Robust Image Hash Spoofing

Azadeh Amir Asgari

Department of Applied Signal Processing
Blekinge Institute of Technology
SE – 37179, Karlskrona, Sweden
www.bth.se

Fraunhofer-Institut für Sichere Informationstechnologie
Rheinstrasse 75, 64295 Darmstadt, Germany
Phone: +49 6151 8690
www.sit.fraunhofer.de
This thesis is submitted to the Department of Applied Signal Processing at Blekinge Institute of Technology of Sweden in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering with Emphasis on Signal Processing.

**Contact Information:**

**Author:**
Azadeh Amir Asgari  
E-mail: aziamirasgari@gmail.com

**Advisors:**

Dr. -Ing. Martin Steinebach  
Fraunhofer-Institut für Sichere Informationstechnologie  
Media Security and IT Forensics  
Phone: +49-6151-869-349  
E-mail: martin.steinebach@sit.fraunhofer.de  
Address: CASED building, Mornewegstrasse 32, 5th floor

Prof. Dr. Stefan Katzenbeisser  
CASED  
Security Engineering Group  
Computer Science Department  
Technische Universität Darmstadt  
Phone: +49-6151-16-5016  
E-mail: katzenbeisser@seceng.informatik.tu-darmstadt.de  
Address: CASED building, Mornewegstrasse 32, 4th floor, room 4.3.25

Dr. Huajian Liu  
Fraunhofer-Institut für Sichere Informationstechnologie  
E-mail: huajian.liu@sit.fraunhofer.de  
Address: CASED building, Mornewegstrasse 32, 5th floor

**Supervisor:**

Dr. Benny Lövström  
Blekinge Institute of Technology  
Senior Lecturer  
Phone: +46-455-385704  
E-mail: benny lovstrom@bth.se  
Address: Room J3612-Blekinge Tekniska Högskola, 371 79 Karlskrona
ABSTRACT

With the intensively increasing of digital media new challenges has been created for authentication and protection of digital intellectual property. A hash function extracts certain features of a multimedia object e.g. an image and maps it to a fixed string of bits. A perceptual hash function unlike normal cryptographic hash is change tolerant for image processing techniques. Perceptual hash function also referred to as robust hash, like any other algorithm is prone to errors. These errors are false negative and false positive, of which false positive error is neglected compared to false negative errors. False positive occurs when an unknown object is identified as known. In this work a new method for raising false alarms in robust hash function is devised for evaluation purposes i.e. this algorithm modifies hash key of a target image to resemble a different image’s hash key without any significant loss of quality to the modified image. This algorithm is implemented in MATLAB using block mean value based hash function and successfully reduces hamming distance between target image and modified image with a good result and without significant loss to attacked imaged quality.

Keywords: Robust hash function, Hamming distance, Block mean value, Spoofing attack.
ACKNOWLEDGMENT

I would like to express my sincere thanks to Dr.-Ing. Martin Steinebach who gave me the opportunity to be involved his research team and provided all the necessary facilities and the valuable guidance and encouragement. I sincerely wish to express my gratitude and appreciation to Prof. Stefan Katzenbeisser who guided me from the commencement to the successful completion of my study. I also thank my supervisor Prof. Benny Lövström at Blekinge Institute of Technology for his guidance and support through this thesis. I would like also to thank Dr. Huajian for his comments and suggestions. Last but not least, I would like to state my sincere thanks to my parents and my family especially Arash Hassanzadeh who without their unconditional support, I would not have been able to reach this point.
Table of Contents

ABSTRACT ................................................................................................................................. I
ACKNOWLEDGMENT ..................................................................................................................... II
Table Of Contents ....................................................................................................................... III
List Of Figures ........................................................................................................................... IV
List Of Tables ........................................................................................................................... V
1. Introduction ..................................................................................................................................... 1
2. Background ..................................................................................................................................... 2
   2.1. INTRODUCTION ...................................................................................................................... 2
   2.2. AUTHENTICATION ................................................................................................................. 2
   2.3. CRYPTOGRAPHIC HASH FUNCTION ..................................................................................... 3
   2.4. HASH FUNCTIONS AND MULTIMEDIA ................................................................................ 6
   2.5. PERCEPTUAL HASH FUNCTIONS ......................................................................................... 7
   2.6. COMPARISON PERCEPTUAL HASH FUNCTIONS ................................................................... 11
   2.7. SURVEY OF RELATED WORKS .......................................................................................... 12
   2.8. METRICS ............................................................................................................................. 15
   2.9. FALSE NEGATIVE/FALSE POSITIVE .................................................................................. 17
   2.10. IMAGE HASH SPOOFING .................................................................................................... 17
3. Implementation ........................................................................................................................... 18
   3.1. REQUIREMENTS .................................................................................................................... 18
   3.2. PRELIMINARY WORK .......................................................................................................... 19
   3.3. DESIGN .................................................................................................................................. 23
      3.3.1 Stage1: ............................................................................................................................. 23
      3.3.2 Stage2: ............................................................................................................................. 27
   3.4. MATLAB® CODE ................................................................................................................ 28
4. Verification and Results ............................................................................................................. 30
   4.1. METHOD 1 ............................................................................................................................ 30
   4.2. METHOD 2 ............................................................................................................................ 41
5. Conclusion and Future Work ....................................................................................................... 50
   5.1 CONCLUSION ......................................................................................................................... 50
   5.2 FUTURE WORK ...................................................................................................................... 50
References: ..................................................................................................................................... 51
Appendix A: Variance Thresholding ............................................................................................... 53
Appendix B: Tables of Hamming Distance Reduction by stage .................................................... 55
Appendix C: MATLAB Codes ......................................................................................................... 61
List Of Figures

FIGURE 1: BOX PLOT ................................................................. 19
FIGURE 2: HAMMING DISTANCE HIST OGRAM FOR SELECTION NO.14. YELLOW LINE IS THE THRESHOLD APPLIED.............. 21
FIGURE 3: HAMMING DISTANCE HIST OGRAM IN SELECTION NO.14 FOR VALUES BELOW THE THRESHOLD................. 21
FIGURE 4: HAMMING DISTANCE BOX PLOT FOR ALL SELECTIONS................................................................. 22
FIGURE 5: HAMMING DISTANCE BOX PLOT FOR SELECTION NO.14........................................................................ 22
FIGURE 6: TARGET IMAGE FIGURE 7: SEGMENTED TARGET IMAGE .............................................................................. 23
FIGURE 8: TARGET IMAGE HASH ................................................................. 24
FIGURE 9: ATTACKED IMAGE FIGURE 10: ATTACKED IMAGE HASH........................................................................ 24
FIGURE 11: MASKED ATTACKED IMAGE FIGURE 12: MASKED ATTACKED IMAGE HASH........................................... 25
FIGURE 13: FILTERED ATTACKED IMAGE HASH FIGURE 14: FILTERED ATTACKED IMAGE........................................ 25
FIGURE 15: SAMPLES OF PREDEFINED HASH MASKS.................................................................................................. 26
FIGURE 16: PSNR HIST OGRAM FOR 163 ATTACKED IMAGE................................................................. 31
FIGURE 17: METHOD 1–IMAGE 1 ............................................................................................................................. 32
FIGURE 18: METHOD 1–IMAGE 1 ............................................................................................................................. 33
FIGURE 19: METHOD 1–IMAGE 2 ............................................................................................................................. 34
FIGURE 20: METHOD 1–IMAGE 2 ............................................................................................................................. 35
FIGURE 21: METHOD 1–IMAGE 3 ............................................................................................................................. 36
FIGURE 22: METHOD 1–IMAGE 3 ............................................................................................................................. 37
FIGURE 23: METHOD 1–IMAGE 4 ............................................................................................................................. 38
FIGURE 24: METHOD 1–IMAGE 4 ............................................................................................................................. 39
FIGURE 25: HIST OGRAM OF HAMMING DISTANCE REDUCTION .................................................................................. 40
FIGURE 26: BOX PLOT OF HAMMING DISTANCE REDUCTION .................................................................................. 41
FIGURE 27: METHOD 2–IMAGE 1 ............................................................................................................................. 42
FIGURE 28: METHOD 2–IMAGE 1 ............................................................................................................................. 43
FIGURE 29: METHOD 2–IMAGE 2 ............................................................................................................................. 44
FIGURE 30: METHOD 2–IMAGE 2 ............................................................................................................................. 45
FIGURE 31: METHOD 2–IMAGE 3 ............................................................................................................................. 46
FIGURE 32: METHOD 2–IMAGE 3 ............................................................................................................................. 47
FIGURE 33: HIST OGRAM OF HAMMING DISTANCE REDUCTION .................................................................................. 48
FIGURE 34: BOX PLOT OF HAMMING DISTANCE REDUCTION .................................................................................. 49
List Of Tables

TABLE 1: HAMMING DISTANCE EXAMPLES .................................................................................................................................16
TABLE 2: 30 SELECTED IMAGES HAMMING DISTANCE TO OTHER IMAGES ..............................................................................20
TABLE 3: VARIANCE CRITERION FOR QUALITY CONTROL ........................................................................................................27
TABLE 4: HAMMING DISTANCE REDUCTION FOR METHOD 1 - IMAGE 1 ..................................................................................................33
TABLE 5: HAMMING DISTANCE REDUCTION FOR METHOD 1 - IMAGE 2 ................................................................................................35
TABLE 6: HAMMING DISTANCE REDUCTION FOR METHOD 1 - IMAGE 3 .............................................................................................37
TABLE 7: HAMMING DISTANCE REDUCTION FOR METHOD 1 - IMAGE 4 .............................................................................................39
TABLE 8: HAMMING DISTANCE REDUCTION FOR METHOD 2 - IMAGE 1 .............................................................................................43
TABLE 9: HAMMING DISTANCE REDUCTION FOR METHOD 2 - IMAGE 2 .............................................................................................45
TABLE 10: HAMMING DISTANCE REDUCTION FOR METHOD 2 - IMAGE 3 ............................................................................................47
1. Introduction

Throughout the last two decades, improving the technology of digital media imposes new problems for managing large multimedia databases in terms of authentication and managing intellectual properties as well as broadcast monitoring and network filtering. Robust Hashing is a technology as a change tolerant alternative to cryptographic hashes since normal cryptographic hashing methods are error prone to image processing techniques. As any other field of technology, perceptual hash functions have two major drawbacks to be named "false negatives" and "false positives". False negatives occur if a known object is not recognized and false positives occur if an unknown object is recognized as known content. False positive errors are neglected compared to false negative errors in the field of cryptography.

This thesis work is intended to devise a new method in order to raise false positive alarms in robust image hashing for evaluation purposes through global and local modification of another image to have a hash similar to a target image. This method is implemented in MATLAB by block mean value based Hashing algorithm.

In the chapter 2 basic concepts such as perceptual hash function, hamming distance, comparison between several hashing methods have been discussed. Block mean value based hashing method is completely described in the latter part of the chapter since it has been used through this work.

In chapter 3, implementation of the developed method is discussed in full depth and at chapter 4 the devised algorithm is verified and results have been discussed. In the last chapter we have a conclusion on this job.
2. Background

2.1. Introduction

Due to rapid growth in digital media, content authentication of multimedia is a major concern and is used extensively. It has been said that hiding the information or protecting the privacy has a history as long as the writing itself. Mankind has been trying many ways to hide information during its history which is called steganography, i.e., the art or practice of concealing a message, image, or file within another message, image, or file by replacing fixed symbols, and cryptology or cipher. By changing technology from handwritten words on paper sent by courier to the communication of the information via both local and worldwide communication networks and the saving and processing in the form of digital data on computers definitely has enhanced the risk of exposing of information to eavesdropping. Cryptography was the only solution which has been borrowed from the secret world of army commanders and politicians into the global commercial applications. Beside the concealing or privacy protection, it should be noted that both the contents and the originator of the information are not changed. Both of these conditions are expressed in the term “authentication.” A hacker who attempts to change contents or origin of information is called an active attacker. The increasing of relative importance of this threat might be captured by the advent of malicious software programs. These digital threats are best exemplified in the form of computer viruses. Others include worms, Trojan horses, and logical bombs.

2.2. Authentication and related history

The origin of the authentication is from the Greek word αὐθεντικός which means real or genuine and in network security discourse it means confirming the identity of multimedia object. Authentication is an approach to protect communicating parties from a third party attack. But, when communicating parties are distrustful to each other and try to refuse their authorities, it is likely to emerge different threat. It means that sender or receiver try to change a message or deny to have sent or received data.

The protection of authenticity could include two aspects:

- The protection of the originator of the information, or in ISO terminology data origin authentication.

- The fact that the information has not been modified, or in ISO terminology the integrity of the information. There are two basic methods for protecting the authenticity of information.
2.3. Cryptographic Hash Function

As it is described below cryptographic hash functions are used to authentication process extensively and first the definition of hash functions is provided. Cryptographic hash function generates hash value from an arbitrary data array while keyed hash function uses a secret key as well [1].

Def. 2-1: *A hash function is [...] a function H which has, as a minimum, the following two properties:

- Compression - H maps an input \( x \) of arbitrary finite bit length, to an output \( H(x) \) of fixed bit length \( n \).
- Ease of computation - given \( H \) and an input \( x \), \( H(x) \) is easy to compute.*

Hash functions are mostly used to accelerate table lookup or data comparison tasks such as finding items in a database, detecting duplicated or similar records in a large file, finding similar stretches in DNA sequences [7]. The term hash functions has its historic roots in computer science where a hash function or a cryptographic hash function is denoted to any algorithm or subroutine that maps large data sets of variable length, called keys, to smaller data sets of a fixed length as uniformly as possible. The value which is returned by a hash function in cryptographic literature is called hash total, hash result, hash sums, hash code, imprint, (cryptographic) checksum, compression, compressed, encoding, seal, authenticator, authentication tag, fingerprint, test key, condensation, Message Integrity Code (MIC), message digest or simply hashes [15].

Cryptographic hash functions may be divided in two categories [1]:

1- Unkeyed Hash Functions
2- Keyed Hash Function

These two major approaches to protect authenticity of information are of a great importance and it deserves to have a more detailed explanation. The first approach is similar to the approach of a symmetric or asymmetric cipher in which the concealment of a large amount of data is according to the concealment and authenticity of a short key. Here, the authentication of the information would be based on the concealment and authenticity of a key. To do so, the information is compressed to the amount in length which is called a “hash code”. Consequently, the hash code is annexed to the information. The operation that accomplishes this process is called a hash function. The main concept of the protection of the integrity is to add redundancy to the information. The presence of this redundancy provides the receiver the ability to make the distinction between authentic information and bogus information. In order to assure the origin of
the data, a secret key associated to the origin should intervene in the process. The secret key could be included in the compression process or can be employed to protect the hash code and/or the information. In the first case the hash code is called a Message Authentication Code or MAC, while in the latter one it is called a Manipulation Detection Code or MDC.

