2009

Feature selection, statistical modeling and its applications to universal JPEG steganalyzer

Jaikishan Jalan
Iowa State University

Feature selection, statistical modeling and its applications to universal JPEG steganalyzer

by

Jaikishan Jalan

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Iowa State University

Ames, Iowa

2009

DEDICATION

To

my wife

&

family

TABLE OF CONTENTS

LIST OF TABLES

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 and foremost, Dr. Jennifer Davidson for her guidance, patience and support throughout this research and the writing of this thesis. Her insights, active research meetings and words of encouragement have often inspired me and renewed my hopes for completing my graduate education. I would also like to thank my committee members Dr. Hridesh Rajan and Dr. Clifford Bergman, for their insightful comments and encouragement.

This work was possible due to funding by the National Institute of Justice, through the Midwest Forensics Resource Center at Ames Laboratory under Interagency Agreement number 2008-DN-R-038. The Ames Laboratory is operated for the U.S. Department of Energy by Iowa State University, under contract No. DE-AC02-07CH11358. I would also like to thank Gerard Meyer for agreeing to beta test our software and provide non-monetary support in the goal of fulfilling the objective of this research. I would also like to thank Dr. Jessica Fridrich and Dr. Yun Q. Shi for providing code for the steganalyzers used in this thesis. Special thanks to Dr. Gwaneal Doerr for providing a rich database of JPEG image which helped in this research. Last, but not the least, I would like to acknowledge the support of John Dickerson and College of Engineering, Iowa State University for granting access to Matlab grid which saved a lot of computational time in our experiments.

ABSTRACT

Steganalysis deals with identifying the instances of medium(s) which carry a message for communication by concealing their exisitence. This research focuses on steganalysis of JPEG images, because of its ubiquitous nature and low bandwidth requirement for storage and transmission.

JPEG image steganalysis is generally addressed by representing an image with lower-dimensional features such as statistical properties, and then training a classifier on the feature set to differentiate between an *innocent* and *stego* image. Our approach is two fold: first, we propose a new feature reduction technique by applying Mahalanobis distance to rank the features for steganalysis. Many successful steganalysis algorithms use a large number of features relative to the size of the training set and suffer from a "curse of dimensionality": large number of feature values relative to training data size. We apply this technique to state-of-the-art steganalyzer proposed by Tomás Pevný (54) to understand the feature space complexity and effectiveness of features for steganalysis. We show that using our approach, reduced-feature steganalyzers can be obtained that perform as well as the original steganalyzer. Based on our experimental observation, we then propose a new modeling technique for steganalysis by developing a Partially Ordered Markov Model (POMM) (23) to JPEG images and use its properties to train a Support Vector Machine. POMM generalizes the concept of local neighborhood directionality by using a partial order underlying the pixel locations. We show that the proposed steganalyzer outperforms a state-of-the-art steganalyzer by testing our approach with many different image databases, having a total of 20000 images. Finally, we provide a software package with a Graphical User Interface that has been developed to make this research accessible to local state forensic departments.

CHAPTER 1 INTRODUCTION TO STEGANOGRAPHY AND STEGANALYSIS

Steganography is the practice of communicating a hidden message in such a way that no one, apart from the sender and intended recipient, suspects the existence of the message. The goal of steganography is to embed a *payload* into a *cover* object to obtain a *stego* object in such a way that the presence of hidden information cannot be detected by either perceptual or statistical analysis of the stego object. The counterpart of steganography is steganalysis. The main goal of steganalysis is to identify whether a given object has a payload embedded in it. Other information about the payload is often sought, including identification of the steganography algorithm, estimation of payload length, recovery of the payload, or obliteration of the payload. If there exists an algorithm that can determine whether or not a given image contains a secret message with a success rate better than random guessing, the steganographic scheme is considered to be broken. A more detailed introduction to steganography and steganalysis can be found in (10; 14; 21) .

Steganalysis can be broadly classified into two categories: *passive* or *active*. Passive steganalysis is concerned with detecting the presence or absence of hidden messages in a stego signal and identifying the stego embedding algorithm. On the other hand, active steganalysis addresses further issues such as estimating the embedded message length, locating the hidden message, finding the secret key used in embedding, identifying some parameters of the stego embedding algorithm, or extracting the hidden message itself, which is often the ultimate goal. Attacking steganography can also be classified as *targeted* and *blind* steganalysis. In targeted steganalysis, known embedding signatures, such as characteristic histogram shapes, are exploited to create specific feature values that can distinguish between stego and cover images. Blind steganalysis uses a set of generic feature values that model image statistics so as to distinguish between cover and stego images. Blind methods can be used on a variety of steganographic algorithms and do not target a specific algorithm. Recent advances in steganalysis allow for some blind detection algorithms to be almost as accurate as targeted detection algorithms.

With the advent of digital media and the Internet, multimedia objects such as still images and videos have become popular and are shared easily. Image and video data make a good choice for hiding payload. These objects are readily available and their broad presence on the Internet makes it difficult to check each one for hidden payload and thus difficult to detect the use of steganography. A single image can hold a reasonable amount of information, and a video file can hold more. In addition, there is a plethora of freeware available for hiding secret information, as can be seen by visiting the site stegoarchive.com (6). MSU StegoVideo is a public video steganographic tool; see (7). In this research, we restrict steganalysis of image data to Joint Photographic Experts Group (JPEG) format because of its wide use in consumer cameras and on the Internet. It also has the advantage of low bandwidth for storage and transmission, unlike raw or other uncompressed formats.

Our objective is to develop a passive blind steganalysis system which offers increased potential for classifying unknown embedding algorithms. Blind JPEG image steganalysis is generally addressed by representing an image with lower-dimensional features such as statistical properties, and then training a classifier on the feature set to differentiate between an innocent and stego image. To tackle this problem, we first analyze existing state-of-the-art steganalysis systems. In Chapter 2, we presents a filter-type feature selection algorithm that selects a reduced feature set using the Mahalanobis distance measure. It provides one solution to the problem of computational complexity, without loss of accuracy in detection. The experiment is applied to a well-known JPEG steganalyzer, and shows that using our approach, a reduced-feature steganalyzer can be obtained that performs as well as the original steganalyzer. The steganalyzer is that of Pevný et al. (54) that combines DCT-based feature values and calibrated Markov features. Our results demonstrate that as few as 10-50 features can be used to create a classifier that gives comparable results with the full suite of 274 features. This further helps us to understand which features are most useful for steganalysis. A manuscript describing the reduced feature steganalyzer has been accepted to the SPIE Media Forensics and Security XII, 2010 in San Jose, USA and Dr. Jennifer Davidson will be presenting the results at the conference.

In Chapter 3, we propose a new modeling technique by developing a Partially Ordered Markov Model (POMM) to characterize JPEG images. Our rationale to do so is based on our experimental observation from Chapter 2, which shows that the Markov based features used by (54) contribute significantly to the detection process. A POMM generalizes the concept of local neighborhood directionality by using a partial order underlying the pixel locations. It has been shown that this property results in a computational advantage of POMMS over Markov Random Fields (MRF) in two ways (41): whenever the normalizing constant needs to be calculated, such as in determining the joint probability distribution function (pdf), the joint pdf of a POMM is available in closed form; and the normalizing constant for a

POMM is always known and equal to the value one (24). We also show that our steganalyzer can beat state-of-the-art on four different databases, namely BOWS2 (67), Camera, Corel and NRCS (25) with more than 20,000 images.

In Chapter 4, we present Canvass, a software package that has been developed in Java to make this research accessible to the Iowa Department of Criminal Investigation forensic lab. This is followed by conclusions and suggestions for future work.

The rest of the chapter is as follows. We discuss the nature of JPEG images since we restrict steganalysis to JPEG images, and provide necessary details to the reader concerning the JPEG format. We then introduce a simple case of steganography in JPEG images followed by a popular steganographic algorithm, which we use later to test our proposed steganalyzer. We end this chapter by a small discussion on state-of-the art steganalyzers, some of which are used for comparison with the proposed steganalyzer.

1.1 JPEG Image Format

A digital image can be viewed as a spatial, multivariate array of pixels (picture elements). Suppose a generic pixel location is written as s, a vector in $\mathbb{R}^2$. The quantity $g(s)$ denotes the pixel value, such as the intensity of radiation in a band of the electromagnetic spectrum, at pixel location s. Then we write an image as

$$g \equiv \{g(s) : s \in D\}$$

where D is the index set of pixel locations. This set is typically finite with regular spacing, so that, without loss of generality, we assume

$$D = \{(x, y) : x = 1, ..., M; y = 1, ..., N\}$$

Thus, an image is an MxN rectangular array of pixel values. In an 8 bit image, $g(s) \in \{0, 1, ..., 255\}$. The Joint Photographic Experts Group (JPEG) format stores image data in a lossy compressed state as quantized frequency coefficients. Image files that employ JPEG compression are commonly called "JPEG files". Figure 1.1 shows the compressing steps performed. The JPEG compressor first partitions the uncompressed image into sets of 8 by 8 pixels. The discrete cosine transformation (DCT) transforms the 8x8 spatial brightness values into 8x8 frequency coefficients (real numbers) as follows:

$$G_{u,v} = \alpha(u)\alpha(v)\sum_{x=0}^{7}\sum_{y=0}^{7} g_{x,y} cos\left[\frac{\pi}{8}\left(x+\frac{1}{2}\right)u\right] cos\left[\frac{\pi}{8}\left(y+\frac{1}{2}\right)v\right]$$

where u is the horizontal spatial frequency, for the integers $0 \leq u \leq 7$; v is the vertical spatial frequency, for the integers $0 \leq v \leq 7$; $g_{x,y}$ is the spatial pixel value at coordinates (x, y); $G_{u,v}$ is the DCT coefficient at coordinates (u, v) and α is a normalizing function given as

$$\alpha(n) = \begin{cases} \sqrt{\frac{1}{8}} & \text{if } n = 0, \\ \sqrt{\frac{2}{8}} & \text{otherwise.} \end{cases}$$

After applying the DCT, a quality matrix Q is used to quantize the frequency coefficients to integers in the range -1024 to 1023. This step loses information. The quantized DCT coefficients are computed as

$$B_{j,k} = \text{round}\left(\frac{G_{j,k}}{Q_{j,k}}\right)$$

where $0 \leq j, k \leq 7$ and B is the set of quantized DCT coefficients. After lossy quantization, the Huffman coding ensures the lossless coding of the quantized coefficients. A more detailed description of the JPEG compression can be found in (11).

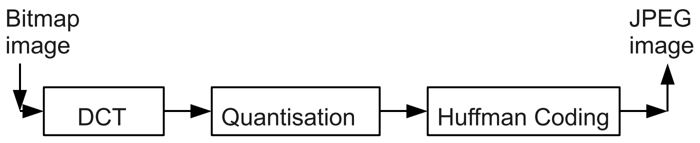


Figure 1.1: JPEG compression steps

Note that the correlations among DCT coefficients can be exploited in two following ways:

- **Intra block correlations**: It capture correlations that occur with each 8x8 block of quantized DCT coefficients.
- **Inter block correlations**: It capture correlations that occur between DCT coefficients in different blocks but the same (or close to the same) relative positions.

1.2 JPEG Steganographic Algorithms

A steganography algorithm embeds a payload into a cover object in such a way that the presence of hidden information cannot be detected by either perceptual or statistical analysis of the stego object. A payload consists of a vector of uniformly randomly generated bits 0 and 1, representing an encrypted bitstream. A JPEG steganography algorithm embeds payload by changing the quantized DCT coefficient values B. The most popular, frequently used and easy to implement steganographic method is the

Least Significant Bit (LSB) steganography. The LSB steganographic methods can be classified into the following two categories: LSB replacement; and LSB matching, also called plus/minus one embedding (50). The LSB replacement method works by replacing the least significant bit of a DCT coefficient value with the payload bit. In LSB matching, a coefficient value is modified as needed by increasing or decreasing the base 10 coefficients randomly to match the payload bit. If the bit must change, a value of +1 or -1 is added randomly to make the DCT bit value match that of the payload's. This seemingly innocent modification of the LSB embedding is significantly harder to detect, because changed pixel values are no longer paired as in LSB replacement.

Jsteg (4) is probably the first steganographic tool to embed in JPEG images. It was developed by Derek Upham in 1993. This hiding algorithm embeds a payload bit by replacing the LSB of the quantized DCT coefficients and skipping all those coefficients whose value is 0 or 1. It can embed data roughly 12% of file size of cover image, and embedding is performed in a sequential order on the coefficients.

OutGuess (56) was developed by Neil Provos in 2001. The algorithm identifies redundant coefficients that have the least effect on the cover image and modifies them if necessary during embedding of message bit using LSB replacement. For each bit changed due to payload embedding, the algorithm changes another untouched coefficient. The original global histogram of JPEG coefficients is preserved after embedding is complete. The algorithm voluntarily limits the maximum size of data that can be embedded to 6% of file size of cover image in order to make the algorithm more robust to statistical analysis.

The F5 (68) steganographic algorithm was introduced by Pfitzmann and Westfeld in 2001. It was developed from Jsteg in an iterative fashion with each new version being less detectable. It embeds message bits into randomly chosen DCT coefficients. If the payload is small enough, it uses "matrix embedding", an approach based on coding theory that minimizes the necessary number of changes to embed a message of certain length. The message length and the number of non-zero non-DC coefficients are used to determine the best matrix embedding that minimizes the number of modifications of the cover image. For payload too large for matrix embedding, it decrements the absolute value of DCT coefficients if LSB of the DCT coefficient and payload bit does not match. The author claims that its maximum steganographic content size is roughly 13% of size of cover image.

Steghide (2) uses a graph-theoretic approach to steganography. At first, the secret message is compressed and encrypted into the payload bitstream having n bits. Then a random site visitation sequence is generated in the DCT domain. The n quantized DCT coefficient values (cover values) from those locations are listed, and paired with payload bits. The pixel locations $(u_i,\ v_i)$ $1 \leq i \leq n$ for each cover value are retained along with the coefficient value. For each payload bit i, $1 \leq i \leq n$, the corresponding cover value bit is inspected. If the payload bit value matches the corresponding cover value bit, then nothing is done. If payload bit i does not match the corresponding cover bit, then a graph-theoretic search algorithm is used to find a different cover value, say at position j in the original list of DCT values, whose bit matches the payload bit. The DCT values at the two sites are switched so that the DCT value at position i with pixel location $(u_i,\ v_i)$ is now at location j $(u_j,\ v_j)$ and vice versa, and the bit values of the newly located cover values now match with the corresponding payload bits at each location i and j in the payload list. This is done for all such possible pairs until all the payload bits match with switched cover bits, or until it is not possible to make any more exchanges. If there are any remaining payload bits not embedded, that is, not having bits that correspond to the cover value bits in the list, then those remaining cover value bits are used to embed those payload bits by LSB replacement (overwritten).

JPHide (3) uses DCT coefficients based on a fixed table such that coefficients with higher numerical value are selected first. All coefficients in the current class are used first to hide the information before the next class is chosed. The data hiding process continues in the current coefficient class even after the complete message has been hidden. A pseudo-random number generator determines if coefficients are skipped. The probability of skipping bits depends on the length of the hidden message and how many bits have been embedded already. An interesting property of JPHide is that it not only modifies the least-significant bits of the DCT coefficients, it can also switch to a mode where the second-least-significant bits are modified.

