Digital Forensics Detection and Disruption of JPEG Steganography

by

George Jasper Trawick II

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
August 9, 2010

Keywords: Steganography, Digital Forensics, Information Assurance

Approved by

John A. Hamilton Jr., Chair, Professor of Computer Science
Kai H.Chang, Professor of Computer Science
Eric S. Imsand, Assistant Research Professor of Computer Science

Abstract

The use of digital media and internet communications has grown for business and personal use, and so has the use of digital communications by criminal and terrorist elements. Of particular interest in both criminal investigations and national security is the use of covert communications channels by terrorists and criminals. A primary challenge faced by law enforcement is founded in the tremendous capacity of modern storage devices. The terabyte sized digital storage available to the public today allows for enormous amounts of evidence items to be hidden among millions of mundane, generic files. To forensically image and analyze these files can take days, sometimes weeks. When a criminal uses steganography to hide illicit content inside the otherwise mundane files, the investigators' mission becomes many times more difficult, if possible at all. Criminals are using applications that implement digital steganography to secretly communicate, plan, coordinate and execute their unlawful activity. When criminals add encryption to the immense number of steganographic implementations available it results in a combination proving to be nearly insurmountable for law enforcement and government agencies. This research introduces a unique and novel method that allows for the detection and possible tracking of steganographic messages hidden on a suspect's digital storage device.

The current state of the research shows that there are opportunities in exploration of specific areas of JPEG implementation combined with non-traditional hashing

techniques; to find new methods for detection of steganographic images. This research focuses on a particular genre of steganography that is implemented using JPEG images. In particular this research focuses on those steganographic implementations that exploit the transform domain of the JPEG compression algorithm. The findings in this research demonstrate that within the results from the JPEG compression algorithm are artifacts that remain constant between repeated compressions. This research uses emerging methods of non-traditional hashing to expose likely steganographic images within a finite set or database of images and identify a possible digital fingerprint allowing law enforcement new capabilities in coping with the use of steganography as a tool for criminal activities.

Acknowledgments

I am truly proud and humbled by all of the people who have willingly sacrificed for me and have contributed to my research, both directly and indirectly. It has been my privilege to work with a great collection of outstanding professors and students during my time at Auburn University. I begin by thanking my advisor, John Andrew Hamilton Jr. Drew your unwavering support has helped me in succeeding in the completion of this research. I am honored to have been a part of the Information Assurance Exercise and the Wounded Warrior Forensics training that Auburn University provides to our nations wounded service members. I will keep the memories of teaching those men and women with me for the rest of my days. I must also recognize the sacrifices of my family during this long struggle of research. To Mom and Dad, watching over me still, your love and wisdom instilled in me those traits necessary to achieve all that I have. Dad, thank you for inspiring me to become a leader of men, a seeker of the light that knowledge provides and teaching me to always choose the path of greatest reward, not of least resistance. To my wife and daughters, Robbie, Tiffany, and Tasha, each of you have provided me with the inspiration I needed throughout my time at Auburn and in the U.S. Army, to keep moving forward. Thank you, for keeping me constantly striving to set the best example for you and to make you as proud of me as I am of each of you.

To the Long family, Cheryl, Ben, Keller and Abbey, you may never fully realize how much your generosity and sacrifices have enabled Robbie and I to achieve the completion of this research. I am forever indebted to each of you. I must also pay a special tribute to my brothers and sisters in arms, those members of the United States Armed Forces, who stood watch on the front lines; while I labored safely here at home. Not a day passes that I am not grateful for your sacrifice and aware of my blessings. There are several dozen folks who have stood beside me over the years. Of note are the leaders and mentors I had the fortune to work for at the United States Joint Forces Command. First, a special thanks to Lynn Schug, the true Chief Information Officer (CIO) of Joint Force Command. Thank you, for believing in me and putting your reputation on the line to see that I was not overlooked by the labyrinth of leadership within Joint Forces Command. Thank you for helping me to set the foundations that led me to this point and never letting me travel too far off track. To Colonel Mark Vanous, USMC, thank you, for your mentorship, leadership and friendship. It was your commitment to my success that assured that I would have this once in a lifetime opportunity and your unwavering support that motivated me to see it through. Finally to Colonel Robert Troisi USAF, thank you for your guidance and support. It was your experience and no non-sense leadership that kept me focused on the goals of achieving all that I possibly could both militarily and academically.

Finally, I want to thank God for all the blessings I have received in my life, the glory of my achievement most surely belongs to You.

Table of Contents

List of Tables and Figures

Definitions

Zero Variance Minimal Computer Unit (ZVMCU): The nomenclature used to describe a grouping of 64 pixels extracted as an 8 X 8 block, where the color values of every one of the 64 pixels are equal. Having identical RGB values results in the JPEG compression routine having minimal impact on the DCT coefficients of that MCU.

Image Categories: A compartmentalization of images based upon their origin, size, resolution, and size -resolution relationship. There are eight image categories used in this research.

Data Sets: A data set is a compartmentalization within an Image Category that further divides the group of images into a manageable number based upon the number of baseline ZVMCUs within the image. Each Image Category is divided into six individual datasets.

Minimal Computer Unit: Terminology used to describe the 8 x 8 block of pixels used by the JPEG compression routine and used in this research as the building blocks of the partial image hash routine.

Hidden Data: Can be any type of digital data that is concealed using steganography. The hidden data is embedded into a cover image.

Cover Image: An image file that is used as a container for embedding hidden data. A cover image does not have hidden data; it is merely a container. Once data is hidden within the image, it becomes a Steganographic Image.

Steganographic Image: Also called a Stego Image: is an image file with data hidden inside the file structure. The data may be hidden with any number of steganographic techniques, as detailed in Appendix A. A steganography image is created using a cover image and hidden data.

Consanguineous: From the American Heritage dictionary "Of the same lineage or origin; having a common ancestor." (The American Heritage® Dictionary of the English Language 2004.) In this dissertation, consanguineous is used to differentiate JPEG images which are visually identical, but are not from the same original parent image, from those images that originate from the same singular parent JPEG image. For readability the term sibling is used interchangeably with consanguineous.

Digital Hash: A hash is an alphanumeric value mathematically derived by applying a hashing algorithm to a digital file. A digital hash must be one way, meaning you cannot derive the digital file contents from the digital hash. Also, every digital hash for every digital file must be unique; no two different files should ever have the same digital hash value.

Chapter 1

Introduction

Steganography is the art and science of secret communications. The use of steganography can be dated back thousands of years. Some of the earliest known steganographic uses were military in nature. Beginning with Demaratus' use of wax tablets to cover secret orders to his commanders in the war on Greece around 480BC, (McCullagh 2001) (Hosmer and Hyde 2003) through World War I and II with the use of microdots and invisible inks, and continuing today with Al-Qaida's use of steganography to organize attacks on America and other countries around the globe, covert communications has proven a necessary and valuable tool. (Kelly 2001) (Kellen 2001)

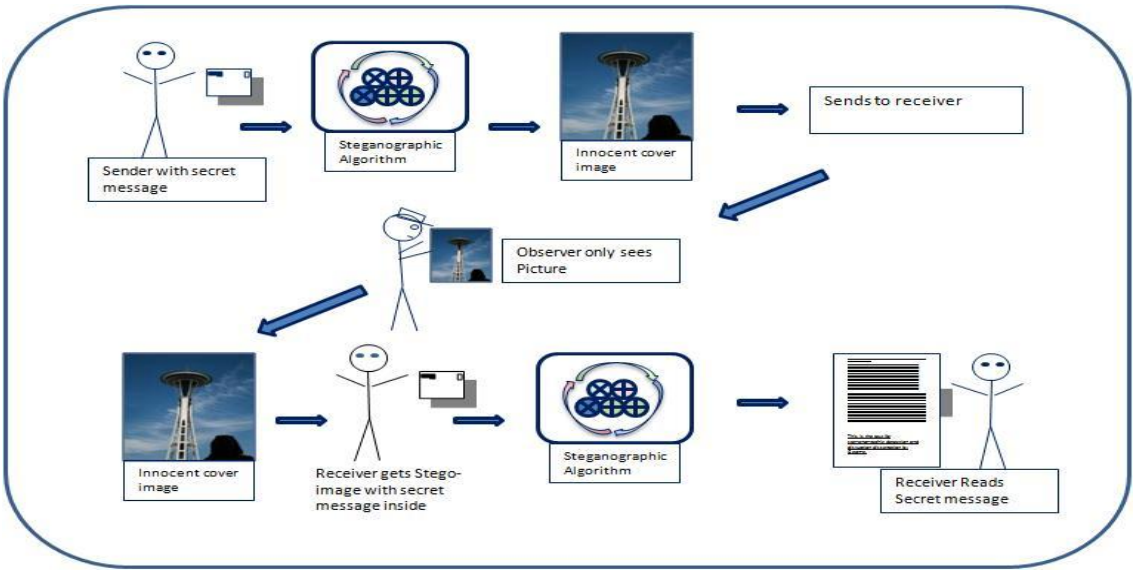


Figure 1: Steganographic-Secret communications process.

When applied in today's digital world, the use of steganography allows for the secret transmission of large amounts of information from one source to many receivers covertly or it may be used to simply hide criminal activities from outside detection. The use of this technology is not limited to terrorists and warriors. All types of criminals are exploiting the power of steganography to hide their illegal activities from law enforcement. Of all the criminals, child pornographers are of particular interest. Child pornographers are using steganography to secretly transmit and store illegal images of children free from detection by current law enforcement wiretap and tracing techniques. Steganography is nearly impossible to detect in transmission, and it is just as difficult to detect within data at rest on a computer. Law enforcement and national security analysts need improved methods to detect hidden content, in order to find, track, prevent, and prosecute these criminals.

While there is ongoing research in both the embedding and detection algorithms of steganography, most of the research is limited in scope to a particular type of media or

to a particular set of known steganographic programs. Further, the current state of the art steganography detection shows that there is still vast room for improvement in both the detection and extraction of covert messages within steganographic images. The research presented here demonstrates previously unknown methods enabling law enforcement the ability to detect and possible track JPEG steganographic images residing within a finite set of digital images. This capability will prove especially useful for use by computer forensic investigators.

This research offers the following contributions to the computer science field. This research produced an ability to identify unique, stable, and persistent landmarks within an image that endures the JPEG lossy compression algorithm, which does not exist today. Further, these methods are the first to use non-traditional hashing techniques (Hurlburt 2009) (J. Kornblum 2006) (Roussev, Golden and Marziale, Multi-resolution similarity hashing 2007) as a method to compare images with the specific purpose of steganography detection among a captured set of JPEG images. Next, by using new techniques and combinations of old techniques, this research is unique in offering law enforcement methods that will reduce the amount of time and effort needed in finding and extracting only those images that are most likely to have hidden content. The results of this research provide law enforcement investigators and national security analysts a new ability to find consanguineous or sibling images on a computer that have different digital message digest. Having the ability to find persistent landmarks allows law enforcement to discover and possibly track how and where images are being shared.

The remainder of this dissertation is organized as follows. Section 2 summarizes related work in the area of image steganography, steganalysis and non-conventional

hashing techniques. Section 3 is motivational example. In Section 4, the experimental design, experiment processes and experiment results are detailed. Section 5 details the key contributions and conclusions of the research. Section 6 gives an overview of future research and work.

Chapter 2

Related Work

2.1 Strengths/Limitations of Related Work

The related works to this research are widely varied covering areas from digital steganography applications, Joint Photographic Experts Group (JPEG) standard implementation and compression, digital hashing algorithms and steganalysis. While the effects of the JPEG compression routine on the implementation of certain steganography algorithms formed the genesis for this research, the limitations in the areas of steganography, steganalysis, and non-traditional hashing compose the main motivators of the research.

2.1.1 Steganographic Applications

Currently there are many researchers who are involved in extensive work researching to identify and quantify the implementation strengths and weakness of the currently available steganographic implementations (Kharrazi, Sencar and Memon 2005) (Lin and Delp 1999) (Dunbar 2002). Evidence shows that researchers are focused on new and innovative techniques to improve the current state of steganographic technologies, (Chae and Manjunath 1999) (Fridrich, Pevy and Kodovsk, Statistically undetectable jpeg steganography: dead ends challenges, and opportunities 2007) (Filler and Fridrich 2009). Researchers for professional organizations and corporations seem interested in detection

and extraction of steganographic images (Cole 2003) (Dunbar 2002) (Hurlburt 2009) (Kellen 2001). In their research of the current strengths and weakness of universal steganalysis, Kharrazi et al, worked with both steganalysis and steganographic technologies, and demonstrated several embedding techniques used for JPEG images (Kharrazi, Sencar and Memon 2005). In other research, Lin and Delp explained that there are generally three common techniques used to embed information in digital images and classified these techniques as Least-Significant Bit embedding (LSB), transform embedding, and perpetual masking (Lin and Delp 1999). Other authors, like Karen Bailey and Kevin Curran (Bailey 2004), describe the embedding techniques from a slightly different perspective. These authors describe image noise, clutter, texture and signal as four general models for embedding a message in an image. LSB embedding would fall into the noise model, transform embedding would be categorized as signal, and perpetual masking would be in the clutter model. The texture model is more suited for watermark technologies as the manipulation within this domain could be made visible in the image. Yet another author, Eric Cole, has a different classification scheme.

In his book, Hiding in Plain Sight, Cole describes three classifications for steganography, insertion based, substitution based, and generation based (Cole 2003). This categorization is broader and more general than the previous examples such that Cole's substitution classification would encompass all three areas previously described. The insertion based techniques do not alter the original bits that represent the image; instead they exploit areas of the file that are disregarded by applications that read the file. This has an advantage of allowing a larger size hidden message but a disadvantage in that

it is easier to detect and extract. The generation class steganographic applications create a new image file from the message file.

The major drawback to creation method is that the file created, usually a fractal image, (Cole 2003) does not represent a real object and would draw attention from most examiners. There are hundreds of available steganographic programs (appendix A). This study will use only a few of the more well-known applications from each model or classification set as an example to explain the overall methodology of the implementation routines.

Least-Significant Bit (LSB) embedding is used by the applications like S-tools and EZsteg. (Johnson and Jajodia 1998) S-Tools and EZsteg employ the LSB technique each using a slightly different implementation with S-tools focused on 24 bit images and EZsteg working with 8 bit images. Neither of these implementations will work with JPEG images, however, because the lossy JPEG compression algorithm would destroy any embedded data (Johnson and Jajodia 1998). As noted by Neil Provos, (Provos 2001) the LSB embedding process inevitably alters the statistical characteristics of an image file, making any rudimentary LSB implementation statistically detectable. Another technique is employed by the program Outguess, which uses two steps in its steganographic algorithm. First, Outguess uses a method to determine the maximum amount of data that can be embedded without detectable changes to the images' first order statistics and the second step is the use of a pseudo-random number generator (PRNG) to generate a seed starting point overcoming the limitations of serial insertion into the LSB of an image. However, because Outguess replaces the bits that represent the colors of an image, it suffers from visual degradation of the image as the embedded file

size reaches its maximum. The next steganographic technique researched is the method employed by the F5 application.

F5 does not use LSB to embed the secret message. Instead, F5 falls into the category of transform techniques. F5 embeds its digital message in non-zero AC-DCT[1] coefficients, if the embedding process causes the coefficient to become zero; it skips to the next non-zero AC-DCT coefficient. Using the Discrete Cosine Transform (DCT) coefficients combined with a PRNG that is seeded by a user entered password, F5's decrements the absolute value of the DCT coefficients by one. Then using a matrix embedding technique F5 spreads the secret message across the cover image in such a way as to be undetectable by most statistical attacks, according to the author of the algorithm (Fridrich, Goljan and Hogea 2003). As pointed to by Quach et al. (Quach, Perez-Gonzalez and Heileman 2009) many steganographic programs try to avoid detection by preserving first order statistics, hoping to make the files statistically identical.

In all there is plenty of research in the area of embedding information into digital images in a multitude of manners, each with its own strengths and weaknesses. This area of research is useful and necessary for discovering new techniques, as well as, improving on old techniques of steganography. Understanding this area of research is indispensable for anyone interested in the field of steganalysis specifically for the intentional detection, disruption and extraction of steganographic messages.

[1] The terminology AC-DC is a reference to electrical current. The term meaning Alternating or Direct Current for electrical circuits, here denotes the difference DCT coefficients. The AC coefficients are subject to quantization, and thus change, where the DC coefficients are not quantized and remain stable.

2.1.2 Steganalysis

Where steganography is defined as the art and science of covered or concealed writing, Steganalysis is the art and science of the discovery of the covered writings or secret communications. Germane to the research here, steganalysis is any technique or procedure that can be used to discover information embedded in a digital file. Just as cryptanalysis is to cryptology, steganalysis is to steganography. Similar to cryptanalysis, there are several broad areas of Steganalysis. Quach et al., point out two categories they call "method-specific and universal". (Quach, Perez-Gonzalez and Heileman 2009) These terms are very descriptive. The method- specific steganalysis category aims to detect only certain types of steganography or steganographic images produced from a particular program. Universal steganalysis looks to discover the presence of any hidden data in a steganographic image regardless of technique. (Goljan, Fridrich and Holotyak 2006) There are databases of known steganographic signatures available to the forensic investigators (Wingate 2005) that fall into the method-specific category, and while useful, there are increasing numbers of new and undocumented techniques that demonstrate a need for universal detection.

Several interesting examples of universal techniques come from Fridrich et al. (Fridrich, Goljan and Du, Steganalysis based on JPEG compatibility 2001) and Lyu et al. (Lyu and Farid n.d.) Fridrich's approach uses the DCT quantization tables of both the cover image and steganographic image. The quantization tables are examined and compared to determine if the resulting DCT coefficients have been modified. In another work by Lyu et al, they used "multiscale and multiorientation image decomposition" to analyze the first and higher order magnitude and phase statistics of digital images. They

developed an effective universal steganalysis technique but its capability was limited to detection of steganographic images that had hidden data that exceeded 95% of the capacity of the cover image. This technique would be ineffective against steganographic images that embed to less than 50% of capacity. These two techniques have a couple of elements in common. First, is the reliance on the availability of the original cover image for comparison. Second, both techniques are looking for a set of features that can be used to distinguish a steganographic image from a non-perturbed cover image. The relationship of the cover image, hidden message, steganographic algorithm and steganographic image is shown in figure 2.

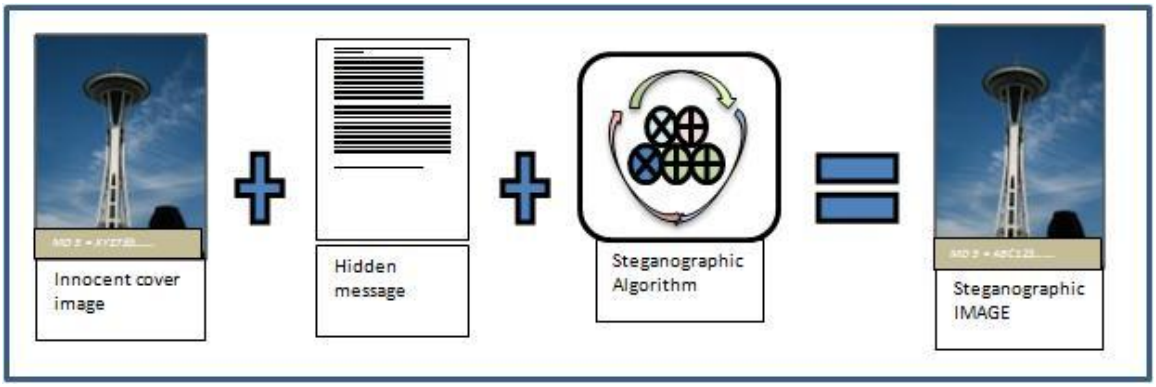


Figure 2: Steganographic Component Relationships

Unlike cryptanalysis, to defeat steganalysis there is no need to reveal the hidden message. In steganalysis revealing that there is secret communications occurring removes the power of steganography. But similar to cryptanalysis, steganalysis still attempts to reveal the hidden or secret message.

In the discussion of universal steganalysis, it is useful to compare quickly with universal cryptanalysis.

Three scenarios for uncovering the encrypted or in this case hidden message exist:

The analyst has:

1. The original cover image file and suspected steganographic image file.
2. The hidden message file and the known steganographic image file.
3. The steganographic embedding algorithm and steganographic image file.

Steganalysis applications and theory rely on having access to more than one portion of the embedding method. Universal steganalysis uses any two of the three in combination to create a data set of known artifacts. Then using analytical methods, they use these known artifacts to create steganographic classifiers to identify suspected steganographic files (Quach, Perez-Gonzalez and Heileman 2009). This research will provide a method that allows the analyst access to the original cover image if it's available

1.3 JPEG Implementation Standards.

The Joint Photographic Experts Group (JPEG) is responsible for the publication of the JPEG standard, formally known as ISO/IEC IS 10918-1 (Wallace 1991) (Khayam 2003) and is officially titled *Digital Compression and Coding of Continuous-tone Still Images* (ITU81). The standard is a collaborative effort between the International Telecommunications Union (ITU)[2] and the International Standards Organization (ISO) to develop standardized methods for the representation, storage, and transmission of graphical images. Though the standard outlines requirements for both lossy and lossless implementations, the default and most widely used implementation is the lossy format and is the focus of this research. It is the changes to the digital signature of an image file during the compression and decompression of digital images that presents the forensic

[2] ITU (International Telecommunication Union) is the United Nations Specialized Agency in the field of Telecommunications. Some 166 member countries, 68 telecom operating entities, 163 scientific and industrial organizations and 39 international organizations participate in CCITT which is the body which sets world telecommunications standards (Recommendations).

examiner with the challenge of finding and matching suspected image files with known image files. The quantization of the DCT coefficients during the JPEG compression process is the fundamental area of data loss, and hence changes to the digital hash of an image file. However, data loss during the compression process is not the only reason that two identical inputs will produce digitally different outputs. As noted by Wallace, "independently designed implementations of the very same (Forward Discrete Cosine Transform) FDCT or (Inverse Discrete Cosine Transform) IDCT algorithm which differ even minutely in precision by which they represent cosine terms or intermediate results, or in the way they sum and round fractional values will eventually produce slightly different outputs from the very same input." (Wallace 1991) This is the reason that an image opened and saved in an image editing software package will have a different digital signature from the original image or the exact same image opened and saved by a second image editing software package. An in-depth examination of the JPEG compression routine introduces several areas of interest for research into digital artifacts. Figure 3 gives an overview of the JPEG compression/decompression components in sequence. Of note is that the quantization tables and Huffman coding tables used are stored in the header information of the compressed file in accordance with the EXIF standard. (JEITIA, EXIF Format for Digital Still Cameras: Version 2.2 2002)

A deeper look at each of these is important to understand where in this process we might find an exploitable artifact that will remain constant between compressions as well

as, any limitation that will exist in the stability of such an artifact.

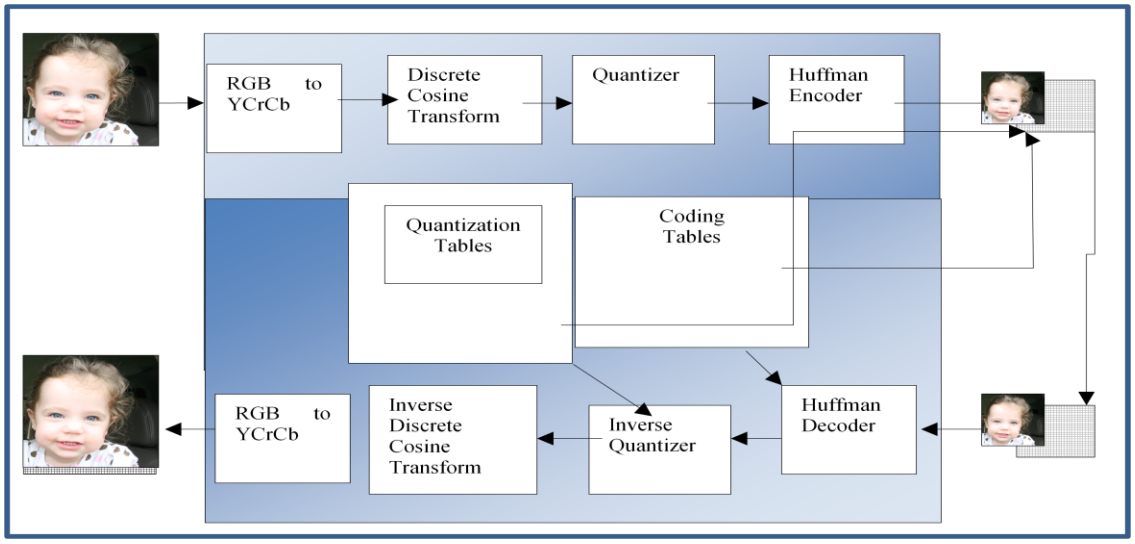


Figure 3: Visual Example of the JPEG Compression Routine

Figure 3 is a visual representation of the JPEG compression and decompression routine, of note is the location of the quantization tables in the process.

The formal equation for the two dimensional discrete cosine transform takes the form that follows:

$$\boldsymbol{C(u,v) = \propto (u) \propto (v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos\left[\frac{\pi(2x+1)u}{2N}\right] \cos\left[\left[\frac{\pi(2y+1)v}{2N}\right]\right]}$$

In the DCT transform equation above,

$$\boldsymbol{\propto (u,v) = \left\{\sqrt{\frac{1}{N}}\right\} for\ u,v = 0}$$

$$\boldsymbol{\propto (u,v) = \left\{\sqrt{\frac{2}{N}}\right\} for\ u,v \neq 0}$$

In the example, u and v are horizontal and vertical spatial frequencies for an 8x8 sample.

An image is a series of pixels laid out in a 2 dimensional X-Y plain. The two dimensional Discrete Cosine Transform yields a set of frequencies based upon the values derived from 8x8 blocks of these pixel values. The 8x8 blocks (F(x,y)) of image values are extracted starting in the upper left corner and working from left to right, top to bottom of the image. (Khayam 2003) The first step in the compression of a JPEG image is to transform the pixel data from the 24 bit Red Green Blue (RGB) domain into the 8 bit YCbCr (YCC) domain. The algorithm takes as input three 8 bit numbers ranging from (0-255) representing red, green, and blue colors of a pixel. The algorithm seen in figure 4 demonstrates how the numbers are then parsed through a reversible, but not lossless, color space transform into Luminance(Y),Chrominance Red (Cr) and Chrominance Blue (Cb), commonly written as YCrCb, as mandated in ITU-R BT.601. (JEITIA, EXIF Format for Digital Still Cameras: Version 2.2 2002)

$$Y = 0.299R + 0.587G + 0.114B$$
$$Cb = (-0.299R - 0.587G + 0.886B) * 0.564 + offset$$
$$Cr = (0.701R - 0.587G - 0.114B) * 0.713 + offset$$

Figure 4: The RGB to YCbCr transform from the EXIF standard version 2.1.

As noted by Sorell (Sorell 2008) and Khayam (Khayam 2003) transforming the values from the RGB color space to the YCrCb color space, allows for the targeted reduction of only those values that are least detectable by the human eye. These YCrCb values are further adjusted from unsigned values to signed values before they become the input to the DCT. Figure 5 is used to demonstrate a common visual example of the 64 DCT coefficients of an 8x8 image block. The horizontal frequencies are represented by u in the transform equation and the vertical frequencies are represented by v.

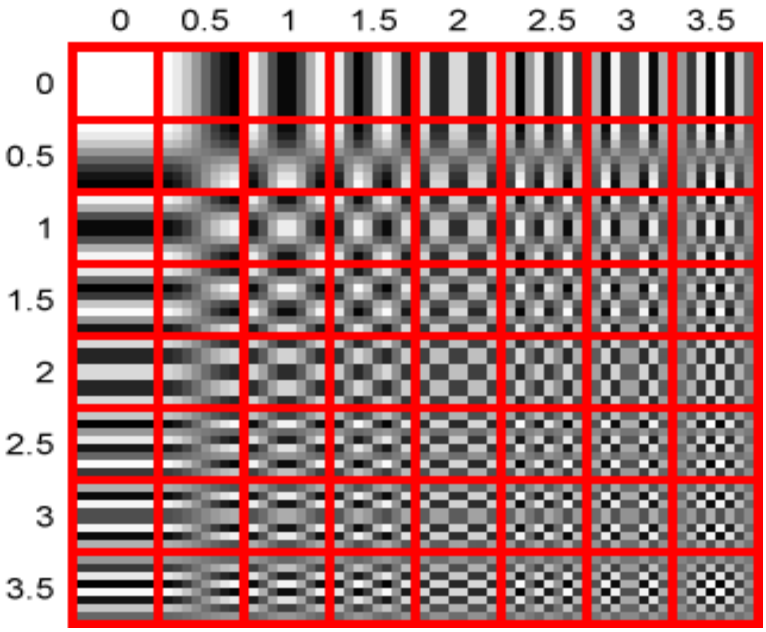


Figure 5: Visual Representation of 64 DCT coefficients (Khayam 2003)

As a visual example, figure 5 also demonstrates how the lower frequencies of an image block are represented in the upper left corner, where the higher frequencies are presented in the lower right corner. This critical step allows the use of quantization tables in the quantization step to be predetermined for certain levels of quality. These predetermined tables may offer an opportunity to find stable artifacts or a fingerprint that remains through the JPEG compression routine. Kornblum noted in his research that it is possible to distinguish images that have been modified by software from those untouched images through the analysis of the quantization tables. (J. D. Kornblum 2008)

The quantization tables have larger values towards the lower right corner which when used as the divisor of the quantization coefficients allows for greater reduction of the higher frequency values within the DCT coefficients.

Specifically, splitting the signal in to three channels, Y is the brilliance or brightness of a particular pixel and CbCr or chrominance of blue and red channels respectively. The samples values are then shifted by -128 to center the values on zero.

(Wallace 1991) The focus of the JPEG DCT lossy compression is on the color channels. The more detectable channel of brilliance is recorded as the DC component in the most upper left hand corner of the coefficient transform matrix. These 8x8 blocks of shifted CbCr values are then used as input into the 2D DCT with the output being the 64 DCT coefficients for the 8X8 block (F(u,v)).

The outcome of the DCT is that the 8x8 blocks have been transposed from their DCT coefficients into the spatial frequency[3] signal components, specifically the DC or non fluctuating and AC or changing components. Figure 6 demonstrates a commonly used example of a resulting 2D DCT transform. Of note in figure 6 is the relative positioning of the absolute values of the coefficients. Notice how the values progress from upper left to lower right. Also, notice the DC value in the upper left corner is much larger than the remaining AC values.

$$\begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix}$$

Figure 6: 2 Dimensional DCT Transform

This feature of the 2 dimensional DCT to always place the DC (non-changing) value in the upper right corner provides an avenue of research for a stable JPEG artifact. However, this value is not stored and is dependent on the actual Y channel value of the preceding 8x8 block (MTU). This value is subject to change from one compression to the next and does not remain absolute through the compression decompression cycle. However there is evidence that it is possible to identify the tables used in prior

[3] Spatial Frequency is a measure of changes in pixel values across an image block

compressions and therefore, possible to determine the original DC value of an image. (Sorell 2008)

Another area of possible exploration is the use and storage of the quantization tables.

An example quantization table that is often used in literature is shown in figure 7.

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Figure 7: Quantization matrix. Courtesy: images.digitlmedianet.com

Though, there are commonly used tables that are recommended by the JPEG standard (JEITIA, EXIF Format for Digital Still Cameras: Version 2.2 2002), applications and developers are free to implement their own. However, most standard and readily available applications use the recommended tables; this was problematic to Kornblum's research (J. D. Kornblum 2008). Kornblum researched the possibility of determining the origin of an image, either by camera type or software developer, based upon the values used in the quantization tables. So the use of standardized tables is problematic for Kornblum's categorization research. However, it is the standardized implementation of the tables that is germane to the research presented here not the values within the tables. Further, there is evidence that the steganographic community has knowledge of the flaws of the current quantization tables and some are looking to use custom tables to improve steganographic performance (Almohammad, Ghinea and Hierons 2009).

2.1.4 Digital Hashing

Roussev et al addressed some of the issues facing digital forensic investigators today. "Large-scale digital forensic investigations present at least two fundamental challenges. The first one is accommodating the computational needs of the large amount of data to be processed. The second one is extracting useful information from the raw data in an automated fashion". (Roussev, Richard and Marziale, Multi-resolution similarity hashing 2007) This research addresses these two challenges using novel methods to identify only those images most likely to contain steganographic material and sibling images within a dataset.

2.1.5 Overview of traditional cryptographic hashing.

A hash function is a mathematical algorithm that takes as its input an arbitrary amount of data and produces an output of a predetermined fixed length. The length of the output is the same regardless of input size. Cryptographic hash functions are regarded as primitive functions for many other cryptographic functions. (Silva 2003) The viability of a hash function is dependent upon several basic principles.

For a one way hash function:

The hash function must be published with no obscured input.

The output of x' is always a fixed and predictable length regardless of input x.

The hash function h(x) must generate a unique output x' for all input x.

There can be no information about input x derivable from x'. (Preneel 2005)

For the area of digital images, as with other digital files, hash functions are useful and powerful tools that allow the detection of even the slightest change of a document or file. Further, for the field of digital forensics, hash values offer a powerful method for

examiners to filter through the enormous amounts of digital data and separate known files from unknown files. Companies such as Access Data provide databases of cryptographic hashes of known file types for just this purpose. (Hurlburt 2009) There are other companies and agencies that offer specialized hash databases such as the Steganography Analysis and Research Center (SARC) that provide a database for known steganographic programs. (SARC Editort 2009) There is also the National Center for Missing and Exploited Children (NCMEC) that provides a hash database of images of known exploited children to law enforcement agencies. (NCMEC n.d.) These applications of cryptographic hashing while useful do not fulfill the purpose of this research. There are several areas of non-traditional hashing that shows promise for allowing the comparison and extraction of visually similar digital files from a finite set of digital files.

2.1.6 Piecewise Similarity Hashing

Jesse Kornblum detailed his technique of context triggered piecewise hashing in 2006.

(J. Kornblum 2006) The author describes a method to use traditional hash functions in a method he refers to as context triggered, to determine similarity between two digital files whose traditional hashes would not match. As outlined in the paper, if a single file (F) is modified even by one bit, either through insertion or deletion, then saved as a separate file (F') the cryptographic hash h(F) will not equal the cryptographic hash h(F'), however, F & F' will share large sequences of identical bits. A simple example would be changing a few words or even a single character in a text document or email then saving the document. Comparison of the cryptographic hashes of the two files would produce different hash signatures. However, the majority of the content in the two

documents is identical. Context triggered piecewise hashing breaks down the contents of a file into context triggered segments and then hashes each segment separately. The similarity between the files becomes a function of the number of identically hashed segments found within the file, see figure 8. This will prove to be a powerful tool for the computer forensic examiner allowing him to find text files that have been edited but are in fact the same document. There is a limitation of the author's method that was noted. This method is not effective on JPEG image files. The JPEG compression algorithm modifies nearly every portion of the image file; the piecewise context triggered hashing would not be effective. (J. Kornblum 2006)

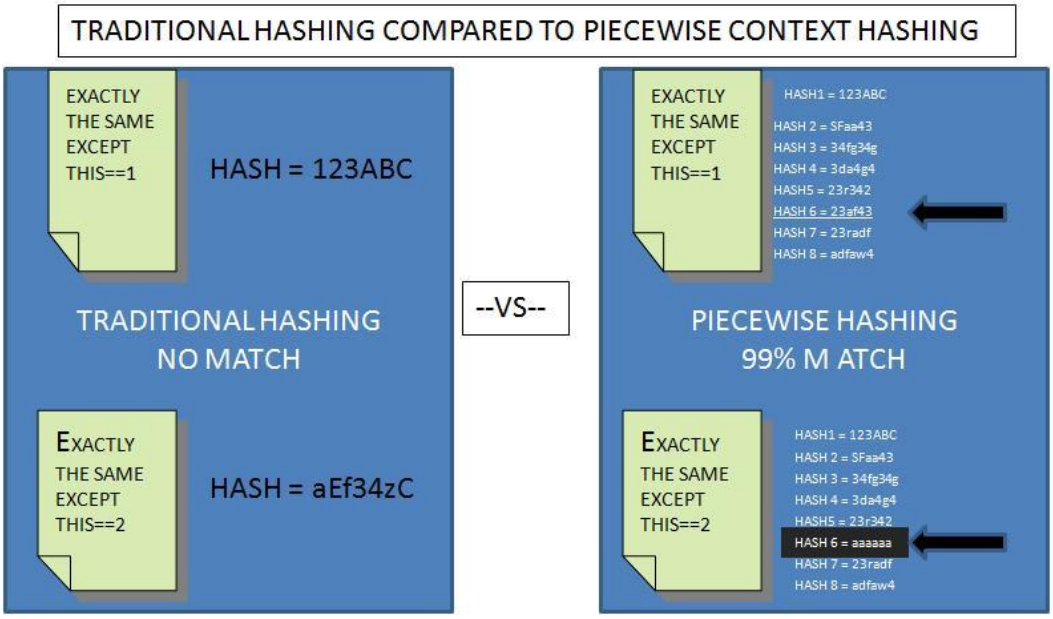


Figure 8: Traditional Hash routine compared to Piecewise Context Hash Routine

2.1.7 Multi-resolution Similarity Hashing

In working to address the fundamental challenges facing digital forensic investigators mentioned earlier, Roussev et. al worked to help reduce the effort and time required when creating hashes of large amounts of data, such as a multi-gigabyte storage device. In traditional forensic cases, the first order of business is to acquire a forensically

sound copy or image of the data set under investigation. With the large sizes of today's storage devices, this process can take several hours to complete. Once complete a digital hash will be computed to validate the data integrity of the image and then the investigator can begin work on the collected data image. In their work, Roussev et al set out to develop a multi-resolution similarity hash that will run concurrently with the copy routine. (Roussev, Golden and Marziale, Multi-resolution similarity hashing 2007) The content is split into variable sized portions and then hashed into a series of Bloom filters.[4] This allows for context discovery while the initial copy is being performed.

The authors list several other benefits such as the technique works on arbitrary pieces of data, it is scalable allowing comparison of data sets that are orders of magnitude different in size, and maintains privacy, allowing partial hashes of datasets to be compared against known files without having to seize or open the questionable data.

This concept of multi-resolution similarity hashing is useful for forensics, but falls short of being able to accommodate the bit level changes that are incurred when a JPEG file is manipulated by steganographic software. Therefore, it would not be capable of extracting consanguineous JPEG image files.

The following graphic demonstrates the differences between the traditional methods for collecting forensic evidence from a suspect's computer and the Multi-Resolution Similarity Hashing method.

[4] A *probabilistic algorithm* to quickly test membership in a large set using multiple *hash functions* into a single *array* of bits. (http://www.itl.nist.gov/div897/sqg/dads/HTML/bloomFilter.html)

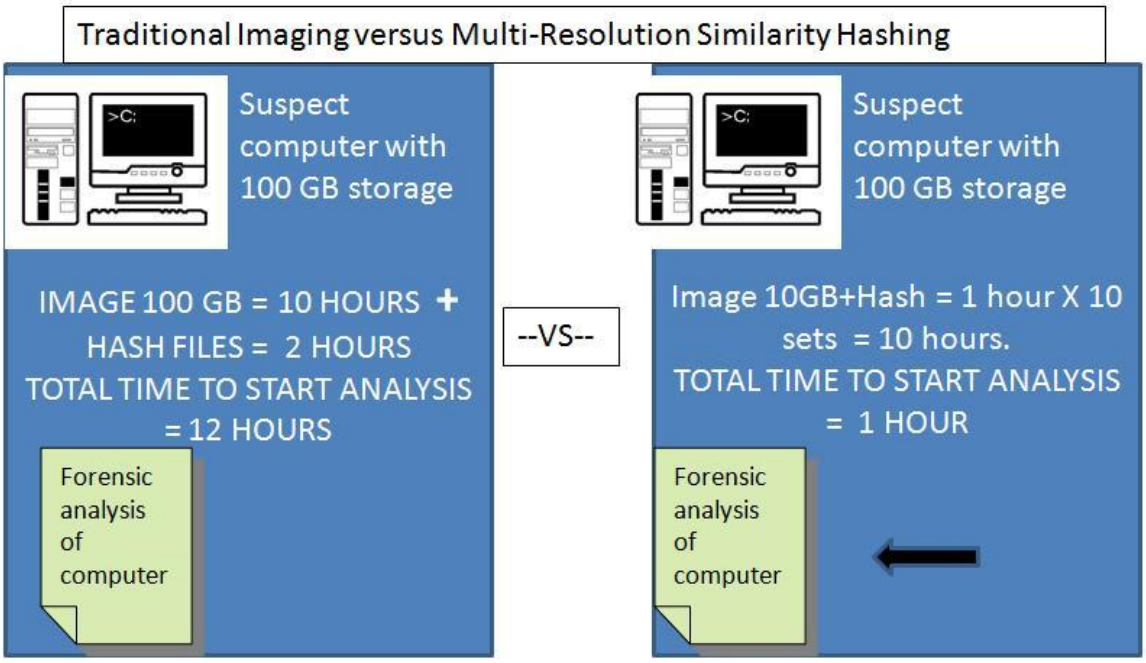


Figure 9: Demonstration of Digital Forensic imaging

Another researcher that is looking at non-traditional hashing with an eye for both forensic and steganographic applications is Dustin Hurlbut from Access Data. In his publication, "Fuzzy Hashing for Digital Investigators" (Hurlburt 2009) the author details the methods used by Access Data's Forensic Tool Kit implementation of a fuzzy hash. This publication describes an actual forensics oriented implementation for the work done by Kornblum in his piecewise similarity hashing. There are a few minor differences in how the trigger values are determined and the implementation of the rolling hash algorithm. Otherwise, fuzzy hashing as described by Hurlbut is a corporate instantiation of context triggered piecewise hashing with the same JPEG oriented limitation as the original process.

2.2 Observations

As explained in the previous section, there are gaps in the current state of digital steganography and steganalysis research. Further, there are possible discoveries that can be made in exploration of the JPEG implementation scheme and non-traditional hashing that could produce significant improvements in digital forensic investigations. The discoveries presented in this research bridge these gaps, explores the possibilities for digital forensics, and furthers the current state of the research providing an innovative method for the discovery of possible steganographic images and consanguineous JPEG image files.

1. Identify and bridge gaps in research between state of the art fuzzy hashing, serial hashing and JPEG compression for the identification of nearly identical sibling JPEG image files in a finite population.
2. Develop a method that will allow for the unique identification of JPEG images that is robust enough to survive recompression by JPEG lossy compression techniques.
3. Develop a quantifiable method for the reduction of suspect population of images, significantly reducing the forensic effort necessary to detect, identify and/or disrupt steganographic communications and techniques.

Chapter 3

Motivational Example

Digital forensic investigators are taking notice of the use of steganography by ordinary users. The fact that steganography is available to hide communication and data from an investigator coupled with the extreme difficulty in detection provides the initial motivation for this research. In a whitepaper from BackBone security steganalysis is described as "very young and evolving extension of traditional forensics" (Wingate 2005). BackBone Security is the parent company of the Steganography Analysis and Research Center (SARC). SARC provides a comprehensive database of known steganographic signatures, as well as, state of the art steganalysis software to the nation's digital investigators and law enforcement officers. In researching the current state of steganography in digital forensics, I found there are areas of opportunity that might be exploited that can further improve the state of steganalysis, such as the fact that different JPEG manipulation software likely use different quantization tables resulting in different hashes of an identical file (J. D. Kornblum 2008), the use of different quantization tables has been explored by Farid, and he has produced a method to determine if a JPEG image has been altered by editing software and to identify which software most likely last edited the image. (Farid 2006)

Other research has indicated possible landmarks in JPEG images. Fridrich alluded to a "semi-fragile watermark or unique fingerprint" that is available for comparison. (Fridrich and Goljan, Practical steganalysis of digital images: state of the art 2002). The apparent viable avenue of exploration in the JPEG compression routine may provide a method to identify stable landmarks in a JPEG image provides another motivation for this research. Additional motivation comes from a related areas of research, cryptographic hashing. Cryptographic hashing has proven a valuable and necessary tool in digital forensics. It is common to use databases of hashes of known files to discover or discard digital evidence in a storage device. Research has proven the usefulness of using partial or piecewise hashing to discover related text files, but has stopped short in being able to reliably do so for JPEG images. Even though doing so would greatly enhance an investigators chance of finding a steganographic cover object. (Hosmer and Hyde 2003) This lack of useful hash methods for JPEG images provides the final piece of motivation for this research. The research presented in this dissertation demonstrates that using the previously unexplored JPEG landmarks called zero variance minimal computer units (ZVMCUs) and a variation of the context triggered piecewise hash, called ZVHash it is possible to find visually similar JPEG files in a finite dataset. Further, using the comparisons of the ZVMCU quantities of the cover image and steganographic image it is possible to predict potential JPEG steganography in the same finite dataset.