The second approach includes making the authenticity (both integrity and origin authentication) of the information based on the authenticity of a Manipulation Detection Code or MDC. A well-known example of this method is a computer user calculating an MDC for all its important files. An individual may perform storage process in two ways: whether storing this collection of MDC’s on a hard disk, which is locked in his/her safe, or writing them down on a piece of paper. If he/she is to transfer the files to a distant friend, they can be sent easily and MDC’s can be communicated via telephone. Here, the authentication of the telephone channel is performed by voice identification. The second application for cryptographically secure hash functions is manifested through optimizing the digital signature schemes and the building up the digital signature schemes which are not based on a trapdoor one-way permutation. The optimization is performed via signing the MDC of a message rather than each bit or block. The description of the hash function might be public and it does not depend on any secret parameter. The advantages of this method are: the signature includes a fixed short length which minimizes the computational operation. Sometimes, it is possible to increase the security level of the signature scheme. In some cases, the hash function is even an embedded part of a scheme. In general, digital signature schemes based on one-way functions practically are less potent, but can be used as a substitution if one is not permitted or wants to employ a scheme according to a trapdoor one-way permutation. In the following the hash function will be denoted with \( h \), and its argument, i.e. the information to be protected, with \( X \). The image of \( X \) under the hash function \( h \) will be denoted with \( h(X) \) and the secret key with \( K \) [15].
**One-way hash function (OWHF)**

The first rough definition of an OWHF was evidently suggested by R. Merkle [10, 11] and M. Rabin [12].

A one-way hash function is a function $h$ satisfying the following conditions:

1. The description of $h$ must be publicly known and should not need any secret information for its operation (extension of Kerckhoffs’s principle).
2. The argument $X$ can be of arbitrary length and the result $h(X)$ has a fixed length of $n$ bits.
3. Given $h$ and $X$, the computation of $h(X)$ must be “easy”.
4. The hash function must be one-way in the sense that given a $Y$ in the image of $h$, it is “hard” to find a message $X$ such that $h(X) = Y$ and given $X$ and $h(X)$ it is “hard” to find a message $X' \neq X$ such that $h(X') = h(X)$.

**Collision resistant hash function (CRHF)**

The first formal definition of a CRHF was evidently suggested by I. Damgård [13, 14]. A rough definition was suggested by R. Merkle in [11].

A collision resistant hash function is a function $h$ satisfying the following conditions:
1. The description of \( h \) must be publicly known and should not require any secret information for its operation (extension of Kerckhoffs’ principle).

2. The argument \( X \) can be of arbitrary length and the result \( h(X) \) has a fixed length of \( n \) bits.

3. Given \( h \) and \( X \), the computation of \( h(X) \) must be “easy”.

4. The hash function must be one-way in the sense that given a \( Y \) in the image of \( h \), it is “hard” to find a message \( X \) such that \( h(X) = Y \) and given \( X \) and \( h(X) \) it is “hard” to find a message \( X’ \neq X \) such that \( h(X’) = h(X) \).

5. The hash function must be collision resistant; this means that it is “hard” to find two distinct messages that hash to the same result.

**Message Authentication Code (MAC)**

Message Authentication Codes have been used for a long time in the banking community and are thus older than the open research in cryptology that started in the mid-seventies. However, MAC’s with good cryptographic properties were only introduced after the start of open cryptologic research.

A MAC is a function satisfying the following conditions:

1. The description of \( h \) must be publicly known and the only secret information lies in the key (extension of Kerckhoffs’s principle).

2. The argument \( X \) can be of arbitrary length and the result \( h(K,X) \) has a fixed length of \( n \) bits.

3. Given \( h \), \( X \) and \( K \), the computation of \( h(K,X) \) should be “easy”.

4. Given \( h \) and \( X \), it is “hard” to determine \( h(K,X) \) with a probability of success “significantly higher” than \( 1/2^n \) with \( n \) the number of bits of hash code. Even when a large set of pairs \( \{X_i, h(K,X_i)\} \) is known, where the \( X_i \) have been selected by the opponent, it is “hard” to determine the key \( K \) or to compute \( h(K,X’) \) for any \( X’ \neq X_i \). This last attack is called an adaptive chosen text attack.

### 2.4. Hash Functions and Multimedia

A multimedia object e.g. an image can have different digital forms which from a human perception point of view, all are the same. These different forms are the consequence of wide range of image processing techniques such as cropping, compression, flipping, compression and equalization which each changes the binary form of the image. Due to this problem normal cryptographic hash function does not work for multimedia applications. On the other hand image identification methods, such as semantic models or face detection algorithms, although show good performance in identifying illegal multimedia objects have big drawbacks such as high computational complexity and high false alarm rates.

Therefore perceptual hash functions have been introduced in order to circumvent the problem of distinguishing perceptual equality of multimedia content. Recently due to extensive demand in the industry new perceptual hash functions have been introduced by scientific researchers. These
perceptual hash functions extract certain properties from the multimedia content and produce hash values based on these features. In order to have a measure for comparison between two perceptual hash values, there are defined functions for their distance/similarity scoring such as hamming distance, Bit Error Rate (BER) and Peak of Cross Correlation (PCC).

Perceptual hash functions are interdisciplinary field of research which includes cryptography, digital watermarking and signal processing. These perceptual hash functions because a lack of standard or uniform nomenclature may be addressed with different terms such as [2]:

- Fingerprint
- Passive fingerprint
- Perceptual checksum
- Robust hash
- Soft hash

In passive fingerprint the content of multimedia is unchanged but in active fingerprint the content differs from the original.

### 2.5. Perceptual Hash Functions

**Discrete Cosine Transform (DCT) based Hash function**

Discrete cosine transform (DCT) based functions utilize Fourier analysis theory in order to produce hash keys and like any other Fourier transform, it represents the finite sequence of data points as the weighted composition of sinusoids (cosine function) with different frequencies. DCT is similar to DFT operating on real data with even symmetry but on the other hand DCT in contrast with DFT uses real values. There are eight types of DCT and most common is type-II which simply is referred as DCT and type-III that is called inverse DCT.

**Def. 2.2:** Let $x[n], n=0,\ldots, N-1$, denote an $N$ point real sequence the DCT type-II is defined:

$$X(k) = \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} x(n) \cos \left( \frac{\pi}{N} \left( n + \frac{1}{2} \right) k \right), \quad k = 0, \ldots, N-1$$
The scaling factor $\sqrt{2/N}$ makes DCT matrix orthogonal but breaks the direct correspondence with a real-even DFT of half-shifted input. DCT is separable operation and may be computed along the axes separately.

In practice DCT of a time limited sequence may be computed by use of DCT matrix as below:

$$X(k) = \sum_{n=0}^{N-1} c(k,n).x(n), \quad k = 0, ..., N - 1$$

And $c(k,n)$ is defined as:

$$c(k,n) = \sqrt{\frac{2}{N}} \cos \left[ \frac{\pi}{N} \left( n + \frac{1}{2} \right) k \right], \quad k, n = 0, ..., N - 1$$

For a square image $I$, the two-dimensional is equal to one dimensional DCT while a single dimensional followed by the one dimensional DCT and it may be computed by using of DCT matrix $D$ as below:

$$DCT(I) = D \cdot I \cdot D^T$$

Various properties of the DCT can be utilized to create perceptual image hash functions. Low-frequency DCT coefficients of an image are mostly stable under image manipulations. [30] That is because most of the signal information tends to be concentrated in a few low-frequency components of the DCT. This property is also utilized by the JPEG image compression standard. [31] There, the two-dimensional type-II DCTs of $N \times N$ pixel blocks are computed and the results are quantized. [3] $N$ is typically 8 and elements closer to the top left corner represents lower frequencies in the horizontal and vertical direction in the image. DCT coefficients and their corresponding frequencies may be used as hash value of the image. Interest in DCT is because of its strong energy compaction property and this is important particularly in image processing applications like lossy image compression (JPEG is the simplest example). This lies in the fact that normally most of image energy is concentrated in lower frequency component of DCT. Performance of this method is discussed later.

**Mar-Hildreth operator based Hash functions**

As mentioned before a perceptual hash function extracts certain feature of image and uses them to produce fixed length string namely hash key. Mar-Hildreth operator algorithm core is extracting edges of image and use them to produce the hash key. Edge definition is depending on the context of application but it can be defined as contours or boundaries that separate different
regions of image and regions can be classified based on texture, color and luminance. Normally luminance is used and the result of edge detection algorithm is named edge map. Edge map contains information about image classification feature as well as amplitude and orientation information and for luminance case, first and second derivatives with respect to spatial location are used for edge detection. So first derivative (gradient based) approach is to locate the positions that first derivative of luminance (gray scale level) are at local extremum and second derivative approach (Laplacian based method) is to identify zero crossing points of luminance function. Since these methods are used for two dimensional images some point must be considered. Discrete nature of digital image implies approximation of derivation. Image has additional property of direction so directionally sensitive edge detector is used for some applications.

Since edge detection is a high pass filter, image noise is a problem and wide range of algorithms have been proposed to deal with this effect. Generally detector error increases with noise. There is a trade-off between correct detection of edges and their location. The reason lies in the fact that good localization needs small spatial filter and conversely better noise suppression is obtained by spatially large filter.

Considering these points one simple approach is the one invented by David Marr and Ellen C. Hildreth which is convolving image with the Laplacian of the Gaussian function i.e. if we define Laplacian filter as provided by definition 2.3 then estimate of given image’s Laplacian can be computed by convolution of filter kernel by image. The Laplacian-of-Gaussian image operator is sometimes also referred to as the Mexican hat wavelet due to its visual shape when turned upside-down.

**Def. 2.3:** The Laplacian of Gaussian (LoG), denoted as \( h_c(x, y) \) filter can be defined as

\[
h_c(x, y) = \nabla^2 g_c(x, y) = \frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} \cdot e^{-\frac{x^2+y^2}{2\sigma^2}}
\]

Where \( g_c(x, y) \) is Gaussian filter

\[
g_c(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}}
\]

Then the LoG estimate of an image is obtained by convolving the Log with image \( f(x, y) \).

\[
\nabla^2 f(n1, n2) = f(n1, n2) * g_c(n1, n2)
\]

The implementation of LoG filter in digital domain may be achieved by sampling of kernel in spatial domain after choosing a value for \( \sigma \). Computation cost can be reduced by utilizing the fact
that LoG filter is a separable filter and using 1-D convolution instead of 2-D counterpart. The Marr–Hildreth operator, however, has its drawbacks. It produces estimates that do not correspond to edges, so-called "false edges", and the localization error may be severe at curved edges.

Radial Variance based Hash functions

The Radon transform is calculated by taking the integrals of a two-dimensional image $f(x, y)$ along a set of lines with different directions. The line integral along a particular direction with angle $\theta$ is called a projection. The line integral of the function $f(x, y)$ along the line $L$ defined by direction $\theta$ and the distance $x'$ from the origin in the coordinates $(x', y')$ is given by

$$R_\theta(x') = \int_L f(x' \cos \theta - y' \sin \theta , x' \sin \theta + y' \cos \theta) dy'$$

To apply radon transform to discrete images the line integral along the $d = x \cos \theta + y \sin \theta$ can be approximated by variance of pixels along the line projections. Luminance discontinuities caused by the edges are orthogonal to the projection line so we can define “radial variance vector, $R[\theta]$” as follow.

**Def. 2.4:** Let $\Gamma(\theta)$ denotes the set of pixels $(x, y)$ on the projection line corresponding angle $\theta$ and let $I(x, y)$ denote the luminance value of the pixel $(x, y)$, the radial variance vector $R[\theta]$ where $\theta = 0, 1, \ldots, 179$ is then defined by:

$$R[\theta] = \frac{\sum_{(x,y) \in \Gamma(\theta)} I^2(x, y)}{\# \Gamma(\theta)} - \left( \frac{\sum_{(x,y) \in \Gamma(\theta)} I(x, y)}{\# \Gamma(\theta)} \right)^2$$

Since radon transform is symmetric it is sufficient to extract 180 points instead of 360 and by application of DCT to radial variance vector result will be improved.

Block Mean Value Based Hash Function
In 2006, Yang and colleagues devised a block mean value based perceptual image hash function in four variations with slight difference. The first method is described below as indicated by Zauner [2] since this method is used in our work.

**Method 1**

Normalize the original image into a preset size;

Divide the size-normalized image $I$ into non-overlapped blocks $I_1, I_2, \ldots, I_N$, in which $N$ is the block number equal to length of the final hash bit string;

Encrypt the indices of the block sequence $\{I_1, I_2, \ldots, I_N\}$ using a secret key $K$ to obtain a block sequence with a new scanning order $\{I'_1, I'_2, \ldots, I'_N\}$;

Calculate the mean value sequence $\{M_1, M_2, \ldots, M_N\}$ from corresponding block sequence $\{I'_1, I'_2, \ldots, I'_N\}$ and obtain the median value $M_d$ of this sequence as

$$M_d = \text{median}(M_i), \quad i = 1, 2, \ldots, N$$

Normalize the mean value sequence into a binary form and obtain the hash values $h$ as:

$$h(i) = \begin{cases} 
0, & M_i < M_d \\
1, & M_i \geq M_d
\end{cases}$$

**Methods 2, 3, 4**

In the second variation for increasing the robustness 50% overlapping is used between segments and in the third variation for robustness against any flipping attack a rotation is added to the first method. The fourth method is a combination of method two and three with 50% overlap between blocks and rotation and it is of a great concern that rotation increases the complexity of algorithm [2].

### 2.6. Comparison Perceptual Hash Functions

Since robust hashing is an approved method for analysis of image sets due to their low false alarm rates and reasonable computational complexity their evaluation and validation is desired for their integration in forensic analysis tools [4]. For this purpose Rihmark is used as a benchmarking framework for perceptual image hash functions in terms of speed, inter score distribution (discriminative capabilities) and intra score distribution (robustness) [3]. Several hash functions performance is evaluated by this benchmarking tool namely:
In terms of speed, Block mean value method is fastest and DCT is the slowest by far. Although Marr-Hildreth is the most discriminative method and DCT the second most, test values can be improved by combining different functions.