1.3 Universal JPEG Steganalyzers

Steganalysis can be considered as a two-class pattern classification problem if the test image needs to be classified as either a cover image or a stego image. Generally, the classification consists of two parts, feature extraction and pattern classification. Since image data is typically very large, a lower-dimensional representation of the information in the image, relative to the classification task at hand, is required. A feature is such a lower-dimensional representation of the image data and is crucial for many classification problems, including steganalysis. The best features for steganalysis should contain information about the changes incurred by data hiding rather than by the content of the image.

Calibration is a well known technique in steganalysis which improves detection accuracy by making features dependent on the changes incurred by data hiding rather on the image content itself. This was first introduced by Fridrich in (27). Calibration is a process used to estimate macroscopic properties of the cover image from the stego image. During calibration, the stego JPEG image I^s is decompressed to the spatial domain, cropped by few pixels in both directions, and compressed again with the same quality matrix as the stego image. It is assumed that the newly obtained JPEG image has most macroscopic features similar to the original cover image. This is because the cropped image is visually similar to the original image. Moreover, the cropping brings the 8x8 DCT grid "out of sync" with the previous compression, which effectively suppresses the influence of previous JPEG compression and the embedding changes. We will see how this technique can be used later in Chapter 3. More information on calibration can be found in (27; 52; 53; 42).

1.3.1 Support Vector Machine

Once the feature set is fixed, the detection performance will vary based on the pattern classifier used and the actual feature values. The support vector machines (13) (SVM) are very powerful for two-class classification. Given a training set of instance-label pairs $(\mathbf{x}_i, y_i), i = 1, ..., l$ where $\mathbf{x}_i \in \mathbb{R}^n$ are the feature vectors and $y_i \in \{1, -1\}$ are the two classes, the SVM requires the solution of the following optimization problem:

$$\begin{aligned} \max \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{subject to } 0 \leq \alpha_i \leq C, \\ \sum_i \alpha_i y_i = 0. \end{aligned}$$

Here training vectors $\mathbf{x}_i$ are mapped into a higher (perhaps infinite) dimensional space by the function ϕ. Then the SVM finds a linear separating hyperplane with the maximal margin in this higher dimensional space. $C > 0$ is the penalty parameter of the error term. Furthermore, $K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i).\phi(\mathbf{x}_j)$ is called the kernel function. Once a SVM has been trained, it can be used to determine the class of an unknown sample $\mathbf{x}$ by computing the sign of

$$f(\mathbf{x}) = \sum_{i=1}^{N_s} \alpha_i y_i K(\mathbf{s}_i, \mathbf{x}) + b$$

where $\mathbf{s}_i$ are training instances $\mathbf{x}_i$ (support vectors) with $\alpha_i > 0$ and N_s is the number of support vectors. In our experiments, we use a SVM with radial basis function (RBF): $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2)$, $\gamma > 0$. For a more comprehensive introduction to SVMs, please see (13).

1.3.2 Previous work on related Steganalyzers

In one of the earliest papers to appear on blind steganalysis, Avcibas et al. (12) use image quality measures (IQM) to detect watermarks. ANOVA is used to pick IQM measures that have greater discriminating power between watermarked and non-watermarked images in the spatial domain.

A work by Fridrich (27) has clearly been a fundamental contribution to blind JPEG detection. Using a linear classifier, 23 DCT-based features and a training set of 1600 images, accuracies for correct stego classification obtained were state-of-the-art at the time. A more recent work of Farid's (46) describes the use of various Support Vector Machine (SVMs) and 432+216 = 648 features trained on 32000 images, with five JPEG embedding algorithms. The authors performed a forward feature selection search to select individual features ranked by classification accuracy, and constructed (linear) SVMs for the sets of selected features. Following Fridrich's and Farid's works, Shi et al. (62) introduced a JPEG steganalyzer using Markov transition matrices calculated on four directional differences in neighboring values in the DCT coefficients. Their steganalyzer used 324 feature values, a polynomial kernel (nonlinear) SVM classifier, and approximately 7500 training images. Their results show an improvement over both Farid's and Fridrich's features to distinguish between stego and cover image data formatted in jpeg, trained on 800 images. Shi's work presented an investigation into the use of each of the four directions individually and how each direction performs in detection. They concluded that the combination of the four sets of features performs better than each set individually, and did not reduce the number of features in the model. In Pevný et al.'s more recent work (54), the authors combine the four Markov feature sets into one set by averaging the four values at a location, producing 81 instead of 324 feature values, a reduction in the number of Markov feature values. They add the 81 to an extended set of 193 DCT features, producing a total feature set of 274 values. This set of feature values produces an improvement in classification accuracy over Shi's Markov model. Later on in 2008, Chen and Shi(18) extended the steganalyzer in (62) to 486 features by computing transition probability matrices for each difference JPEG 2-D array to utilize the intrablock correlation, and "averaged" transition probability matrices for those difference mode 2-D arrays to utilize interblock correlation. Although this steganalyzer is an improvement over preceding ones, we believe that a better accuracy rate can be achieved.

CHAPTER 2 FEATURE SELECTION FOR STEGANALYSIS USING THE MAHALANOBIS DISTANCE

As mentioned in Section 1.3, blind JPEG image steganalysis is generally addressed by representing an image with lower-dimensional features such as statistical properties, and then training a classifier on the feature set to differentiate cover and stego image. Many successful steganalysis algorithms use a large number of features relative to the size of the training set and suffer from a "curse of dimensionality": large number of feature values relative to training data size. High dimensionality of the feature space can reduce classification accuracy, obscure important features for classification, and increase computational complexity. This chapter presents a filter-type feature selection algorithm that selects reduced feature sets using the Mahalanobis distance measure, and develops classifiers from the sets. The experiment is applied to a well-known JPEG steganalyzer, and shows that using our approach, reduced-feature steganalyzers can be obtained that perform as well as the original steganalyzer. The steganalyzer is that of Pevný et al. (54) that combines DCT-based feature values and calibrated Markov features.

We show that it is possible to build a classifier having many fewer features than the full suite of features, and that these lower-dimensional classifiers have performance results very comparable to that of the classifier with all features. This also gives an indication about the features which are better for the purpose of steganalysis. This research will appear in print and by presentation at the SPIE Media Forensics and Security XII, 2010.

2.1 General Feature Reduction Technqiues

Computationally intensive, analysis for steganography content requires many hours and even days of processing image data to develop a reliable classifier. If a smaller subset of the features can be selected to represent the *intrinsic dimensionality* (35) of the data at its full dimension, then it may be possible to develop a classifier that is as effective as the one with all features, with less computation time and complexity.

There are two basic approaches to feature reduction or feature selection: filter approaches and wrapper approaches. A filter method provides a reduced feature set by ranking the feature values independently of the classifier, while a wrapper method uses a classifier to assess the effectiveness of the feature subset chosen. As such, wrapper methods are typically dependent on the type of classifier used, such as artificial neural network or support vector machine, and a classifier is developed with each choice of subset chosen. Filter methods tend to be much quicker than wrapper methods. A disadvantage of filter methods is that because of their independence from the classifier, it is not known how the selected subset will perform until a classifier is built. However, with fewer features it may be possible to develop several classifiers from which a satisfactory one can be picked.

Feature selection methods can be divided further into two categories, univariate and multivariate. Univariate methods examine features individually to determine discriminating power, while multivariate methods examine groups of feature values. Multivariate methods allow correlations and dependencies between features to be accounted for, while univariate methods do not. Univariate methods applied to steganalysis include the Bhattacharyya distance (65; 64; 70; 45). There appear to be no attempts to use the Bhattacharyya distance in a multivariate environment for steganalysis as of this time. Multivariate feature selection methods used in steganalysis include principal components analysis (PCA) (60; 44), although there are known limitations to PCA. The Mahalanobis distance measure along with an empirically determined threshold value was used in (31) for classification of feature values (Center of Mass, COM), and not for feature reduction.

The availability of statistical information from the data guides the selection of distance measure. The Bhattacharyya distance measure can be used if the probability density functions for the classes are known. Let $p_0(x)$ be the probability distribution for one class and $p_1(x)$ be the probability distribution for the second class. Then the Bhattacharyya distance between the two classes is defined as

$$B(p_0, p_1) = -log \int \sqrt{(p_0(x)p_1(x)}dx. \tag{2.1}$$

In theory, the Bhattacharyya distance gives upper and lower bounds of the Bayes error (26), although in practice the bounds are often not tight enough. While this univariate filter-approach is straightforward to compute for image feature data, it does not take into consideration other information, such as mean or variance values.

PCA can be used for feature reduction. It provides the optimal solution to the linear projection problem of feature reduction when using the least mean square measure of error. The general goal of using PCA is to provide a representation of the data that is de-correlated, where second-order dependencies are removed. It can also be used to reduce the number of feature variables necessary for classification. PCA has been useful in reducing the thousands of variables in microarray data (51) to a manageable amount. PCA ranks features according to the variance of features in a transformed and uncorrelated feature space described by the singular value decomposition (63). If the data has higher order dependencies or nonlinear dependencies, then PCA will be insufficient in revealing all the structure in the data. Because PCA simply ranks the feature variables according to the variance of the data, PCA may or may not provide an adequate clustering of the data classes. Selecting features according to largest eigenvalue ranking given by principal component analysis can be suboptimal (17; 15), including for steganalysis (44; 70).

If estimates of the population mean values, variances, and covariance matrix are available from the data as is often the case in steganalysis, then the Mahalanobis distance can be used. The Mahalanobis distance between two classes can be written as

$$D^2 = (\mu^s - \mu^c)' V^{-1} (\mu^s - \mu^c) \tag{2.2}$$

here expressed for two classes s = stego and c = cover. Here, μ^s (μ^c) represents the mean vector for the stego (cover) population, prime denotes matrix transpose, and V is the covariance matrix.

The Mahalanobis distance has been used in other areas of pattern recognition for several decades, including species identification in zoology (61), diagnostic validity in neurology (36), and many other fields. It is used to provide a measure of similarity between multivariate populations and uses covariance information between variables to weight the contributions to the distance. The Euclidean distance, on the other hand, in essence gives excess weight to variables that are highly correlated and gives additional weight to variables that have similar information. The Mahalanobis distance gives less weight to those variables that have high variance and to those variables that have high correlation, so that other feature variables with lower correlations can contribute to the distance. When correlations between variables are known to exist, the Mahalanobis distance can offer an advantage for clustering. That is the case for the feature sets under consideration in this chapter.

2.2 Previous Work on Feature Reduction in Steganalysis

While the work described in Section 1.3.2 shows JPEG steganalysis on how individual features classify (as in (27)) or linear classification of selected subsets of features (as in (46)), only a few other papers present systematic methods for reducing features in JPEG steganalysis and creating comparable classifiers. A filter approach using the Bhattacharyya distance for steganalysis feature selection was investigated in (70); however, the application to steganalysis was very brief, comparing the detection rate on one spatial domain LSB embedding algorithm only. In (44), the authors describe three wrapper methods for grayscale steganalysis. In (40), the authors investigate the use of PCA for total variance of the data for spatial domain embedding, and perform forward and backward feature selection on the 27 features for several databases, comparing detection accuracies. In (49) , the authors describe a feature selection method for steganalysis on JPEG images using a K Nearest Neighbor approach, which takes some time to process as it is itself a classifier. In (48), the authors use a wrapper-type method to reduce the number of features for JPEG steganalysis. Authors in several papers have chosen fewer feature values so as to make their number of features manageable in the course of their feature design (19; 34).

With the exception of the grayscale steganalysis research in (33; 44; 40; 46), and the JPEG steganalysis in (49; 48), none of these ten previous works perform any in-depth investigation into the selection of fewer features for the express purpose of overcoming the dimensionality limitations. That is the goal of our work in this chapter. Our work is the first (to our knowledge) use of the Mahalanobis measure for feature selection in steganalysis. We show how the Mahalanobis distance can be used to provide a good ranking of the features and maintain accuracy of detection relative to the original full suite of features. The benefits of using a ranking of feature variables that maintain the ability to classify well with smaller numbers of features include reduced complexity in both feature dimension and training computation time. This can lead to improved ability of the classifier to generalize its solution to unseen data. We select support vector machines for the classifiers, as they are known to offer the potential for generalizable solutions in high-dimensional feature spaces (35).

2.3 Mahalanobis distance

We now describe the Mahalanobis distance measure. Let X_i be a random variable (r.v.) representing feature i, and let x^c_{ij} (x^s_{ij}) denote a sample from cover (stego) image i of feature value j. Denote by $\bar{x}^c_i$ ($\bar{x}^s_i$) the sample mean for feature i for the cover (stego) class, and by $v^c_{jk} = \frac{1}{N-1}\sum_{i=1}^{N}(x^c_{ij} - \bar{x}^c_j)(x^c_{ik} - \bar{x}^c_k)$ the sample covariance between feature j and feature k. The sample covariance values v^s_{jk} are similarly defined for stego images. Let $V^c = (v^c_{jk})$ denote the cover sample covariance matrix, and $V^s = (v^s_{jk})$ denote the stego sample covariance matrix. Let p denote the number of features and let N denote the number of training images. In our particular case, $N = 5000$ each for cover and stego, and $p = 274$. After calculating the sample covariance matrices, we use the pooled covariance matrix to calculate the Mahalanobis distance (47):

$$V = \frac{N}{2N-2}V^c + \frac{N}{2N-2}V^s = \frac{N}{2N-2}(V^c + V^s). \tag{2.3}$$

For a set of $k \leq p$ features, the Mahalanobis distance $D^2_{(k)}$, in quadratic form, between the centroids of those features is given by:

$$D^2_{(k)} = (\mu^c - \mu^s)' V^{-1} (\mu^c - \mu^s) \tag{2.4}$$

where $\mu^c = (\bar{x}^c_1, \bar{x}^c_2, ..., \bar{x}^c_k)'$ and $\mu^s = (\bar{x}^s_1, \bar{x}^s_2, ..., \bar{x}^s_k)'$ represent the vector of sample mean values for the cover and stego images, respectively, and V^{-1} is the inverse of the pooled covariance matrix for the k features. In practice, V may be only positive semi-definite (that is, not full rank) or ill-conditioned. The problem of inverting V may be regularized by offsetting the diagonal of V and thus creating a positive definite matrix out of V in the following way. Let $\epsilon > 0$ be a number that is significantly smaller than the average variance of the class clusters, and write $V = QDQ'$ as the diagonalization of V by orthogonal matrix Q. The values of $D = diag(\lambda_1, ..., \lambda_n)$ are the eigenvalues for V. By adding ϵI to the matrix V, where I is the identity matrix, we can solve the problem of inverting V effectively:

$$V + \epsilon I = QDQ' + \epsilon I = QDQ' + Q(\epsilon I)Q' = Q(D + \epsilon I)Q' \tag{2.5}$$

Since the eigenvalues in $V + \epsilon I$ are now strictly positive and ϵ was chosen "large enough" to avoid ill-posedness, we let $\hat{V} = V + \epsilon I$ replace the covariance matrix V in the calculation of the Mahalanobis distance, Eq. 2.4. Our value for ϵ is the minimum of the diagonal entries of the pooled covariance matrix V, divided by 10 000.

