Winter 12-2014

A Study on Resource Efficient Digital Multimedia Security Measures in Mobile Devices

Prabhat Dahal
University of Nebraska-Lincoln, prabhatsmail@gmail.com

A STUDY ON RESOURCE EFFICIENT DIGITAL MULTIMEDIA SECURITY MEASURES IN MOBILE DEVICES

by

Prabhat Dahal

A THESIS

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfillment of Requirements

For the Degree of Master of Science

Major: Telecommunications Engineering

Under the Supervision of Professor Dongming Peng

Lincoln, Nebraska

December, 2014

A STUDY ON RESOURCE EFFICIENT DIGITAL MULTIMEDIA SECURITY FOR MOBILE SYSTEMS

Prabhat Dahal, M.S.

University of Nebraska, 2014

Advisor: Dongming Peng

Advanced image and video processing abilities in smart phones and digital cameras make them popular means to capture multimedia. In addition, the integration of internet into such devices users seek to capture and easily share multimedia right from their smartphone while most steganography techniques are computer based. Hence, it is of utmost importance that the multimedia be processed for steganography right within the devices for multimedia authentication.

In this thesis, we first implement steganography into mobile smart devices that can capture multimedia. For devices such as smart phones, we propose a method to hide payload bits within video frames. The solution takes relatively less time and memory to process as opposed to existing computer based solutions. This is a major achievement over traditional techniques that have longer running times leading to power inefficiencies. The idea proposed is to divide the video frames being processed into smaller blocks and perform embedding at block levels, thus localizing any processing that is to be performed.

Simulation results show that the solution proposed can perform about 60 percent faster and 40 percent BER improvement than conventional approach of video steganography.

This thesis takes the foregoing solution to a greater height by using the same algorithm for steganography within Image Sensor Pipeline in digital cameras. The objective behind this is to ensure all images generated from all forms of digital cameras are watermarked automatically. The solutions that exist now are largely dependent on extraction of camera component information. The proposed steganography technique is image centric and aims to resolve existing issues in areas such as image source identification, discrimination of synthetic images and basic image forgery.

After experiments, Peak Signal to Noise Values with a least value of 70 dB even for the worst compression quality (Q) factor of 50 shows how the perceptual quality of the image is preserved. Bit Error Rate of about 5 % for the same quality (Q=50) puts light on the robustness of the technique against JPEG compression.

Dedicated to my parents.

Acknowledgements

First of all, I would like to thank my advisor Dr. Dongming Peng. It has been an incredibly wonderful and life changing experience working with him. His continuous support and contributions have been immensely helpful throughout this program and research.

I would also like to thank Dr. Hamid Sharif for guiding me throughout the program to accomplish both as a student and an individual.

I appreciate and thank Dr. Yaoqing Yang for helping me pave my way into research by constantly mentoring me in my field of study.

I would also like to extend my thanks to Dr. Michael Hempel for being a part of my Thesis Examination Committee.

I am immensely grateful to my colleagues in our research group for their kind support and help.

This acknowledge would not be complete without thanking my parents and my sister who have been showering me with perennial blessings in all my pursuits. I am overwhelmed with the love they have always bestowed me with.

I express my gratitude to all my friends for constantly being by my side, during highs and lows of my life.

Contents

List of Figures

List of Tables

Chapter 1. Introduction

The term mobility is heard quite often these days. With breakthroughs in different areas of electronics and communication, people seek mobility in all possible ways. And this desire in people has motivated the creators of technology to come up with different forms of mobile devices. As advancements in technology continues to skyrocket, the use of such mobile devices continues to find different forms at the same pace. With smart devices coming into limelight, such devices no longer fit within the boundaries of traditional forms of communication. We have seen incredible changes in the way people stay connected these days, with the introduction of several features that enable us to communicate at the push of a button. Internet in smart phones has only contributed to intensify such communication. According to the CTIA- The Wireless Association's Semi Annual Wireless Survey [18], about 25% of the total internet uses these days are mobile-only users. This means, they surf internet only through their mobile devices. Looking into this fact, it shouldn't be a surprise if people rely solely on their mobile devices to connect with each other and share data.

It is in this context of information sharing that the advent of cameras in smart devices has amplified sharing voice, image, video or audio clips. People no longer use smart phones for a mere voice calls or texting. Multimedia sharing has been the new form of communication and with the creation of numerous applications that enable users to do so, bloom in multimedia acquisition and sharing is quite obvious. It is with such mobility and mass distribution of multimedia files that issues such as security, authenticity and ownership of such files come into concern. Moreover, in an era when such multimedia can be

extensively used as evidences, processing multimedia in mobile devices for security becomes all the more important.

One way to implement secured multimedia communication is digital steganography. Steganography is the science of communicating secret data in an appropriate multimedia carrier like image, video or audio files with the concealment of the very existence of the embedded data. Over the years, countless mechanisms have been developed to secure digital multimedia before they are distributed, with each new method being more robust than the previous ones. On the other hand, steganalysis has also developed in a similar fashion. Steganography and steganalysis shall continue to improve with many researchers working on this field tirelessly [36], [37]. However, majority of such efforts in steganography are concentrated on processing multimedia on computers. There are plethora of steganography and watermarking algorithms that are computer based. For this, the multimedia has to be transferred to computers after they are captured, processed for security and then redistributed. The relatively larger resources in computers (memory, processors, etc.) as compared to their small mobile counterparts rarely pose any limitations in successfully implementing such steganographic algorithms. However, for reasons cited earlier, multimedia sharing is at the fingertips of each person and transferring multimedia to computers before sharing is an overhead and undesirable. In addition, the requirement for multimedia to be transferred to computers before they can be shared foils an otherwise pleasant user experience that can come with instant capture and sharing. More importantly, such multimedia lack the very basic feature such as authentication. Such reasons call for mechanisms that can allow users to at least watermark multimedia within the smart devices

as the very basic security processing. Furthermore, automatic watermarking of every multimedia file coming out of the camera should be an added advantage.

It has not been a long time that steganography in mobile devices has garnered interest. Over the past several years, researches have tried to use mobile devices for steganography like watermarking or decoding hidden data present in printed images. Such mobile devices also vary from digital cameras to smart phones. No matter what, algorithms in mobile devices cannot be implemented in a manner similar to implementation in computers. Care has to be taken to make sure the algorithms don't end up using critical resources in mobile devices that is limited. The primary idea in this thesis is to tailor a steganography algorithm specifically for mobile devices. Here, we start with seeking to implement steganography within a smart mobile device and exploring further to integrate this algorithm within the multimedia acquisition phase so as to ensure that every multimedia coming out of the camera is secure and contains authenticity information by default without requiring a user to be involved. However, it all starts with finding a good robust technique, from a pool of various mechanisms that exist, that can first be believed to do well in a resource rich computer environment.

In general, as seen with the existing algorithms, the simplest way to implement digital steganography is to exploit the multimedia file format. A simple example of this, in case of a digital image, would be to insert secret information bits in the image headers, End of File (EOF) tags, Exchangeable image file format , also known as Exif, metadata etc. [1]. A more advanced approach might be to hide information within the core image data. An instance of this is steganography in spatial domain where the data encoding is performed within the Least Significant Bits (LSBs) in the spatial domain of the cover image data. Steganography

in LSB is just a central idea and several variations of this exist in literature. On the other hand efforts were made to detect the same. In fact, according to the authors in [2], even a small change in the LSB method, for instance flipping LSBs of one pixel in a Joint Photography Experts Group (JPEG) image, can be effectively detected. This called for improvements over the LSB methods and in fact, over the spatial domain techniques as a whole. This led to the more robust methods that implemented embedding within the Discrete Cosine Transform (DCT) and hence the advent of frequency domain based steganography.

The development of DCT based methods resulted in the steganography to cause less visual and statistical artifacts as compared to their LSB in spatial domain counterparts. Algorithms like F5 [17] became widely popular to implement steganography in DCT domain. But improvements over any technology are inevitable and only matter of time. Despite DCT methods being less prone to statistical attacks and also more robust than spatial domain methods, another form of frequency domain technique developed that exploited components of the wavelet transform. This Discrete Wavelet Transform (DWT) based methods have shown promising results when it comes to robustness of steganography. From a holistic point of view, embedding in frequency domain is undoubtedly more robust and secure as compared to the spatial domain techniques. As a result, DCT and DWT [3] based steganography are extensively used to process digital images and videos these days. They are widely popular in areas of image and video compressions. An example of robustness of the wavelet based method is presented in [4] by Abduaziz and Pang, where they use vector quantization and one stage discrete Haar wavelet transform and conclude that data modification using wavelet transform results in multimedia quality being preserved with a very minimal perceptual artifact.

With tireless efforts put into steganography by researchers, steganography has advanced to adaptive steganography which effectively exploits and utilizes various “features” of the cover into which data is to be hidden. As an instance, for a digital image chosen as a cover medium, these features could be edge regions, skin textures, regions of smoothness etc. depending upon the contents of the image. Since such features are considered to be important parts of the image and hence their alteration or removal from the media is undesirable, they can be exploited to hide data at pixel locations corresponding to the feature regions. Study in [5] presents an example of utilizing edge feature for embedding and shows that such method indeed produce highly desirable output media which are distortion free for all the embedding domains- Spatial, DCT or DWT. The only drawback such methods might have to face is that the size of bits that can be embedded which we often call payload, is limited since they can be embedded only in feature locations and not all throughout the image/video.

Considering the ideas and studies presented above, this research aims at utilizing feature regions based embedding in multimedia. It will be wise to embed payload bits in the frequency domain instead of spatial domain to make the embedding more robust and survive certain attacks. Watermarking or payload embedding is first implemented on a JPEG image and extended for video, with the idea that video is merely collection of different images. However, since there are numerous algorithms that achieve the same results, the algorithm used for this research is modified targeting it for resource constrained devices. Hence the primary purpose of this research is to make the proposed algorithm efficient relative to the same algorithm when implemented on a computer with no research constraints. Once the primary goal has been achieved which proves that the algorithm can be efficiently used, the

study aims to embark on further utilization of the algorithm. Despite the fact that being able to watermark a multimedia file generated from cameras in a smart device might help include authentication and copyright information without having to transport the media to computer, there is still a chance of original media being misused. Unwanted users can still have access to the unwatermarked media and use it with malicious intents. It is in the hands of the user whether or not to watermark the media. This is because there is a gap existing between the media acquisition and processing phase before it can be deemed fit for sharing.

The only way to avoid this would be to remove the gap between the image capture and watermarking or information embedding before redistribution. There have been only limited efforts in literature that actually try to utilize this gap and bring watermarking close to media acquisition stage. There are quite a few efforts made to completely coincide the process of watermarking with acquisition in order to obtain a real-time watermarking solution to obtain that is deployed within the camera hardware. Not every existing algorithm can be made into such real time watermarking solution. There exists profusion of watermarking techniques in literature [21-25], [38-44]. Each method has advantages and disadvantages that come along with the implementation. The major problem here is that such methods cannot be readily implemented within the camera hardware to achieve the results i.e. watermarking within the acquisition phase.

Image acquisition phase in itself is a combination of several other stages. There are specific stages that the sensor data (first set of digital information that a camera produces from a scene) has to go through in order to complete acquisition and generate a perceivable image [54]. The collection of these stages is often known as Image Sensor Pipeline (ISP) which is responsible for producing an image ready for human perception. The closing of the

gap that this research talked about earlier is nothing but accommodating a resource efficient watermarking algorithm within this pipeline. Each stage of the ISP modifies the input data starting from the sensor data and passes the output to the next stage. After a series of modifications, the final output media is created. In terms of image, this is often the JPEG image that we use for different purposes. Since inputs to each of the ISP stages are different from each other and are acted upon to undergo different changes, simple insertion of an existing watermarking algorithm within ISP makes no sense. Existing watermarking or embedding algorithms often assume that the input image is a JPEG or similar image and processes it. This when applied to intermediate image data within the ISP might lead to unwanted results in both- the original image and the hidden data. This is highly undesirable. Also, it is extremely crucial to understand what changes each ISP stage makes to the input data so as to carefully plan where watermarking might be the safest. We don't want the existing ISP processes to interfere with any watermarking algorithm that is added. One of the prime caution is to leave the basic ISP unmodified so as to be able to add the steganographic algorithm within any camera's existing ISP. Hence, here the basic ISP that is common to all digital cameras has be properly studied and the embedding algorithm that has been customized to fit the resource constraints is finally included within the camera ISP so as to produce a human perception ready JPEG image that has been automatically watermarked during the acquisition phase. This ensures that every image coming out of the camera hardware contain authentication information by default.

Chapter 2. Background and Literature Review

2.1. Overview

Steganography for smart devices hasn't been as mature as general steganography based on computers. Although the need for mobile steganography has been pointed out in the literature, it hasn't been fully explored. Handful of attempts have been to make solid contributions in this regard. Implementation of any algorithm that has primarily been developed for computer use within smaller devices such as smartphones isn't that straightforward. Despite the fact that the processing abilities of such devices have rocketed over a span of few years now, the physical size of the device still limit the availability of memory and power. And since there are numerous applications running at the same time, it is desirable that the steganographic algorithm, if and when added for such platforms, take minimal memory and process faster so as not to degrade the existing performance. Nevertheless, the basic principle for any steganography algorithm and the file format of the cover media for both mobile devices and computers are pretty much the same.

2.2. Related Works

2.2.1. Mobile Steganography

Despite the fact that efforts in mobile steganography are not as much as that in general steganography [63-70], it will be unfair to not notice the diversity of the efforts made. The study of mobile steganography varies from hardware implementation in digital cameras to pure software manipulation in smart phones. One of such implementations was devising steganography in a Very Large Scale Implementation (VLSI) processing unit of a digital camera [6]. The primary purpose in [6] is to assure intellectual property protection and this thesis is based on similar motivation. The authors in [6] embed visible watermark as a

secondary translucent image overlaid into the cover image. The watermark inserted can be recovered only with appropriate extraction techniques. In particular, the authors aim behind proposing a VLSI based architecture is easy integration into any existing digital camera framework. The authors consider this to be the first VLSI architecture for visible watermark implementation. In order to prove the point they are making, they design a prototype chip with 28469 gates using 0.35-µm technology. The chip has pixel-by-pixel and block-by-block watermark processing abilities. The major drawback in the proposed method can be considered to be the choice of spatial domain for watermark embedding. Spatial domain steganography is no longer considered robust. However, their use of spatial domain can be understood given the complexity of implementation on a chip. Overall, this paper can be considered to be a good attempt in the inclusion of steganography in digital cameras.

Another effort to implement algorithms using microcontrollers and Digital Signal Processor (DSP) chips to obtain secure communication over public telephone network is made in [8]. The authors in [8] call it Speech Information Hiding Telephone (SITH) which is a technique based on information hiding steganographic scheme. The embedded system design uses one fixed point DSP, three floating point DSPs and a single –chip microcontroller unit working in conjunction. The authors hide secret information on normal speech transferred over Public Switching Telephone Network (PSTN) without attracting eavesdroppers. It proved to work when testing with China PSTN but the very fact that this was only meant for speech signals limits its use for copyright protection and authentication of other digital multimedia. Also, the requirement of additional hardware is cumbersome as compared to software only implementations.

Alvarez in [1] implement a basic form of steganography using EXIF headers in images. The author specifically mentions the problem in the case of child pornography where the pictures need to be tested for authentication and see whether they have been altered or not. They simply point out the fact that altered pictures somehow change the EXIF information and hence authenticity can be proved by analyzing the EXIF headers. This implementation would be rather easy for mobile phones and digital cameras since they don't require additional processing. However, there are photo editors like Adobe Photoshop 6.0 and higher attempt to preserve EXIF header data by replicating the original data. This might prove to be a hindrance in utilizing EXIF header for authenticity testing. Also, for someone who is expert in digital image processing, mimicking the original header file shouldn't be a problem.

The authors in [7] take the process of making steganography fit for mobile devices a step ahead by implementing algorithms in embedded devices. Considering steganography in mobile phones to be equally important as classic computing, the authors try to show that steganography can be successfully implemented into the new generation of mobile phones that are known to have enhanced image and video processing abilities. The major focus in the paper is the implementation of steganography algorithms in three different processors- an ARM7 based microcontroller, a multi-core processor called ISSAC and a Personal Computer (PC) and to present the comparison. They specifically examine the execution times of existing algorithms in these three platforms and conclude that execution time is highly influenced by the size of the carrier image. With the idea that processors like ISSAC and ARM are used in mobile phones, they try to find which algorithm might be the best fit for a chosen processor among those three. The study whatsoever makes no attempt in

further polishing an algorithm that can essentially prove to be better in any mobile platform. Rather than making an algorithm fit for mobile environment, the authors try to figure out which gives the best performance in terms of execution time.