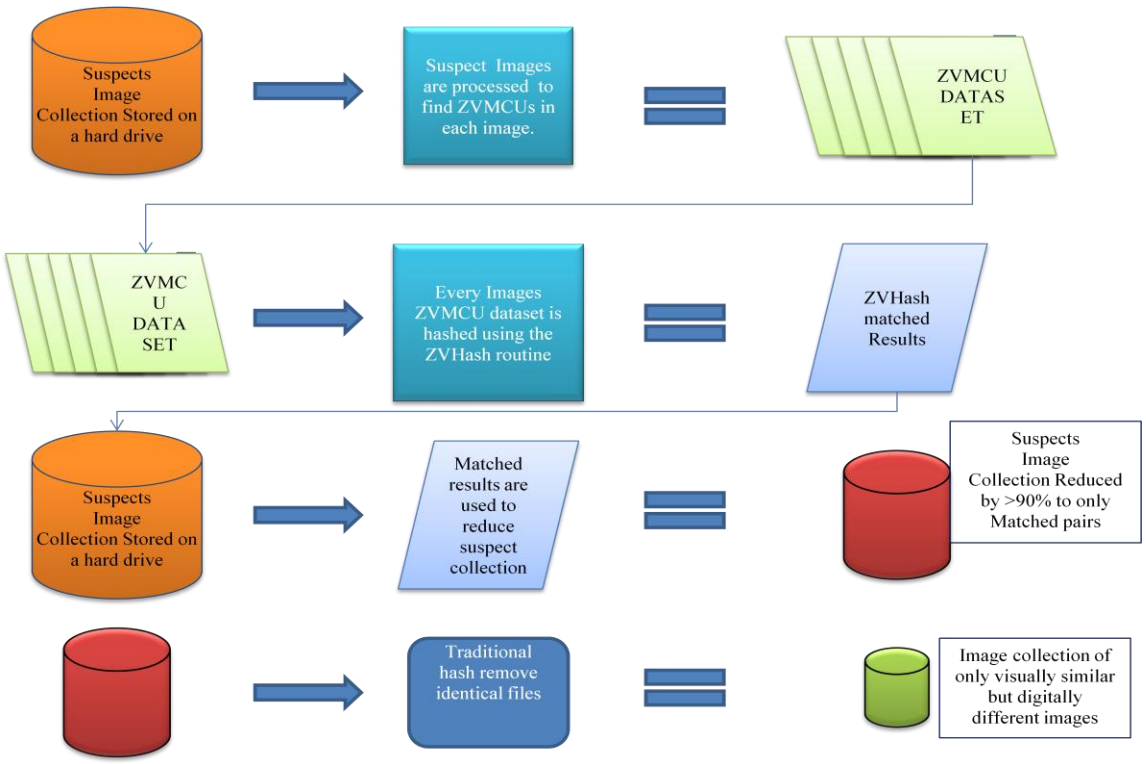


Figure 10: Steganographic & Sibling image detection process overview

Figure 10 gives a graphical introduction to this process that is described in detail in the rest of this work. The research presented in this dissertation shows that in the implementation of the JPEG compression standard there are avenues for the exploitation of the piecewise hash method that allows an investigator to match a suspected steganographic object with its cover object. The findings in this research provide new found methods for the detection of JPEG steganography and for the location and forensically viable matching of consanguineous JPEG images within a finite dataset.

Chapter 4

Experimental Design and Theory

In the creation of a sample set of JPEG images on which to conduct experiments, the ability to sample the universal population of JPEG images is nearly impossible as there are no methods that would allow for the random sampling neither of every possible storage device nor to account for every JPEG image created. The size of the sample used in this research was a primary consideration. The sample size needed to be large enough as to facilitate reducing any unintentional bias that might occur from the limited sampling techniques. However, the sample size could not be so large as to make it computationally infeasible to manage. Initial calculations of an estimated sample size was conducted using the following formula from Creative Research Systems (Creative Research Systems 2010)

$$\mathbf{ss} = \mathbf{Z^2} * \mathbf{P}(\mathbf{1} - \mathbf{P}) \Big/ \mathbf{c^2}$$

Where:

ss = sample size

Z= confidence level (95% for this research)

P = decimal value of likelihood of selection of wrong choice (50%)

c= confidence interval, normally between 1 and 5. (3 for this research)

Selection of 95% confidence level provides assurance that the sample size will be adequate to mirror the universal population of JPEGs giving us a relative assurance that a any sample JPEG drawn from the parent population would have a 95% probability of fitting into our sample population this helps maintain the integrity of the findings of the research. The selection of a confidence interval also called the margin of error is normally expressed as a plus or minus expression of acceptable error in a survey. A margin of +/- 3% is a typically acceptable error rate for most sampling.

Given these parameters the recommended sample size is

$$ss = .95^2 * .50 *(1-.50)/.03^2 = 251$$

Careful consideration was given to the additional selection criteria for the images used in this research as well. These additional criteria also influenced the research sample size. The history of the image is an important factor in the experiments conducted in this research, as such; the availability of JPEG images with a known history became a determining factor for the sample size. A sample set of 1500 images, approximately 2.5 gigabytes, were selected from a population of 3000 known history images available. These images where further categorized by size into three groups of 500 each that created the standard for the rest of the research sample. Therefore, a like number of 1500 images with unknown history were acquired as were 1000 additional special category images. The final sample image database consists of 4,000 distinct JPEG images ranging in size from 346 bytes and a resolution of 24 X 24 pixels to 14 Megabytes and more than ten million pixels and a total size of 4.46 gigabytes. The upper bounds of the image size were set at 14 megabytes. A recent study of popular digital cameras shows that the more common, compact point and shoot style cameras

have a pixel range from seven to twelve megapixels and the more expensive professional grade digital SLR cameras produce between twelve and fifteen megapixels. (Goldborough 2008) These upper limit capacities result in uncompressed image size of approximately 45MB, with JPEG compression the storage size of these images become approximately 3MB). (Forret 2010) However, the efficiency of the JPEG compression routine is dependent upon the frequency distribution in the image; some images may have far less compression. Therefore, the sample database includes image sizes ranging up to 14MB. The image database is divided into eight image categories based on the images history and image size. Image history is important for realistic testing. Because we are investigating the impact of image editing software on the viability of indelible image landmarks, we need to be able to compare between raw images straight from the capture device and those images that have been opened and possibly modified by image editing software. As the realization of zero variance minimal computer units may be dependent upon the image size and number of overall pixels. Image size and resolution are also used as factors for categorization. These images were collected through two primary means. First, the known history categories 1-3 are all original images taken by the author using a variety of digital cameras. Second, categories 4-7 are images that were located and downloaded from a variety of sources on the Internet. Category 7 is a special category in that the images are very small in size and low resolution, commonly referred to as thumbnail images. Category 8 is a mixture of known and unknown history images, creating a realistic mixture that replicates what might be found on a typical user's storage device. The statistical methods used in the validation of the determination for sample

size and confidence of a random sample is detailed in section 4.1.3 Data Sample and Image Category Validation.

4.1 Persistent Discernable Landmarks in JPEG Images (Theory I)

Problem: A computer investigator using traditional hashing methods to locate visually similar JPEG images will find only limited success. This traditional method of finding identical computer files fails to find all visually similar JPEG images, because, the DCT compression routine of JPEG images changes the traditional message digest or hash values between two, otherwise, identical images (Hurlburt 2009). The massive size of storage devices today and the number of possible images an investigator may need to review makes a manual review impracticable if not impossible. Digital forensic investigators need a method to find those JPEG images that are visually similar among the possible tens of thousands of image files on a suspect storage device. The discovery of persistent digital landmarks allows investigators to search for visually similar or sibling JPEG images on the storage device. Further, the method is useful in the discovery of multiple copies of visually similar but digitally different images on a storage device which might be another indicator of steganographic use.

Other efforts at locating digitally similar files have produced some very good products such as fuzzy hashing techniques (Hurlburt 2009), context triggered piecewise hashing (J. Kornblum 2006) and multi-resolution similarity hashing. (Roussev, Golden and Marziale, Multi-resolution similarity hashing 2007) However, each of these approaches differs in how the files are located, and none have proven useful for locating visually similar, digitally different JPEG images.

The foundational theory to this research is that JPEG images possess certain landmarks that can be used to find visually similar but digitally different JPEG files in a finite dataset. The hypothesis for the first set of experiments is those JPEG images possess certain landmarks that endure the JPEG compression routine. In addition to finding visually identical but digitally different images, more germane to the research here is discovery of consanguineous images. The term consanguineous, is used occasionally in this dissertation, it distinguishes between two types of visually similar but digitally different JPEG images.

JPEG images that share a common parent image are consanguineous. However, two pictures taken of the exact same object just seconds or months apart are two distinct, non-consanguineous images, though they maybe visually identical and indistinguishable. This important distinction is necessary as legal considerations may dictate the need to present and defend an images history or lineage.

4.1.1 Experimental Data and Software Collection

The database of images used in this research is categorized into eight image categories based on size, history, and resolution. For every image in the database a digital hash or MD5 message digest is stored in a separate file. An MD5 message digest is a common method used to uniquely identify digital files. The MD5 algorithm takes in a digital file of any length and produces a unique 32bit alphanumeric message digest that is unique to that digital file. Each image category is then evaluated for the number of zero variance minimal computer units (ZVMCUs) using searching algorithm implemented in a program written by the author called ZVMCUlocator.py. (Appendix K) A zero variance minimal computer unit is a grouping of 64 contiguous, non-overlapping

pixel values collected in a manner that mimics the JPEG compression routine of using 8 by 8 blocks and whose individual values are all identical. Figure 11 is an example of ZVMCU compared to a normal MCU. The examples in figure 11 are actual MCUs from database used in this research.

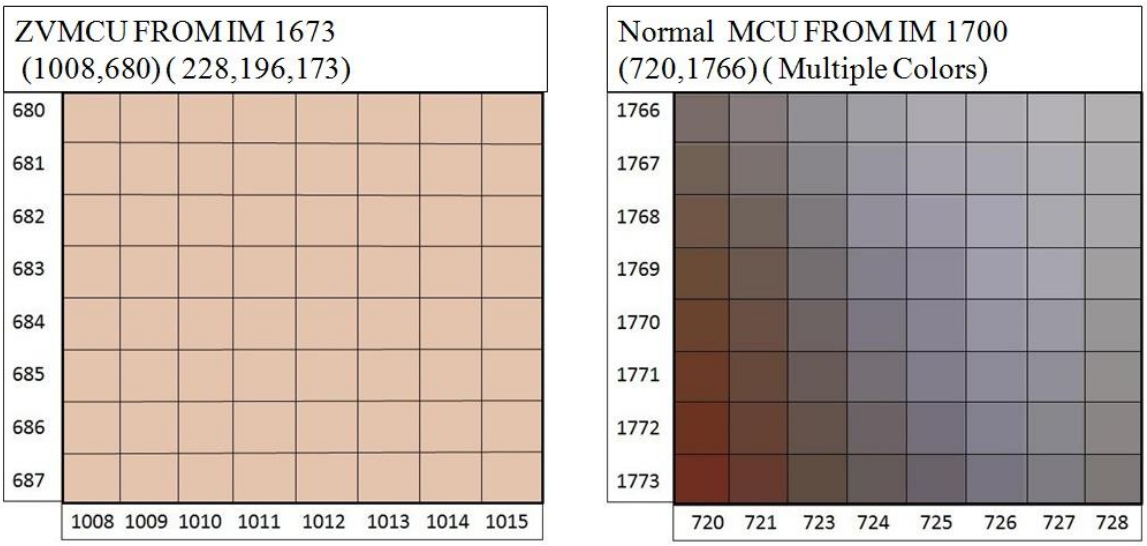


Figure 11: A ZVMCU compared to a Non-Zero Variance MCU

The example on the left is a ZVMCU with all 64 pixels having the same value of 228,196,173 and the MCU having a starting X-Y coordinate value of 680,1008. The MCU on the right is a normal MCU with pixel values that vary across the 8x8 block. The results from ZVMCUlocator.py are then used to further categorize the image categories into six datasets based upon the number of ZVMCUs found in each image. For testing, eight different methods, discussed in detail in the following sections, each baseline image is manipulated. Each resulting manipulated image is stored and evaluated independently. This program is used to locate any minimal computer units (MCU) that have a zero factor frequency variation (ZV) which results in a set of AC DCT coefficients of all zero values.

In figure 12, a general understanding of how the ZVMCUlocator examines an image creating an output that includes the number of ZVMCUs in the image and the color-coordinate pairs associated with each ZVMCU in the image.

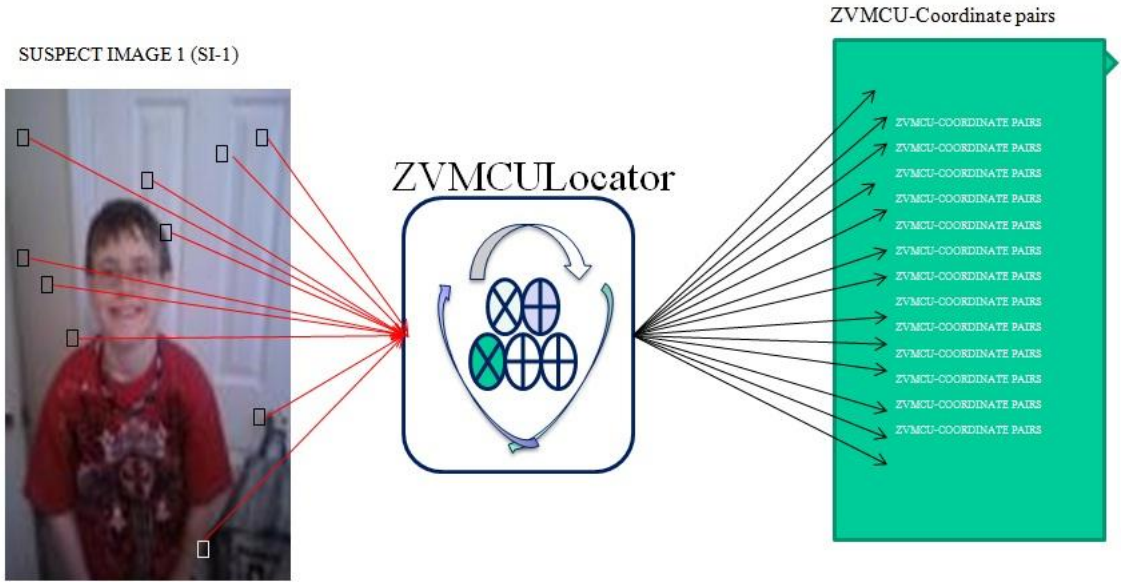


Figure 12: The ZVMCU Locator Program

After all images from all categories are manipulated and evaluated, the resulting data of the number and location of the ZVMCUs in each image is evaluated. The evaluation of this data generates the necessary proof that ZVMCUs are present in JPEG images that they remain between manipulations and are stable enough for further examination as a viable landmarks within a JPEG image. Figure 13 gives a visual representation of the comparison process. Each manipulation is compared against the baseline to determine the effects of the manipulation on the ZVMCU population of the image.

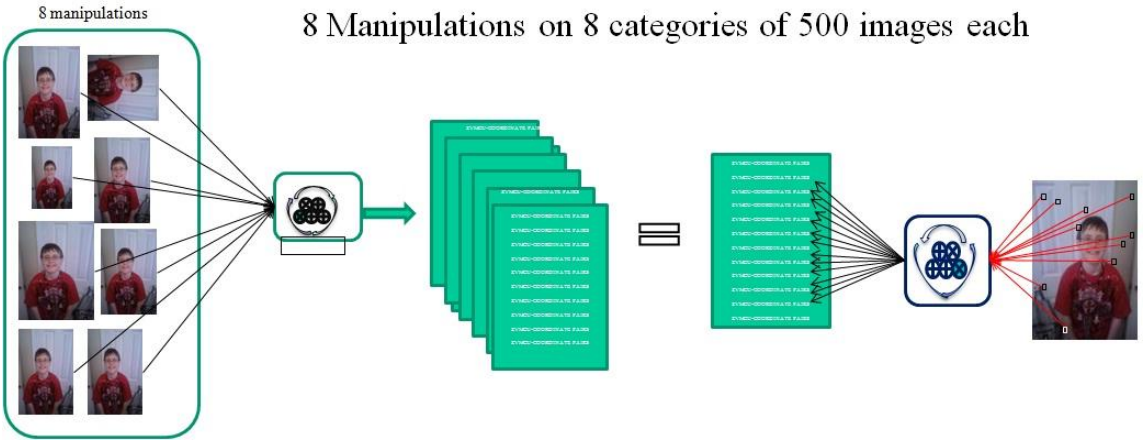


Figure 13: Manipulation Comparisons to Baseline for Establishment of ZVMCUs

Software selection:

GIMP image editor: This software package is chosen because of its popularity as an open source image editor and the powerful features it offers for batch processing images. GIMP is used to perform seven of the eight manipulations, quality reduction to 80%, rotation of 180 degrees, size reduction to 80%, quality reduction to 50%, size increase to 120%, quality restoration of test1 to 100%, and quality restoration of test 4 to 100%. The final manipulation opens each image independently then performs a save as operation renaming and relocating the image to a new directory. This final manipulation was performed with a custom program written by the author, Mover.py (Appendix N).

Adobe Photoshop 8: This software is used because of it powerful feature set and it is a very popular commercial offering of image editing software. This software was used as a validation tool to verify the actions being performed by GIMP.

Using Photoshop to open a select set of images and evaluate that the expected manipulations were reproduced in a separate software package and that both software programs were reporting the same metadata about the images.

Microsoft Photo manager: Microsoft Photo manager was used sparingly in the research, but is chosen for use because of its default installation on Microsoft Operating Systems makes it widely available for use.

IRFANVIEW: This freeware, open source image editing software package is chosen for its ability to view HEX and EXIF information of JPEG images. While useful mainly in the preliminary stages of the database creation; the ability to view, record, and evaluate the HEX and EXIF data proved useful in the decision making process for image selection.

Image selection:

For the purpose of this research a sample database of 4,000 JPEG images was collected. The images were collected from several sources and are categorized based upon each images' size, dimensions, and history.

The categorizations provide the ability to differentiate the viability of the landmarks found in the images. Allowing for the examination of the ratio of ZVMCU's to image size and evaluation of the impact of image editing software on the ZVMCU's.

The first three categories are raw images that are known not to have been edited by software. These images have all been acquired by the author from several cameras and loaded directly into the database. The three categories are divided by image size and resolution, with category I containing the largest images average size 4.1MB, category II contains medium sized images with an average size of 857KB, and category III the

smallest with an average size of 115KB. Categories IV through VI contain 1500 images downloaded from a variety of internet sites. The histories of these images are unknown. This sample of 1500 images is divided into three categories. These categories are characterized by image size and resolution with category III having the largest images and category VI the smallest. Category VII and VIII are special cases. Category VII is comprised of thumbnail sized images less than 3 kilobytes. Category VIII is special in that it contains both known and unknown history images with sizes that are all less than 1 megabyte. A detailed description of each category follows:

- Image Category I is 500 known history photographic images with an average size of 4MB and a resolution between seven and nine megapixels.
- Image Category II is 500 known history photographic images with an average size of 857KB and a resolution between one and two megapixels.
- Image Category III is 500 known history photographic images with an average size of 115KB and a resolution just under one megapixel.
- Image Category IV is 500 images with unknown history with an average size of 2.9MB and a resolution between three and nine megapixels.
- Image Category V is 500 images with unknown history with an average size of 552KB and a resolution between one and three megapixels.
- Image Category VI is 500 images with unknown history with an average size of 133KB and a resolution less than one megapixel.
- Image Category VII is 500 thumbnail size images with unknown history and an average size of 3KB and a resolution less than 10 thousand pixels.

- Image Category VIII is 500 images with known & unknown history with an average size of 433KB and with a resolution greater than 3 Megapixels.

Hash Selection:

MD5: used to validate the integrity of the original images prior to testing

SHA1: alternate method for image integrity validation

4.1.2 Theory I Experiment and Data Structure

The foundational theory to this research is that JPEG images possess certain landmarks that can be used to find visually similar JPEG files in a finite dataset. The process for experimentation of this theory begins with the examination of the base line images in each category. First, an MD5 hash is obtained and stored for all base line images. Next, using the custom program ZVMCUlocator.py, each image is examined for the existence of zero variance minimal computer units. The ZVMCUlocator program mimics the JPEG compression routine in traversing the image and examining each pixel as a member of an 8x8 blocks or minimal computer unit (MCU). If the RGB values of all the pixels within the MCU are equal to one another, this block is labeled as a zero variance minimal computer unit (ZVMCU) and its color value and coordinates are recorded along with other metadata, such as image height, width and size. Each Image Category is then sub-divided into six datasets based upon the number of baseline ZVMCUs found in the images. Dataset 1 is a collection of images that have zero ZVMCUs in the images. Dataset 2 is a collection of images with between one and ten ZVMCUS. Dataset 3 is a collection images with greater than 10 but less than 100 ZVMCUs. Dataset 4 is a collection images with greater than 100 but less than 1000 ZVMCUs. Dataset 5 is a collection images with greater than 1000 but less than 10K

ZVMCUs. Dataset 6 is a collection images with greater than 10K ZVMCUs. Each image is manipulated in a series of eight tests and then the image is revaluated for changes in the number of ZVMCUs.

In the testing of the images for theory I, 8 independent tests are conducted as detailed in the following sections. The results from the test manipulations are recorded and the results are presented. Each of the 8 tests performs a singular manipulation of each of the 4000 images in the baseline dataset with the exception of test 6 and test 7. Test 6 and 7 use image copies that resulted from manipulations in test one, reduction in quality by 80% and test four, reduction in quality by 50%, respectively, allowing the examination of the impact of multiple manipulation on an image.

4.1.3 Data Sample and Image Category Validation

The images used in this body of work represent a cross section of images that would constitute a typical users JPEG image collection. Statistical tests were conducted on all 4000 images of the baseline dataset as well as on the resulting datasets from each of the test manipulations conducted in this dissertation. The intent of this testing is to provide a level of assurance of the random nature of the image collection and to minimize any unintentional bias of image selection or of the population of ZVMCU's within the images. The tests chosen to examine the randomness of the data sample used in this work, the Wald-Wolfowitz Runs test, a Frequency test, and the Kolmogorov-Smirnov test (K-S test). The detailed results from each test are available for review in Appendices Q, R, and S respectively.

The Runs test is used to determine if the input data comes from a random process. (Filliben and Heckert 2010) The null hypothesis for the Wald-Wolfowitz Runs test ($H_{0)}$

is that the sample collection is the result of a random process. The alternative hypothesis for this ($H_{1)}$ is that the sample population demonstrates characteristics that are not consistent with a random sample.

In figure 14, the results from the Wald-Wolfowitz Runs test demonstrate that we cannot reject the null hypothesis that our distribution comes from a random process. The National Institute of Standards and Test (NIST), handbook on statistics states that the z-score is compared to a standard normal table and that at a 5% significance level, a z-score with an absolute value greater than 1.96 indicates non-randomness. (Filliben and Heckert 2010)

Wald-Wolfowitz RUNS test for randomness									
BASELINE		**TEST 1**		**TEST 2**		**TEST 3**		**TEST 4**	
Runs		**Runs**		**Runs**		**Runs**		**Runs**	
Observed	1940	Observed	3161	Observed	248	Observed	144	Observed	3501
Expected	1942.055	Expected	3163.289	Expected	248.323	Expected	141.972	Expected	3497.341
2-tailed Test		**2-tailed Test**		**2-tailed Test**		**2-tailed Test**		**2-tailed Test**	
Z	P	Z adjusted	P	Z adjusted	P	Z adjusted	P	Z adjusted	P
0.391	0.348	0.583	0.560	-0.113	1.090	1.126	0.260	1.670	0.095
TEST 5		**TEST 6**		**TEST 7**		**TEST 8**		**ALL ZVs ALL TESTs**	
Runs		**Runs**		**Runs**		**Runs**		**Runs**	
Observed	1703	Observed	3166	Observed	3536	Observed	2857	Observed	2849
Expected	1698.954	Expected	3163.340	Expected	3537.027	Expected	2855.624	Expected	2837.843
2-tailed Test		**2-tailed Test**		**2-tailed Test**		**2-tailed Test**		**2-tailed Test**	
Z adjusted	P	Z adjusted	P	Z adjusted	P	Z adjusted	P	Z adjusted	P
0.941	0.347	0.705	0.481	0.268	0.789	0.307	0.759	0.715	0.475

Figure 14: Wald-Wolfowitz Runs Test Results

As seen in figure 14 all z scores from all image categories individually, as well as a single collective run, were all below the 1.96 necessary to reject the null hypothesis. Figure 15, shows a Runs Test conducted on the size attribute of the entire image sample

set. In this case, all 4000 images were used in the Runs Test. At a 5 % error rate, the adjusted Z-score was -1.389, reinforcing the failure to reject the null hypothesis for the sample image dataset used in this research. The details of the calculations and the Wald-Wolfowitz Runs test algorithm are defined in detail in appendix Q.

Wald Wolfowitz Runs Test for Randomness	
Data Range = 4000 ALL IMAGES	
Runs	
Observed	3929
Expected	3928.912
2-tailed Test	
Z	P
-1.389	0.082

Figure 15: Wald Wolfowitz Runs test on All images (size)

The next test conducted to test the randomness of the data sample is a frequency test, followed by a K-S test for normality. According to Hamilton et al. the Frequency test and the K-S test are useful in measuring the similarities between the empirical distribution and the distribution of the sample population. (Hamilton, Nash and Pooch 1997) The Frequency test conducted on the sample set contained in this work was calculated using Stat Plus 2009 v.5 software package, which creates a frequency of the occurrence of values across the distribution of a series of partitions based upon the overall size of the sample. 58 partitions were selected using the formula of

$$\boldsymbol{Partitions} = \underline{\sqrt{\boldsymbol{N}}}$$

With the following results in the frequency distribution of ZVMCUs across all images in the sample set demonstrated in figure 16 and detailed in Appendix S.

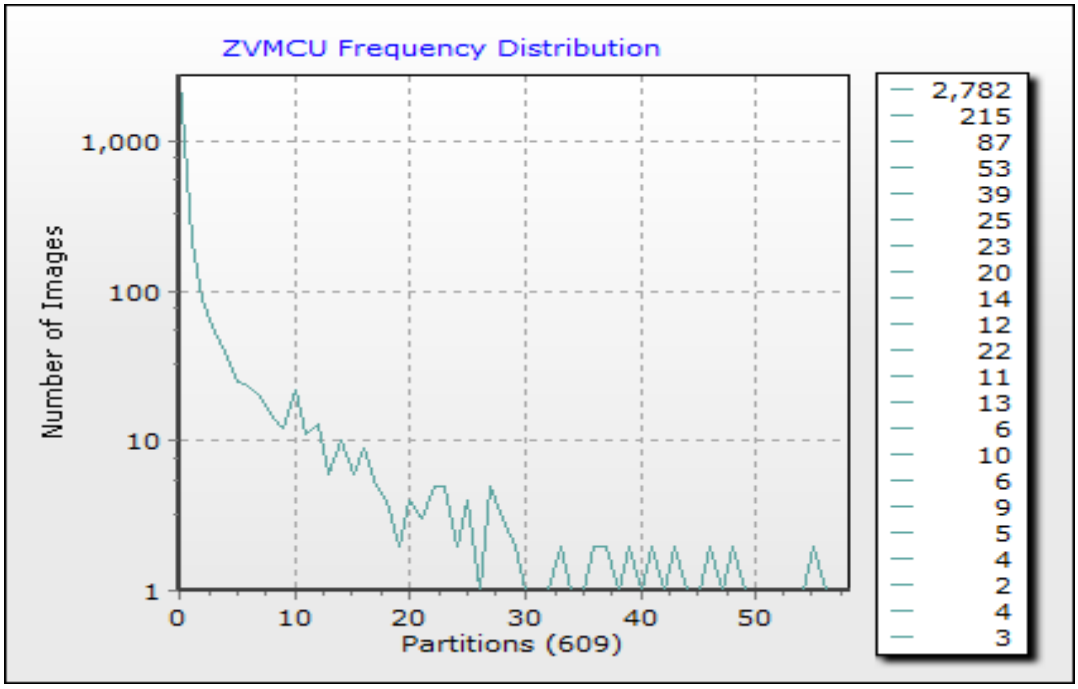


Figure 16: ZVMCU Frequency Test (StatPlus2009 v5.8)

The results from this frequency test indicates a non-normal distribution of values with 2782 images out of 3430 images containing less than 304 ZVMCUs. This distribution is expected as the ZVMCU quantity in an image is influenced by the size and resolution of the image and the distribution of size is commensurate to the distribution displayed in figure 16.

4.1.4 TEST 1 Quality Reduction by 20%

Test 1 uses image editing software to reduce the image quality of the original image by 20%. This reduction in quality is the result of the software invoking a discrete cosine transform quantization table with values necessary to reduce the quality of the image. All other JPEG configuration and compression criteria are set to negate any unwanted additional changes.

As the quantization table used for a 20% reduction in quality will have higher divisor values than the quantization table used for 100% quality, this will cause larger number of the DCT coefficients to be set to zero and the expected outcome of the reduction in quality will be an increase in the total number of ZVMCUs.

4.1.5 TEST 2 Image Rotation by 180°

This test rotates the original image 180 degrees. The image is then saved using criteria that will minimize any further influence of the JPEG compression routine. Quality remains at 100%. All sub-sampling is set to 1:1. Compression optimization is turned off. The rotation of the image will alter the starting values and the following series of values of the JPEG compression routine. With a new starting point for the compression routine, the impact on the number of ZVMCUs is stochastic; however, this test is useful and necessary to measure the changes in the quantity and position of the ZVMCUs in the image as it is possible that landmarks will endure the rotation.

4.1.6 TEST 3 Size Reduction by 20%

In test three the size of the original image is reduced by 20%. This is a reduction in the size of the image maintains quality at 100%. The result is a reduction in image resolution, though the number of bytes necessary to store the image increases. As the reduction in resolution is a reduction in the total number of pixels making up the image, the expected effect would be either no change or a reduction in the number of ZVMCUs. This reduction though not predictable or quantifiable should not change the value or location of the remaining ZVMCUs in the image.

4.1.7 TEST 4 Quality Reduction by 50%

Test four reduces the quality of the original image by 50%. This is a drastic and sometimes visible reduction in quality. This test is necessary to help scale the amount of influence changes in quality has on the number of ZVMCUs in an image. As with the prior quality reduction experiment in test one, the expected outcome is an increase in the total number ZVMCUs available in an image.

4.1.8 TEST 5 Size Increase by 20%

In test five the size of the original image is increased by 20%. This increase in the size of the image maintains quality at 100%. Using the software editing programs described earlier, when each image is saved, the software editing program GIMP is instructed to increase the images size by 20 percent of its original size. GIMP employs interpolation to increase the size of the image. The result is an increase in image resolution by 20%. The space necessary to store the file on disk will increase accordingly. As the increase in resolution is an increase in the total number of pixels making up the image, the expected effect would be either no change or an increase in the number of ZVMCUs. This increase in ZVMCUs is not predictable; however it should not change the value or location of other ZVMCUs in the image.

4.1.9 TEST 6 Restore Test1 Quality

Test six attempts to restore the quality of an image whose quality was previously reduced. Test six differs slightly from the other test. In test six, the original image is not used. Instead, the resulting image from test one (quality reduction by 20%) is used. The need to use an image with a verifiable starting point and a known reduction was

necessary to accurately measure the pure effects of the image quality restoration to 100%. As this test takes an image with previously reduced image quality and there are no methods to recover from the lossy compression routine, there should be no change in the number or location of ZVMCUs in the image.

4.1.10 TEST 7 Restore Test 4 Quality

Test seven increases the quality of an image whose quality was previously reduced. Test seven differs slightly from the other tests in the theory I experiments. In test seven, the original image is not used. Instead, the resulting image from test four (quality reduction by 50%) is used. The need to use an image with a verifiable starting point and a known reduction was necessary to accurately measure the pure effects of the image quality restoration to 100%. As this test takes an image with previously reduced image quality and there are no methods to recover from the lossy compression routine, there should be no change in the number or location of ZVMCUs in the image.

4.1.11 TEST 8 No Manipulation Save As only

Test eight does not perform any image manipulations. This test is a realistic and likely the most common cause for alterations in an images message digest. This test opens each image independently then using a save as function to save each image file with a new name and in a new directory. This action changes the message digest of the image without any changes to the image size, color, geometry or quality as performed in previous test. Appendix P is a demonstration of the changes to the MD5 signature for images between the baseline and the save as only images. (Appendix P)

4.2 Experimental Results

4.2.1 Test 1 All Image Categories

The first manipulation in the experimental series reduces the quality of the image by 20%. The results as detailed in Appendix C show that a quality reduction causes a dramatic increase in the number images with zero variance minimal computer units (ZVMCUs) present. The number of ZVMCUs per image was also dramatically increased. Figure 17 demonstrates the delta in the number of images with ZVMCUs from baseline to post manipulation. The largest delta in the pre-existing versus the manipulated image set is in category 1 while the smallest change was in image category 7.

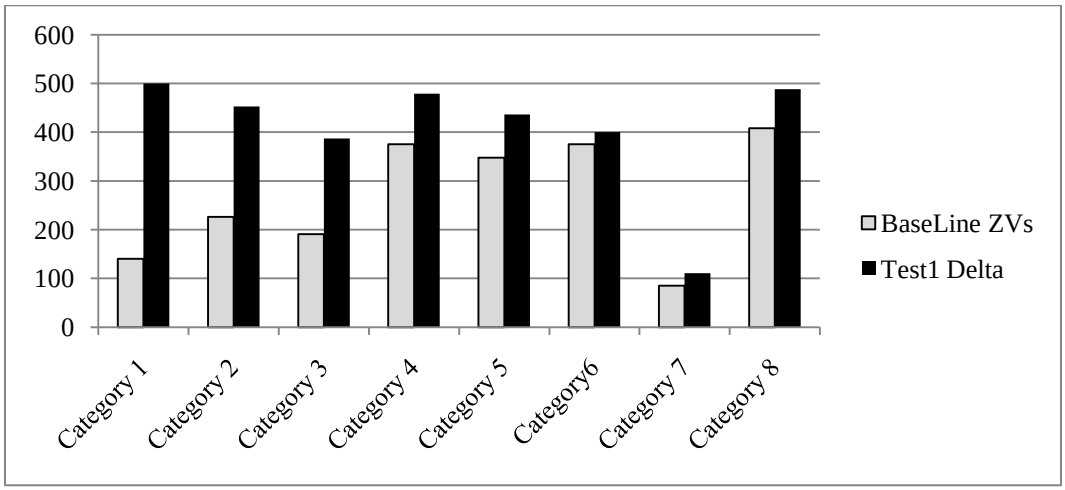


Figure 17: Test 1 ZVMCUs Delta

These finding indicate that the history of the image influences the number of pre-existing ZVMCUs as well as, the number that will be created from quality reduction. As seen in Categories 1-3, these are all known history images with no previous manipulation made to the images. The higher resolution category 1 saw the greatest increase in the number of ZVMCUs, with ZVMCUs found in every image in the Category. Image size

is also a factor in an images tendency to have ZVMCUs. Category 7 contains the smallest images in the database. As expected, with fewer pixels and overall image space, there are fewer opportunities for ZVMCUs to occur or be created. This is also evidenced by the trending of fewer overall images with ZVMCU's from Categories 1 to 3 and from Categories 4 to 6. As noted earlier, Categories 7 and 8 are special case categories.

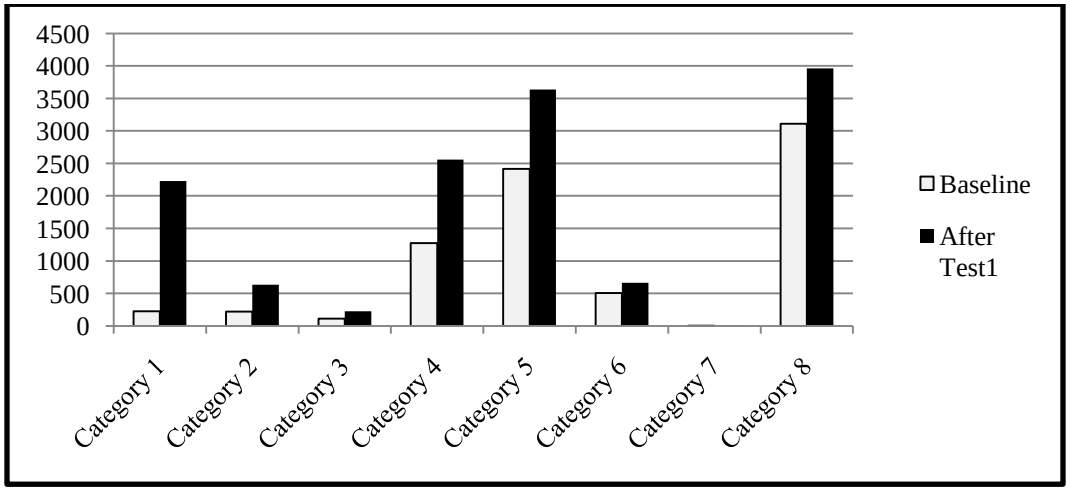


Figure 18: Average number of ZVMCUs per Image

Figure 18 demonstrates the influence of size and resolution on the average number of zero variance minimal computer units contained within an image. Quality reduction by only 20% has a great impact on the number of available ZVMCUs. The smallest average numbers of starting ZVMCUs appear in the known history images. This group averaged a total of 226, 222 and 111 ZVMCUs per image respectfully. This group also had the largest percentage increase in the average numbers, with over 100% increases in each category. This indicates a couple of points for the unknown history categories 4-6. First, based on the percentage of ZVMCUs before test 1 manipulation, it is apparent that these images had at least one, but possibly multiple editing or quality

reductions done in the past. Further, these findings tend to indicate that additional manipulation will result in continued creation of ZVMCUs in an image.

The findings of test one will be useful in building the metrics necessary to locate visually similar but digitally different JPEG files in a known dataset.

4.2.2 Test 2 All Image Categories

The second manipulation experiment consisted of rotating each image 180 degrees. Rotation is a common manipulation for images and results in modifications of both a derived MD5 hash value and the number of zero variance MCUs in an image. The results from this manipulation signify that the known history categories, which are images without previous manipulation, experience the slightest amount of deviation in the number of images with ZVMCUs. Those datasets with unknown histories, likely having been edited previously demonstrated an overall reduction in the total number of images with ZVMCUs. The average reduction in images without zero variance minimal computer units was 18.75 per image category. Figure 19 shows the deviation in the number of images with ZVMCUs from the baseline dataset. Notice that image category 7 demonstrates the greatest deviation. Category 7 is a special class of exceptionally small images, in both size and resolution. Category seven's diminutive size images limit the opportunity for a great number of ZVMCUs so any reduction will be a large percentage of the whole.

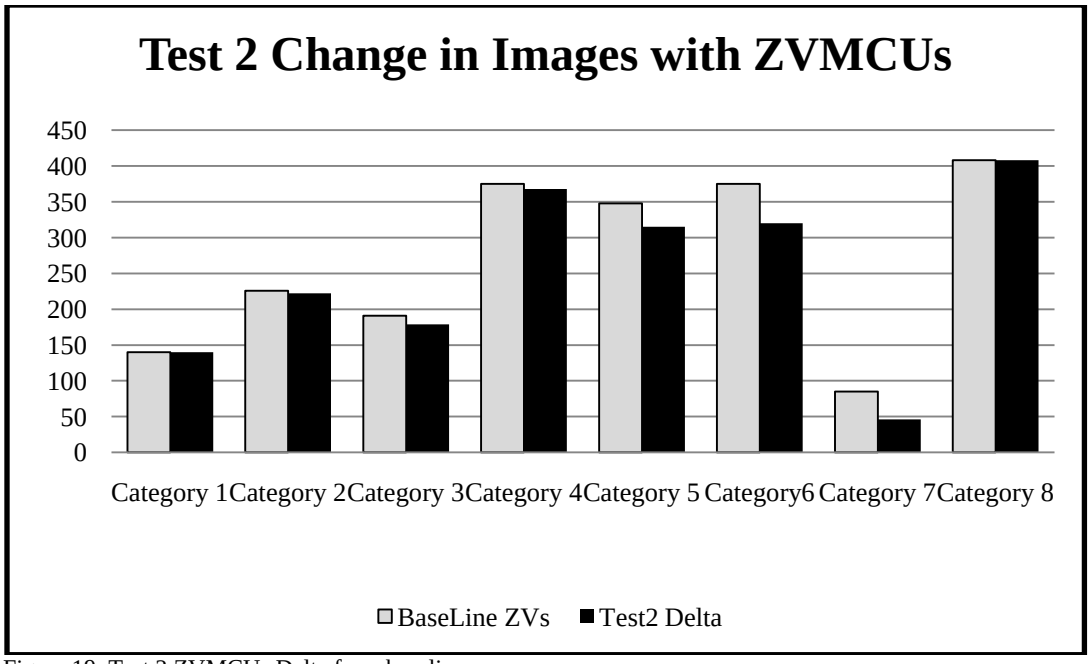
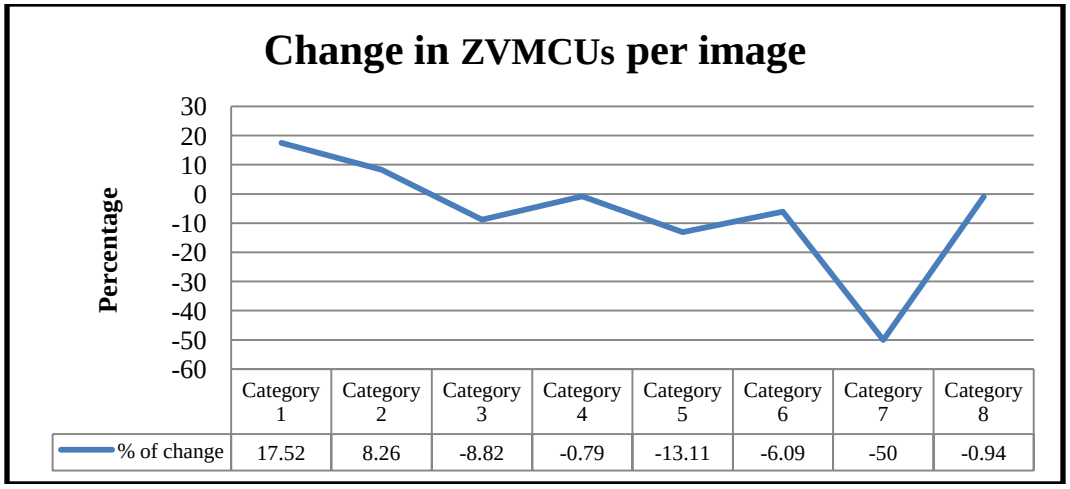


Figure 19: Test 2 ZVMCUs Delta from baseline

Of equal importance to the research is the overall number of ZVMCUs found in each image. The overall number or changes in the number per image will impact the ability to demonstrate indelible landmarks within the image. In Figure 20, the results from a rotation of the image show a generally negative impact on the average number of ZVMCUs found per image.



	Category 1	Category 2	Category 3	Category 4	Category 5	Category 6	Category 7	Category 8
% of change	17.52	8.26	-8.82	-0.79	-13.11	-6.09	-50	-0.94

Figure 20: Impact of rotation on average ZVMCUs per image

The overall average number of zero variance MCUs decreased by 6.75 %. The major deviation from the average were in Category 1, which contains the largest images in both resolution and size, this category had an overall increase in the number of ZVMCUs per image. While Category 7, containing the smallest of all the images in the database, demonstrates the greatest percentage of decrease of almost 50%, the actual decrease was from an average of 3 ZVMCUs per image to 2 ZVMCUs per image. These results show that rotation of an image does not exclude an image from having indelible landmarks identified and indicates in fact that a majority of the ZVMCUs remain intact through both the rotation and subsequent recompression of the image.

