

Air Force Institute of Technology

AFIT Scholar

Faculty Publications

2-2005

A New Blind Method for Detecting Novel Steganography

Brent T. McBride

Air Force Institute of Technology

Gilbert L. Peterson

Air Force Institute of Technology

Steven C. Gustafson

Air Force Institute of Technology

Follow this and additional works at: <https://scholar.afit.edu/facpub>



Part of the [Computer Sciences Commons](#)

Recommended Citation

McBride, B. T., Peterson, G. L., & Gustafson, S. C. (2005). A new blind method for detecting novel steganography. *Digital Investigation*, 2(1), 50–70. <https://doi.org/10.1016/j.diin.2005.01.003>

This Article is brought to you for free and open access by AFIT Scholar. It has been accepted for inclusion in Faculty Publications by an authorized administrator of AFIT Scholar. For more information, please contact AFIT.ENWL.Repository@us.af.mil.

A New Blind Method for Detecting Novel Steganography

Brent T. McBride

BRENT.MCBRIDE@MAXWELL.AF.MIL

*Department of Electrical and Computer Engineering
Air Force Institute of Technology
2950 Hobson Way
Wright-Patterson AFB, OH 45433-7765*

Gilbert L. Peterson

GILBERT.PETERSON@AFIT.EDU

*Department of Electrical and Computer Engineering
Air Force Institute of Technology
2950 Hobson Way
Wright-Patterson AFB, OH 45433-7765*

Steven C. Gustafson

STEVEN.GUSTAFSON@AFIT.EDU

*Department of Electrical and Computer Engineering
Air Force Institute of Technology
2950 Hobson Way
Wright-Patterson AFB, OH 45433-7765*

Keywords: Steganography, steganalysis, blind classification, jpeg.

Abstract

Steganography is the art of hiding a message in plain sight. Modern steganographic tools that conceal data in innocuous-looking digital image files are widely available. The use of such tools by terrorists, hostile states, criminal organizations, etc., to camouflage the planning and coordination of their illicit activities poses a serious challenge. Most steganography detection tools rely on signatures that describe particular steganography programs. Signature-based classifiers offer strong detection capabilities against known threats, but they suffer from an inability to detect previously-unseen forms of steganography. Novel steganography detection requires an anomaly-based classifier.

This paper describes and demonstrates a blind classification algorithm that uses hyper-dimensional geometric methods to model steganography-free jpeg images. The geometric model, comprising one or more convex polytopes, hyper-spheres, or hyper-ellipsoids in the attribute space, provides superior anomaly detection compared to previous research. Experimental results show that the classifier detects, on average, 85.4% of Jsteg steganography images with a mean embedding rate of 0.14 bits per pixel, compared to previous research that achieved a mean detection rate of just 65%. Further, the classification algorithm creates models for as many training classes of data as are available, resulting in a hybrid anomaly/signature or signature-only based classifier, which increases Jsteg detection accuracy to 95%.

1 Introduction

The word steganography comes from the Greek words *steganos* and *graphia*, which together mean "hidden writing" [1]. Steganography is the art of hiding a message in plain sight. In the digital realm, it involves embedding a secret message file into an inconspicuous cover file, such as a jpeg image. The digital steganography process has three basic components: 1) The data to be hidden, 2) the cover file, in which the secret data is to be embedded, and 3) the resulting stego file.

Digital images make popular cover files because they are ubiquitous and potentially contain large areas of space in which to hide data. Steganography programs, many of which are freely available on the Internet, are appealing to those who wish to conceal their communications. When terrorists, criminals, and other hostile entities use steganography to conceal the planning and coordination of their illicit activities, it becomes more difficult to detect and counter them. Further, steganography can be used to defeat traditional safeguards designed to restrict the distribution of sensitive information.