Further exploring digital steganography in mobile phones, K. Papapanagiotou et al. in [9] examine steganography in the context of Multimedia Messaging System (MMS). MMS enables a mobile phone user to communicate using multimedia objects such as images, video and audio in addition to normal texts. Since MMS is getting popular, the authors explore the possibility of hiding information, particularly in images. In a time when most security research in mobile environment involved cryptography, [9] actually presents some of the widely used algorithms and their application in MMS. S. Mohanpriya in [10] designs and implements steganography along with MMS in order to secure information over mobile phones. The paper uses relatively better domain- DCT instead of traditional spatial domain to do the data hiding. In addition, tiny encryption algorithm is utilized so as to further make the data more difficult to decrypt. The tiny encryption algorithm is a block cipher algorithm which the author claims to be simple and fast and hence the best for mobile applications. The embedding algorithm chosen is F5 [17]. The implementation is to ensure that the information passed from source to destination is safe and secure. By combining cryptography and steganography over MMS, the author seeks to achieve this purpose. However, the basic flaw observed in this study is that, despite the fact that this algorithm claims to be for MMS essentially in mobile phones, the author mentions no point as to what makes it suitable for mobile devices. The implementation looks no different from a normal classical digital multimedia steganography that is computer based other than

the mention of tiny encryption algorithm being fast and suitable for mobile phones. There is no logic that proves it to be specifically suitable for MMS.

Likewise the authors in [19] try to address the issue of photos in camera smartphones being used without the owner's consent. The obvious solution to this problem is adding visible and invisible watermark. However, this requires an extra process to be performed by user to the image they want to share. This is particularly cumbersome when there are a large number of images that need to be watermarked before sharing. Taking this in mind, the authors in [19] propose a copyright embedding system for Android platform where a pre-specified copyright information is watermarked into the images while the images are captured instead of adding an extra process to watermark them. The authors also claim to have an option to selectively save the original unwatermarked images as well. They tend to make their proposed method a highly desirable one as they claim that their method is specially tailored to make the watermarking process computationally efficient for mobile devices and that the watermark can be retrieved without the need for the original image. Furthermore, they say that the embedded watermark is robust against basic image processing operations and their process automatically watermarks, resizes and uploads images to the internet without the need for user intervention. They deploy Haar wavelet transform as the embedding domain in order to make the process efficient. The process looks good in general. However, the major issue with the solution provided by [19] is that it is more of a watermarking application that is Android based. It doesn't provide a generic solution for all smart phones, let alone digital cameras. This doesn't guarantee that an image coming out of a digital camera is watermarked with copyright information as this method is application based and not incorporated within firmware.

The examples mentioned above are general steganography techniques that the authors in the papers claim to be useful for mobile systems. They claim the proposed methods to be efficient. However, all the applications explained above fall short of presenting a performance analysis of proposed scheme for mobile systems with the PC based implementations. No factual information has been presented that justifies the claim that the proposed methods are fit for mobile platforms. They claim so merely based on the implementation of steganography in MMS or images that they say are taken from cameras in smart phones. Also, majority of the data hiding techniques are based upon traditional spatial based LSB techniques. Some of the methods use DCT or DWT implementations but are not tested for robustness.

To ensure that the watermarks, copyright information or any other hidden information are robust, they need to be resistant to different attacks such as JPEG compression and geometric distortion for instance. Mere inclusion of information within multimedia won't ensure this and if the watermark isn't resistant to such basic attacks, steganography serves no purpose. One way to make steganography robust is to make sure that the embedding is done in certain regions of the multimedia file that can survive such attacks. And this is where featured based steganography is the key. There are several multimedia processing methods in existence that modify media one way or the other. In feature based steganography, the data to be hidden in embedded into key feature regions of the multimedia that are likely to be preserved throughout all such methods. Furthermore, such feature locations can also be used as reference points for synchronizing embedding and extraction of information without the need for original media for recovery of hidden information.

2.2.2. Feature based Steganography

Talking about feature based steganography for robustness, first we need to be able to extract the feature of interest. There are many feature extraction algorithms discussed in literature. There are also many of such methods that are used in combination with image and video analysis. As explained above, steganographers can trickily use such algorithms to help them hide messages in feature locations- be it image, video or audio. J. Xu and L. Feng in [11] present a watermarking scheme for images that is feature based. Their method performs both embedding and blind extraction. Image normalization and scale-invariant feature transform methods are first used to extract the stable image feature points from the cover image. Scale Invariant Feature Transform (SIFT) detector is used as the feature transform method to extract local features. Watermarks are inserted into the DWT coefficients of the Local Feature Regions (LFR). Blind extraction technique is devised that is resistant against de-synchronization attacks. Then experiments are performed to test the invisibility and robustness of the proposed scheme. Attacks performed were PEG compression, salt and pepper noise, median filtering and geometrical attacks such as rotation, scaling etc. The basic underlying principle in this technique is the use of SIFT detector that plays the major role in making the scheme robust.

The authors in [12] make an attempt to achieve image authentication and protection at the same time and they deploy feature based steganography for that. Hessian-Affine feature detector is first used to extract feature regions of a digital image. In order to achieve copyright protection, a copyright watermark is embedded into the extracted characteristic regions. Since the authors seek to achieve image authentication as well, the remainder of the image or the non-characteristic regions that were unused for copyright watermarking

are utilized for image authentication. For this, block-wise fragile watermarking is adopted. Similar techniques are used to blindly extract the watermarks for both copyright and authentication. The robustness is proved against basic geometrical attacks. The major drawback of the proposed scheme is pointed out by the author themselves. The use of Hessian- Affine Transform for feature detection makes the process resource demanding and very complex. Hessian-Affine is an iterative method and increases the complexity of any process that utilizes it. Also, since the fragile watermark for authentication is embedded into non-characteristic regions, it is susceptible to de-synchronicity attack as the location for watermark detection could be affected. This could be improved by embedding into characteristic regions, however, which again calls for the feature detection and hence increases complexity.

J. Zhao et al. in [13], a feature based fusion approach for embedding watermark in a host image in multiwavelet domain is proposed. They seek to embed watermark information into salient features of the cover image. The paper utilizes phase congruency in extracting salient features like the step edges and lines for the purpose of embedding and extraction. This combination of feature region and steganography is further used in [14] by John N. Ellinas. He presents a robust watermarking algorithm using wavelet transform and edge detection. As is the case with using characteristic region in watermarking, the efficiency of the proposed technique in [14] depends upon the preservation of the significant feature regions. In order to achieve that, the author tried to embed the watermark with the maximum strength possible over the sub-band wavelet coefficients of the feature regions that are the edges in the images. The strength of embedding is dependent on the level of the sub-band. Sobel detector is used to detect edge regions. The coefficients corresponding

to edges in the wavelet domain are the high frequency regions where the distortions are less noticeable to human perception. The author utilizes this idea to embed into this region so that the modifications due to embedding are not noticeable. The proposed method is computer based and would be interesting to see its application in mobile platforms.

In [15], S. Kay and E. Izquierdo take the feature based steganography a step ahead by combining characteristics of both spatial and frequency domain to attain a higher level of robustness against different image processing techniques. The proposed scheme first estimates Just Noticeable Distortion (JND) in the image and watermark is embedded by adaptive spreading the watermark information in the frequency components. In order to extract watermark, the spatial distribution of pixels in original image is considered. The use of JND is to insert pseudo-random watermark so as not to make the modification exceed the distortion sensitivity of the pixel into which the watermark bit is embedded. Embedding in frequency domain helps make the method robust against compression. In order to extract, the salient feature points into which the watermark bits are embedded are detected using the concept of first order differential invariants. The scheme proposed is devised in order to make the watermarking robust and no attention is paid to making the technique efficient since that isn't the primary concern of the paper. It only bolsters the fact that embedding into feature regions makes steganography robust to basic geometric attacks and JPEG compression.

It's now pretty clear that there are limited efforts made to integrate steganography within mobile systems- be it smartphones or digital cameras. There are studies done to implement steganography in mobile platforms. However, the underlying techniques are very basic and not quite robust. They don't deploy feature based steganography that could have made their

technique robust. Even if they did, the overall algorithm would be quite complex and time consuming. Also, majority of the studies that present steganography methods that are proclaimed to be fit for mobile platforms do not actually present any experimental data that shows that their schemes are different from PC based methods and are mobile platform centric. It is in this scenario that a feature based steganographic method tailored for mobile platforms would be really beneficial. But this again could be prone to multimedia misuse as it could be up to the user to deploy the designed steganography technique for mobile systems. However, if the mobile based steganography is used within the image acquisition phase this could be avoided. This would also allow the technique to be used not only on smartphones that include cameras, but also within all digital cameras.

2.2.3. Steganography in Digital Camera Systems

There have been very few but praise worthy researches in implementing watermark into camera firmware. Paul Blythe and Jessica Fridrich in [26] propose a concept of secure digital camera. The underlying objective of the study is to address the issue of integrity of digital images when used as evidences in the court of law. They propose lossless data embedding into digital images to identify the camera, the time of image capture, the photographer and the integrity of the image. This first thing for this is to create the information to be watermarked which in this case is the combination of biometric data of the photographer with cryptographic hashes and other forensic information. They design a camera system, using software on a chip, which is capable of using the photographer's iris as biometric information. In order to obtain the biometric information or the iris image, the camera viewfinder had to be modified. The watermark embedded is invisible and removable. This is an exciting and laudable advancement in digital forensics. However,

this calls for a major hardware and software overhaul in existing camera models and might be difficult to achieve in smart phone cameras where the user doesn't use any viewfinder. Also, the embedding is done is DCT domain not utilizing feature based techniques for robustness. In addition, the embedding done isn't a part of the camera ISP as the proposed watermarking utilizes final JPEG image produced by the camera instead of the intermediate sensor data.

In [27], the authors try similar approach of watermarking images captured by digital cameras. The scheme employs both semi-fragile and robust watermarks. The watermark information are generated by combining the image's frequency components and the owner's biometric data. They propose using this for integrity detection as well as ownership protection. The paper however doesn't show how the proposed scheme is integrated into any digital camera. The study only claims the method to be suitable for watermarking during image capture from a digital camera. It appears this is only based on the usage of iris as biometric information to be embedded into the image and lacks the experimental results of actual integration into camera firmware or hardware.

Mohanty, Kougianos and Ranganathan in [28] actually try to make steganography implementations in hardware. Their primary objective in [8] is to be able to contribute in the development of high-performance, low power consuming, reliable and secure, real time watermarking systems within a chip. In order to prove their point, they present a Very Large Scale Integration (VLSI) chip capable of doing this. The watermarking can embed both invisible robust and fragile watermarks. In order to demonstrate the hardware implementation, they prototype two designs. The first is a Xilinx Field Programmable Gate Array (FPGA) and the second one is by building a custom integrated Circuit (IC). The

motivation behind designing a watermarking chip is to be able to use it within any JPEG encoder in any digital camera. However there are several shortcomings associated with the design solution that has been proposed for watermarking on a chip. First and foremost, the processing is done on a pixel-by-pixel basis which is really slow. The authors plan on doing a study for block-by-block based processing to speed up the system. Secondly, the implementation that has been proposed can only be performed for grayscale images and implementations for color images are under study. Despite the fact that this hardware implementation looks promising, the authors' description in [8] prelude the integration of the design within the camera ISP. The embedding performed is DCT when more secure and robust wavelet based techniques have evolved. Nevertheless, the work in [8] is really important in terms of analyzing steganography in hardware.

This trend of trying to implement digital steganography continues with the research presented by G. R. Nelson et al. in [29]. They address the issue of the lack of sensor level integration of digital watermarking schemes. The paper presents a Complementary Metal Oxide Semiconductor (CMOS) Active Pixel Sensor (APS) imager that has a built in image watermarking feature. In order to embed authenticity information, watermarks specific to each chip are generated. This study is indeed laudable in creating an environment where all images will be watermarked but since this requires extra circuitry and hardware design, this cannot be ready implemented into the existing camera ISP as a software extension.

In [30], the authors R. Lukac and K. N. Plataniotis introduce watermarking solution for single-sensor digital cameras. They propose embedding a visible watermark into the camera sensor data, for a single-sensor camera, and then transferring the watermark to the final output image using the demosaicing [24] algorithm. The watermark is first inserted

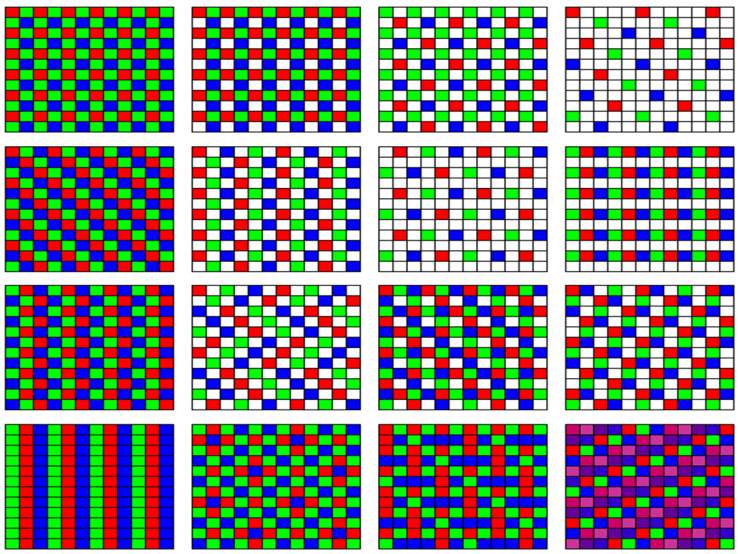

Figure 2.1. Different types of Color Filter Arrays

into a gray-scale image that is the image data coming out of the Color Filter Arrays, as shown in Figure 2.1. The watermark is then carried unto the final image using the process of demosaicing to generate a final color image. The final product includes a visible watermark. This is an interesting solution to protect digital property coming from single-sensor digital cameras. However the main problem here lies in the fact that the method is not generic to all camera models and since the final image contains visible watermark, this could be useless for applications that seek to use the images for purposes that do not require visible watermarking information. The major purpose of the solution proposed is to verify image authenticity by visual inspection of the visible watermark.

The authors in [31] put forward an approach of digital steganography for camera platforms that differs from the techniques and implementation described above in the sense that it is an entirely software based solution. The authors investigate a software only solution for real time watermarking of digital images coming from single sensor digital cameras. Even though it is unlike previous methods and doesn’t provide hardware solutions

for cameras, it looks more realistic in terms of integrating it with camera firmware. One reason for this is that they test their design on the CHDK firmware add-on for digital cameras from Canon. There could be different demosaicing techniques deployed within the camera ISP and hence the authors in [11] provide comparative results analyzing performance for different interpolation techniques. However, it uses simple spread spectrum additive embedding. So no matter how feasible it is integration wise, the method might not be robust as compared to advanced embedding schemes.

Looking into the past where several laudable efforts were made by researchers to integrate steganography within camera firmware, it is also important to note that camera manufacturers also tried to accomplish the same. Manufacturers like Epson and Kodak have manufactured digital cameras that had watermarking abilities in the past [26]. However, the watermarking abilities were not that straightforward to use. Epson required the users to purchase Image Authentication System (IAS) software to achieve watermarking. Kodak, on the other hand, has inbuilt features within the camera to insert visible watermark in digital images. But for some reason the Kodak cameras that had such features have been discontinued and are no longer available.

Hence, literature shows plethora of work being done in the field of digital steganography with the new ones being better and robust than previous ones. However, very few of such efforts are channeled towards mobile steganography. With increasing multimedia use in such devices, steganography in smart phones becomes inevitable. Also, it would be better to propose solutions that are feasible enough to integrate within existing technologies without demanding a lot of resources and hardware changes.

Chapter 3. Motivation and Problem Statement

3.1. Motivation

As discussed in the previous chapter, there have been exciting works that try to relate steganography with mobile devices. However, there are handful of researchers trying to actually cater steganography for mobile devices and digital cameras. Also, the majority of them require a major overhaul for actual implementation because of at least one of the following reasons:

(i) Requiring additional hardware for implementation making the existing devices useless to perform proposed solutions.

(ii) Being more focused on encryption (cryptography) rather than data hiding (steganography).

(iii) Using primitive embedding techniques like Least Significant Bit (LSB) embedding which are no more considered safe and secure.

(iv) More focused on embedding and not concerned to see the extracted message's integrity.

Also, digital image forensics is one of the critical field that utilizes image processing and steganography. With the proper use of steganography techniques, an image can act as an evidence to successfully solve cases in the court of law [53], [54]. There are plenty of image forensic techniques that extract information from digital images to trace the image's authenticity, integrity and forgery [50-54]. Component forensics seeks to extract information from the images to relate it to specific camera component and trace the image

source [47-49]. The solutions that exist in this field make use of the steganography techniques described in previous sections but suffer from several flaws such as [56-62]:

(i) Being camera brand centric and often unable to distinguish between different camera models.

(ii) Heavily relying on underlying digital camera technologies that can be the same for different vendors.

(iii) Based on image acquisition process that can again be the same for different digital cameras.

(iv) Need to be trained thus requiring a large number of authentic tamper free original images before actual use.

(v) Often ambiguous and unable to reliably detect time varying information.

The aforementioned issues make it important that a solution be proposed that can really be implemented without demanding additional resources. We also seek to address the issues inherent in existing authentication techniques by integrating a unique information in all images captured by digital cameras.