From robustness point of view several image manipulation techniques are considered:

- Horizontal flipping
- Resizing
- Jpeg Compression
- Rotation

According to the results none of the methods may be considered robust in case of horizontal flipping and against resizing radial variance based method shows poor performance. In case of resizing all methods except Marr-Hildreth give satisfactory results and for rotation block mean value based method is the best.

It is a matter of application that which method may be chosen since it depends on which characteristic is desirable although block mean based perceptual hash function is the fastest and it is either most robust or approximately equal with other functions [3].

Steinbach and colleagues [4] have evaluated an optimized block based hash function which is optimized by segmentation of hash to four sub areas. The segments’ mean value is used for decision and introducing automatic mirroring of the image during the hash calculation in such a manner that darkest part of image is on upper left. This leads to resilience of algorithm to any type of mirroring. Also they have used a weighted distance in addition to hamming distance in order to thresholding and decision making. This method may be a suitable replacement for alternative cryptographic hashes [4].

In this thesis work, block mean value based hash function is used and it is proper to have a detailed review of this function.

### 2.7. Survey of related works

By reviewing several works which has used different approaches for perceptual hashing, it can be summarized in four different categories:
Statistic approach

The calculation of the parameters such as mean, variance and the intensity of the images blocks were obtained by using the statistic computation. The results of such a statistic approach must have good properties by small perturbations of the images. The main disadvantage of this method is easy to modify an image without the change of the intensity of its histogram. It caused the weakness security problems of any scheme which comply upon the intensity statistics. Venkatesan et al. [1] develop an image hash based on an image statistics vector extracted from the various sub-bands in a wavelet decomposition of the image. This paper is based on the observing the statistic such as averages and variance of the sub-bands would be remain stable under a significant modification of the content preserving to the image[16,17,18].

Relation based approach

Lin and Chang [19] proposed a typical relation-based method to image authenticate which is relies on JPEG compression. The digital signature extracted by using the invariant relation between each two discrete cosine transform (DCT) coefficients, which has the same position of the two different 8x8 blocks. The result shows that the invariance properties would preserved before and after JPEG compression during the perceptually lossless. This method is also robust against of JPEG compression, and it would remain vulnerable to various other trivial perceptually modifications. (It must be considered that the nature distortion of statistical is different with the blur which caused by compression).

Lately, Lu and Liao [20] proposed a “structural digital signature” by observing the sub-band wavelet decomposition which parent and the child node are uncorrelated but they are statistically dependent. The result of their observation shows that for the several content preserving manipulations the difference of the magnitude of wavelet coefficients at their consecutive scales remain largely preserved. (i.e., a parent node and their four child nodes)

It produced such a robust digital signature while the identifying the parent-child pairs and subsequently encoding the pairs form. The achievement of the method [20], however is very sensitive to global insignificant rotation and furthermore local geometric attacks.
The coarse image represents

This method proposed a robust hash based on preserving selected (low frequency) discrete cosine Transform (DCT) coefficients [21]. The method of Fridrich and Goljan based on the observing the significant modification to the low-frequency DCT coefficients of the image made change the appearance of the image dramatically.

Mihcak and Venkatesan [22] proposed another image hashing algorithm which used an iterative approach to make binary the DC sub-band (lowest resolution wavelet coefficients) in a wavelet decomposition of the image.

Swaminathan et al. propose an image hash [23] based on selecting rotation-invariant Fourier–Mellin coefficients.

Although their approach was down well under large amounts of global rotation, meantime the robustness of Fourier- Mellin coefficients must be desired under many other classical signal processing distortions.

Image feature extraction approach

The hypothesis of this method is based on the use of vision based feature points for perceptual image hashing [24]-[26]. The aim of the robustness of the schemes in [24] and [25] is far as unsatisfactory for robustness of application. Although the corner-based image features in [26] is robust comply under a large class of attacks, beside an expensive search is necessary to handle the geometric manipulations.

The results obtained from the above methods shows that a common weakness of the methods [16]-[26] is poor robustness against of the geometric attacks, particularly the most wasted ones is as cropping.

Recently Kozat et al. [27] proposed using low-rank matrix approximations obtained via the well-known singular value decomposition (SVD) for image hashing. Since the SVD-based hashing scheme in [27] exhibits good geometric attack robustness, which this method described as such a different images mapping to the same hash value.

Lately, Lee et al. proposed non-negative matrix factorization (NMF) [28] and Vishal Monga and M. Kvanç Mihçak[14] developed the robust image hashing algorithms based on recently proposed dimensionality reduction technique by using the NMF which is distinguished from the traditional matrix approximation method such as QR and SVD, while it used the non-negativity constraints. They proposed [14] that the geometric distortions on images result in approximately additive and independent, identically distributed noise on NMF vectors, Mean time they exploit
this mentioned purpose to obtain pseudorandom linear statistics of NMF vectors which is significantly enhanced hash robustness since the hash allowed to use the small length.

2.8. Metrics

As mentioned earlier there should be measures for comparison of hash values. The most common functions which are used for similarity/distance scoring are hamming distance, BER and PCC. In this sequel the measure may indicate similarity between two strings as it is the case for PCC or may specify distance between them such as hamming distance.

Hamming Distance

Hamming distance may be defined as below:

Def. 2.2: Let $A$ denote an alphabet of finite length. $x = (x_1, \ldots, x_n)$ denotes an even length-string, whereas $x \in A$. The same holds true for $y = (y_1, \ldots, y_n)$. Then the hamming distance $\Delta$ between $x$ and $y$ is defined as:

$$\Delta(x, y) = \sum_{x_i \neq y_i} 1, \ i = 1, \ldots, n.$$ 

Normalized Hamming Distance

For comparison different hamming distances regardless of their correspondent string length normalized hamming distance may be defined as below:

Def. 2.3: Hamming distance can be normalized by with respect to length $n$ of string as:

$$\Delta_n(x, y) = \frac{1}{n} \sum_{x_i \neq y_i} 1, \ i = 1, \ldots, n.$$ 

Calculation of hamming distance for binary strings is possible with XOR operation. Two examples for hamming distance is provided in table 1 for binary and Latin alphabet.
Table 1: Hamming distance examples

<table>
<thead>
<tr>
<th>String 1</th>
<th>String 2</th>
<th>Hamming Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>11011001</td>
<td>01010101</td>
<td>3</td>
</tr>
<tr>
<td>fake</td>
<td>Take</td>
<td>1</td>
</tr>
</tbody>
</table>

Equality Percentage (EP)

Equality percentage may be computed as:

\[
EP = 100 \times \Delta_n
\]

EP has a range between 0 and 100 percent. Higher value indicates more similarity and lower value means more distinction between two perceptual hash values.

Bit Error Rate (BER)

**Def. 2.4:** \( \rho \) is BER and defines as number of \( i \) bit errors of the perceptual hash normalized by length of the perceptual hash value \( k \).

\[
\rho = \frac{i}{k}, \quad i \in \{0, 1, \ldots, k\}
\]

\( i \) is equal to hamming distance of perceptual hash values and obviously \( 0 \leq \rho \leq 1 \). Lower \( \rho \) yields perceptually similar images with a minimum of \( 0 \) which indicates similar hashes.

Peak of Cross Correlation (PCC)

Cross correlation is a measure of similarity between two sequences i.e. higher cross correlation indicates more similarity between two discrete sequences. Definition of cross correlation for two discrete signals is presented below:

**Def. 2.5:** For two discrete finite, with length of \( N \), sequences \( x(n) \) and \( y(n) \) the cross correlation is defined as:
\[ r_{xy}(k) = \sum_{n} x(n)y(n + k) , \quad n = 0,1,\ldots,N - 1 \]

\( x(n) \) and \( y(n) \) are deterministic real valued sequences and \( k \) is the time shift.

Cross correlation can be normalized with respect to mean values of \( x(n) \) and \( y(n) \) which are denoted as \( m_x \) and \( m_y \).

**Def. 2.6:** For two discrete finite, with length of \( N \), sequences \( x(n) \) and \( y(n) \) the normalized cross correlation is defined as:

\[
\frac{\sum_{n}[x(n) - m_x][y(n + k) - m_y]}{\sqrt{\sum_{n}[x(n) - m_x]^2\sum_{n}[y(n) - m_y]^2}} , \quad n = 0,1,\ldots,N - 1
\]

PCC is the maximum value of cross correlation between these two sequences.

### 2.9. False Negative/False Positive

As briefly discussed in introduction this work’s intention is to raise false positive errors for evaluation purposes. According to Sheskin [5]:

In statistics, a type I error (or error of the first kind) is the incorrect rejection of a true null hypothesis. A type II error (or error of the second kind) is the failure to reject a false null hypothesis. A type I error is a false positive and a type II error is a false negative. This means false positive occurs when an unknown object is identified as known one and false negative occurs when a known object is identified as unknown.

### 2.10. Image hash Spoofing

As mentioned before perceptual hash algorithms offer certain degree of robustness by extracting some perceptual features from multimedia object. They are different from generic hashing algorithms in two important points. First they are sensitive only to significant content modification and tolerant to medium level of content preserving image processing techniques. Second difference lies in the fact that they usually utilize a secret key for hashing process in order to increase resilience to malicious manipulations and the length of the secret key
determines the degree of protection without any compression. The latter case lies in the fact that algorithms are publicly known and using them solely is not secure.

According to Li Weng and Perneel two types of attack is possible [9].

- Counterfeiting both hash and content
- Gradually introduction of changes until the content is severely distorted

As suggested by the authors incorporation of a secret key in order to generate hash key protects perceptual hash algorithms against malicious manipulations. The main drawback is that it would not be easy to establish information protection protocol easily. Another problem is that the hashing process would not be efficient.

According to Wikipedia “in the context of network security, a spoofing attack is a situation in which one person or program successfully masquerades as another by falsifying data and thereby gaining an illegitimate advantage”. Since here our objective is to develop a method of producing an image hash which closely resembles another image hash code, based on robust image hashing, for evaluation false positive alarms, our developing algorithm is named image hash spoofing. This algorithm must be robust in the sense of showing resilience to different image manipulation techniques such as cropping, compression and scaling. Since our approach consists of modifications to an image in order to imitate another image which causes negative false alarms the terms spoofing and attacking is used interchangeably.

3. Implementation

3.1. Requirements

As previously mentioned our goal is to develop a mechanism that an image masquerades another image and this process must be done automatic and robust in the sense of that it has no limitation against any normal image processing techniques. In order to achieve this, MATLAB® is used as a powerful interactive programming tool for implementation. All images are of 256×256 size in JPEG format and hash key length of 256 bits. As a measure of similarity between hash keys, computed by block mean based algorithm, hamming distance is used.
3.2. Preliminary Work

As first step a database of approximately 150,000 hashed images is analyzed and the task was choosing 30 random images and comparing each to the all other pictures. This has been done in MATLAB by uniformly random selection of 30 images and calculation of hamming distance in order to compare with the entire of database. Regarding the threshold of 16 is imposed on hamming distances and the images with hamming distance below the threshold is considered similar. The result is shown in table 2 and a total of 10 collisions have occurred according the thresholding criterion.

In order to have a better understanding, histogram of hamming distances for selection number 14 has been shown in figure 2 and a magnified version for distances under threshold in figure 3. It must be noticed that corresponding normal distribution have been superimposed on the histogram. In figure 4, boxplot of hamming distances for selected all images has been depicted. In the statistical approach it is a graphical way to represent a set of data through quartiles as shown in figure 1 provided by www.wellbeingatschool.org. In figure 5, boxplot of selected image No.14 is shown.

<table>
<thead>
<tr>
<th>Image No</th>
<th>Report</th>
</tr>
</thead>
<tbody>
<tr>
<td>121323</td>
<td>Number of similar images for selection No 1 is: 1</td>
</tr>
<tr>
<td>134885</td>
<td>Number of similar images for selection No 2 is: 1</td>
</tr>
<tr>
<td>18910</td>
<td>Number of similar images for selection No 3 is: 0</td>
</tr>
<tr>
<td>136014</td>
<td>Number of similar images for selection No 4 is: 3</td>
</tr>
<tr>
<td>94167</td>
<td>Number of similar images for selection No 5 is: 0</td>
</tr>
<tr>
<td>14526</td>
<td>Number of similar images for selection No 6 is: 0</td>
</tr>
<tr>
<td>Image ID</td>
<td>Number of similar images for selection No 7 is: 1</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>41473</td>
<td>Number of similar images for selection No 8 is: 0</td>
</tr>
<tr>
<td>81438</td>
<td>Number of similar images for selection No 9 is: 0</td>
</tr>
<tr>
<td>142586</td>
<td>Number of similar images for selection No 10 is: 0</td>
</tr>
<tr>
<td>143685</td>
<td>Number of similar images for selection No 11 is: 0</td>
</tr>
<tr>
<td>23471</td>
<td>Number of similar images for selection No 12 is: 0</td>
</tr>
<tr>
<td>144534</td>
<td>Number of similar images for selection No 13 is: 1</td>
</tr>
<tr>
<td>142535</td>
<td>Number of similar images for selection No 14 is: 2</td>
</tr>
<tr>
<td>72279</td>
<td>Number of similar images for selection No 15 is: 0</td>
</tr>
<tr>
<td>119173</td>
<td>Number of similar images for selection No 16 is: 0</td>
</tr>
<tr>
<td>21129</td>
<td>Number of similar images for selection No 17 is: 0</td>
</tr>
<tr>
<td>62806</td>
<td>Number of similar images for selection No 18 is: 0</td>
</tr>
<tr>
<td>136365</td>
<td>Number of similar images for selection No 19 is: 0</td>
</tr>
<tr>
<td>117970</td>
<td>Number of similar images for selection No 20 is: 0</td>
</tr>
<tr>
<td>142881</td>
<td>Number of similar images for selection No 21 is: 0</td>
</tr>
<tr>
<td>97649</td>
<td>Number of similar images for selection No 22 is: 0</td>
</tr>
<tr>
<td>5318</td>
<td>Number of similar images for selection No 23 is: 0</td>
</tr>
<tr>
<td>126447</td>
<td>Number of similar images for selection No 24 is: 1</td>
</tr>
<tr>
<td>139084</td>
<td>Number of similar images for selection No 25 is: 0</td>
</tr>
<tr>
<td>101073</td>
<td>Number of similar images for selection No 26 is: 0</td>
</tr>
<tr>
<td>112838</td>
<td>Number of similar images for selection No 27 is: 0</td>
</tr>
<tr>
<td>110663</td>
<td>Number of similar images for selection No 28 is: 0</td>
</tr>
<tr>
<td>58408</td>
<td>Number of similar images for selection No 29 is: 0</td>
</tr>
<tr>
<td>25492</td>
<td>Number of similar images for selection No 30 is: 0</td>
</tr>
</tbody>
</table>

Table 2: 30 selected images hamming distance to other images
Figure 2: Hamming distance histogram for selection No.14. Yellow line is the threshold imposed.