Note that the Mahalanobis distance gives less weight for features having larger variance and more weight for features having smaller variance. This can be seen by looking at the one dimensional case (only one feature), where the Mahalanobis distance is

$$D^2_j = \frac{(\bar{x}^c_j - \bar{x}^s_j)^2}{\sigma^2_j}, j = 1, ..., p \tag{2.6}$$

where $\bar{x}^c_j$ ($\bar{x}^s_j$) are the respective sample means for the j-th cover (stego) feature, and $\sigma^2_j = \frac{N}{2N-2}[\sum_{i=1}^{N}(x^c_{ij} - \bar{x}^c_j)^2 + \sum_{i=1}^{N}(x^s_{ij} - \bar{x}^s_j)^2]$ is the common variance. In higher dimensions with more than one feature, $\frac{1}{\sigma^2}$ is replaced by the inverse of the (regularized) covariance matrix. A larger Mahalanobis distance value indicates more class separation when the Mahalanobis distance is used, with a threshold, as a linear classifier.

The Mahalanobis distance can be expressed in terms of principal components, that is, in terms of the eigenvectors and eigenvalues that are used in principal component analysis. Following a result by (17), the relation between eigenvalues of the covariance matrix for the feature data and values for the Mahalanobis distance measure becomes clear. The PCA in this case is performed on the feature data matrix Y consisting of all mean-centered cover and stego feature vectors vertically concatenated and having dimension $2N$x p. Let W be the p dimensional covariance matrix for Y: $W = Y'Y$. Let D^2 denote the Mahalanobis distance between the two classes $c = cover$ and $s = stego$ where c takes proportion q and s takes proportion $1 - q$. Then for $d = \mu^c - \mu^s$, the respective mean vectors of length p for the two feature classes, and V in Eq. 2.3, it can be shown that

$$W = q(1-q)dd' + V \tag{2.7}$$

$$D^2 = d'V^{-1}d \tag{2.8}$$

Let $P_1, P_2, ..., P_p$ (the columns of P) denote the p eigenvectors of W, with eigenvalues $\gamma_1, \gamma_2, ..., \gamma_p$ respectively. Recall the spectral representation of W: $W = \sum_{i=1}^{p} \gamma_i P_i P_i'$. For given $k \leq p$, let $B_k = (P_1, P_2, ..., P_k)$ be a basis and denote $D^2_{(k)}$ as the distance between the clusters using B_k in place of W. Here it is not assumed that $\gamma_1 \geq \gamma_2 \geq .... \geq \gamma_p$. It has been shown in (17) that

$$D^2_{(k)} = \frac{\sum_{i=1}^{k} \frac{\left(P_i'd\right)^2}{\gamma_i}}{\left(1 - q(1-q)\sum_{i=1}^{k} \frac{\left(P_i'd\right)^2}{\gamma_i}\right)} \tag{2.9}$$

In particular, for $k = 1$, for any single feature i,

$$D^2_{(1)} = \frac{\frac{\left(P_i'd\right)^2}{\gamma_i}}{\left(1 - q(1-q)\frac{\left(P_i'd\right)^2}{\gamma_i}\right)} \tag{2.10}$$

Eq. 2.9 shows clearly that the maximum value of the Mahalanobis distance for k features may not correspond to the top principal components corresponding to the largest k eigenvalues.

In short, our use of the Mahalanobis distance is as a (linear) multivariate classifier, where we use a forward feature selection algorithm to rank features. We then follow by classification using a support vector machine on increasingly nested subsets of features.

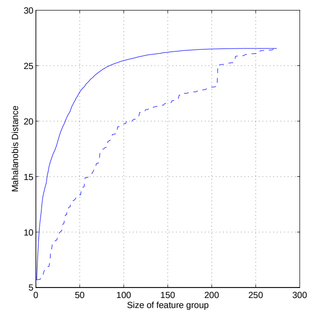
2.4 Description of Experiments

Using the heuristic of selecting approximately 10 times as many training examples as there are features (35), we used 5000 original image data for training. Recent research results indicate (48) that using a specified steganalysis scheme - the set of extended DCT feature values from the Merged model having 193 feature variables - a minimum of 5000 image training data was needed to produce classification results that have less than 1 percent standard deviation in their output values. For our experiments, we use the BOWS2 database (67), containing 10 000 images, and pick half of them to create the training database and use the remainder for testing. With approximately 10 000 images for the training set (5000 cover and 5000 stego), we assume that this large data set will deliver relatively low variance in classifier predictability and that the detection accuracies we obtain produce reliable steganalysis in a comparable sense. Our main interests are in comparing results of the full complement of features to classifiers trained with fewer features. Our assumption is not unreasonable, in particular the data used in (48) was also the BOWS2 database with the extended DCT feature set, very similar to our experiment. Those authors showed that for that data set and feature set, variance of results was experimentally verified to be under 1 percent. Other authors have also used approximately 5000 images for similar research (38; 44; 20).

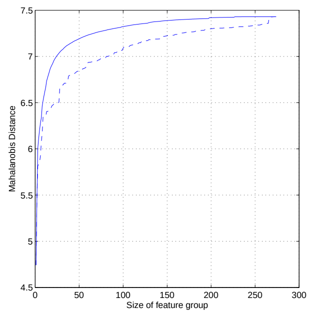
The BOWS2 database consists of images in pgm format of size 512x512. The images were divided randomly into two disjoint groups of equal size. The first group was used to create the training examples with both cover and stego data. The second group contained the remaining images that were used for testing. We did not use those stegoimages for which the steganography embedding algorithm exited unsuccessfully. Thus, no image or its different variations were simultaneously seen by the SVMs for testing and training, and there were an equal number of images used for cover and stego. This strict division of images enabled us to estimate the performance on never seen images. We intentionally chose to generate our cover and stego images for training and classification in a way to avoid double compression, reformatting the raw pgm files into JPEG files saved with 75% quality factor. Recall that a JPEG image is double compressed when it is first compressed using quantization matrix Q^1, then uncompressed and re-compressed using quantization matrix Q^2, where $Q^2 \neq Q^1$. When an image undergoes double compression, the statistics of DCT coefficients can change, and may resulting in misclassification if the steganalyzer is not designed to handle detection of double compression. Our detection scheme assumes that the data has not been double-compressed.

We generate stego images by embedding data with different message lengths and different embedding algorithms. The five different steganography embedding methods we consider are: OutGuess (56), F5 (68), JPHide& Seek (3), StegHide (2), and JSteg (4). Each of these methods embeds bits of value 0 or 1 directly into the quantized DCT coefficient array. The payload is assumed to be an encrypted bitstream. We use *bits per nonzero ac coefficient*, or *bpnz* to describe message length, with $bpnz = 0.05, 0.1, 0.2, 0.4$. Images embedded using Outguess have only $bpnz = 0.05, 0.1, 0.2$. Once the stego images are generated, we extract feature values from the cover and stego images according to the steganalyzer by Pevný et al. in (54), which we call *Merged*. The full suite of feature values are extracted initially, and subsets of the feature values are used in accordance to the task at hand.

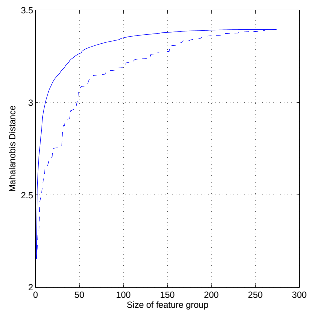
First, to determine the feature subset using the Mahalanobis distance, we gather all the feature data calculated from the training set for all cover and all embedding algorithms at all levels of embedding. We used 5000 cover and 1250*4 stego images (four levels of embedding) for calculating Mahalanobis distance for each steganography algorithm. Note that in Eq. 2.4, the size of the (pooled) covariance matrix is equal to the number of feature variables used. The regularized covariance matrix is invertible but for each new trial set of k features, the inverse matrix $\hat{V}^{-1}$ must be calculated, and thus this method is fairly computationally time-consuming, especially as k increases. Once the ranking is complete, a classifier is built.



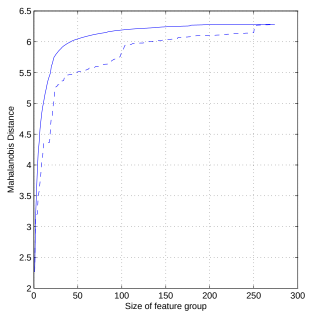
(a) Jsteg



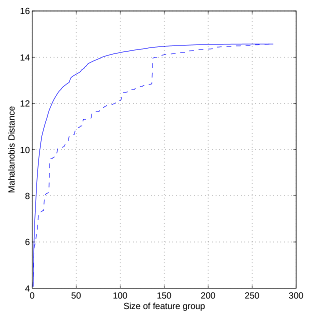
(b) Outguess



(c) F5



(d) Steghide



(e) JPHide

Figure 2.1: Mahalanobis distance vs. feature group size for five embedding algorithms using features in (54). Solid line: multivariate distances. Dashed line: univariate distances. Note different scales on distance axes.

The SVM classifier we implemented was a soft margin support vector machine with gaussian kernel (13) (using LIBSVM (16)). We determined the training parameters of the C-SVMs by grid-search performed on the following multiplicative grid

$$(C, \gamma) \in \left\{ \left(2^i, 2^j\right) | i \in Z, j \in Z \right\}.$$

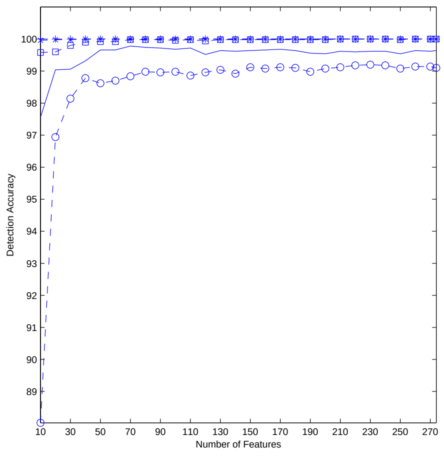
Although we do not use them, SVMs that use Mahalanobis kernels have been developed for pattern recognition problems. In (30), the authors present a comparison of kernel Mahalanobis distance classifiers giving experimental evidence that these types of classifiers can be advantageous for nonlinear pattern distributions.

We provide two rankings of the features. For the *univariate* ranking, we computed the Mahalanobis distance using individual features. We use Equation 2.6 and a single feature j from the two classes cover and stego, and compute p univariate values. The ranking for the univariate case is simply in decreasing values determined by Equation 2.6.

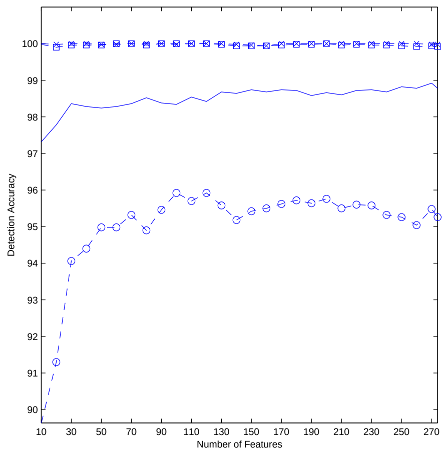
The Mahalanobis distance values for the univariate case are computed separately for each of the five embedding algorithms, where the data for all levels of embedding available for the specified embedding algorithm are combined (4*1250=5000). Once the univariate ranking is determined, the Mahalanobis distance for groups of features of size k, $k = 1, ..., 274$ are calculated. Graphs for these groups of univariate Mahalanobis distance values are shown in Figure 2.1 and indicated by the dashed line in each of the five graphs.

For a second experiment, we used a forward feature selection method to create subsets of features that collectively give Mahalanobis distance values. We call this *multivariate* feature selection. Starting with the feature that gave the largest univariate Mahalanobis distance value in Equation 2.6, the Mahalanobis distance values for all remaining feature vectors paired with the first feature were calculated ($k = 2$ in Equation 2.4), and the feature that gave the largest Mahalanobis distance value over all pairs was selected as the second feature. This continued in an iterative fashion, using the multivariate Mahalanobis distance measure as given by Equation 2.4 for $k = 2, 3, ..., 274$, selecting the feature that gave the largest Mahalanobis distance value at each iteration. This produced a second ranking of the features that was different from the univariate ranking. The multivariate Mahalanobis distance values for groups based on this second ranking are shown in Figure 2.1 and indicated by the solid line in each of the graphs. Based on the graphs in Figure 2.1, it is clear that every feature used in Pevný et al. (54), paper contributes some information that increases the Mahalanobi distance monotonically.

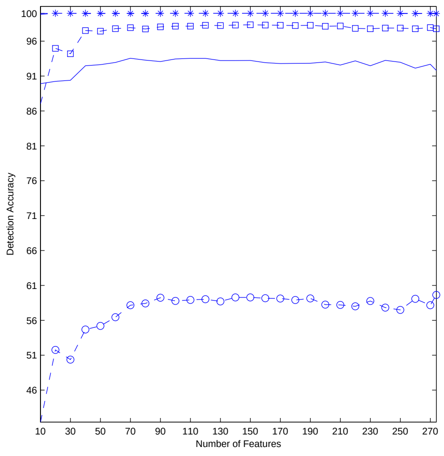
Since our goal is to produce classifiers using these subsets and determine if fewer features can produce classification results comparable to the full suite of features, we next created SVMs for feature sets of increasing size. We elected to choose feature subsets of size $10 * M, M = 1, 2, ..., 27$, plus the entire set of 274 features. For each set of features and for each embedding algorithm, we create an SVM, and produce detection accuracy data using the test set. An SVM was created for the 28 feature sets in the univariate ranking, and for the 28 feature sets in the multivariate ranking. To train each SVM, the cover image data provided 5000 feature vectors, and the stego image data (in total) provided 5000 training data. We test for accuracy on each level of *bpnz* for each embedding algorithm and separately