Those who hide communications through steganography are opposed by those who wish to discover such communications. The field devoted to defeating steganography is known as *steganalysis*. The first goal of steganalysis is detecting the presence of steganography so that the secret message may be extracted from the cover file, spoofed, and/or corrupted. Currently, the most widely-available steganalysis tools are signature-based. These tools, trained to effectively recognize the footprints of known steganography programs, suffer from a general inability to detect previously-unseen steganographic techniques. This limitation is of concern because the number of steganography tools grows as data-embedding techniques evolve to foil current detection schemes. Further, organizations with sufficient resources (including hostile nation-states, terrorist organizations, and criminal entities) may create and use homegrown steganography tools. Thus, it is not possible to collect examples of all steganography tools that currently exist or that will ever exist from which to create an all-encompassing signature-based detector.

To overcome this drawback, an alternative approach to steganalysis attempts to detect steganography by creating a blind model of clean (steganography-free) files such that a stego-file will not match the clean file model and be declared anomalous. An anomaly detector recognizes deviations from normalcy and can thus detect novel steganography that would be missed by a signature detector.

1.1 Research Objectives and Scope

This paper focuses on the creation of a blind classification method that uses hyper-dimensional geometric constructs, such as polytopes, hyper-spheres, and hyper-ellipsoids, to create a class model of a clean image without referencing other classes (i.e., various kinds of stego images). The principles of this hyper-geometric classifier may be expanded to produce a hybrid signature/anomaly or signature-only based classifier by creating class models for existing steganography programs. This research focuses on popular steganographic embedding in jpeg images. However, the principles of the hyper-geometric classification paradigm are applicable to other cover file types by modifying the file attributes employed.

Much image steganalysis research focuses on discovering new image features to better discriminate between clean and stego images. This paper leverages a sampling of this existing research and focuses instead on developing a new classification paradigm to operate on known image features. As newer and better features are discovered, they may also be incorporated into the classification framework.

2 Background

The success of information hiding depends on a number of factors, including the characteristics of the cover file and the embedding technique. Figure 1 shows an example of a simple text message hidden inside a grayscale image. To the naked eye, the stego image appears indistinguishable from the original cover image. There are several methods whereby data may be covertly embedded into a cover image, and LSB and DCT encoding are two of the most prevalent.



Figure 1. Example of Image Steganography

2.1 Least Significant Bit (LSB) Encoding

Pixel LSB encoding is one of the simplest embedding techniques used for digital image steganography [2]. A digital image is represented by a two dimensional array of pixels. Each pixel maps to a single grayscale, or several color, values. Pixel LSB encoding changes only the very last, or least significant, bit of the pixel value(s), in the hope that the resulting change to the cover image is imperceptible. Consider the following trivial example: Suppose we have a 2x2 pixel image, where each of the four pixels has one of 256 possible grayscale values (8 bits per pixel); this is our cover image. We can encode up to one bit per pixel. Suppose we wish to hide the bit stream 0110. Moving from left to right and top to bottom, we replace the least significant bit of each pixel with a bit from the secret bit stream, as shown in Figure 2. It is possible to increase cover image storage capacity by also embedding data into more significant pixel bits, but doing so is not wise. As shown in Figure 3, the impact on the cover image increases drastically with the significance of the flipped bits.