3.2. Problem Statement

There exists a gap between the powerful multimedia processing ability of hardware in smart phones and resource efficient steganography methods for such devices. The multimedia processing ability of such devices can be rightly utilized by implementing steganography methods that are tailored for such hardware. This can resolve the current requirement of multimedia to be transferred to Personal Computers (PCs) to be processed for steganography before they can be safely redistributed.

The critical gap between image acquisition and image steganography often tends to leave a lot of digital images vulnerable to tampering, thereby defeating the purpose of digital forensics. This can be rightly resolved by moving digital steganography as close as possible to image acquisition phase, such that each image coming out of any digital camera is already laden with a unique information that can prove beneficial for a variety of digital forensics application.

The research for this thesis is done in two parts. First, an attempt is made to come up with a working embedding algorithm that seems reasonably good for watermarking or information hiding within multimedia. After the algorithm is devised, we try to implement that within the camera ISP to make it close to the image acquisition phase.

Chapter 4. Proposed Video Steganography

4.1. Introduction

The primary idea behind the watermarking technique in this thesis is the embedding of watermark information or any other data within blocks, considering each block as an independent unit where information can be hidden, instead of processing an entire image. Here, we try to devise an embedding technique for video. The reason behind this is we try to do the watermarking within each video frame treating it as if it were an image. Hence, successful implementation of embedding within video would also enable us to use the same algorithm for images.

The video under consideration that has to be watermarked first goes through a frame retrieval process. We all know that video can be seen as a composite of multiple images called frames. Frame retrieval process simply splits up the video into its component frames. Now that the splitting has been done, each frame of the video is divided into numerous blocks of specific sizes. It is to be noted that the block size should be smaller than the frame size. The blocks that are thus formed are then read, one at a time, and fed to a characteristic recognition algorithm. There are different features that could be extracted, but here we chose corner detection. The output of the corner detection algorithm would now deliver blocks passed into it, along with the pixel locations where corners are present within the input blocks.

As per different instances mentioned in the literature review, the feature locations are extracted considering such locations of the block to be fit for data embedding. This is done to achieve robustness. After the embedding locations have been decided upon, the block

undergoes transformation to the domain in which data hiding is to be performed. Again, from literature frequency domain, and DCT or DWT in particular have been very popular and seem to give better results as compared to spatial domain. Hence, here the block with corners now undergoes DWT so that data could be embedded within the DWT coefficients of the pixel values at the corner locations. This is done for sequential blocks of the first frame as long as the data to be embedded isn't over. This is the exact process of how we would perform watermarking within an image.

After the first frame is done, Motion Vector (MV) [46] comes into play. Instead of going through successive frames doing the exact process that was done for the first frame, we try to make the process a little more efficient. MV is now deployed to find blocks in the successive frames that correspond to the watermarked blocks in the first frame. The MV maps each pixel from a reference frame to the next frame. For simplicity we choose this reference frame to be the first frame. With this reference frame and an array of MV, we find out corresponding blocks in all frames following the first frame and try to do embedding in those locations. Since characteristic region extraction is a computationally demanding process, an attempt to avoid this step after the first frame is made. In addition, since the scheme described above works on one block at a time instead of an entire frame that is relatively much larger than the block, the scheme is expected to be memory thrifty. The reason behind this is that instead of having to store a large frame to process, a smaller block can be saved into memory at a time, thus freeing the memory for other processes. As the memory available for smaller mobile phones and cameras are limited as compared to PC, this is a step toward making the algorithm fit for mobile devices.

4.2. Embedding Algorithm Development

The cover video is the output of any video camera in smart phones. For simplicity, we further refer to smartphones or mobile phones as devices unless otherwise stated. This cover video is initially stored within the permanent memory of the device. The entire process of information hiding or watermarking starts by reading the video to be watermarked. This video is then subject to frame retrieval and MV extraction process.

The frame retrieval is a relatively simple process and requires the Frame per Second (FPS) information of the video. Depending upon the FPS value for the video, a certain number of frame is produced for the video. The MV is a key element used in the proposed algorithm. MV, in general, is used in video compressing mechanisms. It is used to determine the position of a certain pixel or a block in a particular frame of the video based upon the position of corresponding block in the reference frame. We mainly focus on the embedding part here and for this research, a matrix with random values is chosen as the MV matrix. This MV matrix is considered to be an array of offsets for each pixels in the frames relative to the reference or the first frame. Actual computation of the MV is beyond the scope of this research and can be considered to be delved into in the future.

Let a cover video of duration t seconds, have a frame rate of f_R FPS (Frames per Second). This video is to be divided into N number of frames through the frame retrieval process. This satisfies the following,

$$F_i \in \{F_1, F_2, F_3, \dots, F_N\};\ i \in \{1,2, \dots, N\} \tag{4.1}$$

where $N=f_R*t$ and F_i represents a frame where i can take any value from 1 to N.

The next step is the corner extraction process. In order to achieve this, the first frame, F_1, is read and divided into fixed sized smaller blocks, B_j, as shown below,

$$B_j \in B = \{B_1, B_2, B_3, \dots, B_m\};\ j \in \{1,2,3, \dots, m\} \tag{4.2}$$

where m is the total number of blocks from the first frame, i.e. $F_{i=1,}$ and B_j, with j taking any value from 1 to m, represents a block. The size of each block is fixed, say, n_B pixels x n_B pixels.

After the first frame has been divided into smaller blocks, each block is now treated as a unit and read one at a time. All blocks that are read go through similar processes until the process of embedding is over. The first block is now fed into a corner detection algorithm. The corner detection algorithm used here is the Smallest Univalue Segment Assimilation Nucleus (SUSAN) algorithm [16].

The corner detection algorithm SUSAN uses a circular mask to be placed over a pixel to be tested for corner. This center pixel that is to be tested is now called the nucleus of the mask. Rest of the pixels that fall within the mask are now compared to the nucleus for corner detection. This is shown in Figure 4.1 and mathematically expressed as following,

$$corner(c, c_0) = \begin{Bmatrix} 1\ if\ |I(c) - I(c_0)| \leq bt \\ 0\ if\ |I(c) - I(c_0)| > bt \end{Bmatrix} \tag{4.3}$$

where c_0 is the position of the nucleus pixel within the two dimensional image block, c is the position of any other pixel point that lies within the mask, $I(c)$ is the intensity of any pixel at location c, bt is the brightness difference threshold used for comparison and *corner* is the final out of the comparison for corner.

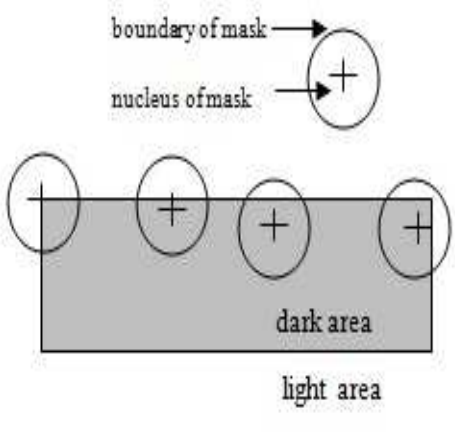

Figure 4.1. Different mask positions using SUSAN corner detection

Localizability is another factor to be looked into in order to attempt to make an image processing algorithm efficient. By localizability here we mean the ability of the algorithm to independently work on a pixel level. Also, the effect of the processing of an algorithm on a pixel should not be dependent on processing of other pixels. Such localizability of the corner detection algorithm lets us work with each block of an entire frame as an independent unit. And the block can be as small as it could be without affecting other blocks or the frame that the blocks are parts of. This, undoubtedly, is the primary benefit of feature detection algorithms and this has been exploited in this research. Since each blocks can be independently treated, the algorithm can be implemented on block level, thus, allowing us to limit memory usage.

After the blocks have been tested for corners and if any block is found to possess corners, it is split into Red (R), Green (G) and Blue (B) components since a color image is composed of RGB components. One of those components, here R, is chosen to undergo one level DWT. One level DWT decomposes the block into wavelet coefficients in different bands.

Figure 3 shows how a one level DWT of an image produces four sub blocks. The coefficients in the sub blocks are termed as Approximation (A), Horizontal (H), Vertical (V) and Diagonal (D) coefficients. The choice behind DWT in this study is that it allows us to continue the localization ability that the corner detection algorithm provided. Each coefficient in the sub band of the DWT of the image is a value corresponding to the pixel in the same location in the spatial domain and is not dependent on other pixels or coefficients. For the sake of comparison, DCT can be considered. In DCT, each coefficient is the result of processing of the entire block.

This localizing ability of the DWT allows this process to be combined with the previous SUSAN corner detection without messing up the ability to process at block level. So far, each unit of the frame can be processed on its own and the result in not contingent on results from other blocks. As a result, a unit much smaller than a frame can be processed at a time and this in turn immensely helps in reducing the amount of space required to save the working unit. This leads to a smaller memory requirement.

Now since the block has to undergo DWT, the next thing to be considered is what form of DWT to use. Given the popularity, this research adopts the Haar wavelet. The Haar wavelet's mother wavelet function $\psi(t)$ is expressed as,

$$\psi(t) = \begin{cases} 1 & 0 \leq t < 1/2, \\ -1 & \frac{1}{2} \leq t < 1, \\ 0 & \text{otherwise.} \end{cases} \tag{4.4}$$

where *t* is the unit of time.

After choosing the block with corners for embedding and making the block ready for information hiding in wavelet domain, information bits can be finally embedded. The algorithm chosen to hide information is adopted from [20] which is proposed by Nagham et al. As per [20], after the block undergoes DWT and frequency components have been generated, the horizontal and wavelet coefficients are selected in a raster way for embedding as shown in Figure 4.2. The binary message bits are then embedded within the chosen coefficients by altering the corresponding horizontal and vertical coefficients.

Say a bit, *b*, of the watermark is to be embedded. The block with corners undergoes DWT and this generates A, H, V and D wavelet coefficients. Let $Hw(x,y)$ and $Vw(x,y)$ be the representation of the horizontal and vertical wavelet coefficients, respectively. These H and V values correspond to the pixel location (x,y) where bit *b* is to be hidden. In order to achieve different strength of information invisibility, a threshold is chosen. This threshold can be represented by ϕ.

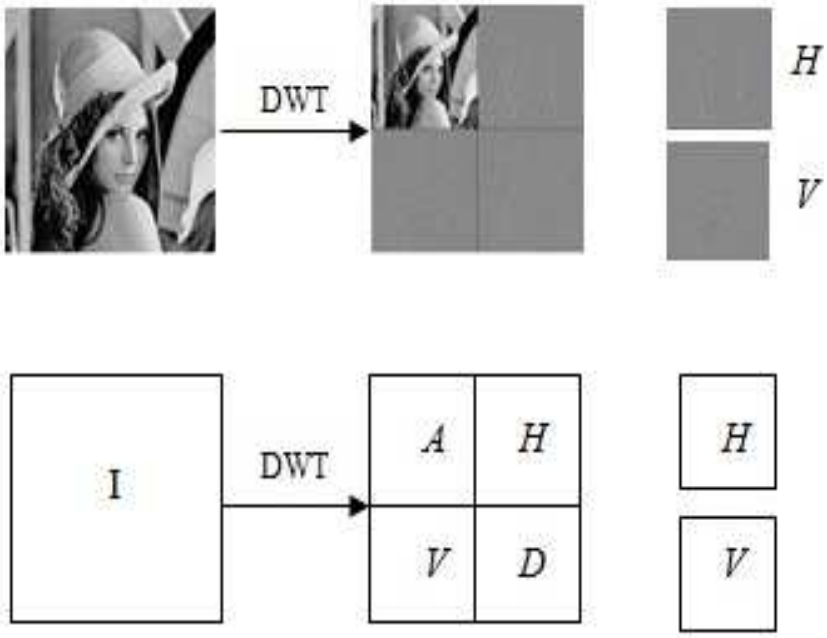

Figure 4.2. One level DWT of an image

Then, as per the technique proposed by the authors in [20], the embedding technique is mathematically expressed as shown below.

If b=0 and $\delta 1 = Vw(x,y) - Hw(x,y) < \phi$,

$$\begin{cases} Hw'(x,y) = Hw(x,y) - \dfrac{\phi - \delta 1}{2} \\ Vw'(x,y) = Vw(x,y) + \dfrac{\phi - \delta 1}{2} \end{cases} \tag{4.5}$$

else if $\delta 1 = Vw(x,y) - Hw(x,y) \geq \phi$; no change in coefficients.

If b=1 and $\delta 2 = Hw(x,y) - Vw(x,y) < \phi$

$$\begin{cases} Hw'(x,y) = Hw(x,y) + \dfrac{\phi - \delta 1}{2} \\ Vw'(x,y) = Vw(x,y) - \dfrac{\phi - \delta 1}{2} \end{cases} \tag{4.6}$$

else if $\delta 2 = Hw(x,y) - Vw(x,y) \geq \phi$; no change in coefficients.

where $\delta 1$ and $\delta 2$ are used to represent the difference of the horizontal and vertical coefficient values.

Since the number of bits embedded within any frame is always less than the number of coefficients for frame, the number of blocks used for embedding are less than total blocks generated from a frame. As explained earlier, total blocks from a frame is m. Let us assume that ε is the total number of blocks that undergo wavelet transform and are used for actual embedding such that $\varepsilon < m$. Let each modified block be represented by W_b, then this can be expressed as,

$$W_{bi} \in W = \{W_{b1}, W_{b2}, W_{b3}, \ldots, W_{b\varepsilon}\};\ C \subset B \tag{4.7}$$

where W_{bi} represents each modified block where i can be any value from 1 to ε.

As long as the message bits to be embedded aren't over, each block undergo same processing starting from corner detection to embedding. After the bits are over, the blocks need to be placed back into the original frame. This starts with inversing the wavelet transform of the blocks. Inverse Discrete Wavelet Transform (IDWT) is performed on the blocks to convert them back to the spatial domain. After each of the modified block is placed back on the original frame, processing on the first frame is over. If an image were to be watermarked instead of a video, it would be exact process that the first frame went through. Rest of the discussion that follow are strictly for video.

Basically all message bits are first embedded within the first frame. Now, in order to make the method robust, successive video frames are also utilized for embedding. Once the first frame is over, remaining frames i.e. $F_{i>1}$, are now read one at time. As explained earlier, MV is used to predict the embedding locations in these frames. Since the locations of corners in the first frame are already known, this information in conjunction with the MV matrix is used to determine the blocks with corners in the remaining frames. The primary benefit of doing this is that all these frames, other than the first frame, are spared from going through the rigorous feature detection process and saves a lot of computational load on the processor and also a lot of time. Now that the blocks with corners in all the other frames are determined, data bits are hidden within each block just like in the first frame. After modification these blocks are replaced with the original blocks in their corresponding frames. This hiding mechanism is presented graphically in the flowchart in Figure 4.3.

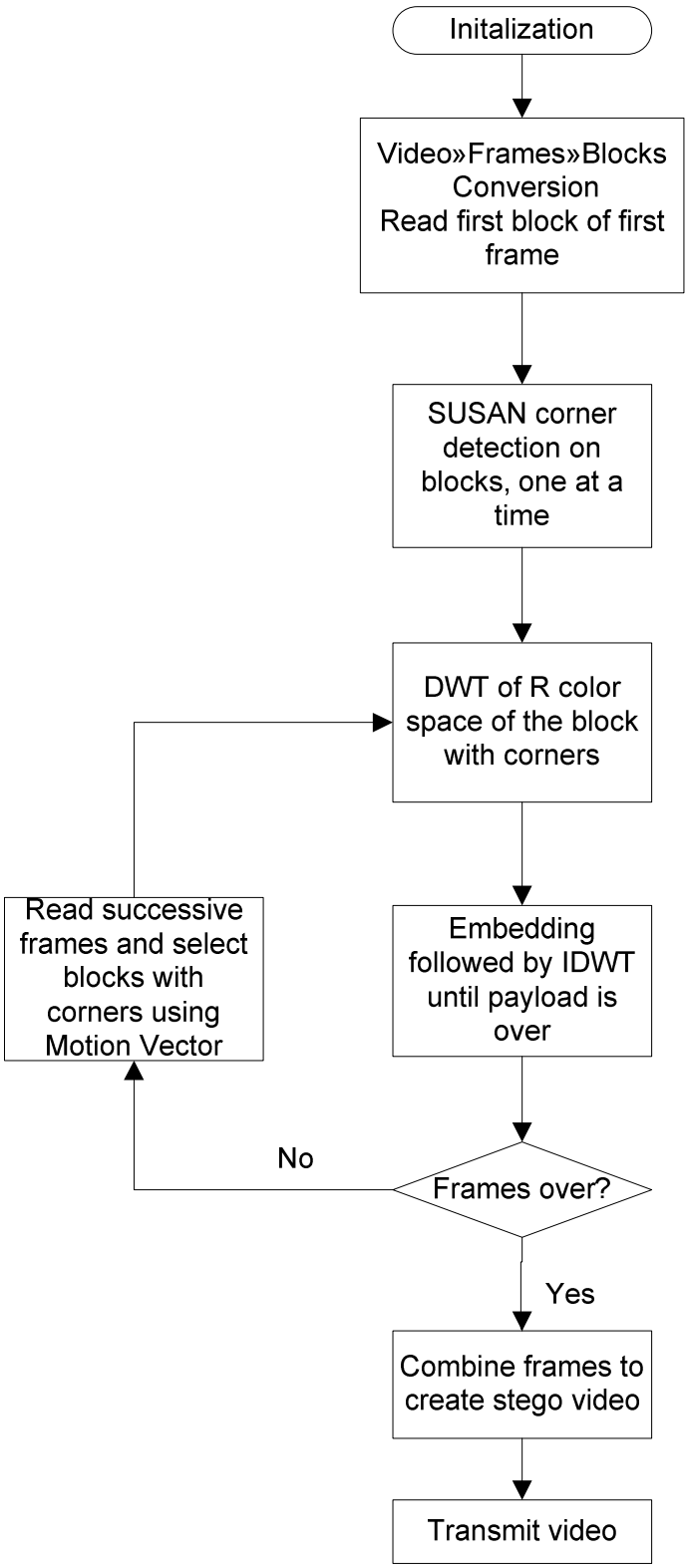

Figure 4.3. Flowchart for the embedding method

4.3. Time Efficiency Analysis

From the discussion above, one can infer that ε blocks from the first frame F_1 go through both the SUSAN corner detection and embedding processes. For all the other frames i.e. from F_2 to F_N frames, all the ε blocks need to go through only embedding algorithm as the computation intensive process.

