Analysis of the singular value decomposition in data hiding

by

Eric Tyler Hansen

A Creative Component submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Mathematics

Program of Study Committee:
Ryan Martin, Major Professor
Jennifer Davidson
Sung-Yell Song

Iowa State University

Ames, Iowa

2007

DEDICATION

I would like to dedicate this thesis to my mother Gail and grandmother Jean whose guidance in my early education was invaluable. I would also like to thank my friends and family for their guidance and support during the writing of this work.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this thesis.

First, to my committee members, Dr. Ryan Martin, Dr. Jennifer Davidson, and Dr. Sung-Yell Song for their encouragement and insight.

Many thanks to Clifford Bergman for great help in providing access to computing resources. Some of the basic SVD embedding code was written by Dan Wengerhoff. The second chi-square attack was conceived during discussions with Andrew Regenscheid.

Most of the images used for testing were obtained from the Watermarking Evaluation Testbed (http://www.datahiding.org), the others are copyright Marvel Comics.

ABSTRACT

The singular value decomposition has appeared in several capacities since 2002 in data hiding, specifically watermarking and steganography. Several of the applications of the singular value decomposition are inspected and reviewed.

The singular value decomposition embedding algorithm by Bergman and Davidson (2005) is one of these applications whose robustness has not been thoroughly tested. Implementing the linear chi-square attack suggested by Westfeld and Pfitzmann (2000) on images containing information hidden with this singular value decomposition data hiding algorithm yielded two main results.

First, implementing the chi-square attack on images with embedded messages was successful, but required interpretation of the results. Second, chi-square attacking the singular value decomposition matrices of an image with data embedded did correctly identify both embedded images and clean images.

These results suggested certain changes could be made to improve the algorithm by Bergman and Davidson (2005). These changes were implemented and the resulting algorithm was tested and attacked using an automated method.

CHAPTER 1. Motivation

Modern data hiding has a variety of uses but two of the primary fields are watermarking and steganography.

Digital watermarking is of high priority with increasing use of digital media and increasing conflict concerning copyright infringement and piracy. A secure watermarking algorithm would allow creators of digital property to prove their ownership of files. It would also allow for tracking of files in controlled distribution, resulting in better control of piracy.

Thusfar, no watermarking algorithms have been developed that satisfy the security expectations of the music and movie industries.

A survey of recent watermarking techniques using the singular value decomposition is included.

Digital steganography refers to the practice of altering an innocuous looking file to contain a secret message so that an observing adversary would be unaware that the secret message is being delivered, but only the innocent-looking file. This has diverse military and espionage purposes in the current global environment. Steganalysis, the pursuit of analyzing data files that may contain secret messages, has similar applications as well as domestic applications for online crime.

Many steganographic algorithms exist, as well as attacks to observe instances of these algorithms being used. Previous analysis of these algorithms and attacks have suggested that algorithms employing a certain type of embedding, in the least significant bit, are only resistant to attack if very few alterations are made, resulting in only a small message sent in a given file. A different algorithm that is able to securely embed a larger message in the same file would be desirable. Bergman and Davidson developed an embedding algorithm that does have a higher

safe embedding rate. However, it has not been thoroughly tested against other kinds of attack.

Several attacks on Bergman and Davidson's embedding algorithm have been performed in the course of this research and changes to the algorithm based on the results of these attacks are described in this paper.

CHAPTER 2. Definitions

Describing the singular value decomposition as applied to watermarking and steganography requires defining some terminology and establishing certain notation.

2.1 Singular value decomposition

Basic theorems and terms from linear algebra are taken from the Handbook of Linear Algebra [Hogben, et al (2006)]. A singular value decomposition of an $n \times n$ real-valued matrix A is a factorization $A = U(\Sigma)V^T$ with several constraints.

- Σ is a diagonal matrix with real, nonnegative diagonal entries $\sigma_1, \sigma_2, \ldots, \sigma_n$ such that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$.

- U and V^T must be real and orthogonal, which means $V^TV = U^TU = I$, the identity matrix; alternately a matrix is orthogonal if its column vectors are pairwise orthogonal unit vectors.

- **Singular values** of A are the diagonal entries of Σ, $\sigma_1, \sigma_2, \ldots, \sigma_n$. These are of the form $\sqrt{\lambda}$ where λ is an is an eigenvalue of AA^T. These eigenvalues λ are always real and nonnegative, and are uniquely determined.

- **Singular vectors** of A are the columns of U and V, respectively called the **left singular vectors** and **right singular vectors**. Singular vectors for U will be pairwise orthogonal

unit vectors, as will those for V.

- An example [Hogben, et al (2006)] of such a decomposition follows: Let

$$A = \begin{bmatrix} 1 & 2 \\ 2 & 0 \\ 0 & 2 \end{bmatrix} \text{ and } A^T = \begin{bmatrix} 1 & 2 & 0 \\ 2 & 0 & 2 \end{bmatrix}$$

The eigenvalues of $A^TA = \begin{bmatrix} 5 & 2 \\ 2 & 8 \end{bmatrix}$ are 9 and 4. So, the singular values of A are 3 and 2.

Normalized eigenvectors for A^TA are $v_1 = \begin{bmatrix} \frac{1}{\sqrt{5}} \\ \frac{2}{\sqrt{5}} \end{bmatrix}$ and $v_2 = \begin{bmatrix} \frac{2}{\sqrt{5}} \\ \frac{-1}{\sqrt{5}} \end{bmatrix}$

To compute the orthogonal matrix U, compute $u_1 = 1/3Av_1 = \begin{bmatrix} \frac{1}{3\sqrt{5}} \\ \frac{2}{3\sqrt{5}} \\ \frac{4}{3\sqrt{5}} \end{bmatrix}$ and $u_2 = 1/2Av_2 = \begin{bmatrix} 0 \\ \frac{2}{\sqrt{5}} \\ \frac{-1}{\sqrt{5}} \end{bmatrix}$ then the Gram-Schmidt process on u_1, u_2, and e_1 yields $u_3 = \begin{bmatrix} 2/3 \\ -1/3 \\ \frac{-1}{\sqrt{5}} \end{bmatrix}$

A has the singular value decomposition $A = U(\Sigma)V^T$ with

$$U = \frac{1}{3\sqrt{5}} \begin{bmatrix} 5 & 0 & 2 \\ 2 & 6 & -\sqrt{5} \\ 4 & -3 & -2\sqrt{5} \end{bmatrix}, \Sigma = \begin{bmatrix} 3 & 2 \\ 0 & 2 \\ 0 & 0 \end{bmatrix}, V = \frac{1}{\sqrt{5}} \begin{bmatrix} 1 & 2 \\ 2 & -1 \end{bmatrix},$$

- A vector $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ is **lexicographically positive** if its first nonzero component is positive.

- A singular value decomposition $A = U(\Sigma)V^T$ is **normal** if the columns of U are lexicographically positive and the diagonal entries of Σ are in non-increasing order.

- A matrix has a **unique normal singular value decomposition** if its singular values are pairwise distinct and nonzero [Bergman and Davidson (2005)].

- The example given previously is a normal singular value decomposition.

