correction factor γ is often approximated to be 1/2.2 and is applied to f_B to make it a nonlinear image by raising each element of f_B to the power of γ as,

$$f_{nl}(x,y,S) = f_b(x,y,S).^{\wedge\gamma} \tag{16}$$

where the dot (.) represents element wise operation.

Step 8: JPEG compression

As the display ready 3D color image from step 7 is a large array with more information than what is required, camera manufacturers tend to store this image following a JPEG compression. Hence, the final watermarked JPEG image that is available to the user will be a 3D array I which is relatively smaller than the size of f_{nl}, i.e,

$$I(x,y,S) = JPEG(f_{nl}(x,y,S) \tag{17}$$

where *JPEG* represents a JPEG compression function specific to the camera manufacturers.

In general, the pseudocode presented below best explains the process of inserting a watermark *WM* into RAW image f_{Raw} on a block by block basis of size *blocksize*, to give a final JPEG image compressed with Quality factor *Q*, with rest of the terminologies being the same as explained earlier in this section.

```
Algorithm: EMBEDDINGINRAW ($f_{Raw}$, $WM$, offset, $M1$, $M2$, $blocksize$, $\gamma$, $Q$)
$f_R \leftarrow f_{Raw}$   //Step 1
for each $f_R(x,y)$
do $f_N \leftarrow$ OFFSET ($f_R$, $offset$)   //Step 2
for each $f_N(x,y)$
do: if $f_N(x,y) < 0$
        do $f_C(x,y) \leftarrow 0$
    else if $f_C(x,y) > 1$
        do $f_C(x,y) \leftarrow 1$
    else
        do $f_C(x,y) \leftarrow f_N(x,y)$

$w_r \leftarrow w_s$[green_component]
$w_m \leftarrow w_s/w_r$
$f_{WB} \leftarrow$ WHITEBALANCE($f_C$, $w_m$)   //Step 3
$f_{COLOR} \leftarrow$ DEMOSAIC ($f_{WB}$)   //Step 4

$f_{RGB} \leftarrow$ COLORSPACECONVERSION($f_{COLOR}$, $M1$, $M2$)   //Step 5

while $i \le WM$(length)
do: $corner \leftarrow$ SUSAN($f_{RGB}$, $blocksize$)
    $f_{EMBED} \leftarrow$ EMBEDDING($f_{RGB}$, $blocksize$, $corner$)   //Step 6

$f_{BRIGHT} \leftarrow$ BRIGHTGAMMA($f_{EMBED}$, $\gamma$)   //Step 7
$f_{STEGO} \leftarrow$ JPEG ($f_{BRIGHT}$, $Q$)   //Step 8

return ($f_{STEGO}$)
```

The procedures OFFSET, WHITEBALANCE, DEMOSAIC, COLOR_SPACE_CONVERSION_CODE, STEGO_EMBEDDING, BRIGHT_GAMMA and JPEG in the algorithm each refer to the steps 2 through 8, as described earlier in this section, respectively. The procedure SUSAN refers to the generic SUSAN corner detection algorithm as explained in [14].

The watermark bits from the final image can be extracted by deploying an extraction algorithm in a manner similar to the embedding technique. The cover image is again processed on a block by block fashion with blocks being passed through the detection algorithm one at a time. The detection process ends once the given number of bits have been decoded. SUSAN corner detection is again used to find the embedding locations and the bits are extracted using the equation (18) explained below. For a message bit to be decoded from a particular pixel location that was embedded using equations (13) and (14), the following relation is used, as proposed in [13].

$$b = \begin{cases} 1 \text{ if } Hw(x,y) > Vw(x,y) \\ 0 \text{ if } Hw(x,y) < Vw(x,y) \end{cases} \tag{18}$$

where b is the bit extracted from a pixel location (x, y) with horizontal and vertical wavelet coefficients being $Hw(x, y)$ and $Vw(x, y)$ respectively.