4.2.3 Test 3 All Image Categories

Test 3 in the experimental series reduces the size of the image by 20%. As seen in the detailed results in Appendix E. Reducing the size of an image exhibits an increasing reduction in the number of ZVMCUs per image as the size of the images decrease. This was consistent with both the known and unknown history images and with the special case categories 7 and 8. Of particular interest in the results from test 3 is the large percentage of decrease in the average number of zero variance MCUs found in each image. Figure 21 illustrates the reduction in the average number of ZVMCUs per image.

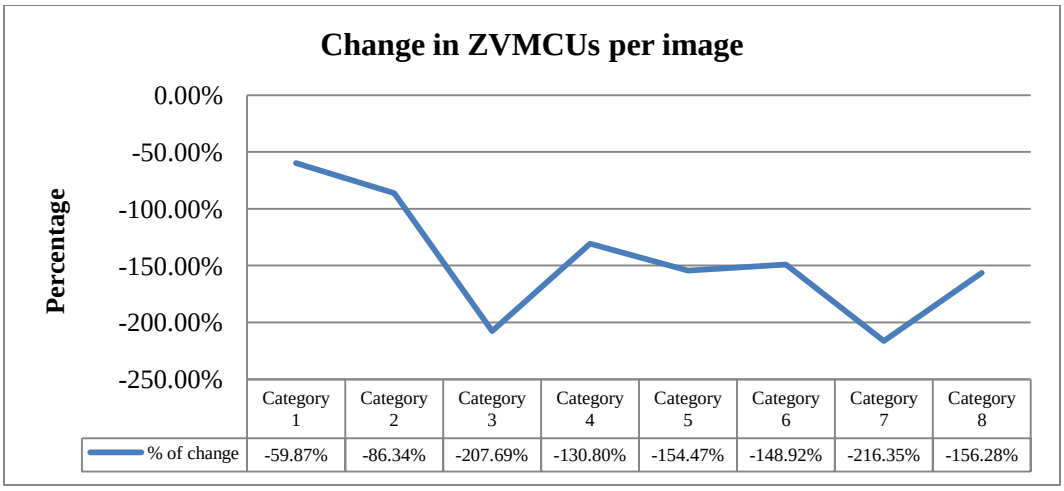


Figure 21: Test 3 impact on average number of Zero Variance MCUs per image

The results of reducing an image in size by 20%, while expected, are an area for concern for the stability of the possible landmarks found in an image. Additionally, occurrences of images with baseline ZVMCUs greater than 100 have been reduced to zero. These findings indicate that using ZVMCUs as landmarks may not be universal across all images and circumstances. Further, if the ZVMCU count falls to zero, other methods will have to be incorporated to prevent the occurrence of false negatives.

4.2.4 Test 4 All Image Categories

The impact of reducing the quality of an image was tested previously in test 1. In Test 4 the quality is reduced further, to a 50% reduction in image quality. The testing of additional reduction in quality is necessary to scale the impact of the quality reduction on images. As some JPEG software implementations will impart quality reduction unless the user specifies that the image quality is to be maintained at 100%. The results of test four are as expected and the number of ZVMCUs continues to increase. In figure 22, the percentage of images in each dataset with ZVMCUs for both test 1 and test 4 are compared against the baseline.

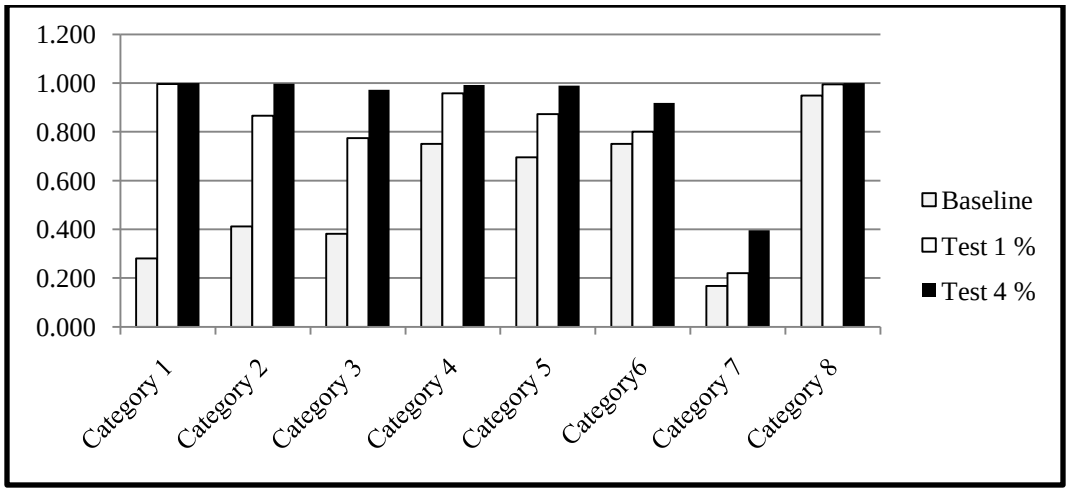


Figure 22: Percentage of images with ZVMCUs compared from test 1 & test 4

The increase in images with ZVMCUs is consistent throughout all the image categories. These findings indicate that not only are the landmarks stable through the quality reduction and subsequent recompressions, but that the available number of ZVMCUs actually increases to near 100% for all image categories. Further, these findings lend credibility to the establishment of ZVMCUs as viable landmarks for future use.

4.2.5 Test 5 All Image Categories

Test five increases the images size by 20%. As detailed in the test description, the increase in size in accomplished through software manipulation. The outcome of the increase in size is dependent on the starting size of the image, resulting in varying effects in both the total number of images with ZVMCUs and the average number of ZVMCUs per image. In figure 23 the variation of the average number of ZVMCUs per image is readily noticeable. Image Category 1, the grouping with the largest images and the greatest resolution, experiences increases in the average number of ZVMCUs per image.

Though, in figure 24 it shows that the total number of images with ZVMCUs in category 1 decreases.

While all categories experienced a decline in the total number of images with available ZVMCUs, the average number of ZVMCUs per image proved to be less predictable. The trend appears to be that the larger images experience an increase whereas the smaller images remained constant or decreased.

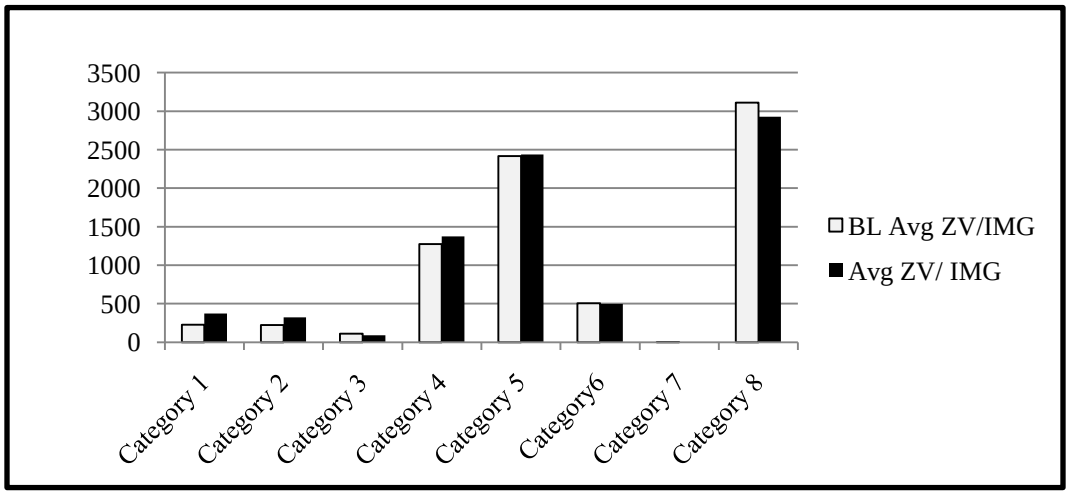


Figure 23: Test 5 Average ZVMCUs per image to Baseline data

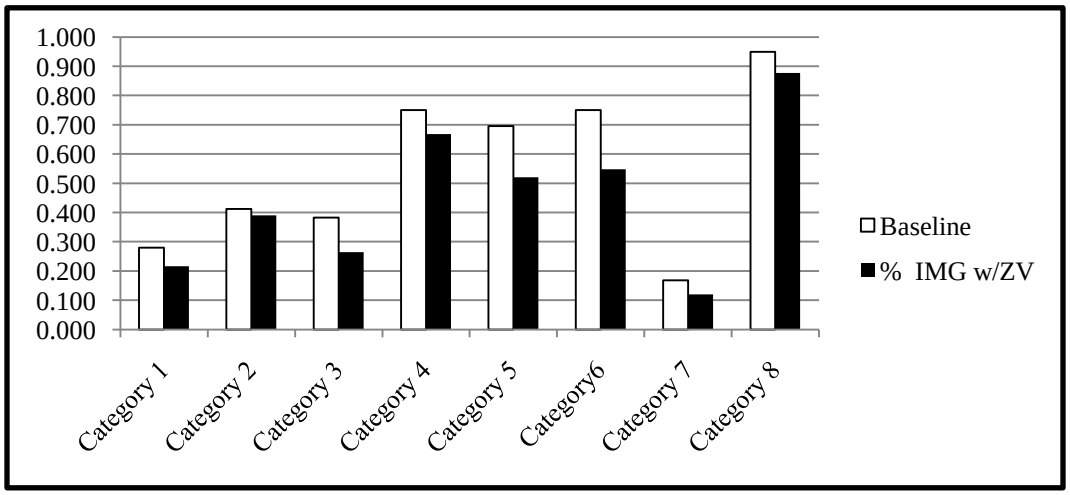


Figure 24: Test 5 Change of Percentage of Images with ZVMCUs to Baseline

Test 5 shows that while there is stability throughout all image categories, this stability is not absolute. Although, a majority of the images retain or in some cases gain ZVMCUs, the small reduction in the percentage of images with viable ZVMUs indicates a possible need for an additional method, such as the use of EXIF data for matching to help prevent false negative returns.

4.2.6 Test 6 All Image Categories

Test 6 examines the impact of saving an image whose quality has been previously reduced with a quality setting of 100%. This particular test is very useful in determining the reaction of the ZVMCUs within an image that is being repeatedly modified. For our purposes these results will be most telling in the legitimacy of the use of ZVMCUs as possible image landmarks. In the first step of this experiment, test 1, the image quality was reduced by 20%. Next, the resulting images were then opened and using the features provided in the image editing software program, the quality settings were changed to save the image at 100% quality. While the actual image quality will remain constant, the selection and storage of a quantization table will change, resulting in changes to the images message digest. Table 1 and Table 2 are the actual quantization tables used and stored with image 1, image category 2, for test one and test six respectively. Of note here is the difference in values in the quantization tables. The majority of divisor in table 1 is 40, where in table 2, all divisors are equal to 1. This means that there will be very little if any changes to the image made by the process of increasing quality.

Image 1, Category 2, Test 1
Precision=8 bits
Destination ID=1 (Chrominance)

DQT, Row #0:	7	7	10	19	40	40	40	40
DQT, Row #1:	7	8	10	26	40	40	40	40
DQT, Row #2:	10	10	22	40	40	40	40	40
DQT, Row #3:	19	26	40	40	40	40	40	40
DQT, Row #4:	40	40	40	40	40	40	40	40
DQT, Row #5:	40	40	40	40	40	40	40	40
DQT, Row #6:	40	40	40	40	40	40	40	40
DQT, Row #7:	40	40	40	40	40	40	40	40

Approx quality factor = 79.97

Table 1: Quantization table for Test 1, Image 1, and Category 2

Image 1, Category 2 Test 6
Precision=8 bits
Destination ID=1 (Chrominance)

DQT, Row #0:	1	1	1	1	1	1	1	1
DQT, Row #1:	1	1	1	1	1	1	1	1
DQT, Row #2:	1	1	1	1	1	1	1	1
DQT, Row #3:	1	1	1	1	1	1	1	1
DQT, Row #4:	1	1	1	1	1	1	1	1
DQT, Row #5:	1	1	1	1	1	1	1	1
DQT, Row #6:	1	1	1	1	1	1	1	1
DQT, Row #7:	1	1	1	1	1	1	1	1

Approx quality factor = 100.00

Table 2: Quantization table for Test 6, Image 1, and Category 2

The recorded changes for test 6 are miniscule. The maximum change in any category was less than .4% in the average number of zero variance MCUs per image as seen in figure 25. Further, there were no reductions in the number of images with ZVMCUs out of 4000 images processed.

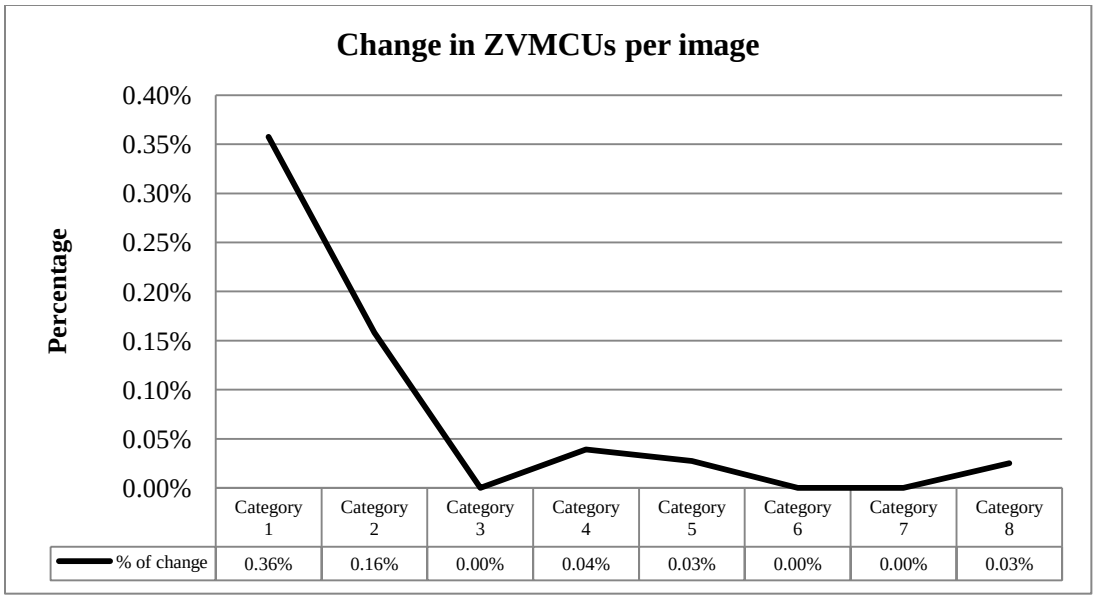


Figure 25: Change in ZVMCUs per Image Test 6.

Out of all the images processed, images in category 1 experienced the greatest change.

Of the images in category 1, image number 474 demonstrated the highest increase in the number of zero variance MCUs from test 1 to test 6 with 130 additional MCUs created. This number is less than one percent of the total. Average increases in all categories remained at less than one percent. As indicated in the earlier tests, the known history categories seem to demonstrate the greatest fluctuation from the baseline indicating that the initial editing of an image will have the greatest impact on the ZVMCU population in an image. Test 6 results favor the use of ZVMCUs as persistent landmarks for image identification and steganographic discovery.

4.2.7 Test 7 All Image Categories

Test 7 is an extreme case study on the attempted restoration of an image from a 50% quality reduction to 100% quality. The resulting implementation in the software

editing program causes a choice of vastly dissimilar quantization tables to be used in the compression routine for the image.

```
Precision=8 bits
  Destination ID=0 (Luminance)
    DQT, Row #0:  16  11  10  16  24  40  51  61
    DQT, Row #1:  12  12  14  19  26  58  60  55
    DQT, Row #2:  14  13  16  24  40  57  69  56
    DQT, Row #3:  14  17  22  29  51  87  80  62
    DQT, Row #4:  18  22  37  56  68 109 103  77
    DQT, Row #5:  24  35  55  64  81 104 113  92
    DQT, Row #6:  49  64  78  87 103 121 120 101
    DQT, Row #7:  72  92  95  98 112 100 103  99
    Approx quality factor = 50.00 (scaling=100.00 variance=0.00)

Destination ID=1 (Chrominance)
    DQT, Row #0:  17  18  24  47  99  99  99  99
    DQT, Row #1:  18  21  26  66  99  99  99  99
    DQT, Row #2:  24  26  56  99  99  99  99  99
    DQT, Row #3:  47  66  99  99  99  99  99  99
    DQT, Row #4:  99  99  99  99  99  99  99  99
    DQT, Row #5:  99  99  99  99  99  99  99  99
    DQT, Row #6:  99  99  99  99  99  99  99  99
    DQT, Row #7:  99  99  99  99  99  99  99  99
    Approx quality factor = 50.00 (scaling=100.00 variance=0.00)
```

Table 3: DCT quantization tables for Image 8, Image Category 1 Test 4

```
Destination ID=0 (Luminance)
    DQT, Row #0:   1   1   1   1   1   1   1   1
    DQT, Row #1:   1   1   1   1   1   1   1   1
    DQT, Row #2:   1   1   1   1   1   1   1   1
    DQT, Row #3:   1   1   1   1   1   1   1   1
    DQT, Row #4:   1   1   1   1   1   1   1   1
    DQT, Row #5:   1   1   1   1   1   1   1   1
    DQT, Row #6:   1   1   1   1   1   1   1   1
    DQT, Row #7:   1   1   1   1   1   1   1   1
    Approx quality factor = 100.00 (scaling=2.99 variance=6.13)

Destination ID=1 (Chrominance)
    DQT, Row #0:   1   1   1   1   1   1   1   1
    DQT, Row #1:   1   1   1   1   1   1   1   1
    DQT, Row #2:   1   1   1   1   1   1   1   1
    DQT, Row #3:   1   1   1   1   1   1   1   1
    DQT, Row #4:   1   1   1   1   1   1   1   1
    DQT, Row #5:   1   1   1   1   1   1   1   1
    DQT, Row #6:   1   1   1   1   1   1   1   1
    DQT, Row #7:   1   1   1   1   1   1   1   1
    Approx quality factor = 100.00 (scaling=1.54 variance=1.58)
```

Table 4: DCT quantization table for Image 8, Image Category 1 Test 7

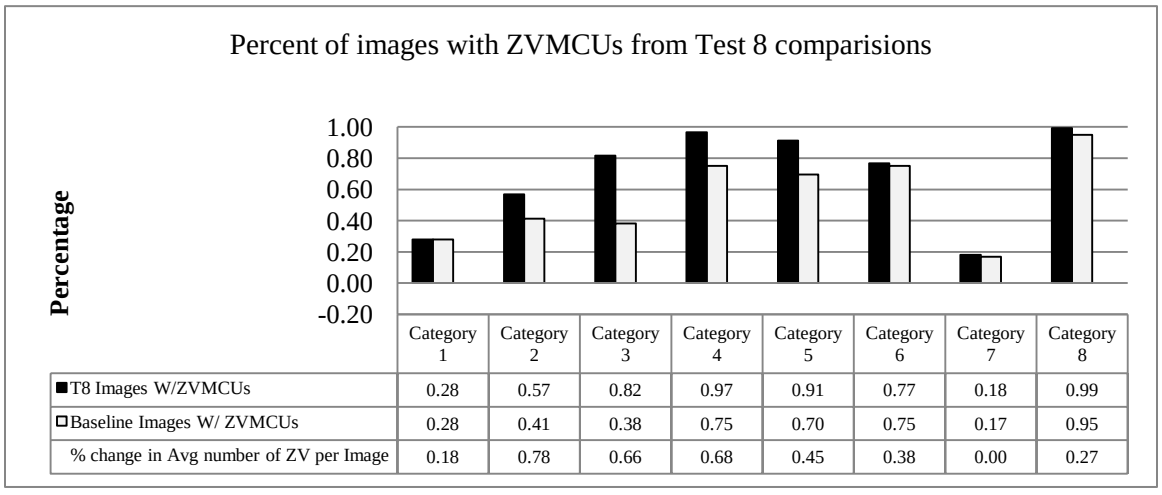
The numbers shown in Table 3 and Table 4 represent the divisors used in the JPEG compression routine when the program divides the DCT coefficients in step 2, shown in figure 1 by the quantization table. They are used again to multiply the resulting compressed values during the decompression phase.

The differences in these numbers result in changes to some of the remaining non-zero coefficients in the JPEG image. These changes in turn, cause slight variation in the number of ZVMCUs per image. So even though, the quality of the image is unchanged and all other factors remain the same, there are slight changes to the actual pixel values within an image. These changes affect not only the ZVMCU population but the MD5 hash calculations as well. Resulting in differing hash sets for what is visually exactly the same image.

4.2.8 Test 8 All Image Categories

Test 8 is the most common type of manipulation an image is likely to undergo. That is the image is opened, viewed and saved to a different location and file name. This test replicates this action. Opening each image and saving it to a new location. The results of this test show conclusively that the using ZVMCU's as a landmark criteria is a viable way to locate visually similar JPEG images files in a dataset.

In figure 26, the viability and stability of ZVMCUs is demonstrated. Every category experiences either no change or a very slight increase in the available number of images with ZVMCUs.



	Category 1	Category 2	Category 3	Category 4	Category 5	Category 6	Category 7	Category 8
T8 Images W/ZVMCUs	0.28	0.57	0.82	0.97	0.91	0.77	0.18	0.99
Baseline Images W/ ZVMCUs	0.28	0.41	0.38	0.75	0.70	0.75	0.17	0.95
% change in Avg number of ZV per Image	0.18	0.78	0.66	0.68	0.45	0.38	0.00	0.27

Figure 26: Number of Images with ZVMCUs & Average ZVMCUs per image

4.3 Theory I Conclusion

The hypothesis in the experimentation for theory I is that JPEG images possess certain landmarks that are robust enough to endure the JPEG compression routine. The results are conclusive that JPEG images do in fact possess zero variance minimal computer units and these ZVMCUs endure the more common image manipulations and subsequent image compression routine. These number of ZVMCUs found in an image are influenced by many factors. The factors shown here to influence the availability of ZVMCUs include, image size, image resolution, and image history. Further, the availability of ZVMCUs as viable landmarks is influenced by manipulation operations on the images. Some operations have a greater impact than others. Of concern to the viability of this research are those operations that have a negative impact on the population of predetermined ZVMCUs.

In test 3, image size reduction by 20%, we see a profound impact on the number of both images with ZVMCUs and the average number of ZVMCUS per image. This result is not unexpected as the default image rescaling algorithm in most image processing software relies upon some form of interpolation for the reduction or addition of pixels in an image. GIMP uses linear interpolation in its image resampling and as implemented in these experiments results in pixels with the least variance from its neighbors being removed, hence a reduction in ZVMCUs.

While test 3 demonstrates the necessity for the addition of a method to reduce false negatives, it does not discount the usefulness of ZVMCUs as viable landmarks for image identification. The preponderance of the data and analysis of the experimental results

demonstrates that the foundational theory in this research is well founded and credible for additional experimentation and research.

4.4 Landmark ZVMCUs can be used for Image Identification (Theory II)

Theory II builds upon the foundation theory proven in the first set of experiments. That JPEG images do possess certain viable landmarks, referred to as ZVMCUs. These landmarks are present in all categories of images and are relatively stable through most manipulations. The Theory II hypotheses is that the zero variance minimal computer units can be combined with non-traditional hashing methods to produce a fingerprint of a JPEG image that will be useful in locating consanguineous files in a finite dataset. It has been shown that most, if not all, steganographic embedding routines leave behind some trace elements of their operations (Fridrich, Goljan and Hogea 2003) (Fridrich and Goljan, Practical steganalysis of digital images: state of the art 2002) (Hosmer and Hyde 2003) (Goljan, Fridrich and Holotyak 2006). Most of the detection techniques rely on either discovery of a known signature or through careful analysis of first and second order statistics. (Lyu and Farid n.d.) (Kharrazi, Sencar and Memon 2005) (Provos 2001).

However, a majority of these techniques researched rely on having the original cover image as a baseline in order to extract the hidden data. This presents a problem to the forensics investigator. The problem is how to match a cover image to a possible steganographic image. Most knowledgeable users of steganalysis understand that leaving the original cover in place is equivalent to leaving your encryption key available to investigators, so finding the original image or similar images could prove a vital tool for the investigator. Additionally, an investigator would find it difficult to distinguish a cover image from any other copy of an image. The method presented here to find the

consanguineous files in a finite dataset will use non-traditional hashing techniques to locate identical ZVMCUs that can be cross referenced between images.

4.4.1 Theory II Experimental Process

Building on the results from experiments conducted in theory one, each of the now 36000[5] images are evaluated for the presence ZVMCUs. The result of each examination is a dataset that consists of the coordinates and color values of each ZVMCU in the image. The combination of the 16.7 million possible colors with the hundreds of thousands or even millions of possible coordinate locations provides a sufficient total of unique data points for image matching[6]. This dataset is used as input to the custom program ZVHashComparer.py. (Appendix M) The program is described in detail in the next section.

4.4.2 Theory II ZVHash Value Comparison Test

The ZVHashComparer.py program written specifically for this research, takes as input the results from the ZVMCUlocator.py. The program creates a unique vector for each image containing the message digest of each ZVMCU color-coordinate combination that was located by the ZVMCUlocator program. These vectors are stored using the image name. Each signature within each vector is then systematically compared to each signature of every suspect image. The results of the ZVHashcomparer is a text output of which images were matched, the number of ZVMCUs found in each image, and the number of message digest signatures that were matched. Figure 27 is a visual

[5] Each Test in theory one produced 4000 independent images (8 * 4000) + 4000 original baseline images = 36000 images each with a unique MD5 signature.

[6] 100 x100 image resolution X RGB combinations = 1.678E+11: images with 2588x2592 = 1.691E+14

representation of the ZVHashcomparer routine.

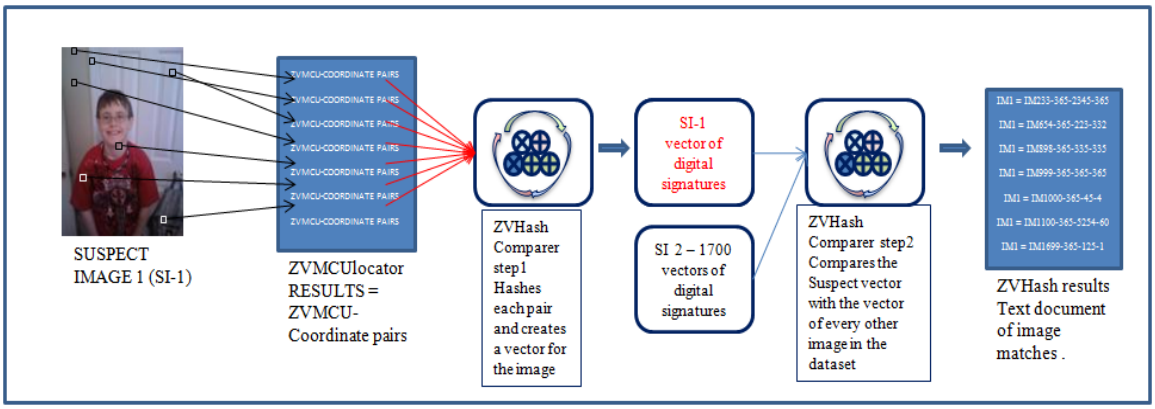


Figure 27: Visual representation of the ZVMCU Hash Comparison process

Comparisons were made between all similar images across all test sets though not all images possessed ZVMCUs so some images were not useful in the experiment. As demonstrated in the experimentation of Theory I, every image category exhibited a percentage of images that did not possess ZVMCUs. As the hashing method here depends upon the RGB-coordinate data point of ZVMCUs within an image, those images without ZVMCUs were excluded from the final results. The image category test results that follow reflect this exclusion of images without viable ZVMCUs.

4.4.3 Theory II False Positive Test

In creating a method that will locate visually similar JPEG images that share a common lineage within a finite data set it is necessary to determine the possibility of false returns. The rate of false positives is determined by recording the matches returned when comparing known dissimilar images. In this test, all 500 images from two categories were compared to all 500 images in two separate categories. Unlike the comparison test in shown in section 4.4.2, in this test every image in the category was compared to every image in the second category, at total of 10,500 comparisons. This

test was conducted twice using two sets of independent images. A total of 21,000 comparisons were made in search of false positive matches.

False negatives will be determined by extrapolating the number of matches made in test one from the number of known matches.

4.5 ZVHash Experimental Results Theory II

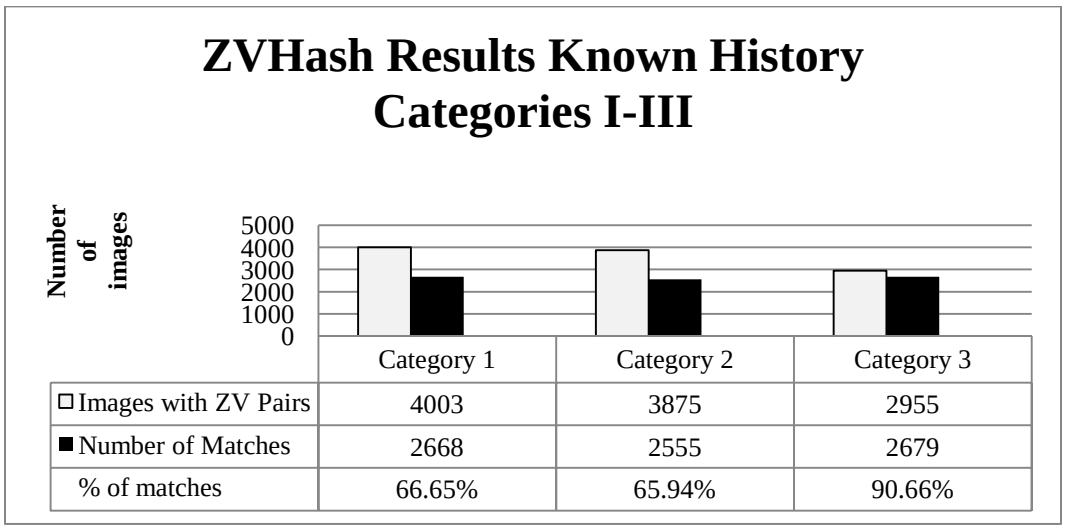
The results from the ZVHash experiment are positive and prove the viability of using ZVMCUs as a means to uniquely identify sibling JPEG image files that have different message digest signatures. Each image in each category has a possible eight matches in the database, one resulting from each test performed for theory I. The program ZVHashComparer.py conducts both the hash of the ZVMCUs contained in each image as well as a comparison of those hashes to return possible matches from the database.

For this series of experiments several factors influence the results. First, it is apparent that the pixel value saturation points need to be considered when comparing ZVHash values. Unlike the possible 16.7 million other possible color combinations, contiguous areas with ZVMCU values of either (0,0,0) or (255,255,255) are more common and might be introduced into almost any individual image. Additionally, careful review of those images with ZVMCUs of saturated values, demonstrated that these ZVMCUs tended to be located in large contiguous groupings. This is expected from an image that has either extreme under or over exposed areas. The effect of removing these saturated ZVMCUs from the hashing algorithm is detailed in 4.5.5. Another area that needed to be taken into account is images that have only a miniscule number of ZVMCUs. In repeated testing, when a floor of 5 ZVMCUs was set as a discriminator, the matching results were better by a factor of approximately 5%. However, for accuracy and consistency across

the experiments, no floor value was used in the following results. Though in future implementations, floor values should be used as a quality factor to facilitate the reduction of false negative returns.

4.5.1 Known History Categories I-III ZVHash Results

The first set of ZVHash results are from categories 1-3. These are the known history categories containing images that had not been previously altered by photo editing software. The resulting images from each of the manipulation test conducted in theory one were used to find matches throughout the database of images. The results demonstrated in Figure 28 show that in the majority of cases sibling images can be located using the non-traditional hashing technique implemented in ZVHashCompare.py.



	Category 1	Category 2	Category 3
Images with ZV Pairs	4003	3875	2955
Number of Matches	2668	2555	2679
% of matches	66.65%	65.94%	90.66%

Figure 28: ZVHash Results Categories I-III

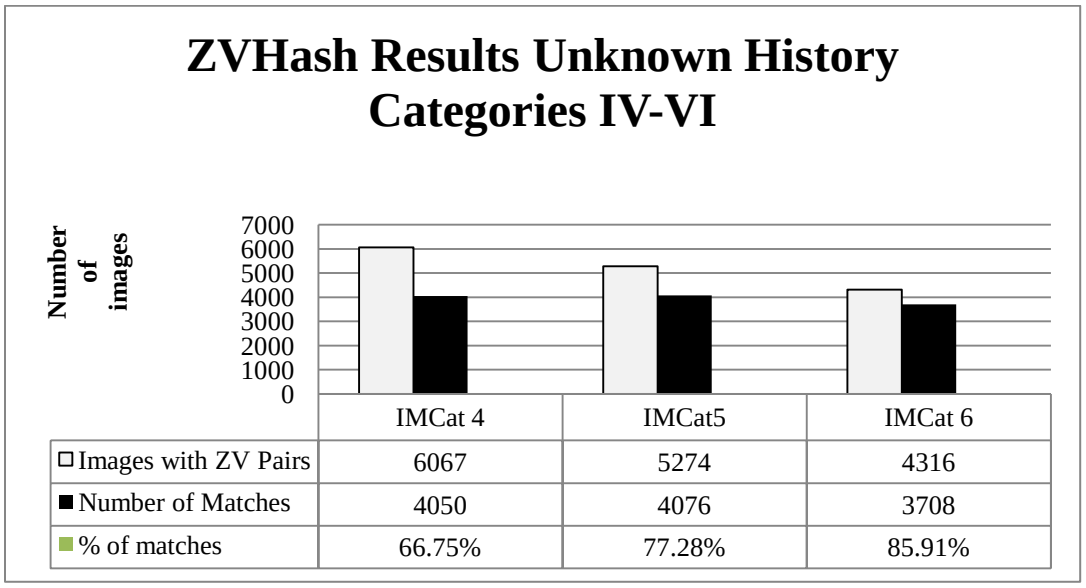
The percentage of matches found demonstrate that using a non-traditional hash technique of ZVMCU value/coordinate pairs is a viable means for locating sibling JPEG files. These image categories began with a known history of no prior editing, all manipulations to these image categories was done during the research. The match rate of

the known history images is analogous with the match rate returned for the unknown history images. These images have each experienced just one manipulation, with the baseline images experiencing zero manipulations. This accounts for the difference in the overall images with viable ZV pairs when compared to the unknown history images. The results from this test indicate that the ZVHash algorithm is equally useful as an image detection mechanism for images that have not been altered as it is for images that have had prior manipulations.

4.5.2 Unknown History Categories IV-VI ZVHash Results

The results from this experiment track closely with the results from the previous categories. The main difference in the results is in the total number of image pairs that were determined to be viable. The unknown history category found almost 2000 more images per category as viable versus the known history categories. As mentioned earlier, this is a result of the total number of available ZVMCU's in each image. The general trend in both the unknown and known history categories seems as image size and resolution decreases the percentage of matches increase. This is contrasted by the fact that the larger images returned a greater number of viable image pairs when compared to smaller image categories. This trend can be seen in Figure 29, where the number of viable image pairs decrease by approximately 1000 images across the categories, but the number of matches increase by almost 10% over the same range. This change across categories is the result of fewer independent and isolated ZVMCUs in the images, resulting from the smaller image size and reduced number of possible pixel locations. This is reinforced in the findings of special case category 8. As discussed in 4.5.1, setting

floor values will help improve the rate of matching by eliminating those images with less than 5 viable ZVMCUs.



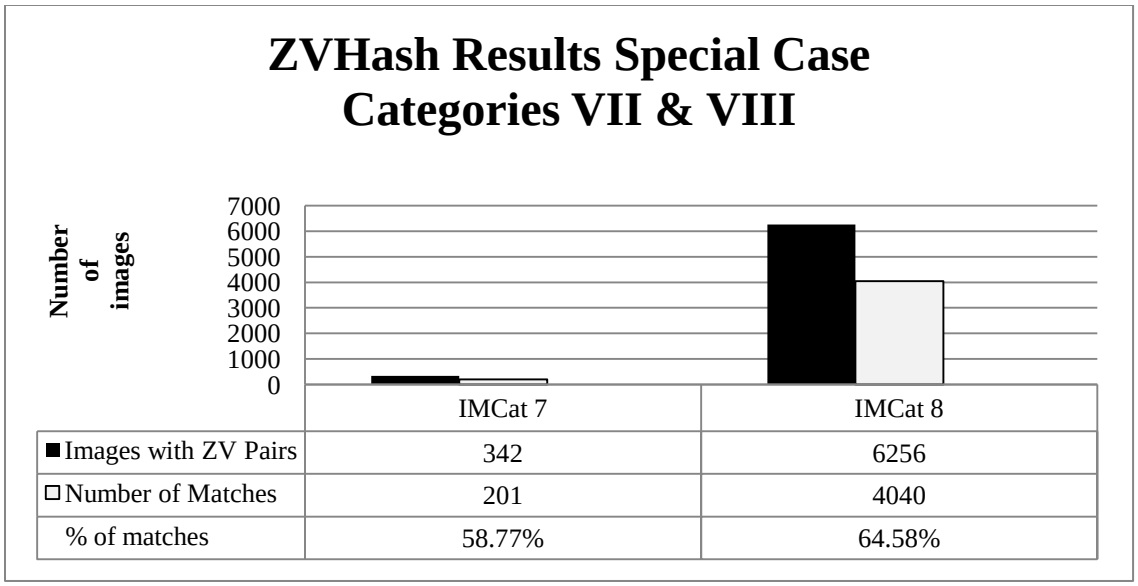
	IMCat 4	IMCat5	IMCat 6
Images with ZV Pairs	6067	5274	4316
Number of Matches	4050	4076	3708
% of matches	66.75%	77.28%	85.91%

Figure 29: ZVHash Results for Unknown History categories

4.5.3 Special Case Categories VII-VIII ZVHash Results

These two categories of images are included mainly for completeness and consistency of the research. Of most interest in these results is that the percentage of matches remains fairly consistent with all the other categories of images. Category VIII's results confirm the findings in the previous experiment that image resolution impacts the likelihood of matching ZVMCU hashes in an image. All the images in category VIII are less than one megabyte in size, which if size was a factor would put the match results on par with image categories III and VI, however, the results are more closely aligned with those of categories I and IV. Categories I and IV, like category VIII have resolutions

greater than 4 megapixels and all three categories have similar returns in the percentage of matches at approximately 65%.



	IMCat 7	IMCat 8
Images with ZV Pairs	342	6256
Number of Matches	201	4040
% of matches	58.77%	64.58%

Figure 30: ZVHash results for special cases

Category VII demonstrates a particularly interesting result. As seen in Figure 30, category VII maintains a match percentage that is generally equivalent to all the other image categories this is in spite of the diminutive size of the images and the extremely small number of actually useable image pairs. These results, while included for completeness and consistency, support the use of ZVMCU's for image matching.

4.5.4 False Positive Experimental Results

A false positive return in this collection of experiments is defined as there being at least one ZVHash match between the two dissimilar images. The two categories of images compared are known to have no sibling or likely visually similar files between them. The expected false positive rate is zero. In reality, the false return rate is 3.8% for the photographic images in categories 3 and 5. An image that has zero ZVMCUs is not a viable image for comparison so those images are not factored into the equations. The first false positive test used categories three and five. As found in theory one and demonstrated in figures 18 and 24, 38% of category three and 69% category five contained images with viable ZVMCUs, at baseline. This percentage changed as the result of the differing manipulations. The average viable ZVMCUs resulting from all tests are 59% and 75%, for categories 3 and 5 respectively. The combined results of all the manipulations are experienced in the false positive results. The percentage of images with viable ZVMCUs found in this experiment is 53%, with 1751 viable images being used out of a possible 3314. The number of false positive matches from the viable population of 1751 images is only 35 or just 1.9% of the population. Figure 31 presents a graphical representation of these results.

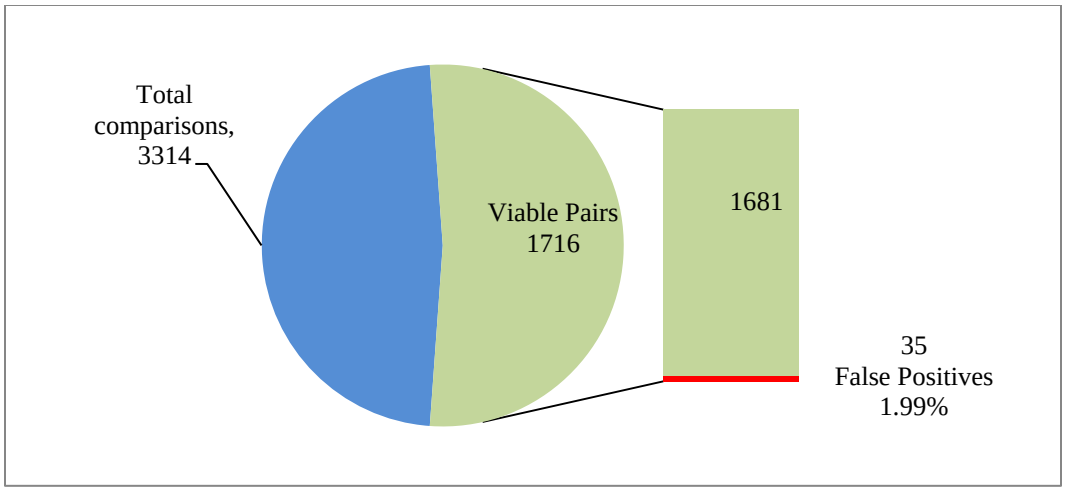


Figure 31: False Positive test one Categories 3-5

This percentage of false positives is not exceedingly high, but for the purpose of image discrimination of large data sets the actual number of false images returned is still too large. A closer examination of the actual image pairs returned as matches demonstrate that image over-exposure or under-exposed is a leading factor in the number of false returns, Figure 32. Therefore, modifications to the ZVHash algorithm that excludes pixel saturation points should reduce the number of false positives and bring the false positive return rate into a more acceptable range.

Figure 32: Demonstration of ZVMCU value saturation impact on false positives

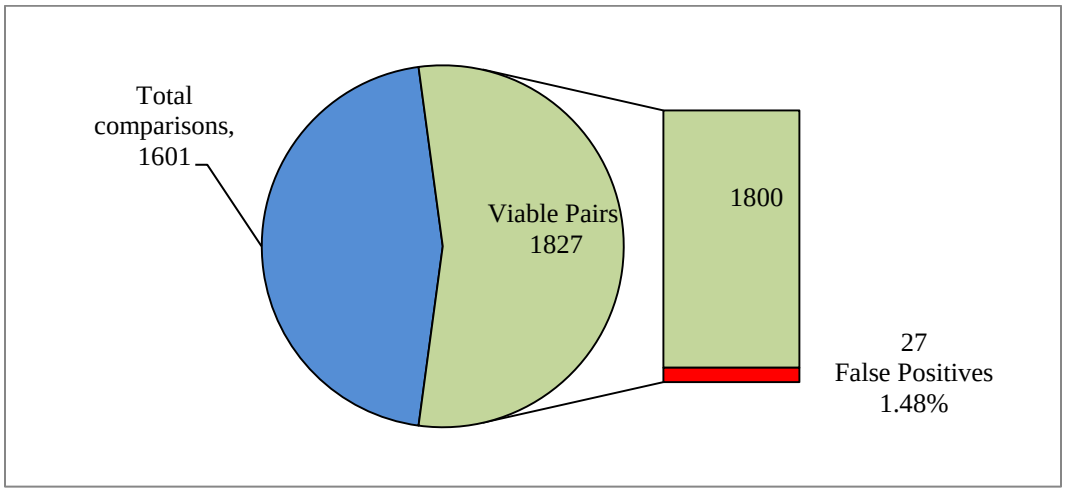


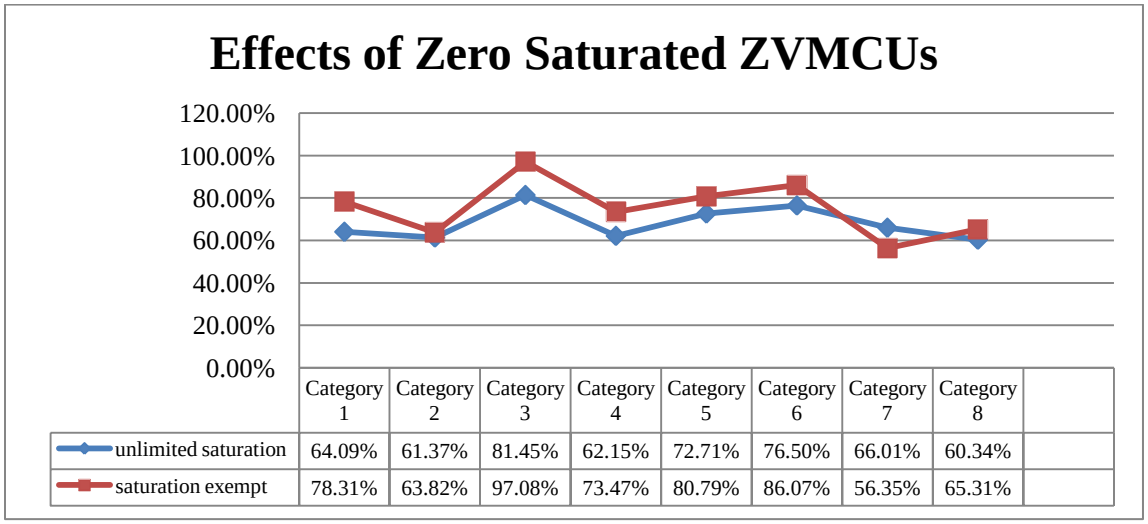
Figure 33: False Positives test two Categories 2-6

In figure 33 the results of false positive test run on categories 2 and 6. These categories are chosen because, as in categories 3 and 5, they have zero probability of containing any possible matches or even likely visually similar images. Category 2 contains images taken by the author and downloaded directly to the computer, where category 6 contains images downloaded from the Internet. These results reinforce the finding in the previous false positive test conducted with categories 3 and 5.