2.2 Steganography

Steganography comes from the Greek "covered writing," and refers to the hiding of messages in a medium with the intent that this medium can be inspected by external parties and that these outside observers remain unaware that there is any message being transported besides the medium itself.

This practice has an extensive history. Herodotus, in Ancient Greece, mentions its use in sending messages regarding forthcoming attacks. Modern steganography includes many ingenious tales from World War II including microdots (an entire message scaled to the size of a period in another innocent-looking message) and a New York City doll dealer sending messages through her doll shipping orders.

With the advent of modern computing, steganography is often achieved by embedding an encrypted message into a computer file, like a picture, movie, or sound file.

- A **cover file** is the file into which a secret message will be embedded. This is also called the carrier medium.

- **Payload** is one name for the message that is to be embedded and delivered.

- A **stego file** is a file which has been altered to contain a message.

- A **clean file** has not been altered since its creation. For example, this could refer to a digital photograph, a company logo, or a music file.

2.2.1 Digital Image Steganography

The following definitions will refer specifically to steganography applied to digital, greyscale images.

- A **cover image** is an image file into which a secret message will be embedded.
- A **stego image** is an image file which has been altered to contain a message. This is also referred to as a **steganogram**.
- A **clean image** is an image which has not been altered since its creation.

2.2.2 Least significant bit techniques

One basic steganographic embedding technique for image files is the least significant bit embedding. This technique exploits the human eye's inability to detect insignificant changes in the values of the pixels of an image file.

- Bitmap image files consist of an array of pixels.
- A **pixel** is a single point in a graphic image. Each pixel has a value corresponding to the intensity of colors (or the intensity of gray) that pixel represents. This value is recorded in the data file for that image.
- The **least significant bit** (LSB) of a pixel, or pixel value, in an image file is the 1's digit of its binary expansion. It corresponds to the evenness or oddness of the pixel's value; i.e., even numbers' LSBs are 0, odd numbers' LSBs are 1.
- The **least significant bit plane** in an image is the collection of least significant bits of the pixels of the image.

- Changes made directly to the pixels of an image are said to be made in the spatial domain, that is the space of the pixels of the image.

Each pixel in an 8-bit grayscale bitmap file, for example, is stored using a number from 0 to $255 = 2^8 - 1$. Color bitmap images are stored similarly, with a number for each of three color planes, giving $(2^8)^3$ values. For now, consider only grayscale images, since the concepts easily extend themselves to color images.

Changing every one of the pixel values in a bitmap file by 0 or 1 will not impact the way that a human consciously perceives the picture. This is a well-known result in data-hiding and one demonstrated in figure 2.1.

Figure 2.1 Original grayscale image and stego grayscale image with encrypted message hidden in the least significant bit plane

Thus, a steganographer is able to manipulate the least significant bit, that is, the even or odd parity, of these numbers between 0 and 255 to hide a message of 0s and 1s which can be extracted later by a co-conspirator. In such an embedding, one can consider the image bits in a linear fashion, i.e., in order; or visit the bits in a pseudo-random order; or in a visitation order determined by a key.

2.3 Steganalysis

In contrast to steganographic embedding techniques which embed messages with the intent of keeping them hidden, steganalysis is a subfield of steganography which seeks to discover the presence of such hidden messages, most often through statistical observations and comparisons. Both the embedding and steganalysis sides are sometimes collectively referred to as steganography.

- Perceptibility is used to describe how well an observer can detect changes to a file when observing it in the standard way. That is, looking at a picture file, listening to an audio file, etc. This is not an exact measurement as human senses vary and are subjective.

- Detectability describes how vulnerable a steganographic algorithm is to other means of observation, like statistical attacks or examinations of the actual data in the file.

2.3.1 Chi-square attack

The chi-square statistic was proposed by Karl Pearson in 1900. It is a measure of goodness-of-fit of a set of data to what would be expected if that data were random. It derives its name from the chi-square distribution, a special case of the gamma distribution. It was later applied in a steganographic context [Westfeld and Pfitzmann (2000)] to embedded images, motivated by the following observations.

First, consider that messages encrypted to use least significant bit embedding must contain only 0s and 1s.

Next, consider that a cryptosystem is considered most secure when its encrypted data is indistinguishable from random data, because a potential cryptanalyst would have less structure to work from. By using as much of the codeword space as possible, one derives a more secure cryptosystem, and this leads to encrypted messages which appear random. It is true that one can design a cryptosystem whose encrypted data tends to have $x\%$ of 1s and $(100 - x)\%$ of 0s, but an equal distribution of bits will appear more often, and is implemented in some of the existing steganography programs such as EzStego.

In contrast, it was observed by Westfeld and Pfitzmann (2000) and tested independently by McAdams and McKay (2005), that most naturally created, clean image files, for example, those taken with a digital camera, that the distribution of pairs of values (POVs), that is pixels with values $2i$ and $2i+1$, is not uniform.

There does not immediately arise a definitive line between what is random enough to be suspected to be a stego image and what is nonuniform enough to be suspected to be a clean image. This is where the chi-square statistical test comes in, as it is a test suited to examine this very difference and give a quantitative, statistical result.

Suppose that in a random sample of observations, there are k categories of values. Each value must fall in one and only one category. The chi-square statistic deals with measuring how well a given sample matches a theoretically random sample.

We will consider these k categories to be pairs of values of palette indices of our image, i.e., 0 and 1 are one pair of values, 2 and 3 are another, and so on. These pairs are examined because of the way LSB embedding works; that is, it will make the distribution of these pairs conspicuously similar because it only changes the evenness and oddness of the bits.

1. Let n_i be the number of pixels in the sample of the image with grey value $2i$. Consider those values with even parity without loss of generality.

2. Let n_i^* be the expected value of pixels in the sample of the image with grey value n_i, assuming that our image has been embedded with an uniformly distributed payload. That is,

$$n_i^* = \frac{|\text{pixels with gray value 2i}| + |\text{pixels with gray value 2i+1}|}{2}$$

3. A table of pairs of values can be crated. In the first column, store grey value $2i$ and in the second, grey value $2i+1$. When fewer than 8 pixels have values in $\{2i, 2i+1\}$, that i^{th} bin is combined with one of the bins adjacent to it, with even values combined together and odd values combined together.

4. The chi-square (χ^2) statistic with $k-1$ degrees of freedom is given as

$$\chi^2_{k-1} = \sum_{i=1}^{k} \frac{(n_i - n_i^*)^2}{n_i^*}.$$

This χ^2 statistic is related to, but not to be confused with, the χ^2 *distribution*, a special case of the gamma distribution, a mainstay of continuous probability density functions.

- A random variable Y is said to have a gamma distribution with parameters $\alpha > 0$ and $\beta > 0$ if and only if the density function of Y is

$$f(y) = \begin{cases} \dfrac{y^{\alpha-1}e^{-y/\beta}}{\beta^{\alpha}\Gamma(\alpha)}, & 0 \leq y < \infty; \\ 0, & \text{elsewhere.} \end{cases}$$

- A random variable Y is said to have a chi-square distribution with ν degrees of freedom if and only if Y is a gamma-distributed random variable with parameters $\alpha = \nu/2$ and $\beta = 2$.