5. SIMULATION AND DISCUSSION

The model proposed in Section III was implemented in MATLAB on an Intel(R) Core(TM) i5-3210M CPU @ 2.50 GHz processor. The raw images required for simulation purposes were taken from a Nikon D5100 camera. The choice of the camera was solely based on the fact that Nikon D5100 lets the user have access to raw sensor data. However, the proposed algorithm is generic and not specific to a particular camera model. The discussion below are the results of embedding two watermarks of bit length 100 and 200 respectively on the raw images.

It was important to see that the embedding process didn't destroy the image quality of the image. The image processing pipeline described in section IV was implemented and Peak Signal-to-Noise Ratio (PSNR) for the image was computed. The graphs in Figure 5 show the intermediate (raw) and final PSNR values for the images after embedding watermarks of 100 and 200 bit lengths, for a standard JPEG compression quality factor (Q) of 75. JPEG compression reduces a bit of image quality by itself. So the PSNR values shown in the graph only represent that due to the process of embedding.

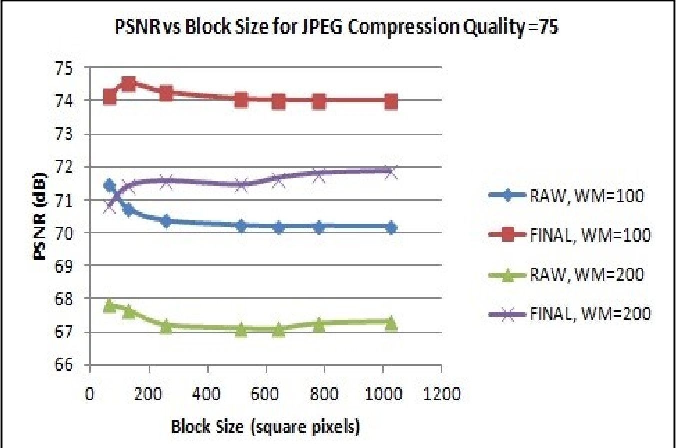

Figure 5. PSNR due to embedding Watermark (WM) of length 100 and 200 bits

As seen in Figure 5, for a particular payload size, PSNR is more or less constant for different block sizes. This means the choice of block size for embedding purposes doesn't necessarily affect the quality of the image. PSNR is only seen to change when the payload size changes. So, from the graph, when the number of watermark bits increases from 100 to 200, more pixels in the cover image deviate from the original values and hence the PSNR degrades. Hence, the maximum number of payloads that can be embedded on a cover medium depends on the desired PSNR.

For a particular payload, the graphs in Figure 5 show two PSNR curves labeled RAW and FINAL. The RAW PSNR corresponds to the PSNR computed immediately after embedding while the FINAL PSNR corresponds to the PSNR values computed after embedding is followed by JPEG compression. It is observed that the FINAL PSNR is slightly better than the RAW values. This means that the difference in cover and stego images decreases after compression. This is because the number of affected and total pixels (noise and signal pixels) in the image after undergoing compression is slightly less as compared to the number of such pixels in an uncompressed image, thereby giving a better signal to noise ratio after compression. The improvement in PSNR in this case is not to be misunderstood as improvement in image quality after compression but the differences between the cover and stego images are less distinct when both undergo compression.

The PSNR for different block sizes and different Q factor are tabulated in Table 1. The results show that the noises introduced due to embedding a certain number of watermark are more or less same for different Q factors and hence the choice of Q factor should only be dependent on the desired visual perception of the final image.

In order to ensure that the embedded bits can be successfully recovered following JPEG compression, Bit Error Rates (BER) for different payloads were computed after extraction. Figure 6 shows the BER for a payload of 100 bits against different image block sizes. The BER is then compared for different Q factors. As seen in Figure 6, BER is zero for higher Q factor like 90, meaning that all watermark bits are successfully recovered showing the robustness of the proposed method of embedding since JPEG compression at Q=90 didn't corrupt the watermark bits. As Q decreases, BER also starts to increase due to possible corruption of few image pixels by compression. However, the BER is still around 2 percent for a Q value of 75 which is not alarming and the method is still robust against compression.