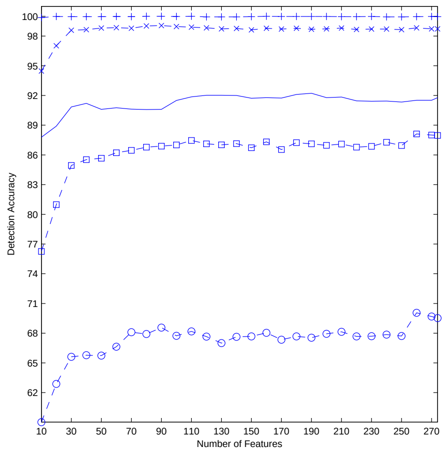
(a) Jsteg



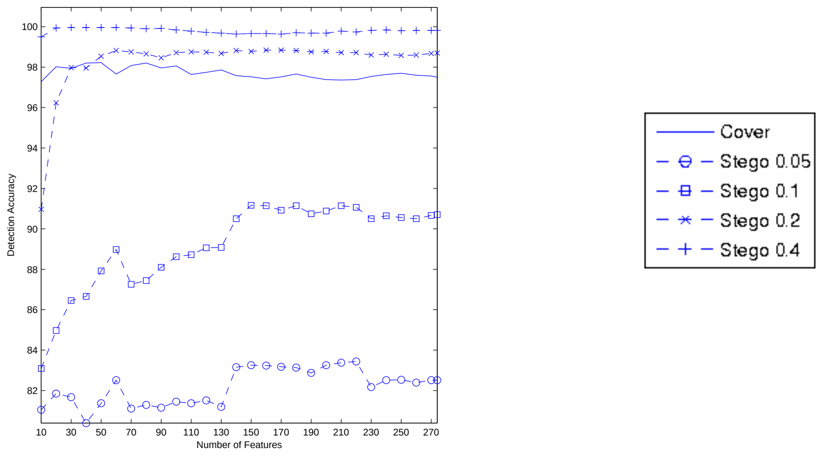
(b) Outguess



(c) F5

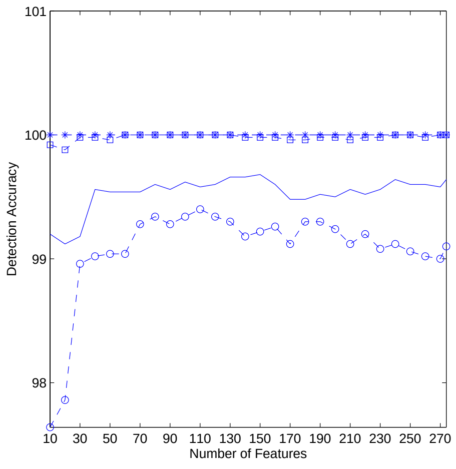


(d) Steghide

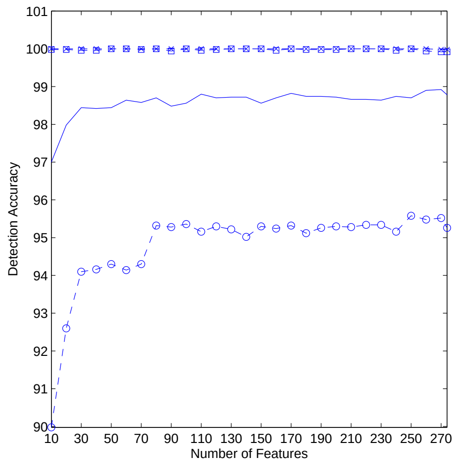


(e) JPHide

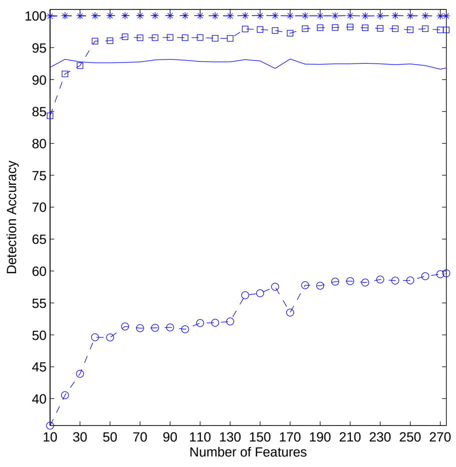
Figure 2.2: Detection accuracy results using univariate feature ranking for groups increasing by 10 features, cover images (solid line) and four embedding rates for five algorithms. Note different scales on distance axes.



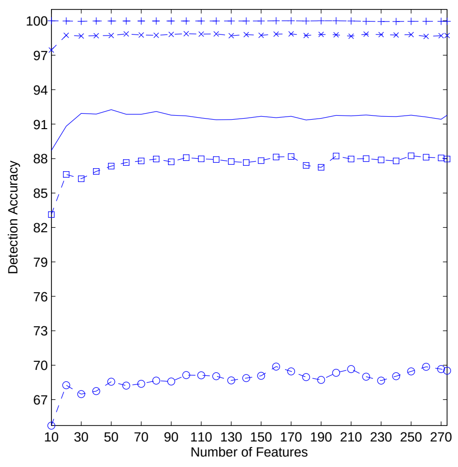
(a) Jsteg



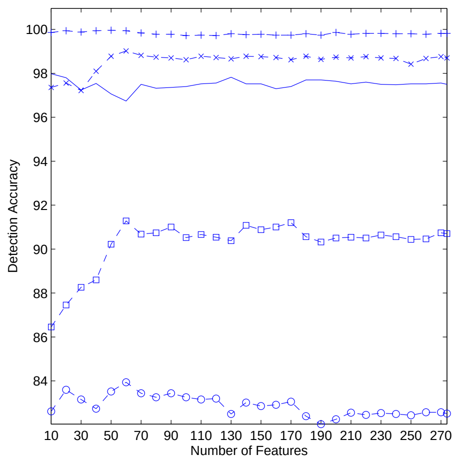
(b) Outguess



(c) F5



(d) Steghide



(e) JPHide

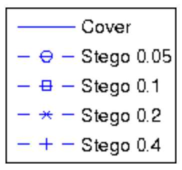


Figure 2.3: Detection accuracy results using multivariate feature ranking for groups increasing by 10 features, for cover images (solid line) and four embedding rates for five algorithms. Note different scales on distance axes.

for the cover data (no hidden data). For testing, we use the unseen data with 5000 cover images and 5000 stego images for each embedding level, for a total of 25000 test data.

Table 2.1: Three Top Ranked Classifiers for Univariate and Multivariate Rankings. Displayed for each embedding algorithm and level of embedding. k = number of features.

		Merged	Univariate Ranking						Multivariate Ranking					
			1st	k	2nd	k	3rd	k	1st	k	2nd	k	3rd	k
Jsteg	Cover	99.64	99.78	70	99.74	80	99.72	90	99.68	150	99.66	130	99.66	140
	0.05 bpnz	99.10	99.20	230	99.18	220	99.18	240	99.40	110	99.34	80	99.34	100
	0.10 bpnz	100.00	100.00	210	100.00	220	100.00	230	100.00	60	100.00	70	100.00	80
	0.20 bpnz	100.00	100.00	40	100.00	50	100.00	60	100.00	10	100.00	20	100.00	30
	0.40 bpnz	100.00	100.00	10	100.00	20	100.00	30	100.00	10	100.00	20	100.00	30
Outguess	Cover	98.78	98.92	270	98.82	250	98.78	260	98.92	270	98.90	260	98.82	170
	0.05 bpnz	95.26	95.92	100	95.92	120	95.76	200	95.58	250	95.52	270	95.48	260
	0.10 bpnz	99.92	100.00	60	100.00	70	100.00	90	100.00	50	100.00	60	100.00	80
	0.20 bpnz	99.98	100.00	10	100.00	30	100.00	40	100.00	10	100.00	20	100.00	30
F5	Cover	91.82	93.56	70	93.54	110	93.54	120	93.22	170	93.16	20	93.16	90
	0.05 bpnz	59.64	59.64	274	59.28	140	59.28	150	59.64	274	59.50	270	59.18	260
	0.10 bpnz	97.78	98.36	150	98.32	140	98.32	160	98.24	210	98.16	200	98.12	190
	0.20 bpnz	99.96	100.00	20	100.00	30	100.00	90	100.00	50	100.00	60	100.00	70
	0.40 bpnz	99.98	100.00	20	100.00	30	100.00	40	100.00	20	100.00	30	100.00	40
Steghide	Cover	91.78	92.22	190	92.10	180	92.02	120	92.26	50	92.10	80	91.94	30
	0.05 bpnz	69.52	70.06	260	69.68	270	69.52	274	69.88	160	69.86	260	69.66	210
	0.10 bpnz	87.96	88.12	260	88.00	270	87.96	274	88.24	250	88.22	200	88.18	170
	0.20 bpnz	98.72	99.06	90	99.02	80	98.96	100	98.88	100	98.86	60	98.86	120
	0.40 bpnz	99.96	100.00	80	100.00	90	100.00	110	100.00	10	100.00	160	100.00	170
JPHide	Cover	97.50	98.22	50	98.20	40	98.20	80	97.98	10	97.82	130	97.80	20
	0.05 bpnz	82.51	83.44	220	83.38	210	83.26	150	83.94	60	83.60	20	83.52	50
	0.10 bpnz	90.70	91.16	150	91.14	160	91.14	180	91.28	60	91.20	170	91.08	140
	0.20 bpnz	98.70	98.84	160	98.84	170	98.82	60	99.02	60	98.82	70	98.78	50
	0.40 bpnz	99.82	99.96	30	99.96	40	99.96	50	99.96	50	99.94	20	99.94	40

The accuracies for each embedding algorithm and each embedding level are displayed in Figures 2.2 and 2.3. The graphs show that for many instances, accuracies close to the full set of features are reached with many fewer features than the full 274. Table 2.1 displays a few of the top results in number format that supports this observation.

2.5 Discussion of Results

From the graphs and table, we can see there are many instances of feature sets that produce results within 1-2% of the Merged feature set. In general, the multivariate feature sets were able to give results comparable to the univariate feature sets but with fewer features, although for both rankings, the lower levels of embedding required more features than the higher levels of embedding to give comparable results to the full Merged set. Jsteg in its sequential embedding form is known to be straightforward to detect. For the freeware available on the internet, Jsteg embeds payload bits are embedded in a lexicographical order from top left to bottom right of the coefficient array. Embedding in a random order is of course preferable but software code must be written by a user to implement this. Evidence of highly accurate detection rates are displayed in graphs in Fig. 2.2(a) and Fig 2.3(a). Also, full embedding at the 0.4 bpnz level is detected at high levels of accuracies, near 100% for all the steganalyzers of all sizes except for size 10 features in JPHide using the univariate ranking, and sizes 10-30 features in Steghide for multivariate ranking. Lower rates of embedding have lower rates of detection, especially for 0.05 bpnz in F5 where matrix embedding is used. Detection rates for the class cover are generally in the upper 90s% except for F5 and Steghide. The false positive rate for each steganalzyer is 1-(cover detection rate). The way we implemented the SVMs did not allow us to choose detection accuracies based on false positive rates. Higher or lower false positive rates may be desirable or necessary for certain applications. Forensic investigations typically require higher lower positive rates. The general trend was for detection accuracies to increase as more features were added to the classifiers with increases leveling off or oscillating between 50-100 features. However, some classifiers had jumps up or down in accuracy rates as a group of 10 features were added to the current set. This is apparent in Outguess-Univariate and Multivariate, where accuracies increased from 80 to 90 features, and in JPHide-Univariate, where accuracies decreased from 60 to 70 features, and a curious dip in F5-Multivariate from 160 to 170 features. Another phenomenon that occurred was in several instances, cover detection rates changed in the opposite direction of stego detection rates from one size to another. For example, if a user wanted to choose a feature subset based on a high cover detection rate so that the false positive rate was low, then the lowest embedding detection rate may not be as high as possible. This happens in F5-Multivariate, where one of the highest cover detection rates occurs for n=170 features, but that is also one of the lower detection rates for 0.01 embedding. Trade-offs in detection accuracies must be made.

In Table 2.1, we give a listing of the top three ranked feature sets for each ranking algorithm, univariate and multivariate. If a user wanted to reduce the feature size significantly, there are a number of classifiers that give results close to the Merged results. For example, to detect JPHide and Seek cover images to within 1% of the Merged accuracy, 10 multivariate features could be used, and 50-60 could give as good as or better accuracy than the Merged for all levels of embedding for JPHide and Seek. We also found that many of the top features are from Calibrated Markov feature set for all the steganographic algorithm. This gives a strong indication that features which exploit neighborhood dependency among DCT coefficients are effective in steganalysis. We use this information to build a model for steganalysis in the next chapter.

CHAPTER 3 STEGANALYZER FEATURES DESIGNED ON STOCHASTIC MARKOV MODEL

In this section, we present a new set of features for steganalysis. Experiments in Section 2.4 show that features which exploit neighborhood dependency among DCT coefficients are well suited for steganalysis. A Markov based process has been used with success in (62; 18) to model DCT coefficients for the purpose of steganalysis. In this chapter, we show the use of the Partially Order Markov Model (POMM) (24; 22; 32) for steganalysis which generalizes the concept of local neighborhood directionality by using a partial order underlying the pixel locations.

Markov random fields are a well-known modeling tool and have been used successfully in many areas of image analysis. However, the use of MRFs continues to be problematic when problems require computing an explicit joint probability, such as for texture classification and parameter estimation. A nice subclass of MRFs was introduced by Abend et al (9) for image analysis, called *Markov mesh models* (MMMs). MMMs allow, under minimal and reasonable assumptions, an explicit closed form for the joint probability of the random variables (r.v.s) at hand, expressed in terms of a conditional probability. The conditional probabilities express the spatial dependency of the data, via a directional neighborhood, unlike the undirected neighborhood of a MRF model. Abend et al. also showed that the conditional probability of one r.v. given the rest of the r.v.s can be expressed in terms of r.v.s in a local spatial neighborhood. A partially ordered Markov model generalizes the concept of local neighborhood directionality by using a partial order underlying the pixel locations. It has been shown that this property results in a computational advantage of POMMS over MRFs: whenever the normalizing constant needs to be calculated, such as in determining the joint probability distribution function (pdf), the joint pdf of a POMM is available in closed form, and the normalizing constant for a POMM is always known and equal to the value one (24).

It is beyond the scope of this chapter to give a detailed introduction to POMMs, but a few ideas pertinent to the discussion at hand can be discussed. We assume we have a partial order $\prec$ placed on the set of pixel locations in the image, or a poset. It can be shown that for a given poset $(A, \prec)$, we

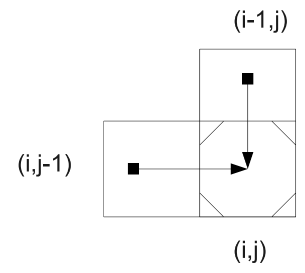


Figure 3.1: The adjacent lower neighborhood $adj_{\prec}(a_{i,j})$. The hash marks represent the location (i, j).

have a class of acyclic directed graphs where each class corresponds to the same poset $(A, \prec)$ but the individual acyclic directed graphs in the class have different edge sets. The edge sets give rise to the neighborhood relationship between pixel values, which in turn are used to describe conditional probabilities on neighborhoods of pixels. We shall be interested in the Markovian neighborhood relationship, described for POMMs in Definitions 1 and 2, where A is the set of r.v.s in the image, and E a set of edges. Given a directed edge (C, B), tail on C, head on B, we write $C \prec B$ under the partial order $\prec$.

Definition For any $B \in A$, the *cone* of B is the set *cone* $B = \{C \in A : C \prec B, C \neq B\}$.

Definition For any $B \in A$, the *adjacent lower neighbors* of B are those elements $C \in A$ such that (C, B) is a directed edge in the graph (A, E). Formally, $adj_{\prec}B = \{C : (C, B)$ is a directed edge in $(A, E)\}$.

We give a simple example of the adjacent lower neighborhood in Figure 3.1. Here $a_{i,j-1} \prec a_{i,j}$ and $a_{i,j-1} \prec a_{i,j}$. The definition of a POMM is as follows, where L_0 is the set of minimal elements in the poset. The notation $P(A)$ denotes the (discrete) probability measure A, based on the r.v. A. We use the common notation that an upper-case letter denotes the r.v. A, while a realization of A is denoted by a. $P(A)$ is used to denote the probability measure that defines the r.v. A, and $P(A|B)$ is used to denote the conditional probability measure of A given another r.v. B.

Definition The partially ordered Markov model (POMM) is defined as follows: Let $B \in A$ where (A, E) is a finite acyclic digraph of r.v.s and $(A, \prec)$ is its corresponding poset. Describe the set of r.v.s not related to B by $Y_B = \{C :$ B and C are not related $\}$. Then $(A, \prec)$ is called a *partially ordered Markov model* (POMM) if for any $B \in A \backslash L^0$ and any subset $U_B \subset Y_B$ we have

$$P(B|cone\ B, U_B) = P(B|adj_{\prec}B).$$

For our purposes, we assume that the acyclic digraph underlying the r.v.s of the random image A is the one induced by replicating $adj_{\prec}(a_{i,j})$ at all locations (i,j) in the pixel domain. In other words, the neighborhood is translational invariant. It must be checked that the digraph thus generated is indeed acyclic. Some restrictions that guarantee an acyclic digraph can be found in (24). For example, the digraph generated by $adj_{\prec}(a_{i,j})$ in Figure 3.1 is acyclic.