In order to see the efficiency in computational time achieved by this block based method, let the time taken for the SUSAN corner detection method for a frame block of size n_B x n_B be T_c and the total time taken for embedding and DWT computation for the same block be T_e, then the following equation holds true,

$$T = \varepsilon(T_c + T_e) + (n_B - 1)\varepsilon T_e \tag{4.8}$$

where, T is the total time taken to embed messages in all the frames and includes the time each modified has to go through for corner detection, DWT and embedding.

It is worth noticing that MV is used to avoid feature detection on frames other than the first frame. Instead, if MV is not used, all m blocks of all N frames have to go through the intensive corner detection algorithm. Out of these m blocks on each frame, ε blocks finally go through the DWT and embedding processes. Let us assume that the total time taken for embedding under this scenario is T, and is mathematically explained as,

$$T' = NmT_c + N\varepsilon T_e \tag{4.9}$$

where NmT_c is the term for total time taken to test for corner detection only and $N\varepsilon T_e$ is the term representing the time taken for the DWT computation and embedding process .

In order to see the difference in time brought about by the block based method, subtracting (4.8) from (4.9), we get,

$$T' - T = (NmT_c + N\varepsilon T_{\mathrm{e}}) - (\varepsilon(T_{\mathrm{c}} + T_{\mathrm{e}}) + (N-1)\varepsilon T_{\mathrm{e}}) \quad (4.10)$$

Cancellation of terms leads to the reduced form as shown below,

$$T' - T = (Nm - \varepsilon)T_{\mathrm{c}} \quad (4.11)$$

We've already seen that $m > \varepsilon$. In addition, there is a minimum of one frame in each video. In other words, for a video with duration greater than zero seconds, the number of frames in the video is definitely $N > 0$ and which in turn leads to $T_{\mathrm{c}} > 0$. Based upon this, equation (4.11) can follow the following inequality,

$$T' - T > 0 \quad \text{or, } T' > T \quad (4.12)$$

which simply means that the time taken to embed information bits using the proposed method of processing smaller units or blocks instead of entire frames, *T*, is less than the time taken when working on an entire frame or image, *T*'.

Once embedding is complete on all frames, the modified frames in the spatial domain are recombined to form a video. This video has information hidden within and is called stego video. This is stored back in the permanent memory and is ready to be transferred. From the discussion above, it can be easily inferred that, since only one block at a time is read into Random Access Memory (RAM) to do the corner detection and embedding, this saves a considerable RAM for other applications unlike tradition method that required entire frame to be processed at a time.

4.4. Extraction And Performance Evaluation

Now that embedding has been efficiently done, extraction of the embedded bits has to be ensured as well. Only then it will be possible to tell how, if at all, robust the embedding is. The extraction of the hidden bits is also important in order to evaluate the performance of the proposed technique and compare with general approaches. After passing the watermarked stego video through an extraction algorithm, the extracted watermark bits can then be compared against the original watermark and Bit Error Rate (BER) is computed.

The extraction process is pictorially presented in Figure 4.5 in the later part of this section. However, since we are talking about implementation in mobile systems, the stego video is likely transmitted wirelessly to a particular recipient. This wireless transmission is prone to many channel noises. The study of different channel noises could itself be a massive research area. Just to make sure our extraction process is complete, we choose to deal with the most common Additive White Gaussian Noise (AWGN) for the sake of simplicity. To ensure our video survives the basic noise, AWGN is introduced to our channel which corrupts the stego video signal as it passes through the wireless medium.

AWGN is one of the simplest noises to understand amongst the plethora of noises that a wireless channel might have to bear. However, it is also one of the major problems in any Line Of Sight (LOS) wireless channel. By LOS channel we mean the transmitter and the receiver be within a line unobstructed by any hindrance. AWGN has a continuous spectrum which is uniform over the channel bandwidth. The amplitude of AWGN has a Gaussian probability density function (p.d.f.). As the signal of concern passes through the channel with AWGN, this amplitude gets added to the transmitted signal. Under this scenario, if x be the transmitted signal, n be the AWGN noise signal and y be the received signal, then,

$$y_i = x_i + n_i \tag{4.13}$$

where the index *i* represents a particular pixel of the video frame being transmitted. Hence, x_i refers to the i^{th} transmitted pixel value, y_i refers to the corresponding received pixel value at the receiver end and n_i represents a sample amplitude value of the AWGN function. It is assumed that a certain value that is a sample of the overall AWGN amplitude pdf is added to every transmitted pixel value at any point of time.

By standard definition from literature, AWGN is a random variable denoted by $N(\mu,\sigma^2)$ and expressed mathematically as,

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{4.14}$$

where $f(x)$ is the x^{th} sample of the amplitude value of the p.d.f. for all $x \in R$, μ is the mean and σ^2 is the variance of the distribution , as shown in Figure 4.4.

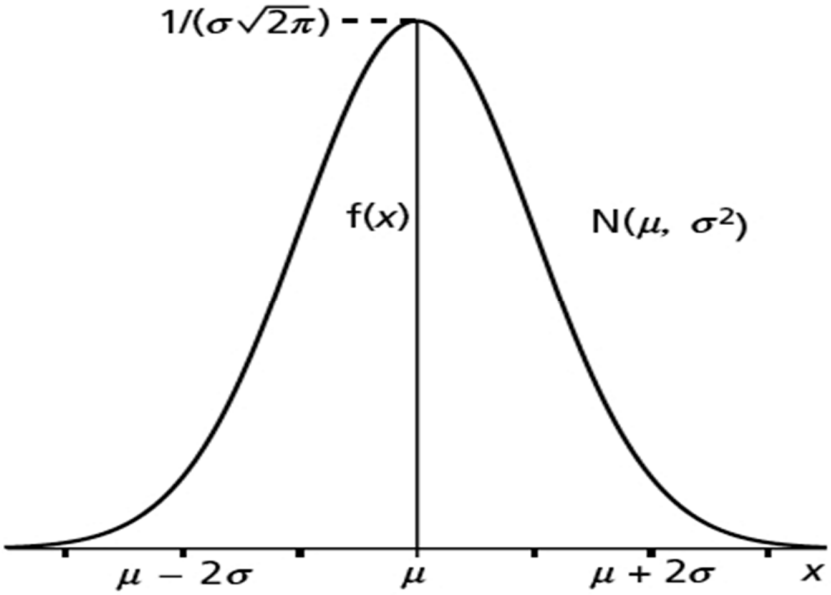

Figure 4.4. AWGN probability density function

As seen in Figure 4.4, AGWN is completely defined in terms of its mean and variance. This white noise when added to any signal, corrupts the signal. Here, our signal of interest is the transmitted video pixel value. Since the original video pixel has been modified to accommodate the watermark bits as well, AGWN noise when added to the transmitted pixels affects not only the original video pixels but also the watermark bits embedded within those pixels. This is because channel cannot make a distinction between the original pixel and the watermark bits. Hence, not only the original video is corrupted but also the watermark embedded. The video would have been affected by the noise even if it were transmitted without the watermark bits. But since the message bits are added with a purpose, it has be ensured that the message bits are recovered as much as possible.

Again, as per the proposed method in [20], for the message bits that were embedded using wavelet coefficient modification equations (4.5) and (4.6), extraction is done using equation (4.15). In order to be able to use (4.15), the modified pixels or blocks have to be first identified using the SUSAN corner detection mechanism, as shown in the flowchart in Figure 4.5. This is the exact same process that was done on the embedding part just before the message bits were hidden. The extraction equation proposed by Nelson et al. is presented as,

$$b = \begin{cases} 1 \text{ if } Hw(x,y) > Vw(x,y) \\ 0 \text{ if } Hw(x,y) < Vw(x,y) \end{cases} \quad (4.15)$$

where b is the bit decoded from a pixel location (x, y) that has horizontal and vertical wavelet coefficient values as $Hw(x, y)$ and $Vw(x, y)$ respectively. The decoding is a fairly simple comparison of the wavelet coefficients of the each pixel from where the embedded bits have to be extracted.

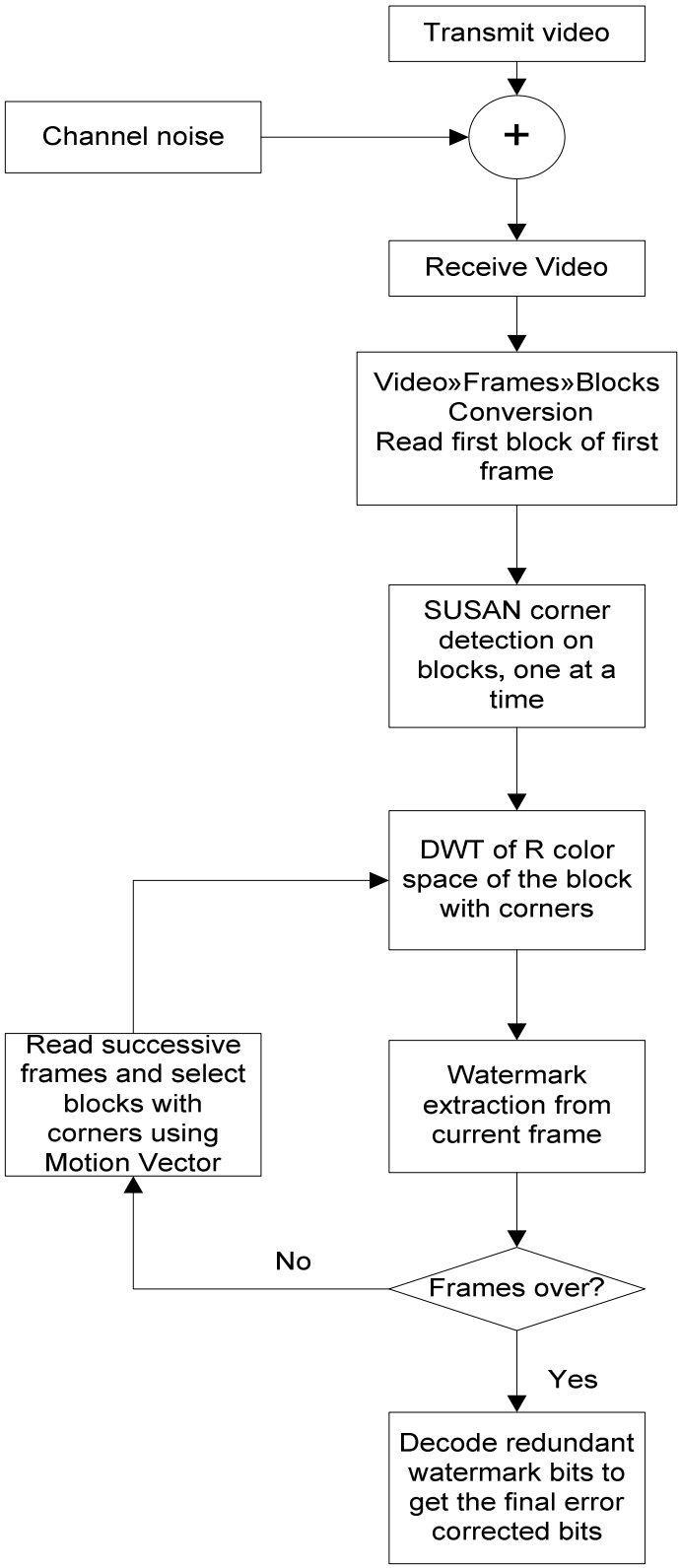

Figure 4.5. Flowchart for the extraction of watermark

Since these decoded bits might be a result of noise acting upon the original bits, they need to be compared against the original bits that were embedded to figure out if they are correct. BER is chosen as the metric to quantify the correctness of the decoded bits. BER is the defined as the ratio of erroneous bits to correct bits. It is computed by comparing the extracted bits with the embedded bits and is considered to be an important quality measure of the extracted watermark. This eventually leads to the quality measure of the proposed method.

It is not hard to sense that the extracted bits are not one hundred percent error free. There is every possibility that the erroneous channel modifies the pixel values, however small the modification be. This leads to a straightforward point that the erroneous channel increases the value of BER as compared to an error free channel. On the other hand, we seek to make the recovered bits as good as possible. However, it is not under the control of the user to make an erroneous channel devoid of channel noises. So, there has to be other mechanisms to address this issue and reduce the BER that has been increased by an unavoidable noisy channel. There are several techniques that aim to address this. Here, the research makes use of a widely popular Forward Error Correction (FEC) coding technique to ensure the decoded bits are as close to original as possible.

4.5. Error Minimization

Forward Error Correction (FEC) is also popularly known as channel coding. FEC encodes the information bits to be transmitted in a redundant fashion. This is in order to allow the recipient to correct error bits without requiring the information to be retransmitted. FEC really helps correct errors by avoiding the requirement for data retransmission but this comes at the cost of higher forward channel bandwidth requirement to fit added redundancy.

With redundancy, we are transmitting the multiple copies of the same information. Fortunately, for this case of extra bandwidth requirement can be eliminated in this case of video transmission. This FEC was specifically chosen for this particular application with that in mind. Given a limited channel bandwidth in any system, the video frames that are to be transmitted, no matter what, can be utilized to implement FEC without extra bandwidth. All the video frames have to be transmitted through the channel anyways, with or without any information bits. Also, all the message bits are accommodated within a single video frame. All the remaining frames can be utilized to encode the redundant message bits. This will result in multiple copies of the same information being transmitted within the original video size.

The receiver can now use this redundancy coding to extract information bits from all the video frames. It is probable that the same bit might undergo different changes because of varied noise value at different points of time in the channel. After all embedded bits have been extracted, the property of repetitive or redundant encoding can be used to correctly decode the bits. Let m_i be the i^{th} message bit of the hidden message, M, of total length n, i.e. $m_i \in M$ such that $i = 1$ to n. Since there are a total of N video frames and the same M is embedded into all the frames, there will be N copies of message M transmitted i.e. there will be N copies of m_i at bit position i within the message sequence. Let us introduce a new subscript j to represent a particular frame i.e. $j = 1$ to N. Now a i^{th} message bit in the j^{th} frame can be represented as m_{ij}. It is highly unlikely that all N copies of a bit m_i will be corrupted in the same way. Since, a message bit can be either 0 or 1, if m_{Rij} be an extracted bit at the receiver end, then, it is decoded as,

$$m_{Ri} = \begin{cases} 0 & \text{if } \Sigma(m_{ij} = 0) > N/2 \\ 1 & \text{otherwise} \end{cases} \quad \text{for } j = 1 \text{ to } N \tag{4.16}$$

where the subscript R in m_{Ri} signifies the received i^{th} bit and $\Sigma(m_{\mathrm{ij}}=0)$ represents the total number of i_{th} bits that are zero (0) throughout all frames (all values of j).

Describing in words, equation (4.16) means that the message bit at any position within the message sequence is decoded to hold the same value as the majority of the same bit in the frame sequence. For instance, let us take an example of an original bit that has a value of 1. This bit can be sent as a sequence of a hundred 1's assuming that there are a total of 100 video frames. As these hundred bits are being transmitted, each one of them can undergo random change within two possibilities i.e. a change from 1 to 0, or remain unchanged at 1. Upon reception, there will be 100 copies of the sent bit extracted from 100 frames, each with a value of either 0 or 1. If there are at least 51 bits that are 1 (as intended), the bit will be correctly decoded as a 1. If not, the bit will be wrongly interpreted as a 0. A relatively simpler scenario with all possible cases for 3 frames and one message bit is shown below in Table 4.1.