4.5.5 Saturation Exempt ZVHash Results

As discussed in the previous findings, when JPEG files have large areas that are either under-exposed (255,255,255) or over exposed (0, 0, 0), there is an increased probability for false matches between otherwise dissimilar files. In order to account for this the ZVHash algorithm was modified to include the ability to indicate the amount of acceptable pixel saturation within an image. The ability to set different saturation points allows the user the ability to select a level of acceptable false positive and false negatives as may be needed for a particular data set. As seen in these results, reducing the number

of saturated ZVMCUs from unlimited, demonstrated in 4.5.1, to a setting that disregards all ZVMCUs with saturated color values improved the match rate by as much as 16%. (Figure 34)



	Category 1	Category 2	Category 3	Category 4	Category 5	Category 6	Category 7	Category 8
unlimited saturation	64.09%	61.37%	81.45%	62.15%	72.71%	76.50%	66.01%	60.34%
saturation exempt	78.31%	63.82%	97.08%	73.47%	80.79%	86.07%	56.35%	65.31%

Figure 34: Effects of Zero Saturated ZVMCUs on match rates

4.5.6 Saturation Adjusted False Positive ZVHash Results

The real improvements to the algorithm and to the overall stability of theory two is found in including a saturation setting in the reduction of false positives. Figure 35 shows a dramatic decline in the false positive rate when even a small portion of the saturated ZVMCUs are excluded from being matched.

It is evident that for any practicable use of ZVMCUS as landmarks in identifying consanguineous JPEG files in a finite dataset, saturation of the ZVMCUs will have to be taken into account. The best method for this is to allow for an on the fly adjustment of the number of saturated ZVMCUs dependent upon the needs of the investigator and the dataset that is being examined.

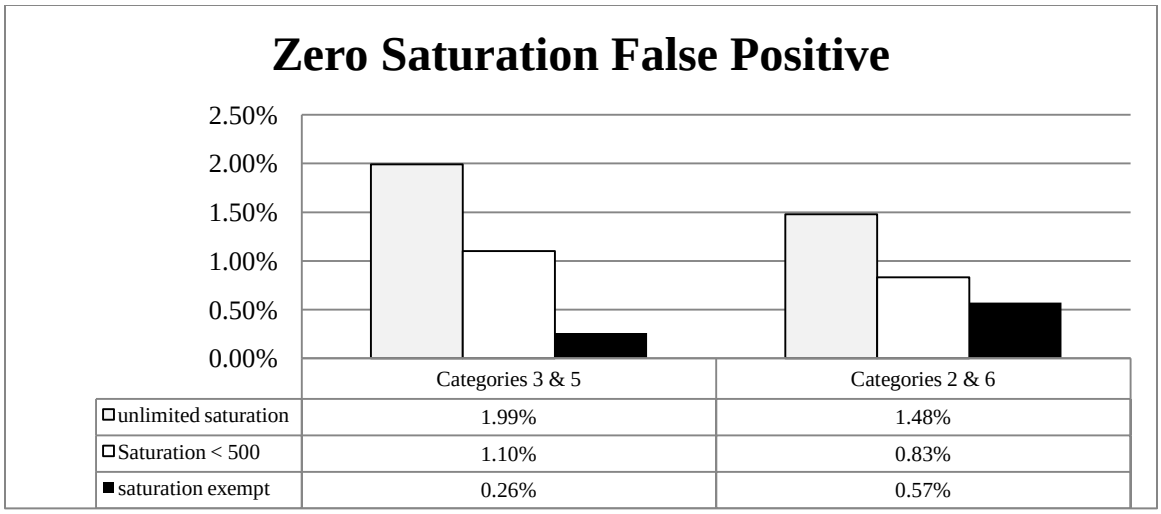


Figure 35: Saturation settings effect on false positives

4.5.7 Realistic Test Condition Results

To illustrate the actual results possible with the ZVHash method. This section presents the results from a simulated real world scenario. In this experiment 16 images were selected from the test results database, two from each image category based upon the overall average ZVMCUs for that image category. The 16 images represent the suspect images the investigator is attempting to locate in a large dataset. As described in the initial problem statement, a major challenge for investigators is locating visually similar but digitally different images in a finite dataset. This experiment replicates that problem by searching a dataset of 4000 images for just 16 sibling JPEG image files. The search was conducted using the ZMHashcomparer.py to first create a ZVHash of each of the suspect images. These ZVHash results were then compared to the ZVHash results of each of the 4000 images in the dataset. The following results show that the ZVHash method is a viable means for image identification. This ZVHash method correctly identified all 16 sibling JPEG image files in the dataset of 4000. As detailed in Table 5,

the average false positive rate is just 0.32% and a maximum of 71 false images were returned. The average number of false positive images returned from the database of 4000 is just 14; this number is easily manageable in a human visual review of the images. However, a visual review may not be necessary if a further qualifier, such as EXIF data or first order statistics is added to the selection process.

As seen in table 5, the average rate of matches of individual ZVHash results between the similar images in the dataset is consistently greater than 95%, with an average matching rate of 99.47%. False positive returns could be further reduced when the match rate of individual ZVHash results is added to the routine as selection criteria. In this example, if the match rate was used as a criterion to exclude images that did not have a match rate of over 90%, then those false returns with less than a 90% would be excluded. The MD5 hashes of each of the images are included to illustrate two very important points. First, the MD5 hashes demonstrate that the sibling JPEG image pairs have different traditional hashes. Second, it reconfirms that using traditional implementation of the MD5 routine will return zero matches from the database.

Suspect Image number	Matches out of 4000	False Positive / 4000	Database Image number	Datebase Image ZVMCUs	Suspect Im ZVMCUs	Matched ZVMCUs	% of ZVMCUs Matched	MD5Hash (original)	MD5Hash (Suspect)	Matched MD5's
1	1	0.00	Ct6Im15	504	1196	504	1.0000	66c8d3d9a7eff	c5d515a092e	0
2	1	0.00	Ct7Im25	3	8	3	1.0000	33ddf2eb3fcd8	4985e8bd1d1	0
3	1	0.00	Ct5Im26	2313	8782	2313	1.0000	328168ef71fal	232afd32e59	0
4	72	0.02	Ct7Im28	6	18	6	1.0000	59d9c721b32c	19ad1d22ab7	0
5	1	0.00	Ct6Im35	593	1186	593	1.0000	49bcb6bc0a53	124484ff2b5	0
6	1	0.00	Ct8Im38	2959	6234	2959	1.0000	ae20687f0c4d0	d972df4f79c	0
7	15	0.00	Ct1Im43	217	219	216	0.9954	6c2d8c06882b	b218491536	0
8	29	0.01	Ct2Im45	210	253	200	0.9524	efbb7c24454ee	8885eccd13d	0
9	38	0.01	Ct5Im49	2433	2433	2433	1.0000	dfe0a062cfd5f	6ecbd297900	0
10	11	0.00	Ct4Im51	1129	1288	1127	0.9982	03c5bfb307a1	4519bb76827	0
11	1	0.00	Ct8Im52	3822	4933	3750	0.9812	fd28b087747d	10b066d35c2	0
12	1	0.00	Ct4Im66	1168	1656	1163	0.9957	7d03153b3bc7	e8d382478d6	0
13	10	0.00	Ct1Im79	280	283	278	0.9929	369951b248d7	e8db6ecd7c2	0
14	1	0.00	Ct3Im82	95	159	95	1.0000	c74ccd62a458	bad0668519a	0
15	1	0.00	Ct3Im87	162	184	162	1.0000	e04d3df36d99	f5d67037a26	0
16	39	0.01	Ct2Im106	341	371	341	1.0000	35964e3680af	a1921de2277	0
Averages	13.938	0.32%					0.9947			0

Table 5: Real World Example Test Results for 16 Suspect Images

4.6 Theory II Conclusion

The findings in this phase of the experimentation is conclusive proof of the viability of using the ZVHash method as a means to identify JPEG images that were at one time digitally identical, but now differ in their digital construction, such that traditional hash routines are ineffective.

The average match rate established in sections 4.5.1, 4.5.2 and 4.5.3 of 72.06% [7] is acceptable considering the images have endured multiple and contrasting manipulations that greatly affect the viability of the ZVMCUs and further change their digital and visual similarity to the original. Likewise, the results of testing for false positive returns were very encouraging. Although the unconstrained returns were as high as 1.99% the

[7] All image category match rates detailed in figures 16, 17 & 18 divided by eight.

ZVHash still served to reduce the possible population of possible images by 98% and reduced the actual number of images to a visually manageable number. However, it was proven in section 4.5.6 that the false positive return rate can be reduced to less than 1% of the population when saturated ZVMCUs are accounted for in the search algorithm. The most influential findings of the effectiveness of the ZVHash and its contributions to digital forensics are illustrated in 4.5.7. This experiment replicates a possible real world problem searching for possible matches of 16 JPEG image files in a dataset of 4000. ZVHash matched 100% of the suspect image files with zero false negatives. The false positive returns from the unconstrained ZVHash were just 0.32% with an average of just 14 misidentified matches per category. As explained in the detailed results, a constrained version that accounts for ZVMCU saturation and incorporates a metric for individual ZVHash match percentages will reduce this number further. These results confirm the original theory established for this section, that using a non-traditional hash method on zero variance minimal computer units located in JPEG image files is a practical and viable means to locate consanguineous JPEG images files in a finite dataset.

4.7 ZVHash can be used to Identify Steganographic Images (Theory III)

The final theory in this research is the most complex and complicated. It brings together the findings of the previous experiments and introduces the unknown effects of several different steganographic algorithms. Theory III is that JPEG steganographic images can be identified in a finite dataset using the ZVMCUs and the ZVHash routine. The theory is based on the findings of the persistence of the ZVMCUs and on the match rates returned by ZVHash on known pairs of JPEG image files. It is believed that the

steganographic embedding routines that exploit the transform domain of the JPEG compression algorithm will perturb the ZVHash match rate resulting in an identifiable anomaly in the dataset.

4.7.1 Theory III Experimental Process

This experiment will take a set of known cover images (CI) created in experiment I, a set of known secret messages (SM) and three known steganographic embedding routines (SP) and create a dataset of steganographic images (SI) that will be the basis for the final experiment. Each steganographic image will be evaluated for identifiable landmarks using the ZVHash routine detailed in theory II. The creation of the steganographic dataset involves three basic steps, selection of the cover image, selection of the secret message, and embedding using the selected software package.

The selection of the steganographic program is determined by the programs ability to use JPEG images as containers and to save the steganographic image in the JPEG image format.[8] The three programs chosen, JPhide and Seek (JPHS), PJSteg, and Steghide, are all capable of creating JPEG steganographic images and are similar in design but differ in implementation of their algorithms.

JPhide & Seek (JPHS) uses a cryptographic algorithm called Blowfish. Incorporating the users' passphrase and a pseudo-random number generator, the algorithm determines how to embed the secret message in the transform domain of the compression routine. The program introduces some noise into the cover image and with

[8] Not all image based steganographic programs are capable of creating JPEG steganographic images. Some, like Cryptapix and Secure Engine, can use JPEG's as a container but will create the steganographic image in a non-lossy format like BMP.

the matching original image there are programs available to detect JPHS. This program is of particular interest because of its easy to use Windows graphical users interface. The ease that a person can download and employ JPHS makes it a very viable tool for anyone looking to conceal communications or digital content.

The second steganographic tool chosen is PJSteg. This is a windows based JPEG steganography tool that includes an easy to use graphical user interface. This program stores the embedded data in the least significant bits of the JPEG image. While statistically detectable if the embedded data is too large, this is one of the few programs that can store the steganographic image in the JPEG format.

The third software selection is Steghide. Steghide is a UNIX based, command line JPEG steganography program. As outlined in the Steghide manual pages, Steghide uses a graph-theoretic approach to Steganography. The algorithm works by first compressing and encrypting the secret data. Locations of pixels where the secret data will be stored are selected using a pseudo-random number generator that is seeded by the user password. First, pixel values that match the embedding secret data are sorted out and used. Then the remaining data is embedded across the image data. The program claims that the images' first order statistics are maintained as a result of the embedding routine.

The cover images used in this portion of the research were selected using three criteria, image size, image history, and average ZVMCU count. In all 48 images were selected as cover images from the database. Six images from each category where selected based upon their average number of available ZVMCUs. By selecting cover

images from each category, a cross representation of the database is achieved and an equal number from each size and history category were selected. Although the characteristics of the images in category 7 make them ill suited for use as cover images, six cover images from this category was included in the experiments.

As noted earlier, the size of the hidden data impacts the steganographic images probability of detection. Larger embedded files perturb more of the images data so as the embedded file size increases so does the likelihood of introducing detectable changes. The hidden data for this portion of the experiments were all text based files that allowed for precise manipulation of the file size. The selection criteria for the files to be used as hidden data was based upon the reports from JPHS on the maximum data size its embedding routine would allow as seen in table 6. The text files that were used as hidden data were created using Notepad and saved in various sizes from 50 bytes up to 1024 Kilobytes allowing the best possible match to the storage capabilities of the individual programs. The hidden data files were characterized into three groups, large, medium and small based upon the JPHS reported embedding capacity. Large files are greater than 70% of the maximum embedding size. Medium hidden data files are approximately 50% of the maximum embedding size and small hidden data files are less than 30% of the maximum.

	Hidden Data Sizes for Each Category in Kilobytes			
Image Category 1	530.00	371.00	265.00	159.00
Image Category 2	135.00	94.50	67.50	40.50
Image Category 3	14.33	10.03	7.17	4.30
Image Category 4	258.83	181.18	129.42	77.65
Image Category 5	73.67	51.57	36.83	22.10
Image Category 6	25.00	17.50	12.50	7.50
Image Category 7	2.93	2.05	1.47	0.73
Image Category 8	102.50	71.75	51.25	30.75
	Avg Maximum	large>70%	medium ~50%	small <30%

Table 6: Size chart for hidden data embedded in cover images.

4.7.2 Steganographic Embedding Experimental Results

Using each of the 3 steganographic embedding programs each of the 48 cover images was embedded with a small text file, a medium text file and a large text file, creating 144 individual steganographic JPEG files for each steganographic program. In all, a total of 432 individual steganographic images were created using the three embedding programs.

Each of these 432 steganographic images were then evaluated as detailed in Theory II, for the quantity of ZVMCUs present per image and for total number of images with viable ZVMCUs.

A set of ZVHash results was created for each of the steganographic images. The ZVHash of each of the 432 steganographic images was then compared to the ZVHash of each of the 48 cover images for possible matches and possible anomalies.

As seen in Figure 36, the ZVHash routine proved to be a viable means for identification of consanguineous image files, even if those files have been manipulated with steganographic software. The ideal return would have been 48 matches for each of the size categories in each of the software programs. The ZVHash routine found 100% of

the image files pairs in the dataset. However, as seen in the previous experiments, there are a number of false positives that are present when the image saturation points are not taken into account. Figure 36 details the results for all three embedding routines and demonstrates viability of the ZVHash as an image matching algorithm. The false positive results varied from 67 to 73, per category. In figure 37, the results of removing pixel saturation is demonstrated. It is clear that accounting for the saturation points is necessary and useful means to control the number of false positives.

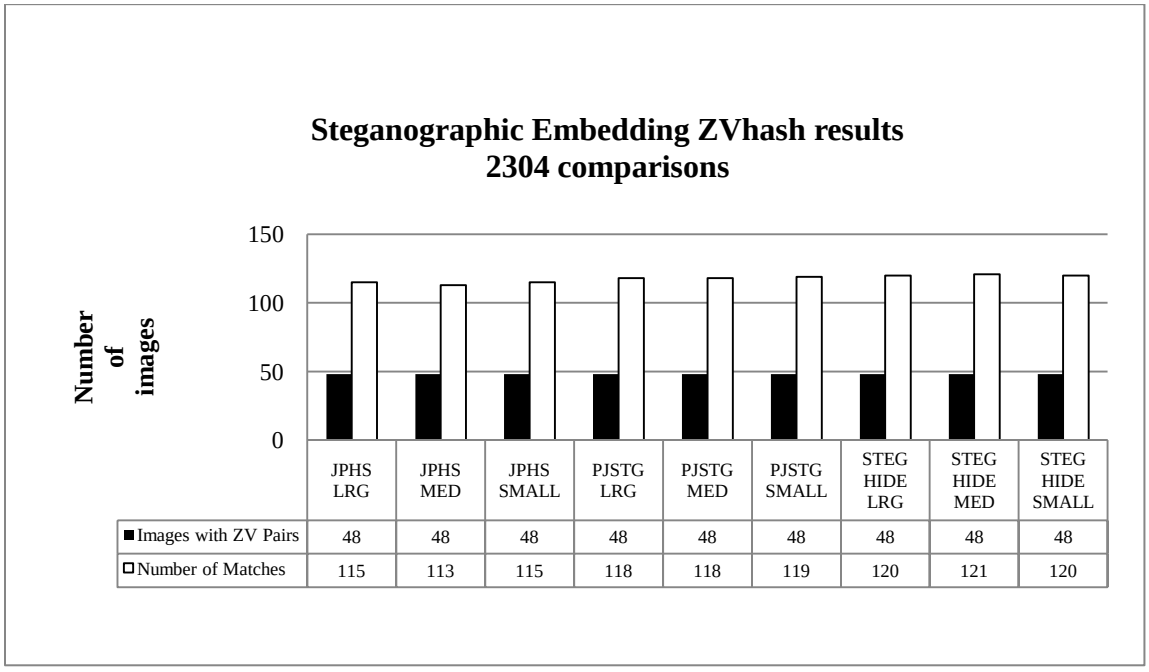


Figure 36: Steganographic Routine embedding results

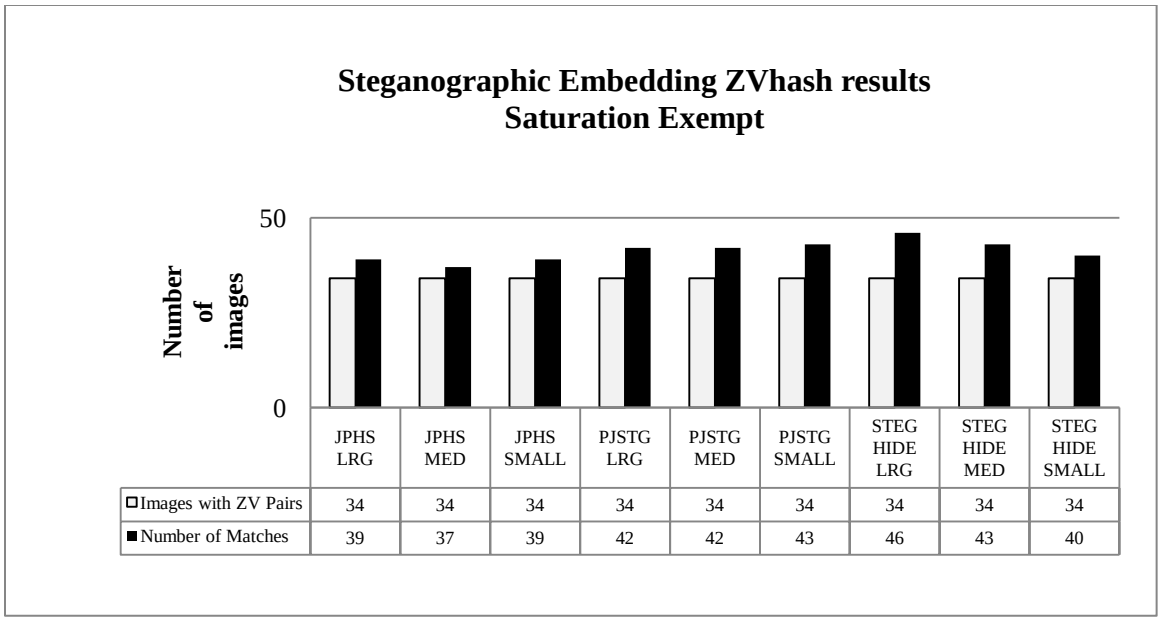


Figure 37: Saturation Exempt ZVHash matches for Steganographic images

4.7.3 Test 8 to Steganographic Embedding Routine Examinations

In looking for anomalies in the steganographic embedding routines it is necessary to have a solid base line for comparison. In building this baseline, test 8 was chosen as the model because the save as method provides the most predictable and repeatable results of all the manipulation tests. For this experiment, four software editing tools, Microsoft Photo Manager, Photoscape, GIMP and IRFANVIEW, were used to conduct the 'save as' method on all 500 images from Category 8. Category 8 was chosen because it provides a mixture of known history and unknown history images of a variety of sizes. Also, Category 8 provides the most images with viable ZVMCUs for experimentation. After the save as operations were complete, the next step is to verify if the photo editing software packages altered each of the 500 images digital signatures. This was accomplished by comparing the digital signatures of the base line images from category 8 to each of the 1,500 post save images. (Appendix O). The next step was to verify that the digital signatures between each of the separate save as procedures were

identical. This is to verify that there were no changes from one implementation to the next across the 3 tests for each software package. (Appendix O) These two procedures were necessary to verify the baseline comparisons were similar to the steganographic embedding routine comparisons detailed in section 4.7.2.

4.7.4 Steganographic Embedding Routine Anomalies

A careful analysis of each of the individual matches from the ZVHash routine has indicated some small anomalies in the steganographic embedding routine that are not present in the similar testing done in Theory II. In Test 8, the test where an image was not manipulated but simply saved as a new name, the ZVMCUs average value changed across the images. In the steganographic image pairs the average ZVMCU value remained constant across the embedding routines.

Further, the number of ZVMCUs present in each image fluctuated between the individual manipulations in the other 7 tests, as well as, in test 8. However, for the steganographic embedding programs, the number of ZVMCUs found in each image remained unchanged between the steganographic image and the cover image. This difference between normal manipulations and steganographic embedding routine suggest an avenue for the detection of steganographic images that are embedded in the transform domain of JPEG images. However, as seen in the following testing the fluctuations in the ZVMCU average value for the 8 manipulation tests proofed to be statistically insignificant.

4.4.5 Statistical Testing of Variance of ZVMCUs for Predictable Anomalies

The population of the ZVMCUs for the baseline images (control group), the steganographic image (test group 1) and the save-as images (test group2) all have non-normal distributions (Figure 38). The distribution of the ZVMCU quantity found in the test set of images as seen in Figure 38 matches the distribution of the size variable for the same set of images. (Figure 39)

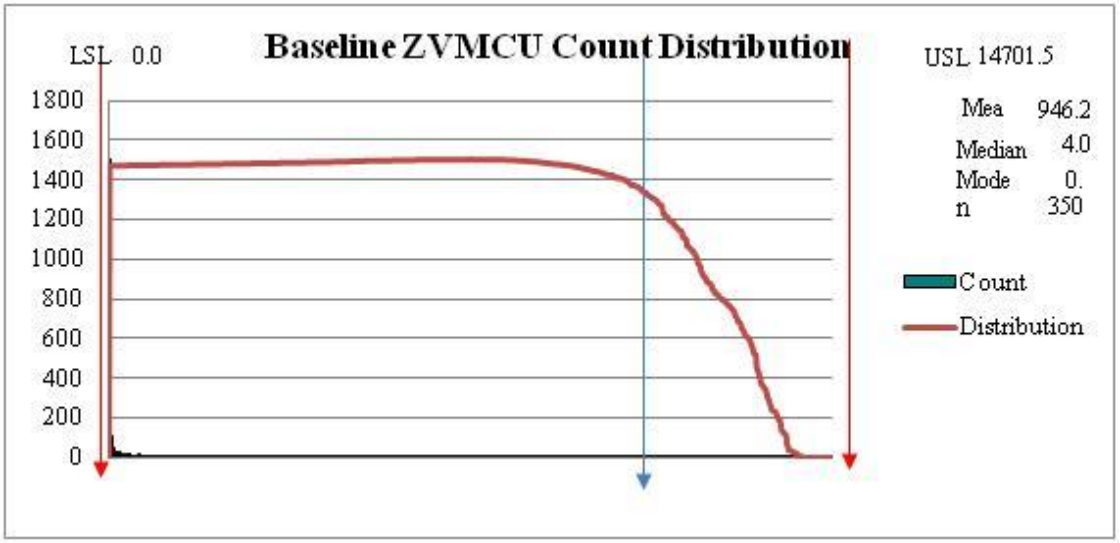


Figure 38: ZVMCU Count Distribution for Baseline images

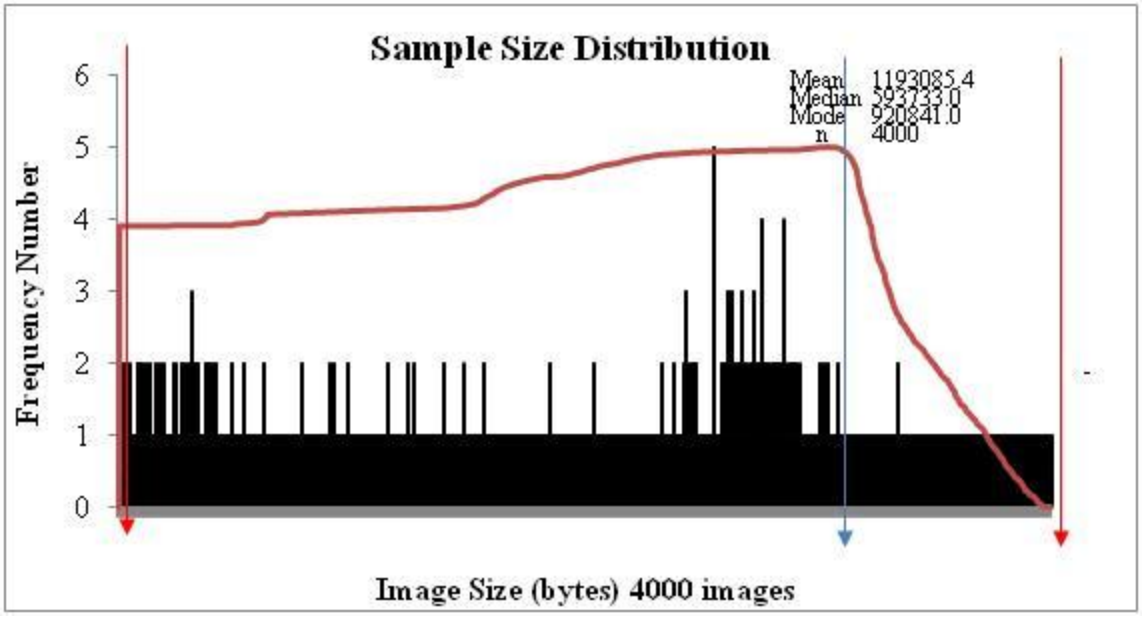


Figure 39: Image Sample Size Distribution All 4000 Images in Sample Set

The statistical model normally used to measure differences in variance is the F-Test. However, because the sample population does not follow a normal distribution, the Levene Test for Equality of Variance is a more appropriate choice. (Filliben and Heckert 2010) The Levene test is used to test for homogeneity across variance. In testing the null hypothesis for theory III, that there is zero variance in the quantity of ZVMCUs in the results between the control group and the test groups, the Levene Test for Equality of Variance is the best choice due to the non-normal distribution of the dataset.

The formal definition of the Levene test follows:

$H_{0:}$ $\boldsymbol{\sigma_1 = \sigma_2 \ldots = \sigma_k}$

$H_{1:}$ $\boldsymbol{\sigma_i \neq \sigma_j}$

Informally and relevant to the data presented in this research. The (H_0) or null hypothesis is that the variance of the control group is equal to the test group. The (H_1) or alternative hypothesis is that steganographic embedding software cause anomalies in the variance of the quantity of the ZVMCUs of JPEG images.

The test statistic for the Levene test is defined as:

$$W = \frac{(N-k)\ \sum_{i=1} k\, N_i \left(\overline{Z_i} - \overline{Z_{..}}\right)^2}{(k-1) \sum_{i=1} k \sum_{j=1} N_i\ \left(Z_{ji} - \overline{Z_i}\right)^2}$$

Where N is the sample size of the image groups (144 images), k is the number of sample groups (3 groups) subset of the sample. According to the National Institute of Standards and Technology (NIST), the selection of the value for Z_{ij} is dependent upon the actual distribution of the data. For a dataset with a skewed distribution such as the image samples, the recommended definition for Z_{ij} is

$$Z_{ij} = \left| Y_{ij} - \widetilde{Y}_{\iota} \right|$$

Where Y_{ij} is the median of the group. (Filliben and Heckert 2010) Other definitions are available but are not as well suited to our dataset.

4.4.6 Results of Levene Test for Equality of Variance

The results of the Levene Test for Equality of Variance test fail to reject the null hypothesis and therefore do not encourage the use of the variance of ZVMCUs in an image as an indicator for steganographic embedding in JPEG images. The control group began with the 500 baseline images from category 8. These 500 images were then manipulated using GIMP, Microsoft Photo manager, IRFANVIEW, and Photoscape. Each image is then opened and saved in a new directory using the save- as routine from each program. Subsequently, a dataset of ZVMCUs for each image is computed and a ZVHash of the color coordinate pair is obtained. The resulting ZVHash for each image is then compared to the baseline ZVHash. A total of 2309 images out of a possible 2500[9] were found with viable ZVMCUs. The Levene Test for Equality of Variance was

[9] 500 Baseline + 2000 Save As images = 2500

conducted on the value of ZVMCUs for all 5 datasets. The results are demonstrated in Table 7.

	BaseLine	*Gimp*	*Msphoto*	*PhotoScape*	*IRFANVIEW*
Median	311.5	325.5	321	617.5	635
Mean	256.4	257.0	255.9	323.3	323.9
Variance	68905.8	68883.9	68422.5	75826.3	758035.
n	408	408	411	406	406
df	407	407	410	405	405
	Levene's				
Test	0.611				
p	0.655				
	a	0.05			
Accept Null Hypothesis because p >0.05					

Table 7: Levene Test for Equality of Variance for Save As images

In Table 8, the results from the Levene variance test of the steganographic image population also demonstrates a p value of >0.05. Interestingly, the p value for the steganographic images is 1.0. This indicates, as seen in the details, that there is no variance at all between the values. Whereas, the Save As experiment in Table 7, p = 0.655, indicating there is at least some change, though not significant enough to show that it is not random. The varying Mean and Median values compared to the steganographic image ZVMCUs allows for further investigation.

	Baseline	*JPHS-L*	*JPHS-M*	*JPHS-S*	*JPSTG-L*	*JPSTG-M*	*JPSTG-S*	*Steghide-L*	*Steghide-m*	*Steghide-s*
Median	255.00	255.00	255.00	255.00	255.00	255.00	255.00	255.00	255.00	255.00
Mean	339.64	339.64	339.64	339.64	339.64	339.64	339.64	339.64	339.64	339.64
Variance	76814.11	76814.11	76814.11	76814.11	76814.11	76814.11	76814.11	76814.11	76814.11	76814.11
n	47	47	47	47	47	47	47	47	47	47
df	46	46	46	46	46	46	46	46	46	46
	Levene's									
Test	0.000									
p	1.000									
	a	0.05								
Accept Null Hypothesis because p >0.05										

Table 8: Levene Test for Equality of Variance for Steganographic images

The results from this variance test informs us that the variance demonstrated in the save- as experiment is not significant enough to say the variance is not a result of random sampling. Though both sample populations have an insignificant variance in the number of ZVMCUs per tested image, the homogeneity in median, mean and variance across the three different steganographic embedding routines provides an indicator of a non normal manipulation that may provide a means of identifying steganographic images.

4.7.5 ZVMCU Homogeneity as Steganographic Indicator

The culmination of the experiments in the research has produced a method that serves as a possible indicator of steganographic embedding in JPEG images that has not existed until now. Using the combined techniques described in this dissertation, it is possible to distinguish just those images that have embedded steganographic data and their related cover images.

The final hypothesis is that steganographic images can be culled from the population of images in a finite dataset, reducing the possible suspect images from thousands down to a manageable set of pairs. An overview of the process can be seen in figure 40 and an explanation follows. A finite dataset is searched for sibling image files using the ZVMCU locator method as demonstrated in theory I and II. The new population of suspect images is expected to be at least 98% smaller than the original image population, as demonstrated in theory II. This creates the initial set of images in the suspect pool of images.

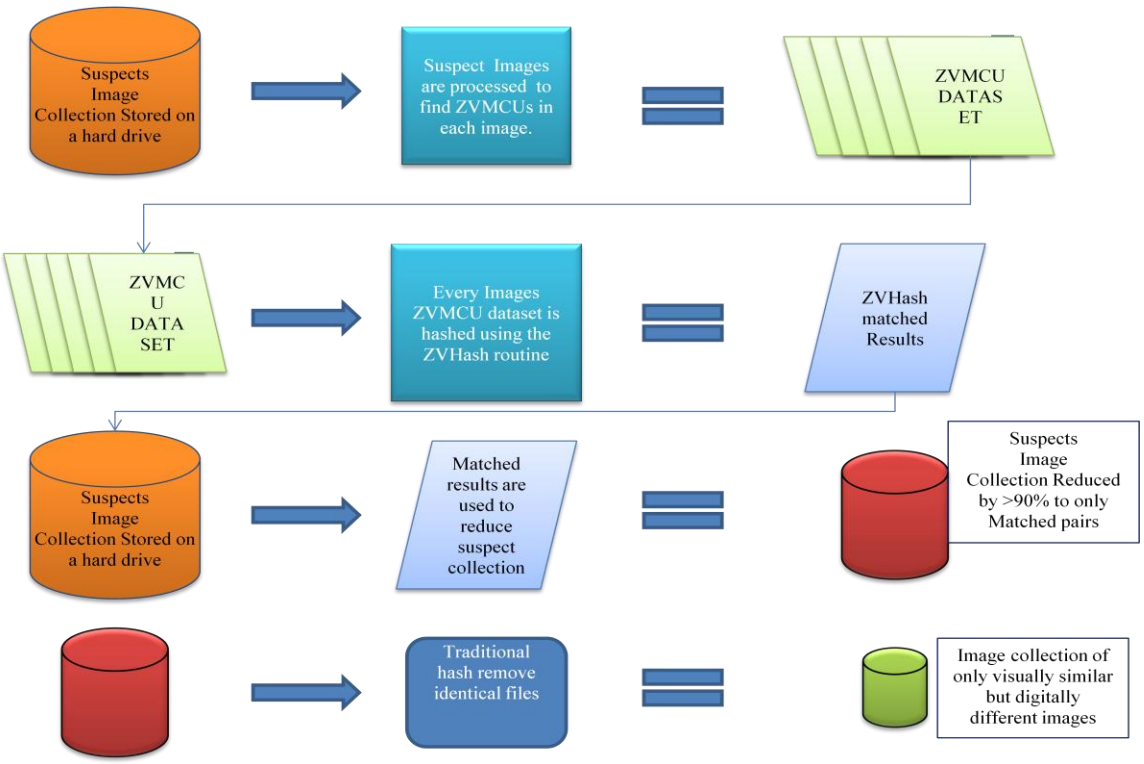


Figure 40: Overview of Steganalysis process using ZVMCUs & Non-Traditional Hashing

The remaining images are then inspected for matching MD5 signatures. If multiple images are found with matching MD5 signatures, only one of the MD5 matches needs to remain in the suspect pool. Next the remaining images are compared. If two images having differing MD5 signatures but exactly the same ZVMCU quantity then it is plausible that one of the images is embedded with steganographic content and the other sibling image is the cover image.

The hypothesis is then; given two images, IM1 and IM2.

If F (MD5) IM1!=F(MD5) IM2 && Count (ZVMCUs) IM1 == Count (ZVMCUs) IM2:

Then steganographic embedding is plausible.

In the pilot test for this theory 9 cover images from the sample database were selected to become steganographic images and were subsequently hidden in a simulated suspect

dataset of 1700 images. Each of the three steganographic programs was then used to embed secret data into the 9 cover images. Each program embedded a large, a medium and a small hidden file in one of 3 cover images, just as performed in the earlier experimentation of section 4.7.2. The decision process for the number of images for the suspect collection is based on the following sample size formula: (Creative Research Systems 2010)

$$ss = {Z * P * (1 - P)}/{c^2}$$

Where:

ss = sample size

Z= confidence level (95% for this research)

P = decimal value of likelihood of selection of wrong choice (50%)

C = confidence interval, normally between 1 and 5. (3 for this research)

In this study, the traditional use of a confidence level of 95% provides sufficient assurance that our sample size is of adequate size. The selection of a confidence interval of 3%, is a typically acceptable error rate for most sampling.

Given these parameters the recommended sample size is

Ss = .95 * .50 *(1-.50)/.03^2 = 264

An adjustment is made for the fact that we have a finite sample dataset of 500.

That adjustment is made using the following formula:

$$New\ ss = ss/({ss - 1}/{ss}) + 1$$

This calculation is straight forward and gives us a suggested sample size of 133

A sample size of 133 from 8 categories would result in a suspect collection of only 1064. In order to create a more accurate representation of a suspect's image collection

additional copies of images were added to the selection process. In all 224, samples were taken from each category. This sample size exceeds the minimum requirements and produces a realistic size suspect data collection. A total of 1700 images constitute the suspect image dataset including the nine steganographic images. The method used to build the suspect image dataset intentionally insured that a copy of the original cover image was included, as well as, multiple other duplicate images. The frequency distribution of the suspect dataset is demonstrated in the following graphs. Figures 41 and 42 demonstrates an image size and ZVMCU quantity frequency distribution of the suspect image set that is equivalent to the distribution demonstrate across the entire sample dataset used in this research as seen in figures 38 and 39.

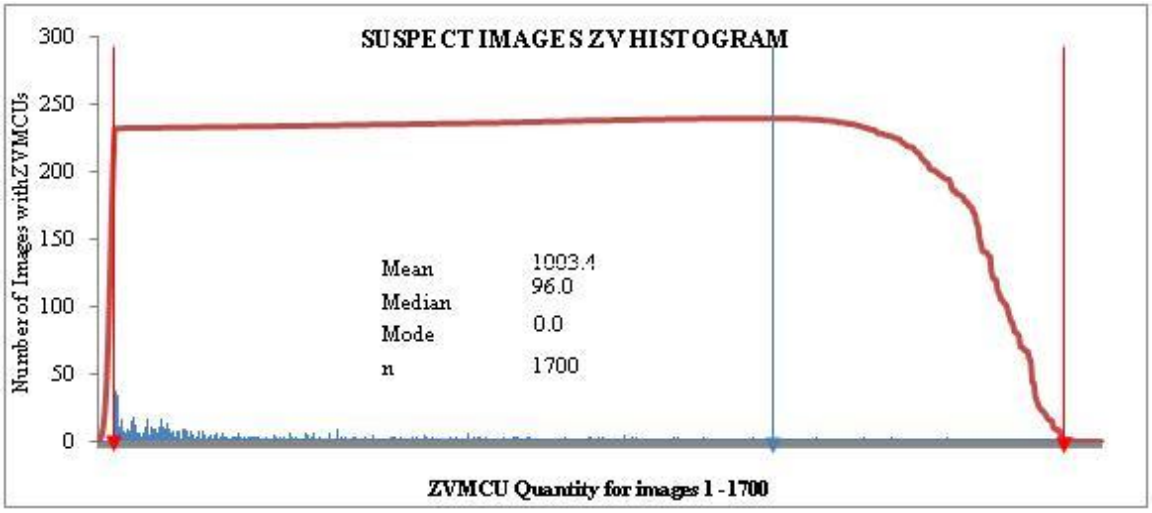


Figure 41: Histogram for Suspect image set ZVMCU quantity

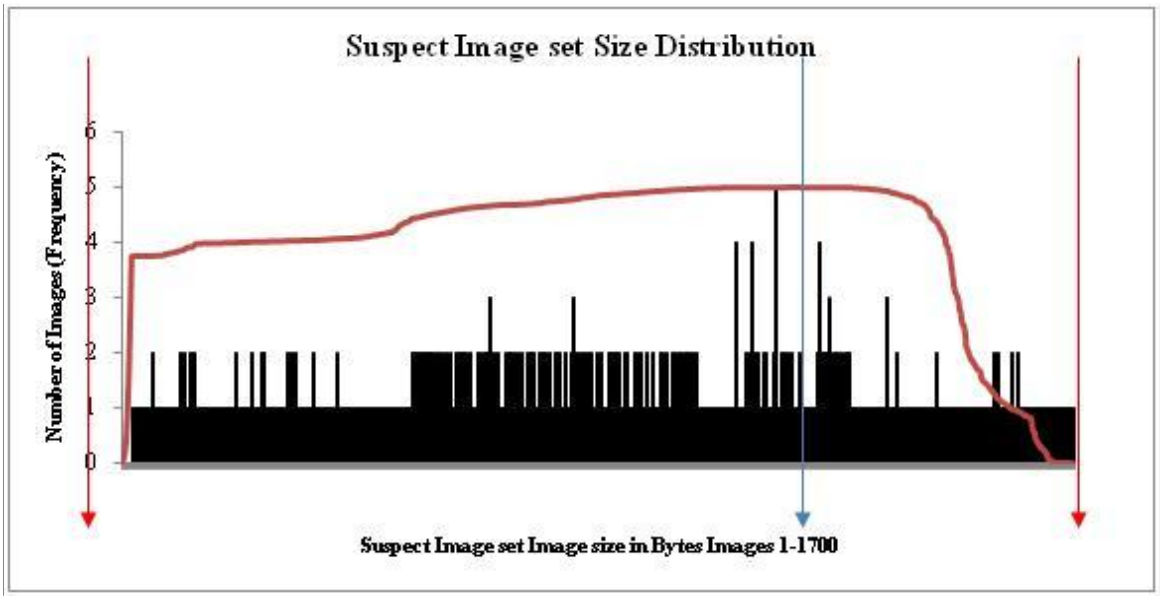


Figure 42: Suspect Image Dataset Size Distribution

The methods that have been detailed throughout this work were then applied to the suspect dataset. The ZVMCUlocator program was used to create a dataset of ZVMCUs for the 1700 images in the suspect image collection. Next, the dataset of ZVMCUs is used as input for the ZVHash program. The ZVHash program compares every image to every other image in the collection (except itself) resulting in over 1.4 million comparisons[10] for this collection of 1700 images. This process produces a dataset of image matches based on the percentage of ZVMCU hash matches. Along with the expected false matches, it is anticipated the hashing routine will return the nine steganographic images, in addition to intentional additional copies injected into the suspect data set. The addition of intentional image copies simulates what a user may have stored across multiple files on their computer and a variable that any investigator would have to expect to encounter. At a 1% and greater match rate the ZVHash routine returned 975 possible matched image files. This number is expectedly high as no metrics were applied to neither eliminate the matches resulting from pixel saturations as detailed

[10] The total comparisons = 0.5 x 1700(1700 +1) = 1,444,150.

in theory II nor to eliminate digitally perfect matches. However, by scaling the matches based on the percentage of ZVMCUs matched reduced the number of matches to a manageable amount. In table 9, the actual image matches returned by the ZVHash routine are presented. Of note in this table is how quickly the false matches are eliminated when the returns are scaled based on the percentage of ZVMCUs matched.