As Q decreases beyond 75 towards 60 and 50, BER starts to increase further. It is highly unlikely that a Q value of 50 will be applied to an image that degrades the image's visual quality. But even if it were, BER is well below 10 percent as seen in the graphs in Figure 6 showing that the robustness of the algorithm against compression is indeed noteworthy.

Figure 6 further reveals that despite the overall BER not rising beyond an acceptable threshold, it is the lowest for a block size of 64 x 64 pixels. This is due to the fact that for a two level DCT (as per our implementation) of the block size gives a total number of 16 x 16 workable pixels for embedding. This is close to the number of watermarks to be embedded. Also, given that the extraction algorithm compares a pixel with neighboring pixels to decode a watermark bit, higher the chances of successfully decoding from small blocks. This is more like higher probability of decoding from a pool of less pixels than from a pool of more number of pixels. Hence, even if the BER values for other block sizes aren't that worse, it is logical to process smallest possible blocks of image since that saves time and memory when it comes to processing efficiency in addition to giving least BER as just discussed.

Comparing BER values for payloads of 100 and 200 bits as shown in Figures 6 and 7, it is observed that the trends are pretty similar except for more watermark bits resulting in higher BER values. This is quite obvious as more the number of embedded bits, more chances that they get corrupted. The compared result is shown in Figure 8 for a standard Q factor of 75. Despite the size of the watermark bits being doubled from 100 to 200, the BER only degrades to about 2.5 from a value of about 1.5 percent, both of the values being not being that detrimental to the robustness of the proposed method.

TABLE I. PSNR VALUES AS RESULT OF WATERMARKING

Q	Final PSNR (dB) for Block Sizes (square pixels)						
	64	***128***	***256***	***512***	***640***	***780***	***1024***
50	70.77	72.18	72.97	71.28	71.07	71.72	71.69
60	70.82	72.29	72.22	71.42	70.97	71.66	71.72
75	70.85	71.42	71.57	71.46	71.66	71.80	71.86
80	70.65	72.27	72.59	71.19	70.82	71.61	71.72
90	70.39	72.03	72.19	70.91	70.53	71.36	71.47

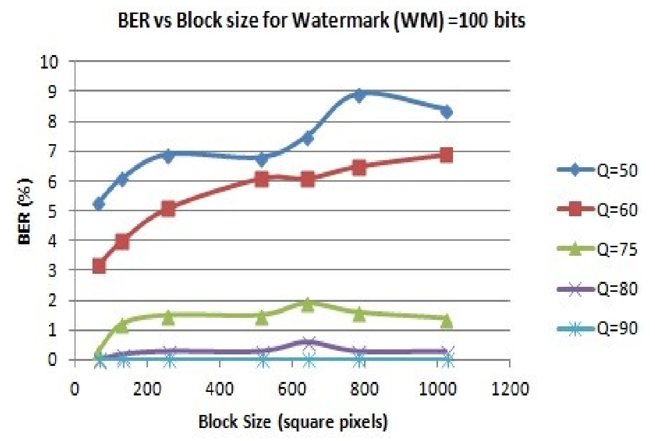

Figure 6. BER after the embedding followed by extraction of 100 watermark bits

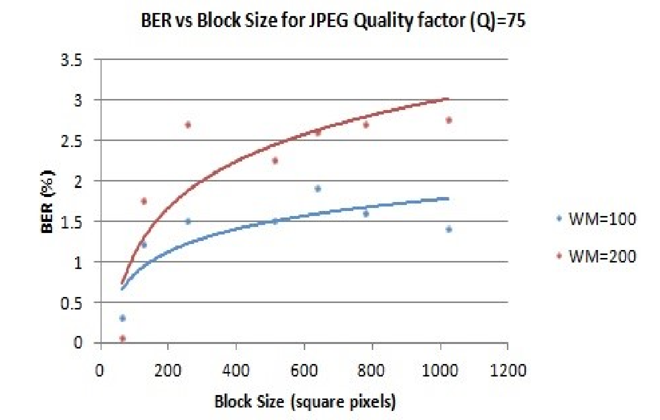

Figure 8. BER comparison for watermark lengths of 100 and 200 at a O factor of 75