Table 4.1. Example of Decoding Redundant Bits

Extracted bit triplets (N=3)	000	001	010	011	100	101	110	111
Σ(m=0)	3	2	2	1	2	1	1	0
Decoded bit (m)	0	0	0	1	0	1	1	1

The explanation above shows how FEC can be utilized to correct bits corrupted by the noisy channel without compromising the embedding capacity. This is because the redundant bits do not demand extra space and fit themselves within the frames that will have to be transmitted anyways. An entire frame is available for embedding a message sequence and it can be replicated in the remaining frames. The results section shows how BER is improved with the use of FEC.

By proposing the method introduced above, we not only seek to improve BER and make the method efficient and robust but also see to it that no extra time is incurred in the process of making it robust. In order to see time efficiency of the overall method, we need to compute time complexity of the major time consuming internal methods. The primary idea here is to avoid reading an entire frame and perform repetitive processing of smaller blocks, one at a time. In addition, the computation heavy feature detection for robustness is also performed only on the first frame and MV is used as offset to find embedding locations in successive frames. It is difficult to make a generalized prediction of how many blocks will be processed for different video frames. An example case is analyzed below to see the time complexity of the major processes involved in embedding. Let us say, for instance, there is a video consisting of x_f video frames the proposed method is set to process n_b x n_b sized block at a time. The size of the entire frame is N_f x N_f. Assuming that corners are found on the very first n_b x n_b block and c is a factor such that $n_b = N_f/c$, Table 4.2 shows the time taken for major processes to complete. Considering SUSAN corner detection, DWT and IDWT to be the most time consuming processes amongst all, as per MATLAB profiler, these three functions have been listed on the table. Other processes were observed to not

make much of a difference in the overall time consumed by the entire method for different inputs.

Table 4.2. Time Complexity Comparison

Major functions	**Proposed Method**			**Conventional method**		
	1st frame	***2nd to x-th frames***	***Time complexity***	***1st frame***	***2nd to x-th frames***	***Time complexity***
SUSAN	✓	×	$O(n_b^2)$	✓	✓	$O(N_f^2)$
DWT	✓	✓	$O(n_b^2)$	✓	✓	$O(N_f^2)$
IDWT	✓	✓	$O(n_b^2)$	✓	✓	$O(N_f^2)$
Overall complexity	$O[(2x_f+1)\ n_b^2] =$ $O[(2x_f+1)\ N_f^2/c^2]$			$O(3x\ N_f^2)$		

To get a clearer picture of what Table 4.2 is trying to portray, let us consider a video with 294 frames (x_f=294) each of size 720 x 720 pixels (N_f=720). The time taken by the general approach which does not break down frames into blocks and processes an entire frame would be 457 x 10^6 unit time while that for our approach of block level processing with 16 x 16 sized blocks will only be 15 x 10^4 unit time, again assuming that corners are present in the first block. Even if corners were present in the very last block and the algorithm had to go through all blocks of the frame, the time complexity for our approach would still be about 67 times less than that of the general approach.

Chapter 5. Proposed Camera Steganography

5.1. Overview

The technique of processing a video proposed in the previous section is essentially to enable video steganography in smart phone systems. The proposed method takes a video input and efficiently embeds information bits into it. The objective in the previous section was to come up with a steganography method that is more efficient than its normal implementation in a personal computer system. However, it is still in the hands of the user whether or not to invoke the steganography algorithm. If the user chooses not to watermark the multimedia, no information bits are hidden and the media cannot be proved to be authentic. Considering the scenario, this section takes the algorithm proposed in the previous section to be implemented in a manner such that the user can no more control the operation of the watermarking system. In order to make that happen all media coming out of the camera should be laden with information bits by default. In this section, the previously proposed algorithm is tested for images produced by all digital camera systems.

5.2. System Model

The solution to the problem discussed above is to integrate steganography within the camera image acquisition system that already exists within all digital camera systems. The design proposed here is quintessentially a new camera ISP, different from the ones existing only in terms of an additional steganography process. In order to do that, it is important that we first understand how a basic camera ISP looks like. The image we obtain from a digital camera is the last set of digital data from the ISP. Often, the first set of digital data produced within the ISP is an array of numbers. These numbers are single channel intensity values

and the array is most commonly known as the raw image. This array represents the true information from any scene in the purest digital form possible and is therefore referred to as the raw image. The component in the camera responsible for generating this raw array is the camera's photo sensor. The photo sensor in combination with Color Filter Array (CFA) [15] of the camera produces the raw image. There are different types of CFA patterns. One specific pattern of CFA is shown in Figure 5.1. As evident from the figure, the CFA can only trap one color information (intensity) at a particular pixel location. The pixel corresponding to the green square in the CFA traps only the intensity for green and the same applies for red and blue. However the CFA pattern is designed as such that it allows the missing color information at any location to be interpolated with the help of color values in the neighboring locations. This process is called demosaicing [24] and there are different types of demosaicing algorithms catered for different CFA patterns. The most common CFA pattern in use is 'rggb'. Also, the camera manufacturers often don't reveal the technology they are using. Hence for simplicity, we consider the pattern 'rggb' to develop the remaining part of this thesis.

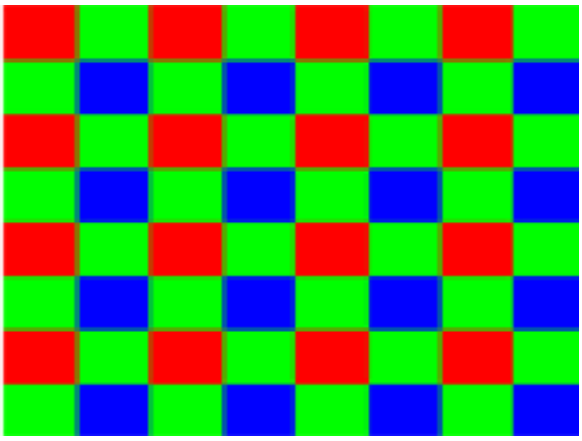

Figure 5.1. A Typical Color Filter Array with 'rggb' Pattern

To understand a CFA with ‘rggb’ pattern let us consider a 2 x 2 array of the Color Filter. In such an array there are one red, two green and one blue filter elements in a raster wise alignment. This pattern of a 2 x 2 array repeats itself until the camera resolution has been reached and produces the full sized image of size, say, m x n. But this m x n image coming from the CFA is essentially not the final color image. The result of demosaicing or interpolation is an m x n x 3 RGB (Red Blue Green) image that we expect out of any camera. This final image has one m x n array for each of the color components R, G and B.

Demosaicing is one the prime process within a camera ISP that helps transform a not-so-significant two dimensional intensity information to color image. But it is also to be noted that there are a series of other steps that the sensor data goes through one after the other to make the color image more meaningful and realistic. These steps can be different in different camera models and more often not made available for public. However, general basic steps are more or less the same and common to all manufacturers. Different manufacturers can have different ways to achieve the same result for each step.

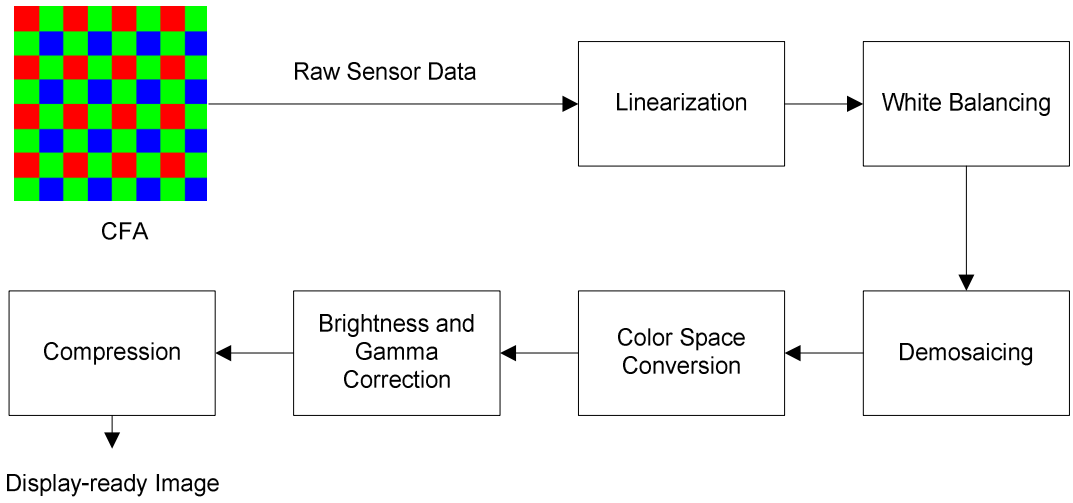

Figure 5.3. A General Camera Image Sensor Pipeline

A generic Image Sensor Pipeline is shown in Figure 5.2 and it presents the least amount of processes for converting a raw image to a final color image. In the order of their implementation, the major processes are linearization, white balancing, demosaicing, color space conversion and brightness and gamma control before the final color image is produced.

As explained earlier, here we are looking to accommodate an information embedding algorithm within the ISP. The proposed spot to do so is right after the image has been color space corrected, just before the image is about to go through the final brightness and gamma correction phase. This spot is chosen for a reason that the watermark needs to be safe as the image progresses through the ISP. It needs to be ensured that the hidden information, which is already something meaningful, doesn't undergo any irreversible changes which the raw data might have to go through as moving through the ISP. The final meaningful color image will be ready only after the final compression stage as depicted in Figure 5.2 and we don’t necessarily care how the intermediate image looks like. But the watermark might not be able to survive all stages of the ISP if included very early within the ISP. Hence, it is important to understand what changes the sensor data goes through before actually choosing an embedding location within the ISP.

5.3. Proposed Image Sensor Pipelining Algorithm

The proposed system model is shown in Figure 5.3 and is fundamentally a camera ISP present in all digital cameras but integrated with a watermark embedding block. The aim here is to propose a model that can be realized in software. Since the basic ISP already exists

in a camera with different image processing algorithms, mere addition of a watermarking block should be a problem. The basic embedding and extraction algorithms are adapted from [20] as explained earlier. The watermark to be embedded can be any binary sequence. Here the sequence chosen is a pseudo-random set of binary bits for demonstration purposes. Depending upon the application and requirement, this watermark can be something more meaningful as camera footprint or date and time.

Since we do not have access to stages within the intermediate sensor data within a digital camera, we need to start our processing with the set of raw image the camera manufactures provide the users with. And often this is a processed form of the raw data coming out of the CFA. In order to be able to watermark, the obtained raw image first needs to be taken back as close as possible to the original raw format. The entire process is explained below in a series of steps to be performed in a sequential order.

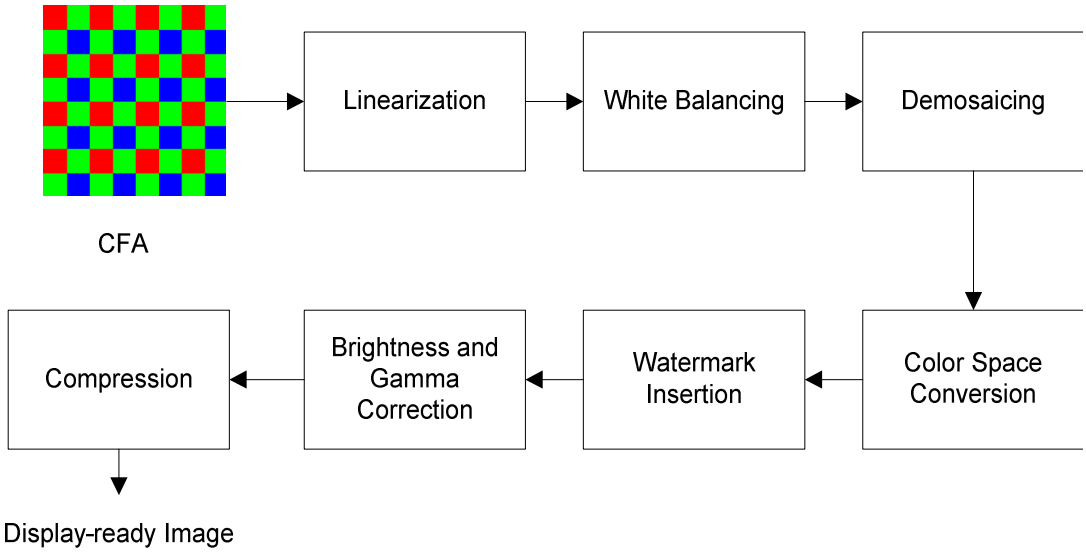

Figure 5.5. Proposed Camera Image Processing Pipeline with Watermarking

Step 1: Reading the raw image array

The first step is to read the raw image provided by the digital camera. The raw image is two dimensional and includes an intensity value for each pixel location. If $P_S(x, y)$ be the raw pixel value at (x, y) location of the image, then it can be represented as,

$$P_S(x, y) = RAW_s(x, y) \tag{5.1}$$

where the subscript *S* represents the color channel and can be either R, G or B color channel. $RAW_S(x, y)$ represents the raw image that is simply single channel intensity per pixel captured by the color filter of the CFA at each pixel location (x, y) corresponding to the color channel represented by *S*.

Step 2: Neutralizing offset and scaling

For cameras that provide the user with processed raw images, there might be a need to neutralize or reverse the processing. Such raw images might have some offset and random scaling applied to it. For raw data coming directly from the sensor, this step might be optional as the data is still pure. For scaled raw images, the camera also stores scaling and offset factors which can be retrieved and utilized to neutralize such processing. Let us denote the neutralized image by $P_{S\_N}(x, y)$, then,

$$P_{S\_N}(x, y) = \frac{[P_s(x, y) - \delta]}{(\theta - \delta)} \tag{5.2}$$

where $\boldsymbol{\delta}$ represents the offset factor and θ represents the saturation factor applied by the camera to the original raw data to store it.

Neutralizing the raw pixel values might throw the values off the limit (i.e. above theoretical limit and below the black level or 0) due to unwanted sensor noise. These values that are beyond limits have to be clipped off, as they are not part of the original raw data. They are simply the bi-products of this extra step incurred by the image data. Since this clipping is irreversible and is optional as explained earlier, we choose not to embed information bits before this stage to make the proposed algorithm generic. The clipped raw data can be represented as P_{S_C} and expressed as,

$$P_{S\_C}(x,y) = \begin{cases} 0; & \text{if } P_{S\_N}(x,y) < 0 \\ 1; & \text{if } P_{S\_N}(x,y) > 1 \\ P_{S\_N}(x,y); & otherwise \end{cases} \tag{5.3}$$

Step 3: White balancing

The raw data includes independent R, G or B values at each pixel. The pixel values are real illumination values from the scene filtered to capture only one color channel. The intensity value at each pixel is captured independent of illumination at other location. In this manner, the raw data is a large collection of independent intensity values and might appear meaningless. This array of different illumination values need to be converted to a data that represents true color. Since a true color is a balanced combination of R, G and B values, the independent values need to be scaled relative to each other. This can be achieved by choosing a reference pixel that can be considered to represent a certain true color and tweaking the R, G and B values until that chosen color is represented. This is the essence of white balancing. In order to make this happen, in this case, green is chosen as a reference while red and blue values are adjusted according to it. One does not have to worry about the scaling factors for white balancing. The cameras are made smart enough to store

corresponding scaling values at the time of capturing an illuminated scene. These scaling values are commonly represented as white balancing multipliers in the Exif information of the image and stored as a multiplier matrix. Here multipliers from Exif header are extracted and used for demonstration purposes. Let us represent the multiplier matrix by *WB* that has three multiplier values for R, G and B color channels respectively, expressed as,

$$WB = [WB_R \; WB_G \; WB_B] \tag{5.4}$$

where WB_R, WB_G, and WB_B are the multipliers for R, G and B elements of P_{S_C} respectively. Let us represent each element of *WB* by WB_S. Now, since green is chosen as reference, WB_S has to be scaled with respect to WB_G before applying the multipliers to all pixels for white balancing. If WB_M be the new multiplier matrix as a result of scaling *WB*, it is expressed as shown below,

$$WB_M = \frac{WB}{WB_G} = [WB_{MR} \; WB_{MG} \; WB_{MB}] \tag{5.5}$$

where $WB_{MG} = 1$ so that WB_{MG} when applied to the reference green pixel values in P_{S_C}, they remain unchanged. A white balanced 2D image, P_{WB} can now obtained by multiplying the values corresponding to R, G and B color channels in P_{S_C} by WB_{MR}, WB_{MG} and W_{MB} respectively. This can be mathematically expressed as matrix multiplication of the intensity array and white balancing matrix as shown below,

$$P_{WB}(x, y) = P_{S\_C}(x, y) * WB_{MS} \tag{5.6}$$

where index S can be R, G or B. The above equation represents an element wise matrix multiplication such that, for instance, if S=R, the pixel values corresponding to red color channel in P_{S_C} will be scaled by red multiplier WB_{MR}. The same is true when S=G and S=B.