- Define p as the probability of our statistic, that is, likelihood that our image has a message embedded in it, under the condition that the distributions n_i and n_i^* are equal.

$$p = 1 - \frac{1}{2^{\frac{k-1}{2}}\Gamma(\frac{k-1}{2})} \int_0^{\chi^2_{k-1}} e^{-\frac{x}{2}} x^{(\frac{k-1}{2}-1)}\, dx$$

- Since the chi-square test inspects the differences between pairs of values, it is sometimes called the Pairs of Values (POV) test or attack.

The values of the chi-square statistic are always positive; when the sum is taken over all categories, a value of the total variation for all the data is given. Larger values of the chi-square statistic result in a larger upper limit on the integral for p and thus a lower probability of embedding. Ways of interpreting p may vary based on a steganographer's goals. If a steganographer wishes to identify more stego images at the expense of falsely identifying some clean images, she would set her threshold T for values of p at a lower value; if she wanted to never falsely identify a clean image as a stego image, she would set the threshold higher. The choice of threshold T is explained in detail below in the ROC Curves section.

2.3.2 Iterated Chi-Square for length of embedding.

If only part of an image is changed for embedding, for example, the first half, then the chi-square test applied to the entire image will report it as a clean image, due to the presence of unmodified pixels at the end.

In the event that only part of an image is modified to contain a message, the chi-square test can still be used. The previous chi-square test is performed, but only examines a sample of the image at a time, namely the first $t\%$ of the image. The test is then repeated for values of t from 1 to 100. In this way, the chi-square test can also detect how much of the image has been altered. In practice, for an image to be considered likely to have been altered, a good guideline is to ignore the first five percent of the image because the chi-square statistic is very unstable there because on a very small scale, a clean image may have characteristics of an embedded image or vice versa. After that, at least one iteration should have a value higher than the determined threshold to be considered an embedded image.

The graph in figure 2.2 illustrates this process. The chi-square test has been run 100 times on the image, taking the first $t\%$ of the image on the t^{th} iteration. On approximately the 45^{th} iteration, corresponding to 45% of the image, the probability of embedding drops off, meaning approximately 45% of the image was used for embedding.

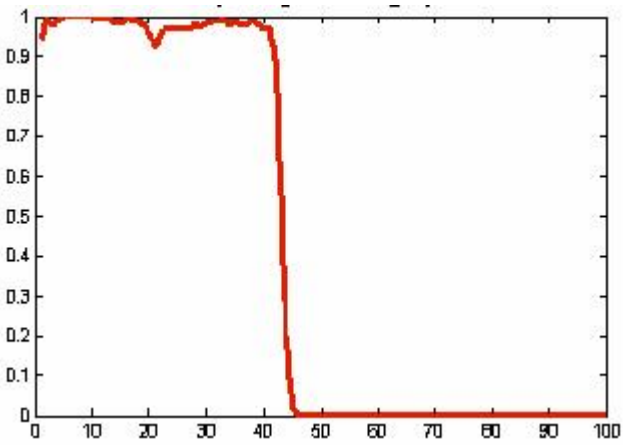

Figure 2.2 Graph of Sample Size vs. Probability of Embedding

2.3.3 ROC Curves

Receiver Operator Characteristic (ROC) Curves were developed in signal processing theory and have been applied to steganalysis as a measure of the effectiveness of a given steganalytic attack. They are a graphical representation of this effectiveness; the same data may be given in a table.

- For each image, we calculate the probability of embedding graph. We select a threshold, T, and count the number of steganographic images with values of p above T and divide by the number of steganographic images. This is our **true positive rate**, TPR, for this value of T.

  For the same value of T, we repeat the calculation on non-steganographic images to obtain our **false positive rate**, FPR, for this value of T. (FPR, TPR) becomes one point on the ROC curve. By varying values for T, we can produce several points for the ROC curve and interpolate remaining values.

  A test which chooses randomly would have its average (FPR, TPR) points on the diagonal between (0,0) and (1,1). A good test should have its curve as close to the upper left corner as possible; that is, the area under the curve should be as large as possible.

  In the ROC curves created in testing, threshold T values were 5k%, for k between 0 and 20.

- **Sensitivity** is another term for TPR, referring to how well the test correctly detects stego images.

- **Specificity** is equal to 1-FPR, and refers to how well the test rejects clean images.

- The ROC curve given in figure 2.3 shows 20 different points corresponding to different threshold value. This was generated by examining the results of an attack on LSB embedding in the SVD domain described later.

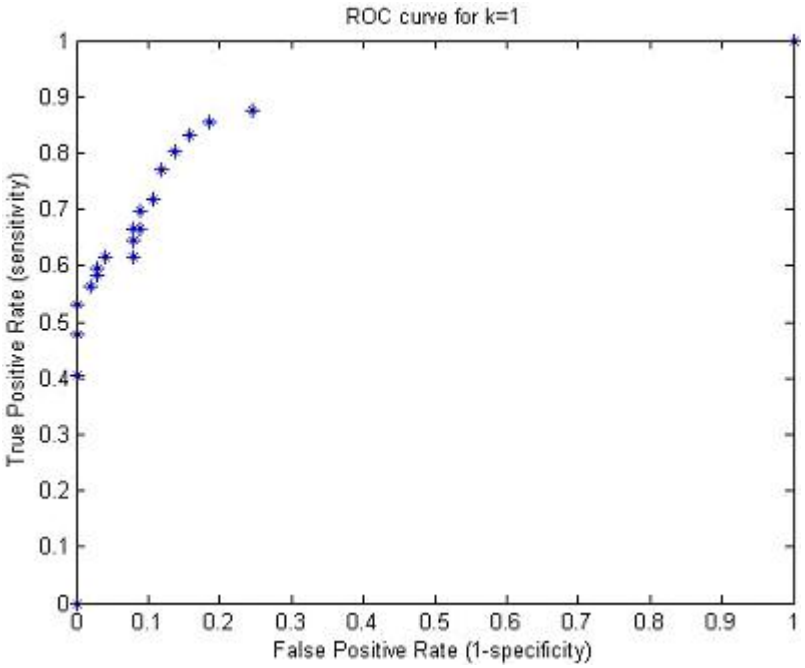

Figure 2.3 Sample ROC Curve

2.3.4 Error-Correcting Codes

- Error-correcting codes are used in signal processing to maintain the integrity of a message when a noisy transmission medium is used. This is done in a variety of ways, but for our purposes it involves taking a message word of length k and encoding it into another word of length $l > k$. The words in the encoded message space should be chosen with enough distance between them so that given a small number of errors, the intended word is still recoverable.

- For example, consider an original message with 4 words, 00, 01, 10, 11. Any error in a message will be construed to a different word. So, we encode this in the following way 00=000000, 01=011010, 10=100011, 11=111111. In this example, one error can be corrected all the time. Given the received message $M' = 010010$ and confidence that no more than one error occurred, we see that M' differs from 011010 by only 1 bit and thus the intended message $M = 011010$ which decodes to 01.

2.3.5 Stream Cipher

There are many different kinds of encryption available but due to the constraints of the SVD embedding algorithm, stream ciphers will be the encryption of choice.