Figure 3: Hamming distance histogram in selection No.14 for values below the threshold.
Figure 4: Hamming distance boxplot for all selections

Figure 5: Hamming distance boxplot for selection No.14
After extracting the original random images from the database and checking them visually it is clear that only one true collision has been occurred which indicates a negative false alarm. The other collisions were the same image with different sizes. Collided images have not been shown because the pornographic nature of images.

### 3.3. Design

In this section we are going to describe how an image, let’s call it an attacked image, masquerades another image that to be called favorite image. This spoofing attack must be done in a way that loss of perceptual quality of attacked image would not be significant. In order to achieve this, black mean based hash will be used to produce a similar hash bit to favorite image. In general the whole process can be divided in two stages:

#### 3.3.1. Stage 1:

All image are 256×256 pixels and gray scale and if not they will be resized and converted to 256×256. Then hash key attacked image (figure 6) is calculated by block mean based method as below:

- First Favorite image is segmented to 16×16 grid as figure 7.

![Figure 6: Target image](image1)

![Figure 7: Segmented target image](image2)

- Now mean value of each block is calculated. $M_i$, $i = 1, ..., 256$

- Median of $M_i$s is calculated so, $M_d$ is median of $M_i$s.

- Now using following criteria the hash key for the image is calculated as below:

$$h(i) = \begin{cases} 0, & M_i < M_d \\ 1, & M_i \geq M_d \end{cases}$$
- Now we convert this hash key to an image mask as shown in figure 8.

![Figure 8: Target image hash](image)

- All the above steps will be done on attacked image too as shown in figures 9 and 10.

![Figure 9: Attacked image](image) ![Figure 10: Attacked image hash](image)

- As first modification on attacked image we use mask which is obtained from favorite image to manipulate intensities of attacked image according to favorite mask. This is done by multiplication of pixel intensities on attacked image to numbers higher than 1 where mask is white and lower than 1 in locations that mask is black. It has to be mentioned that hash key is 16×16 so before this operation it must be resized to 256×256. After this operation since intensities have changed then picture will appear unnatural as it can be seen in figures 11 and 12 along its new hash key then some compensation is needed.
Now by use of Gaussian filter the FAVORITE image’s mask will be blurred to overcome this problem but higher degree of blurring means more hamming distance at the end so we have a tradeoff here. In our implementation degree of blurring will be controlled by BD parameter. After blurring the attacked image mask and its corresponding masked image is shown at figures 13 and 14.

Now the first stage has finished and in this stage we will get lower hamming distance. Further operation in order to obtain hamming distance will be done in the next stage. Here another option is utilized which is instead of using extracted mask from target image, it is possible to use predefined masks although the resulting hamming distance is not as good as the extracted mask method. Some of these masks are depicted in figure 15. Software searches for nearest mask to target image’s mask in terms of hamming distance and use it for manipulating attacked image.
- There is also an option that we can define the minimum hamming distance required to enter the stage one i.e. if an image’s hash key has a hamming distance below this threshold to target image software directly goes to stage two and no operation is done in terms of intensity manipulation and filtering.
3.3.2. **Stage2:**

In this stage the produced attacked image from the previous stage is subjected to changing intensities in those particular blocks that their mean value is different from mean value of target (favorite) image. This process is done by adding 1, 2, … to intensities of those blocks that their mean value will exceed the median if they are lower than median or fell below the median if they are above the median. There are two limitations for these operations i.e.:

- Changing is done in a way that median remains unchanged. It means if a changing of value for a particular block changes the median then another change is done in the reverse direction in order to reach a target value in hamming distance.

- Variance of each block is used as a measure of quality which limits number of change for each block change. This works as a feature in the software i.e. if keeping the quality of the attacked image is important for us then regardless of what is the target hamming distance is, a particular block is subjected to predefined number of intensity changes according the value of the block variance as shown in table 4. These values have obtained by testing on many images. The reason behind is higher the number of variance, changes in the block mean value has lesser effect in human perception.

Values in table 4 have been obtained by several runs of the program. For more information refer to appendix A.

<table>
<thead>
<tr>
<th>Block Variance</th>
<th>Maximum allowed distance to change</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-100</td>
<td>2</td>
</tr>
<tr>
<td>100-1000</td>
<td>4</td>
</tr>
<tr>
<td>1000-20000</td>
<td>5</td>
</tr>
<tr>
<td>20000-30000</td>
<td>8</td>
</tr>
<tr>
<td>30000-2500000</td>
<td>15 (varies from 10 to 20)</td>
</tr>
<tr>
<td>≥2500000</td>
<td>24</td>
</tr>
</tbody>
</table>

*Table 3: Variance criterion for quality control*
3.4. MATLAB® Code

In this Master thesis entire program has been written in MATLAB®. About the code there are some important notes that must be noticed. First is that in the code the burden of the job is on two functions which constructs core of the software namely Stage1M.m and Stage2M.m. In Stage1M.m software calculates the hash bits for original and target image and it uses some input parameters such as HammingDistanceLimit1, BD, wh, bl and method. HammingDistanceLimit1 is threshold for attacked image i.e. if hamming distance of attacked image fell below this value then stage 1 is discarded and attacked image after computation of hash bits directly will be send to stage 2. BD is controlling parameter for blurring and higher blurring means higher final hamming distance and a typical value for this parameter in between 20 and 30. wh and bl are the intensity multiplication coefficients for light and dark areas of hash key respectively and value equal to unity means image is remained unchanged and typically for wh value is above one and for the bl is below one. method defines whether target image or predefined masks is used for masking the image and value equal to 1 means that target image will be used for masking the attacked image and method equal to 2 means software will use one of predefined masks with lowest hamming distance to target image.

There are other functions that are used by mentioned main functions as below:

- HashDist.m:
  Calculates hamming distance for two images’ hash codes.

- ChangeMean.m:
  Changes the mean value of the desired block in positive or negative direction.
function [I1,H2,Dist2,Dist1]=Stage1M(OriginalImageName,cond,BD,wh,bl,method,ShRe,SaRe)
%Input Arguments:
%OriginalImageName: Attacked Image name
%cond: Hamming distance limit1
%BD: Blurring Degree (Gaussian LPF)
%wh: White area multiplication coefficient
%bl: Black area multiplication coefficient
%method: 1 (Uses target image hash key)-2 (Uses predefined hash key)
%ShRe: Show result (1) Dont show (0)
%SaRe: Save Result (1) Dont Save (0)
%Output Arguments:
%I1: Modified attack image after stage1
%H2: Hash bit of target image
%Dist2: Hamming Distance after modification
%Dist1: Hamming Distance before modification

function [I,Dist1]=Stage2M(I,H0,limit,BEST,name,ShRe,SaRe)
%Input Arguments:
%I: Attacked Image name after stage1
%limit: Hamming distance limit2
%BEST: 'yes'quality is preserved regardless of hamming distance limit
%      'no' hamming distance is reduced till reaching hamming distance
%      limit 2
%name: Target image name
%ShRe: Show result (1) Dont show (0)
%SaRe: Save Result (1) Dont Save (0)
%Output Arguments:
%I: Modified attack image after stage2
%Dist1: Hamming Distance after modification
4. Verification and Results

4.1. Method 1

In this section the implemented spoofing program is tested for a set of 163 images. Here the favorite (target image) has been used in order to obtain the mask. In figures 17-24 and tables 4-7 step by step outcome can be observed for four cases. The target and attacking image is different perceptually and from hash key point of view.

For the first case hash pattern of original image is significantly different from attacking image in the top and bottom of image as it can be observed in figure 17 but the variance of intensities in the attacked image is significantly high and this allows high number of block mean changes. In order to mask attacking image by original image mask its hash mask is smoothed by a Gaussian low pass filter and by using scaling factors for black and white regions the final mask is obtained as titled in figure 17 as final mask of original image by offset. Applying this mask to target image, we reach the end of stage one and by changing the mean value of the selected blocks consequently the overall result can be in figure 18. The visual effect of process can be observed as shadows on the image but there is no artifact and damage to image texture and blurring effect of smoothing filter is not significant. At the same time hamming distance between original image and attacking image is reduced significantly by 103 units at the expense of appearing shadows.

In second and third case again the hash keys of target image and attacking image are significantly different and shadows appear on particular areas that original image has low intensity. The hamming distance reduction is high with 96 and 93 points for second and third example respectively. For the fourth case hash key of target and attacking image is more similar compare to previous examples and original image hash has a dark pattern approximately in all areas. As it is expected the final image appears a darker version of the image before attack with hamming distance reduction of 77.

The dark regions on the final regions may be explained by high scaling factors for black and white region which are respectively are chosen 2 and 0.5 which means in the attacking image is intensities according to target image hash key are two times brighter in white regions and one half in black regions. These values are chosen to achieve high hamming distance reduction with a target of 16. For pictures with similar objects such as face the result would be better in terms of intensity distortions.

In order to have a better understanding in figures 25 and 26 histogram of hamming distance reduction after first, second and first plus second stages as well as their boxplot is provided. Histogram and boxplot graphs indicate that most of hamming distance reduction occurs at the first stage with mean of 74.36 whereas in stage 2 is only about 7 degree reduction. Maximum reduction of hamming distance in stage 1 is 106 while in stage 2 is equal to 33. Total hamming distance reduction based on block mean value has a mean of 81.41, maximum value of 106 and
median equal to 85. This significant reduction in hamming distance can be described by choosing the proper attack image set. Images with higher block variance, in other words with many small objects, is a proper choice in order to be chosen as attacking image. As it can be observed in provided figures, images perceptual quality in terms of texture has been preserved. In order to have a measure of contaminated noise in the images peak signal to noise ratio (PSNR) have been calculated and its respective histogram is shown in figure 16.

![Figure 16: PSNR histogram for 163 attacked image](image-url)
Figure 17: Method 1-image 1
In the program following quantities have been used:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HammingDistanceLimit1</td>
<td>20</td>
<td>Limit for stage 1 and below this value SW discards stage 1</td>
</tr>
<tr>
<td>HammingDistanceLimit2</td>
<td>16</td>
<td>Target Hamming Distance</td>
</tr>
<tr>
<td>BD</td>
<td>20</td>
<td>Blurring Coefficient</td>
</tr>
<tr>
<td>Wh</td>
<td>2</td>
<td>Intensity Multiplication Coefficient in bright parts of the mask</td>
</tr>
<tr>
<td>Bl</td>
<td>0.5</td>
<td>Intensity Multiplication Coefficient in dark parts of the mask</td>
</tr>
<tr>
<td>Quality</td>
<td>on</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Hamming distance reduction for Method 1- Image 1

<table>
<thead>
<tr>
<th>Hamming Distance Before Attack</th>
<th>Hamming Distance After First Stage</th>
<th>Hamming Distance After Second Stage</th>
</tr>
</thead>
<tbody>
<tr>
<td>113</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

Figure 18: Method 1- Image 1
Figure 19: Method 1- Image 2
Table 5: Hamming distance reduction for Method 1 - Image 2

<table>
<thead>
<tr>
<th>Hamming Distance Before Attack</th>
<th>Hamming Distance After First Stage</th>
<th>Hamming Distance After Second Stage</th>
</tr>
</thead>
<tbody>
<tr>
<td>113</td>
<td>40</td>
<td>17</td>
</tr>
</tbody>
</table>

In the program following quantities have been used:

- HammingDistanceLimit1 = 20 (Limit for stage 1 and below this value SW discards stage 1)
- HammingDistanceLimit2 = 16 (Target Hamming Distance)
- BD = 20 (Blurring Coefficient)
- Wh = 2 (Intensity Multiplication Coefficient in bright parts of the mask)
- Bl = 0.5 (Intensity Multiplication Coefficient in dark parts of the mask)
- Quality = on
Figure 21: Method 1 – Image 3
Figure 22: Method 1 – Image 3

<table>
<thead>
<tr>
<th>Hamming Distance</th>
<th>Hamming Distance After First Stage</th>
<th>Hamming Distance After Second Stage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before Attack</td>
<td></td>
<td></td>
</tr>
<tr>
<td>109</td>
<td>33</td>
<td>16</td>
</tr>
</tbody>
</table>

Table 6: Hamming distance reduction for Method 1- Image 3

In the program following quantities have been used:
- HammingDistanceLimit1 = 20 (Limit for stage 1 and below this value SW discards stage 1)
- HammingDistanceLimit2 = 16 (Target Hamming Distance)
- BD = 20 (Blurring Coefficient)
- Wh = 2 (Intensity Multiplication Coefficient in bright parts of the mask)
- Bl = 0.5 (Intensity Multiplication Coefficient in dark parts of the mask)
- Quality = on
Figure 23: Method 1 – Image 4
Table 7: Hamming distance reduction for Method 1 - Image 4

<table>
<thead>
<tr>
<th>Hamming Distance Before Attack</th>
<th>Hamming Distance After First Stage</th>
<th>Hamming Distance After Second Stage</th>
</tr>
</thead>
<tbody>
<tr>
<td>93</td>
<td>33</td>
<td>16</td>
</tr>
</tbody>
</table>

In the program following quantities have been used:

- HammingDistanceLimit1 = 20 (Limit for stage 1 and below this value SW discards stage 1)
- HammingDistanceLimit2 = 16 (Target Hamming Distance)
- BD = 20 (Blurring Coefficient)
- Wh = 2 (Intensity Multiplication Coefficient in bright parts of the mask)
- Bl = 0.5 (Intensity Multiplication Coefficient in dark parts of the mask)
- Quality = on
Figure 25: Histogram of hamming distance reduction after stage1 (up), stage2 (middle) and stages 1+2 (bottom)
4.2. Method 2