Step 4: Demosaicing (Interpolation)

The two dimensional raw image data with one intensity value representing one color channel per pixel location is now converted to a three dimensional color matrix by interpolation. The three dimensional color matrix has values for one color per dimension. Three intensity values chosen from all three dimensions per pixel location make up the actual RGB color for that pixel. Several interpolation techniques exist in literature and the technique used by a camera manufacturer is unknown. Hence, a popular demosaicing technique called gradient corrected bi-linear interpolation is chosen here. The way this interpolation works is that for every pixel location, each missing color value is computed using neighboring pixel values for that color. For instance, we want to calculate the missing green (G) value at red (R) position. This missing value is represented with a question mark sign (**?**) as shown in Figure 5.4. The adjacent G values are interpolated in the missing location as follows,

$$\hat{G}_R(x,y) = \frac{\sum_{i,j} G(x+i, y+j)}{4} \tag{5.7}$$

where (i, j) represents offsets to adjacent red pixel locations from the target location (x, y), with i and j being offset values for x and y respectively, and $\hat{G}_R(x, y)$ represents the interpolated G value at the R location (x, y). Since, (i, j) needs to lead to the immediate adjacent pixels from (x, y), in this case of missing G at R location, offset (i, j) takes four different values as (-1, 0), (1, 0), (0, -1), (0, 1). B value at the same location can be

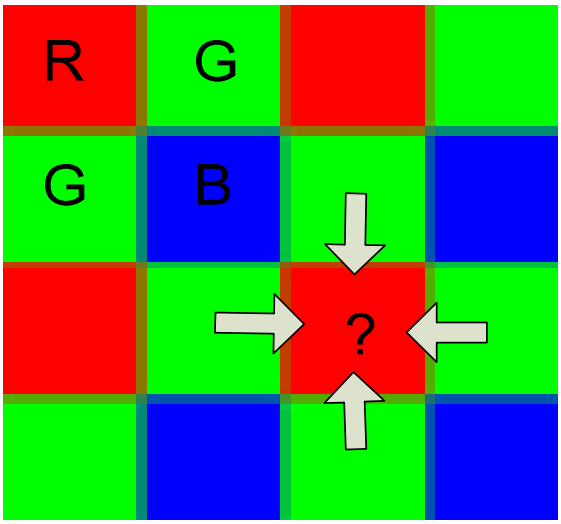

Figure 5.7. Interpolation for Demosaicing

interpolated similarly with a different set of (i, j) values to lead to adjacent blue pixels and not green. Other missing color values i.e. R and B at G locations and R and G at B locations are interpolated using similar concept. Here, $\hat{G}_R$ is taken as an example of interpolated values for further discussions.

Although $\hat{G}_R$ is the new interpolated green value at R, it should be noted, however, that this is not the true green value for that pixel (x, y). So the interpolated value needs to be adjusted in order to make it as close as possible to the original value as it would be if it were captured at the same location. Intensity of R at (x, y) is utilized.to make adjustments to $\hat{G}_R$ by the process of gradient correction. Let us use $G_{grad}(x, y)$ to represent the corrected G value at pixel location (x, y) then the gradient correction is done as,

$$G_{grad}(x, y) = \hat{G}_R(x, y) + \alpha \Delta_R \tag{5.8}$$

and

$$\Delta_R = R_R(x, y) - \bar{R} \tag{5.9}$$

where $\bar{R}$ is the average of adjacent red values with reference to the location (x, y) and α is a gain factor that can vary. Assuming $\boldsymbol{\Delta}_{\mathrm{R}}$ to be difference between the interpolated R and original R values, this difference is applied to correct interpolated G value as well. Let us now represent the three dimensional demosaiced color image matrix as $P_{DM}(x, y, S)$ with S representing one color channel from R, G or B color domain.

Step 5: Converting to RGB color space

The image is now in colored RGB format but not ready for display yet. The pixels in the camera have to be converted to co-ordinates fit for the computer's display. The demosaiced image's coordinates in the camera is now converted to a correct RGB space that a computer or other display screens can accept for viewing purposes. For this, the demosaiced image pixels are multiplied with a 3x3 color space conversion matrix represented as $CON_{Cam\text{-}RGB}$. This matrix is computed as,

$$CON_{Cam-RGB} = MAT1_{cam-xyz} * MAT2_{xyz-RGB} \tag{5.10}$$

where the matrix $MAT1_{\text{cam-xyz}}$ can be obtained from the image's metadata in the camera which converts pixels from camera co-ordinates to xyz co-ordinates and $MAT2_{\text{xyz-RGB}}$ is a standard value for converting from xyz co-ordinates to computer's RGB color space computed as shown in [12] as,

$$MAT2^{-1}_{xyz-RGB} = \begin{bmatrix} 0.4124564 & 0.3575761 & 0.1804375 \\ 0.2126729 & 0.7151522 & 0.0721750 \\ 0.0193339 & 0.1191920 & 0.9503041 \end{bmatrix} \tag{5.11}$$

where $MAT2^{-1}{}_{\text{xyz-RGB}}$ is simply $MAT2_{\text{RGB-xyz}}$, a RGB to XYZ matrix obtained from [12].

So, if [*Ri Gi Bi*] be a pixel representation of the demosaiced image matrix $P_{DM}(x, y, S)$ after interpolation with *i* indexing a particular pixel position within the matrix, then,

$$[R_i' \; G_i' \; B_i'] = [R_i \; G_i \; B_i] * CON_{Cam-RGB} \tag{5.12}$$

where [*Ri' Gi' Bi'*] represents new R,G,B values for the pixel in the same position *i* in the new RGB color space. Equation (5.12) is applied to all pixels of $P_{DM}(x, y, S)$ to generate a color space corrected P_{RGB} in the display ready RGB color space.

Step 6: Embedding watermark in the cover media

The raw image after demoasaicing and color space conversion is now considered ready for message bits embedding. From this stage onwards there are no more irreversible changes that the image is likely to suffer. Hence, it is possible to reverse the process until this stage to recover the embedded watermark. Any watermark sequence, a pseudo-random bit sequence 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. The embedding technique is the same as explained for video in the previous section. The only difference being this image can be considered to be a single frame of the video and no more frames exist. So no MV exists. The basic steps that happen are block wise division of the image, corner detection, DWT, embedding and IDWT. The result of this stage is a stego raw image in the RGB color space denoted by P_{stego}.

Step 7: Gamma and brightness correction

Finally the stego raw image that exists in the right color space for display is gamma and brightness corrected to produce a meaningful and eye-pleasing final image. This step is not mandatory but almost all cameras do this to produce an image that looks good. The simplest way to brighten up an image in practice is to scale the image by a reasonable fraction of the mean luminance of image at this point as shown in the following equation,

$$P_{Bright}(x, y, S) = P_{stego} * \frac{1}{k}\left[\mu(\sum I_{stego}(x, y))\right] \tag{5.13}$$

where P_{Bright} is the new brightened image, μ refers to the mean of parameter within the parenthesis, $I_{stego}(x, y)$ is the intensity or grayscale value at each (x, y) and $1/k$ represents a certain fraction of the computed mean intensity.

After brightness correction, the image can be made nonlinear by applying a gamma correction power function. This is done to make the image look more realistic. The difference can be seen by comparing images with and without gamma correction applied. The gamma correction factor γ is often approximated to be 1/2.2. Since this is a power function, it makes an image nonlinear by raising each element of the image to the power of γ as expressed below,

$$P_{NL}(x, y, S) = P_{Bright}(x, y, S)^{\wedge}_{.}{}^{\gamma} \tag{5.14}$$

where the dot (.) represents element wise operation of the image matrix.

Step 8: JPEG compression

After the seventh step, the image is absolutely ready to be displayed. This step (step 8) is done so as to ensure that the embedding survives compression since the large image is commonly JPEG compressed for storing or distribution. The image stored within the camera itself is JPEG compressed. Hence, the final stego JPEG image that is available to the user will be a three dimensional compressed image I_{Final}, i.e.

$$I_{Final}(x, y, S) = JPEG(P_{NL}(x, y, S), Q) \tag{5.15}$$

where *JPEG* represents a JPEG compression function specific to the camera manufacturers or applications with a quality factor *Q*.

The overall operation of embedding information bits in a raw image within a camera ISP is summarized by the pseudo code presented below. Here, *MS* represents the message sequence, *RAW* represents original raw image, *BS* represents the block size chosen for embedding purpose and *Q* represents the compression quality factor of the JPEG compression. Other terminologies follow the same description as explained in the algorithm earlier in this section.

5.4. Pseudocode

Algorithm: EMBEDDINGINRAW (*RAW, MS,* offset, *MAT*1, *MAT*2, *BS, γ, Q)*

$P_S \leftarrow RAW$ //Step 1

for each $P_S(x, y)$

do: $P_{S\_N} \leftarrow$ OFFSET ($Ps, \boldsymbol{\delta}, \theta$) //Step 2

for each $P_{S\_N}(x, y)$

do: **if** $P_{S\_N}(x, y) < 0$

do $P_{S\_C}(x, y) \leftarrow 0$

else if $P_{S\_N}(x, y) > 1$

do $P_{S\_C}(x, y) \leftarrow 1$

else

do $P_{S\_C}(x, y) \leftarrow P_{S\_N}(x, y)$

$WB\mathrm{r} \leftarrow WB_G$

$WB_M \leftarrow WB/WB\mathrm{r}$

$P_{WB} \leftarrow \mathrm{WHITEBALANCE}(P_{S\_C}, WB_M)$ //Step 3

$P_{DM} \leftarrow \mathrm{DEMOSAIC}(P_{WB})$ //Step 4

$P_{RGB} \leftarrow \mathrm{COLORSPACECONVERSION}(P_{DM}, MAT1, MAT2)$ //Step 5

while $i \leq MS$(length)

do: $corners \leftarrow \mathrm{SUSAN}(P_{RGB}, BS)$

$P_{stego} \leftarrow \mathrm{EMBEDDING}(P_{RGB}, BS, corners)$ //Step 6

$P_{Bright} \leftarrow \mathrm{BRIGHTGAMMA}(P_{stego}, \gamma)$ //Step 7

$I_{Final} \leftarrow \mathrm{JPEG}(P_{Bright}, Q)$ //Step 8

return (I_{Final})

The procedures OFFSET, WHITEBALANCE, DEMOSAIC, COLORSPACECONVERSION, EMBEDDING, BRIGHTGAMMA and JPEG in the pseudo code each refer to the steps 2 through 8, as explained in the algorithm earlier. The procedure SUSAN is the SUSAN corner detection algorithm as explained in [34].

The extraction of the embedded bits from the stego image is done in a fashion similar to that described in equation (4.15). BER and Pseudo Signal to Noise Ratio (PSNR) can be computed to see the correctness of the extracted bits and the change in image quality after embedding.

Chapter 6. Simulation and Numerical Results

The method proposed in this thesis have been tested separately for two implementations. First the idea proposed in initial part of Chapter-4 is tested where a video is watermarked, transmitted in the presence of noise and watermarks extracted. On the second implementation, information hiding on raw images are tested within the camera ISP. All experiments are performed in MATLAB on an Intel(R) Core(TM) 2Duo 2.00 GHz processor.

6.1. Implemenation- Resource Efficient Video Steganography

Information bits were hidden on several videos of '.avi' format. To present representative results in this chapter, information was hidden on a video (*butterfly.avi*, video of a butterfly flapping its wings) that consisted of 294 frames. Experiments were performed by embedding pseudorandom bits of different lengths (10, 60 and 250 bits) into each frame of the video. All experiments are repeated 100 times and the average values presented in the results that follow.

The video is first split into all the component frames. The first frame is again divided into smaller blocks. These blocks are then read one at a time on a raster wise order and passed on to the SUSAN corner detection function. The block that tests positive for corners undergo DWT to generate the wavelet coefficients. Payload bits are then embedded into the coefficients. This is repeated for each block as long as the payload bits are not over. After the first frame is done with the embedding process is repeated for all successive frames with

the only difference that blocks from these frames do not undergo corner detection. Blocks to be modified in these frames are detected using the motion vector.

In order to compare, the embedding is also done using the traditional approach of processing the entire frame instead of smaller blocks. This requires an entire frame to be read and placed into memory until embedding is over. The processing time (simulation running time from MATLAB) for complete embedding of payload bits for each approaches are then tabulates as shown in Table III. The different block sizes chosen for experimenting are 8 x 8, 16 x 16, 32 x 32 and 64 x 64.

6.1.1. Memory-Time Evaluation

Table 6.1. Algorithm Execution Times (Average)

Payload (bits)	**Processing Time (seconds)**				
	Proposed Method				**Conventional method**
	***8* x *8* blocks**	***16* x *16* blocks**	***32* x *32* blocks**	***64* x *64* blocks**	***Frame by Frame***
10	88	88	90	92	240
60	96	89	95	96	242
250	130	98	97	99	245

The data tabulated above in Table 6.1 are presented graphically in Figure 6.1. The curves show the time taken to complete the embedding process for different block sizes and for different payload lengths. The curve in the middle of Figure 6.1 present the times taken to embed a payload size of 60 bits (M=60) using different block sizes. As observed, the time taken to completely embed this message in the video frames is the lowest for a blocks sized

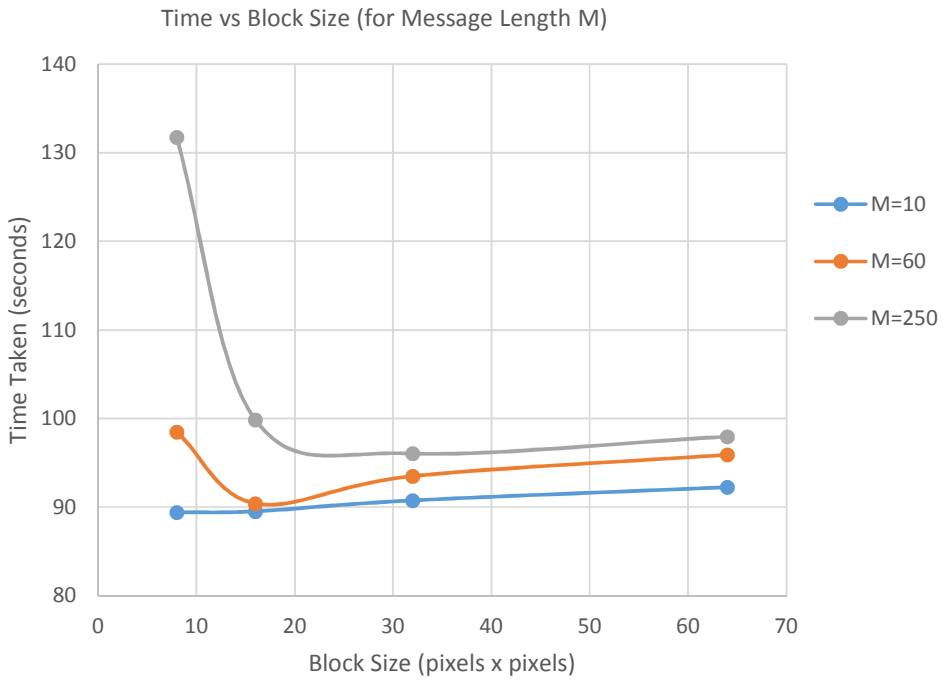

Figure 6.1. Time taken vs block size for different message lengths

16 x 16 are used. This is because the number of DWT coefficients (64) for this block size is the closest to the number of bits to be embedded (64) as compared to other block sizes. This leads to all message bits being embedded in the very first block and no additional blocks have to undergo processing saving a substantial amount of time. Other curves in Figure 6.1 are for different payload size but exhibit similar trend and can be explained by similar reasoning. For example, for the grey curve at the top in Figure 6.1 where 250 bits are embedded in the video frames, it can be observed that embedding is the fastest when choosing to process 32 x 32 block sizes with this time being about 97 seconds.

In addition to time, memory constraint is another issue that the proposed technique seeks to address. After repeated experiments, it is observed that maximum memory consumed at any time during processing videos using the proposed method is also less than that when

working on entire frame at a time. This inference is quite obvious and easy to understand. Smaller blocks need smaller memory to be stored than larger frames of which the blocks are part of. Since each major process like corner detection, DWT and embedding can now work on localized blocks, one at a time, smaller memory requirements lead to a memory thrifty solution.