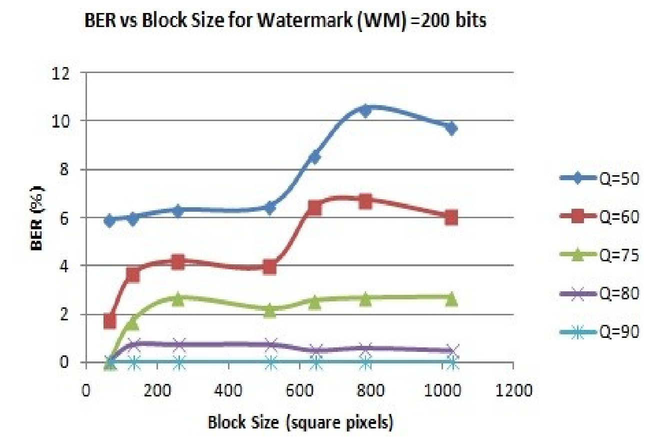

Figure 7. BER after the embedding followed by extraction of 200 watermark bits

6. CONCLUSION

In this paper, a process of watermarking raw image data is proposed. The proposed method helps ensure that all images are protected by watermarking them within the camera hardware before a display ready image is formed. This paper shows how a general image processing pipeline can be effectively combined with a watermarking algorithm to create a robust and novel watermarking process for raw image data. The proposed method ensures that algorithms are efficient in order to ease their implementation on mobile devices. Also, the simulation results with acceptable PSNR and BER values show the robustness of the proposed method against JPEG compression making it suitable to be used within a generic image processing pipeline to achieve a desired watermarked image.

7. REFERENCES

[1] J. Tsai et al., "A feature based digital image watermarking for copyright protection and content authentication," *IEEE International Conference on Image Processing (ICIP)*, San Antonio, TX, 16 Sept.-19 Oct., 2007, pp. 469-472.

[2] J. Zhao, Z. Liu, R. Langaniere, "Digital watermarking by using a feature based multiwavelet fusion approach," *Canandian Conference on Electrical and Computer Engineering*, 2-5 May, 2004, vol. 1, pp. 563-566.

[3] V.M. Potdar, S. Han, E. Chang, "A survey of digital image water- marking techniques," *Proceedings of the IEEE Third International Conference on Industrial Informatics (INDIN)*, Perth, Australia, 10–12 Aug., 2005, pp. 709–716.

[4] N.K. Abdulaziz, K.K. Pang, "Robust data hiding for images," *Proceedings of IEEE International Conference on Communication Technology*, WCC-ICCT, 21–25 Aug., 2000, vol. 1, pp. 380–383.

[5] S. Areepongsa et al., "Exploring on steganography for low bit rate wavelet based coder in image retrieval system," *Proceedings of IEEE TENCON*, Kuala Lumpur, Malaysia, 2000, vol. 3, pp. 250–255.

[6] P. Blythe and J. Fridrich, "Secure digital camera," In *Digital Forensic Research Workshop*, Baltimore, MD, August 2004, pp. 11-13.

[7] L. Tian and H.M. Tai. "Secure images captured by digital camera," In *2006 Digest of Technical Papers International Conference on Consumer Electronics*, January 2006, pp. 341-342.

[8] S. P. Mohanty, E. Kougianos and N. Ranganathan," VLSI architecture and chip for combined invisible robust and fragile watermarking," *Computers & Digital Techniques*, IET 1, No. 5, 2007,pp. 600-611.

[9] G. R. Nelson, G. A. Jullien and Y.P. Orly, "CMOS image sensor with watermarking capabilities," In *IEEE International Symposium on Circuits and Systems,* ISCAS, IEEE, 2005, pp. 5326-5329.

[10] R. Lukac and K. N. Plataniotis, "Camera image watermark transfer by demosaicking." In *48th International Symposium ELMAR-2006 focused on Multimedia Signal Processing and Communications*, IEEE, 2006, pp. 9-12.

[11] P. Meerwald and A. Uhl, "Watermarking of raw digital images in camera firmware: embedding and detection," In *Advances in Image and Video Technology*, Springer, Berlin, Heidelberg, 2009, pp. 340-348.