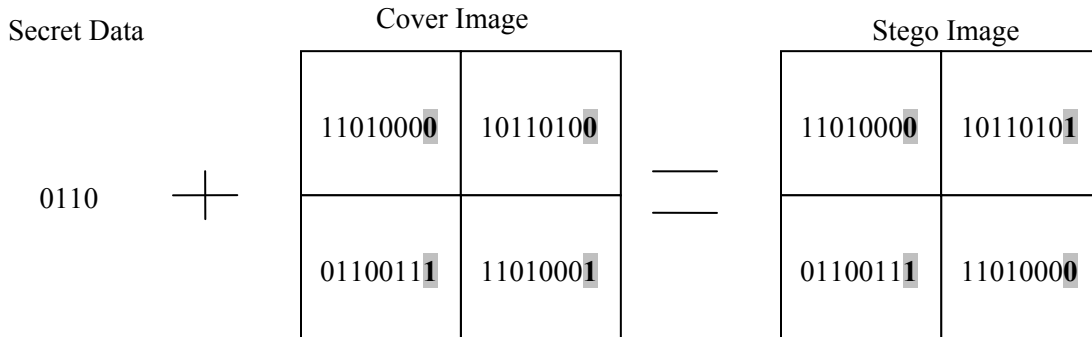


Figure 2. LSB Encoding Example

Encoding data into pixel LSBs is appropriate for uncompressed or losslessly compressed images. Lossy-compressing such a stego image may corrupt the embedded data by changing the pixel values. Most images posted on web pages or otherwise sent electronically over the Internet are lossy-compressed to conserve bandwidth and speed transmission times. Therefore, the transmission of an uncompressed image stands out and undermines the inconspicuousness that steganography requires. A file format that blends in with background traffic makes a more suitable medium for steganographic communication.



Figure 3. Pixel LSB Flipping Impact

The jpeg image format, defined by and named after the Joint Photographic Experts Group, is currently the most prevalent image storage format [3]. Lossy-compressed jpegs achieve good compression ratios while minimizing image quality degradation, making them the most popular choice for electronic image transmission. The ubiquitousness of lossy-compressed jpegs on the Internet makes them ideal cover images for the steganographic transmission of secret data. However, their lossy-compression precludes embedding data into pixel LSBs. A different encoding paradigm is employed to embed data into such images.

2.2 Discrete Cosine Transform (DCT) Encoding

Lossy jpeg compression exploits the fact that the human eye is less sensitive to higher frequency information (such as edges and noise) in an image than to lower frequencies. The jpeg encoding process [4] is illustrated in Figure 4. First, the raw image is broken into blocks, usually sized to 8-by-8 pixels. 64 DCT coefficients are computed for each block, converting the block from the spatial domain to the frequency domain. The higher frequency DCT coefficients are then rounded off according to the values of the quantization matrix, which determines the tradeoff

balance between image quality and compression ratio. The matrix of quantized DCT coefficients is then encoded into a binary stream with lossless Huffman compression. An image is extracted from a jpeg file by reversing this process.

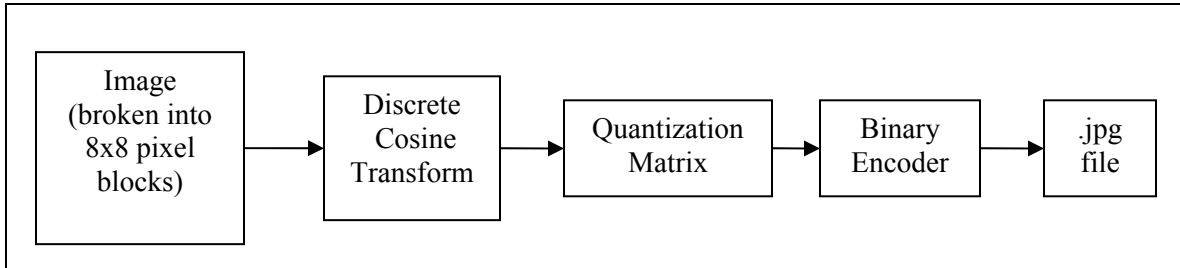


Figure 4. Jpeg Compression Process

Image information is discarded only with the rounding of DCT coefficients by the Quantization Matrix. Data embedded before this rounding (such as in pixel LSBs) may be destroyed. However, the subsequent steps in the encoding process are lossless and are therefore suitable for data hiding.

2.3 Steganography Programs

There are a number of freely-available steganography tools that embed data into lossy jpegs by modifying the rounded DCT coefficients. Depending on how it is done, DCT encoding can introduce statistical irregularities that can be used to detect its presence. As with pixel LSB encoding, minimizing the proportional changes made to the cover image DCT coefficients increases the difficulty of detecting DCT steganography. This research uses the four steganography tools summarized in Table 1. All four tools embed data into the rounded DCT coefficients of the jpeg file format, although each does so in a slightly different manner.