- A cipher is a algorithm for performing encryption. Encryption consists of altering a message (plaintext) into an encrypted message (ciphertext) with the intent that anyone who views the ciphertext is unable to understand it without some secret information.

- A stream cipher is a specific cipher which takes as input a message and a secret key which is random or pseudorandom. A basic but secure stream cipher follows.
  Bit by bit, or letter by letter, it adds the message bit to the key bit and takes the remainder when divided by the alphabet size. That is, $c = (m + k) \mod |alphabet|$. The resulting ciphertext should appear random. Decryption is performed by subtracting the key from the ciphertext and taking the remainder when divided by the alphabet size. That is, $m = (c - k) \mod |alphabet|$. So, only another party who knows the secret key should be able to decipher the message. Sometimes, imperfections in pseudorandomly generated keys can cause weaknesses that cryptographers can exploit.

2.4 Watermarking

Watermarking refers to the practice of altering an object in some way with the intent of establishing ownership or authenticity of that object in the future. For our purposes, watermarking will refer specifically to digital watermarking, which is the technique of altering a computer file in some way with the intent of establishing the file's history, often motivated by copyright.

The manner of digital watermarking may vary. For example, a watermark might be a visible modification to a picture or video file that resembles a stamp of ownership, or an sequence of

bits added to the file which are imperceptible to visual inspection but are retrievable using a computer. In the latter case, watermarking can be considered a type of data hiding.

- **Robustness** is a watermarking method's resistance to various kinds of manipulation.
- Attacks on a watermark can be motivated by a desire to alter the image to destroy the watermark, or by a desire to replace the existing watermark with a new one.
  These attacks include, but are not limited to: compression, adding noise, applying filters, rotating the image, clipping the image.
- The general concept in digital image watermarking is to take an image, A, and a watermark, W, and combine them using some algorithm E to obtain a watermarked image $A_W = E(A, W)$.
- A watermarking scheme can be defined to be **invertible** if an attacker can take a given watermarked image A_W and create a fake original A_F and a fake watermark W_F such that $A_W = E(A_F, W_F)$. The attacker could then claim A_F was the original and claim ownership of A_W. If this is not possible, the scheme is called noninvertible.
  Informally, noninvertibility means that it is computationally infeasible for an attacker to produce a fake image and watermark such that when combined by a known watermarking algorithm the result is the watermarked image produced by the real owner.

CHAPTER 3. Previous work in data hiding with the singular value decomposition

The singular value decomposition has been applied previously in several ways related to data hiding. It was first applied to watermarking and later applied to steganography.

3.1 Singular value decomposition in watermarking

3.1.1 Liu and Tan

One of the prominent publications concerning the application of the singular value decomposition to watermarking was by Liu and Tan and dealt with a number of important concepts such as invertibility/noninvertibility, robustness to attacks and manipulation, and perceptibility of the watermark in their SVD-based watermarking algorithm.

In this algorithm, the singular value decomposition $A = U(\Sigma)V^T$ is taken of the entire image. The watermark, also taken as a matrix, W, is added to the Σ matrix in the singular value decomposition, with a scale factor a to vary the impact of the watermark on the image. The SVD is performed on the new matrix $\Sigma + aW = U_W \Sigma_W V_W^T$. Then the watermarked image $A_W = U(\Sigma_W)V^T$ is recomposed. By reversing the steps, the watermark can be extracted, given U_W, Σ, and V_W^T.

Liu and Tan state that while this algorithm is not strictly noninvertible, it does have resistance to counterfeiting. An attacker may be able to create a watermark and fake image such that $E(A_F, W_F) = A_W$, but it is computationally intractable to create such a W_F which is semantically meaningful. That is, if the original creator of the file chooses a meaningful watermark, e.g., a company logo or a copyright statement, he should be able to extract that, while any watermark that an attacker could produce would be meaningless noise.

This algorithm produces watermarked images which are resistant to many types of manipulation and attacks. Upon extraction, all of the semantically meaningful watermarks, in this case, a logo, were recognizable if somewhat blurred or marred.

The watermarked images are perceptibly different from the originals but the difference seems mainly to be an addition of noise or static to the image.

One potential attack on this algorithm could be to acquire a watermarked file, embed the attacker's own watermark in it using this same watermarking algorithm, and distribute many copies of this new file into the market. Now both attacker and owner have a claim on a file which exists in significant numbers; original ownership is now obfuscated.

3.1.2 Ganic, Zubair, and Eskicioglu

Ganic, Zubair, and Eskicioglu consider what Liu and Tan have done and attempt to improve their watermarking scheme by improving its robustness.

Their algorithm consists of two parts.

First, the $k \times k$ pixel cover image A is divided into square blocks of $n \times n$ pixels. The SVD of the $j \times j$ ($j \leq k$) watermark W is computed. The singular values are randomized and each singular value of the watermark is embedded into one $n \times n$ block of the cover image. Recompose this matrix for now as A_1. This has the effect of embedding some specific watermark data locally into each block.

The second layer of watermarking involves taking the SVD of A_1 and the SVD of the watermark W to obtain the singular values of A_1, λ_i, $i = 1, \ldots, k$, and the singular values of W, $\lambda_{w,i}$, $w = 1, \ldots, j; i = 1, \ldots, k$. New singular values are computed with another scaling factor, $\lambda_{2,i} = \lambda_i + \alpha\lambda_{w,i}$. Recombine U and V^T with a Σ_2 containing these singular values to finish with a two-layer watermarked image. This second layer has the effect of embedding watermark data globally throughout the entire image. Extraction of these watermarks is again straightforward, simply reversing the process.

The results of experimentation on this algorithm were even more compelling than the previous results. When subjected to several common manipulations (JPEG 2000 compression, Gaussian

blur, rescaling, cropping, rotation), the extracted watermark were still very recognizable as a meaningful image.

3.1.3 Ganic and Eskicioglu

This foray into watermarking by Ganic and Eskicioglu (2005) using SVD was significant because it combined two different decompositions which had been used before.
The discrete wavelet transform (DWT) decomposes a matrix into regions of information, HH, HL, LH, and a fourth region which itself may be divided up depending on what level of decomposition is desired. In the two level decomposition, seven regions are obtained, HH1, LH1, HL1, HH2, LH2, HL2 and LL2. Recorded in these regions are details about the matrix. The HH regions record high frequency data in both directions, HL regions record high frequency data in the horizontal direction while LH regions record high frequency data in the vertical direction. Embedding of watermark in this method involves

- Using DWT, decompose the cover image A into 4 subbands LL, HL, LH, HH.

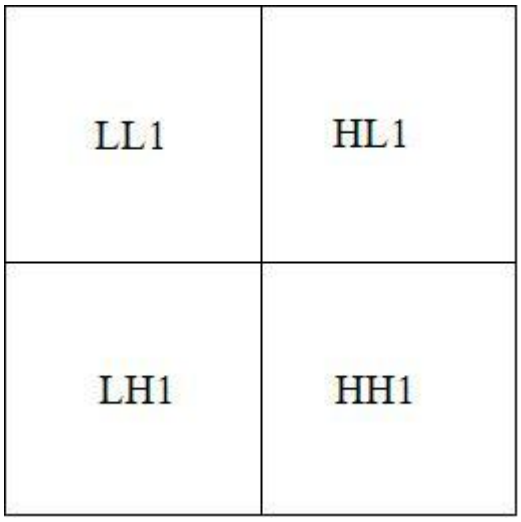

Figure 3.1 Discrete Wavelet Transform decomposition with one level

- Apply the SVD to each subband and note the singular values of each, $\lambda_{k,i}$, with $k = 1, 2, 3, 4$; and $i = 1, \ldots, n$ with k value denoting which subband.