[12] Bruce Lindbloom. (2011). *RGB/XYZ Matrices*. [Online]. Available: http://www.brucelindbloom.com/index.html?Eqn_RGB_XYZ_Matrix.html

[13] N. Hamid et al., "Characteristic region based image steganography using speeded-up robust features technique*", 2012 International conference on Future Communication Networks (ICFCN)*, Baghdad, 2-5 Apr., 2012, pp.141-146.

[14] S. M. Smith and J. M. Brady, "SUSAN- a new approach to low level image processing," *International Journal of Computer Vision*, May 1997, vol. 23, Issue 1, pp. 45-78.

[15] R. Lukac and K. N. Plataniotis, “Color filter arrays: Design and performance analysis, ” *IEEE Transactions on Consumer Electronics*, 2005, vol.51(4), pp. 1260-1267.

[16] P. Singh and R. S. Chadha. "A survey of digital watermarking techniques, applications and attacks." *International Journal of Engineering and Innovative Technology (IJEIT)*,2013, vol. 2, no. 9 .

[17] C. Song, S. Sudirman, and M. Merabti. "Recent advances and classification of watermarking techniques in digital images." *Proceedings of the 10th of PostGraduate Network Symposium,* Oct. 2009, pp. 283-288.

[18] I. J. Cox, J. Kilian, F. T. Leighton and T. Shamoon, “Secure spread spectrum watermarking for multimedia”, *Transactions on Image Processing*, December, 1997, vol. 6, no. 12.

[19] X. Kang et al., “A DWT-DFT composite watermarking scheme robust to both affine transform and JPEG compression”, *IEEE Transactions on Circuits and Systems for Video Technology*, Aug. 2003, vol. 13, Issue 8, pp. 1051-8215.

[20] M. Noorkami and R. M. Mersereau, “Digital video watermarking in P-Frames with controlled video bit-rate increase”, *IEEE Transactions on Information Forensics and Security*, 2008.

[21] N, Ibrahim,Y. Weng and J. Jiang, “A new robust watermarking scheme for color image in spatial domain”, *Third International IEEE Conference on Signal-Image Technologies and Internet-Based System,* Sanghai, Dec. 2007, pp. 942-947.

[22] T. H. Ngan Le, K. H. Nguyen and H. Bac Le, “Literature survey on image watermarking tools, watermark attacks, and benchmarking tools”, *Second International Conferences on Advances in Multimedia*, 2010.

[23] W. Yan, H. Guo-Qiang, Z. Jian-Wei and Z. Bo, “Image authentication resilient to translation, rotation and scaling”, *International Conference on Machine Learning and Cybernetics*, 2006.

[24] X. Li, B. Gunturk and l. Zhang, “Image demosaicing: a systematic survey”, *SPIE Proceedings of Visual Communications and Image Processing,* Jan. 2008, vol. 6822.

[25] R. Liu and T. Tan, “A SVD based watermarking scheme for protecting rightful ownership”, *International IEEE Transactions on Multimedia*, Mar 2002, vol. 4, Issue 1, pp. 121-128.

[26] Rodriguez, Tony F., Trent J. Brundage, and Steven Michael Shovoly. "Methods involving maps, imagery, video and steganography." U.S. Patent No. 8,976,998. 10 Mar. 2015.

[27] Kester, Quist-Aphetsi, Mohammed Dolapo Asisat, and Ozoemena Willis Chibueze. "Security of Surveillance Wireless Camera Sensor Network Systems using Embedding Techniques for Authentication." International Journal of Computer Applications 123.12 (2015).

[28] Sanguinetti, Bruno, et al. "Perfectly secure steganography: hiding information in the quantum noise of a photograph." Physical Review A 93.1 (2016): 012336.

[29] Stanescu, Daniela, Valentin Stangaciu, and Mircea Stratulat. "Steganography on new generation of mobile phones with image and video processing abilities." *Computational Cybernetics and Technical Informatics (ICCC-CONTI), 2010 International Joint Conference on*. IEEE, 2010.