Table 1. Steganography Tools.

Program	Version	Author	Year
Jsteg	4	Derek Upham	1997
Outguess	0.2	Niels Provos	2001
JPHide	0.51	Allan Latham	1999
F5	1.2 beta	Andreas Westfeld	2002

Jsteg is the oldest of the four programs and was the first publicly available DCT-encoding program for jpeg images [5]. Prior to this time “It was believed that this type of steganography [DCT encoding] was impossible, or at least infeasible, since the JPEG standard uses lossy encoding to compress its data.”[6]. *Jsteg* embeds data by flipping the LSBs of the DCT coefficients in one continuous block, beginning with the first coefficient. Because *Jsteg* does not spread out the encoded bits among the coefficients, it is quite easy to detect the resulting irregularities in the coefficient first-order statistics.

Outguess [7] also flips DCT coefficient LSBs, but it spreads out these changes by selecting coefficients with a pseudo-random number generator seeded by the user-selected password [8]. *Outguess* embeds data either with or without statistical correction. In statistical correction mode, *Outguess* attempts to minimize the first-order statistical irregularities that result from modifying the DCT coefficients to make detection more difficult.

JPHide [9] modifies DCT coefficients in a pseudo-random fashion according to the ordering of a fixed table [8]. The table lists the coefficients in descending order of probability that the coefficient has a high value, the idea being that modifications to higher-valued coefficients are less detectable than changes to lower values. Even when a coefficient is next in line to be

modified, it has a certain probability of being skipped as a function of the total message size, the number of bits already encoded, and a pseudo-random number generator. This process spreads out the encoded bits in an unpredictable fashion that makes detection more difficult than with Jsteg.

The *F5* program [10] was written as a challenge to the steganalysis community. While Jsteg, Outguess, and JPHide flip LSBs of the DCT coefficients, *F5* encodes data into DCT coefficients by decrementing their absolute values. This renders *F5* undetectable by statistical analysis techniques that are effective for the LSB-flipping DCT encoders. The *F5* program randomly selects coefficients and uses matrix embedding to minimize the number of changes made to encode a message of a particular size [11].

3 Hyper-Geometric Data Classifier

The basic idea behind a geometric classifier involves representing training/testing instances as points and blind class models as simple geometric structures that can be generalized to arbitrary dimensions. The next section 1) examines previous work using a computational immune system model for blind classification, 2) shows how it can be re-interpreted to demonstrate the basic concepts of a geometric classifier, and 3) describes how it may be enhanced to form a more powerful classification paradigm.

3.1 Computational Immune System Model

Previous research created a blind steganography detection algorithm using a Computational Immune System (CIS) [12]. A CIS is a two-class classification algorithm based on a simplified model of the human Biologic Immune System (BIS) [13]. The main classification mechanism of the BIS is a set of antibodies. The proper working of a BIS calls for antibodies to detect only the presence of anomalous matter (infections, cancerous cells, etc) and trigger a defensive response. New antibodies must therefore go through a screening process, called negative selection, in which those that match against the body's own biologic uniqueness are eliminated. If such self-matching antibodies were released into circulation, they would trigger false intrusion alarms against the body's own tissues and cause the immune system to attack the body it is meant to defend. The system of antibodies is thus trained to recognize two classes: *self* and *non-self*. Instances of *non-self* trigger an immune response while instances of *self* are ignored. The anti-body creation process is blind because it trains only on instances of class *self*. The CIS uses this model to create antibodies, or classification mechanisms, that ignore clean images (steganography-free or class *self*) and trigger on anything else (stego images or *non-self*).