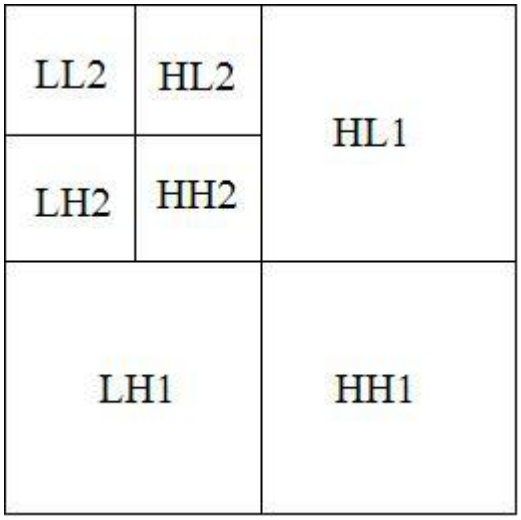

Figure 3.2 Discrete Wavelet Transform decomposition with two levels

- Apply the SVD to the watermark matrix, noting its the singular values, $\lambda_{w,i}$.
- Modify the singular values of the cover image in each subband by adding to them a multiple of the singular values of the watermark, $\lambda^*_{k,i} = \lambda_{k,i} + a\lambda w, i$ where a is the strength of the watermark, chosen based on keeping the watermark imperceptible as possible.
- Recombine each subband matrix by recomposing the each new singular matrix with the original U and V.
- Apply the inverse DWT using these new subband matrices.
- When extracted, 4 copies of the watermark image are obtained. Suppose some modification was applied to the image in an effort to destroy the watermark. Since the watermark image is stored in 4 different frequency domains, at least one copy of the watermark can survive any modifications that do not modify all the frequency domains. Thus, the watermark image becomes more resistant to a wider range of attacks.

  This hybrid method also results in a watermarked image that visually resembles the original very closely.

  As in the previous case, visual interpretation of the semantic value of the extracted

potential watermark is necessary.

3.2 Capacity of embedding in previous steganographic methods

3.2.1 Fridrich, Goljan and Du

Capacity of linear LSB embedding was addressed by Fridrich, Goljan, and Du (2001). Fridrich, Goljan, and Du (2001) have estimated that to avoid detection with the chi-square attack and other well known methods, embedding techniques which embed in the least significant bits must have an embedding rate of no more than 0.005 bits/byte, i.e., only one bit per 200 bytes. This does not offer a very high level of capacity. However, the singular value decomposition method can embed at a capacity of approximately 0.1 bits/byte, as reported by Bergman and Davidson (2005); however, this 0.1 bits/byte embedding rate may not be safe.

3.3 Singular value decomposition in steganography

Inspection of the previous work done in watermarking using singular value decomposition suggests that using the SVD for steganographic purposes may be successful.

3.3.1 Bergman and Davidson

Bergman and Davidson's algorithm computes the SVD of submatrices of an image file, then embeds a message in the U matrix. By making changes in the SVD domain, this algorithm may be able to defeat some of the steganalytic statistical attacks which analyze the pixel values directly. This is because while the algorithm embeds into the U matrix in a predictable way, this will have an unpredictable effect once the $A = U\Sigma V^T$ is recomposed.

3.3.1.1 Description of the Algortithm

Begin with a message payload encrypted into 0s and 1s, and a cover image in which in which to embed. Apply the singular value decomposition to $n \times n$ pixel blocks of this image. If the image dimensions are not divisible by n, do not embed in the extra bits. In practice, as

found by Bergman and Davidson (2005), $n = 8$ is a good balance of embedding capacity and imperceptibility.

Treat each of these $n \times n$ blocks as a matrix and, considering each block separately, apply the singular value decomposition, obtaining its representation $A = U\Sigma V^T$. This is where the standard singular value decomposition ends and the steganographic portion begins. Next, divide U into three sections.

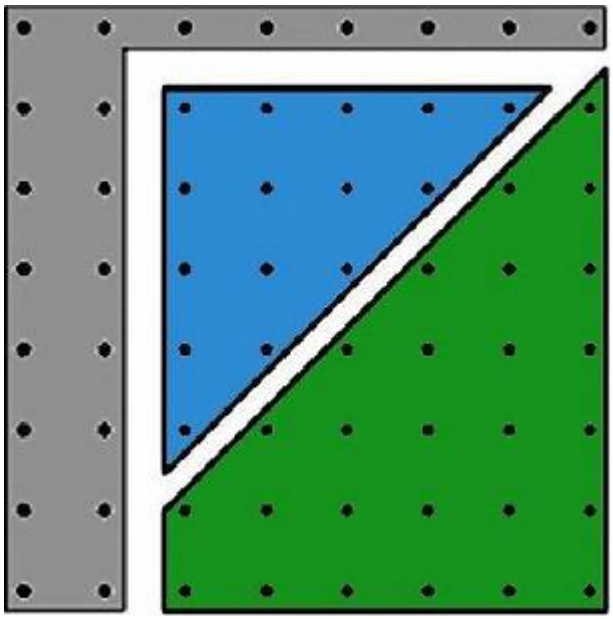

Figure 3.3 Division of an 8×8 block with 2 protected columns for use in SVD embedding

The first section, which consists of the top row and leftmost m columns, are not modified. The leftmost singular vectors correspond to the singular values with highest magnitude and thus the most perceptible impact on the resulting image; so to keep changes imperceptible, the m leftmost columns are unmodified. The choice of m depends on the tradeoff between perceptibility, and data capacity. In practice, a choice of $m = 2$ works well, as tested by Bergman and Davidson (2005).

The first row of U is unmodified to keep all its entries lexicographically positive and thus maintain a normal singular value decomposition.

Also, to maintain orthogonality of the singular vectors, we must divide the remaining

rectangle into our remaining two sections. First, the "middle triangle" is where we may actually embed data. For $m = 2$ and $n = 8$, this will consist of 15 entries. The message is stored in these entries by altering the +/- sign of the entry as necessary to record the message. The third section, the "lower right trapezoid" is reserved for the changes necessary to keep the new matrix orthogonal. Note, changing the sign of any entry will not change the contribution that entry makes to the norm of the vector. For each subsequent column, another variable must be dedicated to maintaining orthonormality. For the i^{th} column of the triangle, we have $n - i - 2$ degrees of freedom. The remaining i entries of the i^{th} vector must be altered to make the i^{th} vector orthonormal with respect to the previously modified vectors. Once these changes are made, we have our new U'. Then the matrix is recomposed to achieve $A' = U'\Sigma V^T$, and this A' is stored in the image.