This material is sufficient to discuss the features described next. The interested reader is directed to (24).

3.1 POMMS for Steganalysis

With the notation introduced in the previous section, we now discuss the application of POMMs to steganalysis. We develop POMM directly on quantized DCT coefficients array as the embedding takes place directly in that domain. Assume that $\mathbf{A}$ is on a rectangular pixel set, $\mathbf{A} = \{A_{i,j} : 1 \leq i \leq M, 1 \leq j \leq N\}$. Let $\Theta_{i,j}$ be an ordered indexing set that is invariant to shifts on the array where $|\Theta_{i,j}| = n$. For example

$$\Theta_{i,j} = ((i,j),(i,j+1)) \tag{3.1}$$

is such a set and $|\Theta_{i,j}| = 2$. We describe an invariant cluster set $C_{i,j}$ on the array of r.v.s $\mathbf{A}$ by $C_{i,j} = \{A_{k,h} : (k,h) \in \Theta_{i,j}\}$. A new set of r.v.s $\mathbf{C}$ is defined as $\mathbf{C} = \{C_{i,j} : 1 \leq i \leq M, 1 \leq j \leq N\}$. Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function that exploits the dependency among DCT coefficients in $\Theta_{i,j}$. Apply f via an induced manner to $\mathbf{C}$ by $f(\mathbf{C}) = \mathbf{W}$ where $\mathbf{W} = \{W_{i,j} : W_{i,j} = f(C_{i,j}) \in \mathbb{R}, 1 \leq i \leq M, 1 \leq j \leq N\}$. For example, let $\Theta_{i,j}$ be as in Equation 3.1 and define $f : \mathbb{R}^2 \to \mathbb{R}$ by

$$f(C_{i,j}) = C_{i,j}[0] - C_{i,j}[1] \tag{3.2}$$

Then f applied to $\mathbf{C}$ produces $f(C_{i,j}) = f(A_{i,j}, A_{i,j+1}) = A_{i,j} - A_{i,j+1} = W_{i,j}$. To create a POMM, we define the partial order as $W_{i,j} \prec C_{i,j}$ for $1 \leq i \leq M, 1 \leq j \leq N$. This in turn defines a directed edge with tail on $W_{i,j}$ and head on $C_{i,j}$. This is shown in Figure 3.2. The adjacent lower neighbors is the set of r.v.s $adj_{\prec}C_{i,j} = W_{i,j}$. The POMM defined by its conditional probabilities, given by $P(C_{i,j}|W_{i,j})$.

This satisfies the definition of a POMM and all the properties now apply. For the probabilities at hand, we are specifically interested in the function f given in Equation **??** and applied to the four directions horizontal h, vertical v, diagonal d, and minor diagonal m. In this case, we have

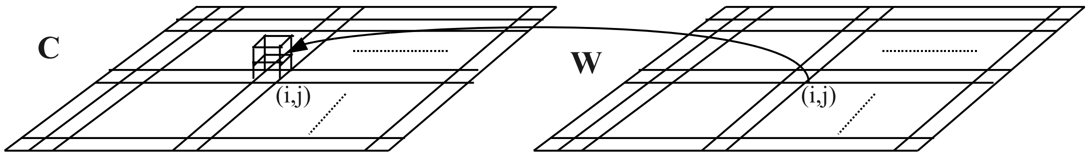


Figure 3.2: Partially Ordered Markov Model for steganalysis

$C^h_{i,j} = (A_{i,j}, A_{i,j+1})$, $C^v_{i,j} = (A_{i,j}, A_{i+1,j})$, $C^d_{i,j} = (A_{i,j}, A_{i+1,j+1})$, $C^m_{i,j} = (A_{i+1,j}, A_{i,j+1})$. Thus in each direction, we calculate $P(C_{i,j}|W_{i,j})$ by noting that

$$\begin{aligned} P(C_{i,j}|W_{i,j}) &= P(C_1, C_2|f(C_1, C_2)) && (3.3)\\ &= P(C_1, C_2|C_1 - C_2) && (3.4)\\ &= \frac{P(C_1, C_2, C_1 - C_2)}{P(C_1 - C_2)} && (3.5) \end{aligned}$$

which is then perform by histogram binning of the data. If we assume that $-T \leq a_{i,j} \leq T$ for $1 \leq i \leq M, 1 \leq j \leq N$, then $-2T \leq w_{i,j} \leq 2T$ and it is easy see that there will be $(2T+1)^2$ number of such conditional probabilities for each direction. We calculate conditional probabilities, given by $P(C_{i,j}|W_{i,j})$ for all the four different directions and use average of the 4 directional probabilities to characterize f on $\mathbf{A}$. These conditional average conditional probabilities, denoted by $\mathbf{F}$ will be used as features for steganalysis.

We fit the POMM described on the array of quantized DCT coefficients as the embedding takes place directly in this domain. Since the embedding process generally does not take into account the dependency among DCT coefficients, correlations between DCT coefficients in cover and stego image is affected. We fit a POMM as follows to exploit both inter and intra block dependency among DCT coefficients. Before we fit the POMM, we clip quantized DCT coefficients to between [-T,T]. This is necessary, otherwise the pdf for the POMM will be very sparse as DCT coefficient value lies within [-1024,1023]. In our case, we use T=5 as it is well known that more than 96% of the DCT coefficients are found to be within [-5,5] (28; 69). This results in 121 intra and 121 inter block features and hence a total of 242 features for steganalysis.

3.1.1 Intra block features

Certainly, steganographic embedding causes disturbance on the smoothness, regularity, continuity, consistency, and/or periodicity of quantized DCT coefficients, and therefore correlations among DCT coefficients. To quantify this change, we apply the above developed POMM on the quantized DCT coefficients array **B** as described in Section 1.1. We use the conditional probabilities **F** of this POMM as intra block features. Since this POMM models the dependency among DCT coefficients within a 8x8 DCT block, we refer to these probabilities as intra block features.

3.1.2 Inter block features

Inter block correlation is reflected among those JPEG modes, i.e., coefficients located in the same position within the 8x8 blocks, which capture the frequency characteristics of those blocks. JPEG steganographic embedding will disturb this kind of interblock correlation. Let the quantized DCT coefficient array **B** is of size MxN. Then there are $N_r * N_c$ number of 8x8 DCT blocks where $N_r = \lceil \frac{M}{8} \rceil$ and $N_c = \lceil \frac{N}{8} \rceil$. Let $X^{i,j}$ be the array formed by collecting DCT coefficients located at i, j from every 8x8 blocks. Equivalently, $X^{i,j}_{u,v}$ is the dct coefficient located at i, j in u, v block where $1 \leq u \leq N_r$, $1 \leq v \leq N_c$. The array $X^{i,j}$ is called a *mode array* as it represents mode or specific frequency from every 8x8 block.

To capture inter block dependency, we calculate POMM probabilities as given in Equation 3.5 on every mode array $X^{i,j}$. We use the averaged conditional probabilities of all the mode image as inter block features.

We further apply calibration to reduce the dependency of the feature values on the image content itself. Let I^o be the given image, and let its calibrated image be I^{cal}. We calculate intra and inter block features for I^{cal} also, and then finally use $\mathbf{F^o} - \mathbf{F^{cal}}$ as our features.

3.2 Experiments

It has been shown recently in (40) that the performance of a steganalyzer depends on the database used for training and testing the steganalyzer. Therefore, for our experiments, we use four different databases to compare our steganalyzer based on the proposed feature set with other steganalyzers.

- **Bows2**: This database contains 10000 images of size 512x512 in pgm format. Please refer to Section 2.4 for a detailed description.

- **Camera**: This database consists of 3164 images captured using 24 different digital cameras (Canon, Kodak, Nikon, Olympus and Sony) previously used in (29). They include photographs of natural landscapes, buildings and object details. All images are of size 512x512 and stored in a raw format (tif) i.e. the images have never undergone lossy compression.

- **Corel**: This database consists of 8185 images from the Corel database (8). They include images of natural landscapes, people, animals, instruments, buildings, artwork, etc. Although there is no indication of how these images have been acquired, they are very likely to have been scanned from a variety of photos and slides. This database has been previously used in (71). All images are of size 512x512 and stored in a raw format (tif).

- **NRCS**: This database consists of 2375 images from the NRCS Photo Gallery (5).The photos are of natural scenery, e.g. landscape, cornfields, etc. There is no indication of how these photos were acquired. This database has been previously used in (39). All images in this database too are of size 512x512 and stored in a raw format (tif).

The last three databases have been downloaded from (25). We generate the training and testing set for each database separately with a similar process as described in 2.4. For each database, feature set and for each algorithm we train a soft margin support vector machine with gaussian kernel (13) (using LIBSVM (16)). We used the Matlab grid provided by College of Engineering, Iowa State University for feature extraction. It consists of Master - Slave architecture which scheduled the feature extraction jobs on multiple 64 bit computers. Each machine runs on Quad Core Intel Xeon CPU 2.83 GHz and 4GB RAM. This sped up the feature extraction time considerably, as we used another Quad Core Intel Xeon CPU 2.93 GHz and 3GB RAM computer to run the SVMs.

We determined the training parameters of the C-SVMs by grid-search performed on the following multiplicative grid

$$(C, \gamma) \in \left\{ \left(2^i, 2^j\right) | i \in Z, j \in Z \right\}.$$

We compare our results with steganalyzer from feature set proposed in (62), (18) and (54) abbreviating them as Markov324, Markov486 and Merged, respectively. Figure 3.3, 3.4, 3.5 and 3.6 shows the detection accuracy results.

3.3 Discussion of results

We first understand the effect of threshold parameter T on the detection accuracy. Note that different value of T will develop a different POMM and hence we will get a different feature set. We determine the detection accuracy rate for different POMM with $T = 1, 2, 3, 4, 5$ across 4 different database as described in experimental section. It is clear from Table 3.1, 3.2, 3.3 and 3.4 that POMM with $T = 1$ (18 features) and $T = 2$ (50 features) does not capture steganographic changes at lower embedding

		T=1	**T=2**	**T=3**	**T=4**	**T=5**
Jsteg	Cover	92.67	99.75	99.75	99.83	100.00
	0.05 bpc	24.43	98.15	99.58	98.99	98.99
	0.10 bpc	41.62	100.00	100.00	100.00	100.00
	0.20 bpc	77.93	100.00	100.00	100.00	100.00
	0.40 bpc	98.74	100.00	100.00	100.00	100.00
Outguess	Cover	81.89	99.58	99.92	99.66	99.66
	0.05 bpc	37.18	97.39	98.31	99.16	98.90
	0.10 bpc	61.64	99.92	100.00	100.00	100.00
	0.20 bpc	90.37	100.00	100.00	100.00	100.00
F5	Cover	90.31	93.01	93.18	93.43	92.00
	0.05 bpc	29.65	48.19	56.53	50.72	49.71
	0.10 bpc	74.47	95.79	97.39	96.46	95.20
	0.20 bpc	99.92	99.92	100.00	100.00	100.00
	0.40 bpc	100.00	99.75	100.00	100.00	100.00
Steghide	Cover	71.61	95.37	96.04	96.29	96.12
	0.05 bpc	36.82	74.05	74.56	72.11	73.38
	0.10 bpc	40.86	89.81	92.25	91.24	90.73
	0.20 bpc	49.03	97.73	98.99	98.99	99.16
	0.40 bpc	62.59	99.58	99.75	100.00	100.00
JPHide	Cover	83.40	84.92	89.22	91.49	90.90
	0.05 bpc	29.16	37.45	30.68	28.23	25.44
	0.10 bpc	32.94	44.45	39.63	39.46	37.85
	0.20 bpc	51.86	69.71	84.43	88.83	92.22
	0.40 bpc	90.76	96.02	98.39	99.07	99.66

Table 3.1: Detection accuracy results for POMM based features on NRCS database

		T=1	**T=2**	**T=3**	**T=4**	**T=5**
Jsteg	Cover	86.22	99.18	99.75	99.56	99.24
	0.05 bpc	35.08	91.66	94.69	94.56	94.75
	0.10 bpc	50.51	99.30	99.62	99.62	99.62
	0.20 bpc	78.07	99.81	99.87	99.94	99.94
	0.40 bpc	98.48	99.94	100.00	99.87	100.00
Outguess	Cover	74.53	97.47	97.91	97.47	96.65
	0.05 bpc	52.39	89.80	94.14	94.39	93.69
	0.10 bpc	72.71	99.35	99.81	99.81	99.81
	0.20 bpc	94.03	99.93	99.93	100.00	100.00
F5	Cover	85.21	87.29	87.42	85.40	90.14
	0.05 bpc	44.82	52.97	53.29	54.99	45.70
	0.10 bpc	86.09	93.11	93.55	93.99	89.82
	0.20 bpc	99.05	99.43	99.49	99.56	99.37
	0.40 bpc	99.18	99.43	99.49	99.62	99.49
Steghide	Cover	58.85	90.01	90.52	90.14	89.51
	0.05 bpc	56.01	76.80	79.71	80.28	79.14
	0.10 bpc	60.05	87.55	90.39	90.39	89.82
	0.20 bpc	67.13	96.97	98.23	97.98	97.98
	0.40 bpc	77.24	99.56	99.94	99.81	99.75
JPHide	Cover	85.40	85.71	85.40	90.52	91.28
	0.05 bpc	61.90	69.18	66.90	60.95	60.57
	0.10 bpc	63.12	71.28	70.53	64.45	64.71
	0.20 bpc	72.51	80.30	87.97	87.84	89.42
	0.40 bpc	95.89	97.91	99.05	99.24	99.37

Table 3.2: Detection accuracy results for POMM based features on Camera database

		T=1	**T=2**	**T=3**	**T=4**	**T=5**
Jsteg	Cover	87.02	97.80	99.76	99.80	99.95
	0.05 bpc	29.79	97.39	99.61	99.58	99.39
	0.10 bpc	54.72	99.98	100.00	100.00	100.00
	0.20 bpc	86.78	100.00	100.00	100.00	100.00
	0.40 bpc	98.53	100.00	100.00	100.00	100.00
Outguess	Cover	75.81	99.00	99.61	99.80	99.80
	0.05 bpc	42.79	97.48	98.58	98.34	98.34
	0.10 bpc	62.07	100.00	100.00	100.00	100.00
	0.20 bpc	89.89	100.00	100.00	100.00	100.00
F5	Cover	86.19	93.55	92.55	92.18	91.79
	0.05 bpc	27.32	37.51	41.10	40.81	40.74
	0.10 bpc	50.42	88.76	91.30	90.69	90.27
	0.20 bpc	99.07	100.00	100.00	100.00	100.00
	0.40 bpc	99.88	100.00	100.00	100.00	100.00
Steghide	Cover	60.68	92.67	94.26	93.94	94.79
	0.05 bpc	46.29	73.34	74.98	75.66	73.78
	0.10 bpc	49.39	89.10	91.59	91.94	90.32
	0.20 bpc	55.33	98.44	99.17	99.19	99.10
	0.40 bpc	66.01	99.95	100.00	99.98	100.00
JPHide	Cover	79.06	83.68	86.17	85.46	85.24
	0.05 bpc	42.18	42.59	43.26	44.33	44.41
	0.10 bpc	44.52	48.80	52.54	54.94	56.23
	0.20 bpc	54.37	72.63	89.44	92.66	92.88
	0.40 bpc	81.80	99.14	99.93	99.90	99.95