This CIS model can also be interpreted from a three-dimensional geometric standpoint. Each image instance in the CIS is defined by three attributes (although limited testing was done using six attributes) derived from image wavelet coefficient statistics [8]. Each instance can be represented as a point in a three-dimensional attribute space. The CIS uses stochastic search techniques to create "antibodies" in the form of 3D boxes enclosing portions of the attribute space. Any antibody that matches with *self* (i.e., a box that encloses a clean image point) is discarded. Antibodies that survive the negative selection process are retained and used to produce a new generation of antibodies. This training process is repeated until the *non-self* attribute space is well-enclosed by boxes. A test image is declared to be clean if its point in the attribute space is not enclosed by any of the antibody boxes. In summary, the CIS employs simple geometric constructs (boxes) to enclose the *non-self* space of a class (steganography-free images) in three dimensions. There are a number of ways in which a more powerful, versatile geometric classifier can be achieved.

First, creating a class model by enclosing the *self* space rather than the *non-self* space could result in a more compact class model, especially when the *non-self* space is significantly larger than the *self* space (or even infinite), and facilitate construction of several co-existing class models for a hybrid signature/anomaly based classifier rather than merely the anomaly classification of

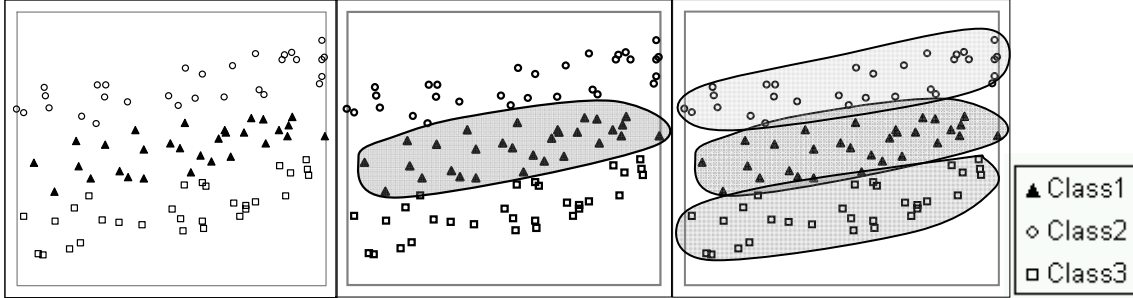


Figure 5. Hyper-Geometric Data Classifier in 2D. (a) Attribute space of a 2D 3-class problem. (b) Single class model: Class1’s training points enclosed by geometric structure, external Class2 and Class3 instances jointly classified as anomalous with respect to Class1. (c) Three class models: Attribute space of each class enclosed by a distinct geometric model for a hybrid signature/anomaly based classifier.

the CIS. Second, geometric structures more versatile than boxes can be employed to capture a wider range of complexities in a class attribute space. Figure 5 presents a 2D example of both concepts. Also, creating a geometric model deterministically rather than with a stochastic process, allows for more consistent results. Third, increasing the dimensionality of the attribute space to arbitrary dimensions allows for more powerful classification models generalizable to arbitrarily complex attribute spaces.

3.2 Hyper-Geometric Classifier Building Blocks

The instances of the training and testing sets are mapped to d -vectors representing points in d -space. Next a class model is created by geometrically enclosing the set of class training points. A test point is declared to be a member of the class iff it is enclosed by the geometric class model. A false-positive matching error occurs when a class model wrongly encloses a test instance of a different class. A false-negative error occurs when a class model fails to enclose a test instance of the same class. Decreasing the probability of one error type occurring generally increases the probability of the other type. Good classifier performance depends upon finding an appropriate balance between these two opposing error probabilities. Three different geometric enclosures (polytopes, hyper-spheres, and hyper-ellipsoids) are examined.

A d -polytope is a closed geometric construct bounded by the intersection of a finite set of hyperplanes, or halfspaces, in d dimensions. It is the generalized form of a point, line, polygon, and polyhedron in zero, one, two, and three dimensions, respectively [14], but it is also defined for arbitrarily higher dimensions (Table 2). As the number of dimensions rises, the polytope structure becomes increasingly complex and unintuitive.