Suspect Images ZV matches pairs	
Image matches	975
Image matches = 100 %	508
Perfect ZV match PAIRS	48

Table 9: Suspect Image Set total ZV matches

However, not all of the 96 images (48 pairs) are possible candidates as steganographic or cover images. As outlined in the hypothesis for this experiment, the application of the traditional MD5 hash routine will identify identical images whose duplicate copies can be also be removed. As images with matching MD5 signatures are visually and digitally identical so only one need to remain for examination as a cover image or steganographic image. An MD5 signature was obtained for each of the 96 matches and 48 duplicates were removed from the suspect pool.

The final step manually reviewed and removed 31 additional image files which did not have a compatible match from the suspect image collection of 48 images resulting in a suspect steganographic image set of just 17 images. Within these 17 images were seven of the nine steganographic images that were hidden in the initial suspect data set of 1700 images. (Figure 43)

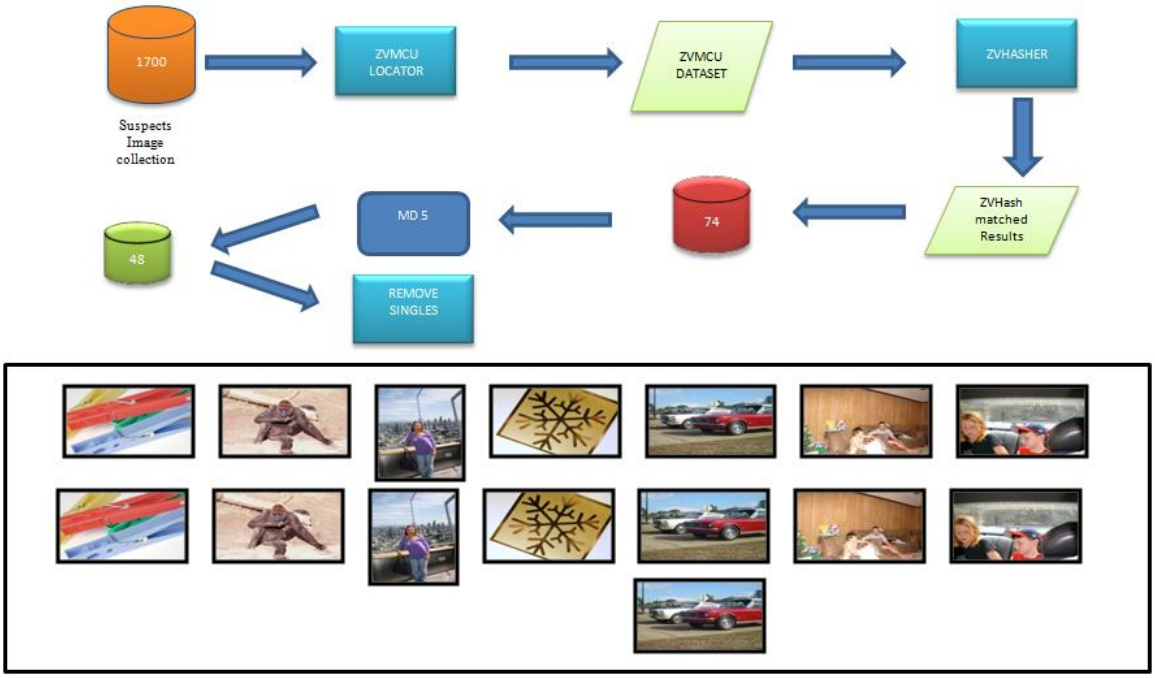


Figure 43: Steganographic Detection process overview

In the final test of the methods described in this research, a few criteria are modified. The final test increases the number of steganographic images that are introduced in to the suspect population of 1700 images. To maintain a realistic suspect data set, the percentage of steganographic images and cover images was kept at 5%. Therefore, a total of 45 images were selected to be used as cover images. Using the same methods described previously, each of the three embedding routines was used to embed 5 large, 5 medium and 5 small hidden files. This resulted in 15 steganographic images per embedding routine and a total of 45 steganographic images. This process created 90 image files (45 CI + 45 SI) which were hidden in the suspect dataset of 1700 images. To accommodate the additional 45 steganographic images being embedded an equal number of images were removed from the original dataset of 1700.

The exact process as described earlier and visualized in figure 43, is used to process this new suspect collection of images with the following results.

Suspect Images ZV Perfect matches pairs	
Image matches 100%	108prs
Images with duplicate filenames removed	148 ea
Images with MD5 Duplicates removed	122 ea
Steganography matches out of 45	34
Suspect Images ZV Matches +/- 1 ZVMCU	
Image matches 100%	171 prs
Images with duplicate filenames removed	191 ea
Images with MD5 Duplicates removed	165 ea
Steganography matches out of 45	41

Figure 44: Steganographic suspects from dataset of 1700

Figure 44 demonstrates some very important finding in the research. First, the number of initial match returns is significantly less. These results are expected because the pilot suspect collection contained more legitimate copies intentionally embedded in the suspect dataset of 1700 that were removed to accommodate the new steganographic images. These numbers demonstrate the ability to reduce a suspect dataset from a size that is visually unmanageable to a number that can be reasonably analysed visually. The expected return for matches in this experiment is a minimum of 90 individual files or 45 pairs, plus the unknown number of copies remaining in the suspect collection. Just as was done the pilot experiment, the final step manually reviewed the remaining 122 images in the suspect dataset, image pairs where matched and examined. In addition to the legitimate steganographic pairs found, there were also 11 pairs of other image matches found in the dataset. The results from the perfect matched pairs are 34 out of 45 steganographic images and cover images were discovered in the suspect dataset of 1700. This is 75.5% success rate and compares to the 77.7% success rate in the pilot test. Of

the steganographic images not found by the methods described in this research, each steganographic image had one of two metrics in common. Either the image was embedded with the JPhide and Seek routine or the image was embedded with a hidden file that was larger than the recommended capacity.

In an effort to improve the success rate and test the ability of adjusting the metrics for image matching, a single parameter was changed to attempt to increase the success rate while minimizing the false positive returns. The parameter that was changed is seen in figure 44; the matching algorithm was allowed to choose images whose ZVMCU matches differed by plus or minus 1 ZVMCU. The new parameters increase the successful returns to 91.1%. The four images that were not returned after this singular parameter was altered were all images whose embedded file exceeded the recommended capacity. By exceeding the recommended capacity, the embedding routine is unable to maintain the first order statistics, making it vulnerable to other steganalysis procedures discussed during the introduction.

4.7.6 Theory III Conclusion

The findings from these experiments conclusively support the combination of ZVMCUs and the non-traditional hash techniques of ZVHash as viable tools in finding probable steganographic images in a finite dataset, without any priori information on the steganographic content, algorithm or cover image. The result from the testing of variance between the quantities of ZVMCUS per image produced results showing that the variance in the ZVMCUs across the sample population is insignificant and does not qualify it for use as an indicator for steganographic embedded files. However, the fact that every dataset in the steganographic image group returned exactly the same Median, Mean, and

Variance demonstrates how the embedding routines manipulate the underlying pixel structure in order to maintain the first order statistics. This is in contrast to the save-as routine, which did cause changes, though these changes were statistically insignificant across each of the different photo editing software packages. This insignificance however, does not exclude the ZVMCU as a possible tool for the detection of steganographic images. It does however; rule out the use of the variance in the ZVMCUs in the population as a detection tool. As demonstrated ZVMCUs can be used to find sibling images in a dataset, then using the comparison of the quantity of the ZVMCUs with the traditional hash routine, images can be categorized with a level of confidence to their likelihood of containing embedded information.

The effectiveness of the methods discovered and described in this research is clearly present in the results from the final test. The final test demonstrated that the ZVMCU landmarks combined with traditional MD5 hashing and non-traditional hashing of the ZVHash routine, is effective in locating JPEG steganography in a finite data set. The methods defined in this research located 77% of the steganographic images in the pilot test and 91% of the steganographic images out of a possible 1700 suspect images, in the follow on test. The notable aspect of this final test is that the blind nature of the examination. The method uses no prior knowledge or metrics in locating the probable steganographic images, setting it apart from other steganalysis methods. Another factor setting this research apart from other steganalysis routines is that the ZVHash routine is effective with small and medium size hidden files, where other universal steganalysis methods fail to detect smaller embedded files.

Chapter 5

Key Contributions and Conclusion

5.1 Key Contributions

- Discovered previously unknown robust landmarks named zero variance minimal computer units (ZVMCU) in JPEG images that remain viable through the JPEG compression routine

- Developed a method that allows for the unique identification of consanguineous JPEG images in a finite population by incorporating the newly discovered ZVMCUs and a unique implementation of the traditional message digest routine that is robust enough to survive recompression by JPEG lossy compression techniques.

- Developed a quantifiable method for the reduction of suspect population of images, significantly reducing the forensic effort necessary to detect, identify and/or disrupt steganographic communications and techniques.

- Successfully demonstrated the combination of methods useful in detection of steganographic embedding in JPEG images that has not previously existed.

5.2 Conclusion

There are several significant contributions are accomplished in this research. The first contribution delivered with is the discovery of a method to uniquely identify JPEG images that succeeds in matching consanguineous images across the JPEG compression routine where traditional digital signatures fail. The second contribution from this research is the discovery of a method useful in assisting digital forensic investigators by effectively and efficiently reducing the dataset of images they need to review in search of steganographic images. The third contribution to the academic and scientific communities is the development of a new process capable of detecting JPEG steganographic images in a finite dataset without prior knowledge of the steganographic algorithm, hidden content or cover image. These contributions were accomplished through a meticulous and detailed set of experiments each culminating in a successful demonstration of the objective. The first set of experiments established the existence and persistence of landmarks named ZVMCUs that are present in a large portion of JPEG images. Further, the first set of experiments demonstrated the robustness of the ZVMCUs to multiple types of image manipulations across a full range of image size and resolutions. Building upon the first findings, the second set of experiments set out with the goal of finding sibling images in a finite dataset. Combining non-traditional hashing techniques with the newly discovered ZVMCUs, the second set of experiments concluded in a successful demonstration of the ZVHash routines' ability to match visually similar images in cases where a traditional hash routine would fail. The final set of experiments, ambitiously undertook the mission to discover a reliable method to identify JPEG steganographic embedding in a finite data set of JPEG images. By exploring the qualities

of the ZVMCUs variance in comparison to the normal population, an anomaly of the JPEG steganographic embedding routines was discovered and exploited. This anomaly is that the steganographic embedding routines tendency to maintain the images' first order statistics. This caused a zero variance in the quantity of ZVMCUs between the cover image and the steganographic image. The techniques discovered in the first set of experiments were brought to bear on this anomaly with positive results. The combination of the landmark ZVMCUs, traditional message digest, and image matching capabilities of the ZVHash routine, resulted in a method that successfully identified 91% of the steganographic images in a finite data set. Each of these accomplishments contributes its own unique and new found capability to the law enforcement and security communities. However, it is the contributions of the new found landmarks and embedding anomalies to the computer science research community that truly opens the door for further exploration into JPEG steganalysis. In summary, this research proved that there are viable landmarks that can be located in JPEG images and these landmarks can be used to accurately find consanguineous JPEG images and detect transform domain JPEG steganography.

Chapter 6

Future Work

The field of steganography and steganalysis is a dynamic and growing area of research. There are many exciting avenues that have been uncovered in the current research that will be exciting to explore further. In continuing with this research the first project is the refinement of the ZVHash routine to eliminate false positives and to expand its capabilities by incorporating established image discrimination techniques such as using EXIF data as additional matching and selection criteria. The refinement of the ZVMCU locator and ZVHash software in effort to make its JPEG image matching capabilities more powerful and adjustable is also a planned area of continued research. In addition to the continued research in universal jpeg steganography detection, I plan to continue to research is finding methods to possibly extract the hidden data in digital image steganography. Further, I plan to continue searching for methods that will create a forensically appropriate technique to track a known image across the Internet to facilitate law enforcement in their efforts to prosecute criminals, possibly through the injection of indelible landmark ZVMCUs. In the unexplored territory, I plan to extend my research into other areas of steganography such as covert channels in TCP-IP transmissions, digital music and videos.

Bibliography

Almohammad, Adel, Gheorghita Ghinea, and Robert M. Hierons. "JPEG Steganography: A Performance Evaluation of Quantization Tables." *Advanced Information Networking and Applications, International Conference on* (IEEE Computer Society) 0 (2009): 471-478.

Bailey, Curran Kevin. *Steganography The Art Of Hiding Information.* BookSurge Publishing, 2004.

Chae, Jong J., and B. S. Manjunath. "Technique for image data hiding and reconstruction without host image." Edited by Ping W. Wong and Edward J. Delp. SPIE, 1999. 386-396.

Cole, Eric. *Hiding in Plain Sight: Steganography and the Art of Covert Communication.* John Wiley & Sons, Inc., 2003.

Dunbar, Brett. "Sans Reading Room." *San.ORG.* SANS. January 18, 2002. http://www.sans.org/reading_room/whitepapers/covert/a_detailed_look_at_steganographic_techniques_and_their_use_in_an_opensystems_environment_677 (accessed June 15, 2009).

Farid, Hany. *Digital Imaage BAllistic from JPEG Quantization.* Hanover: Dartmouth, 2006.

Filler, Tomas, and Jessica Fridrich. "Complete characterization of perfectly secure stego-systems with mutually independent embedding operation." IEEE Computer Society, 2009. 1429-1432.

Filliben, James J., and Alan Heckert. *NIST/SEMATECH e-Handbook of Statistical Methods.* Feburary 16, 2010. http://www.itl.nist.gov/div898/handbook/ (accessed may 10, 2010).

Forret, Peter. *Megapixel Calculator Digital Camera Resolution.* May 20, 2010. http://web.forret.com/tools/megapixel.asp (accessed June 8, 2010).

Fridrich, Jessica J., Miroslav Goljan, and Dorin Hogea. "Steganalysis of JPEG Images: Breaking the F5 Algorithm." Springer-Verlag, 2003. 310-323.

Fridrich, Jessica, and Miroslav Goljan. "Detecting LSB steganography in color and gray-scale images." *IEEE Multimedia* 8 (2001): 22-28.

Fridrich, Jessica, Miroslav Goljan, and Rui Du. "Steganalysis based on JPEG compatibility." Edited by Andrew G. Tescher, Bhaskaran Vasudev, V. Michael Bove and Jr. SPIE, 2001. 275-280.

Fridrich, Jessica, Tomas Pevy, and Jan Kodovsk. "Statistically undetectable jpeg steganography: dead ends challenges, and opportunities." ACM, 2007. 3-14.

Goldborough, Ried. *infotoday.com.* June 15, 2008. http://www.infotoday.com/linkup/lud071508-goldsborough.shtml (accessed June 8, 2010).

Goljan, Miroslav, Jessica Fridrich, and Taras Holotyak. "New blind steganalysis and its implications." Edited by Edward J. Delp and Ping Wah Wong. SPIE, 2006. 607201.

Hendricks, Gary. *Top 10 Cameras Under 300 dollars.* June 1, 2010. http://www.basic-digital-photography.com/top-10-popular-digital-cameras-under-300.html (accessed June 8, 2010).

Hosmer, Chet, and Christopher Hyde. "Discovering Covert Digital Evidence." Cleveland, Oh: Wetstone Technologies, 2003.

Huang, Fangjun, Bin Li, and Jiwu Huang. "Attack LSB Matching Steganography by Counting Alteration Rate of the Number of Neighbourhood Gray Levels." 2007. I--401--I--404.

Hurlburt, Dustin. "Fuzzy Hashing for Digital Forensic Investigators." Seattle, Wa: Access Data, 2009.

"Japan Electronics and Information Technology Industries Association, Exchangeable Image File Format for Digital Still Cameras: EXIF Version 2.2." *Japan Electronics and Information Technology Industries Association, Exchangeable Image File Format for Digital Still Cameras: EXIF Version 2.2.* 2002.

Johnson, N. F., and S. Jajodia. "Exploring steganography: Seeing the unseen." *Computer* 31 (1998): 26-34.

Kellen, Tom. "SANS Reading Room." *SAN.ORG.* Feburary 2001. http://www.sans.org/reading_room/whitepapers/stenganography/hiding_in_plain_view_could_steganography_be_a_terrorist_tool_551 (accessed June 6, 2009).

Kelly, jack. "USA today." *USAToday.* November 19, 2001. http://www.usatoday.com/tech/news/2001-02-05-binladen.htm (accessed August 1, 2009).

Kharrazi, Mehdi, Husrev T. Sencar, and Nasir Memon. "Benchmarking steganographic and steganalysis techniques." Edited by Edward J. Delp and Ping W. Wong. SPIE, 2005. 252-263.

Khayam, Syed Ali. "The Discrete Cosine Transform: Theory and Aplication." Michigan State University, 2003.

Kornblum, Jesse D. "Using JPEG quantization tables to identify imagery processed by software." *Science Direct* (Elsevier) 5, no. s21-s25 (2008).

Kornblum, Jesse. "Identifying almost identical files using context triggered piecewise hashing." *Digital Investigation* (Elsevier Ltd) 3S, no. S91-S97 (June 2006): 91-97.

Lin, Eugene T., and Edward J. Delp. "A Review of Data Hiding in Digital Images." 1999. 274-278.

Lyu, S., and H. Farid. "Steganalysis Using Higher-Order Image Statistics."
McCullagh, Declan. "Wired News." *Wired News.* Feburary 20, 2001. http://www.wired.com/politics/law/news/2001/02/41861 (accessed July 15, 2009).

NCMEC. http://www.ncmec.org/ (accessed 10 15, 2009).
Preneel, Bart. "National Institute of Standard and Technology." *csrc.nist.gov.* U.S. Government. October 1, 2005. http://csrc.nist.gov/groups/ST/hash/documents/preneel_nist_v2.pdf (accessed September 25, 2009).

Provos, Niels. "Defending against statistical steganalysis." USENIX Association, 2001. 24-24.

Quach, Tu-Thach, Fernando Perez-Gonzalez, and Gregory L. Heileman. "Model-based steganalysis using invariant features." Edited by Edward J. Delp, Jana Dittmann, Nasir D. Memon and Ping Wah Wong. SPIE, 2009. 72540B.

Roussev, Vassil, Golden Richard, and Lodovico Marziale. "Multi-resolution similarity hashing." (Elsevier) 4S, no. S105-S113 (2007): 9.

Roussev, Vassil, Richard G. Golden, and Lodovico Marziale. "Multi-resolution similarity hashing." *Digital Investigation* (Elsevier) 4S (June 2007): 105-113.

SARC Editort. *SARC.* 10 15, 2009. http://www.sarc-wv.com/default.aspx (accessed 10 15, 2009).

Silva, John Edward. "An overview of Cryptographic Hash Functions and their Uses." *SANS Reading Room* Version1.4b (2003): 13.

Sorell, Matthew James. "Conditions for effective detection and identification of primary quantization of re-quantized JPEG images." ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008. 1-6.

"The American Heritage® Dictionary of the English Language." Houghton Mifflin Company, 2004.

Wallace, Gregory K. "The JPEG still picture compression standard." *Commun. ACM* (ACM) 34 (1991): 30-44.

Wingate, Jim. *The Perfect Dead Drop the use of cyberspace for covert communications.* white paper. BackBone Security. 2005.

Appendix A: Publically available Steganographic programs

Source 1:http://www.jjtc.com/Steganography/toolmatrix.htm
Source2: http://www.topology.org/soft/steganography.html
Source3: http://home.comcast.net/~ebm.md/stego/softwaredos.html
BMPEmbed v1.54
Camouflage: Windows GUI, insertion based steganographic program uses the right click menu to hide/reveal data
Contraband Hell Edition- a windows based edition of contraband that uses SHA1 encryption in the embedding process.
Contraband – a command line, 16 bit program embeds and extracts any file into 24 24Bit BMP files. Uses a 'scrambler' to embed the hidden file.
Courier v1.0
Covert-TCP
Cryptapix-an image file management, steganography, and encryption program for Windows
Data Stash v1.1, v1.1a
DC-Stego – Written by Nikola Injac, a small, DOS-based steganography program that hides data in PCX image files.
DCT-Steg
Digital Picture Envelope
Diskhide
Dmagic v6.0
Data Privacy Tool -Strong encryption with optional BMP steganography
EasyPrivacy Pro v2.1.1
EIKONAmark
Empty Pic- a gif based steganography program
Encrypt Pic v1.3
Encrypted Magic Folders
EzStego
F5 v0.9
FFEncode – hides files in text files, by using Morse code of the NULL characters.
Folder Guard v4.11,
Ghost Host
Gif-It-Up
Gifshuffle
Giovanni
GzSteg-hides data in GZip files. compiled by Preston Wilson
HD v3.0
Hidden v8.2
Hide and Seek 5- GUI based steganography program allows for hiding in GIF files.

Hide In Picture- program that lets you "hide" files inside bitmaps and GIFHide Unhide
Hide v2.0
Hide4PGP v2.0 -command-line steganographic program for Windows, DOS, and OS/2 that hides data within BMP, WAV, and VOC files. Developed by Heinz Repp.
Hide Drive
Hideme
InThePicture
Invisible Encryption
Invisible Files 2000
JK-PGS
JPHS- Windows GUI, JPEG transform domain steganographic embedding routine
Jsteg-Jpeg- indirectly hides data within JPG format. Files must be saved in the TGA (targa) format.
JSTEGO- java solution to hide data in JPEG images. Algorithm contains Jsteg and F5.
Magic Folders
Mandelsteg- a creation based steganographic program producing images from input data.
Mimic
MP3Stego-command-line program that runs in DOS and hides information in MP3 files.
Nicetext
Outguess
Paranoid
PGE v2.0- this version does not include encryption. Allow hiding data inside most file types, except JPEG.
PGM Stealth
PJStgTST-Hides datafiles in the LSB of JPEG images. Modifies Transform Domain.
S-Tools - hides files in BMP, GIF, and WAV files. Developed by Andy Brown.
Safer v2.0
SandMark
Scramdisk
Scytale
SecureEngine-JPEG steganography freeware program from Brother soft.
http: //www. brothersoft.com / securengine-download-23496.html
SGPO
Smail- Windows based Steganography program that hides files inside .EXE and .DLL Files.
Snow- text-based program by Matthew Kahn, that hides messages in text files by appending tabs and spaces.
Snowdisk
Spyder
Stash
Stealth- a simple filter for PGP which strips off all identifying header information

Steganos- Dos based steganography program by Fabian Hansmann. Hide data inside BMP,WAV and ASCII files.
Steghide
StegFS
StegComm
StegoWav-hides data in WAV files.
Texto- a creation based steganography program that creates a text like document from the hidden file. Results are similar to what is seen in spam messages.
Wnstorm- hides files inside PCX images using both steganography and encryption.

Appendix B: References for Sample Size Calculations.

The equation is given from http://www.surveysystem.com/sample-size-formula.htm

This calculator gives a recommended sample size of: 217 less than used in this study.

Uses the following formula:

$$\boldsymbol{ss} = {}^{Z * \boldsymbol{P} * (\mathbf{1} - \boldsymbol{P})}\big/_{\boldsymbol{c^2}}$$

Where Z is the Z value or confidence level of the study in our case 1.96 for 95%

P = is the percentage of choice of selection (.5 or 50% is normal selection)

C = confidence interval as a decimal. Normally >1 but <5.

The following additional calculation is necessary to account for the finite dataset of 500.

$$\boldsymbol{New\ ss} = \boldsymbol{ss}/({}^{\boldsymbol{ss} - \mathbf{1}}\big/_{\boldsymbol{ss}}) + \mathbf{1}$$

An additional validation of the sample sizes used in this study was conducted on an online calculator at http://www.raosoft.com/samplesize.html: This online calculator gives a recommended sample size of 218 which is less than the sample size used in the study. This website presents the following formula for its estimation of sample size.

$$x = Z\left(\frac{c}{100}\right)^2 r(100 - r)$$

$$n = \frac{Nx}{((N-1)E^2 + x)}$$

$$E = Sqrt[{}^{(N-n)x}\big/_{n(N-1)}]$$

Appendix C: Detail results for Test One all image Categories

Test I: Image Quality Reduction to 80 %
Program I: Gimp 2.6

This experiment is designed to quantify the effects of image manipulation on the quantity and location of zero variance minimal computer units located in the image. This test reduces the quality of each picture in the data set from 100 to 80. The resulting image size will be analysed to determine the change in the number and location of all ZVMCUs in the image. The GIMP image editing software will impart into the image its own implementation of the JPEG compression routine, its own Discrete Cosine Transform quantization tables, and Huffman coding tables. This experiment is vitally important to determine how differing photo editing software programs might change the location, values and quantity of the zero variance minimal computer units in an image that will be used to provide the landmarks for further processing.

Image Category 1: 500 images average size 4.1 MB average resolution of 9.8 MP

Image Category statistics prior to image quality reduction:

Total number of images with at least 3 ZVMCUs:	140
Total number of images with no ZVMCUs:	360
Percentage of images with ZVMCUs:	28 %
Average number of ZVMCUs per image:	226

Image Category statistics after image quality reduction

Total number of images with at least 3 ZVMCUs:	498

Total number of images with no ZVMCUs: 2

Percentage of Images with ZVMCUs: 99.6

Average number of ZVMCUs per image: 2229

Image Category 2: 500 images average size of 856.9 KB average resolution of 1.9 MP

Image Category statistics prior to image quality reduction:

Total number of images with at least 3 ZVMCUs: 206

Total number of images with no ZVMCUs: 294

Percentage of images with ZVMCUs: 41%

Average number of ZVMCUs per image: 222

Image Category statistics after image quality reduction

Total number of images with at least 3 ZVMCUs: 433

Total number of images with no ZVMCUs: 67

Percentage of Images with ZVMCUs: 86.6

Average number of ZVMCUs per image: 632

Image Category 3: 500 images average size of 115.0 KB average resolution of .30 MP

Image Category statistics prior to image quality reduction:

Total number of images with at least 3 ZVMCUs: 191

Total number of images with no ZVMCUs: 309

Percentage of images with ZVMCUs: 38.2%

Average number of ZVMCUs per image: 111

Image Category statistics after image quality reduction

Total number of images with at least 3 ZVMCUs: 387

Total number of images with no ZVMCUs: 113

Percentage of Images with ZVMCUs: 77.4%

Average number of ZVMCUs per image: 226

Image Category 4: 500 images average size of 2.97 MB average resolution of 6.3 MP

Image Category statistics prior to image quality reduction:

Total number of images with at least 3 ZVMCUs: 375

Total number of images with no ZVMCUs: 125

Percentage of images with ZVMCUs: 75.0%

Average number of ZVMCUs per image: 1274

Image Category statistics after image quality reduction

Total number of images with at least 3 ZVMCUs: 479

Total number of images with no ZVMCUs: 21

Percentage of Images with ZVMCUs: 96.2%

Average number of ZVMCUs per image: 2558

Image Category 5: 500 images average size of 553.0 KB average resolution of 2.3 MP

Image Category statistics prior to image quality reduction:

Total number of images with at least 3 ZVMCUs: 348

Total number of images with no ZVMCUs: 152

Percentage of images with ZVMCUs: 69.0%

Average number of ZVMCUs per image: 2415

Image Category statistics after image quality reduction

Total number of images with at least 3 ZVMCUs: 436

Total number of images with no ZVMCUs: 64

Percentage of Images with ZVMCUs: 86.6%

Average number of ZVMCUs per image: 3635

Image Category 6: 500 images average size of 133.6 KB average resolution of .49 MP

Image Category statistics prior to image quality reduction:

Total number of images with at least 3 ZVMCUs: 375

Total number of images with no ZVMCUs: 125

Percentage of images with ZVMCUs: 75.0%

Average number of ZVMCUs per image: 505

Image Category statistics after image quality reduction

Total number of images with at least 3 ZVMCUs: 400

Total number of images with no ZVMCUs: 100

Percentage of Images with ZVMCUs: 80.0%

Average number of ZVMCUs per image: 664

Image Category 7: 500 images average size of 3.045 KB average resolution of .04 MP

Image Category statistics prior to image quality reduction:

Total number of images with at least 3 ZVMCUs: 85

Total number of images with no ZVMCUs: 415

Percentage of images with ZVMCUs: 17.0%

Average number of ZVMCUs per image: 3

Image Category statistics after image quality reduction

Total number of images with at least 3 ZVMCUs: 111

Total number of images with no ZVMCUs: 389

Percentage of Images with ZVMCUs: 22.2%

Average number of ZVMCUs per image: 4

Image Category 8: 430 images average size of 701 KB average resolution of 4.5 MP

Image Category statistics prior to image quality reduction:

Total number of images with at least 3 ZVMCUs: 408

Total number of images with no ZVMCUs: 22

Percentage of images with ZVMCUs: 94.9%

Average number of ZVMCUs per image: 3110

Image Category statistics after image quality reduction

Total number of images with at least 3 ZVMCUs: 428

Total number of images with no ZVMCUs: 2

Percentage of Images with ZVMCUs: 99.5%

Average number of ZVMCUs per image: 3963

Appendix D: Detail results for Test Two all image Categories

Test II: Image Rotation 180 degrees Program I: Gimp 2.6

This experiment is designed to quantify the effects of image manipulation on the quantity and location of zero variance minimal computer units located in the image. This test will change the baseline orientation of the image by 180 degrees. All other JPEG influencing factors are set as to minimize any further influence. The resulting image size will be analyzed to determine the change in the number and location of all ZVMCUs in the image. The GIMP image editing software will impart into the image its own implementation of the JPEG Compression routine, its own Discrete Cosine Transform quality tables, and Huffman coding tables. This experiment is vitally important to determine how differing photo editing software programs might change the location, values and quantity of the zero variance minimal computer units in an image that will be used to provide the landmarks for further processing.

Image Category 1: 500 images average size 4.1 MB average resolution of 9.8 MP

Image Category statistics prior to image rotation by 180 degrees:

Total number of images with at least 3 ZVMCUs:	140
Total number of images with no ZVMCUs:	360
Percentage of images with ZVMCUs:	28 %
Average number of ZVMCUs per image:	226

Image Category statistics after image rotation by 180 degrees

Total number of images with at least 3 ZVMCUs:	140

Total number of images with no ZVMCUs: 360

Percentage of Images with ZVMCUs: 28 %

Average number of ZVMCUs per image: 274

Image Category 2: 500 images average size of 856.9 KB average resolution of 1.9 MP

Image Category statistics prior to image rotation by 180 degrees:

Total number of images with at least 3 ZVMCUs: 206

Total number of images with no ZVMCUs: 294

Percentage of images with ZVMCUs: 41%

Average number of ZVMCUs per image: 222

Image Category statistics after image rotation by 180 degrees

Total number of images with at least 3 ZVMCUs: 202

Total number of images with no ZVMCUs: 298

Percentage of Images with ZVMCUs: 40.4%

Average number of ZVMCUs per image: 243

Image Category 3: 500 images average size of 115.0 KB average resolution of .30 MP

Image Category statistics prior to image rotation by 180 degrees:

Total number of images with at least 3 ZVMCUs: 191

Total number of images with no ZVMCUs: 309

Percentage of images with ZVMCUs: 38.2%

Average number of ZVMCUs per image: 111

Image Category statistics after image rotation by 180 degrees

Total number of images with at least 3 ZVMCUs: 179

Total number of images with no ZVMCUs: 321

Percentage of Images with ZVMCUs: 35.8%

Average number of ZVMCUs per image: 103

Image Category 4: 500 images average size of 2.97 MB average resolution of 6.3 MP

Image Category statistics prior to image rotation by 180 degrees:

Total number of images with at least 3 ZVMCUs: 375

Total number of images with no ZVMCUs: 125

Percentage of images with ZVMCUs: 75.0%

Average number of ZVMCUs per image: 1274

Image Category statistics after image rotation by 180 degrees

Total number of images with at least 3 ZVMCUs: 368

Total number of images with no ZVMCUs: 132

Percentage of Images with ZVMCUs: 73.9%

Average number of ZVMCUs per image: 1264

Image Category 5: 500 images average size of 553.0 KB average resolution of 2.3 MP

Image Category statistics prior to image rotation by 180 degrees:

Total number of images with at least 3 ZVMCUs: 348

Total number of images with no ZVMCUs: 152

Percentage of images with ZVMCUs: 69.0

Average number of ZVMCUs per image: 2415

Image Category statistics after image rotation by 180 degrees

Total number of images with at least 3 ZVMCUs: 0

Total number of images with no ZVMCUs: 0

Percentage of Images with ZVMCUs: 0

Average number of ZVMCUs per image: 0

Image Category 6: 500 images average size of 133.6 KB average resolution of .49 MP

Image Category statistics prior to image rotation by 180 degrees:

Total number of images with at least 3 ZVMCUs: 375

Total number of images with no ZVMCUs: 125

Percentage of images with ZVMCUs: 75.0

Average number of ZVMCUs per image: 505

Image Category statistics after image rotation by 180 degrees

Total number of images with at least 3 ZVMCUs: 320

Total number of images with no ZVMCUs: 180

Percentage of Images with ZVMCUs: 64.0%

Average number of ZVMCUs per image: 476

Image Category 7: 500 images average size of 3.045 KB average resolution of .04 MP

Image Category statistics prior to image rotation by 180 degrees:

Total number of images with at least 3 ZVMCUs: 85

Total number of images with no ZVMCUs: 415

Percentage of images with ZVMCUs: 17.0

Average number of ZVMCUs per image: 3

Image Category statistics after image rotation by 180 degrees

Total number of images with at least 3 ZVMCUs: 45

Total number of images with no ZVMCUs: 455

Percentage of Images with ZVMCUs: 9.0%

Average number of ZVMCUs per image: 2

Image Category 8: 430 images average size of 701 KB average resolution of 4.5 MP

Image Category statistics prior to image rotation by 180 degrees:

Total number of images with at least 3 ZVMCUs: 408

Total number of images with no ZVMCUs: 22

Percentage of images with ZVMCUs: 94.9%

Average number of ZVMCUs per image: 3110

Image Category statistics after image rotation by 180 degrees

Total number of images with at least 3 ZVMCUs: 408

Total number of images with no ZVMCUs: 22

Percentage of Images with ZVMCUs: 94.9%

Average number of ZVMCUs per image: 3082

Appendix E: Detail results for Test Three all image Categories

Test III: Image Size Reduction to 80%

Program I: Gimp 2.6

Image Category I: 500 images with average size 2.2 MB

This experiment is designed to quantify the effects of image manipulation on the quantity and location of zero variance minimal computer units located in the image. This experiment reduces the size of the image by 20% of original, while maintaining its aspect ratio. All other JPEG influencing factors are set as to minimize any further influence. The resulting image size will be analyzed to determine the change in the number and location of all ZVMCUs in the image. The GIMP image editing software will impart into the image its own implementation of the JPEG Compression routine, its own Discrete Cosine Transform quality tables, and Huffman coding tables. This experiment is vitally important to determine how differing photo editing software programs might change the location, values and quantity of the zero variance minimal computer units in an image that will be used to provide the landmarks for further processing.

Image Category 1: <u>500 images average size 4.1 MB average resolution of 9.8 MP</u>

<u>Image Category statistics prior to size reduction by 20 %:</u>

Total number of images with at least 3 ZVMCUs:	140
Total number of images with no ZVMCUs:	360
Percentage of images with ZVMCUs:	28 %
Average number of ZVMCUs per image:	226

Image Category statistics after size reduction by 20 %

Total number of images with at least 3 ZVMCUs: 94

Total number of images with no ZVMCUs: 406

Percentage of Images with ZVMCUs: 18.8%

Average number of ZVMCUs per image: 141

Image Category 2: 500 images average size of 856.9 KB average resolution of 1.9 MP

Image Category statistics prior to size reduction by 20 %:

Total number of images with at least 3 ZVMCUs: 206

Total number of images with no ZVMCUs: 294

Percentage of images with ZVMCUs: 41%

Average number of ZVMCUs per image: 222

Image Category statistics after size reduction by 20 %

Total number of images with at least 3 ZVMCUs: 157

Total number of images with no ZVMCUs: 343

Percentage of Images with ZVMCUs: 31.4%

Average number of ZVMCUs per image: 119

Image Category 3: 500 images average size of 115.0 KB average resolution of .30 MP

Image Category statistics prior to size reduction by 20 %:

Total number of images with at least 3 ZVMCUs: 191

Total number of images with no ZVMCUs: 309

Percentage of images with ZVMCUs: 38.2%

Average number of ZVMCUs per image: 111

Image Category statistics after size reduction by 20 %

Total number of images with at least 3 ZVMCUs: 101

Total number of images with no ZVMCUs: 399

Percentage of Images with ZVMCUs: 20.2%

Average number of ZVMCUs per image: 36

Image Category 4: 500 images average size of 2.97 MB average resolution of 6.3 MP

Image Category statistics prior to size reduction by 20 %:

Total number of images with at least 3 ZVMCUs: 375

Total number of images with no ZVMCUs: 125

Percentage of images with ZVMCUs: 75.0%

Average number of ZVMCUs per image: 1274

Image Category statistics after size reduction by 20 %

Total number of images with at least 3 ZVMCUs: 262

Total number of images with no ZVMCUs: 248

Percentage of Images with ZVMCUs: 52.3%

Average number of ZVMCUs per image: 552

Image Category 5: 500 images average size of 553.0 KB average resolution of 2.3 MP

Image Category statistics prior to size reduction by 20 %:

Total number of images with at least 3 ZVMCUs: 348

Total number of images with no ZVMCUs: 152

Percentage of images with ZVMCUs: 69.0%

Average number of ZVMCUs per image: 2415

Image Category statistics after size reduction by 20 %

Total number of images with at least 3 ZVMCUs: 0

Total number of images with no ZVMCUs: 0

Percentage of Images with ZVMCUs: 0

Average number of ZVMCUs per image: 0

Image Category 6: 500 images average size of 133.6 KB average resolution of .49 MP

Image Category statistics prior to size reduction by 20 %:

Total number of images with at least 3 ZVMCUs: 375

Total number of images with no ZVMCUs: 125

Percentage of images with ZVMCUs: 75.0%

Average number of ZVMCUs per image: 505

Image Category statistics after size reduction by 20 %

Total number of images with at least 3 ZVMCUs: 214

Total number of images with no ZVMCUs: 286

Percentage of Images with ZVMCUs: 42.8%

Average number of ZVMCUs per image: 203

Image Category 7: 500 images average size of 3.045 KB average resolution of .04 MP

Image Category statistics prior to size reduction by 20 %:

Total number of images with at least 3 ZVMCUs: 85

Total number of images with no ZVMCUs: 415

Percentage of images with ZVMCUs: 17.0%

Average number of ZVMCUs per image: 3

Image Category statistics after size reduction by 20 %

Total number of images with at least 3 ZVMCUs: 36

Total number of images with no ZVMCUs: 464

Percentage of Images with ZVMCUs: 7.2%

Average number of ZVMCUs per image: 1

Image Category 8: 430 images average size of 701 KB average resolution of 4.5 MP

Image Category statistics prior to size reduction by 20 %:

Total number of images with at least 3 ZVMCUs: 408

Total number of images with no ZVMCUs: 22

Percentage of images with ZVMCUs: 94.9%

Average number of ZVMCUs per image: 3110

Image Category statistics after size reduction by 20 %

Total number of images with at least 3 ZVMCUs: 325

Total number of images with no ZVMCUs: 105

Percentage of Images with ZVMCUs: 75.6%

Average number of ZVMCUs per image: 1214

Appendix F: Detail results for Test Four all image Categories

Test IV: Image Quality Reduction to 50 %

Program I: Gimp 2.6

Image Category I: 500 images with average size 2.2 MB

This experiment is designed to quantify the effects of the photo image editing software GIMP on the dataset of images. This test will reduce the quality of each picture in the data set from 100 to 50. This drastic reduction in quality helps to scale the finding from test 1 and should provide useful insight in to how and when ZVMCUs are created. The resulting image size will be smaller with significantly reduced quality. The quantization tables used in this test will have the largest divisors of the entire experimental test set, resulting in the greatest number of zero cosine coefficients. The expected results of this experiment are a vast increase in the numbers of ZVMCUs in the image and the dataset. This experiment is vitally important to determine how differing photo editing software programs might change the location, values and quantity of the zero variance minimal computer units in an image that will be used to provide the landmarks for further processing.

Image Category 1: 500 images average size 4.1 MB average resolution of 9.8 MP

Image Category statistics prior to Quality reduction by 50%:

Total number of images with at least 3 ZVMCUs:	140
Total number of images with no ZVMCUs:	360
Percentage of images with ZVMCUs:	28 %

Average number of ZVMCUs per image: 226

Image Category statistics after Quality reduction by 50%

Total number of images with at least 3 ZVMCUs: 500

Total number of images with no ZVMCUs: 0

Percentage of Images with ZVMCUs: 100%

Average number of ZVMCUs per image: 23603

Image Category 2: 500 images average size of 856.9 KB average resolution of 1.9 MP

Image Category statistics prior to Quality reduction by 50%:

Total number of images with at least 3 ZVMCUs: 206

Total number of images with no ZVMCUs: 294

Percentage of images with ZVMCUs: 41%

Average number of ZVMCUs per image: 222

Image Category statistics after Quality reduction by 50%

Total number of images with at least 3 ZVMCUs: 499

Total number of images with no ZVMCUs: 1

Percentage of Images with ZVMCUs: 99.8%

Average number of ZVMCUs per image: 3924

Image Category 3: 500 images average size of 115.0 KB average resolution of .30 MP

Image Category statistics prior to Quality reduction by 50%:

Total number of images with at least 3 ZVMCUs: 191

Total number of images with no ZVMCUs: 309

Percentage of images with ZVMCUs: 38.2%

Average number of ZVMCUs per image: 111

Image Category statistics after Quality reduction by 50%

Total number of images with at least 3 ZVMCUs: 486

Total number of images with no ZVMCUs: 14

Percentage of Images with ZVMCUs: 97.2%

Average number of ZVMCUs per image: 785

Image Category 4: 500 images average size of 2.97 MB average resolution of 6.3 MP

Image Category statistics prior to Quality reduction by 50%:

Total number of images with at least 3 ZVMCUs: 375

Total number of images with no ZVMCUs: 125

Percentage of images with ZVMCUs: 75.0%

Average number of ZVMCUs per image: 1274

Image Category statistics after Quality reduction by 50%

Total number of images with at least 3 ZVMCUs: 496

Total number of images with no ZVMCUs: 4

Percentage of Images with ZVMCUs: 99.6%

Average number of ZVMCUs per image: 13103

Image Category 5: 500 images average size of 553.0 KB average resolution of 2.3 MP

Image Category statistics prior to Quality reduction by 50%:

Total number of images with at least 3 ZVMCUs: 348

Total number of images with no ZVMCUs: 152

Percentage of images with ZVMCUs: 69.0%

Average number of ZVMCUs per image: 2415

Image Category statistics after Quality reduction by 50%

Total number of images with at least 3 ZVMCUs: 495

Total number of images with no ZVMCUs: 5

Percentage of Images with ZVMCUs: 99.0%

Average number of ZVMCUs per image: 8469

Image Category 6: 500 images average size of 133.6 KB average resolution of .49 MP

Image Category statistics prior to Quality reduction by 50%:

Total number of images with at least 3 ZVMCUs: 375

Total number of images with no ZVMCUs: 125

Percentage of images with ZVMCUs: 75.0%

Average number of ZVMCUs per image: 505

Image Category statistics after Quality reduction by 50%

Total number of images with at least 3 ZVMCUs: 459

Total number of images with no ZVMCUs: 41

Percentage of Images with ZVMCUs: 91.8%

Average number of ZVMCUs per image: 1509

Image Category 7: 500 images average size of 3.045 KB average resolution of .04 MP

Image Category statistics prior to Quality reduction by 50%:

Total number of images with at least 3 ZVMCUs: 85

Total number of images with no ZVMCUs: 415

Percentage of images with ZVMCUs: 17.0%

Average number of ZVMCUs per image: 3

Image Category statistics after Quality reduction by 50%

Total number of images with at least 3 ZVMCUs: 198

Total number of images with no ZVMCUs: 302

Percentage of Images with ZVMCUs: 39.7%

Average number of ZVMCUs per image: 5

Image Category 8: 430 images average size of 701 KB average resolution of 4.5 MP

Image Category statistics prior to Quality reduction by 50%:

Total number of images with at least 3 ZVMCUs: 408

Total number of images with no ZVMCUs: 22

Percentage of images with ZVMCUs: 94.9%

Average number of ZVMCUs per image: 3110

Image Category statistics after Quality reduction by 50%

Total number of images with at least 3 ZVMCUs: 430

Total number of images with no ZVMCUs: 0

Percentage of Images with ZVMCUs: 100%

Average number of ZVMCUs per image: 17389

Appendix G: Detail results for Test Five all image Categories

Test V: Image Size Increase to 120%

Program I: Gimp 2.6

Image Category I: 500 images with average size 2.2 MB

This experiment is designed to quantify the effects of the photo image editing software GIMP on the dataset of images. This increases the size of an image by 20%. The resulting image size and resolution will be larger but with reduced visual quality. This manipulation forces the software program to employ a fresh set of Discrete Cosine Transform quantization tables, which should cause modifications in the number and location of ZVMCUs in the image. This experiment is vitally important to determine how differing photo editing software programs might change the location, values and quantity of the zero variance minimal computer units in an image that will be used to provide the landmarks for further processing.