Some rounding and clipping will be necessary to keep the values of A' between 0 and 255 and to make the values integers. This step is a source of errors during message extraction. Methods of handling these errors include error-correcting codes, iteration of embedding, and instituting small perturbations in the matrix. These are addressed later in this section.

Any time two or more eigenvalues of a given block have distinct but similar value, their associated eigenvectors can be volatile. That is, small changes can cause them to vary wildly. This results in message errors when embedding, so to combat this, the basic embedding algorithm has an additional step which spaces apart the singular values of the Σ matrix of the SVD.

It has been observed that the message errors instituted by clipping and rounding can be eliminated be repeating the embedding on that same block. After being altered repeated times, a block will evolve into a more stable state that holds the message. No proof is given that iterating the embedding process will converge to the desired error-free matrix, but experimental evidence suggests it does result in a reduction of errors.

In practice, one can decide on an acceptable number of errors to allow per block, embed, extract, and check the number of errors, and iterate until this error threshold is achieved. The choice of this error threshold will depend on the type of error-correcting code used.

Sometimes iterating will not eliminate enough errors. An iteration threshold, i.e., an upper bound on the iterations performed to fix errors, should also be used to avoid falling into an infinite loop where the error threshold can not be met. Necessary thresholds are discussed in chapter 4.

Since iteration alone is not enough, error-correcting code is used. The BCH code has a good capacity and error-correcting capability for our purposes, and has parameters that fit our needs. For example, the BCH (15,7,5) code, for example, breaks the message into words of length 7, codes them into words of length 15, and can correct 2 errors per word. For a block size $n = 8$ and columns $m = 2$ protected, there are 15 embeddable entries, so a BCH (15,x,y) would work well.

Now, since the code words of the BCH code are repeatable and have distinct structure, embedding these words directly in inadvisable as this structure could betray the presence of a message. So, a stream cipher is implemented as well. It is important to implement the stream cipher after the error-correcting code is applied to the message.

Bergman and Davidson found that their best balance between message capacity and error rate came with a block size $n = 8$ and with $m = 2$ columns protected. Their resultant message capacity with these parameters was 0.23 message bits per byte or pixel. Considering also the error correcting code with either 7 message bits per block or 5 message bits per block makes this capacity $0.23 \times \frac{7}{15} = 0.14$ or $0.23 \times \frac{5}{15} = 0.08$ respectively. Each of these is significantly higher than the safe embedding capacity for least significant bit embedding reported by Fridrich, Goljan, and Du (2001).

The experimental error rate was found by Bergman and Davidson (2005) to be approximately 0.0006 after using the BCH (15,7,5) error-correcting code. This was considered to be satisfactory.

One aspect of this embedding algorithm that was not explored was the detectability. Given that this algorithm alters positive or negative parity in the SVD domain, an attack that investigates this domain for statistical presence of changes may be successful. Such an attack is the subject of chapter 4.

CHAPTER 4. Analysis of the singular value decomposition data hiding algorithm

In order to test the detectability of Bergman and Davidson's SVD embedding algorithm, I apply two tests. One is the chi-square test on the pixels of an image. This will test if the SVD embedding algorithm does defeat a known statistical attack meant to find embedding on the pixels directly, namely LSB embedding.

The second test will decompose an image according to the SVD embedding algorithm, extract a potential message, then perform the chi-square attack on that data.

4.1 Chi-Square attack on the Spatial Domain

To test the detectability of the SVD algorithm by a standard chi-square attack, messages were embedded using the SVD embedding algorithm in 100 pgm format images that were known to be clean. Then the standard spatial chi-square attack was performed on the clean images and the stego images. The ROC curve generated follows as figure 4.1 and shows that the embedded images do not betray their embedded status to the chi-square attack in a traditional sense. However, some sort of vulnerability may exist, as we see that as the thresholds increase, the false positive rate increases (as expected) but the true positive rate does not, which would imply that the embedded images' values become more skewed to either even or odd than a normal clean image's values would.

In fact, it is likely that this unexpected horizontal line behavior in the ROC curve is a result of the clipping step of the SVD embedding algorithm. Recall that the embedding process, altering values in U, then recombining $U'\Sigma V^T = A'$ may result in the values of A' being outside the range of 0 to 255, so they were clipped up to 0 or down to 255. This means that

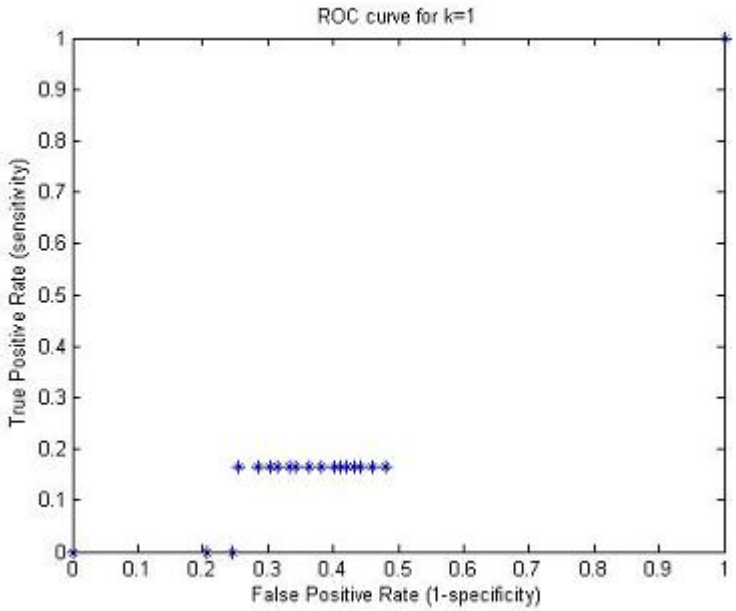

Figure 4.1 ROC curve for spatial chi-square attack

even if the values that lay above 255 and below 0 were evenly distributed as they would be in an embedded image, they now all have been pushed down to 255, making a large discrepancy in the 254-255 bin value and the 0-1 bin value. This will indicate to the test that there is a larger amount of variation in the image than would be consistent with an embedded image; therefore, many of the true positives are mistaken for clean images. In fact, this variation is higher than the variation occuring in a normal clean image, and thus embedded images are interpretted to be clean by a traditional interpretation of this test.

Ironically, to the perspective of the chi-square test, these images now hide too well and thus may stick out; a new test may be able to be developed that looks for a large number of 255 pixel values and 0 values corresponding to clipping. A similar property, however, may be a result of increasing the contrast of an image; so, this test may not be reliable.

One solution to this clipping problem is given in chapter 5.

4.2 Chi-square test on the SVD domain

To test the detectability of the SVD algorithm by a standard chi-square attack, messages were once again embedded using the SVD embedding algorithm in 100 .pgm images that were known to be clean. Next, a potential message is extracted by examining the positive/negative parity of the appropriate entries of the U matrix of the SVD. This time, in the implementation of the chi-square attack on this potential messsage, there arose a problem.

Typically, the chi-square attack groups together and counts pixels in an image with pairs of values, and these pixels take on many different values between 0 and 255. In this case, however, the embedding didn't take place in least significant bits but rather in the signs of certain singular values. So, any chi-square attack should then look at these signs for a potential message. This only created two values, 0 and 1 (corresponding to -1 and 1, extracted from the U matrix of the SVD) in our potential message.