Here the program implemented based in predefined masks and is tested on a set of 50 images. Its operation on three of the attacked images has been depicted on in pictures 27 to 31 and tables 8 to 10. In figures 32 and 33 histogram of hamming distance reduction for stage one, stage two, stage one plus stage two as well as corresponding boxplot is depicted. Here hamming distance reduction is declined drastically. For instance at stage one and stage two mean values of hamming distance reduction are 30.14 and 23.24 respectively. Mean of total hamming distance reduction declines from 81 to 53 compared to first method.
Figure 27: Method 2 – Image 1
Table 8: Hamming distance reduction for Method 2 - Image 1

<table>
<thead>
<tr>
<th>Hamming Distance Before Attack</th>
<th>Hamming Distance After First Stage</th>
<th>Hamming Distance After Second Stage</th>
</tr>
</thead>
<tbody>
<tr>
<td>113</td>
<td>63</td>
<td>41</td>
</tr>
</tbody>
</table>

In the program following quantities have been used:

- $\text{HammingDistanceLimit1} = 20$ (Limit for stage 1 and below this value SW discards stage 1)
- $\text{HammingDistanceLimit2} = 16$ (Target Hamming Distance)
- $\text{BD} = 20$ (Blurring Coefficient)
- $\text{Wh} = 2$ (Intensity Multiplication Coefficient in bright parts of the mask)
- $\text{Bl} = 0.5$ (Intensity Multiplication Coefficient in dark parts of the mask)
- Quality = on
Figure 29: Method 2 – Image 2
Figure 30: Method 2 – Image 2

<table>
<thead>
<tr>
<th>Hamming Distance Before Attack</th>
<th>Hamming Distance After First Stage</th>
<th>Hamming Distance After Second Stage</th>
</tr>
</thead>
<tbody>
<tr>
<td>113</td>
<td>96</td>
<td>60</td>
</tr>
</tbody>
</table>

Table 9: Hamming distance reduction for Method 2 - Image 2

In the program following quantities have been used:
- HammingDistanceLimit1 = 20   (Limit for stage 1 and below this value SW discards stage 1)
- HammingDistanceLimit2 = 16   (Target Hamming Distance)
- BD = 20                     (Blurring Coefficient)
- Wh = 2                      (Intensity Multiplication Coefficient in bright parts of the mask)
- Bl = 0.5                    (Intensity Multiplication Coefficient in dark parts of the mask)
- Quality = on
Figure 31: Method 2 – Image 3
Hamming Distance Before Attack | Hamming Distance After First Stage | Hamming Distance After Second Stage
--- | --- | ---
109 | 92 | 52

Table 10: Hamming distance reduction for Method 2 - Image 3

In the program following quantities have been used:

- \( \text{HammingDistanceLimit1} = 20 \) (Limit for stage 1 and below this value SW discards stage 1)
- \( \text{HammingDistanceLimit2} = 16 \) (Target Hamming Distance)
- \( \text{BD} = 20 \) (Blurring Coefficient)
- \( \text{Wh} = 2 \) (Intensity Multiplication Coefficient in bright parts of the mask)
- \( \text{Bl} = 0.5 \) (Intensity Multiplication Coefficient in dark parts of the mask)
- \( \text{Quality} = 0 \)
Figure 33: Histogram of hamming distance reduction after stage 1 (up), stage 2 (middle) and stages 1+2 (bottom)
Figure 34: Boxplot of hamming distance reduction
5. Conclusion and Future Work

5.1 Conclusion

This thesis work is undertaken to design and implement a novel method for imitating hash key of a target image by another image. This hash spoofing method implemented based on block mean value hash algorithm by MATLAB and verified. Prior to local modification of specific blocks mean value in the image, a global modification of image is achieved by use of a picture mask and weighting intensities of the image to reach a lower hamming distance. This increase in the similarity of images' hashes had to be done without any visual quality loss of image. For this purpose a thresholding criterion for maximum modifiable distance to mean is obtained by any runs on the program. This thresholding criterion is based on the variance of each block and higher variance means more allowable block's mean modification. Testing the software with a set of images proved that this method of hash spoofing is more effective if target image’s hash key is used as an intensity modification mask with a mean of hamming distance reduction equal to 81 while with predefined mask method it is 51. In overall this methods shows acceptable performance especially with the first approach. Since this algorithm uses simplest block mean value based algorithms its cost and complexity is relatively low.

5.2 Future Work

As proposed by Yang, Gu and Niu instead of simple block mean value based algorithm other variations with 50% overlap and rotation may be used. As discussed in [2], it is anticipated better reduction in hamming distance at the expense of complexity and computational cost. For second method better set of predefined masks may lead to improve in hamming distance reduction. Also it is anticipated that in order to improve the algorithm in stage 2, more precise in variance thresholding may lead to better performance of the algorithm.
References:


Appendices:

Appendix A: Variance Thresholding

<table>
<thead>
<tr>
<th>Name</th>
<th>Mean</th>
<th>Variance</th>
<th>Block number</th>
<th>Mean of Block</th>
<th>Variance of Block</th>
<th>Max Pass intensity for +</th>
<th>Max Pass intensity for -</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>120.69</td>
<td>4.9053e+005</td>
<td>26</td>
<td>122.0469</td>
<td>1.4942e+006</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>97</td>
<td>196.8867</td>
<td></td>
<td>2.6184e+005</td>
<td></td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>140.2500</td>
<td></td>
<td>2.3523e+006</td>
<td></td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>170</td>
<td>202.1367</td>
<td></td>
<td>18.9699</td>
<td></td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>87</td>
<td>53.4453</td>
<td></td>
<td>1.8760e+006</td>
<td></td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>34</td>
<td>124.9883</td>
<td></td>
<td>6.9623e+005</td>
<td></td>
<td>10</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>120</td>
<td>146.6328</td>
<td></td>
<td>1.2084e+006</td>
<td></td>
<td>15</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>59</td>
<td>100.5195</td>
<td></td>
<td>1.0453e+006</td>
<td></td>
<td>30</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>193</td>
<td>129.1523</td>
<td></td>
<td>1.4958e+006</td>
<td></td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>148</td>
<td>72.0508</td>
<td></td>
<td>4.2283e+005</td>
<td></td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>201</td>
<td>163.1406</td>
<td></td>
<td>2.0434e+006</td>
<td></td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>108.5977</td>
<td></td>
<td>2.1323e+006</td>
<td></td>
<td>15</td>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Mean</th>
<th>Variance</th>
<th>Block number</th>
<th>Mean of Block</th>
<th>Variance of Block</th>
<th>Max Pass intensity for +</th>
<th>Max Pass intensity for -</th>
</tr>
</thead>
<tbody>
<tr>
<td>desert</td>
<td>129.8907</td>
<td>4.1868e+004</td>
<td>16</td>
<td>144.7070</td>
<td>7.5226</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>45</td>
<td>99.6367</td>
<td></td>
<td>3.5061</td>
<td></td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>115.5977</td>
<td></td>
<td>0.4933</td>
<td></td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>38</td>
<td>115.3828</td>
<td></td>
<td>220.3143</td>
<td></td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>54</td>
<td>112.3008</td>
<td></td>
<td>1.4261e+003</td>
<td></td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>69</td>
<td>101.5586</td>
<td></td>
<td>0.3592</td>
<td></td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>78</td>
<td>139.2734</td>
<td></td>
<td>3.3485e+003</td>
<td></td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>135</td>
<td>130.7813</td>
<td></td>
<td>626.4725</td>
<td></td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>166</td>
<td>128.3008</td>
<td></td>
<td>9.6057e+003</td>
<td></td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>183</td>
<td>135.7695</td>
<td></td>
<td>7.9358e+003</td>
<td></td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>194</td>
<td>155.9453</td>
<td></td>
<td>6.1390e+003</td>
<td></td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>245</td>
<td>120.0547</td>
<td></td>
<td>2.5116e+004</td>
<td></td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>255</td>
<td>139.8750</td>
<td></td>
<td>0.1422</td>
<td></td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Name</td>
<td>Mean</td>
<td>Variance</td>
<td>Block number</td>
<td>Mean of Block</td>
<td>Variance of Block</td>
<td>Max Pass intensity for +</td>
<td>Max Pass intensity for -</td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
<td>------------------</td>
<td>--------------</td>
<td>---------------</td>
<td>---------------------</td>
<td>--------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>river (6)</td>
<td>120.0362</td>
<td>3.5216e+005</td>
<td>28</td>
<td>140.3086</td>
<td>3.7625e+006</td>
<td>26</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>112.2695</td>
<td>6.4191e+005</td>
<td>39</td>
<td>114.3086</td>
<td>6.4191e+005</td>
<td>24</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>99.8203</td>
<td>2.1623e+006</td>
<td>56</td>
<td>109.3086</td>
<td>2.1623e+006</td>
<td>25</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>181.4609</td>
<td>1.5694e+005</td>
<td>64</td>
<td>167.3086</td>
<td>1.5694e+005</td>
<td>20</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>107.4805</td>
<td>1.0396e+007</td>
<td>79</td>
<td>113.3086</td>
<td>1.0396e+007</td>
<td>27</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>95.3438</td>
<td>6.4626e+006</td>
<td>85</td>
<td>105.3086</td>
<td>6.4626e+006</td>
<td>18</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>104.7344</td>
<td>2.6235e+005</td>
<td>94</td>
<td>111.3086</td>
<td>2.6235e+005</td>
<td>22</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>112.5938</td>
<td>1.0286e+006</td>
<td>134</td>
<td>108.3086</td>
<td>1.0286e+006</td>
<td>20</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>157.6250</td>
<td>4.5382e+006</td>
<td>155</td>
<td>154.3086</td>
<td>4.5382e+006</td>
<td>24</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>111.1172</td>
<td>4.1756e+006</td>
<td>198</td>
<td>112.3086</td>
<td>4.1756e+006</td>
<td>24</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>123.9453</td>
<td>3.2216e+006</td>
<td>206</td>
<td>121.3086</td>
<td>3.2216e+006</td>
<td>28</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>96.3789</td>
<td>3.1393e+006</td>
<td>245</td>
<td>94.3086</td>
<td>3.1393e+006</td>
<td>28</td>
<td>26</td>
</tr>
<tr>
<td>sea (2)</td>
<td>130.1862</td>
<td>2.6821e+005</td>
<td>28</td>
<td>90.1133</td>
<td>5.3403e+003</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>143.2695</td>
<td>4.0912e+004</td>
<td>35</td>
<td>141.3086</td>
<td>4.0912e+004</td>
<td>18</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>91.4414</td>
<td>1.5365e+003</td>
<td>48</td>
<td>89.3086</td>
<td>1.5365e+003</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>135.6875</td>
<td>4.9744e+004</td>
<td>53</td>
<td>133.3086</td>
<td>4.9744e+004</td>
<td>19</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>159.8281</td>
<td>2.0161e+004</td>
<td>66</td>
<td>157.3086</td>
<td>2.0161e+004</td>
<td>15</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>122.4492</td>
<td>3.1277e+004</td>
<td>76</td>
<td>120.3086</td>
<td>3.1277e+004</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>102.9414</td>
<td>419.3829</td>
<td>89</td>
<td>100.3086</td>
<td>419.3829</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>80.8633</td>
<td>564.4349</td>
<td>111</td>
<td>78.3086</td>
<td>564.4349</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>151.2422</td>
<td>5.9005e+004</td>
<td>134</td>
<td>149.3086</td>
<td>5.9005e+004</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>140.1289</td>
<td>5.0154e+003</td>
<td>155</td>
<td>138.3086</td>
<td>5.0154e+003</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>142.4219</td>
<td>2.3915e+004</td>
<td>198</td>
<td>140.3086</td>
<td>2.3915e+004</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>86.1367</td>
<td>1.0651e+004</td>
<td>236</td>
<td>84.3086</td>
<td>1.0651e+004</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>75.3633</td>
<td>35.1890</td>
<td>255</td>
<td>73.3086</td>
<td>35.1890</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>
### Appendix B: Tables of Hamming Distance Reduction by stage