Image Category 1: 500 images average size 4.1 MB average resolution of 9.8 MP

Image Category statistics prior to Size increase by 20%:

Total number of images with at least 3 ZVMCUs:	140
Total number of images with no ZVMCUs:	360
Percentage of images with ZVMCUs:	28 %
Average number of ZVMCUs per image:	226

Image Category statistics after size increase by 20%

Total number of images with at least 3 ZVMCUs:	0
Total number of images with no ZVMCUs:	0

Percentage of Images with ZVMCUs: 0%

Average number of ZVMCUs per image: 0

Image Category 2: 500 images average size of 856.9 KB average resolution of 1.9 MP

Image Category statistics prior to size increase by 20%:

Total number of images with at least 3 ZVMCUs: 206

Total number of images with no ZVMCUs: 294

Percentage of images with ZVMCUs: 41%

Average number of ZVMCUs per image: 222

Image Category statistics after size increase by 20%

Total number of images with at least 3 ZVMCUs: 195

Total number of images with no ZVMCUs: 305

Percentage of Images with ZVMCUs: 39.0%

Average number of ZVMCUs per image: 325

Image Category 3: 500 images average size of 115.0 KB average resolution of .30 MP

Image Category statistics prior to size increase by 20%:

Total number of images with at least 3 ZVMCUs: 191

Total number of images with no ZVMCUs: 309

Percentage of images with ZVMCUs: 38.2%

Average number of ZVMCUs per image: 111

Image Category statistics after size increase by 20%

Total number of images with at least 3 ZVMCUs: 132

Total number of images with no ZVMCUs: 368

Percentage of Images with ZVMCUs: 26.4%

Average number of ZVMCUs per image: 89

Image Category 4: 500 images average size of 2.97 MB average resolution of 6.3 MP

Image Category statistics prior to size increase by 20%:

Total number of images with at least 3 ZVMCUs: 375

Total number of images with no ZVMCUs: 125

Percentage of images with ZVMCUs: 75.0%

Average number of ZVMCUs per image: 1274

Image Category statistics after size increase by 20%

Total number of images with at least 3 ZVMCUs: 334

Total number of images with no ZVMCUs: 166

Percentage of Images with ZVMCUs: 67.1%

Average number of ZVMCUs per image: 1375

Image Category 5: 500 images average size of 553.0 KB average resolution of 2.3 MP

Image Category statistics prior to size increase by 20%:

Total number of images with at least 3 ZVMCUs: 348

Total number of images with no ZVMCUs: 152

Percentage of images with ZVMCUs: 69.0%

Average number of ZVMCUs per image: 2415

Image Category statistics after size increase by 20%

Total number of images with at least 3 ZVMCUs: 260

Total number of images with no ZVMCUs: 240

Percentage of Images with ZVMCUs: 52.0%

Average number of ZVMCUs per image: 2440

Image Category 6: 500 images average size of 133.6 KB average resolution of .49 MP

Image Category statistics prior to size increase by 20%:

Total number of images with at least 3 ZVMCUs: 375

Total number of images with no ZVMCUs: 125

Percentage of images with ZVMCUs: 75.0%

Average number of ZVMCUs per image: 505

Image Category 1 statistics after size increase by 20%

Total number of images with at least 3 ZVMCUs: 274

Total number of images with no ZVMCUs: 226

Percentage of Images with ZVMCUs: 54.8%

Average number of ZVMCUs per image: 496

Image Category 7: 500 images average size of 3.045 KB average resolution of .04 MP

Image Category statistics prior to size increase by 20%:

Total number of images with at least 3 ZVMCUs: 85

Total number of images with no ZVMCUs: 415

Percentage of images with ZVMCUs: 17.0%

Average number of ZVMCUs per image: 3

Image Category statistics after size increase by 20%

Total number of images with at least 3 ZVMCUs: 60

Total number of images with no ZVMCUs: 440

Percentage of Images with ZVMCUs: 12.0%

Average number of ZVMCUs per image: 3

Image Category 8: 430 images average size of 701 KB average resolution of 4.5 MP

Image Category statistics prior to size increase by 20%:

Total number of images with at least 3 ZVMCUs: 408

Total number of images with no ZVMCUs: 22

Percentage of images with ZVMCUs: 94.9%

Average number of ZVMCUs per image: 3110

Image Category statistics after size increase by 20%

Total number of images with at least 3 ZVMCUs: 377

Total number of images with no ZVMCUs: 53

Percentage of Images with ZVMCUs: 87.7%

Average number of ZVMCUs per image: 2929

Appendix H: Detail results for Test Six all image Categories

Test VI: Test I image results Quality restoration to 100%

Program I: Gimp 2.6

Image Category I: 500 images with average size 2.2 MB

This experiment is designed to quantify the effects of the photo image editing software GIMP on the dataset of images. This test attempts to restore the quality of each picture in the data set from 80 back to 100. As a result of the change the program will process the image using its unique JPEG Compression routine, its own Discrete Cosine Transform quality tables, and Huffman coding tables. This experiment is vitally important because the default setting on most image editing software is 100% quality. Therefore, if a previously edited image is opened and saved in image editing software, this type of manipulation will occur without any further action by the user. This will also indicate if software programs might change the location, values and quantity of the zero variance minimal computer units in an image that will be used to provide the landmarks for further processing.

Image Category 1: 500 images average size 4.1 MB average resolution of 9.8 MP

Image Category statistics prior to Quality restoration to 100%:

Total number of images with at least 3 ZVMCUs:	498
Total number of images with no ZVMCUs:	2
Percentage of images with ZVMCUs:	99.6 %
Average number of ZVMCUs per image:	2229

Image Category 1 statistics after Quality restoration to 100%

Total number of images with at least 3 ZVMCUs: 497

Total number of images with no ZVMCUs: 3

Percentage of Images with ZVMCUs: 99.6%

Average number of ZVMCUs per image: 2237

Image Category 2: 500 images average size of 856.9 KB average resolution of 1.9 MP

Image Category statistics prior to Quality restoration to 100%:

Total number of images with at least 3 ZVMCUs: 433

Total number of images with no ZVMCUs: 67

Percentage of images with ZVMCUs: 86.6%

Average number of ZVMCUs per image: 632

Image Category 2 statistics after Quality restoration to 100%

Total number of images with at least 3 ZVMCUs: 433

Total number of images with no ZVMCUs: 167

Percentage of Images with ZVMCUs: 86.6%

Average number of ZVMCUs per image: 633

Image Category 3: 500 images average size of 115.0 KB average resolution of .30 MP

Image Category statistics prior to Quality restoration to 100%:

Total number of images with at least 3 ZVMCUs: 387

Total number of images with no ZVMCUs: 113

Percentage of images with ZVMCUs: 77.4%

Average number of ZVMCUs per image: 226

Image Category 1 statistics after Quality restoration to 100%

Total number of images with at least 3 ZVMCUs: 387

Total number of images with no ZVMCUs: 113

Percentage of Images with ZVMCUs: 77.4%

Average number of ZVMCUs per image: 226

Image Category 4: 500 images average size of 2.97 MB average resolution of 6.3 MP

Image Category statistics prior to Quality restoration to 100%:

Total number of images with at least 3 ZVMCUs: 498

Total number of images with no ZVMCUs: 2

Percentage of images with ZVMCUs: 96.2%

Average number of ZVMCUs per image: 2558

Image Category 1 statistics after Quality restoration to 100%

Total number of images with at least 3 ZVMCUs: 479

Total number of images with no ZVMCUs: 21

Percentage of Images with ZVMCUs: 96.2%

Average number of ZVMCUs per image: 2559

Image Category 5: 500 images average size of 553.0 KB average resolution of 2.3 MP

Image Category statistics prior to Quality restoration to 100%:

Total number of images with at least 3 ZVMCUs: 436

Total number of images with no ZVMCUs: 64

Percentage of images with ZVMCUs: 87.2%

Average number of ZVMCUs per image: 3635

Image Category 1 statistics after Quality restoration to 100%

Total number of images with at least 3 ZVMCUs: 436

Total number of images with no ZVMCUs: 64

Percentage of Images with ZVMCUs: 87.2%

Average number of ZVMCUs per image: 3635

Image Category 6: 500 images average size of 133.6 KB average resolution of .49 MP

Image Category statistics prior to Quality restoration to 100%:

Total number of images with at least 3 ZVMCUs: 400

Total number of images with no ZVMCUs: 100

Percentage of images with ZVMCUs: 80.0%

Average number of ZVMCUs per image: 664

Image Category statistics after Quality restoration to 100%

Total number of images with at least 3 ZVMCUs: 400

Total number of images with no ZVMCUs: 100

Percentage of Images with ZVMCUs: 80.0%

Average number of ZVMCUs per image: 664

Image Category 7: 500 images average size of 3.045 KB average resolution of .04 MP

Image Category statistics prior to Quality restoration to 100%:

Total number of images with at least 3 ZVMCUs: 110

Total number of images with no ZVMCUs: 390

Percentage of images with ZVMCUs: 22.0%

Average number of ZVMCUs per image: 3

Image Category 1 statistics after Quality restoration to 100%

Total number of images with at least 3 ZVMCUs: 110

Total number of images with no ZVMCUs: 390

Percentage of Images with ZVMCUs: 22.0%

Average number of ZVMCUs per image: 3

Image Category 8: 430 images average size of 701 KB average resolution of 4.5 MP

Image Category statistics prior to Quality restoration to 100%:

Total number of images with at least 3 ZVMCUs: 428

Total number of images with no ZVMCUs: 2

Percentage of images with ZVMCUs: 99.5%

Average number of ZVMCUs per image: 3963

Image Category statistics after Quality restoration to 100%

Total number of images with at least 3 ZVMCUs: 428

Total number of images with no ZVMCUs: 2

Percentage of Images with ZVMCUs: 99.5%

Average number of ZVMCUs per image: 3964

Appendix I: Detail results for Test Seven all image Categories

Test VII: Test V image results Quality restoration to 100%

Program I: Gimp 2.6

Image Category I: 500 images with average size 2.2 MB

This experiment is designed to quantify the effects of the photo image editing software GIMP on the dataset of images. This test attempts to restore the quality of each picture in the data set from 50 back to 100. Due to the lossy compression used in the JPEG routine, it is not possible to restore the image to the original quality. However, the image editing software will use the high quality quantization tables as a result of the request to save the image at 100% quality. The use of these tables differ from the ones stored with the image may cause some changes in how the image is finally processed and stored. This experiment is fundamental to learning the effects of quality restoration attempts. This experiment provides indicators of software programs tendency to change the location, values and quantity of the zero variance minimal computer units in an image.

Image Category 1: 500 images average size 4.1 MB average resolution of 9.8 MP

Image Category statistics prior to Quality restoration to 100%:

Total number of images with at least 3 ZVMCUs:	500
Total number of images with no ZVMCUs:	0
Percentage of images with ZVMCUs:	100 %
Average number of ZVMCUs per image:	23603

Image Category 1 statistics after Quality restoration to 100%

Total number of images with at least 3 ZVMCUs: 500

Total number of images with no ZVMCUs: 0

Percentage of Images with ZVMCUs: 100%

Average number of ZVMCUs per image: 31527

Image Category 2: 500 images average size of 856.9 KB average resolution of 1.9 MP

Image Category statistics prior to Quality restoration to 100%:

Total number of images with at least 3 ZVMCUs: 499

Total number of images with no ZVMCUs: 1

Percentage of images with ZVMCUs: 99.8%

Average number of ZVMCUs per image: 3924

Image Category 2 statistics after Quality restoration to 100%

Total number of images with at least 3 ZVMCUs: 475

Total number of images with no ZVMCUs: 25

Percentage of Images with ZVMCUs: 95.0%

Average number of ZVMCUs per image: 2166

Image Category 3: 500 images average size of 115.0 KB average resolution of .30 MP

Image Category statistics prior to Quality restoration to 100%:

Total number of images with at least 3 ZVMCUs: 486

Total number of images with no ZVMCUs: 14

Percentage of images with ZVMCUs: 97.2%

Average number of ZVMCUs per image: 785

Image Category 1 statistics after Quality restoration to 100%

Total number of images with at least 3 ZVMCUs: 486

Total number of images with no ZVMCUs: 14

Percentage of Images with ZVMCUs: 97.2%

Average number of ZVMCUs per image: 785

Image Category 4: 500 images average size of 2.97 MB average resolution of 6.3 MP

Image Category statistics prior to Quality restoration to 100%:

Total number of images with at least 3 ZVMCUs: 496

Total number of images with no ZVMCUs: 4

Percentage of images with ZVMCUs: 99.6%

Average number of ZVMCUs per image: 13103

Image Category 1 statistics after Quality restoration to 100%

Total number of images with at least 3 ZVMCUs: 496

Total number of images with no ZVMCUs: 4

Percentage of Images with ZVMCUs: 99.6%

Average number of ZVMCUs per image: 13103

Image Category 5: 500 images average size of 553.0 KB average resolution of 2.3 MP

Image Category statistics prior to Quality restoration to 100%:

Total number of images with at least 3 ZVMCUs: 495

Total number of images with no ZVMCUs: 5

Percentage of images with ZVMCUs: 99.0%

Average number of ZVMCUs per image: 8469

Image Category 1 statistics after Quality restoration to 100%

Total number of images with at least 3 ZVMCUs: 495

Total number of images with no ZVMCUs: 5

Percentage of Images with ZVMCUs: 99.0%

Average number of ZVMCUs per image: 8469

Image Category 6: 500 images average size of 133.6 KB average resolution of .49 MP

Image Category statistics prior to Quality restoration to 100%:

Total number of images with at least 3 ZVMCUs: 459

Total number of images with no ZVMCUs: 41

Percentage of images with ZVMCUs: 91.8%

Average number of ZVMCUs per image: 1509

Image Category 1 statistics after Quality restoration to 100%

Total number of images with at least 3 ZVMCUs: 459

Total number of images with no ZVMCUs: 41

Percentage of Images with ZVMCUs: 91.8%

Average number of ZVMCUs per image: 1509

Image Category 7: 500 images average size of 3.045 KB average resolution of .04 MP

Image Category statistics prior to Quality restoration to 100%:

Total number of images with at least 3 ZVMCUs: 198

Total number of images with no ZVMCUs: 302

Percentage of images with ZVMCUs: 39.7%

Average number of ZVMCUs per image: 5

Image Category 1 statistics after Quality restoration to 100%

Total number of images with at least 3 ZVMCUs: 198

Total number of images with no ZVMCUs: 302

Percentage of Images with ZVMCUs: 39.7%

Average number of ZVMCUs per image: 5

Image Category 8: 430 images average size of 701 KB average resolution of 4.5 MP

Image Category statistics prior to Quality restoration to 100%:

Total number of images with at least 3 ZVMCUs: 430

Total number of images with no ZVMCUs: 0

Percentage of images with ZVMCUs: 100%

Average number of ZVMCUs per image: 17389

Image Category 1 statistics after Quality restoration to 100%

Total number of images with at least 3 ZVMCUs: 430

Total number of images with no ZVMCUs: 0

Percentage of Images with ZVMCUs: 100%

Average number of ZVMCUs per image: 17389

Appendix J: Detail results for Test Eight all image Categories

Test 8: No manipulation Save As only

Program V: Image Save.py

This experiment is designed to quantify the effects of renaming and relocating an image on the quantity and location of zero variance minimal computer units located in the image. This test opens each image, renames the image and saves it to a new directory. The resulting image will be analysed to determine the change in the number and location of all ZVMCUs in the image. This experiment is vitally important due to the fact that it is the most common operation that will be performed on an image file.

Image Category 1: 500 images average size 4.1 MB average resolution of 9.8 MP

Image Category statistics prior to Save As Only:

Total number of images with at least 3 ZVMCUs:	140
Total number of images with no ZVMCUs:	360
Percentage of images with ZVMCUs:	28 %
Average number of ZVMCUs per image:	226

Image Category statistics after Save As Only

Total number of images with at least 3 ZVMCUs:	140
Total number of images with no ZVMCUs:	360
Percentage of Images with ZVMCUs:	28.0%
Average number of ZVMCUs per image:	276

Image Category 2: 500 images average size of 856.9 KB average resolution of 1.9 MP

Image Category statistics prior to Save As Only:

Total number of images with at least 3 ZVMCUs:	206
Total number of images with no ZVMCUs:	294
Percentage of images with ZVMCUs:	41%
Average number of ZVMCUs per image:	222

Image Category statistics after Save As Only

Total number of images with at least 3 ZVMCUs:	284
Total number of images with no ZVMCUs:	216
Percentage of Images with ZVMCUs:	56.8%
Average number of ZVMCUs per image:	1031

Image Category 3: 500 images average size of 115.0 KB average resolution of .30 MP

Image Category statistics prior to Save As Only:

Total number of images with at least 3 ZVMCUs:	191
Total number of images with no ZVMCUs:	308
Percentage of images with ZVMCUs:	38.2%
Average number of ZVMCUs per image:	111

Image Category statistics after Save As Only

Total number of images with at least 3 ZVMCUs:	408
Total number of images with no ZVMCUs:	92
Percentage of Images with ZVMCUs:	81.6%

Average number of ZVMCUs per image: 324

Image Category 4: 500 images average size of 2.97 MB average resolution of 6.3 MP

Image Category statistics prior to Save As Only:

Total number of images with at least 3 ZVMCUs: 375

Total number of images with no ZVMCUs: 125

Percentage of images with ZVMCUs: 75.0%

Average number of ZVMCUs per image: 1274

Image Category statistics after Save As Only

Total number of images with at least 3 ZVMCUs: 483

Total number of images with no ZVMCUs: 17

Percentage of Images with ZVMCUs: 96.6%

Average number of ZVMCUs per image: 3946

Image Category 5: 500 images average size of 553.0 KB average resolution of 2.3 MP

Image Category statistics prior to Save As Only:

Total number of images with at least 3 ZVMCUs: 348

Total number of images with no ZVMCUs: 152

Percentage of images with ZVMCUs: 69.0%

Average number of ZVMCUs per image: 2415

Image Category statistics after Save As Only

Total number of images with at least 3 ZVMCUs: 456

Total number of images with no ZVMCUs: 44

Percentage of Images with ZVMCUs: 91.2%

Average number of ZVMCUs per image: 4372

Image Category 6: 500 images average size of 133.6 KB average resolution of .49 MP

Image Category statistics prior to Save As Only:

Total number of images with at least 3 ZVMCUs: 375

Total number of images with no ZVMCUs: 125

Percentage of images with ZVMCUs: 75.0%

Average number of ZVMCUs per image: 505

Image Category statistics after Save As Only

Total number of images with at least 3 ZVMCUs: 384

Total number of images with no ZVMCUs: 116

Percentage of Images with ZVMCUs: 76.8%

Average number of ZVMCUs per image: 812

Image Category 7: 500 images average size of 3.045 KB average resolution of .04 MP

Image Category statistics prior to Save As Only:

Total number of images with at least 3 ZVMCUs: 85

Total number of images with no ZVMCUs: 415

Percentage of images with ZVMCUs: 17.0%

Average number of ZVMCUs per image: 3

Image Category statistics after Save As Only

Total number of images with at least 3 ZVMCUs: 91

Total number of images with no ZVMCUs: 409

Percentage of Images with ZVMCUs: 18.2%

Average number of ZVMCUs per image: 3

Image Category 8: 430 images average size of 701 KB average resolution of 4.5 MP

Image Category statistics prior to Save As Only:

Total number of images with at least 3 ZVMCUs:	408
Total number of images with no ZVMCUs:	22
Percentage of images with ZVMCUs:	94.9%
Average number of ZVMCUs per image:	3110

Image Category statistics after Save As Only

Total number of images with at least 3 ZVMCUs:	427
Total number of images with no ZVMCUs:	3
Percentage of Images with ZVMCUs:	99.3%
Average number of ZVMCUs per image:	4280

Appendix K: Source code: Zero Variance Minimal Computer Unit Locator

```
from PIL import Image
from PIL import ImageFile
from math import *
import os

‘’’This program searches a dataset called IMCATs (image categories)of JPEG images
looking for zero variance Minimal
computer units (MCUS) within each image.  A ZV MCU is one were all 64 pixels
in an 8X8 block are identical.  This will be useful in finding Visually similar
but digitally different JPEG images within a dataset.’’’
''' 15 Jan 10 created color hash module: this module will find MCUs that are not Zero
Variance but
#which are close'''
'''5 feb  create3d the all values module that calculates the total number of ZVMCUs and
their respective
pixel values from all the ZVMCUs in the dataset'''


'''ZVsums module accumulates the total times a particular pixel value occurs
as an ZVMCU value in a data set.  Thie data will be useful to understand
the occurrence or lack of ZVMCUs in an image
this module is called from the xxyz module and takes in as a parameter a tuple value
that represents the pixel value of an ZVMCU
this module returns the accumulated values in dictionary form'''


allValues = {}
def ZVsums(*zvvalues):
 #maintains a running tally of zvMCUs by color "allValues"
 #in the form of ((RGB), number) accumulated for entire database.
    if zvvalues in allValues:
        #increment count
        x = allValues[zvvalues]
        x = x +1
        allValues[zvvalues] = x
    else:
        #add zvvalues to allvalues with count of one
        allValues[zvvalues]=1
```

```
    return allValues

def colorhash(pixels):
    # Creates a singlar number for comparison of similarity of MCU's
    # all values from a each color band are sumed, Each sum is then
    #incremented by one, to prevent multiplication by zero,  The product of
    # all the sums is divided by the total number of individual values.
    #resulting in a nearly unique number value for a particular MCU.

    cr = sum ((r for (r,g,b) in pixels))
    cg = sum ((g for (r,g,b) in pixels))
    cb = sum ((b for (r,g,b) in pixels))
    return ((cr+1) * (cg+1) * (cb+1) / (len(pixels)**3))


def zeroVarMcu (image,mcu_size,i,sets):
    # module called from main, takes in an image, the mcu size and image number
    #returns a vector of zero variance mcus to the main
    #calls the writeOut module to create a text file of values of zvmcus

    parts = []
    ZVmcu = []
    zvvalue=[]
    t = 0 #counter for number of zvmcus in an image
    w , h = image.size
    #control line for managing how much image is processed

        for y in range(0,h,mcu_size):
                for x in range(0,w,mcu_size):
                    c = 0
                    num =0
                    mcu=image.crop((x,y,x +mcu_size,y+mcu_size))# MCU extraction
                    values=list(mcu.getdata())# all 64 RGB values in mcu
                    while c < 64:
                        if values[0]==values[c]: #compares each value in the MCU to the next
                            num=num+1           #if values are equal continue in vector
                            c=c+1               #if values are not equal then not a ZVMCU
                            if num == 64:  # if num = 64 then all values are equal and it is a
ZVMCU
                              ZVmcu.append((x,y,x+mcu_size,y+mcu_size,mcu)) #Create a vector
of ZVMCU
                              zvvalue = values[0]
                              t=t+1
                              writeOut(zvvalue,x,y,t,i,sets)
                              ZVsums(zvvalue)# maintains a tally of ZVmcus by color
```

```
                else:
                    c = 64


    return ZVmcu
#*********************
def equalparts(imagparts):
        print"eq parts"
        dupl = []
        for i in range(len(imagparts)-1):
            acc1, x1,y1,im1 = imagparts[i]
            acc2,x2,y2,im2  =imagparts[i+1]
            if acc1==acc2==0:
                diff=0.0
            else:
                diff = 100.0 * (1.0 -float(acc1)/acc2)# creates % of similarity
                col = 0.5

                if diff > col:
                    if imagparts[i] not in dupl:
                        dupl.append(imagparts[i])
                        if imagparts[i+1] not in dupl:
                            dupl.append(imagparts[i+1])
        return dupl


def markZVMcu(OrigIM,mcus):
    #this module blends markers onto the original image to show the location of the
zvmcus
    #this module returns a marked up copy of the original image

        if mcus:
            marker = Image.new('RGB',mcus[0][4].size,(150,150,150))
            for (x,y,z,t,im) in mcus:
                        img = Image.blend(im,marker,1.5)
                        OrigIM.paste(img,(x,y))
            return OrigIM


def createpath(path):
    #creates a file path for output of the data from various modules
    if not os.path.isdir(path):
        os.mkdir(path)
```

```
def writeOut(zvmcu,x,y,t,i,s):
   #called from zerovarmcu module
   #takes in zvmcu: the RGB value of the mcu
     #x,y: the coordinates of the mcu
     #t: the count of the mcu in the image
     #i: The count for the image being processed, used to create unique filename
     #s: The count for the dataset being processed, used to create unique filename
   #writes to a file the data calculated by various modules

 Writer=open('c:/zvmcu/2IMCat%s/C%s_image%i_RGBdetails.txt'%(s,s,i),'a')#writes to
the 'zvmcu folder, creates a file for each
   # image in the dataset, each file contains ZVMCU number, Location, and value for
every ZVMCU in the image.
   Writer.write('ZVMCU ' + `t`+' (' +`x` + ', ' + `y`+')' + `zvmcu` + '\n')
   Writer.close()



def filedetails(w,h):
   #used to create a separate file of the meta/information about each image file.
   #takes in the attributes for the image to be recorded and creates a text file.
   Writer = open('c:/zvmcu/2IMCAT%s/IMSizeCat%s.txt'%(sets,sets),'a')
   Writer.write('image%i size='%num +`w`+ ' x ' + `h` + '\n')
   Writer.close()


   ##  MAIN  ====== MAIN ===== MAIN ===== MAIN

if __name__ == "__main__":

     sets=1
   while sets < 2:
      createpath('c:/zvmcu/2IMCat%s'%sets)#creates the folders for the storage of the
image category processing

      # path = ('d:\zvmcu\IMtest%s'%sets)#creates userfriendly varible for newly created
path
       num = 1
      print "Image test %i"%sets+" is Processing"

      while num<501:

         im=Image.open
("c:/PhDfiles/THUMBDRIVE_COPY/Doctorate_files/Photo_database/IMCat%i/image%
i.jpg" %(sets,num))
         sizw = im.size[0]
```

```
        sizh = im.size[1]

        filedetails(sizw,sizh)

        lsMcus=zeroVarMcu(im,8,num,sets)
        l=len(lsMcus)
        print "Cat%i Image %i" %(sets,num)
        Writer=open('c:/zvmcu/2IMCAT%s/ZVcountcat%s.txt'%(sets,sets),'a')
        Writer.write('image %s ZV MCUs = '%num + `l` + '\n')
        Writer.close()
        num=num+1

sets=sets+1
```

Appendix L: Source Code: ZVMCU results

```
from math import *
import os
from PIL import Image
from PIL import ImageFile
import time
import manipResultsTally
'''this program will analyze the resulting images from the 7 separate manipulation test that were
performed to change the JPEG compression routines and alter the DCT starting point for each image
the intent its to determine how the manipulation changes the predetermined number of ZVMCUs that were
discovered using the ZVMCUfinderOK program. by wallking through the results of each test and outputing the results to a
folder/text file'''
''' added timer to time the run of each Image category, 18 Mar 09'''
''' zeroVarMCU module is used in 2 programs ZVMCUfinderOK and Results analysis'''

def zeroVarMcu (image,mcu_size,i,sets,z):
    # module called from main, takes in an image, the mcu size and image number
    #returns a vector of zero variance mcus to the main
    #calls the writeOut module to create a text file of values of zvmcus
    #
    colors = []
    ZVmcu = [] # maintains a vector of start and end coordinates of each ZVMCU found
    zvvalue=[]# maintains a vector of the RGB color of each ZVMCU found
    t = 0 #counter for number of zvmcus in an image
    w , h = image.size
    #control line for managing how much image is processed
    #w = 24
    #h = 24

    for y in range(0,h,mcu_size):
            for x in range(0,w,mcu_size):
                c = 0
                num =0
                mcu=image.crop((x,y,x +mcu_size,y+mcu_size))# MCU extraction
                values=list(mcu.getdata())# all 64 RGB values in mcu
```

```
            while c < 64:
                if values[0]==values[c]: #compares each value in the MCU to the next
                    num=num+1           #if values are equal continue in vector
                    c=c+1               #if values are not equal then not a ZVMCU
                    if num == 64:  # if num = 64 then all values are equal and it is a
ZVMCU
                      ZVmcu.append((x,y,x+mcu_size,y+mcu_size,mcu)) #Create a vector
of ZVMCU (sx,sy,ex,ey) coordinates
                      zvvalue = values[0]  #RGB values of the ZVMCUs one per-zvmcu
                      t=t+1

                else:
                    c = 64

    '''writer.open('c:/programResults/IMCAT7/program1/results/colors.txt') #change
'program number as programs process'
    writer.write('colors =' + `colors`)
    writer.close()'''
    return ZVmcu # ((sx,sy,ex,ey),(sx,sy,ex,ey),(sx,sy,ex,ey),(sx,sy,ex,ey)....)

'''Write out module'''
def writeDetails(rgb,x,y,t,i,s,z):
    #called from zerovarmcu module
    #takes in zvmcu: the RGB value of the mcu
      #x,y: the coordinates of the mcu
      #t: the count of the mcu in the image
      #i: The count for the image being processed, used to create unique filename
      #s: The count for the dataset being processed, used to create unique filename
      #z:  the count for which test results
    #writes to a file the data calculated by various modules
    # image in the dataset, each file contains ZVMCU number, Location, and value for
every ZVMCU in the image.
    #R,G,B = zvmcu
    #pixel = (R,G,B)
    #Writer.write(`R`) #for writing individual values not needed for now
    #Writer.write(`G`)
    #Writer.write(`B`)
    #writes to the 'zvmcu folder, creates a file for each

    Writer=open('c:/programresults/IMCAT%i/program1/results/test%s/dataset%s/image
%i_detail.txt'%(IMCatnum,z,s,i),'a')
    Writer.write('ZVMCU ' + `t`+' (' +`x` + ', ' + `y`+')' + `rgb` + '\n')
    Writer.close()

'''ZVsums module called from ZVMCU finder module'''
allValues = {}
```

```
def ZVsums(*zvvalues):
 #maintains a running tally of zvMCUs by color "allValues"
 #in the form of ((RGB), number) accumulated for entire database.
    if zvvalues in allValues:
        #increment count
        x = allValues[zvvalues]
        x = x +1
        allValues[zvvalues] = x
    else:
        #add zvvalues to allvalues with count of one
        allValues[zvvalues]=1
       return allValues

"'file detail module'"
def filedetails(num,w,h):
    #used to create a separate file of the meta/information about each image file.
    #takes in the attributes for the image to be recorded and creates a text file.
    #Writer = open('c:/Georges/zvmcu2/IMCat1/Alltest1s.txt','a')
    Writer = open ('c:/programResults/IMCAT%i/program1/results/alldetails.txt'
%IMCatnum,'a')
    Writer.write('image %i size='%num +`w`+ ' x ' + `h` + '\n')
    Writer.close()

  ##  MAIN  ====== MAIN ===== MAIN ===== MAIN
"' image category number between test runs. this number is input  by the user at the start
May also alter the number of test that are being analyzed, if the number exceeds 7.'"
if __name__ == "__main__":

    IMCatnum = int(raw_input ( "Enter the Image Category you wish to process (1-8)"))
    test = int(raw_input ("Enter the testing start point (1-7)"))
    teststop = int(raw_input ("Enter testing stop point (2-8)"))
     start = time.time()
    print start
    #test = 6    #7 current tests
    while test < teststop:
        dataset = 1  # sets the number for the datasets of images to run (datasets 1-6)
        while dataset <7 :
            num = 1 #sets the image number for beginning of each dataset run.
            while num < 501:
                try:
                    im =
Image.open('c:/programResults/IMCAT%i/program1/test%u/dataset%u/image%u.jpg'%(I
MCatnum,test,dataset,num))
                     #  the image category is made as user input
                    sizw = im.size[0]
                    sizh = im.size[1]
```

```
                filedetails(num,sizw,sizh)#calls the file detail module to record basic,
size/resolution data about image

                lsMcus=zeroVarMcu(im,8,num,dataset,test)#calls the ZeroVarMCU module;
returns tuple of image data
                l=len(lsMcus)

Writer=open('c:/programResults/IMCAT%i/program1/results/test%s/dataset%s/ZVmcuC
ount.txt'%(IMCatnum,test,dataset),'a')
                Writer.write('image %s ZV MCUs = '%num + `l` + '\n')
                Writer.close()
                print "DS%i Image %i" %(dataset,num)

            except IOError:
                pass
            num = num + 1
        print 'dataset %s' %dataset
        dataset = dataset +1
      test=test + 1
    print 'test %s' %test
    finish= time.time()
    elapsed = finish - start
    minutes = elapsed/60
    print 'this run took ', elapsed
    answer = raw_input("do you wish to run talley?")
    if answer == 'yes':
        manipResultsTally()
    else:
        print 'goodbye'
```

Appendix M: Source Code for ZVHashComparer.py

```
from math import *
import os
import shutil
import string
import hashlib

''' 23 APRIL::Hashes and compares the ZVMCU/Coord pair of each image in each test folder to
same image in other test folders.
This program uses the MD5 hash values of the ZVMCU details found within
an image file and creates a probability of image match, using the hash value of the
ZVMCU value and coordinate pairs. The sister programs, ZVHasher and ZVHasherfalsepos are
similar but working
with different fils'''
'''def LineHasher() opens the text files and reads the necessary lines to feed
the hashlib module, creates a vector of message digest for each rgb-coordinate pair'''

def TSHasher(cat,test,num):
    strhash=""
    hashV=0
    try:
        file =
open('C:/programResults/TestsetCoords/Cat%iRGB/TS%iRGB/image%i_detail.txt'%(cat,test,nu
m))
        while 1:
            hashvalue=[]
            line = file.readline()
            if not line:
                break
            if (line.find('0, 0, 0') >1 ):
                continue
            if (line.find('255, 255, 255')>1):
                continue
            size = len(line)
            breakup = string.split(line)
            x = 2
            while x < 6:
                hashvalue.append(breakup[x])
                x = x + 1
```

```
            m = hashlib.md5()
            m.update(strhash.join(hashvalue))
            hashV = m.hexdigest()
            tshash.append(hashV)

    except IOError:
        pass
    except IndexError:
        print "oops Black & White one"
        file.close()

def TSAHasher(cat,test,num):
    strhash =""
    hashV=0

    try:
file = open('C:/programResults/TestsetCoords/Cat%iRGB/TS%iRGB/image%i_detail.txt'
%(cat,test,num))
        while 1:
            hashvalue=[]
            line = file.readline()
            if not line:
                break
            if (line.find('0, 0, 0')> 1):
                continue
            if (line.find('255, 255, 255')>1):
                continue
            size = len(line)
            breakup = string.split(line)
            x = 2
            while x < 6:
                hashvalue.append(breakup[x])
                x = x + 1

            m = hashlib.md5()
            m.update(strhash.join(hashvalue))
            hashV = m.hexdigest()
            tsAhash.append(hashV)

    except IOError:
        pass
    except IndexError:
        print "oops Black & White one"
        file.close()


def returnMatches(a,b):
    return list(set(a) & set(b))
```

```
""def Hasher(I) takes in the text string read from the images detail
text file, and returns the MD5 hash digest for that string"'
def Hasher(value):
    m = hashlib.md5()
    m.update(value)
    tshash = m.hexdigest()
    return imhash

##  MAIN  ====== MAIN ===== MAIN ===== MAIN

if __name__ == "__main__":
    #the folder is currently located in the \\programResults\Imcat#\test#\dataset#\zvmcuscount.txt
file
    #cycle through the IMCat# folder 1-8: open the text file
    #extract the image# and ZVMCUs, then copy that information into a separate text file

    print "this program hashes the RGB Coordinate pair of each ZVMCU found in an image."
    print "The results are stored and compared with other images in the database"
    raw_input("press enter to continue")
    catagory = 1
    while catagory < 9:
        num = 1
        print "catagory%i" %catagory
        while num<501:
            test = 1
            while test < 8:
                testA = test + 1
                while testA < 8:
                    #print "Cat %i Test %i im %i is Processing" %(catagory,test,num)
                    match = []
                    tshash = []
                    tsAhash = []
                    TSHasher(catagory,test,num)
                    TSAHasher(catagory,testA,num)
                    match = returnMatches(tshash,tsAhash)
                    tsA=len(tsAhash)
                    ts=len(tshash)
                    ma=len(match)
  Writer = open('c:/programResults/hashresults/AllTests/C%iHashNoLimits.txt'%(catagory),'a')
                    Writer.write('Image %iTestSet%i= '%(num,test) + `ts` +'\t'+ 'TestSet%i= '%testA +
`tsA`+'\t'+'\t' + 'Matches = '+`ma`  + '\n')
                    Writer.close()
                    testA = testA +1
                test = test + 1
            num = num + 1
        catagory = catagory +1
```

Appendix N: Source Code for Mover.py

```
import os
from PIL import Image
from PIL import ImageFile
'''this program is a specific program for test8. This program opens each image in every
category then renames them and saves them to a new directory. Because of the JPEG
compression routine is implemented when the image is saved to a new location, any MD5
or SHA1 hash is as a means for matching these images.'''
''' added timer to time the run of each Image category, 18 Mar 09'''
 ## MAIN ====== MAIN ===== MAIN ===== MAIN
if __name__ == "__main__":

   IMCatnumstart = int(raw_input ( "Enter the Image Catagory start point you wish to
process (1-8)"))
   IMCatnumstop = int(raw_input ("Enter the IM cat stop point (2-9)"))

   while IMCatnumstart<IMCatnumstop: #the number of image categories to evaluate
can be input by the user as a range from 1-8.

      dataset = 1  # sets the number for the datasets of images to run (datasets 1-6)
      while dataset <7 : # there are 6 datasets per image category in the experimental
database
         num = 1 #sets the image number for beginning of each dataset run.
         while num < 501: #each image category contains 500 JPG images.

            try:
               im = Image.open('c:/zvmcu/IMCat%i/dataset%i/image%i.jpg
'%(IMCatnumstart,dataset,num))
im.save('c:/programResults/IMCAT%i/program1/test8/dataset%i/NMimage%i.jpg'%(IM
Catnumstart,dataset,num))
            except IOError:
               print '_********************** Failed print image # %i  *****' % num
               pass
            num = num + 1
         print 'dataset %s' %dataset
         dataset = dataset +1
      print "image cat%i complete in" %IMCatnumstart
      IMCatnumstart = IMCatnumstart + 1
```

Appendix O: Source Code for FalsePositveTest.py

```
from math import *
import os
import shutil
import string
import hashlib

''' 10 May 10 : FalsePositiveTest.py Compares the coordinate/ZVMCU values of dissimilar
images in two categories of images.
input from user is required.  user selects the categories of dissimilar images 1-8, and inputs the
acceptable saturation rate from
1= no saturated ZVMCUs allowed to 50000 Saturated ZVMCUs allowed.
This program uses the MD5 hash values of the ZVMCU details found within
an image file looking for false positve matching of images using the same techniques as used in
other porgrams. Hashing the ZVMCU value and coordinate pairs. The sister programs, ZVHasher
and ZVHashertest2test are similar but working
with different files'''

'''def LineHasher() opens the text files and reads the necessary lines to feed
the hashlib module, creates a vector of message digest for each rgb-coordinate pair'''

def SatTester(cat,test,num,sat):
    strhash=""
    hashV=0
    satcount=0
    try:
        file = open('C:/programResults/TestsetCoords/Cat%iRGB/TS%iRGB/image%i_detail.txt'%(cat,test,num))

        while 1:
            line = file.readline()
            if not line:
                break
            if (line.find('0, 0, 0') >1 ):
                satcount = satcount +1
                continue
            if (line.find('255, 255, 255')>1):
                satcount = satcount +1
                continue
        file.close()
    except IOError:
        pass
    except IndexError:
        print "oops Black & White one"
        file.close()
```

```
    if satcount>sat:
        return -1
    if satcount<sat:
        return 1
def TSHasher(cat,test,num):
    strhash=""
    hashV=0

    try:
        file = open('C:/programResults/TestsetCoords/Cat%iRGB/TS%iRGB/image%i_detail.txt'%(cat,test,num))

        while 1:
            hashvalue=[]
            line = file.readline()
            if not line:
                break
            '''if (line.find('0, 0, 0') >1 ):
                continue
            if (line.find('255, 255, 255')>1):
                continue'''
            size = len(line)
            breakup = string.split(line)
            x = 2
            while x < 6:
                hashvalue.append(breakup[x])
                x = x + 1
            m = hashlib.md5()
            m.update(strhash.join(hashvalue))
            hashV = m.hexdigest()
            tshash.append(hashV)

    except IOError:
        pass
    except IndexError:
        print "oops Black & White one"
        file.close()


def TSAHasher(cat,test,num):
    strhash =""
    hashV=0

    try:
        file = open('C:/programResults/TestsetCoords/Cat%iRGB/TS%iRGB/image%i_detail.txt'
%(cat,test,num))
        while 1:
            hashvalue=[]
            line = file.readline()
            if not line:
                break
            '''if (line.find('0, 0, 0') >1 ):
```

```
            continue
         if (line.find('255, 255, 255')>1):
            continue'"
         size = len(line)
         breakup = string.split(line)
         x = 2
         while x < 6:
            hashvalue.append(breakup[x])
            x = x + 1

         m = hashlib.md5()
         m.update(strhash.join(hashvalue))
         hashV = m.hexdigest()
         tsAhash.append(hashV)
   except IOError:
      pass
    except IndexError:
       print "oops Black & White one"
       file.close()


def returnMatches(a,b):
      return list(set(a) & set(b))

'"opens the zvmcu details text files and extracts out the ZVMCU details of
RGB & coordinate vaules stored by the ZVMCUfinder program. for each image
this information is  passed to the def Hasher() that will create a MD5 digest
of the data pair. '"
""def Hasher(I) takes in the text string read from the images detail
text file, and returns the MD5 hash digest for that string'"
def Hasher(value):
   m = hashlib.md5()
   m.update(value)
   tshash = m.hexdigest()
   return imhash

##  MAIN  ====== MAIN ===== MAIN ===== MAIN      False Positive Tester

if __name__ == "__main__":

   print '"this program hashes the RGB Coordinate pair of each ZVMCU found in an image.
   The results are stored and compared with other images in the database.
    This program compares known dissimilar
   images attempting to produce false positives from the datasets.  This version 10 May, has
   parameters that adjust the impact the RGB saturation points from the comparison test sets.
   Output is saved in a tab delimited text file "falsePosNoLimits""'

   while 1:
      cat1=int(raw_input("enter first category number (1-8)"))
      cat2=int(raw_input("enter second category number (1-8)"))
      satpoint=int(raw_input("enter saturation cutoff in pixels (500-10000)"))
```

```
        if (cat1 != cat2):
            break
        print "category numbers must be different"
    test = 1
    while test < 8:
        num = 1
        while num < 501:
            satrtn=SatTester(cat1,test,num,satpoint)
            satrtn2=SatTester(cat2,test,num,satpoint)
            if satrtn==1:
                if satrtn2==1:
                    match=[]
                    tshash=[]
                    tsAhash=[]
                    TSHasher (cat1,test,num)
                    TSAHasher (cat2,test,num)
                    match = returnMatches(tshash,tsAhash)
                    tsA=len(tsAhash)
                    ts=len(tshash)
                    ma=len(match)
                    Writer =
open('C:/programResults/hashresults/AllTests/falseposSatTest%i%iS%i.txt'%(cat1,cat2,satpoint),'a')
                    Writer.write('Cat%i Im%i Test%i='%(cat1,num,test)+'\t'+`ts`+'\t'+'Cat%i TS%i=
'%(cat2,test)+'\t' +`tsA` + '\t'+'\t' + 'Matches = '+`ma` +'\n')
                    Writer.close()
        num = num + 1
        test = test +1
        print "done"
```