Given this setup, there were only two bins of nonzero size, or one nonzero value of n_i, to consider, which caused the standard chi-square algorithm no end of trouble. Two ways of modifying the chi-square attack were implemented.

One method was to spread out the extracted potential message into different pairs of bins. A base matrix was created for each iteration of chi-square, which consisted of a "staircase" matrix. In the iteration that inspects the first n bits of the extracted image, I would add to that $n \times 1$ matrix a new matrix whose values were 0 for the first n/128 entries, then 2 for the next n/128 entries, and so on, boosting up the extracted image by even-height steps. This had the effect of spreading out the extracted image values so that the chi-square attack would have a chance to fill out its other bins, otherwise seen as making more non-zero n_i values.

The effectiveness of this process is shown in figure 4.2. The best threshold choice was for T=30% which resulted in 77% TPR and 12% FPR.

This is, of course, just one of several solutions; one could not only spread these values out over 128 steps, but also choose fewer or greater steps. It is not clear what choice of step number maximizes the sensitivity, but a search for maximal sensitivity could be performed.

Another method of altering the chi-square attack on this potential message would be to consider

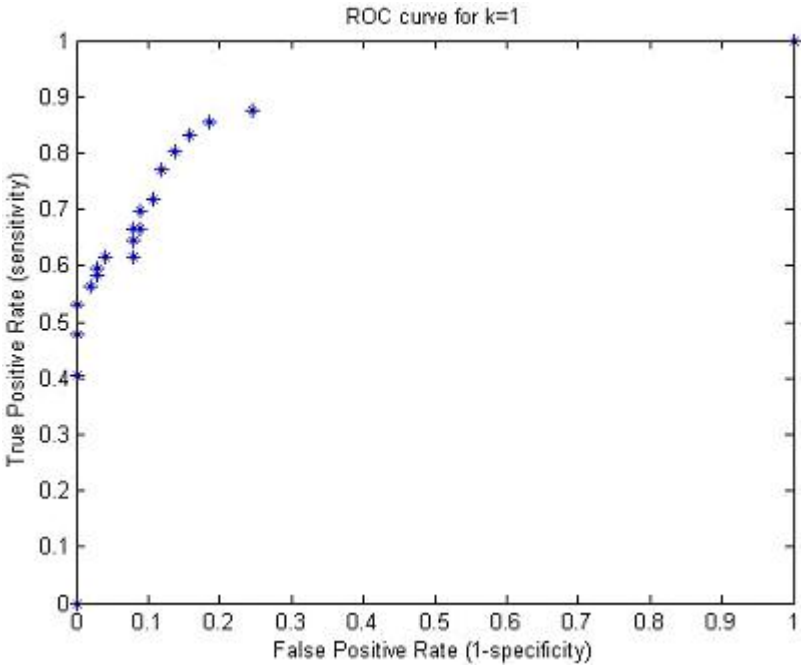

Figure 4.2 ROC curve for first SVD domain chi-square attack

instead of bins for pairs of values 0-1, 2-3, ..., 254-255, bins for each entry of the U matrix of the SVD. That is, each entry in the first embeddable position of U is stored in the first bin and so on. This results in a lower number of bins, but has the possible advantage that each bin contains data which comes spatially from all over the image and thus from all over the potential message. In practice, this method was able to detect embedding very well, as shown in figure 4.3. A threshold of T=5% in this case had a TPR of 100% and an FPR of 8%.

4.3 Testing the error-correcting code

Testing the BCH(15,7,5) code resulted in an error rate of less than 1%, but required an iteration threshold of 120 to achieve this. Recall, BCH(15,7,5) had an error-correction capacity of 2 errors/block so the error threshold was set to 2.

Testing the BCH(15,5,7) code, which had an error-correction capacity of 3 errors/block, resulted in an error rate of 0% with an iteration threshold of only 40.

An interesting effect of the error-correcting code was that it improved the effectiveness of the SVD domain chi-square attacks. It seems that the errors that were occurring in the extracted

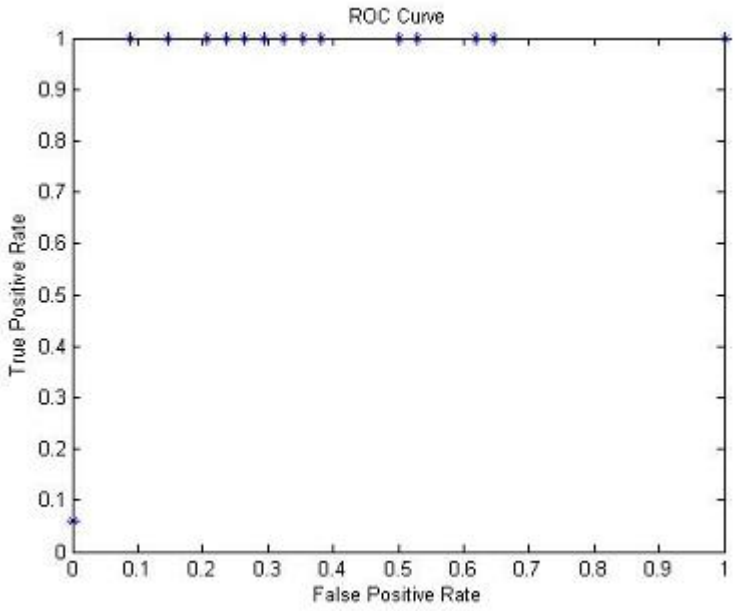

Figure 4.3 ROC curve for second SVD domain chi-square attack

message were enough to deviate from the uniformity of the embedded message and camouflage the stego images from the chi-square attack. With the implementation of the error-correcting code, these errors are eliminated and the uniform randomness of the embedded message is maintained, thus allowing the chi-square attack to increase its precision.

The difference in embedding capacity between the BCH(15,7,5) and BCH(15,5,7) is .14 bits/byte vs. .08 bits/byte respectively. These are both significantly higher than previous safe embedding rates with LSB embedding techniques, so either would be a fine choice.

CHAPTER 5. Conclusions, Summary and Future Work

- SVD Embedding Avoids Traditional Spatial Chi-Square Detection

  This series of experiments verifies that a standard chi-square test will not reliably identify an image which has a message embedded using the SVD algorithm, but the test results are certainly unexpected enough to create ample suspicion of embedding by looking for unusually high rates of pixels with value 0 or 255.

  Since this method has a higher embedding rate which is resistant to normal LSB plane attacks than LSB embedding techniques (Fridrich, Goljan, and Du (2001)), and it avoids detection by a standard steganalysis technique that LSB embedding fails, this is an encouraging result for the future of the SVD method because it is at least as undetectable as simple LSB embedding.

  This testing suggests that to avoid the suspicious property of embedded images of uncharacteristically many 0's and 255's, the clipping step could alternate between clipping values which were above 255 down to 254 and 255 and alternating clipping values below 0 up to 0 and 1. This would eliminate the overabundance of 0s and 255s, but the uniformity could then cause a problem.