**Table 1:** Results from method 1 using the modified mask from the original image.

<table>
<thead>
<tr>
<th>Name of Image</th>
<th>Original Hamming Distance</th>
<th>1st Stage Hamming Distance</th>
<th>2nd Stage (Final) Hamming Distance</th>
<th>PSNR in dB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>113</td>
<td>40</td>
<td>17</td>
<td>15.6338</td>
</tr>
<tr>
<td>2</td>
<td>113</td>
<td>10</td>
<td>10</td>
<td>14.5839</td>
</tr>
<tr>
<td>3</td>
<td>109</td>
<td>37</td>
<td>16</td>
<td>15.6103</td>
</tr>
<tr>
<td>4</td>
<td>111</td>
<td>27</td>
<td>16</td>
<td>14.8646</td>
</tr>
<tr>
<td>5</td>
<td>92</td>
<td>37</td>
<td>16</td>
<td>17.3132</td>
</tr>
<tr>
<td>6</td>
<td>103</td>
<td>29</td>
<td>16</td>
<td>15.1918</td>
</tr>
<tr>
<td>7</td>
<td>92</td>
<td>11</td>
<td>11</td>
<td>13.944</td>
</tr>
<tr>
<td>8</td>
<td>99</td>
<td>15</td>
<td>15</td>
<td>14.0747</td>
</tr>
<tr>
<td>9</td>
<td>88</td>
<td>7</td>
<td>7</td>
<td>14.029</td>
</tr>
<tr>
<td>10</td>
<td>91</td>
<td>7</td>
<td>7</td>
<td>14.7925</td>
</tr>
<tr>
<td>11</td>
<td>95</td>
<td>10</td>
<td>10</td>
<td>13.9382</td>
</tr>
<tr>
<td>12</td>
<td>86</td>
<td>3</td>
<td>3</td>
<td>13.9965</td>
</tr>
<tr>
<td>13</td>
<td>96</td>
<td>24</td>
<td>16</td>
<td>14.7465</td>
</tr>
<tr>
<td>14</td>
<td>97</td>
<td>33</td>
<td>16</td>
<td>17.1763</td>
</tr>
<tr>
<td>15</td>
<td>93</td>
<td>33</td>
<td>16</td>
<td>17.8165</td>
</tr>
<tr>
<td>16</td>
<td>102</td>
<td>32</td>
<td>16</td>
<td>14.7695</td>
</tr>
<tr>
<td>17</td>
<td>87</td>
<td>6</td>
<td>6</td>
<td>13.816</td>
</tr>
<tr>
<td>18</td>
<td>112</td>
<td>9</td>
<td>9</td>
<td>13.6047</td>
</tr>
<tr>
<td>19</td>
<td>107</td>
<td>10</td>
<td>10</td>
<td>13.7455</td>
</tr>
<tr>
<td>20</td>
<td>105</td>
<td>13</td>
<td>13</td>
<td>13.951</td>
</tr>
<tr>
<td>21</td>
<td>108</td>
<td>25</td>
<td>16</td>
<td>15.1693</td>
</tr>
<tr>
<td>22</td>
<td>102</td>
<td>14</td>
<td>14</td>
<td>14.8231</td>
</tr>
<tr>
<td>23</td>
<td>113</td>
<td>10</td>
<td>10</td>
<td>14.7227</td>
</tr>
<tr>
<td>24</td>
<td>89</td>
<td>10</td>
<td>10</td>
<td>14.1853</td>
</tr>
<tr>
<td>25</td>
<td>109</td>
<td>12</td>
<td>12</td>
<td>13.6674</td>
</tr>
<tr>
<td>26</td>
<td>104</td>
<td>28</td>
<td>16</td>
<td>15.3215</td>
</tr>
<tr>
<td>27</td>
<td>110</td>
<td>4</td>
<td>4</td>
<td>14.4032</td>
</tr>
<tr>
<td>28</td>
<td>106</td>
<td>10</td>
<td>10</td>
<td>15.0094</td>
</tr>
<tr>
<td>29</td>
<td>111</td>
<td>19</td>
<td>16</td>
<td>14.938</td>
</tr>
<tr>
<td>30</td>
<td>96</td>
<td>7</td>
<td>7</td>
<td>14.8179</td>
</tr>
<tr>
<td>31</td>
<td>113</td>
<td>32</td>
<td>16</td>
<td>15.0189</td>
</tr>
<tr>
<td>32</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Inf</td>
</tr>
<tr>
<td>33</td>
<td>110</td>
<td>26</td>
<td>16</td>
<td>17.5909</td>
</tr>
<tr>
<td>34</td>
<td>107</td>
<td>24</td>
<td>16</td>
<td>14.4356</td>
</tr>
<tr>
<td>35</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Inf</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
<td>----</td>
<td>----</td>
<td>-----</td>
</tr>
<tr>
<td>36</td>
<td>106</td>
<td>11</td>
<td>11</td>
<td>15.2288</td>
</tr>
<tr>
<td>37</td>
<td>79</td>
<td>19</td>
<td>16</td>
<td>14.9858</td>
</tr>
<tr>
<td>38</td>
<td>97</td>
<td>10</td>
<td>10</td>
<td>13.8789</td>
</tr>
<tr>
<td>39</td>
<td>98</td>
<td>37</td>
<td>16</td>
<td>14.7288</td>
</tr>
<tr>
<td>40</td>
<td>97</td>
<td>13</td>
<td>13</td>
<td>14.419</td>
</tr>
<tr>
<td>41</td>
<td>118</td>
<td>38</td>
<td>15</td>
<td>14.4965</td>
</tr>
<tr>
<td>42</td>
<td>110</td>
<td>4</td>
<td>4</td>
<td>14.4032</td>
</tr>
<tr>
<td>43</td>
<td>91</td>
<td>14</td>
<td>14</td>
<td>14.2107</td>
</tr>
<tr>
<td>44</td>
<td>116</td>
<td>15</td>
<td>15</td>
<td>14.9863</td>
</tr>
<tr>
<td>45</td>
<td>87</td>
<td>12</td>
<td>12</td>
<td>14.1973</td>
</tr>
<tr>
<td>46</td>
<td>112</td>
<td>49</td>
<td>16</td>
<td>15.7424</td>
</tr>
<tr>
<td>47</td>
<td>99</td>
<td>8</td>
<td>8</td>
<td>14.5423</td>
</tr>
<tr>
<td>48</td>
<td>89</td>
<td>25</td>
<td>16</td>
<td>14.7929</td>
</tr>
<tr>
<td>49</td>
<td>94</td>
<td>9</td>
<td>9</td>
<td>13.8939</td>
</tr>
<tr>
<td>50</td>
<td>72</td>
<td>16</td>
<td>16</td>
<td>14.887</td>
</tr>
<tr>
<td>51</td>
<td>102</td>
<td>33</td>
<td>16</td>
<td>14.6393</td>
</tr>
<tr>
<td>52</td>
<td>98</td>
<td>7</td>
<td>7</td>
<td>13.7488</td>
</tr>
<tr>
<td>53</td>
<td>100</td>
<td>35</td>
<td>16</td>
<td>15.7411</td>
</tr>
<tr>
<td>54</td>
<td>100</td>
<td>5</td>
<td>5</td>
<td>13.8375</td>
</tr>
<tr>
<td>55</td>
<td>95</td>
<td>25</td>
<td>16</td>
<td>17.1012</td>
</tr>
<tr>
<td>56</td>
<td>111</td>
<td>14</td>
<td>14</td>
<td>14.3483</td>
</tr>
<tr>
<td>57</td>
<td>92</td>
<td>13</td>
<td>13</td>
<td>14.0084</td>
</tr>
<tr>
<td>58</td>
<td>102</td>
<td>7</td>
<td>7</td>
<td>13.9503</td>
</tr>
<tr>
<td>59</td>
<td>92</td>
<td>17</td>
<td>17</td>
<td>16.4607</td>
</tr>
<tr>
<td>60</td>
<td>105</td>
<td>33</td>
<td>16</td>
<td>14.6503</td>
</tr>
<tr>
<td>61</td>
<td>95</td>
<td>8</td>
<td>8</td>
<td>14.6173</td>
</tr>
<tr>
<td>62</td>
<td>94</td>
<td>11</td>
<td>11</td>
<td>14.1256</td>
</tr>
<tr>
<td>63</td>
<td>103</td>
<td>7</td>
<td>7</td>
<td>13.9503</td>
</tr>
<tr>
<td>64</td>
<td>63</td>
<td>14</td>
<td>14</td>
<td>18.2579</td>
</tr>
<tr>
<td>65</td>
<td>100</td>
<td>30</td>
<td>15</td>
<td>15.7774</td>
</tr>
<tr>
<td>66</td>
<td>105</td>
<td>22</td>
<td>16</td>
<td>14.2337</td>
</tr>
<tr>
<td>67</td>
<td>109</td>
<td>7</td>
<td>7</td>
<td>14.3314</td>
</tr>
<tr>
<td>68</td>
<td>97</td>
<td>9</td>
<td>9</td>
<td>13.8481</td>
</tr>
<tr>
<td>69</td>
<td>96</td>
<td>16</td>
<td>16</td>
<td>14.1644</td>
</tr>
<tr>
<td>70</td>
<td>98</td>
<td>16</td>
<td>16</td>
<td>15.0636</td>
</tr>
<tr>
<td>71</td>
<td>81</td>
<td>14</td>
<td>14</td>
<td>14.0775</td>
</tr>
<tr>
<td>72</td>
<td>104</td>
<td>16</td>
<td>16</td>
<td>15.0201</td>
</tr>
<tr>
<td>73</td>
<td>70</td>
<td>17</td>
<td>17</td>
<td>15.0369</td>
</tr>
<tr>
<td>74</td>
<td>90</td>
<td>22</td>
<td>16</td>
<td>14.354</td>
</tr>
<tr>
<td>75</td>
<td>75</td>
<td>15</td>
<td>15</td>
<td>14.8666</td>
</tr>
<tr>
<td>76</td>
<td>107</td>
<td>10</td>
<td>10</td>
<td>13.7906</td>
</tr>
<tr>
<td>77</td>
<td>92</td>
<td>24</td>
<td>16</td>
<td>14.8782</td>
</tr>
<tr>
<td>78</td>
<td>97</td>
<td>17</td>
<td>15</td>
<td>15.1509</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>79</td>
<td>111</td>
<td>17</td>
<td>16</td>
<td>15.7483</td>
</tr>
<tr>
<td>80</td>
<td>99</td>
<td>10</td>
<td>10</td>
<td>13.8598</td>
</tr>
<tr>
<td>81</td>
<td>94</td>
<td>33</td>
<td>17</td>
<td>17.5301</td>
</tr>
<tr>
<td>82</td>
<td>65</td>
<td>11</td>
<td>11</td>
<td>14.9404</td>
</tr>
<tr>
<td>83</td>
<td>103</td>
<td>5</td>
<td>5</td>
<td>15.4496</td>
</tr>
<tr>
<td>84</td>
<td>80</td>
<td>9</td>
<td>9</td>
<td>14.0569</td>
</tr>
<tr>
<td>85</td>
<td>83</td>
<td>13</td>
<td>13</td>
<td>14.3751</td>
</tr>
<tr>
<td>86</td>
<td>116</td>
<td>39</td>
<td>16</td>
<td>15.1156</td>
</tr>
<tr>
<td>87</td>
<td>85</td>
<td>12</td>
<td>12</td>
<td>14.227</td>
</tr>
<tr>
<td>88</td>
<td>81</td>
<td>26</td>
<td>16</td>
<td>17.8452</td>
</tr>
<tr>
<td>89</td>
<td>100</td>
<td>11</td>
<td>11</td>
<td>13.739</td>
</tr>
<tr>
<td>90</td>
<td>110</td>
<td>40</td>
<td>15</td>
<td>14.8077</td>
</tr>
<tr>
<td>91</td>
<td>101</td>
<td>15</td>
<td>15</td>
<td>15.5761</td>
</tr>
<tr>
<td>92</td>
<td>95</td>
<td>30</td>
<td>16</td>
<td>17.0793</td>
</tr>
<tr>
<td>93</td>
<td>99</td>
<td>36</td>
<td>16</td>
<td>17.1459</td>
</tr>
<tr>
<td>94</td>
<td>87</td>
<td>26</td>
<td>16</td>
<td>17.643</td>
</tr>
<tr>
<td>95</td>
<td>103</td>
<td>24</td>
<td>16</td>
<td>15.987</td>
</tr>
<tr>
<td>96</td>
<td>83</td>
<td>27</td>
<td>15</td>
<td>14.7262</td>
</tr>
<tr>
<td>97</td>
<td>95</td>
<td>32</td>
<td>16</td>
<td>17.7549</td>
</tr>
<tr>
<td>98</td>
<td>99</td>
<td>31</td>
<td>16</td>
<td>14.8433</td>
</tr>
<tr>
<td>99</td>
<td>112</td>
<td>48</td>
<td>16</td>
<td>15.9446</td>
</tr>
<tr>
<td>100</td>
<td>91</td>
<td>9</td>
<td>9</td>
<td>13.9388</td>
</tr>
<tr>
<td>101</td>
<td>78</td>
<td>24</td>
<td>15</td>
<td>14.992</td>
</tr>
<tr>
<td>102</td>
<td>106</td>
<td>5</td>
<td>5</td>
<td>14.5612</td>
</tr>
<tr>
<td>103</td>
<td>101</td>
<td>19</td>
<td>16</td>
<td>14.2475</td>
</tr>
<tr>
<td>104</td>
<td>89</td>
<td>27</td>
<td>16</td>
<td>17.6654</td>
</tr>
<tr>
<td>105</td>
<td>58</td>
<td>13</td>
<td>13</td>
<td>15.2376</td>
</tr>
<tr>
<td>106</td>
<td>102</td>
<td>15</td>
<td>15</td>
<td>15.6779</td>
</tr>
<tr>
<td>107</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Inf</td>
</tr>
<tr>
<td>108</td>
<td>102</td>
<td>16</td>
<td>16</td>
<td>15.619</td>
</tr>
<tr>
<td>109</td>
<td>98</td>
<td>12</td>
<td>12</td>
<td>15.6092</td>
</tr>
<tr>
<td>110</td>
<td>100</td>
<td>26</td>
<td>16</td>
<td>14.3222</td>
</tr>
<tr>
<td>111</td>
<td>103</td>
<td>21</td>
<td>16</td>
<td>15.9963</td>
</tr>
<tr>
<td>112</td>
<td>106</td>
<td>33</td>
<td>17</td>
<td>15.2393</td>
</tr>
<tr>
<td>113</td>
<td>113</td>
<td>39</td>
<td>16</td>
<td>15.8501</td>
</tr>
<tr>
<td>114</td>
<td>87</td>
<td>18</td>
<td>15</td>
<td>16.7025</td>
</tr>
<tr>
<td>115</td>
<td>79</td>
<td>15</td>
<td>15</td>
<td>14.9198</td>
</tr>
<tr>
<td>116</td>
<td>104</td>
<td>29</td>
<td>16</td>
<td>14.6201</td>
</tr>
<tr>
<td>117</td>
<td>99</td>
<td>40</td>
<td>16</td>
<td>17.1178</td>
</tr>
<tr>
<td>118</td>
<td>99</td>
<td>28</td>
<td>14</td>
<td>14.7046</td>
</tr>
<tr>
<td>119</td>
<td>97</td>
<td>29</td>
<td>16</td>
<td>15.9227</td>
</tr>
<tr>
<td>120</td>
<td>87</td>
<td>12</td>
<td>12</td>
<td>16.5258</td>
</tr>
<tr>
<td>121</td>
<td>84</td>
<td>27</td>
<td>16</td>
<td>17.4953</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>122</td>
<td>83</td>
<td>11</td>
<td>11</td>
<td>16.5854</td>
</tr>
<tr>
<td>123</td>
<td>104</td>
<td>29</td>
<td>16</td>
<td>17.3527</td>
</tr>
<tr>
<td>124</td>
<td>86</td>
<td>21</td>
<td>16</td>
<td>15.285</td>
</tr>
<tr>
<td>125</td>
<td>97</td>
<td>26</td>
<td>16</td>
<td>14.601</td>
</tr>
<tr>
<td>126</td>
<td>110</td>
<td>38</td>
<td>16</td>
<td>15.8131</td>
</tr>
<tr>
<td>127</td>
<td>97</td>
<td>35</td>
<td>16</td>
<td>14.8189</td>
</tr>
<tr>
<td>128</td>
<td>78</td>
<td>24</td>
<td>16</td>
<td>14.8608</td>
</tr>
<tr>
<td>129</td>
<td>68</td>
<td>16</td>
<td>16</td>
<td>15.2846</td>
</tr>
<tr>
<td>130</td>
<td>85</td>
<td>33</td>
<td>16</td>
<td>17.5661</td>
</tr>
<tr>
<td>131</td>
<td>85</td>
<td>23</td>
<td>16</td>
<td>17.9415</td>
</tr>
<tr>
<td>132</td>
<td>109</td>
<td>12</td>
<td>12</td>
<td>13.7381</td>
</tr>
<tr>
<td>133</td>
<td>108</td>
<td>11</td>
<td>11</td>
<td>14.7103</td>
</tr>
<tr>
<td>134</td>
<td>107</td>
<td>36</td>
<td>15</td>
<td>13.8569</td>
</tr>
<tr>
<td>135</td>
<td>106</td>
<td>40</td>
<td>16</td>
<td>15.8845</td>
</tr>
<tr>
<td>136</td>
<td>62</td>
<td>14</td>
<td>14</td>
<td>14.9294</td>
</tr>
<tr>
<td>137</td>
<td>81</td>
<td>6</td>
<td>6</td>
<td>14.0135</td>
</tr>
<tr>
<td>138</td>
<td>96</td>
<td>36</td>
<td>16</td>
<td>17.3357</td>
</tr>
<tr>
<td>139</td>
<td>104</td>
<td>31</td>
<td>16</td>
<td>14.8357</td>
</tr>
<tr>
<td>140</td>
<td>70</td>
<td>16</td>
<td>16</td>
<td>15.0719</td>
</tr>
<tr>
<td>141</td>
<td>105</td>
<td>18</td>
<td>16</td>
<td>15.2511</td>
</tr>
<tr>
<td>142</td>
<td>113</td>
<td>35</td>
<td>16</td>
<td>13.6782</td>
</tr>
<tr>
<td>143</td>
<td>108</td>
<td>17</td>
<td>16</td>
<td>14.1683</td>
</tr>
<tr>
<td>144</td>
<td>100</td>
<td>7</td>
<td>7</td>
<td>13.8936</td>
</tr>
<tr>
<td>145</td>
<td>77</td>
<td>14</td>
<td>14</td>
<td>14.8329</td>
</tr>
<tr>
<td>146</td>
<td>107</td>
<td>8</td>
<td>8</td>
<td>13.6847</td>
</tr>
<tr>
<td>147</td>
<td>110</td>
<td>28</td>
<td>16</td>
<td>14.6913</td>
</tr>
<tr>
<td>148</td>
<td>55</td>
<td>15</td>
<td>15</td>
<td>15.1551</td>
</tr>
<tr>
<td>149</td>
<td>108</td>
<td>32</td>
<td>16</td>
<td>15.3651</td>
</tr>
<tr>
<td>150</td>
<td>73</td>
<td>25</td>
<td>16</td>
<td>14.9298</td>
</tr>
<tr>
<td>151</td>
<td>82</td>
<td>5</td>
<td>5</td>
<td>14.0697</td>
</tr>
<tr>
<td>152</td>
<td>108</td>
<td>30</td>
<td>16</td>
<td>15.4116</td>
</tr>
<tr>
<td>153</td>
<td>103</td>
<td>13</td>
<td>13</td>
<td>13.6848</td>
</tr>
<tr>
<td>154</td>
<td>95</td>
<td>16</td>
<td>16</td>
<td>14.2216</td>
</tr>
<tr>
<td>155</td>
<td>91</td>
<td>2</td>
<td>2</td>
<td>13.9672</td>
</tr>
<tr>
<td>156</td>
<td>75</td>
<td>18</td>
<td>16</td>
<td>15.0965</td>
</tr>
<tr>
<td>157</td>
<td>78</td>
<td>24</td>
<td>16</td>
<td>14.9534</td>
</tr>
<tr>
<td>158</td>
<td>101</td>
<td>24</td>
<td>16</td>
<td>15.5052</td>
</tr>
<tr>
<td>159</td>
<td>107</td>
<td>7</td>
<td>7</td>
<td>14.5022</td>
</tr>
<tr>
<td>160</td>
<td>108</td>
<td>31</td>
<td>16</td>
<td>14.5392</td>
</tr>
<tr>
<td>161</td>
<td>100</td>
<td>34</td>
<td>15</td>
<td>15.8422</td>
</tr>
<tr>
<td>162</td>
<td>85</td>
<td>8</td>
<td>8</td>
<td>14.3405</td>
</tr>
<tr>
<td>163</td>
<td>103</td>
<td>36</td>
<td>16</td>
<td>15.6947</td>
</tr>
</tbody>
</table>
Table 2: Results from method 2 using the predefined mask