Table 3.3: Detection accuracy results for POMM based features on Corel database

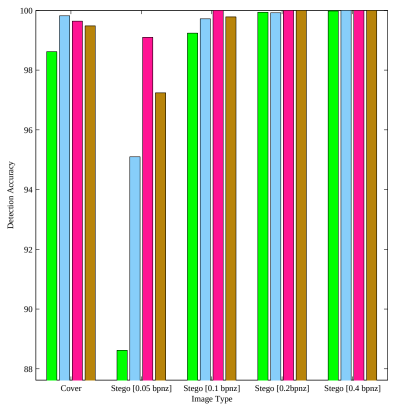
		T=1	**T=2**	**T=3**	**T=4**	**T=5**
Jsteg	Cover	93.18	98.84	99.48	99.46	99.48
	0.05 bpc	33.48	92.72	97.24	96.80	96.38
	0.10 bpc	58.30	99.52	99.78	99.78	99.72
	0.20 bpc	83.48	99.98	100.00	100.00	99.96
	0.40 bpc	97.14	100.00	100.00	100.00	100.00
Outguess	Cover	81.84	98.70	99.38	99.08	99.38
	0.05 bpc	38.63	94.88	96.32	96.90	96.90
	0.10 bpc	62.75	99.84	99.88	99.90	99.92
	0.20 bpc	92.29	100.00	100.00	100.00	100.00
F5	Cover	87.66	90.86	91.12	90.66	89.42
	0.05 bpc	29.16	49.56	51.56	52.66	54.58
	0.10 bpc	66.90	95.90	96.78	96.90	97.18
	0.20 bpc	99.46	99.98	99.98	100.00	99.98
	0.40 bpc	99.78	99.98	99.98	100.00	100.00
Steghide	Cover	66.34	92.48	94.66	93.96	94.34
	0.05 bpc	44.36	74.40	77.28	78.04	77.82
	0.10 bpc	48.60	88.40	91.56	92.14	91.56
	0.20 bpc	55.88	97.36	98.62	98.82	98.74
	0.40 bpc	70.28	99.82	99.88	99.92	99.94
JPHide	Cover	90.34	92.64	94.66	95.14	94.98
	0.05 bpc	64.95	71.73	69.60	70.90	70.80
	0.10 bpc	66.66	74.13	73.25	74.67	75.32
	0.20 bpc	74.18	85.12	94.01	94.25	94.79
	0.40 bpc	92.88	99.14	99.78	99.60	99.58

Table 3.4: Detection accuracy results for POMM based features on Bows2 database

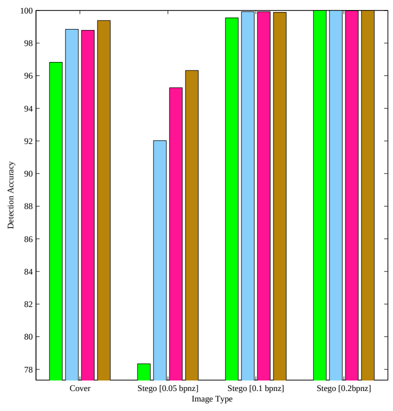
rate which is true across different database. It is also clear that POMM with $T = 3$ performs better than other POMM in smaller database (NRCS and Camera) for JPHide whereas POMM with $T = 5$ performs better on larger database. This trend is also true for F5. For example, POMM with $T = 3$ gives a detection accuracy of 56% for 0.05 bpnz embedding for F5 algorithm whereas POMM with $T = 5$ gives an accuracy of 50%. But the trend is reversed once you go to a larger database. In Bows2 database which has 10000 images, POMM with $T = 5$ gives a detection accuracy rate of 54% for F5 at 0.05 bpnz whereas POMM with $T = 3$ gives 51%. It can also be seen that POMM with $T = 3$ gives an overall better detection for OutGuess irrespective of database size. Performance of POMM steganalyzer with $T \geq 3$ is approximately same for Jsteg and Steghide across different database. It is clear that based on the database size, a POMM can be build by selecting appropriate value of T. Based on above observation, we decide to chose POMM with $T = 3$ to compare our approach with other steganalyzer proposed in the literature.

From Figure 3.3, 3.4, 3.5 and 3.6, it is clear that the steganalyzer based on our proposed feature set clearly beats Markov324 and Makorv486 in all the databases at all the embedding rates for almost every steganography algorithm. Even though the performances of the steganalyzers are very close to 100% for higher embedding rates, their performances vary much more at lower embedding rates. Note that Markov486 is an extension of Markov324 and the added inter block features have helped in boosting the performance. However, our POMM based steganalyzer shows significant improvement in performance at the lower embedding rates compared either Markov scheme. For example, for Bows2 database, Markov324 and Markov486 has a detection accuracy close to 6% whereas POMM based steganalyzer gave a detection accuracy of 71%.

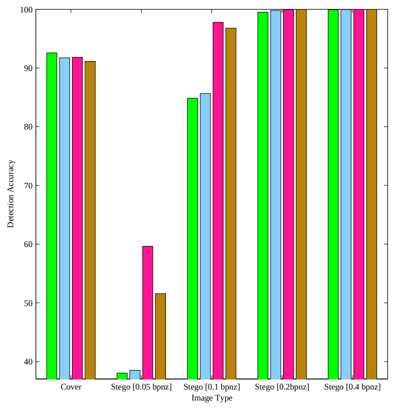
Our proposed steganalyzer has also performed better than Merged for Outguess and Steghide across all the databases whereas Merged performs better at lower embedding rates for F5 and JPHide. Both steganalyzers perform equivalently at higher embedding rates and for detecting Jsteg. For example, the Merged steganalyzer gave a detection accuracy of 55% for Steghide at 0.5 bpnz for Corel database whereas our POMM based steganalyzer gave a detection accuracy of 74%. On the other hand, for the same database, Merged performed better for JPHide at 0.5 bpnz with a detection accuracy of 57% whereas our POMM based steganalyzer detected 44% of the stego images correctly. And this trend holds across different databases.



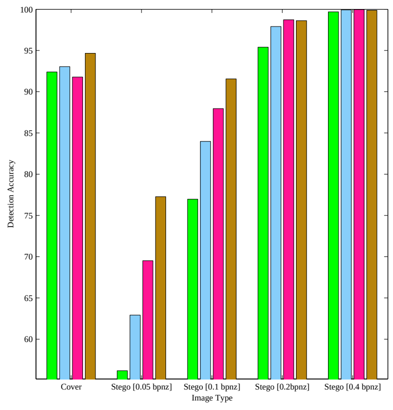
(a) Jsteg



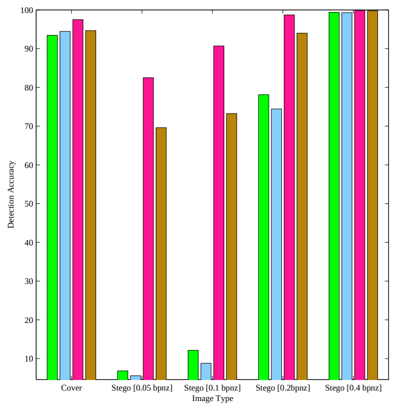
(b) Outguess



(c) F5



(d) Steghide



(e) JPHide

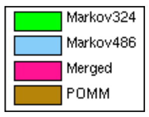
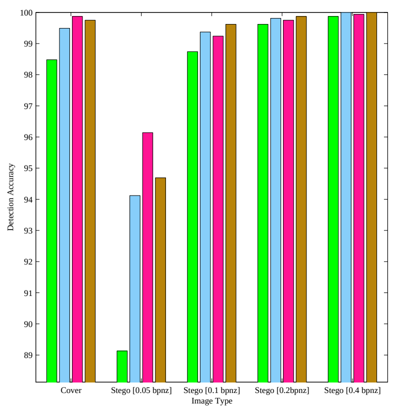
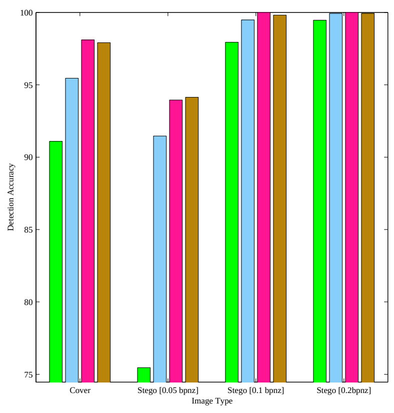


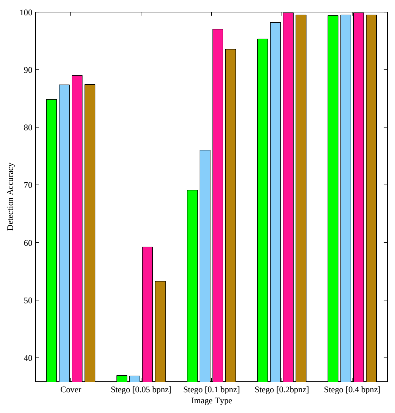
Figure 3.3: Detection accuracy results for different steganalyzers on BOWS2 database



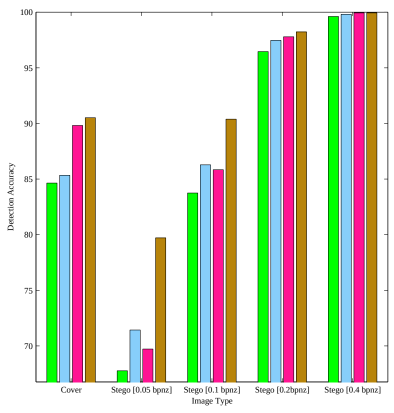
(a) Jsteg



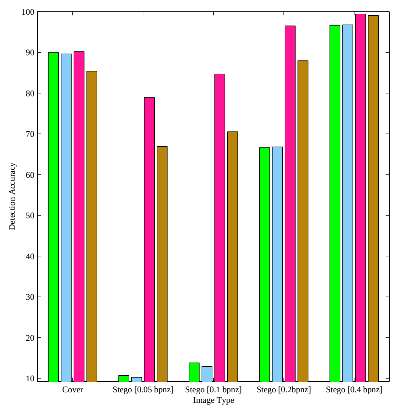
(b) Outguess

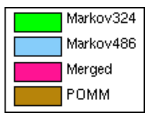


(c) F5



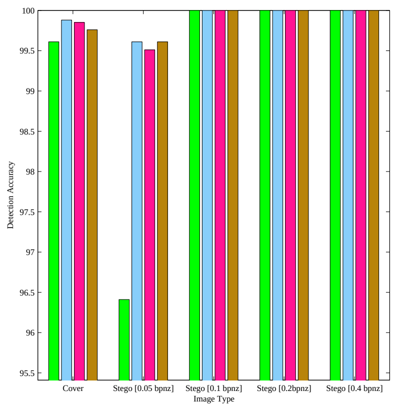
(d) Steghide



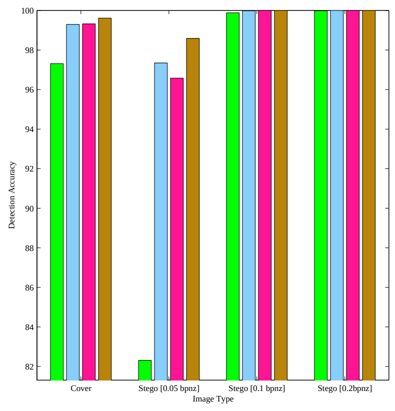


(e) JPHide

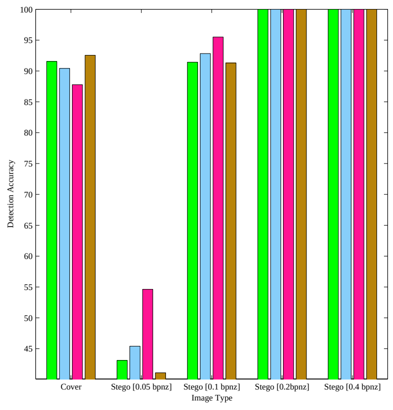
Figure 3.4: Detection accuracy results for different steganalyzers on Camera database



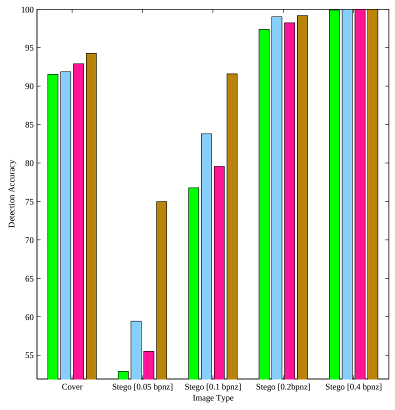
(a) Jsteg



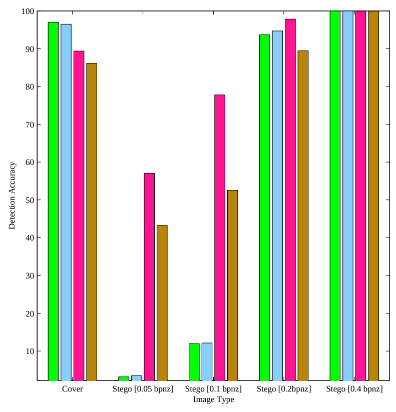
(b) Outguess



(c) F5



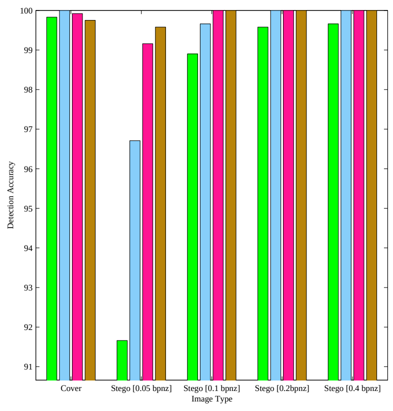
(d) Steghide



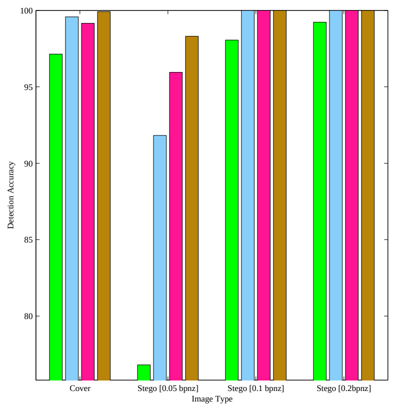
(e) JPHide

🟩 Markov324
🟦 Markov486
🟥 Merged
🟫 POMM

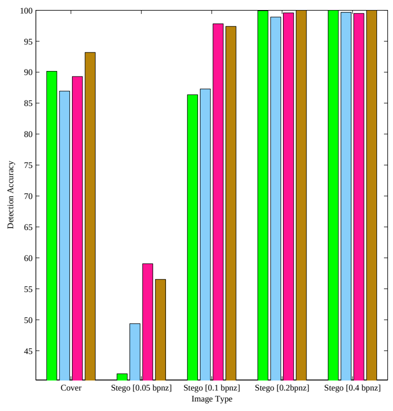
Figure 3.5: Detection accuracy results for different steganalyzers on Corel database



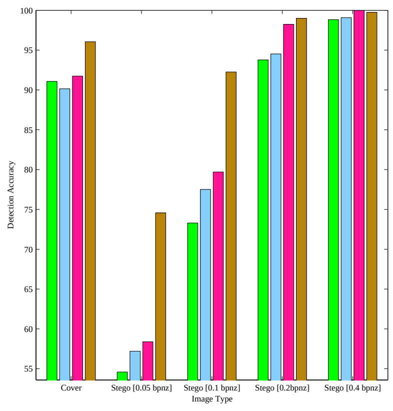
(a) Jsteg



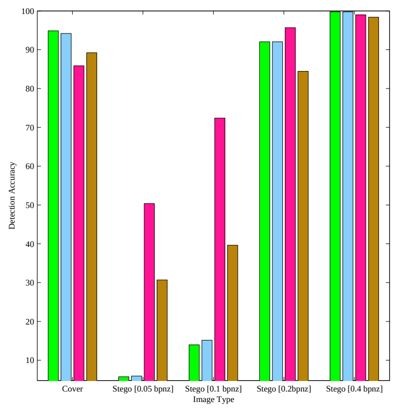
(b) Outguess



(c) F5



(d) Steghide



(e) JPHide

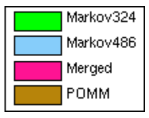


Figure 3.6: Detection accuracy results for different steganalyzers on NRCS database

CHAPTER 4 CANVASS - A STEGANOGRAPHIC FORENSIC TOOL FOR JPEG IMAGES

There is a growing concern within the community that steganography is being used for illicit purposes. The New York Times, USA Today and United States Institute of Peace have reported that terrorists may be using steganography and cryptography on the web as a means of covert communication (37; 43; 66). A recent report from National Institute of Justice encourages investigators to look for steganographic information while dealing with child abuse or exploitation and terrorism cases (1). While steganalysis algorithms are abound in the academic literature, there are few software programs that address the needs of local police departments who perform computer forensic functions for steganalysis.