Figure 6.2 presents curves where memory consumed by the embedding methods for different payload sizes are graphed. While the top curve shows results for a frame-by-frame method, other curves are for the technique proposed in this thesis. The curves are plotted in the same figure for easy comparison. The curves in Figure 6.2 are the memory requirements plotted against the input frame/block sizes. The memory requirement plotted are the

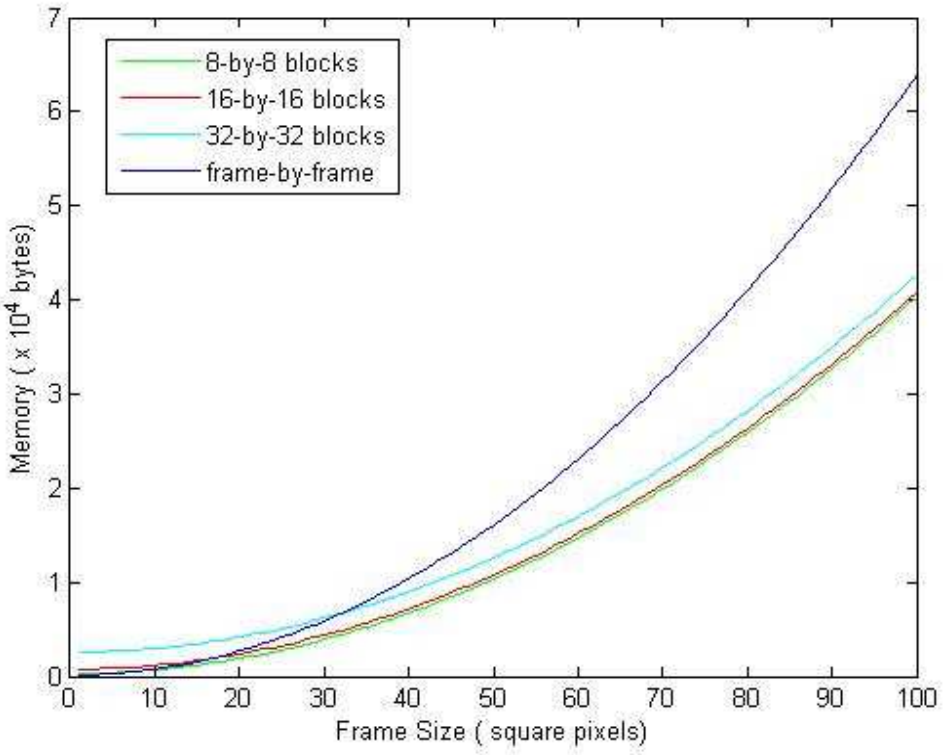

Figure 6.3. Memory requirement as a function of input frame size

memory consumed by different components of the algorithm, for instance - video frames, blocks and other numeric arrays used for computations. One can infer from the graphs that the memory required is proportional to block sizes. In other words, larger the blocks, bigger the memory needed to store those blocks. In this regard, it is not hard to understand that memory required is definitely the largest when entire frame is being processed at a time and this fact is clearly shown in Figure 6.2 where the blue curve for frame based method occupies the top-most position in the graph. Also, for any curve in the figure, increase in input size increases memory requirements.

As per Figure 6.2, any method is memory thrifty when the smallest input size is chosen. But this might not lead to the fastest time for a particular payload as shown in Figure 6.1.

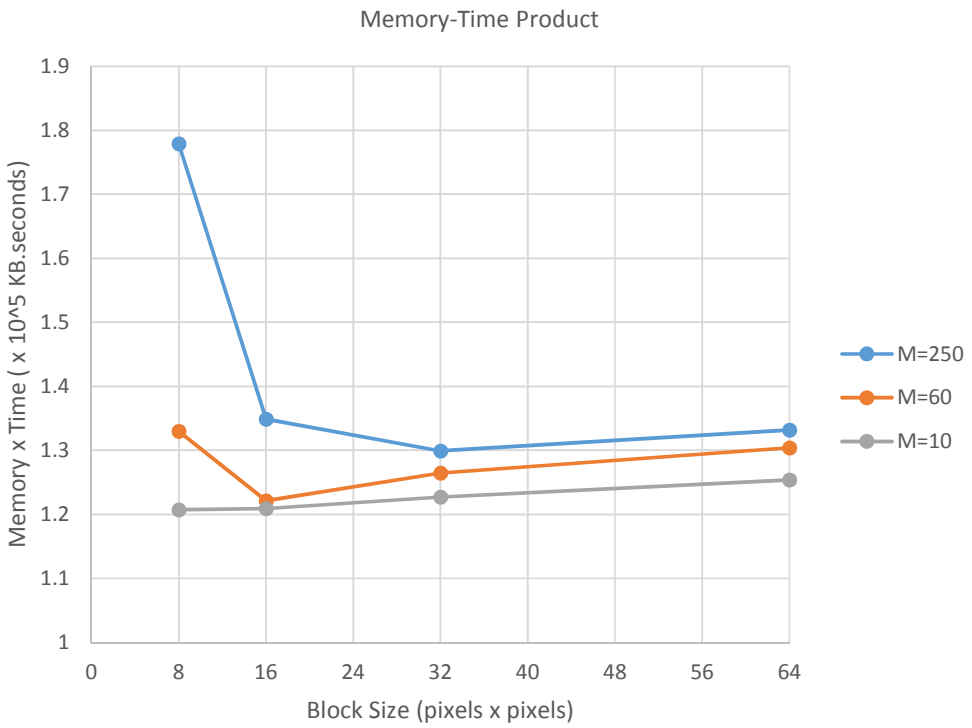

Figure 6.5. Memory-time product requirement as a function of block sizes with 480 pixels x 720 pixels input frame size.

Hence, there has to be a balance between time and memory requirements. It is in this regard that Figure 6.3 includes plots of memory-time product against different block sizes for the proposed method and can be seen as the combined result of Figures 6.1 and 6.2.

As this thesis talks about resource efficient techniques so that the method can be migrated from computer systems to mobile systems, both time and memory need to be taken into account when accessing the proposed method. The conclusions drawn from Figure 6.3 are not seemingly different from discussions just made. The graphs in Figure 6.3 simply reinforce the earlier inferences drawn from the Figures 6.1 and 6.2. Figure 6.3 expresses that for the proposed technique, the embedding method is the most efficient (both time and memory wise) for a block size that generates the DWT coefficients closer in number to the size of payload bits to be embedded within a frame. Also, on comparing the proposed block based technique with frame based technique, it can be said that the execution time of the optimal block based method for a particular payload size is sixty percent less than the frame based method. This is also more evident from figures in Table III.

6.1.2. Error Evaluation

Upon successful implementation of embedding, it is also equally important to be able to retrieve the hidden message from the video frames. Since the video is wirelessly transmitted, in order to test the survivability of the embedded message against noise, AGWN is added to each frame of the stego video. The sample AWGN noise used for simulation is generated using an inbuilt MATLAB function with mean and variance parameters as $\mu = 0$ and $\sigma^2 = 0.1$ respectively.

Table 6.2 shows the BER values for watermark extracted from the transmitted stego video. BER values are first computed by extracting hidden bits from one frame only. For

comparison, another set of BER values are computed using bits decoded using all the frames of the video. These processes are repeated for both noisy and noiseless channels. For noiseless channels, the BER values, as seen in the table, are zero as expected. This only shows the reliability/correctness of our extraction algorithm as all bits are successfully recovered when no noise is present and the bits are not modified. There are non-zero values when using 8 x 8 blocks because that block size isn't large enough for a DWT operation to embed the given payload size, thus yielding unwanted results.

Table 6.2. Algorithm Execution Times (Average)

Payload (bits)	**Block Size**	**Noiseless Channel**		**Noisy Channel**	
		BER for watermark decoded using		***BER for watermark extracted from***	
		1st frame only	***All frames (utilizing FEC)***	***1st frame only***	***All frames (utilizing FEC)***
60	8 x 8	51.667	49.6	53.333	45
	16 x 16	0	0	43.667	24
	32 x 32	0	0	41.667	36.667
	64 x 64	0	0	46.667	41.667
250	8 x 8	49.6	46.2	51.667	46.667
	16 x 16	0	0	47.8	29.9
	32 x 32	0	0	41.667	28
	64 x 64	0	0	43.667	30.4

The non-zero BER values for the channel infested with AWGN show that the message bits are corrupted due to channel noise. The BERs for message sequences extracted from the first frame are above 40. When the same sequences are decoded utilizing redundantly encoded information from multiple frames of the video, BER values drop by a significant amount. This shows that FEC, as explained earlier, contributes to minimize errors introduced by the additive noise in the channel. Tabulated BER values in Table IV make this point clearer. One row in Table 6.2 for each payload size is colored to signify the most suitable block size for that payload size. Furthermore, these values seem to be consistent with the results drawn from Table 6.1.

6.2. Implementation-Camera ISP Steganography for Images

The second set of experiments was performed to test raw image watermarking within camera ISP. This was implemented in MATLAB on an Intel(R) Core(TM) i5-3210M CPU @ 2.50 GHz processor. Since one doesn't have access to raw images from within any camera ISP, the availability of raw images for this implementation was ensured with the help of a Nikon D5100 DSLR camera. This camera model makes the images captured available to the user in both the raw format and the common compressed JPEG format. The raw images however are not in the purest form and some of the processing applied to them need to be reversed. Additional information for this is derived from the raw images' metadata. The choice of this particular camera was entirely driven by the fact that Nikon D5100 lets the user have access to raw sensor data. However, the technique proposed is generic and can be applied to all digital camera ISPs. The analyses presented below are the results of embedding two watermarks length 100 and 200 respectively on the cover raw images.

6.2.1. PSNR Evaluation

Since embedding is a part of image authentication, it is important to see that there is no deterioration of the perceptual image quality as a side effect of the embedding process. Peak Signal-to-Noise Ratio (PSNR) is the metric chosen to see the effect in image quality before and after embedding. Figure 6.4 shows plots of PSNR values tested after embedding message sequences of lengths 100 and 200 bits for the proposed technique. PSNR values are computed for two types of output images. The PSNR labelled RAW in Figure 6.4 represents the PSNR for stego images just after the embedding stage. These output images are taken immediately after the embedding stage and compared to the images at the same stage without embedding done on them. These images have not been JPEG compressed. The PSNR values labelled FINAL are computed from the stego images generated after

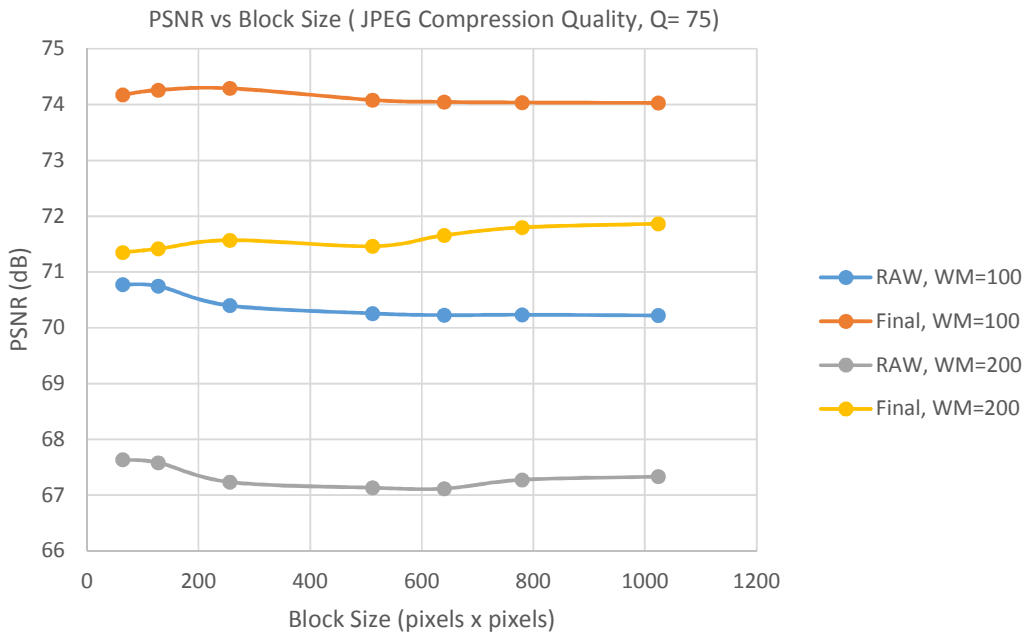

Figure 6.7. PSNR due to embedding Watermark (WM) of length 100 and 200 bits

JPEG compression. The quality factor (Q) chosen for this compression is a standard value of 75.

Looking at the PSNR trend in Figure 6.4 one can infer that, for any payload size, PSNR is more or less constant for different block sizes. In other words, quality of the image does not significantly depend on the block sizes chosen for embedding. However, different curves for different payloads occupy different position in the graph, with PSNR for lower payload size occupying higher position. It can be clearly observed that PSNR changes only when the payload size changes. When the number of message bits embedded into the image increases from 100 to 200, twice more original pixels need to be modified and deviate from the original values. This causes a degradation in PSNR. Seen this way, PSNR is inversely proportional to payload size. Choice of payload size depends on the PSNR value that needs to be achieved or maintained.

Table 6.3. PSNR Values as a Result of Embedding 200 Watermark Bits

Q (%)	**Final PSNR (dB) for Block Sizes (square pixels)**						
	64	***128***	***256***	***512***	***640***	***780***	***1024***
50	72.19	73.62	74.43	72.71	72.49	73.15	73.12
60	72.24	73.74	73.66	72.85	72.39	73.09	73.15
75	72.27	72.85	73.00	72.89	73.09	73.24	73.30
80	72.06	73.72	74.04	72.61	72.24	73.04	73.15
90	71.80	73.47	73.63	72.33	71.94	72.79	72.90

Furthermore, FINAL PSNR values are observed to be slightly better than the RAW values. In other words, this means that the difference in cover and stego images decreases after compression. This is because compression reduces both- the total number of pixels and the number of modified pixels in the stego image which would be greater for an uncompressed image. This gives a better signal to noise ratio (ratio of total to modified pixels) for the compressed image. It might be a bit confusing to read this as PSNR being improved after compression. However, the correct interpretation of this would be that compression did not improve the quality of image but decreased the difference between the cover and the stego image.

Figure 6.4 only shows PSNR values for one particular quality factor of 75. In order to observe that for different Q factors, a series of experiments are performed varying Q from 90 to as bad as 50 (which is not normally done). Then, 200 message bits are embedded for different block sizes for each Q. The PSNR values as a result of these experiments are tabulated in Table 6.3. It is pretty interesting to see that noises introduced are more or less for any level of compression. Hence, any level of compression can be achieved for the method without compromising image quality. The choice of a particular Q factor should be dependent only on the visual perception of the final image.

6.2.2. BER Evaluation

Next, Bit Error Rates (BER) are computed to see if the extracted bits are correct and to what extent. Since JPEG compression modifies the image pixels, BER will be a measure to see if the embedded bits can survive this change. In other words, this is a measure for the robustness of the proposed embedding technique. First, message sequence of 100 bits is embedded into the raw image using the block wise technique and the stego image is

compressed using different Q factors. BER values are then computed to see after the embedded bits are extracted. Figure 6.5 shows some plots of these BER values. The graphs in Figure 6.5 clearly show that BER values are all zero for higher Q factor like 90. This means that for a compression of Q=90, all embedded bits are successfully recovered. This proves the robustness of the method. Comparing result for different Q factors, the embedded bits start getting corrupted with the increase in Q value or increasing the compression. However, it is to be noted that BER does not degrade to a very bad value. For a standard Q factor of 75, BER is still around 2 percent and is very acceptable.

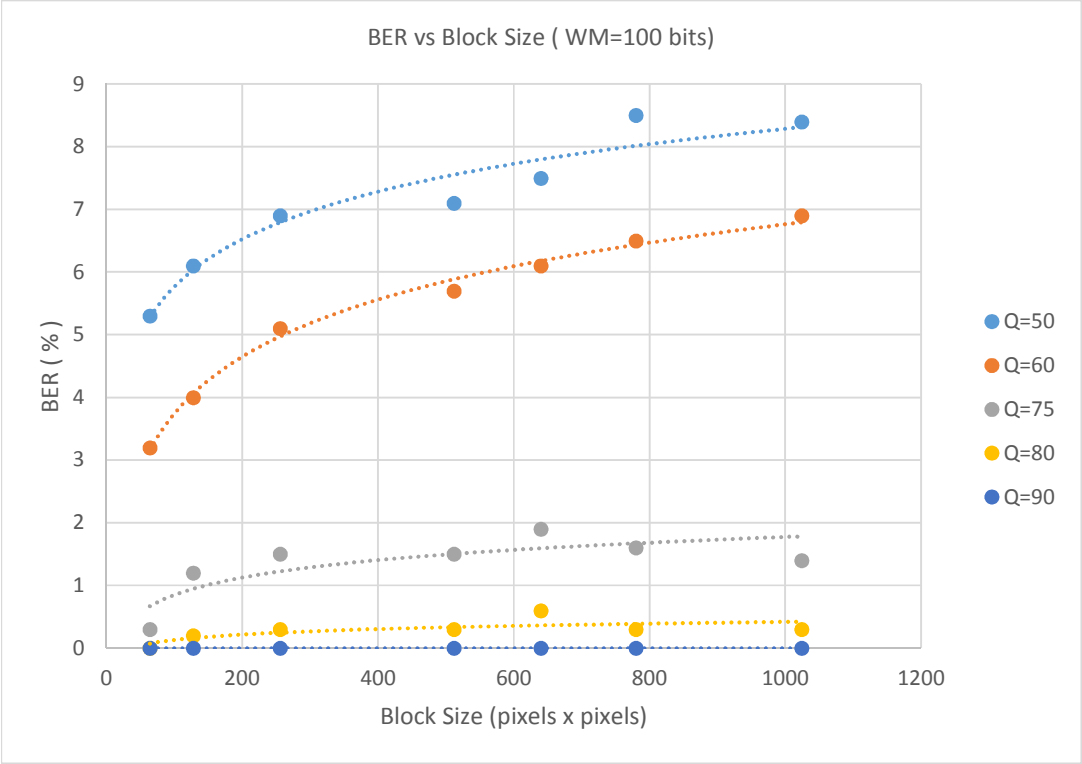

Figure 6.9. BER after the embedding followed by extraction of 100 watermark bits

When Q is reduced beyond 75, to values of 60 and 50, BER degrades further. But again, even for these Q factors, BER values are well within 10 percent proving the robustness of the algorithm compression yet again. Further Q values aren't tested because, it is highly unlikely that quality of an image will be degraded to even a value of Q=50 for any practical purpose.