Appendix P: MD5 Hash Results for Theory III

Image #	Baseline MD5	Image #	IRFANVIEW	PhotoScape	Photoshop
*image1	4ae3136a1cb3a6b	*image1	ba4b174ca85b1fb088	48783c9e8381d754f5473	cc791f7ad34f36d74402e
*image10	2b323b00927affb5	*image10	d057c35c7da832edb	99e39a37dde029f0a4cf0	351144182731735c296f
*image100	06d385a02a12321	*image100	03b1d1e8fa524bae77	4eb300bd9928348832b1	00813c352a42bfadc8315
*image101	8f243e6a3c1fd793	*image101	c2482f25f0f256790c	d4fe84dc22555b0b919e3	205383d1bb551c9520c5
*image102	5cc943896f9eca27	*image102	9cb0e038ba8f51cb9b	4b21dc2531a71e82aed65	b0d204607315a955d81a
*image103	e9a7ed68c247cfb3	*image103	d3b860599fcc5eac81	6ca7d3ec0329e948f2dd2	1ee8ed5e61179f3a1f86a
*image104	732e2dbedc21196	*image104	bd2ec1d249da9c6a86	8d07b3437617298c7032	90b87c89a3047d6b7a82
*image105	5ce90c562278644	*image105	422236be86b2b9e49	5263956411fc3a1be6a6e	cb8485b87da131c42ad6
*image106	9cb0b1405fc3ec27	*image106	8abe408a70502cec20	a2f16e0c5ccddc53cf0cad	704e4d658c0f825694776
*image107	29e2dab97497534	*image107	33b5f08b81496ea19a	abcffb86a50e9538755d5	a8d7a18865701535d2be
*image108	b01d218e0fb44d7	*image108	5fb9078aa6e4654252	ea784a379fe79da67515f	e6be70ba594c567718edd
*image109	f887cbfd5aa60f9ff	*image109	4cb4f3a08b8e918e99	eee0308237dc2166bc211	b47888e1cf27ae3078998
*image11	3a69d4bf045058f5	*image11	329b053b80dedabe3	c6b794ad43e24396ede48	cbd101c4467c4138fc01d
*image110	c865d9247bbc19f	*image110	06cba52d21a400030	000d7d3a0346dbc1ffefe7	ce6a4fc24d04136ca0834
*image111	16979a2d427c94d	*image111	154e0840e93715848	d2ba9db5bbd0ae05e7e29	c5afa2a62407b09ee8f81
*image112	eded70c034b8c0d	*image112	b791cdb206f559fa32	bba0e7fc3a8bc228c07a6	85696b424591940baa90
*image113	fe97dbe7fb46e89d	*image113	ab55c4774b224fe558	48ed6bebd524aecb720b6	466029384aa2fe7180aa0
*image114	0a2c81e76901abef	*image114	60337f9658cb940ca4	d8612357eed90bc849e27	8d2313c013f65e6ece08c
*image115	54db7033ebe7e0c	*image115	4b10be81fbf8df24f0a	865188376f660032cb218	282755c5ddb740905ba7
*image116	2afbe32db8d6d1c	*image116	bdc9212e879ab1baf4	6af08f3a1a355199f2f408	edc359cdf64f920ad2409
*image117	e5a75f873eaa71c0	*image117	0511e81858cc47a27	d0ff6bbbb24770affdc894	2aa787ff9d36ecb646882
*image118	ebc09e60cb7192e	*image118	8fddc25321950b91a8	2458b02330ceb56c8ae55	c672919a0a340017b210
*image119	d0fdac19893d29ac	*image119	d52e59257a11e1986	3a118f4679f53f1b4770a	29b24e7d4821ddd03d71
*image12	cd04d0873a34fa9	*image12	53b0bc2b47aa544c9	930af8ee93b7e28360493	cf3ac821886b64d3baf12
*image120	6091f2bb2cef97cf	*image120	091ef2b9b1cd956f6d	f10e1f8af4197a868210d	5659940df088875793b4f
*image121	4c23293dd3b9e0d	*image121	678bbca7eb92b4d8fd	3f3fb6924ac82be4417cb	12e460363a41c201325b
*image122	75c169edc6c8c3d	*image122	8cb54415f183c2bb40	483b8e802700b7117e7a	56cf5d471dadda3239f1d
*image123	88e908a72ec0de6	*image123	1cfb41de7afdef2b92	d70d430d1fd45187e3773	13d126a15e9df13f07824
*image124	8b15f51ee99296a	*image124	aba48304a0d27c8e3e	8f86cc90c6984eecb8fc89	e97f5ed10de3e603ec2a8
*image125	97f0c1b167a080b	*image125	63097c2324dccd83a	89f973fe07f5f803772df0	f49ebf08d35ed81e28cb9
*image126	58f941ea5aa718a7	*image126	40d3bd8bf7cb82c9fc	0e2b0e4d1430e4fac5503	d8e49c7307f58354fbf07
*image127	32ebcc90942fa8e6	*image127	81fc4d4972aa92f168	989cf3171da8975c75f81	63937f6a452a9e75fff329
*image128	89b22cd373cc4a9	*image128	f6a5e911d573d11198	6a4a0f339338e632d2075	56e44db2f1154046763bc
*image129	c0b402ddacbfad18	*image129	42ab0e4928420df2b7	381054280c83ae2ebb90a	5e9017e7ae6f749879567
*image13	0f9e791b37163b3	*image13	4e52e462715c29aded	73f870ad2fdf251bc703c	6c06535c657ec61f30860
*image130	810cf3e3895de41	*image130	718800bb8858ded97	871d85181960ca02b8eee	1a7329f5fd017fb0140fb
*image131	20485cb4969aab6	*image131	b9c5f696bb0a9ec75e	bd02e36235a13a75dc472	45aecc8cb6d3e6d6766df
*image132	4e491b630649cc7	*image132	2ab7c41997ddee7b3	42a237cb598a8bb0abcfe	7478ec963aa1b2edfa15b
*image133	7f704a157f6d6c58	*image133	05e80ea587c798223	d9f1a858a11fb91788f49	0d40497aad5768dd9ad5
*image134	84e25f43ab94840	*image134	32c5a234e24fdd03d0	9aa9eb03021dcc90794ba	ec0b58655d91c5050c41
*image135	3d70c117cc3bab2	*image135	5b3b686472f2c3480f	6669a2b0fdf7d30009052	9ed96acf4f071be1a2f018
*image136	fb9951142fc51f4e	*image136	f1b3eaaa8454158307	1c7af28585c5c87f3f0cc1	1c0b435acbeb1f7887e1f
*image137	642669aae76313c	*image137	a94b55d39bc0d1bed	4c6fffd28db2b75998a71	443a3f5585a7b141abbd3
*image138	0f205779870dae0	*image138	aa8e731a7a0a6a6725	84c6aee0d4887bf2e4ef9	ea7232abb2cfaf1b97a71
*image139	7ff514d4020955b	*image139	417810c2e66aaf907b	56c1390efcdee31a0ab19	1015f5634b24751c288f7
*image14	3e8897e3898a969	*image14	5a52936f0bf8c1dad4	a7cb22ee0fb5ebfe33d0ef	dd08376464c4703d80dd
*image140	9ab9646177a2f84	*image140	8d50c250373b42761	76f14e70ca4b98b85799e	3ff71c50056c724467b8f
*image141	424b7a4990bdb35	*image141	271a768e6494ed208f	030d657c043f265eee5ea	06a8fbd0db59ca8815893
*image142	0ed6bd4dc8ebe8ef	*image142	62a584e66d8dbf264e	ed8e19b6a8368fc62d316	669f2d5e0b034cf54b674

*image143	338f4ba84467e2d	*image143	86985771da32681a0	a796b4d331dd32a336b7	1af3d6c9ebf8c5cb2f5502
*image144	43027b043c87d23	*image144	b53c4d1b426878d72	ca5e3db067a129cea0e35	69df1a5b833bb94acfb85
*image145	0d4c7d9d206891f	*image145	c0c6f8005021361cb0	4c858c8069e178b8181eb	a00ff140e3b5e2a019a32
*image146	cb1d618cc43992e	*image146	b0427478848793feb	b900e47a58479247a47c1	61483d5c5982cb0e00f94
*image147	74d98f58cf81b78e	*image147	b0f47dd78d0e697e2b	258a50603022c2267967	d0ba5fb9552acb9baac81
*image148	6c74338f8ecc233b	*image148	436635bee221dbb63	4581d4d6c0cab47485ffb	1b904093eccc722780fa4
*image149	b3ecb7661f1124e	*image149	740a0e09b6e3cface6	32557552c17ce9dd7963	c426e9df2cd16dd727c78
Image #	Baseline MD5	Image #	IRFANVIEW	PhotoScape	Photoshop
*image15	d76ad4b4df22e8ec	*image15	6b83417e9b175c558	fe48c85d8666a6fb368aa	754dd0dcafb79afdf7cfc3
*image150	b518797cb458414	*image150	f5d17feed78e134fb1	9f11edc117e5e76e396dd	6e396942750894b62829
*image151	44391dbfffa0f688	*image151	5ae59ff3daf6be3c6cb	e229899c9713bdf5ac762	01caad8918861ceb85e06
*image152	dec93a471988766	*image152	3ae25bc0ae83e75757	e6028072065cb09e47fec	ae7bf9d8915400ee315ce
*image153	1a29d683bd9fc97	*image153	d47bf4bfe6aadf22e2	00b4e42a06653615525b	79a67371bb6a16f33f874
*image154	3f263a1b1870711f	*image154	e7934c2a793057b45	ef88ab265106ac5eaf2d5c	e4f56cadbd69ed70f4137
*image155	142ce5e97b29291	*image155	7eb833428c2c942d3	064c18b9d2e4bd0c2dc21	45179d525cccdd7cb8d5
*image156	cd56e4251e9d0a3	*image156	874b2bdc6e74beab6f	bc7d8b8ad8e74697a02da	a4d4ab4745a6e277c584f
*image157	6be764ae0bf7b23f	*image157	01215ab7b327b7a72f	0ac30a1ac8a82a171268c	1c8aec90a8b001bafe7e8
*image158	c8269d3fd0d022d	*image158	b935d1e853918ca6d	e31fcfef6a03877ea4ed06	3c57ee73af12b17636591
*image159	11414f1e85dd57c	*image159	dbffbe28c4e1905373	48ec3cb589517fd595f84	9bb25ec3589c7e333831
*image16	4ee41678dc22029	*image16	30f9613e1bdb85b9b	7594f599aaae3b4d1a330	c2f6adb6479b21bb5703d
*image160	13c8cc0ab2d0613	*image160	574e5626441f9a88f2	c83e47a7c18d841b1e9ec	bdeefcf484da5ca814649
*image161	f12258d193f8aa68	*image161	676043b0c9a1eba4d	095a09b6d3237bc848c6	f8c9177e32c62192920ba
*image162	535dabf920107efb	*image162	c20938c519fd9ff9fa7	38af709ff316393180950	c079927b5f0265de28fae
*image163	a65840fa2b857dd	*image163	ae0cf27e381a0c4a3e	d272db3dc5443e7f1431b	992d89e74516d60a75f7e
*image164	bfe3f4897250e4e5	*image164	80885a681ce79a7d8a	e9584fa47b8033e506ca3	dd4bdc6234ce6e3d16eb5
*image165	c32083a503a7897	*image165	d640468992b1a610a	671cd75fb042eef00438b	9b20b6fd9641ada52018c
*image166	c73f6e958abd620	*image166	3833478da4d544bb6	9752df79f9eef3dde76e2b	c1700b6092b7a28e1888
*image167	0d028a515158d31	*image167	680aa41e04009dc3cf	dd203d0d109808815cba	6394ea49354e6ef942a6c
*image168	bd9382e87b4c182	*image168	f05e219dd99a6e1542	408f7f72d28b1c3554d5f	9aef66c5145a28f8b4b9e
*image169	4dfa5886f0100de4	*image169	089c578f59e2441a7c	edc2715c6794c44765a34	b8a2d8f881b786230724a
*image17	b30dfcca6b54dfdb	*image17	5f7a52415d746790fa	52a69852e1c60afe230e5	65537d1f9dfb40bb67874
*image170	637402d1bd53a74	*image170	9f6e25e04ecfddcd10	88ba170a933d66901470f	acc7efc1c538ca8046307
*image171	e1b5d83be10639f	*image171	4b68bfa06e8ac7769e	1c85a7f4a67c472869613	03b2a46376e692871558
*image172	996cbf023a9a491	*image172	613fc88b3d81bf68d3	60e57c36c298af7ce9077	d068adb6827b31603236
*image173	d91a88db2283a7b	*image173	a3b92238c9f79eb6f5	70229f60060ac5999d317	8d5c0ce3e3468290be0be
*image174	4587299759d5528	*image174	a6eed35585a10e2566	79f95c094a42f48f551f63	c0277f4cb8a4c48fed771
*image175	0f30d6fa0d454776	*image175	1890971d719d349a7	0b7676f7daf977c672137	deff00b4ce9f85d988cefb
*image176	fa5178f041775eb2	*image176	cba3b89f2396932559	69aef0f860c677741ac87	9591197d6f973a4a08d9f
*image177	6c8476486767613	*image177	4f63f0543e00e65205	ae0ad9afdb5b5655089bb	a2d5f4272a1e44b311030
*image178	be74ab6003fd42e	*image178	15ab75ae0c47d1582	9069fee624cb9c242d0d1	0314a3581de146ebd9c1
*image179	bf0a186232836f30	*image179	0531152e469e51052	e3a73a880da97403f987e	68ea277bb210fd6abcd28
*image18	ef293c76350959e	*image18	6331fcb4e15b03d490	8940bab232c21bd8b819	683547a72ac7075ab383
*image180	75778f7900ab125	*image180	bfa55d72536cbf02b5	d81160b47123315270d3	ee04b3f24fa036c0a56fa1
*image181	ec345cf235e2594c	*image181	401cdb89a3551a4a9f	b53504298f10a4a70759c	9d935bb634d43dfdf42df
*image182	c9fb12dea7ebfd20	*image182	97ade0742660b73fce	d49670ddaaf2f6096255f	d0604e8c3e72ff5c8d706
*image183	0785b9110c0db52	*image183	31142ba70f944f05cb	4956f43d335cade984924	2be0a896107779988d70
*image184	3105961bdc45a0d	*image184	893cbb72c40001831	d34b35aab54185af9f719	d7871c25c53f21e801240
*image185	b660a71ab7f3931	*image185	aa3c90193567ff01de	50f459880642da1776e8a	beccedd560462fb6111a6
*image186	9e7481ce0c14f804	*image186	8e9cc25ec25c41f105	d72509c1534209323308	79024116850f08883734
*image187	841ddd09f113f50	*image187	a87d818eac113d1f9d	1704879c991d0c89db4d	11e05822bcb91a3f7f780
*image188	383bca307b8fc86	*image188	8ec02f99f821e95553	6a63bbc3119eeed48d59d	c3b677ec5f101b2afbaf06
*image189	acfef1bf260b924b	*image189	e7c8361faa50644882	5aa3f84bcf8281115add4	bdce5e054e53882fdd6ae
*image19	d6fed7a356a54dfc	*image19	e7b3c6af04a5ceec6ef	7736253efdeff5b52fcac0	d03c92eb03487bffa46db
*image190	7fff40dd81db7e92	*image190	c1750be291f9b57d78	d269eeac8722b4fa31097	3fe267bba4ef9ae58821b
*image191	9d140e8cf74ee5e6	*image191	35b6d82c21732907a	b07abc18aea738808712c	d2c0719b7413d0cc5de3
*image192	2d5eda51d8f10a4	*image192	9f16347d171574003	feab71e3b81babcd769a1	8af22ddcad27aea3a18ec
*image193	2fc49a055eb5d26	*image193	1114fa447a7c503c7b	afccfea4c3666ed933d550	10e436391e21bbae8dcfc
*image194	dff85f993f1bdaf1a	*image194	9b1b9bc4d8879037b	5c9e905685106b5bb28e	c1beb0565d0bb0d17d2b
*image195	e5b7d53a4151838	*image195	e8642981e16b4299f9	079cdc6c22e8201afdc3d	9eddc2b95306f185cafc7

*image196	42c368d0b6eddfd	*image196	fcd04d56f6a69a50eaf	1c98b109c0d3fbfcf12a56	16359d0946cc1cf0024a2
*image197	e82ae18f72bced92	*image197	49ff71a86463c32b68	ae59c50e73519b6c6b5f7	b1ae1d28e15d3cb833f1f
*image198	ce25e43089a1e37	*image198	6279548cad160b871	0cc75563f47b1a172a10d	34fedde6ae6c8b98dc805
*image199	470e2eb01b63518	*image199	b5d44385ff92f73f25	0bce2d237a40c3dfc688c	b0ebd7efd4e1c1e9ba4f2
*image2	7df9c574e1f0578b	*image2	1b601e2f8d7127570	248c1971d198bd986801	1e2e6e383064e667256df
*image20	9cbaff18529cbe5e	*image20	f55db0d0990fa1701c	41e8ddff1f4434693d099	655b737d9b33225c6406
*image200	4223a63c0121ed1	*image200	704fa6f39884da07c1	b80575e0add7d70bc683	a790424505987734e2b9
Image #	Baseline MD5	Image #	IRFANVIEW	PhotoScape	Photoshop
*image201	2d6c93acb084e44	*image201	8e62e0a0c088e79faf	5c0e5465def7478515942	996cb96c2a7e72afa9db7
*image202	3acafd5581491dd	*image202	3411f79c002eea30d0	77a3501c85ea519f6a892	839c507488f1536526fc9
*image203	ae87ea57ce066b2	*image203	6d3b7a7dc5aca17c2c	8e79b63862cce5b4f0df5	f178ef420284ceb5bd1e3
*image204	19e14d16c783266	*image204	b2a08fcec68f8e88b5	a92cd823fe596f5aaebdb	62d7a374bab7232812d5
*image205	4912195c3e95dc0	*image205	5ed6a9ce6317ef82f2f	4a9337d1ba2425c979eac	98f2e50772e38024018df
*image206	49d43f4cea23863	*image206	f6a106bd88ad1e2f76	9e2e5bc2fee10f1353f2a4	a2413c01fcccc40763de7
*image207	f0418acbe8b50fca	*image207	4f2aba09119aef33b3	efc043fa08744838cabae4	49645a90ab86eef6acae1
*image208	9fa6e4770f3f92a6	*image208	249fe31650918c0889	cbd5cf66bceba6c4fd1c53	44ae429da69279ecf83f1
*image209	7f3a7df1f936ef45	*image209	52f18c43105f9a0c07	6841069f55afe90851417	3cae26e5e7f99a7a98c0a
*image21	1a4948636406e76	*image21	cefb882f181d7069c0	80d56c9b7ead77cfece31	b58aab26d2e7b4f9edefc
*image210	18cbd9a42fa0953	*image210	1522d8dfb371e9112	25864e70832c4388141e	03ec014f0f2d56aadc778
*image211	adbb135fd439915	*image211	cf4c575efdd54d7767	e12db54b13a2aad77d90d	ee3fd1d443a6e2ea23e4fa
*image212	0cd7d7fb5681e99f	*image212	7b2e7be9ecec967f7d	0cfb553113b51d90e1723	e18eb00fa58212336350d
*image213	6758856a447cbb8	*image213	cfa8cdaa86fa795700	616e7d6fb27155e8dc87e	9b49615bcfabdf57e124c
*image214	0e6013137c3f5bb	*image214	e535033e9e0cfbfe7e	bdf151c0e5e70938bf248	3bfdc0a1ccdc07efd1fb4f
*image215	c5284763ca1096a	*image215	a3294c3231fe6bd1af	fdea86cc2ec0da7a08e14a	aeb60715896160973fe45
*image216	d01534809fd4f12	*image216	a214dc4953f59cb6e2	4b6d8cacfab31919a8901	ae52e0063d4d234b89a4
*image217	a826d6cda4c6e7a	*image217	c36b29ccf0c4efad0d	373e1a55899f1dd20e254	e5440cc675439f776341e
*image218	c5bfae54dd203feb	*image218	f61a766c6d92a360b0	afd4229dd58faca5d9097	1763d7431f2f16bdb169a
*image219	b2fa353110b8788	*image219	a4b4d01fed3a4ca1f3	ad00bd47e847d017658c	8a256e1de65f9dc07a072
*image22	393b492036cdcba	*image22	a5fc79232e6d05f5b1	7d087aee51fbac9ee0f5fc	0d6bb93cba551e716359
*image220	29d0d14ff558cd55	*image220	ccf29f1ca53a0f8ed1d	9c12041ad78627fca3ca3	84676450298d1fbe83d1
*image221	383d710286c4a97	*image221	6f4478add7e58341cc	6826ac596087cf8087ab9	41f6bb82ba6e186f389c6
*image222	a95237a1670e82b	*image222	c29d1ebd9e54536b0	477f95ae7ce20a76eadae	d990d385b043bd207d69
*image223	8aab56136d4794b	*image223	bed5028d9389ad5f00	6bd0b856d53c9cb2f8588	388d328b4419ae7de6d4
*image224	36298662e7a7554	*image224	79e17858c3df0c13d7	f372d10cb7f06860c5a5c	063ce1467c6d595c0d5d
*image225	3ca3c68543070ed	*image225	6ac2267b43e722e9a5	6c3e8101dc1e8f9504287	44b817c298f9a7f676317
*image226	87e0038e08edd86	*image226	e7052c80f6ccd872f1	08c00fbad7ddbdfc00f2ae	992e26d0f1edbc51c8f37
*image227	b7463882ed2faffd	*image227	6dfdc791040541323e	fd902b34ec751d0b8f258	188cfc3f261074c58ab37
*image228	84002731637f476	*image228	14d751bc3c27cf9e6e	9ec4a78ef3ac696985f7f8	b835abf658c8860e17f5e
*image229	cfb8f97718ab3bc9	*image229	0f025a4b675428e1f1	d8ca081407b2115a7a68c	d2597e070bf25515fdbd5
*image23	6cccd66368275f1	*image23	c5f887bb95e5c6888b	cb7fb0f2b8d63fcc5acff7	7dcb3052a9c2f68538e6d
*image230	78391b610a7d298	*image230	976140fb122f219d3a	a5381c24486c8cacd2d6e	44b3ad7f4722b5606bf04
*image231	134c907c6934914	*image231	1e45bfc4df8c806914	b75b0cfc20022dfe33039	f44c419f5387f5a3133ec
*image232	da718efe11b07eec	*image232	ab0a8e940deaf25316	d957c661462766c00149	02573b8d736aad45148e
*image233	067d61e7642e4eb	*image233	c1fed87d5ce2acd431	7f5e4d9de12e7bacedd50	e18c3dcc85553f711d5ca
*image234	53fdf895b1a3e773	*image234	3f612fa54b866d1012	41f6edbb3fc1bd845f9c5	921e079c21dc1f08151f4
*image235	de0c51a34afe699d	*image235	46e58a870cef552fe9	a69809f8ce7413304e619	662e4759b7276b2c509e
*image236	c99a58168aebfbc3	*image236	21d60e1315dfd688f7	0f8712ef8e4ade3e35277	17be7269d4e74ad48091
*image237	76163ac8635571e	*image237	e97b7458872d0dbc8	f24f62569773193a95c35	9230b4d0665cba3e323a
*image238	62d71043203a6c1	*image238	9ebc10a4b380b8286	8a56ec98e869239afb363	49e9c2895df2cf189d330
*image239	4527e047fe568c8c	*image239	c454ac528f53a84488	fe86b3feb0fa2009a750fb	5f6183968d4d8da61760
*image24	80d0d75fe5d3de8	*image24	eae040aeab1209be9d	bfd4d7eb48cc2c7db5b42	c59f557233cf614b5dedb
*image240	ddd8336e5685483	*image240	57332016686871275	462f534d75332a12c5cba	4c2f1c240e316b5ebebfc
*image241	d69167a2f8072b1	*image241	bf3bd7fb4bba57d6a7	51492af13079962fb171b	9327d44142be87a49abb
*image242	5322a8868b70113	*image242	0278b922cf9230a359	dde5880b5cb4e7f784d5b	7a8d8cd5cbf2d051c8475
*image243	4b8e76d2b318545	*image243	f52ce7bf4196855723	2fd2a0a053f75983c2fdfd	aa91eb5254bf8dde5dfd3
*image244	20d80f3571a7393	*image244	4709b0190996a4227	d78532bfce39ed6cf3eabe	b0b0ee0b8532881d058e
*image245	e91ef2b9f34350fa	*image245	f0c5940ef7a3932175	72c0ab5fa7e83af573acc3	1dcc16a839b557980ffbb
*image246	57c3623a1c580da	*image246	ea2a8e30a821455073	6f8db7ef1adffd6be91708	08e299a5c77d0cbb6121
*image247	0dee9357ce4f2a8b	*image247	7de24f20ce3a04081d	98521a2f151068fe3ee82f	ace1278ce287721aef0e8

*image248	e4c29dcdf10915f5	*image248	fdd6f70b2090ca9ced	8762cfca2cd3dccc4826fd	8991d34b67544e5525c9
*image249	3acc454f9a4d8c89	*image249	0b81102d6f28e2f305	39f97f0a2403c59613827	3e3debe549ecd98d392b9
*image25	385cb80cf413448	*image25	2efcf5db01a7f407cd	bfe75cba86815f8fcd7754	f93fa079eba1de07cd93c
*image250	d2938cc74634db5	*image250	5fe51e88a96535991b	dc947fdbe7eda5954e885	b2aefc9628cd5e3d38edb
*image251	0489e4acf612794c	*image251	9ed717ef7ec697830b	010426b677d3a1bedaf22	0a255239b154b02dbecb
*image252	73166c7fb3a88a0e	*image252	bbf5e50e5955dc49cd	4a4a62fdcc7df538eed83	78d83e77a1d44d11f92c3
*image253	269833d49d81dde	*image253	126c27ea2480049a6	e6631918d07206144b9d	0bcd32d065e838a52a0cd
Image #	Baseline MD5	Image #	IRFANVIEW	PhotoScape	Photoshop
*image254	5d3c87b995e97f6	*image254	b7d162a5e109d6570	ce5c9d625d31703a0cd12	a2b77860580038bdc285
*image255	087607b0e589c3b	*image255	d5c3b7626b0806f21	39dc3b18a349f1188314a	530240b606ebb31e744f9
*image256	c174fc0ac86c0123	*image256	4c8d31f504964530f2	a4d7f818929f6223ec4d2	b34c26313a97a3a0aa7d1
*image257	7d208bba98e2a14	*image257	a6a940d704a5f10336	bd3bdf4590043b398606	913f2bf5a6d60f61ac3fcf
*image258	386915949564076	*image258	1286f0151a13afa058	820f9d1d9f16293b04f72	be143e59059a341b7300
*image259	c6976228b0c53c5	*image259	8830be22e8b2937f4a	f18e0113341810c4644d8	fab5e38ffc33f17f649e7b
*image26	2cfdea4f198850f5	*image26	1cbac76766e5df33ed	2a2f48d09fece7e2b64dca	be2597560987a5fb3a969
*image260	6abb65d47af1eb9e	*image260	80fdcf65a16197c47c	0f5a68ecf444b3400553d	e23fe2cd1404c76902585
*image261	97010d9887b1f60	*image261	1e4bb75c8d6722d8e	e4679b3e376757ba5e5fe	1d6351842feeb71a46de7
*image262	f37db8d6d29f345	*image262	dc7d4ec285eb292a20	5eb1c2280b99cbbb47aad	4c356e4c60bc12031f5fd
*image263	1928ec1a75b3cdb	*image263	a7e3719fd63e8233ab	6ec26fb36f673f96439c4	dc95c24baa8fa3303db6a
*image264	e6df215ec9a78714	*image264	e5b4bdbdeb9579d8fe	9c4fb19f07c7429ff94385	c22b1f75ed752bb74e020
*image265	aebb1ff62cb4f804	*image265	9f1003410e896c2db5	d921a6cd075ec4692cf91	db040644d0ef129eb93cd
*image266	17ebc2652f3007f9	*image266	5bb977caa6fe1d95b9	31ad1efc4b4c55ec0cb3d	126d4815a325f482e9d84
*image267	a6139db52c6f22b	*image267	d03c7a24ae8013f617	03f58150d21aa2e32fe86	87af73fcba07be6c95176
*image268	3b1d631ec47f5f9d	*image268	5e2f38be33d9780b0f	427d5712fb509ed6befc8	6b87dca52993894514a7
*image269	047a61789126112	*image269	ab88c6c580791c897a	483ace0be5c1d4f91f7c7a	97c73b94957d4789b5a1
*image27	38b415292da1baa	*image27	13aeab268688a8f895	f371305545766405abe9a	0ec7892ab148606e2e034
*image270	0f6efc904105b4b6	*image270	fe30ab641a9a0fedc2	1b9291914385 60b601c0	bc18a6896fda8d18fefb6f
*image271	e94435748e29ee8	*image271	9cfb538b2ff7174e4d	78881e62b0e07a7ebaf38	e4de5106db11fc547faaf2
*image272	cb94194bc571577	*image272	eae61b86460f8398ad	6835a573217e617dd613	531f63b3b32fbbbce995b
*image273	3f417cc08e30e799	*image273	f874dc954e3581009c	e97d61874ca20e8f8e0a0	8177e5bce64773c6a6600
*image274	99c6a63b5106807	*image274	a511b390d50155baa	fc457f3a30fa57c71b28e9	17038bf8ea8cdf09a7a96
*image275	47eab3fea9ef2778	*image275	de5bfbc21ed2c62323	658d27dbaeb62f052b013	a926ca3f52bb90dcc8f7a
*image276	d225147c9d1b1bf	*image276	e27b0ff56ac1bee1fe7	3df02373152a6a502edde	004b69230c3e3f090ccdf
*image277	08e6def89e7989c0	*image277	1eb3ab418b76e23c1	e10551ac6563cdda659fc	af9a326e09c113ae0b46d
*image278	32a68aad0087312	*image278	7a610096d917a0f82f	744b4d2067d45407bdf1a	3ad2ac994c14736d123cf
*image279	e715b35c1caa1b5	*image279	1d8b8e3fdfcdf51f822	0c3124fdf20c8611d6800	4c179ffe42d3cef93416b
*image28	e9fda437b8a5cba0	*image28	5654afe11768190dcc	97ce06a31502591220df8	352fd6632794c0a4abf65
*image280	d5ca85fcaed95a79	*image280	5eafed06d559159839	aee1b586f969ce8ba6e7ff	bea020277e49a0cc7894d
*image281	3d070808a390200	*image281	816ea4a241f15dfa88	b2987152d8b2b325080e	a540ece665332020cdf8e
*image282	d674364171aae4d	*image282	ae5da69836d8ebbd6	8442ed6c2c4b55c31aa81	56ebb1c043f2666285586
*image283	d63287d24e7a915	*image283	14f376cb2fb20f7fdbe	8359568115350d9bbf4a9	592808d5cec7fa5e18e02
*image284	87cfcd51fe11355c	*image284	cee700ddfa6498fcd2	f05de0edaa9cb41abb9d8	dcc147f7d32aa68aa46da
*image285	5713f82c23cfa216	*image285	7ac4f86cf640d8ecc7	a1cc09078db9fd3aaba90	2ea3b09496c0317b876c
*image286	a8559ff1db03b716	*image286	5e57528c9c7e6040b	a42535bee5d6d48a6af62	aeff7ad7e912e322fa1415
*image287	2b5b20e87e03799	*image287	f26a09091c9f7c5992	99b56fdb92851d3e360be	1102143ea3cdfa6e320f1
*image288	ec5065b786f2ae06	*image288	f1ec2b40296dc3b9a4	03b80beeb5e47dbf00556	a2f05707cde8b238aa565
*image289	8d33c9becf2a4bc0	*image289	c320fe6bffdbdbad83	3744e4f04d81f1786ae9e	7b1a2dd4a101d7fe3628b
*image29	615c46989debcc6f	*image29	4cb37c2bedaa1aa623	47797309981631b8154c	a4cabdecd14bf9385dac5
*image290	1814d61054c1fe6	*image290	ddf9221d70b30a2d3e	b66c5433cbb0ba71fb356	52ff0ba5fb3b058b5c78fa
*image291	d25516608bfd4dc	*image291	e3a1330bebf003d3eb	a57f7f5297812296c6c2d	70e4a7054de435dce2700
*image292	7b7475b82a8bd83	*image292	64327b1d382eb76f6e	112a15de85e253efcee2f3	111478619d7131d73cea
*image293	42ed1a4fac949346	*image293	185859cb4bdebf7ab3	97e401ad6afadbf634fa54	680b89aaf0fe977649c47
*image294	780bab1f173fc83a	*image294	c7e7ddfe29b99506c5	920f1695da3edbcbcc45b	40a5738798eaf70a59ff5
*image295	13df3448e4f18f24	*image295	501de8c5455c1e1e4d	be0614dad4eceaeab4c49	9e649d15e65ec96307dcb
*image296	332e8144c57d327	*image296	4243fcaeb2e6148e68	98369e8ff3d794d579b22	d8aeac48f05316c170a24
*image297	d214cbb4a1ccf4a5	*image297	acecb1a4298491fb40	decfc85571391b7c2cac8	c12faa3cfa0359ad23da8
*image298	2d95a3ae35de972	*image298	0f485bead32a83c2b0	2a59b4f92172a312dd1c4	ef773c9feeca3240097a4a
*image299	9190a4735046553	*image299	b5850c491b00046ee	9ccc0428dedb3baa0c0a4	e03ef55f6b1d9acd11502
*image3	98c78623c2e335a	*image3	9b5a29ff9bafdb95b8	677246b55ab461dcb15b	b74878e6fde5895d698f2

*image30	23c75548aa5148b	*image30	b28c0a5d115d52a19	03720dcfeb3af73a33a68	2b9273eacdab576ce4305
*image300	13f30871f5520f2e	*image300	1deba1ec4eb115be32	260b5c347b4c367a1ffe5	924b8b29437ce90fcf3a0
*image301	8f9ba7b492ee5e30	*image301	d9aed589b48a2756d	2c90d87a4335133289d8	b0775d019ccf1ef9feb70
*image302	021d564d5fbf44ee	*image302	f30b6a2cf8c3d915cfe	f1579aeebc6037e52fda2c	89a8352e56d0668aaff12
*image303	24f420e72c6f5916	*image303	c5cd78bfc2b45b4b35	9cbbef9f28dd62dab56d6	b05bb1b6f5654ec006af3
*image304	b93df9f3072ace4f	*image304	6f205d0f6cca3ff883d	d2d06e610c15f5e7dff24	daff94f08ec638293cb0e5
*image305	446f7b3dc96cfe0b	*image305	5472dd7a915d85621	72c44cd842e540a6867c7	5ed3cfcd4c223889666d1
Image #	Baseline MD5	Image #	IRFANVIEW	PhotoScape	Photoshop
*image306	3e81310686aab69	*image306	4d5a6333fb6111b2de	d2ce5460378c57d533e95	1ea4aa863a4e1eead97e0
*image307	8ed1ae14925b4f5	*image307	5c85b22ff71dafc0b5	fbee50e7519200b4b7fe9	79ca08d3d7e55fa0e7de3
*image308	41b043399c82d4a	*image308	8f27ecb8a85d9b2532	70e07cc92ef60eb527738	c19e7d094b7e0ddd860ce
*image309	556cf2c2e14691f3	*image309	d46d9acc3f1ebbcb72	78a026fc9198c8b0ffd46f	527be517d68b41202bac
*image31	b12f4c03922ac8c2	*image31	081e53efa2cd902647	97afa0f16e058c72cf29f8	a859e190b34f5b3c43d31
*image310	34d7e9c02885591	*image310	c79140ab46c6e7f0fc	9f6f91303e1dd21a5e137	6444ef053206653087f43
*image311	0cdc1f17cc024bdd	*image311	748eb9c306ccef8dfb	f7e6efdd9f5bdda6fd8239	e27ee498bbe9fcb24cee4
*image312	c33c517bdfc0c0fe	*image312	023095522a3e03cf2d	e40e9039b443ccd7f0387	80187b3e29f36e8c7736c
*image313	e27d1e9c10c781d	*image313	eb0838ca0fc1a9a134	f976fe09a32f7398af0e92	79ac2d8fbe30ed8c58a24
*image314	18d5c4df76ece5c3	*image314	01183ac1e1de45c76b	7795699136548cd9b1fb6	dfaa831f70ba83564301c
*image315	facc8fe742e79975	*image315	b70d4215e3cc33588	555c389b946b7c555a38	205b68e9fcce82c828f7d
*image316	06f5f24d847a82f4	*image316	31c838fec2710360cd	391d45268d62529ada48	bce2d7bc50ed897e25b42
*image317	fe1053954c8eb23	*image317	c7154003a6f91d8f95	dec5ee42699d67373239e	8580d7eee05e179e34a7d
*image318	136629c368c4006	*image318	870b8faabe99828421	bbc0acc13ac03d81c5ef2	47d223527931add5798d
*image319	9ecccfc4a7fbf9c43	*image319	a1bcfc732797863fce	52558c24769a06e7f6db7	57b592ad090a0e31cb25f
*image32	a5d654cb557ef29c	*image32	b0a60b7b3b0b5b256	3ec96a01b975d0be081aa	3fe819e5205db64d707a9
*image320	ae20687f0c4d070a	*image320	40070214cfe17a5318	d650b208366f005e8a5e4	b699b89eca1ee55b99ced
*image321	8991653df1c6380	*image321	00737f3e44109fb73e	c52cd5cbbcaa103468111	c7ec80115f5dc3f38c2e1
*image322	94efbd221ca9d85	*image322	0fcf54f77240a05811	086549e31c2ce4c463567	7150494184e49dc18f401
*image323	8ff32f44d2d6c292	*image323	6d16852debb81ad84	307471890d692a3d0f6fb	13727ca37fe5b26fe9602
*image324	21fa927c75fe6db5	*image324	ab61ca312ca61ca9da	70711ece97b4a9a5be2d6	469260a1ebb5f87583457
*image325	727305361c3dcb9	*image325	5ba2c2eaa61bbfed5c	e9f4a49f979c9ad9d8751	54e190b1d77409cc819c
*image326	9a28da94941e13c	*image326	98ea9bd8094b4cfe35	88b56fe6ce716eceaad26	eb5822cce126f86331346
*image327	5381e5c3b1d2e9f	*image327	a0ffc6fcaeb6f5b4054	eadf51c4c6ab3b4feca906	b577877c4d39ae3b1aac4
*image328	3af2ef2f64e17043	*image328	74f6595929167fb8ad	8c1691051405ece5398db	137d30ff98872514b0cfc
*image329	5fb0b65dca40823	*image329	d5cb9ecfc8528a8d21	cf9dc73e8a7c4c3fb645f2	a3063634d91f5c0de529c
*image33	2fb3739a0e13408	*image33	9dec369e41d8c1edf1	d9afb322f9057fc07eae43	40c4ab91a2b9839a6d62
*image330	883561bdfa7b7aec	*image330	3bbb171520f2d437d	f658080155032f9f7a1ca	49b6e4c67a5ed82813c50
*image331	47e8e4459f54aff0	*image331	67e0f34dcfa2c7a177	37409298220390da7541	298df7e2ac5d8e72c12d9
*image332	32efe890b9e0c975	*image332	6c0158c00cd0e6a490	bc43d373c5d9224383f3b	ef0fdceccff69234608920
*image333	ded9bca1696c3bb	*image333	9a0fcd437e99b08493	2521630c0663f1de710e9	e451eec07efbcf254b2f3a
*image334	1ec91dc27377936	*image334	17276af532cafa1538	d039922e21509ff9ec823	ab52d8dcff89dfa4598de
*image335	767b4e9f2177114	*image335	2b02798513d04c23b	ea8f8009b2bfe916b01ae	f7582dacd1f1dfe871da6
*image336	2aa5d875816731f	*image336	dc0c04cf07a99fcf1d7	72809f4d8ebd5dc560c6f	dca322e6839bdb9be037c
*image337	60bab6efc42cf2d0	*image337	6e22ddaf34b59049d3	326ba302139665a1b0d6	cfa503c8bc9423043f39d
*image338	af376d6068be0ee5	*image338	1353f878ee574f2fe2	0e84968f8316066f30d2e	2b211b32a360c87ce632
*image339	b35de4a2b49cb9f	*image339	816b9adb5469ecfa08	a8e558e75ba2070ee74e1	bd6e3208fe509f8cb8714
*image34	8e0b0bac341f130	*image34	da4f3d466e003ab571	652b1e412d38045254e6	014e65bba4bd7e88907d
*image340	b6e3f1b9dc25238	*image340	dac8456ceaab3300b2	0ef105c22fba665fda46c0	13f774f76290d1e9b5064
*image341	417e5c1cb949020	*image341	049c5ddfefb5e97dc3	8afb6bb2a71265bce2e73	db39b1dbe73aa835ecd7e
*image342	553092e1a1f606ca	*image342	1432832a038c500d7	94ee9e0ba801d1ce5521b	ac58c113be0daaec60dd6
*image343	01e1fab87a846d8	*image343	62757077ce323c16a	92c3ddcfea58d8729688e	b7b50cbfeab983252c271
*image344	43db0c76a6f4fbe0	*image344	f151ff560dd7bd9a68	f710d6279488ee27ba088	25f5452ce0e46c7f0dfa1d
*image345	881873b6ec7a985	*image345	7257cd07de81b3842f	b2e00ee48ff0a67cd259e	ffcce629a1e32d9811a25
*image346	1825ec2eb20012c	*image346	ad86740bd33cd2d26	47d6331908a34e978961	a8fd070506ff8b0b886a2
*image347	991f677854d937c	*image347	a7fa39d248c15a6a1b	cb937accc93ad25cdb2c7	fd77d8019066dd8ed746
*image348	9b5fc8535f1ecba7	*image348	0a0c195d858436c8c	8d71ce3d64b24b705090	cc81063980a34b54969a
*image349	1afd594e9646cebb	*image349	2b6a882bcb655c6d0	af3659ba312eb28e79a9f	9ab0e8ee427118ff0748f
*image35	27a0cd60c2fd728	*image35	d35222396a5d711e6	abb342dd059e99060766	a8e1ccbb6642f00a9d183
*image350	96c54a6ee370688	*image350	b6ab938a13fb2fc704	56c42763069bc904d2e8	df8818733b00315483dd
*image351	83fdfa01f81eb206	*image351	c0b701a703c653ceb1	6b0db7eb05b353f42256c	adc0755f005e213671c88