Two major stego-detection tools in existence today are StegoSuite and StegDetect. StegoSuite was developed by WetStone Technologies for the U.S. Air Force. This software is expensive and hence is not readily available to state police forensics labs whose budget does not allow purchase of these expensive software. In 2001, Neil Provos developed StegDetect (58) to perform steganalysis on suspected stego images. Using this software, he analyzed millions of JPEG images from sites like eBay (59) and USENET (57), but was unable to detect a single image with hidden data in it.
One of the objectives of this project, as mentioned in the original research proposal submitted to Midwest Forensics Resource Center (MFRC), Ames Laboratory, is to develop a software that address the needs of local police departments who perform computer forensic functions for steganalysis. Canvass is a cross-platform software that has been designed after several meetings with Internet Crime Against Children Lab (ICAC), Iowa investigators and understanding their requirement. It is developed in Java with a graphical user interface which implements the steganalzer proposed in Chapter 3 earlier. The current version 1.0 is shown in Figure 4.1 and provides following features:

- Ability to process multiple images with one command. User can specify the source of images from following locations:

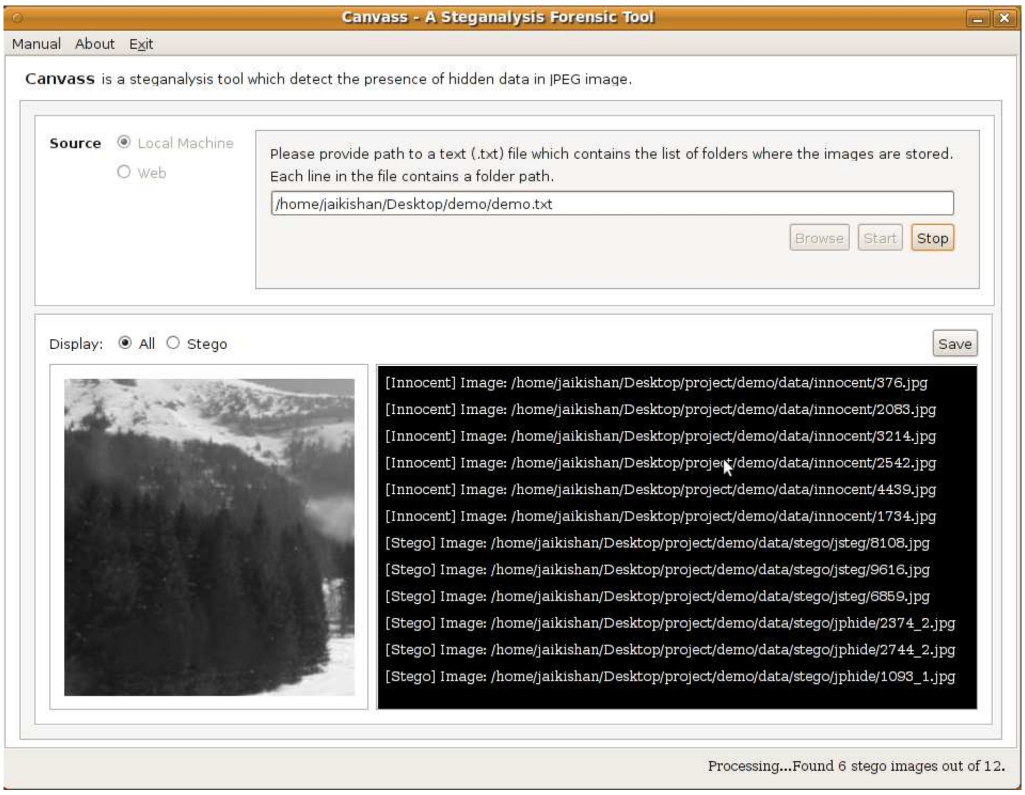
  - *Local Machine*: User can specify multiple images located in different folders on the local machine and the software will classify each image against the steganographic algorithms present in the system.
  - *Web*: User can specify a web address to steganalyze jpeg images on the internet starting from the web address specified. Given a link, it searches for jpeg images on the page, and stores all the links on that page which it has not visited. It then recursively performs a search for images on those links in breadth first manner until there are no more new links to visit.

- Displays detailed processing information in real time. It shows various information such as steganography algorithm used, time of processing along with other useful information. This is shown in Figure 4.2.
- Provides option to save the processing information at any time. A sample report is shown in 4.3c.
- Ability to stop the processing in between.
- Displays image for visual inspection.
- Ability to run on multiple platforms. In this version, it is currently supported for Windows and *nix systems.

However, the current version only accepts JPEG images with width≥ 250 and height≥ 250. This restriction ensures that features contains enough data to characterize the image as cover or stego. This software will be made available from MFRC, Ames Laboratory for limited distribution to recognized police departments.

4.1 Implementation Details

A mutli-class classifier is implemented in Canvass with $\binom{n}{2}$ binary classifiers where n is the number of classes. In our case, $n = 6$ for cover, jsteg, outguess, f5, steghide and jphide. For an unknown image, features are extracted and passed through all the binary classifiers and assign it the class which appeared maximum time in all the binary classifiers. Model-View-Controller (MVC) architecture has been used to design this software. Due to this reason, it can be easily extended using a different steganalyzer from backend, that is, if additional binary classifiers are added.



Canvass is a steganalysis tool which detect the presence of hidden data in JPEG image.

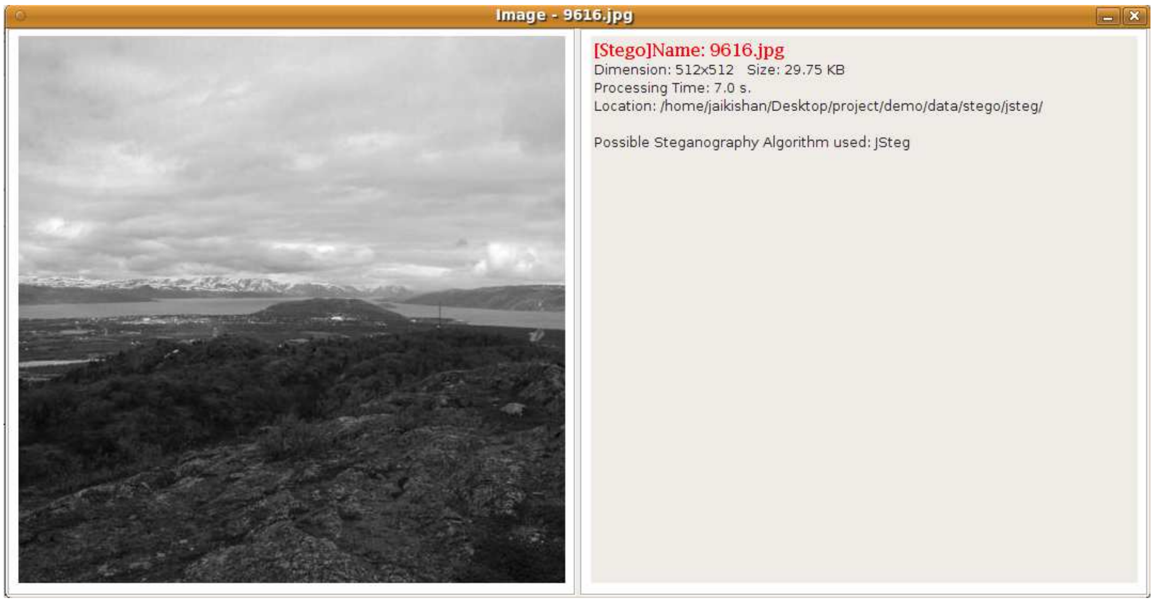
Please provide path to a text (.txt) file which contains the list of folders where the images are stored. Each line in the file contains a folder path.

/home/jaikishan/Desktop/demo/demo.txt

```
[Innocent] Image: /home/jaikishan/Desktop/project/demo/data/innocent/376.jpg
[Innocent] Image: /home/jaikishan/Desktop/project/demo/data/innocent/2083.jpg
[Innocent] Image: /home/jaikishan/Desktop/project/demo/data/innocent/3214.jpg
[Innocent] Image: /home/jaikishan/Desktop/project/demo/data/innocent/2542.jpg
[Innocent] Image: /home/jaikishan/Desktop/project/demo/data/innocent/4439.jpg
[Innocent] Image: /home/jaikishan/Desktop/project/demo/data/innocent/1734.jpg
[Stego] Image: /home/jaikishan/Desktop/project/demo/data/stego/jsteg/8108.jpg
[Stego] Image: /home/jaikishan/Desktop/project/demo/data/stego/jsteg/9616.jpg
[Stego] Image: /home/jaikishan/Desktop/project/demo/data/stego/jsteg/6859.jpg
[Stego] Image: /home/jaikishan/Desktop/project/demo/data/stego/jphide/2374_2.jpg
[Stego] Image: /home/jaikishan/Desktop/project/demo/data/stego/jphide/2744_2.jpg
[Stego] Image: /home/jaikishan/Desktop/project/demo/data/stego/jphide/1093_1.jpg
```

Processing...Found 6 stego images out of 12.

Figure 4.1: Canvass - A software package for steganalysis



[Stego]Name: 9616.jpg
Dimension: 512x512 Size: 29.75 KB
Processing Time: 7.0 s.
Location: /home/jaikishan/Desktop/project/demo/data/stego/jsteg/

Possible Steganography Algorithm used: JSteg

Figure 4.2: Canvass - Detailed Screen

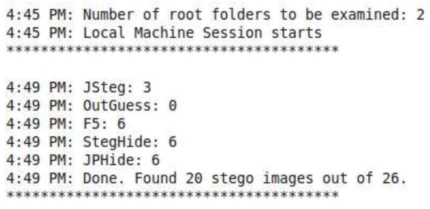
4.2 Installer

Canvass comes with a jar excecutable file - Canvass.jar along with several other folders. To run this software, user must have Java installed on their machine. The software can be started by either double clicking on the software or from command line as

```
java -jar Canvass.jar
```

It comes with the following folder:

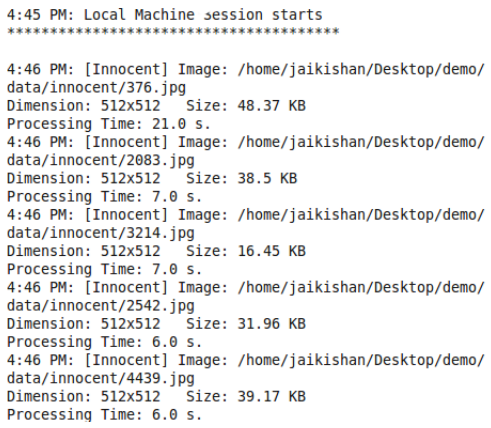
- log: This folder contains various messages generated during the execution of the software. It stores all the error messages under **error** sub folder which can help to debug the application. It also stores the results for every image under **process** sub folder so that results are not lost in case of any unexpected failure of software or system. Finally it also stores a summary of the processing information under **general** sub folder. A sample general and log file is shown in Figure 4.3a and 4.3b.
- external: This folder contains compiled code for feature extraction for Windows and Linux machine. For every image, Canvass calls this code to extract the features from image and pass them to SVM for testing purpose.
- lib: This folder contains necessary Java Swing libraries necessary to support the application.
- data: This folder contains following sub-folders:
  - config: This folder contains various config files, logo of the software and other related files.
  - manual: This folder contains a manual for easy use of the software.
  - steganalyzer: This folder contains SVM model files for different binary SVM's. Files for every binary classifier is stored in sub folder **i_j**. For example **0_1** folder represents SVM model files for binary SVM between cover and jsteg class. We use the following numbering convention for classes: 0 - Cover, 1 - Jsteg, 2- OutGuess, 3 - F5, 4 - Steghide, 5 - JPHide. In the future, if the capability of Canvass needs to be extended to incorporate a new steganographic algorithm, all that is required is to build binary SVM's of the new class with every other existing class.
  - workspace: This folder is used by the software to store any temporary files and images computed during the execution of Canvass.



```
4:45 PM: Number of root folders to be examined: 2
4:45 PM: Local Machine Session starts
***************************************

4:49 PM: JSteg: 3
4:49 PM: OutGuess: 0
4:49 PM: F5: 6
4:49 PM: StegHide: 6
4:49 PM: JPHide: 6
4:49 PM: Done. Found 20 stego images out of 26.
***************************************
```

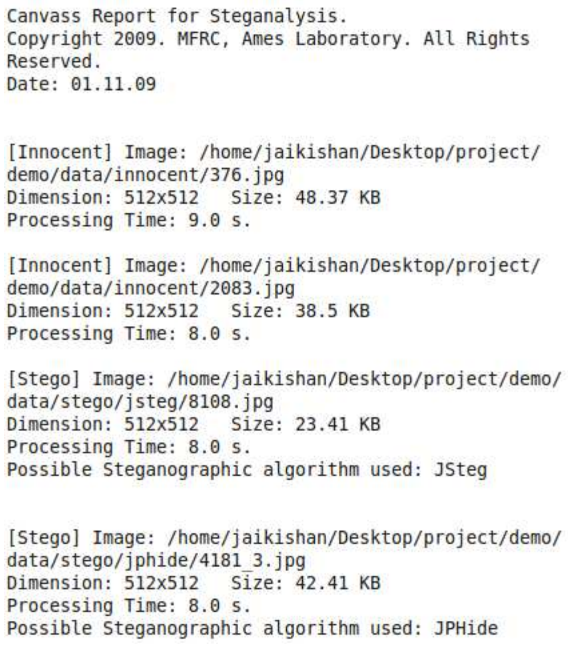
(a) General Log



```
4:45 PM: Local Machine Session starts
***************************************

4:46 PM: [Innocent] Image: /home/jaikishan/Desktop/demo/
data/innocent/376.jpg
Dimension: 512x512   Size: 48.37 KB
Processing Time: 21.0 s.
4:46 PM: [Innocent] Image: /home/jaikishan/Desktop/demo/
data/innocent/2083.jpg
Dimension: 512x512   Size: 38.5 KB
Processing Time: 7.0 s.
4:46 PM: [Innocent] Image: /home/jaikishan/Desktop/demo/
data/innocent/3214.jpg
Dimension: 512x512   Size: 16.45 KB
Processing Time: 7.0 s.
4:46 PM: [Innocent] Image: /home/jaikishan/Desktop/demo/
data/innocent/2542.jpg
Dimension: 512x512   Size: 31.96 KB
Processing Time: 6.0 s.
4:46 PM: [Innocent] Image: /home/jaikishan/Desktop/demo/
data/innocent/4439.jpg
Dimension: 512x512   Size: 39.17 KB
Processing Time: 6.0 s.
```