<table>
<thead>
<tr>
<th>Name of Image</th>
<th>Original Hamming Distance</th>
<th>1st Stage Hamming Distance</th>
<th>2nd Stage (Final) Hamming Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>113</td>
<td>63</td>
<td>41</td>
</tr>
<tr>
<td>2</td>
<td>113</td>
<td>96</td>
<td>60</td>
</tr>
<tr>
<td>3</td>
<td>109</td>
<td>92</td>
<td>52</td>
</tr>
<tr>
<td>4</td>
<td>111</td>
<td>64</td>
<td>55</td>
</tr>
<tr>
<td>5</td>
<td>92</td>
<td>77</td>
<td>61</td>
</tr>
<tr>
<td>6</td>
<td>103</td>
<td>92</td>
<td>61</td>
</tr>
<tr>
<td>7</td>
<td>92</td>
<td>74</td>
<td>39</td>
</tr>
<tr>
<td>8</td>
<td>115</td>
<td>82</td>
<td>68</td>
</tr>
<tr>
<td>9</td>
<td>99</td>
<td>67</td>
<td>35</td>
</tr>
<tr>
<td>10</td>
<td>109</td>
<td>57</td>
<td>48</td>
</tr>
<tr>
<td>11</td>
<td>88</td>
<td>68</td>
<td>39</td>
</tr>
<tr>
<td>12</td>
<td>91</td>
<td>77</td>
<td>37</td>
</tr>
<tr>
<td>13</td>
<td>95</td>
<td>34</td>
<td>30</td>
</tr>
<tr>
<td>14</td>
<td>86</td>
<td>49</td>
<td>47</td>
</tr>
<tr>
<td>15</td>
<td>96</td>
<td>99</td>
<td>47</td>
</tr>
<tr>
<td>16</td>
<td>97</td>
<td>34</td>
<td>27</td>
</tr>
<tr>
<td>17</td>
<td>93</td>
<td>76</td>
<td>60</td>
</tr>
<tr>
<td>18</td>
<td>102</td>
<td>60</td>
<td>46</td>
</tr>
<tr>
<td>19</td>
<td>87</td>
<td>55</td>
<td>26</td>
</tr>
<tr>
<td>20</td>
<td>112</td>
<td>35</td>
<td>20</td>
</tr>
<tr>
<td>21</td>
<td>107</td>
<td>73</td>
<td>45</td>
</tr>
<tr>
<td>22</td>
<td>105</td>
<td>75</td>
<td>43</td>
</tr>
<tr>
<td>23</td>
<td>108</td>
<td>29</td>
<td>25</td>
</tr>
<tr>
<td>24</td>
<td>102</td>
<td>89</td>
<td>56</td>
</tr>
<tr>
<td>25</td>
<td>107</td>
<td>91</td>
<td>74</td>
</tr>
<tr>
<td>26</td>
<td>113</td>
<td>75</td>
<td>51</td>
</tr>
<tr>
<td>27</td>
<td>89</td>
<td>48</td>
<td>44</td>
</tr>
<tr>
<td>28</td>
<td>109</td>
<td>83</td>
<td>55</td>
</tr>
<tr>
<td>29</td>
<td>104</td>
<td>81</td>
<td>55</td>
</tr>
<tr>
<td>30</td>
<td>110</td>
<td>57</td>
<td>29</td>
</tr>
<tr>
<td>31</td>
<td>106</td>
<td>72</td>
<td>42</td>
</tr>
<tr>
<td>32</td>
<td>111</td>
<td>83</td>
<td>54</td>
</tr>
<tr>
<td>33</td>
<td>96</td>
<td>78</td>
<td>35</td>
</tr>
<tr>
<td>34</td>
<td>113</td>
<td>67</td>
<td>44</td>
</tr>
<tr>
<td>35</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>36</td>
<td>94</td>
<td>66</td>
<td>43</td>
</tr>
<tr>
<td>37</td>
<td>110</td>
<td>101</td>
<td>63</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>38</td>
<td>107</td>
<td>113</td>
<td>89</td>
</tr>
<tr>
<td>39</td>
<td>112</td>
<td>80</td>
<td>60</td>
</tr>
<tr>
<td>40</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>41</td>
<td>106</td>
<td>94</td>
<td>61</td>
</tr>
<tr>
<td>42</td>
<td>79</td>
<td>52</td>
<td>32</td>
</tr>
<tr>
<td>43</td>
<td>101</td>
<td>97</td>
<td>60</td>
</tr>
<tr>
<td>44</td>
<td>98</td>
<td>87</td>
<td>53</td>
</tr>
<tr>
<td>45</td>
<td>97</td>
<td>67</td>
<td>48</td>
</tr>
<tr>
<td>46</td>
<td>107</td>
<td>29</td>
<td>20</td>
</tr>
<tr>
<td>47</td>
<td>118</td>
<td>76</td>
<td>60</td>
</tr>
<tr>
<td>48</td>
<td>110</td>
<td>57</td>
<td>29</td>
</tr>
<tr>
<td>49</td>
<td>91</td>
<td>77</td>
<td>43</td>
</tr>
<tr>
<td>50</td>
<td>116</td>
<td>74</td>
<td>48</td>
</tr>
</tbody>
</table>
Appendix C: MATLAB Codes

Manual Test to find the Threshold

```matlab
name='B';
k1=20;
v=20;

clc
addr=sprintf('%s.jpg',name);
A=imread(addr);
I=rgb2gray(A);
x=1;
y=1;
x2=16;
y2=16;
ylim=((y2)-1).*(256./y2)+1;
hb=(x2.*y2);
imdim=256;
k=1;
while(x<=imdim)
  for j=1:1:x2
    for i=1:1:y2
      F(i,j,k)=I(i+y-1,j+x-1);
    end
  end
  if(y<ylim)
    y=y+y2;
  else
    y=1;
    x=x+x2;
  end
  k=k+1;
end

I1=I;
I2=I;

if(rem(k1,16)==0)
  k2=k1-1;
else
  k2=k1;
end
cor=(fix(k2./y2).*x2);
x3=cor+1;
y3=((k1-cor-1).*y2)+1;
for o=x3:1:(x3+x2-1)
  for p=y3:1:(y3+x2-1)
    Q(p-y3+i-o-x3+1)=I(p,o);
  end
end
```

61
Q=double(Q);
m1=mean(mean(Q));
v1=var(var(Q));
Q1=Q;
m2=m1;
while(abs(m2-m1)<v)
    for o=1:16
        for p=1:16
            Q1(p,o)=Q1(p,o)./1.001;
        end
    end
    m2=mean(mean(Q1));
end
if(rem(k1,16)==0)
    k2=k1-1;
else
    k2=k1;
end
cor=(fix(k2./y2).*x2);
x3=cor+1;
y3=((k1-cor-1).*y2)+1;
for o=x3:1:(x3+x2-1)
    for p=y3:1:(y3+x2-1);
        I1(p,o)=Q1(p-y3+1,o-x3+1);
    end
end
Q=double(Q);
Q2=Q;
m3=m1;
while(abs(m3-m1)<v)
    for o=1:16
        for p=1:16
            Q2(p,o)=Q2(p,o).*1.001;
        end
    end
    m3=mean(mean(Q2));
end
if(rem(k1,16)==0)
    k2=k1-1;
else
    k2=k1;
end
cor=(fix(k2./y2).*x2);
x3=cor+1;
y3=((k1-cor-1).*y2)+1;
for o=x3:1:(x3+x2-1)
    for p=y3:1:(y3+x2-1);
        I2(p,o)=Q2(p-y3+1,o-x3+1);
    end
end
I=double(I);
\( M_I = \text{mean(\text{mean}(I))}; \)  
\( V_I = \text{var(\text{var}(I))}; \)  
\text{disp('Name Of Image');}  
\text{disp(name);}  
\text{disp('Mean Of Image');}  
\text{disp(MI);}  
\text{disp('Variance of Image');}  
\text{disp(VI);}  
\text{disp('Block Number=');}  
\text{disp(k1);}  
\text{disp('Mean of Block=');}  
\text{disp(ml);}  
\text{disp('Variance of Block=');}  
\text{disp(v1);}  

\text{subplot(1,3,1); imshow(I,[ ]); title('original image');}  
\text{rectangle('Position',[x3,y3,16,16])}  
\text{subplot(1,3,2); imshow(I1); title('-');}  
\text{subplot(1,3,3); imshow(I2); title('+');}  