- SVD Embedding Is Vulnerable to Singular Value Chi-Square Detection

  These experiments also would suggest that if a steganalyzing adversary knew of this SVD embedding technique and was actively using this attack on each image she encountered, she would have a high probability to correctly discern whether or not there was an embedded message by this implementation of the chi-square attack. However, it should be noted that performing the attack; i.e., extracting each potential message and then applying the chi-square test to it, is a relatively time-intensive task, which lends to the

difficulty of defeating this embedding method.

So, despite the fact that SVD embedding may avoid detection by a standard attack, once a potential steganalyst knows of the embedding technique and can design an attack based on that knowledge, the technique is not secure. This is consistent with the cryptographic principle that the security of a method should not rely on the secrecy of the method.

There are techniques to modify LSB embedding to avoid chi-square test detection. These are based around modifying the embedded message to disguise its statistical properties to more closely resemble a clean image. The results from testing would suggest that any of these techniques could be implemented to avoid the chi-square based SVD attack.

- SVD Embedding Has a Reasonably High Capacity

  This capcacity given in the Bergman and Davidson (2005) paper and has been corroborated. Even when implementing error correcting code a significantly higher capacity is maintained at a capacity of $\frac{15}{64} \times \frac{7}{15} = 0.14$ or $\frac{15}{64} \times \frac{5}{15} = 0.08$ when implementing BCH(15, 7, 5) or BCH(15, 5, 7) codes respectively on block size $n = 8$ and $m = 2$ protected singular values.

- SVD Embedding has robustness to minor changes.

  Alterations which do not change the signs of the entries of U, rather only their magnitude, preserve the message. The error correcting code helps in this endeavor, which may be another reason to choose BCH(15, 5, 7) over BCH(15, 7, 5).

  Further study in robustness may be to investigate how common alterations such as .jpeg encoding, blur, rotation, and cropping may affect the singular value decomposition of an image. The previous work done in testing watermarking robustness may assist these tests.

- SVD Embedding Can Be Done In A Reasonable Amount of Time

  Time complexity of embedding may be taken in relation to the image of size $|i|$ pixels in which embedding will occur or in relation to the message of size $|M|$ intended to be delivered. Since these are related based on block size, number of protected singular val-

ues, and type of error-correcting code, the choice is arbitrary; for now consider $|M|$. As always, perceptibility and detectability should be given priority over speed in most cases. For a fixed block size n, computation of the SVD is has an upper limit of operations, so embedding time increases linearly with image size, that is, $O(|M|)$. Computation of SVD for various values of n seems to increase as $O(n^2)$ since computation involves the Gram-Schmidt process to calculate singular vectors which involves $O(n^2)$ steps. This should, however, be balanced somewhat by having to calculate the SVD for fewer blocks. As long as the limited number of iterations of embedding is implemented, this is still $O(|M|)$.

Making the changes to each block is linear with message length, so takes $O(|M|)$ time.

The error-correcting code is implemented in Matlab by a look-up table, which should have a fixed time for each message block independent of image size or message length, so is also $O(|M|)$.

The stream cipher should also have a $O(|M|)$ time complexity based on the message length, although if a more complex stream cipher is chosen this will increase.

So, with fixed block size n and protected singular values m, total computational complexity should be $O(|M|)$. The coefficients of p are quite sizable for the necessary look-up time for the error-correcting code and also for computing the SVD, so this is not a fast algorithm, but at least one whose time complexity doesn't increase unreasonably fast with increase of message length.

Performing the algorithm with a laptop computer is reasonable; implementation by hand is not tractable.

- SVD Embedding May or May Not Be Visually Perceptible

  Bergman and Davidson's tests seemed to generate embedded images which had very little perceptible change. In this round of testing, there was visual static and blurring of some detail, usually around high frequency areas like edges of objects in the picture. For example, see figure 5.1.

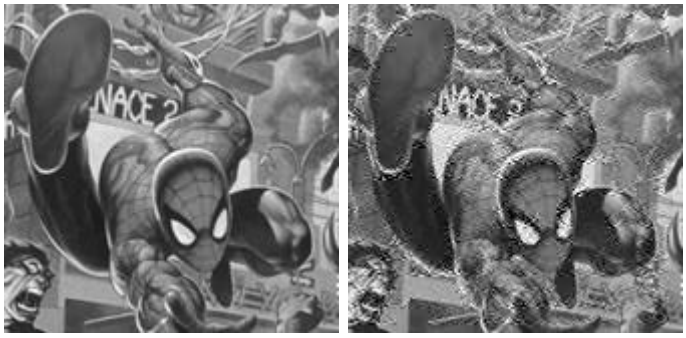

Figure 5.1 A clean image and a stego image embedded with the SVD embedding algorithm. Some visually perceptible distortion is present

Of course, without the original for reference, the distortion in the stego file may be insignificant enough to pass. Since visual distortion depends greatly on the choices of block size and protected singular values, more testing to optimize these choices could be done next.

It is informative to note the difference between the intents of watermarking and steganography and how these intents alter the ultimate effectiveness of the SVD. Watermarking seeks to hide data in a way that is not immediately visible, is resistant to any attacks, and is retrievable in a meaningful fashion, that is, the original watermarker can prove to outside parties that the file was indeed his creation originally.

Steganography wishes to hide a message in a file that is undetectable not only to a passive observer but also to an actively examining adversary. It seems that to obtain this goal, some robustness may need to be lost, in comparison to watermarking.

APPENDIX A. Matlab Code

All computations were performed using Matlab 7.0. My code is available for LSB embedding in spatial and SVD domains, the Chi-square attack on these domains, and a test of the error rate using various error correcting codes. This code is too extensive to include here, so is available by emailing ericthansen@gmail.com .

BIBLIOGRAPHY

Ruizhen Liu and Tieniu Tan. "An SVD-Based Watermarking Scheme for Protecting Rightful Ownership." *IEEE Transactions on Multimedia, Vol. 4, No.1, March 2002*

Emir Ganic, Nasir Zubair and Ahmet M. Eskicioglu. "An Optimal Watermarking Scheme Based on Singular Value Decomposition." *IASTED International Conference on Communication, Network, and Information Security (CNIS 2003)*

Emir Ganic and Ahmet M. Eskicioglu. "Robust Embedding of Visual Watermarks Using DWT-SVD" *Journal of Electronic Imaging, October-December (2005)*

Clifford Bergman and Jennifer Davidson. "Unitary Embedding for Data Hiding with the SVD." *Security, Steganography, and Watermarking of Multimedia Contents VII, SPIE Vol. 5681 (2005)*

Leslie Hogben, Richard Brualdi, Anne Greenbaum, et al. *Handbook of Linear Algebra.* Chapman & Hall/CRC, 2006.

Andreas Westfeld and Andreas Pfitzmann. "Attacks on Steganographic Systems." *3rd International Workshop on Information Hiding*

Aleka McAdams and Tracy McKay. "Using ROC Curves to Automate the Chi-Square Attack for Steganalysis". Unpublished. *Iowa State University REU 2005.*

J. Fridrich, M. Goljan, and R. Du. "Reliable detection of LSB steganography in grayscale and color images." *Proc. ACM Special Session on Multimedia Security and Watermarking 2001*