Another important observation that can be made from Figure 6.5 is that although BER values remain within an acceptable limit for all Q factors and block sizes, they are the lowest for a block size of 64 x 64 pixels for all Q factors. The reasoning behind this is that a two level DWT is chosen for embedding in our implementation. A two level DWT for a 64 x 64 size block gives a total of 16 x 16 pixels for embedding. This is very close to the number of message bits to be embedded which is 100. Since the extraction algorithm compares a pixel

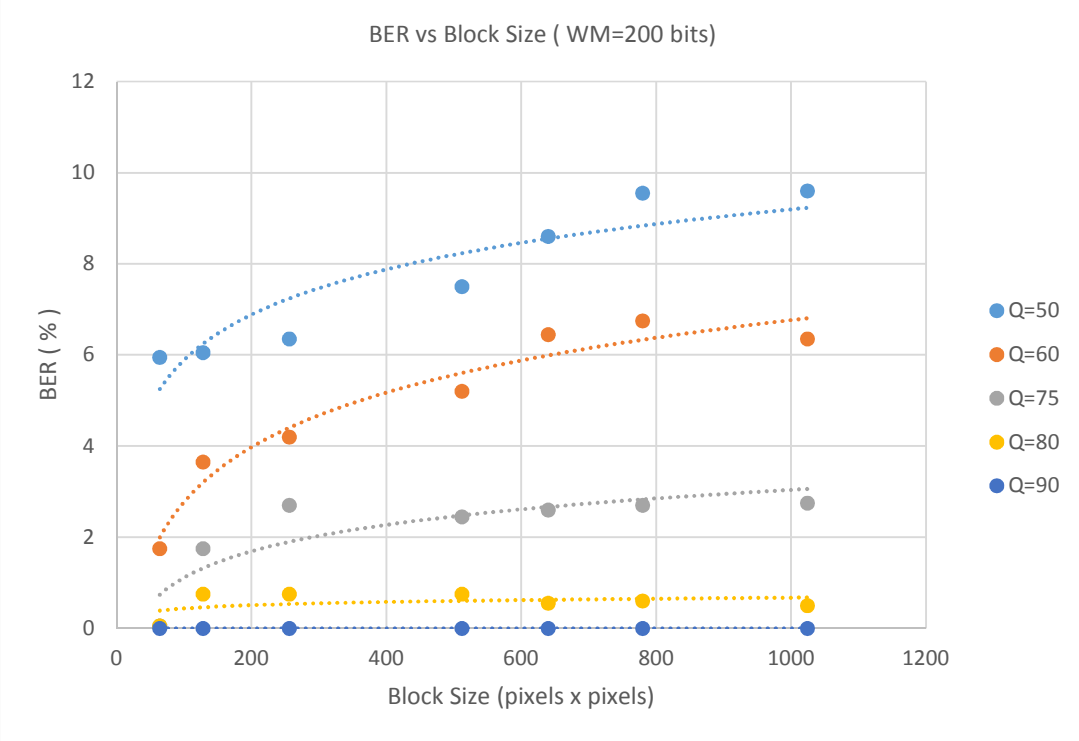

Figure 6.6. BER after the embedding followed by extraction of 200 watermark bits

with neighboring pixels in order to decode a bit, higher the probability of correctly decoding the bits from a smaller pool of pixels. This might not be a significant improvement in BER but it is advisable to choose the smallest possible block size for embedding all bits within a block. The reason being that this saves time and memory as explained in the first set of implementation in this thesis. Similar explanations hold true for message length of 200 bits as shown in Figure 6.6.

Figures 6.5 and 6.6, with BERs for message lengths 100 and 200 respectively, are compared and plotted together in Figure 6.7. The combined plot in Figure 6.6 is the result of embedding 100 and 200 bits into a raw image and compressing it with a Q factor of 75. This figure is to see the effect of increasing the payload length in BER. It is clear that doubling the payload bit length from 100 to 200 does not deteriorate BER at an alarming rate. For this change, BER only degrades to 2.5 percent from a value of 1.5 percent. This means that increasing payload does not necessarily pose a threat to the robustness of the method.

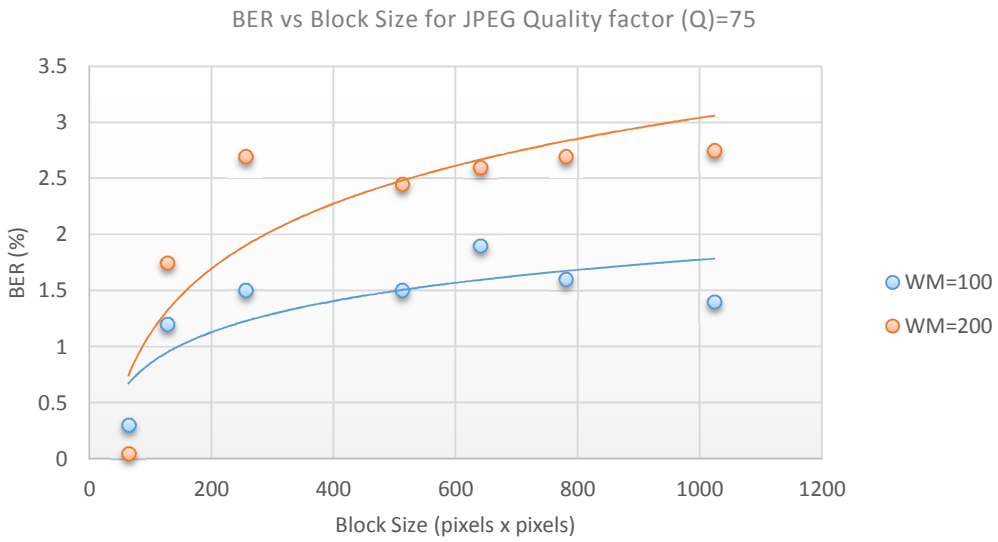

Figure 6.7. BER comparison for watermark lengths of 100 and 200 at a Q factor of 75

Chapter 7. Conclusion

Methods that can successfully implement steganography in digital cameras and smart phones are proposed in this thesis. With the soaring use of smartphones in current times, the proposed methods are made computationally efficient so that they can be used within resource constrained devices. Traditional implementations are mostly used to hide data into multimedia files in a computer system. However, this requires the multimedia files to be transferred to computers for processing. With increasing use of internet in smart phones, user might find this cumbersome. Hence, to provide a better user experience and to make sure the multimedia files are secure, this research proposes to perform multimedia steganography within the mobile devices.

The first idea is to refine one of the existing embedding methodologies to make it efficient such that it creates as less computational overhead as possible. Of course the inclusion of steganography improves digital security at the cost of computation. But in contrast to existing methodologies that are aimed only at implementation and not on efficiency, we strive to make the proposed solution faster and memory thrifty. Also, majority of existing mobile steganography require additional hardware for implementation, we utilize the processing ability of the smart phones. In this regard, the software based solution proposed in this thesis can be readily realized as a smartphone application if needed. We take the basic embedding technique proposed by Nagham *et* al. [20] and implement it in a novel manner to make it much more efficient. The proposed technique for efficiency is to divide video frames into smaller blocks and perform block level processing. This technique rests upon localization of other methods used, such as DWT and feature detection.

Simulation results show significant time saving and efficient memory management with the proposed method as compared to conventional schemes. Furthermore, use of Forward Error Correction to utilize the video frames has proved to make the proposed method robust in the presence of channel noise. The results presented show about 60 percent time saving and 40 percent BER improvement which make the solution better than the existing video steganography schemes that utilize the original algorithm.

The research takes a step forward by attempting automatic watermarking of all images captured by any digital cameras. A method to incorporate previously defined embedding algorithm with existing camera Image Sensor Pipeline is proposed. The primary objective behind this is to ensure that all images are protected without relying on the user to process them, by watermarking them within the camera hardware. Watermarking is done on raw images within the image capturing process before a display ready image is formed. The research shows how the existing ISP can be tweaked to come up with a new ISP that can automatically watermark images and make them secure.

The results after simulation help to shed light on how the proposed technique can be implemented. To ensure that the image quality is not compromised in the process of adding security, PSNR values are computed. For a really bad JPEG compression quality of 50, which is not normally done, a PSNR of 70 dB percent is achieved. With all BER values being below 10 percent reiterates the robustness of the proposed solution.

The steganography solutions proposed in this thesis is aimed to answer existing issues in the field of digital multimedia authentication and forensics applications. The primary benefit of using the solution is making steganography image centric such that each image can be uniquely identified without having to post process them in a computer.

In the future, we intend to further explore the topic and try to integrate video watermarking within the camera ISP as well. Also, it would be exciting to take the research ahead of simulations and actually try to implement them in real hardware, camera and smart phones and compare the simulation results with real time experimentations.

Bibliography

[1] P. Alvarez, “Using extended file information (EXIF) file headers in digital evidence analysis,” *International Journal of Digital Evidence*, Economic Crime Institute (ECI) 2 (3) (2004) 1–5.

[2] J. Fridrich, M. Goljan, D. Hogeg, “Steganalysis of JPEG images: breaking the F5 algorithm,” *Proceedings of Information Hiding: Fifth International Workshop*, IH 2002 Noordwijkerhout, The Netherlands, *Lecture Notes in Computer Science*, Springer, 7-9 Oct., 2002, 2578/2003, pp. 310–323.

[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’02, vol. 1, 21–25 Aug., 2000, 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] Saraju P. Mohanty, Nagarajan Ranganathan, Ravi K. Namballa,“VLSI Architecture for watermarking in a secure still digital camera (DC) design,” *IEEE Transactions on very large scale integration (VLSI) systems*, July 2005, vol. 13, Issue 7, ISSN: 1063-8210.

[7] D. Stanescu, V. Stangaciu, M. Stratulat, “Steganography on new generation of mobile phones with image and video processing abilities,” *International Conference on*

Computational Cybernetics-ICCC, Timisoara, Romania, 27-29 May, 2010, pp. 343-347.

[8] Wu Zhi-Jun, Niu Xin-Xin, Yang Yi-Xian, "Design of speech information hiding telephone" , *Proceedings of 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering, TENCON*, 28-31 Oct., 2002, vol. 1, pp. 113- 116.

[9] K. Papapanagiotou et al., "Alternatives for multimedia messaging system steganography," *Proceedings of the International Conference on Computational Intelligence and Security*, Berlin, Heidelberg, 2005, vol. 2, pp. 589-596.

[10] S. Mohanpriya, "Design and implementation of steganography along with secured message services in mobile phones," *International Journal of Emerging Technology and Advanced Engineering*, May 2012, , vol. 2, Issue 5.

[11] J. Xu, L. Feng, "A feature-based robust digital image watermarking scheme using image normalization and quantization," *2nd International Symposium on Intelligence Information Processing and Trusted Computing (IPTC)*, Hubei, 22-23 Oct., 2011, pp. 67-70.

[12] 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.

[13] 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.

[14] J. N. Ellinas, “A robust wavelet-based watermarking algorithm using edge detection,” *Proceedings of World academy of Science, Engienering and Technology*, Nov. 2007, vol. 25, ISSN 1307-6884.

[15] S. Kay, E. Izquierdo, “Robust content based image watermarking,” *Proceedings of the Workshop on Image Analysis for Multimedia Interactive Services*, 2001.

[16] S. M. Smith, 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.

[17] A. Westfeld, “F5-a steganographic algorithm,” *Information Hiding*, Springer, Berlin, Heidelberg, 2001, pp. 289-302.

[18] CTIA-The Wireless Association, “CTIA-Semi-Annual Wireless Industry Survey”, 2013, *files.ctia.org/pdf/CTIA_Survey_YE_2012_Graphics-Final.pdf.*

[19] Yueh-Hong Chen, Hsiang-Cheh Huang, “A copyright information embedding system for android platform”, *2011 Seventh International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, Dalian, 14-16 Oct., 2011, pp. 21-24.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

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

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

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

[33] 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.

[34] 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.

[35] 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.

[36] 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 .

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

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

[45] 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.

[46] Huang, Yu-Wen, et al. "Survey on block matching motion estimation algorithms and architectures with new results", *Journal of VLSI Signal Processing Systems For Signal, Image and Video Technology*, 2006, vol. 42, no. 3, pp. 297-320.

[47] S. Ashwin, M. Wu, and KJ R. Liu, "Nonintrusive component forensics of visual sensors using output images," *IEEE Transactions on Information Forensics and Security*, 2007, vol. 2, no. 1, pp. 91-106.

[48] S. Ashwin, M. Wu, and KJ R. Liu, "Component forensics," *IEEE Signal Processing Magazine*, 2009, vol. 26, no. 2, pp. 38-48.

[49] S. Ashwin, M. Wu, and KJ R. Liu, "Digital image forensics via intrinsic fingerprints," *IEEE Transactions on Information Forensics and Security*, 2008, vol. 3, no. 1, pp. 101-117.

[50] G. L. Friedman, "Digital camera with apparatus for authentication of images produced from an image file," U.S. Patent 5,499,294, issued March 12, 1996.

[51] J. Lukas, J. Fridrich, and M. Goljan, "Digital camera identification from sensor pattern noise," *IEEE Transactions on Information Forensics and Security*, 2006, vol. 1, no. 2, pp. 205-214.

[52] J. Fridrich, "Digital image forensics," *IEEE Signal Processing Magazine*, 2009, vol. 26, no. 2, pp. 26-37.

[53] H. Farid, "Image forgery detection," *IEEE Signal Processing Magazine*, 2009, vol. 26, no. 2, pp. 16-25.

[54] H. T. Sencar and N. Memon, "Overview of state-of-the-art in digital image forensics," *Algorithms, Architectures and Information Systems Security,* 2008, vol. 3, pp. 325-348.

[55] W. Luo et al., "A survey of passive technology for digital image forensics," *Frontiers of Computer Science in China*, 2007, vol. 1, no. 2, pp. 166-179.

[56] A. Popescu, "Statistical Tools for Digital Image Forensics," Ph. D. Dissertation, Department of Computer Science, Darthmouth College,2005.

[57] Y. Long and Y. Huang, "Image Based Source Camera Identification Using Demosaicing," *Proceedings of IEEE MMSP*, 2006 .

[58] A . Swaminathan, M. Wu and K . J. Ray Liu, "Non-Intrusive Forensics Analysis of Visual Sensors Using Output Images," *Procdeeings of IEEE ICIP*, 2006.

[59] Z . J. Geradts, et al., "Methods for Identification of Images Acquired with Digital Cameras," *Proceedings of SPIE*, 2001, vol. 4232.

[60] E. Dirik, H. T. Sencar and N. Memon, "Source Camera Identification Based on Sensor Dust Characteristics," *Proceedings of IEEE SAFE*, 2007.

[61] S. Lyu and H. Farid, "How Realistic is Photoorealistic?," *IEEE Transactions On Signal Processing*, 2005, vol. 53, No. 2, pp. 845- 850.

[62] S. Dehnie, H. T. Sencar and N . Memon, "Identification of Computer Generated and Digital Camera Images for Digital Image Forensics," *Proceedings of IEEE ICIP*, 2006.

[63] F.A.P. Petitcolas, "Introduction to information hiding", in: S. Katzenbeisser and F.A.P. Petitcolas, (ed.) (2000) Information hiding techniques for steganography and digital watermarking, Norwood: Artech House, INC.

[64] S. Miaou, C. Hsu, Y. Tsai and H. Chao, “A secure data hiding technique with heterogeneous data-combining capability for electronic patient records,” *Proceedings of the IEEE 22nd Annual EMBS International Conference*, July 23-28, 2000, Chicago, USA, pp. 280-283.

[65] Z. Li, X. Chen, X. Pan and X. Zeng, “Lossless data hiding scheme based on adjacent pixel difference,” *Proceedings of the International Conference on Computer Engineering and Technology*, 2009, pp. 588-592.

[66] A. Cheddad et al.,"Digital image steganography: Survey and analysis of current methods," *Signal processing*, 2010, vol. 90, no. 3, pp. 727-752.

[67] S. Bhattacharyya, "A survey of steganography and steganalysis technique in image, text, audio and video as cover carrier," *Journal of Global Research in Computer Science*, 2011, vol. 2, no. 4.

[68] B. Li et al., "A survey on image steganography and steganalysis," *Journal of Information Hiding and Multimedia Signal Processing*, 2011, vol. 2, no. 2, pp. 142-172.

[69] V. M. Potdar, S. Han, and E. Chang, "A survey of digital image watermarking techniques," *3rd IEEE International Conference on Industrial Informatics*, 2005, pp. 709-716.

[70] F. AP Petitcolas et al., "Information hiding-a survey," *Proceedings of the IEEE*, 1999, vol. 87, no. 7, pp. 1062-1078.