(b) Process Log



```
Canvass Report for Steganalysis.
Copyright 2009. MFRC, Ames Laboratory. All Rights
Reserved.
Date: 01.11.09


[Innocent] Image: /home/jaikishan/Desktop/project/
demo/data/innocent/376.jpg
Dimension: 512x512   Size: 48.37 KB
Processing Time: 9.0 s.

[Innocent] Image: /home/jaikishan/Desktop/project/
demo/data/innocent/2083.jpg
Dimension: 512x512   Size: 38.5 KB
Processing Time: 8.0 s.

[Stego] Image: /home/jaikishan/Desktop/project/demo/
data/stego/jsteg/8108.jpg
Dimension: 512x512   Size: 23.41 KB
Processing Time: 8.0 s.
Possible Steganographic algorithm used: JSteg


[Stego] Image: /home/jaikishan/Desktop/project/demo/
data/stego/jphide/4181_3.jpg
Dimension: 512x512   Size: 42.41 KB
Processing Time: 8.0 s.
Possible Steganographic algorithm used: JPHide
```

(c) Sample report

Figure 4.3: (a)General and (b)Process Log file. (c) Sample report

CHAPTER 5 CONCLUSION AND FUTURE WORK

In this thesis, we introduced the use of Partially Ordered Markov Models for the first time in the field of Steganalysis. We have successfully developed a sound theoretical framework for steganalysis from which rich properties can be derived. Our experiments in Chapter 3 showed that a detection accuracy of close to 100 % can be achieved for higher embedding rates and has a comparable or better results at lower embedding rated compared to other steganalyzers. We also developed a technique to reduce feature space for steganalysis using Mahalanobi distance and showed that it is possible to develop a classifier with very few number of features that perform as well as classifier built with full suite of features. This will help to build efficient steganalyzers with lower training computation time. We have also developed a software for steganalysis for police departments to make this research accessible to the investigators and we hope that this will provide increased check on illegal use of images for covert communication.

Even though the steganalyzer proposed has a high detection accuracy at higher embedding rates, it is still not very high for lower embedding. We have shown that with some global DCT features, the detection accuracy can further be boosted. In future, one can look into a complementary set of features that along with POMM can give a better detection accuracy rates at lower embedding. Since POMM provides a closed form for calculating joint pdf, new techniques can be developed by assuming a parameterized model of POMM to estimate the message length by maximizing the joint pdf conditioned on parameters of the model. Another way to estimate the message length could be to use the proposed features and length of message embedded for SVM regression to predict the length of messages in unknown images (55).

BIBLIOGRAPHY

[1] Electronic crime scene investigation: A guide for first responders, second edition, April 2008. National Institute of Justice. Available online at `http://www.ojp.usdoj.gov/nij/publications/ecrime-guide-219941/chapter7-219941.pdf`.

[2] Steghide. Available online at `http://steghide.sourceforge.net/`.

[3] Jphide&seek. Available online at `http://linux01.gwdg.de/~alatham/stego.html`.

[4] Jsteg. Available online at `http://zooid.org/paul/crypto/jsteg/`.

[5] Natural resources conservation service photo gallery. Available online at `http://photogallery.nrcs.usda.gov`.

[6] Stegoarchive. Available online at `http://www.stegoarchive.com/`.

[7] Msu. Available online at `http://www.compression.ru/video/`.

[8] Corel stock photo library 3.

[9] K. Abend, T. Harley, and L. Kanal. Classification of binary random patterns. *IEEE Transactions on Information Theory*, volume 11(4):538–544, 1965.

[10] R. Anderson and F. Petitcolas. On the limits of steganography. *IEEE Journal on Selected Areas in Communications*, 16(4):474–481, May 1998.

[11] T. Aura. Practical invisibility in digital communication. In *Lecture Notes in Computer Science, vol.1174*, pages 265–278. SpringerVerlag, 1996.

[12] I. Avcibas, N. Memon, and B. Sankur. Steganalysis of watermarking techniques using image quality metrics. In *Security and Watermarking of Multimedia Contents VI*, volume 5306, pages 83–97. SPIE, 2001.

[13] C. J. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[14] C. Cachin. An information-theoretic model for steganography. In *Information Hiding*, volume 1525 of *Lecture Notes in Computer Science*, pages 306–318. Springer Berlin / Heidelberg, 1998.

[15] A. Caprihan, G. D. Pearlson, and V. D. Calhoun. Application of principal component analysis to distinguish patients with schizophrenia from healthy controls based on fractional anisotropy measurements. *Neuroimage*, 42:675–682, 2008.

[16] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[17] W.-C. Chang. On using principal components before separating a mixture of two multivariate normal distributions. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 32(3):267–275, 1983.

[18] C. Chen and Y. Q. Shi. Jpeg image steganalysis utilizing both intrablock and interblock correlations. In *IEEE International Symposium on Circuits and Systems*, pages 3029–3032, May 2008.

[19] C. Chen and Y. Q. Shi. Jpeg image steganalysis utilizing both intrablock and interblock correlations. In *IEEE International Symposium on Circuits and Systems*, pages 3029–3032, May 2008.

[20] J. Cheng, A. Kot, and S. Rahardja. Steganalysis of binary cartoon image using distortion measure. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages II–61–II–64, 2007.

[21] I. Cox, M. Miller, J. Bloom, J. Fridrich, and T. Kalker. *Digital Watermarking and Steganography.* Elsevier, 2007.

[22] N. Cressie and J. Davidson. Image analysis with partially ordered markov models. *Computational Statistics and Data Analysis*, 29:1–26, 1998.

[23] J. Davidson, A. Talukder, and N. Cressie. Texture analysis using partially ordered markov models. In *IEEE International Conference on Image Processing*, volume 3, pages 402–406, Nov 1994.

[24] J. L. Davidson, N. Cressie, and X. Hua. Texture synthesis and pattern recognition for partially ordered markov models. *Pattern Recognition*, 32(9):1475–1505, 1999.

[25] G. Doerr. Image database for steganalysis studies [online]. `http://www.cs.ucl.ac.uk/staff/G.Doerr/`, 2007.

[26] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley-Interscience, 2001.

[27] J. Fridrich. Feature-based steganalysis for JPEG images and its implications for future design of steganographic schemes. In *Information Hiding*, volume 3200 of *Lecture Notes in Computer Science*, pages 67–81. Springer Berlin / Heidelberg, 2005.

[28] D. Fu, Y. Shi, D. Zou, and G. Xuan. Jpeg steganalysis using empirical transition matrix in block dct domain. In *IEEE 8th Workshop on Multimedia Signal Processing*, pages 310–313, October 2006.

[29] M. Goljan, J. Fridrich, and T. Holotyak. New blind steganalysis and its implications. In E. J. D. III and P. W. Wong, editors, *Security, Steganography, and Watermarking of Multimedia Contents VIII*, volume 6072, page 607201. SPIE, 2006.

[30] B. Haasdonk and E. Pkalska. Classification with kernel mahalanobis distance classifiers. In *Advances in Data Analysis, Data Handling and Business Intelligence*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 351–361. Springer Berlin Heidelberg, 2009.

[31] J. J. Harmsen and W. A. Pearlman. Steganalysis of additive-noise modelable information hiding. In E. J. D. III and P. W. Wong, editors, *Security and Watermarking of Multimedia Contents V*, volume 5020, pages 131–142. SPIE, 2003.

[32] J. Helterbrand, M. Cressie, and J. Davidson. Optimal closed boundary identification in gray-scale imagery. *Journal of Mathematical Imaging and Vision*, volume 5(3):179–206, 1995.

[33] T. Holotyak, J. Fridrich, and S. Voloshynovskiy. Blind statistical steganalysis of additive steganography using wavelet higher order statistics. In *Communications and Multimedia Security*, volume 3677 of *Lecture Notes in Computer Science*, pages 273–274. Springer Berlin / Heidelberg, 2005.

[34] F. Huang, J. Huang, and B. Li. Universal jpeg steganalysis based on microscopic and macroscopic calibration. In *15th IEEE International Conference on Image Processing*, pages 2068–2071, October 2008.

[35] A. K. Jain, R. P. Duin, and J. Mao. Statistical pattern recognition: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, January 2000.

[36] E. John, L. Prichep, J. Fridman, and P. Easton. Neurometrics: computer-assisted differential diagnosis of brain dysfunctions. *Science*, 239(4836):162–169, 1988.

[37] J. Kelly. Terrorist instructions hidden online. USA Today, February 2001.

[38] A. Ker. Quantitive evaluation of pairs and rs steganalysis. In *Security, Steganography, and Watermarking of Multimedia Contents III*, volume 3677, pages 273–274. SPIE, 2004.

[39] A. Ker. Steganalysis of lsb matching in grayscale images. *IEEE Signal Processing Letters*, 12(6):441–444, June 2005.

[40] A. D. Ker and I. Lubenko. Feature reduction and payload location with wam steganalysis. In *Media Forensics and Security*, volume 7254, page 72540A. SPIE, 2009.

[41] R. Kindermann and J. L. Snell. *Markov Random Fields and Their Applications.* AMS, 1980.

[42] J. Kodovský and J. Fridrich. Calibration revisited. In *MM&#38;Sec '09: Proceedings of the 11th ACM workshop on Multimedia and security*, pages 63–74, New York, NY, USA, 2009. ACM.

[43] G. Kolata. Veiled messages of terrorists may lurk in cyberspace. USA Today, October 2001.

[44] Q. Liu, A. H. Sung, Z. Chen, and J. Xu. Feature mining and pattern classification for steganalysis of lsb matching steganography in grayscale images. *Pattern Recognition*, 41(1):56–66, 2008.

[45] X.-Y. Luo, D.-S. Wang, P. Wang, and F.-L. Liu. A review on blind detection for image steganography. *Signal Processing*, 88(9):2138–2157, 2008.

[46] S. Lyu and H. Farid. Steganalysis using higher-order image statistics. *IEEE Transactions on Information Forensics and Security*, 1(1):111–119, March 2006.

[47] B. F. Manly. *Multivariate Statistical Methods: A Primer, Third Edition.* Chapman and Hall Ltd, 2004.

[48] Y. Miche, P. Bas, A. Lendasse, C. Jutten, and O. Simula. Reliable steganalysis using a minimum set of samples and features. *EURASIP Journal on Information Security*, 2009(901381):13, 2009.

[49] Y. Miche, B. Roue, A. Lendasse, and P. B. and. A feature selection methodology for steganalysis. In *Multimedia Content Representation, Classification and Security*, volume 4105 of *Lecture Notes in Computer Science*, pages 49–56. Springer Berlin / Heidelberg, 2006.

[50] J. Mielikainen. Lsb matching revisited. *Signal Processing Letters, IEEE*, 13(5):285–287, May 2006.

[51] D. V. Nguyen and D. M. Rocke. On partial least squares dimension reduction for microarray-based classification: a simulation study. *Computational Statistics & Data Analysis*, 46(3):407–425, 2004.

[52] T. Pevný and J. Fridrich. Towards multi-class blind steganalyzer for JPEG images. In *Digital Watermarking*, volume 3710 of *Lecture Notes in Computer Science*, pages 39–53. Springer Berlin / Heidelberg, 2005.

[53] T. Pevný and J. Fridrich. Multi-class blind steganalysis for jpeg images. In E. J. D. III and P. W. Wong, editors, *Security, Steganography, and Watermarking of Multimedia Contents VIII*, volume 6072, page 60720O. SPIE, 2006.

[54] T. Pevný and J. Fridrich. Merging markov and dct features for multi-class jpeg steganalysis. In E. J. D. III and P. W. Wong, editors, *Security, Steganography, and Watermarking of Multimedia Contents IX*, volume 6505. SPIE, 2007.

[55] T. Pevny, J. Fridrich, and A. D. Ker. From blind to quantitative steganalysis. In E. J. D. III, J. Dittmann, N. D. Memon, and P. W. Wong, editors, *Media Forensics and Security*, volume 7254, page 72540C. SPIE, 2009.

[56] N. Provos. Defending against statistical steganalysis. In *Proceedings of the 10th conference on USENIX Security Symposium*, pages 24–24, Berkeley, CA, USA, 2001. USENIX Association.

[57] N. Provos. Scanning usenet for steganography. `http://www.citi.umich.edu/u/provos/stego/usenet.php`, 2001.

[58] N. Provos. Steganography detection with stegdetect. `http://www.outguess.org/detection.php`, 2004.

[59] N. Provos and P. Honeyman. Detecting steganographic content on the internet. Technical report, In ISOC NDSS02, 2001.

[60] J. Qin, X. Sun, X. Xiang, and C. Niu. Principal feature selection and fusion method for image steganalysis. *Journal of Electronic Imaging*, 18(3):033009, 2009.

[61] J. W. Robinson and R. S. Hoffmann. Geographical and interspecific cranial variation in big-eared ground squirrels (spermophilus): A multivariate study. *Systematic Zoology*, 24(1):79–88, March 1975.

[62] Y. Q. Shi, C. Chen, and W. Chen. A markov process based approach to effective attacking JPEG steganography. In *Information Hiding*, volume 4437 of *Lecture Notes in Computer Science*, pages 249–264. Springer Berlin / Heidelberg, 2007.

[63] J. Shlens. A tutorial on principal component analysis. `http://www.snl.salk.edu/~shlens/pub/notes/pca.pdf`, 2009.

[64] R. Wang and W. Zhao. Improvement of tnd steganalysis based on image complexity. In *9th International Conference on Signal Processing*, pages 2777–2781, October 2008.

[65] Y. Wang and P. Moulin. Optimized feature extraction for learning-based image steganalysis. *IEEE Transactions on Information Forensics and Security*, 2(1):31–45, March 2007.

[66] G. Weimann. How modern terrorism uses the internet. Special Report 116, United States Institute of Peace, March 2004.

[67] A. Westfeld. Reproducible signal processing (bows2 challenge image database, public).

[68] A. Westfeld. F5: A steganographic algorithm. In *Information Hiding*, volume 2137 of *Lecture Notes in Computer Science*, pages 289–302. Springer Berlin / Heidelberg, 2001.

[69] G. Xuan, X. Cui, Y. Shi, W. Chen, X. Tong, and C. Huang. Jpeg steganalysis based on classwise non-principal components analysis and multi-directional markov model. In *IEEE International Conference on Multimedia and Expo.*, pages 903–906, July 2007.

[70] G. Xuan, X. Zhu, P. Chai, Z. Zhang, Y. Q. Shi, and D. Fu. Feature selection based on the bhattacharyya distance. *International Conference on Pattern Recognition*, 4:957, 2006.

[71] J. Zhang, I. Cox, and G. Doerr. Steganalysis for lsb matching in images with high-frequency noise. In *IEEE 9th Workshop on Multimedia Signal Processing, 2007*, pages 385–388, October 2007.