*image352	25dd005c8b635c1	*image352	2f40e0f8ba7b9bf02f7	cdef84f27f4a9121dcd523	4aa23688865a0fe79086a
*image353	ae66cf4f2e004e2d	*image353	ae33988b9f826710d0	660d657d2ef98645f5fed	d19e7fa0b54ad0bf7b47c
*image354	af61f1f82b117415	*image354	5b67b754bd71778ab	c491bee36b55645e3d263	e162c290966e03e3c8dc9
*image355	9298603a10661ed	*image355	56b95204df5e4f5a20	ef67f6c1660fbebb54352f	0a5e72f4a338ef7ece974
*image356	89717540984c68a	*image356	7d4204109096b2c32	4d2ff06f9ed933c8a6ee09	b1c1325b5eb9d4dd29b0
*image357	22e04e6566e0dac	*image357	72a0f3324780c9dd74	537db16a0de83c2249e9f	9ed8e2e0516ccd77055b6
*image358	a5cb65b8b0a102a	*image358	f0fc94c02e39172675	3aa19851f7a79ccd5d478	64686b5bcf271f40d5a0d
Image #	Baseline MD5	Image #	IRFANVIEW	PhotoScape	Photoshop
*image359	e6f56d70640eb0d	*image359	13a2870ab9b8c573f9	42fe0ba372261947e1a00	31ae6ca706200aa8f003d
*image36	ac45123b8cd7887	*image36	421d30ea43114fb239	c8244536f11fad91b6ca0	3a5731c1780e5119580c
*image360	213db5d688d8c51	*image360	79846146c92146834	0ba78377d7dfb6106760f	28e4e94e0b3ff310e1f89e
*image361	8961076a86fd22c	*image361	f0d6bb30ff0eda7c2d	c00330f317a62a408afff8	2105bb36ecce368ecb47d
*image362	7cda57418398b1b	*image362	0f34ce870464db496e	3cb116e8697bd5b61d16	4ab0d22196bca7a439d3f
*image363	4438e9bc4b12266	*image363	988466cfb38551d2b	04395d4a3ef5dc1ac14cf	657f67bbf86482a51d002
*image364	07d611ba6adaadf2	*image364	bf36a81e262ec4b181	1e772b08f5798bf26f1b2	69e73686a6e92c7b88eae
*image365	028f6060fa7a51e5	*image365	ed4f2d10bc4eec3de3	1dd365503c0d0f6c2a9aa	2d7dd8067ea1092972c1
*image366	7b699acafa7ad042	*image366	4677cf566d0f052e04	2aa7cd5fec6eff998baaae	2034f64e473a4082cfda5
*image367	63be8bd4e2c6555	*image367	29bfca6585153e4e87	f6903c007bec03b1f1ace	bd0094c34e5152f178635
*image368	8eb8adfbeca12821	*image368	ac0483e680b45776c	3a3aa7368d00d365a34a8	a74d75435f74cb5fa318f
*image369	c5e823376e214ba	*image369	d4695269ba871876b	1f9e5aa589ed621a01b22	04ff630e1f67bc49d0479
*image37	19122a16fb1748b	*image37	5362524f4caae6ca96	73448a879be479db5e0df	41d316b3a48e96573835
*image370	d4cb3204c14bea3	*image370	c5b1ce3a86c170f9b8	edec4092238b2c4252545	f7a003a877fbdf7072b11
*image371	e60c8991e90326b	*image371	02d577494b907a0acf	87886ae77218a726fabbd	c8de0d74675cee93f3b4b
*image372	68057d713145534	*image372	43941150ede89561e	fffc17c5c2d23c050b7c47	84b6d4c4f217273da2230
*image373	3374922755495a6d	*image373	5c5fb0dd57eff19048	9b8ec103839645f4dfbf4	f946531538f827523c3ea
*image374	9baa6b2c69b1a13	*image374	5d9c5b677fbf5ef1dd	651d7f1f59e883a0272e8	0af2cf99572a0e656b0bf
*image375	dcd57e69bfd92b8	*image375	e99a4a691e2864540	7fd2bd9ea9505eaa2ca67	f9ad9ccb55d934b66c5ec
*image376	092353b25380ac4	*image376	28ba3ada1a700e469a	47d1d17e70d9f4168ad30	529659227fc407f767ac9
*image377	3c288992f967109	*image377	91455d874cf3ea8494	07b72ddcca72835b0468	2b88a39bc0ef2371d66b8
*image378	ab3435fbb7adcf5a	*image378	085e230905f271e161	501b81f68520065deadf2	7ed8bd4c5b2952237226
*image379	3050ad8b3f284eb	*image379	72892f950c9cbded64	ea70f426708354fd2e29d	eac86789e06fe1b984c25
*image38	9cf0c8ae97233758	*image38	19c9d0a8d60b1ce34	be310e09cfdd749fd1ff63	08d21cb272589c6a9c4f9
*image380	39c4a6745f0e173	*image380	ce4a1d01f343b5a64c	0aa861ed96f3f3e53dab8	8713f1794b101c5bfe088
*image381	fd28b087747d962	*image381	992e5d45df954b2fb2	9ccc2434ba4f488420955	8eecc8f3b43544144eb9d
*image382	df59d9143e254aa	*image382	05e9b76ca1e396cf2a	022caafaf2129651babf26	0657c00fc8a9beb973676
*image383	4b02fb4eddc4ab3f	*image383	3845fb486ffb88a6d0	e80f4926175082d3bd52d	f0d4add8fe552dd0fbf045
*image384	7571bd123a3e883	*image384	0469f9b82a59d1f275	b6cb5b1fcad67704ed50e	a330e350f92bf0066c96e
*image385	24f28383d913cbe	*image385	f80173eb81204ae021	7155b4168d1d41c50fb16	dd645e4477f8ed1d8fac2
*image386	44cb078999c11be	*image386	6e35b7e80fadf57f22	59b12a9209bd49424644	e141dc8fedeed0c6c54c8
*image387	d1438afd5d92864	*image387	46fa0e0d850841c75e	a2d5f794b75e03e597ce4	8e4ed36ad4fdd682b20ce
*image388	4655bfb489031eb	*image388	ff98dab7a5acf536ee1	7186751d316ab9d7cd4ea	2fc1c501b0e93154de695
*image389	e714dde374874cb	*image389	38f3b8111cb8173c10	a437e23262d582293239	1687afd3d7650b7ad7b90
*image39	a4145d1e66aea8b	*image39	30eb430e008b36347	b31b8a1b045f28b1d518a	347c54462745a2909400
*image390	19b9a619c22681e	*image390	a64db2d2f19b1d62de	5cd53820c094fea080174	785025a2b13b3219b18c
*image391	a4145d0c1f0a19fc	*image391	357e885398e5bbb0c	6db948581c1de5bff6629	6bbe2781f3cbc9849e055
*image392	c9295ab7b6c034f	*image392	2fe0ed3add127e207e	ef5926790f559434543cb	4b6c6564234428a53348
*image393	2957232991ede25	*image393	7c8468a382b6dff053	ca0619f5cb14f7da6e689	08b47f117f8f338bb9a87
*image394	cfd6bb884516688	*image394	c1fcd8b4db12a25bf5	17dd318d5018d73e2669f	cf5b78e3fd8aa704f341a3
*image395	f64a73a77d1863d	*image395	79520efe9c6c8d4b87	0d74764eebaef6ecd8c8c	952f95f746e8c590b51e1
*image396	ec74d3d03989672	*image396	5d2f7f207190617785	a44f0e8d7db2808c361ee	090711b104a5f8e393eab
*image397	7952241f87917fb	*image397	2851131565b66738a	f0a8830f0b11f1c8c61b4	4d7a400d42f506f0f8ca0
*image398	ce55becb04e1652	*image398	7317d1f81fb7e25261	c41621b69ac16407ec034	5fb2e9b4aaec9de216caa
*image399	b169d750bc15274	*image399	681473a6b1b1bc17d	428d8fa13e8c15c121f2b	15a3152df937ffaf3fa3b5
*image4	f8c0d4bc7208a97	*image4	25506f35a4ffbdbc07	f3bf175fc3e4f33989a734	c6a1f9fff25ede427d2251
*image40	926e1f0f451bb562	*image40	640c910e8d280556b	41dffe3c65f470e5bc1101	744dba6f1bad80363c5f5
*image400	6dc48f2c35f92266	*image400	d646c9b573bb368edf	91fa79699739d05874d60	0e3f30e789c18683f1947
*image401	e2b11c329f2e942	*image401	97951083906d1d53e	7cc3e2370241b3da68c91	fade5129947b0fcfd8a1fc
*image402	7c35556bab1a376	*image402	2a01e4152f84eed8e8	a2a20e6f7e1e1e5c59f147	04d4725781a16896d460
*image403	26097651ef9fbdf1	*image403	cbb05d9e8148db07ef	6269bb12ad755c5630b6f	cf6ff68f18d5e387008b0c

*image404	9b4e135a8c68a0f6	*image404	f96253cc773e7531f2	f989b2bc5db183f3bd488	9de032487651964fd2a64
*image405	c6d0b136a5b12ff9	*image405	5c87993d17e1c455e	f54b34b4f52fec0ab08d2f	886279c62e02baaf267af
*image406	b20b7d81b5e8734	*image406	bacbdb1a34ce1e89a7	989a216e416832d15091	b0c984e7d1bc8e0af9238
*image407	9f54c65ca25fd379	*image407	4952d303ae05b9ff0a	8adb107a842c5ed001696	dfd42b84568d30c8b293
*image408	8b3063a572a08ea	*image408	115f65836c8931a802	e5641ec4bcbf2277f4a31	0c8107252bcaf47180477
*image409	08bebbe7edb5b25	*image409	0c07e169b903101f67	36ab94608e0ea7ecbc669	e84d483749a0dfe387b06
*image41	dcf3929521ed0bfa	*image41	324366abad91badfa1	6ecb089618abe1d0bedc7	e17519a80fe573685daa9
Image #	Baseline MD5	Image #	IRFANVIEW	PhotoScape	Photoshop
*image410	de3577e569d027e	*image410	f03c0045a773b87a74	baa6798bc71b06b451bc3	af5bcc939b4ecd2c353ad
*image411	5e635e6fa77d80b	*image411	d994fe11b8a5db2d49	2eed9499fffe43250fc443	67dc87c7770bc498c7b7
*image412	15a22c7d76e9d87	*image412	eff7d1a17278546ab9	74f04c48570a5e9bb8df1	bd86efd0ae1124838df72
*image413	fadd3fdceb9e3f69f	*image413	5ab01507692998d17	c2caa46abd38c7d4d772c	22ce5eab3dac4bad01f5d
*image414	10f5ac29e0cd89a2	*image414	d44300e5bfa64adc9b	dac7b3d21a15d7da8da15	207ae4eae6530d21591d1
*image415	463429ee243aeb8	*image415	6422d3561cc550b9e	6ee211bc677569611687a	04ab44491fa6a9273715c
*image416	267a1d1371b6985	*image416	42ff2a6b24c52007c7	54b195cafa2b9cb6746dc	e9be04b331f2f7ccc2742
*image417	a03791b0406a2f7	*image417	08b50dbf70ef5620af	792655a614da2f67dd4c2	d0abaae8265c56c5bf70b
*image418	06a44622fbe8310	*image418	3d0eeefdab8c7efaafe	729576f0c622168d8cda6	7c772b624578899f4b38
*image419	435fb76e21b6126f	*image419	357cc8a2824919f92d	eba58a885d7480afba4cc	077708cd7d4727273dc5
*image42	4df5d6b9bbadbbf3	*image42	a2d28596f3cd87479b	ec935c16bef3b413967b7	c11ac6a6dd3500dd6b37
*image420	5ddae01eb605f58	*image420	23cbf64ad017d2893d	3db8b2af07f8e6cbaeb1ca	563b919c0f4754171122
*image421	824a7d12cc4e4fb3	*image421	3f13af5974d3faf8630	383fa2be0d77a15f01843	30091b7cd3b5c7013aad
*image422	ba5b78b10213a84	*image422	5fb5e0606b7088fb8f	63d454849f85c0bfa0e7fc	fc8669baddf510d13a1e3
*image423	b8a324d291be22a	*image423	c52c425c54cac024dc	62f47a1f80efa03ac31f7d	c83b29f2998811f156c2e
*image424	47b8ef927e8cc754	*image424	6b1bbd8621c8542b5	0d71f7781fa304ab264b2	8745ac4919f98f851c713
*image425	43272035cae0a33	*image425	2e52b585a96e11bc5f	b4bc51f5e8e82174a8575	2b2cce1f5128696e3be76
*image426	6a2ac0655c763ae	*image426	6f55008f68e76cf40b	6fc0f2d3bd000832400fe	c1eea09e3f76ac4acb3a8
*image427	6c53939f0bd2b6d	*image427	21fd0803158e01f029	ca37c2ff46eca14378ffdc	8a89c2830486c072db8ae
*image428	2a3b1a7d3108151	*image428	62d20304bcf25428f4	45f41b8adb558b306b9ef	8c97c656ed756d92afaa1
*image429	6baa203e0489fee0	*image429	b17298625abe0c056f	55179bdb69fb16630fa45	ac2e34e8d8e241de96647
*image43	e901ef2e0888245	*image43	b18328d0a1c0fca3ce	e6af92608ad0c4042529e	7e4b8191d55b6999176a
*image430	10d12220ad71b84	*image430	fedb44ce4c98ac26f9	79cd1d185f6976c4a4d75	b3d787235a75fbc3df71f
*image431	456e2d1e398db87	*image431	62e6d392863355323	ce753175debafc25130df	7924f07da24d09461020e
*image432	eafcb1a9d6d3143d	*image432	29bbf23c5d643ff864	ff856d2fd96437d4c3599	cd59cbb8887387394da9f
*image433	d1901604d930aec	*image433	581985c253dc17b1b	9691c9396ce46b03838b	9ef059495e9a86520214c
*image434	763e4c8ee866b6d	*image434	f8d772f763ff2a43ff4	8f7975c0b64ff75401625	94a3249413385ac61a1d
*image435	52d5e67ff7c19272	*image435	6506a196610694fe35	7f8d5982955566b9fe527	8fc390c93b4c476b4e445
*image436	cd87ee4ce14f7cd7	*image436	f0a9003cba6cc12803	dc608f898febc73af2bf3c	991e99e6d084e3ceafc9ef
*image437	9154190d0336455	*image437	820a8105a0ccbf7a26	44444d9c8b329566fd42b	16a5f97b8bb9a812d0ba5
*image438	646378f0ac620c8	*image438	29192f94780a2c3e32	51f6a03a28bf50d2d3423	e8a6d5e1f8281fffaf4c83
*image439	dfc506b19c168e5	*image439	39f137746668e19df3	ee6c3f620937dc1fe81bfe	b3d7fc8eeac3631f1c1c9a
*image44	84ed7f92aa65ae8f	*image44	7b7d02cc87b610519	f1522e75381b7ffc373a2f	bddc286b661c88b3252a
*image440	85d4741023d96d1	*image440	f868921e8aa3779708	cf7f50c33029e2f447f0a1	36605f1176afd42c12666
*image441	9208ea5c18b9bd5	*image441	7dbf4390bfd4f2d420	86c5a29aa1786a3c9842c	9e651ded87fafbb32c987
*image442	2fb89ed376f03ce9	*image442	6d0e9e7bc05d70fa44	6e19384bb0ca2f3f59ba5	509619853719e197edb4f
*image443	dcb9e8e7072091f	*image443	503a1eb5f27c9d0631	2435ba5ad89b960b0659	b1ffec54e11384bf6159d
*image444	0d7b0d102ea02d2	*image444	1b3123ac069127211	d7ce4249cb3fbc5dda8a5	288e5c70b928b0a45970
*image445	d9bf0ff6fe2ec385	*image445	d2d3b12b266880703	3e3d3f913a22cc1d821cb	853666c5de1dcbb69f533
*image446	187beda7d708edf	*image446	fba124c8fba534aaeb	eface66dc28be9305a731	840877a0fcdc54877d16b
*image447	d6ab721c014c518	*image447	2c2fbea6c012e28a48	78f7230cd233f94eed420	63433f7a095317633976
*image448	62db226b03d5f9b	*image448	e2d31ff71dcb90ba66	93809acab522d9e577235	85896c75ec84b6e8b371
*image449	1a262f814351869	*image449	ad60de0bb56163939	b62c6bc5180319b77d97	ae174cf971ff56c4757cb0
*image45	478a031eeb7c3cf4	*image45	64b80407eb8c67d4a	4171c4476773ec3b78e88	eec7ecc4c21ab883d5c57
*image450	772bad4dfc7ba42	*image450	04a6f15f0c3b83d853	313a8f261fba959a6f06a6	6bc5093ed188f60bfbcaf
*image451	85c34491ea7f532	*image451	f56b13dcfb20513baa	0043c7928d447ed33b17	52bfec2a529f0a9dae37df
*image452	73ce59007a9c42d	*image452	305b4b8517d10dd23	c4afabe4863cf5c0a0a0dc	27022fa1d2bf20751bfb9
*image453	6e5d7ba2f0b1195	*image453	13e1b90abe1c3411f4	8c3b97565a212c65fadc2	ac7512337c5997bab4dcc
*image454	8488b718e840c34	*image454	7731adbeda2223507	0d936589b12fe1a7e66ae	ae3c60d699a5ebd55bb28
*image455	0592492d4070ab8	*image455	0c5137caaf451f0145	19708cb27422585e7b6d	e457b23f7ba0b1f3dfbcc
*image456	330caeefb4c5af84	*image456	e9966ab45bcdc11aa7	770904e364bdb3c8133f9	88b6ca267b1919e70319

*image457	72401ab28308f50	*image457	aa00c2c9eb13e52799	7df76a9a245fad03df33a6	c5942588a9df9a1151957
*image458	9810b6c4fa0f57a4	*image458	0d8638c588dc74db8	b6c5d464e9e7e65a6cdd7	99c6985aaadc4cdb1d9c5
*image459	d289181edf23657	*image459	35d1774615b8bc686	727deb060592ec3b5c8a5	af027d3ea8554aceeefc30
*image46	b0ec86ee73028a0	*image46	50f62bee2adfc81b1f	ef295f1bef2814b326c74c	063ba15259982fc87629a
*image460	a6219ed722ebcdd	*image460	294c0d1a1af9ac433a	d2de54c5971d14fce8fe4	0bab9d93ffd50ee3eeab2
*image461	c29c1657be184c8	*image461	fa260321e20bf1ee07	7d5fc7ae8293872e2bc26	346ff7048ce9db7de4e1b
*image462	c9b37521ee713a7	*image462	8e4ab521e6582d5f2c	2b982d01ba337d2e6644	cad8713d2ee34e455aaad
Image #	Baseline MD5	Image #	IRFANVIEW	PhotoScape	Photoshop
*image463	89312f1f0e3347d3	*image463	586a6f4ee95f5bcd2c	0c1c1baac0e9f38767bc3	a13c94ff6e72cb0563fe53
*image464	542caee7fd4925f0	*image464	3ce240df814c43597f	0b9c12eb909f1ed148550	ac0d05e9c95f72d072e16
*image465	8adac3463bd3bd7	*image465	200eb98b3d1ae9024	abd14405bc6b3a93abeea	5c999c5d0d44a1077eabc
*image466	11b406ebd578084	*image466	7c66bd81333dc4dffa	94143b686a52b19e3181	cb9dac68929479b340a9
*image467	1148b967552e236	*image467	a67200d8722135f29	73739cb00a46aa5814751	054d2a07dc6f102542984
*image468	f31dc935f151900c	*image468	957131c39b8d1e197	cfdfd727e317cdab4349c	903d4175d1db934c3200
*image469	253993d9a1b31be	*image469	63868aa3f9792596e2	9d82e73e7cb053bea046c	0d3956c71de7b8443911f
*image47	2d0826a33c6f3ae4	*image47	0235abaa1d778c200	9e5afe00043cc93fb2482	a7ef87b271693da02a2a4
*image470	d905980df0e052a	*image470	e2cb758292982c3b2	6d90e64ccab94d1d9cc71	c670c40ca56ca4200338f
*image471	c8239080bdf15d4	*image471	db6b1af20a1762d150	b734c6059405496a5f8b2	b07e1bdabe867941cbbcc
*image472	ea34146992cad38	*image472	f5432fc55b7d3f834fe	9ab65feee63a024083815	74ad27b4c5acf9a412786
*image473	65799851bb1df3a	*image473	55d78f4e1e8633c56a	0ccdbd54be3690b7a733f	723c6c02d4cc0b3cdb7c1
*image474	f8e3a920218aba95	*image474	2bcd965161f4da35ed	d5720109c22b6f6df40bc	7d9d29de9394ea6f9f058
*image475	b80e57a00899a49	*image475	75bcf46240665c0849	e31cc0e9df10e4ff47bd9e	60cefddc7f3c378f44e4d7
*image476	74d35f08f6cb2a3e	*image476	f758470518a454b54	cf17c665c356fa3fa3edef	fb086be0cfe63066cea76
*image477	781a6fad434f62f9	*image477	27465e13767a2e6f46	257a5098efdabc478c796	ede2e92216795b5372b7
*image478	5c9ea7da5505a16	*image478	09337ca87dcd62dc7	1b334ca572757beb6591	537ca1d5379a50f167d7e
*image479	f38e829fb7ba5d0e	*image479	d5db046bcfe1b40221	86235e29174b0bee8999e	db3534bab0b2f46468d7c
*image48	e9d8f8a75cfd290f	*image48	74ad17fd3b44510d3	d74323b6bcb12d9346f56	3d231a1146d5f26e802f2
*image480	d8339fb7eef919a3	*image480	c2cb91cd65712bdf45	4293a8e76bfca07333d69	d69818bbf50db165b258
*image481	f0a93f2c97455a4e	*image481	6251a752712699806	16d17405f9b78bc88f3c6	a063ffdce3506b7b4ed2b
*image482	70b6d5f646720c8f	*image482	6a22c17781ec44a022	3e03f03432f5f3604c1af3	3234cf3fecee00bd33bb0
*image483	826d0a41d8c6883	*image483	a4af9ee2e277951d6e	3d45a6b1f22ea05319fcbf	fec588f95371056f99525
*image484	5f691d97c20db4bf	*image484	e269fff43a4b264586	aeb3cdb3b445e8a8b836c	3ae493bde3b9d3ad3538
*image485	2818717bca16308	*image485	c9d0f4d41fabfa17de	6f968b7cc4be999b1b19f	7ba6d1e56976f36d890a0
*image486	b6f1e4e7e474a5ce	*image486	66bcfd415b6d3375ca	3d044397e393dfa08ee01	48e271ae5e7f9335306d4
*image487	aa84e6a6fe540e63	*image487	019ab9f75a7358371c	46dde5e6827a6c8483cb1	b2ebb6126e5a21a84df29
*image488	35435d991a6fa75	*image488	e57d8fe3f900dec761	d4da9b43baf59c4b32d39	aec21fa7f36ce6e30e5ffa
*image489	c16271a8814beab	*image489	d0348dcbacf69a4702	eae2be2cc69755e882826	25b0e1dd0ce8fabfdf7c94
*image49	768147bd15b50b6	*image49	c21b7bf2d33d3501eb	1c9dcb68ae72826e8cb7c	4292ab59eaeb63953b11f
*image490	69ab7a36a328425	*image490	6f4ca361e71137bfbc	f27fbeab59049a08bca6d	b41cc7327d34a468716a
*image491	999906837e27ba3	*image491	eb23be24bec2a46fc5	7c708053c63c75f87849c	ec464a9a3921c1e2395fa
*image492	adbbdcd8fbc99fad	*image492	309c5f7790c5e3f62e	520820410c2f973e2114f	d491fdc219060836e8a58
*image493	5986b99382d3be3	*image493	2b472570fba492df5e	0e8bdfdbe497fea49a954	565b08cae13ed9566d2ef
*image494	06395110aaecd19	*image494	93136270a721879cf3	5a14357f0cef5055392c2	38745451d165121e0e1d
*image495	828226a844110c4	*image495	bb2308795dc31b417	1a0346e1101da0ea17503	aa0819db8593926b4778
*image496	69808dd136f73c6	*image496	790a4dadb95ef0f632	9071e362c223278881f17	d6a77025f32b6cacf7a03
*image497	6784ed00a8f15a1	*image497	7a24f20c7fb4369af1	354763578dc8e868b774	e23a687bd45e6c26c56fa
*image498	f13c6b5a5e85ba96	*image498	d253fc72da01d9f64a	7a2a04b83fbbb7eef4b40	b74c18105bcd61bced2ac
*image499	80504d05ab0ac0b	*image499	a8a646cadacc809b29	35b41997fec75c0fad8f1a	403ad556790a465cb3a6
*image5	b07381cc513c0a7	*image5	9815898359f12a086	49c9a4397d07a98b01cb8	85881b437585454cba02
*image50	f84fa815d730e655	*image50	16b81da211e0dcd0a	38572ebff7e888543f274	7bf0fe4ff267de4ee62f0a
*image500	887cea8a7cf66d6d	*image500	081127da6c30541eec	68e16a4dbff116dffd2f13	2927bf5009a6ed0a7b8a9
*image51	d1fa42864ea1537	*image51	b28f88e98a67586fbc	e20284cd47e8a06c2c9c3	0bc395459b820f5198f83
*image52	496387111987e66	*image52	5dab1bdfda60cca3bd	371142a5ba5d7c623b11e	42af3111e82d74a0d6be2
*image53	dfb4716389e1d9b	*image53	6c73d8f530b4bf5b26	99c1802eba26fd1a49eb4	a5b9c705c5beddf5a387d
*image54	514b58f6776e301	*image54	b0046d605dcf0a8b4d	840c0ba9a5a734e90e744	df44d7966ce24e32a33f0
*image55	90c6cb0979720a7	*image55	81f71fc1ef63167c31	ac13d013ea32bcaaacbe4	6773ae1d755708b9b466
*image56	b631e33e6c76de9	*image56	f13b835801251e8aa4	539fbcef4437d9f92a2ce9	7e94c044e4095aaf19b95
*image57	e8190db7bb9a7a5	*image57	5212f930118bca42d4	1ec10e683642afb0068d4	bfcef5d0f7b74ac32638e7
*image58	e39bdbcb98ed2e6	*image58	29cb5b9b7d980171e	1e28c4d3ac4e24336b7f6	7b35ce52e004f83368b99

*image59	b1b282a847b7813	*image59	8fd0337302ad1358d	77cda55403440c3ca4a44	d001ab9425922fb79d12
*image6	c3fe3aa6455c6233	*image6	19d076ffd7f3718dbf	25d07b02600e775607ec	a34870dee29969167941
*image60	1089b97b6393884	*image60	63c8a7c8c71cdcbbe6	cc1c3bafeee3bca250547	8892ba3729511aa10428
*image61	efeb1ad49ca91807	*image61	7416987832d19dadd	81b5e5ca1f53df419d118	4d2a8d8246a7c5bcedbfa
*image62	0191844301dd3fe	*image62	2858ebfde9c90a757f	5dddd5c8061916978baa	17a0d453a7da2d7fb9d57
*image63	591859c1c42efb4	*image63	fbb8503ba2ff6b5ab1	eee8272022336840c8054	cfe614d7653e2511b7b44
*image64	c3de8091cfe83130	*image64	abb8afe5424298fd86	7620687c386c1854b207	679be1e74e9b6529be2fa
Image #	Baseline MD5	Image #	IRFANVIEW	PhotoScape	Photoshop
*image65	fcd038c568f2e72e	*image65	59921327e7da513e8f	eb1cb79dfcfedefdada39c	98949b8d2ff23ad1c6d95
*image66	efbc67f6dc4b8c9b	*image66	dfcf1e8489da448805	d5de21f1ff9615bba6242	8fca7544ae5e643dba68a
*image67	65baa703e86cd8a	*image67	49d7fbb75356fde244	ef7c2fa80cb710e6bc39a7	5017802c944aac2937e2d
*image68	187efea557e2c7e3	*image68	4e37a55cb989690e0	33c2a14e1ed2ad03c3ddf	c1dcc3bbfbb875adb4453
*image69	05608741cf2d605	*image69	61b0b6bc6d59b1cdd	6e358a9d1af423383ff92	3963cd932410c12d4967
*image7	1df5814be07b08f1	*image7	eac7eb758f7e54d2fff	adb2fb2f66fb10740e514f	3fb4e493cf9fcb95b1108
*image70	b7ed9e36fa8fc6e7	*image70	afa371813b0386a04b	12d58ca99721c81ce6e09	894e25fced538f92ffc7ab
*image71	bd4443a88ed2819	*image71	e397e9c274ffe42afd5	8ad5f05e1e88fbfc9edc37	3b07bfc4d313f835cfd26
*image72	d6b0aea8dc008a8	*image72	85c281494dccfd0e9a	c43406a09d23c77ef7208	9b5f64d5019409f4c7d5f
*image73	eaffc9b469c41ceb	*image73	178f6a71a59460af2b	5d6ab541a6aba20d0d99f	9063cd3dc26fc39a56d5d
*image74	114c8ac83e5308c	*image74	8c5e4b57cb0dd947d	309b6207e18abd7f57afef	c926dc5b9119a9fa9e945
*image75	e992965b8320744	*image75	bfac54029f0627c67f	b2070edd71026514f51ce	2036e4edc62be5a5f9de5
*image76	8e924959a0c2e87	*image76	38573f32152c29fd73	4f6b30fd9e15f60de03cef	27680eb9214b15fe7ecfb
*image77	4f674e947c4b2b2	*image77	800a775cd306f0da1a	511f20aeebfe8198e9f7fc	dad8406392bcccb66685
*image78	ccc91437a16ea7d	*image78	b1c058f09f97c81566	39c6e0717b944a17a950e	a8e274f6864767fc2c32a
*image79	043b55c53602e7d	*image79	8695e474582c383b4	3ce292772b6ce775a9de1	d05728895e8184cc3eb9c
*image8	7c28343558256f4	*image8	2341eb6bc37888553	7457c231357879066520	7395cc49ce39fd70d2c90
*image80	2cf2c06bc14d730	*image80	2b5aa7e5af7a647a5e	806b946cf96a862a8ae38	f3b2b93e243928895083e
*image81	62fe2969635c96fa	*image81	b918a6e8596a442b7	cfb550375a57de0388b35	fe7e9b65f719dd0d7c548
*image82	328a1339af154ea0	*image82	afcff3ec6cda2e6180f	908a23910c37f748e9a46	4a025c5d1b51289884f88
*image83	4c38bdcf1b5dcb1	*image83	e3758eb3fa91c59bc5	6b8dbe07abaf0d752c065	aa7dc942aece2b76df468
*image84	fc0858851f9a0ea8	*image84	f568c9ed56f247daa7	cf0a204075a08b55fd3e3	36a451d1faac6a514b90c
*image85	b0bc9789ebbb12d	*image85	585bc42b674b92b7a	6326c1735ef7f8274b794	420cc156556c1c1ca0f48
*image86	6fba21afb60ed78a	*image86	4af9ffb0ea0bf1132cc	e1da9e6976a38464c97cf	747c1b9687bbf6d693aa8
*image87	d14cda0ca9b7c1fd	*image87	bc9b36c7ec4853db2	95f6551d8dec657a9ac87	472e526dc1453ac30ac33
*image88	b288b6046ad6a77	*image88	9ba06a149541c196aa	4e6ff98ba3bec32579d34	4b441a30774bc487167e
*image89	52b72aa56ab0e23	*image89	13ade436dc673d248	bca0900cd2d046e8d18e9	7ffe8328a9c5e7da89006
*image9	a4e5f4e5a4f4487b	*image9	5acb21c013fc8c43e5	e6f0d177bdcd2fe12bd15	91079b35acdd70349baa
*image90	7728234acdcc208	*image90	534acddda0d447ee2d	e63e92458a376ae5ca8c6	eae658fe61d57baff12f8c
*image91	f6a72ae9921bc749	*image91	9cf48c5468ba5ce76f	503197f9493a38c5778f0	7dc4d4afcd7fd2e8d1bfb
*image92	a6e8b82a686537b	*image92	3e0277b6f925d93c8f	04ef453cd6e398c10a176	8c6f6cf7f3fb05fa8ab323
*image93	49353f35adcf016f	*image93	8c523273cbea9857ab	cf403ac02e02553d305cd	8b8dd0172d7455519d31
*image94	cb8dad232f53811	*image94	20a22b3ea7d6f2a711	61d4ff35a962beaba84c9	66a6a2527744566aeb5ab
*image95	892856c129bd06a	*image95	292116cf4e4f65f5e7	de60db581c0834a7adade	e34a63881a513f58715e7
*image96	dcb173d8e74839e	*image96	9003db99b398a64d5	0264682695e212a5830f4	9563edc4f90f112426f19
*image97	4197242c9263c36	*image97	eee8cae9842a83947e	c024998cbb649b0b9ebfa	1c695a540b829d24ca7ac
*image98	75f59b8133b3cb8f	*image98	72eb5457d9a765775	8d438ce57b983ecacac5e	391beb9064621d346168
*image99	e9a2fa450573cc60	*image99	3d8f9b4d394c8fb5bf	7f5203b212014bccdfaf8	0368e93d6bdea251ab1fe

Appendix Q: Wald-Wolfowitz Runs test Details

James MacCaffrey tells us that "you cannot ever state with certainty that a given pattern was generated randomly; you can only test to see if there is statistical evidence that the pattern was not generated randomly". (McCaffrey 2006) According to the NIST handbook on Statistics the runs test is to be used in determining the null hypotheses H_0 that the sequence of numbers is the result of a random process. (Filliben and Heckert 2010) A run (R) is a set of values that are either all above or below the mean value of the dataset. In the execution of the Runs Test; positive runs are consecutive numbers above the mean and negative runs are consecutive numbers below the mean. From proceedings paper, (Kohler, et al. 2008), the calculations of the runs test are such:

$$\mathrm{E}[R] = \frac{2N^- N^+}{N} + 1$$

The error rate for the algorithm is calculated as shown above where R is a random variable that equals the runs of a random series of numbers either above or below the mean and N is the number of observations in the series. The Variance of R is computed as shown in the next formula,

$$\mathrm{Var}[R] = \frac{2N^- N^+(2N^- N^+ - N)}{N^2(N-1)}$$

With the error rate and the variance computed, the Z test value is determined by subtracting the error rate from the observed number of runs above and below the mean divided by the square root of the variance of R.

$$Z_{test} = \frac{r - \mathrm{E}[R]}{\sqrt{\mathrm{Var}[R]}}$$

According to Kolher et al. and the NIST handbook at a "5% significance level, a z-score with an absolute value greater than 1.96 indicates non-randomness". (Kohler, et al. 2008) (Filliben and Heckert 2010)

Appendix R: Chi-Squared Critical Values

Courtesy of the NIST Statistics handbook (Filliben and Heckert 2010)
Critical values of CHI^2 distribution with ν degrees of freedom

ν	Probability of exceeding the critical value 0.10	0.05	0.025	0.01	0.001
1	2.706	3.841	5.024	6.635	10.828
2	4.605	5.991	7.378	9.210	13.816
3	6.251	7.815	9.348	11.345	16.266
4	7.779	9.488	11.143	13.277	18.467
5	9.236	11.070	12.833	15.086	20.515
6	10.645	12.592	14.449	16.812	22.458
7	12.017	14.067	16.013	18.475	24.322
8	13.362	15.507	17.535	20.090	26.125
9	14.684	16.919	19.023	21.666	27.877
10	15.987	18.307	20.483	23.209	29.588
11	17.275	19.675	21.920	24.725	31.264
12	18.549	21.026	23.337	26.217	32.910
13	19.812	22.362	24.736	27.688	34.528
14	21.064	23.685	26.119	29.141	36.123
15	22.307	24.996	27.488	30.578	37.697
16	23.542	26.296	28.845	32.000	39.252
17	24.769	27.587	30.191	33.409	40.790
18	25.989	28.869	31.526	34.805	42.312
19	27.204	30.144	32.852	36.191	43.820
20	28.412	31.410	34.170	37.566	45.315
21	29.615	32.671	35.479	38.932	46.797
22	30.813	33.924	36.781	40.289	48.268
23	32.007	35.172	38.076	41.638	49.728
24	33.196	36.415	39.364	42.980	51.179
25	34.382	37.652	40.646	44.314	52.620
26	35.563	38.885	41.923	45.642	54.052
27	36.741	40.113	43.195	46.963	55.476
28	37.916	41.337	44.461	48.278	56.892
29	39.087	42.557	45.722	49.588	58.301
30	40.256	43.773	46.979	50.892	59.703
31	41.422	44.985	48.232	52.191	61.098
32	42.585	46.194	49.480	53.486	62.487
33	43.745	47.400	50.725	54.776	63.870
34	44.903	48.602	51.966	56.061	65.247
35	46.059	49.802	53.203	57.342	66.619
36	47.212	50.998	54.437	58.619	67.985
37	48.363	52.192	55.668	59.893	69.347
38	49.513	53.384	56.896	61.162	70.703
39	50.660	54.572	58.120	62.428	72.055

40	51.805	55.758	59.342	63.691	73.402
41	52.949	56.942	60.561	64.950	74.745
42	54.090	58.124	61.777	66.206	76.084
43	55.230	59.304	62.990	67.459	77.419
44	56.369	60.481	64.201	68.710	78.750
45	57.505	61.656	65.410	69.957	80.077
46	58.641	62.830	66.617	71.201	81.400

47	59.774	64.001	67.821	72.443	82.720
48	60.907	65.171	69.023	73.683	84.037
49	62.038	66.339	70.222	74.919	85.351
50	63.167	67.505	71.420	76.154	86.661
51	64.295	68.669	72.616	77.386	87.968
52	65.422	69.832	73.810	78.616	89.272
53	66.548	70.993	75.002	79.843	90.573
54	67.673	72.153	76.192	81.069	91.872
55	68.796	73.311	77.380	82.292	93.168
56	69.919	74.468	78.567	83.513	94.461
57	71.040	75.624	79.752	84.733	95.751
58	72.160	76.778	80.936	85.950	97.039
59	73.279	77.931	82.117	87.166	98.324
60	74.397	79.082	83.298	88.379	99.607
61	75.514	80.232	84.476	89.591	100.888
62	76.630	81.381	85.654	90.802	102.166
63	77.745	82.529	86.830	92.010	103.442
64	78.860	83.675	88.004	93.217	104.716
65	79.973	84.821	89.177	94.422	105.988
66	81.085	85.965	90.349	95.626	107.258
67	82.197	87.108	91.519	96.828	108.526
68	83.308	88.250	92.689	98.028	109.791
69	84.418	89.391	93.856	99.228	111.055
70	85.527	90.531	95.023	100.425	112.317
71	86.635	91.670	96.189	101.621	113.577
72	87.743	92.808	97.353	102.816	114.835
73	88.850	93.945	98.516	104.010	116.092
74	89.956	95.081	99.678	105.202	117.346
75	91.061	96.217	100.839	106.393	118.599
76	92.166	97.351	101.999	107.583	119.850
77	93.270	98.484	103.158	108.771	121.100
78	94.374	99.617	104.316	109.958	122.348
79	95.476	100.749	105.473	111.144	123.594
80	96.578	101.879	106.629	112.329	124.839
81	97.680	103.010	107.783	113.512	126.083
82	98.780	104.139	108.937	114.695	127.324
83	99.880	105.267	110.090	115.876	128.565
84	100.980	106.395	111.242	117.057	129.804
85	102.079	107.522	112.393	118.236	131.041
86	103.177	108.648	113.544	119.414	132.277
87	104.275	109.773	114.693	120.591	133.512
88	105.372	110.898	115.841	121.767	134.746
89	106.469	112.022	116.989	122.942	135.978
90	107.565	113.145	118.136	124.116	137.208
91	108.661	114.268	119.282	125.289	138.438
92	109.756	115.390	120.427	126.462	139.666
93	110.850	116.511	121.571	127.633	140.893
94	111.944	117.632	122.715	128.803	142.119
95	113.038	118.752	123.858	129.973	143.344
96	114.131	119.871	125.000	131.141	144.567

97	115.223	120.990	126.141	132.309	145.789
98	116.315	122.108	127.282	133.476	147.010
99	117.407	123.225	128.422	134.642	148.230
100	118.498	124.342	129.561	135.807	149.449
100	118.498	124.342	129.561	135.807	149.449

Appendix S: Frequency Tables for ZVMCUs

Frequency Tables

Results for layer #1
Frequency distribution of ZVMCUs 3430

Partitions	Count	Cumulative Count	Percent	Cumulative %
-304.657258064516 To 304.657258064516	2782	2782	0.811078717	0.811078717
304.657258064516 To 913.971774193548	215	2997	0.062682216	0.873760933
913.971774193548 To 1523.28629032258	87	3084	0.025364431	0.899125364
1523.28629032258 To 2132.60080645161	53	3137	0.015451895	0.914577259
2132.60080645161 To 2741.91532258065	39	3176	0.011370262	0.925947522
2741.91532258065 To 3351.22983870968	25	3201	0.00728863	0.933236152
3351.22983870968 To 3960.54435483871	23	3224	0.006705539	0.939941691
3960.54435483871 To 4569.85887096774	20	3244	0.005830904	0.945772595
4569.85887096774 To 5179.17338709677	14	3258	0.004081633	0.949854227
5179.17338709677 To 5788.48790322581	12	3270	0.003498542	0.95335277
5788.48790322581 To 6397.80241935484	22	3292	0.006413994	0.959766764
6397.80241935484 To 7007.11693548387	11	3303	0.003206997	0.962973761
7007.11693548387 To 7616.4314516129	13	3316	0.003790087	0.966763848
7616.4314516129 To 8225.74596774194	6	3322	0.001749271	0.96851312
8225.74596774194 To 8835.06048387097	10	3332	0.002915452	0.971428571
8835.06048387097 To 9444.375	6	3338	0.001749271	0.973177843
9444.375 To 10053.689516129	9	3347	0.002623907	0.975801749
10053.689516129 To 10663.0040322581	5	3352	0.001457726	0.977259475
10663.0040322581 To 11272.3185483871	4	3356	0.001166181	0.978425656
11272.3185483871 To 11881.6330645161	2	3358	0.00058309	0.979008746
11881.6330645161 To 12490.9475806452	4	3362	0.001166181	0.980174927
12490.9475806452 To 13100.2620967742	3	3365	0.000874636	0.981049563
13100.2620967742 To 13709.5766129032	5	3370	0.001457726	0.982507289
13709.5766129032 To 14318.8911290323	5	3375	0.001457726	0.983965015
14318.8911290323 To 14928.2056451613	2	3377	0.00058309	0.984548105
14928.2056451613 To 15537.5201612903	4	3381	0.001166181	0.985714286
15537.5201612903 To 16146.8346774194	1	3382	0.000291545	0.986005831
16756.1491935484 To 17365.4637096774	5	3387	0.001457726	0.987463557
17365.4637096774 To 17974.7782258065	3	3390	0.000874636	0.988338192
17974.7782258065 To 18584.0927419355	2	3392	0.00058309	0.988921283
18584.0927419355 To 19193.4072580645	1	3393	0.000291545	0.989212828
19193.4072580645 To 19802.7217741936	1	3394	0.000291545	0.989504373
19802.7217741936 To 20412.0362903226	1	3395	0.000291545	0.989795918
20412.0362903226 To 21021.3508064516	2	3397	0.00058309	0.990379009
21021.3508064516 To 21630.6653225807	1	3398	0.000291545	0.990670554
21630.6653225807 To 22239.9798387097	1	3399	0.000291545	0.990962099
22239.9798387097 To 22849.2943548387	2	3401	0.00058309	0.99154519
23458.6088709678 To 24067.9233870968	2	3403	0.00058309	0.99212828
24067.9233870968 To 24677.2379032258	1	3404	0.000291545	0.992419825
24677.2379032258 To 25286.5524193549	2	3406	0.00058309	0.993002915
25895.8669354839 To 26505.1814516129	1	3407	0.000291545	0.993294461
26505.1814516129 To 27114.495967742	2	3409	0.00058309	0.993877551
27114.495967742 To 27723.810483871	1	3410	0.000291545	0.994169096
28333.125 To 28942.4395161291	2	3412	0.00058309	0.994752187
30770.3830645162 To 31379.6975806452	1	3413	0.000291545	0.995043732
31379.6975806452 To 31989.0120967742	1	3414	0.000291545	0.995335277
32598.3266129033 To 33207.6411290323	2	3416	0.00058309	0.995918367
33207.6411290323 To 33816.9556451613	1	3417	0.000291545	0.996209913
35035.5846774194 To 35644.8991935484	2	3419	0.00058309	0.996793003
38691.4717741936 To 39300.7862903226	1	3420	0.000291545	0.997084548
39910.1008064516 To 40519.4153225807	1	3421	0.000291545	0.997376093
42347.3588709678 To 42956.6733870968	1	3422	0.000291545	0.997667638
52096.3911290322 To 52705.7056451613	1	3423	0.000291545	0.997959184
58189.5362903225 To 58798.8508064516	1	3424	0.000291545	0.998250729
60017.4798387096 To 60626.7943548387	1	3425	0.000291545	0.998542274
66719.939516129 To 67329.254032258	2	3427	0.00058309	0.999125364
70985.1411290322 To 71594.4556451612	1	3428	0.000291545	0.99941691
71594.4556451612 To 72203.7701612902	1	3429	0.000291545	0.999708455
75250.3427419354 To 75859.6572580644	1	3430	0.000291545	1