\text{Hash Function}  
\text{function SaveHash(NumberOfAttackImages)}  
\text{for num=1:NumberOfAttackImages}  
\text{addr=sprintf('attack set\'num');}  
\text{A=imread(addr);}  
\text{I=rgb2gray(A);}  
\text{x=1;}  
\text{y=1;}  
\text{x2=16;}  
\text{y2=16;}  
\text{for k=1:256}  
\text{if(rem(k,16)==0)}  
\text{k2=k-1;}  
\text{else}  
\text{k2=k;}  
\text{end}  
\text{cor=(fix(k2./y2).*x2);}  
\text{x3=cor+1;}  
\text{y3=((k-cor-1).*y2)+1;}  
\text{for o=x3:1:(x3+x2-1)}  
\text{for p=y3:1:(y3+y2-1)}  
\text{Q(p-y3+1,o-x3+1)=I(p,o);}  
\text{end}  
\text{end}  
\text{Q=double(Q);}  
\text{M(k)=round(mean(mean(Q))));}  
\text{end}  
\text{S=sort(M);}  \% \text{sort averages from min to max}  
\text{Md=S(127);}  \% \text{find median}  
\text{H=M>Md;}  
\text{H=sprintf('\%d',H);}  
\text{for p=1:256}  
\text{H1(num,p)=H(p);}  
\text{end}
find similar hash

function num=SimilarHash(NameOfOriginalImage)
addr=sprintf('%s.jpg',NameOfOriginalImage);
A=imread(addr);
I=RGB2gray(A);
x=1;
y=1;
x2=16;
y2=16;
for k=1:256
    if(rem(k,16)==0)
        k2=k-1;
    else
        k2=k;
    end
    cor=(fix(k2./y2).*x2);
x3=cor+1;
y3=((k-cor-1).*y2)+1;
    for o=x3:1:(x3+x2-1)
        for p=y3:1:(y3+x2-1)
            Q(p-y3+1,o-x3+1)=I(p,o);
        end
    end
Q=double(Q);
M(k)=round(mean(mean(Q)));
end
S=sort(M); %sort averages from min to max
Md=S(127); %find median
H0=M>Md;
H0=sprintf('%d',H0);
H1=xlsread('attack set\AH.xls');
x=size(H1);
for i=1:x(1)
    H2=xlsread('%d',H1(i,:));
    Dist(i)=sum(H2~=H0); 
end
[va,num]=min(Dist);
num=sprintf('%d',num);
end
Hash Distance

```matlab
function HashDist(I1,S1,I2,S2)
add1=sprintf('%s.%s',I1,S1);
add2=sprintf('%s.%s',I2,S2);
A1=imread(add1);
A2=imread(add2);
[a,b,c]=size(A1);
if(c==3), A1=rgb2gray(A1); end
[a,b,c]=size(A2);
if(c==3), A2=rgb2gray(A2); end

x=1;
y=1;
x2=16;
y2=16;
%HashNumber Of 1st Image -----------------------------------------------------------------
for k=1:256
if(rem(k,16)==0)
k2=k-1;
else
    k2=k;
end
    cor=(fix(k2./y2).*x2);
x3=cor+1;
y3=((k-cor-1).*y2)+1;
    for o=x3:1:(x3+x2-1)
        for p=y3:1:(y3+x2-1);
            Q(p-y3+1,o-x3+1)=A1(p,o);
        end
    end
Q=double(Q);
me(k)=round(mean(mean(Q))); end
So=sort(me);
Md=So(127);
H1=me>Md;
H1=sprintf('%d',H1)
%---------------------------------------------------------------------------------------
%HashNumber Of 2nd Image -----------------------------------------------------------------
for k=1:256
if(rem(k,16)==0)
k2=k-1;
else
    k2=k;
end
    cor=(fix(k2./y2).*x2);
x3=cor+1;
y3=((k-cor-1).*y2)+1;
    for o=x3:1:(x3+x2-1)
        for p=y3:1:(y3+x2-1);
            Q(p-y3+1,o-x3+1)=A2(p,o);
        end
    end
Q=double(Q);
me(k)=round(mean(mean(Q))); end
end
```

65
So=sort(me);
Md=So(127);
H2=me>Md;
H2=sprintf('%d',H2)
%--------------------------------------------------------------------------
Dist=sum(H2~=H1)
end

Stage 1

%--------------------------------------------------------------------------  Stage 1  %--------------------------------------------------------------------------
function  
[I1,I2,H2,Dist2,Dist1]=Stage1M(OriginalImageName,cond,BD,wh,bl,method,ShRe,SaRe)
%--------------------------------------------------------------------------
nname=OriginalImageName;
nname2=SimilarHash(name);
%--------------------------------------------------------------------------
addr=sprintf('%s.jpg',name);
A=imread(addr);
I=rgb2gray(A);
x=1;
y=1;
x2=16;
y2=16;

%Find Median & Mean of Blocks (1 to 256)for OriginalImage -----------------
for k=1:256
  if(rem(k,16)==0)
    k2=k-1;
  else
    k2=k;
  end
  cor=(fix(k2./y2).*x2);
x3=cor+1;
y3=((k-cor-1).*y2)+1;
  for o=x3:1:(x3+x2-1)
    for p=y3:1:(y3+x2-1);
      Q(p-y3+1,o-x3+1)=I(p,o);
    end
  end
Q=double(Q);
me(k)=round(mean(mean(Q)));
end
So=sort(me);
Md=So(127);
%--------------------------------------------------------------------------
%Caclulate Hash Number of OriginalImage------------------------------------
H=me>Md;
H2=sprintf('%d',H);
%--------------------------------------------------------------------------
%Make Original Image Hash bit as a Picture (MASK)--------------------------
for i=1:16
  for j=1:16
    H00(j,i)=H(j+(16.*(i-1)));
  end
end
%read attack Image
addr=sprintf('%s%s.jpg', 'attack set\', name2);
A=imread(addr);
I2=rgb2gray(A);
I2=uint8(I2);

%Find Median and Mean of Blocks of Original attack Image
for k=1:256
if(rem(k,16)==0)
k2=k-1;
else
k2=k;
end
cor=(fix(k2./y2).*x2);
x3=cor+1;
y3=((k-cor-1).*y2)+1;
for o=x3:1:(x3+x2-1)
for p=y3:1:(y3+x2-1);
Q(p-y3+1,o-x3+1)=I2(p,o);
end
end
Q=double(Q);
me(k)=round(mean(mean(Q)));
end
So=sort(me);
Md=So(127);

%Calculate Hash number of Original attack number
H=me>Md;
H4=sprintf('%d',H);

%Make Modified attack Image Hash bit as a Picture
for i=1:16
for j=1:16
H11(j,i)=H(j+(16.*(i-1)));
end
end

Dist1=sum(H2~=H4); % Hamming Distance of Original Image with Original Attack Image
ad='attack set\';
if((Dist1-2)>cond)
name2=FavSimilarHash(name);
ad='attack set\Fav\';
%read attack Image again
addr=sprintf('%s%s.jpg', ad, name2);
A=imread(addr);
I2=rgb2gray(A);
I2=uint8(I2);

%Find Median and Mean of Blocks of Original attack Image
for k=1:256
if(rem(k,16)==0)
k2=k-1;
else
k2=k;
end
cor=(fix(k2./y2).*x2);
x3=cor+1;
y3=((k-cor-1).*y2)+1;
for o=x3:1:(x3+x2-1)
    for p=y3:1:(y3+x2-1);
        Q(p-y3+1,o-x3+1)=I2(p,o);
    end
end
Q=double(Q);
me(k)=round(mean(mean(Q)));
end
So=sort(me);
Md=So(127);
%Calculate Hash number of Original attack number -------------------------
H=me>Md;
H4=sprintf('%d',H);
%Make Modified attack Image Hash bit as a Picture ------------------------
for i=1:1:16
    for j=1:1:16
        H11(j,i)=H(j+(16.*(i-1)));
    end
end
%------------------------------------------------------------------------
Dist1=sum(H2~=H4); % Hamming Distance of Original Image with Original Attack Image
end
%------------------------------------------------------------------------
BA = double(imresize(H00, 16)); % Resize H00(HashBit Image(Mask) to 256x256)
if(method==2)
    for i=1:1:30
        MaB=im2bw(imread(['Masks\',num2str(i),'.jpg']));
        cor(i)=sum(sum(MaB.*BA));
    end
    [vcor,icor]=max(cor);
    BA=im2bw(imread(['Masks\',num2str(icor),'.jpg']));
end
B3 = BA; %imclose(BA,strel('square',28));
BD=level of bluring as an input for function
B = conv2(double(B3),fspecial('average',BD),'same'); %Smooth Mask by BD...
BD determines the level of bluring, Higher BD -> more blurring -> less changes in picture

%method1-------------------------------------------------------------------
B2=(B.*(wh-bl))+bl;%Complete the mask by setting some degree of effects on Smoothed Mask
%------------------------------------------------------------------------
if(ShRe==1)
    % show result
    subplot(1,3,1); imshow(BA); title('OriginalImage Mask');
% read the attack image again ---------------------------------------------
addr=sprintf('%s%s.jpg',ad,name2);
A=imread(addr);
I1=double(rgb2gray(A));
if((Dist1-2)>cond)
    I1=I1.*B2; % multiply attack image by mask to get a similar hash number
end
I1=uint8(I1);

% Find Median and Mean of Blocks of Modified attack Image ------------------
for k=1:256
    if(rem(k,16)==0)
        k2=k-1;
    else
        k2=k;
    end
    cor=(fix(k2./y2).*x2);
    x3=cor+1;
    y3=((k-cor-1).*y2)+1;
    for o=x3:1:(x3+x2-1)
        for p=y3:1:(y3+x2-1);
            Q(p-y3+1,o-x3+1)=I1(p,o);
        end
    end
    Q=double(Q);
    me(k)=round(mean(mean(Q)));
end
So=sort(me);
Md=So(127);

% Calculate Hash number of Modified attack number --------------------------
H=me>Md;
H3=sprintf('%d',H);
% Make Modified attack Image Hash bit as a Picture -------------------------
for i=1:16
    for j=1:16
        H22(j,i)=H(j+(16.*(i-1)));
    end
end

% Hamming Distance of Original Image with Modified Attack Image
Dist2=sum(H2~=H3);
if(ShRe==1)
    % show result
    subplot(2,3,1); imshow(I,[]); title('Original image');
    subplot(2,3,2); imshow(I2,[]); title('attack image');
    subplot(2,3,3); imshow(I1,[]); title('modified attack image Stage1');
end
Stage 2

%%%%%%%%%%%%%%%%% Stage 2 %%%%%%%%%%%%%%%%%%%%%%%

function [I, Dist1] = Stage2M(I, H0, limit, BEST, name, ShRe, SaRe)

%x=1;
y=1;
x2=16;
y2=16;
for k=1:256
if(rem(k,16)==0)
    k2=k-1;
else
    k2=k;
end
cor=(fix(k2./y2).*x2);
x3=cor+1;
y3=((k-cor-1).*y2)+1;
    for o=x3:1:(x3+x2-1)
        for p=y3:1:(y3+x2-1);
            Q(p-y3+1,o-x3+1)=I(p,o);
        end
    end
Q=double(Q);
M0(k)=round(mean(mean(Q)));
V(k)=var(var(Q));
end
S=sort(M0); %sort averages from min to max
Md=S(127); %find median
H=M0>Md;
H=sprintf('%d',H);
Dist=sum(H0~=H); % hamming distance of stage 1 modified attack image with original image

M=M0;
%Vn LOOKUP TABLE
Vn=[0 0 100 100 1000 1000 20000 20000 30000 30000 30000 30000 30000 30000
30000 2500000 2500000 2500000 2500000 2500000 2500000 2500000 2500000
2500000];

% which bits are different? put them in HDi array------------------------
t=0;
HDI=[];
i=1;
while(i-1~=Dist)
t=t+1;
    if(H(t)==H0(t)), HDi(i)=t; i=i+1; end
end
%-----------------------------------------------------------------------
I1=I;
liD=Limit;% Condition -- if Dist become less than Limit Stop Modifying--as an
input for function

%Start Modifying Stage2-----------------------------------------------
n=0;
z=0;
while(Dist>liD && n<=(23+z))
n=n+1;
    if(z==0), Vn(n)=0; end
    for i=1:numel(HDI)
        if((M(HDI(i))==Md+n) && (V(HDI(i))>=Vn(n)))
            [I1]=ChangeMean(I1,n,HDI(i),'+');
            M(HDI(i))=Md;
            S=sort(M);
            H(HDI(i))='0';
            HDI(i)=0;
        end
    end
    ei=find(HDI==0);
    for i=1:numel(ei)
        HDI(ei(i)-1+1)=[];
    end
    ei=[];
    Dist=sum(H0~=H);
end

m=1;
while(Dist>liD && m<=n)
    for i=1:numel(HDI)
        if((M(HDI(i))==Md-m+1) && (V(HDI(i))>=Vn(n)) && (S(128)==Md))
            I1=ChangeMean(I1,m,HDI(i),'-');
            M(HDI(i))=Md+1;
            S=sort(M);
            H(HDI(i))='1';
            HDI(i)=0;
        end
    end
    ei=find(HDI==0);
    for i=1:numel(ei)
        HDI(ei(i)-i+1)=[];
    end
    ei=[];
    Dist=sum(H0~=H);
end
m=m+1;
end
if((n==24) && (Dist>liD) && strcmp(BEST, 'no') && (z==0)), n=0; z1=255-Md; z2=Md; z=max(z1,z2); end
end

% disp(n) % max change in pixel intensity-----------------------------------
if(ShRe==1)
% show result
subplot(1,2,1); imshow(I); title('Stage1 Modified attack Image');
subplot(1,2,2); imshow(I1); title('Stage2 Modified attack Image (Final)');
figure();
end

% verification--------------------------------------------------------------
I=I1;
x=1;
y=1;
x2=16;
y2=16;
for k=1:256
if(rem(k,16)==0)
k2=k-1;
else
k2=k;
end
cor=(fix(k2./y2).*x2);
x3=cor+1;
y3=((k-cor-1).*y2)+1;
for o=x3:1:(x3+x2-1)
  for p=y3:1:(y3+x2-1);
    Q(p-y3+1,o-x3+1)=I(p,o);
  end
end
Q=double(Q);
M1(k)=round(mean(mean(Q)));
end
S=sort(M1); % sort averages from min to max
Md=S(127); % find median
H1=M1>Md;
H1=sprintf('%d',H1);

Dist1=sum(H0==H1); % hamming distance of Original Image with Stage2 modified attack Image

if(SaRe==1)
% save result
dead=sprintf('%sName=%s--2ndStage--Dist=%d%s' ,'result\',name,Dist1,'.bmp');
imwrite(I,dead);
end

end
**PSNR**

```matlab
function p = psnr(x, y, vmax)
    if nargin<3
        m1 = max( abs(x(:)) );
        m2 = max( abs(y(:)) );
        vmax = max(m1,m2);
    end
    d = mean( (double(x(:))-double(y(:))).^2 );
    p = 10*log10( vmax^2/d );
end
```

**Final Code**

```matlab
HammingDistanceLimit2=16;
% Stage1M(Original ImagName,HammingDistanceLimit1,BD,wh,bl,method,ShRe if =1 ->show,SaReif =1 - >save)
[I1,I2,H2,Dist1,Dist0]=Stage1M(OriginalImageName,HammingDistanceLimit1,20,2,0 .5,1,1,0);
% Stage2M(I1,H2,Hamming Distance Limit,yes/no-> if no=lose quality,OIN,ShRe if =1 ->show,SaReif =1 ->save)
[I,Dist2]=Stage2M(I1,H2,HammingDistanceLimit2,'yes',OriginalImageName,0,0);
p=psnr(I2,I,255);

disp('Original Hamming Distance :')
disp(Dist0)
disp('1st Stage Hamming Distance :')
disp(Dist1)
disp('2nd Stage (Final) Hamming Distance :')
disp(Dist2)
disp('PSNR in dB :')
disp(p)
imshow(I); title('Final attack Image');
```