Table 2. Dimensional Progression of Polytopes

Dimensions	Polytope Name	Example
0	point	•
1	line	—
2	polygon	△
3	polyhedron	⎓
4	polychoron	N/A
d	d -polytope	N/A

A polytope is convex if a line segment between any two points on its boundary lies either within the polytope or on its boundary. A convex hull of a set, S , of points in d dimensions is the smallest convex d -polytope that encloses S [15]. Each vertex of this enclosing polytope is a point in S (Figure 6a). Additionally, a d -simplex is the simplest (i.e. has the smallest number of vertices, edges, and facets) possible polytope that can exist in d dimensions. Each of the polytopes drawn in the third column of Table 2 is a *simplex*.

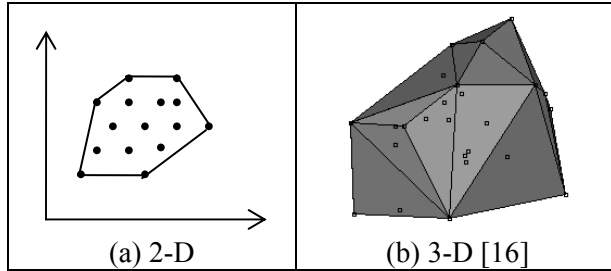


Figure 6. Convex Hulls

The second classifier makes use of the generalized circle, or hyper-sphere. A d -sphere is a hyper-sphere in d dimensions that is defined simply by a center point and a radius. This construct is significantly less complex than the convex polytope, which makes it less computationally expensive but also less flexible in the kinds of shapes that it can enclose.

The third classifier employs hyper-ellipsoids. A hyper-ellipsoid in d dimensions is represented by three parameters which define its size (s : a scalar value), location (μ : a d -vector specifying the center point), and shape (Σ : a d -by- d matrix). Any point, x , on the ellipsoid boundary satisfies Equation 1:

$$(x - \mu)^T \Sigma^{-1} (x - \mu) = s \quad (1)$$

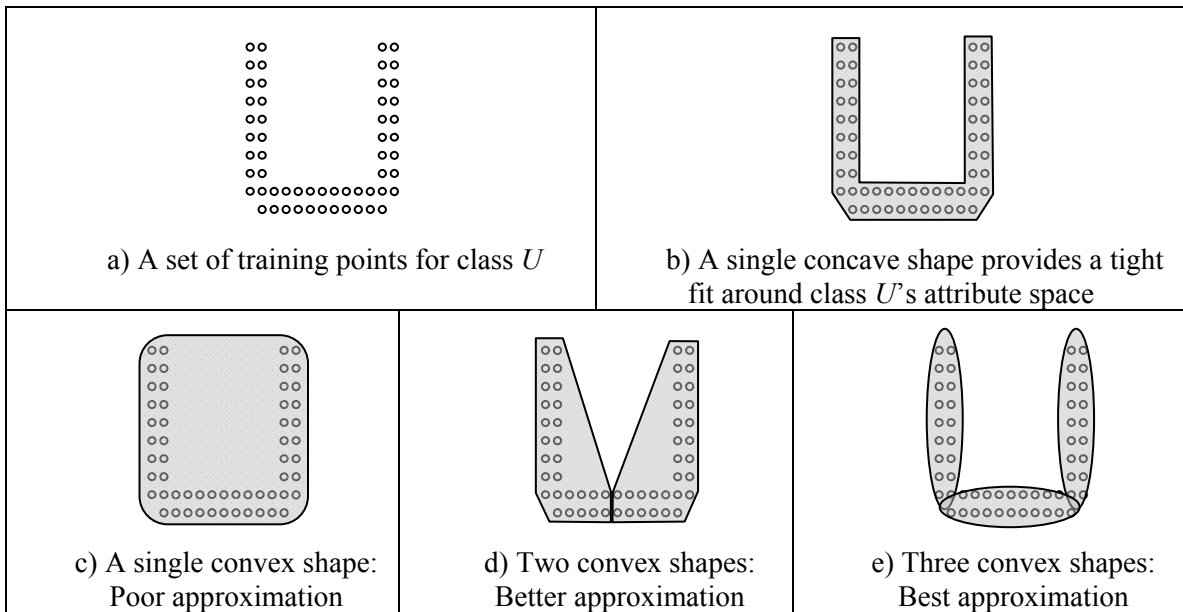


Figure 7. Concave Attribute Space Approximated By Multiple Convex Shapes

3.2.1 Concave and Disjunctive Attribute Spaces

Each of the three shapes (convex polytope, hyper-sphere, and hyper-ellipsoid) is convex. However, a single convex shape may not always provide the best model of a class attribute space. For example, Figure 7 shows several types of enclosures around the two-dimensional attribute space of class U , which has an attribute space best-fitted by a concave shape (Figure 7a-b). A single convex shape placed around such an attribute space encloses much space that does not pertain to the class and is therefore susceptible to declaring false-positive matches (Figure 7c). However, a concave shape can be approximated with multiple convex shapes (Figure 7d-e).

It is possible for a class attribute space to be disjunctive, rather than contiguous. Such an attribute space can be approximated with multiple convex shapes created around spatially disjunctive partitions of the training points. The next section explains one method for dividing a set of training points into disjunctive clusters.

3.2.2 k-means Clustering Algorithm

The k-means algorithm is used for dividing points into k disjunctive sets, or clusters. Each cluster has a *centroid*, the average of all points in the cluster. The k-means algorithm assigns points to clusters by minimizing the sum of squared within group errors, or the sum of the distance squared between each point and the centroid of its assigned cluster. The algorithm goes through iterations of re-assigning points to different clusters until it can no longer reduce the sum of squared within group errors by further shuffling the points. The time complexity of the k-means algorithm is $O(knr)$, for k clusters, n points, and r iterations [17]. While there are different implementations of the k-means algorithm, this research uses the SimpleKMeans program of the WEKA java suite [18].

3.2.3 Classifier Performance

Blind data classification is at a disadvantage compared to more traditional, non-blind methods in that when it creates a class model it cannot compare and contrast the attribute differences with other classes. For this reason, non-blind methods tend to offer better classification accuracy against known classes. However, there are a number of conditions under which a blindly-created hyper-geometric class model performs well. Specifically, the blind hyper-geometric classifier requires the following conditions to achieve good performance:

1. *Numeric Attributes*. Attribute values that are or can be mapped to real numbers.
2. *Discriminating Attributes*: Attributes that individually and/or collectively provide a good hyper-spatial separation between classes. For example, to distinguish between the two classes *man* and *woman*, the attribute *number-of-fingers* is a poor discriminator. *Number-of-X-chromosomes*, on the other hand, discriminates well. Intuitively, this requirement is also important for non-blind classification methods.
3. *Training Diversity*: A training set of instances that are well-representative of the class hyperspace (see Figure 8). Achieving this level of diversity usually requires extensive sampling of the class attribute space and a significant number of training instances. Non-blind classifiers, on the other hand, often perform well with sparser class samples because they can generalize differences between opposing class instances.

If the training set does not contain enough points on or near the boundary of the class portion of the attribute space, then the model is too small (Figure 8b). If there are not enough interior points, then the resulting model can be improperly disjunctive or even hollow. In both cases, much of the class attribute space may not be enclosed by the resulting class model, also known as overtraining on the data.

The generality of such an over-trained model can be increased through a looser-fitting geometric class model such as an oversized hyper-sphere or ellipsoid. However, applying such a poorly fitting model is a “shot in the dark.” Since the model is blindly-created (i.e.,

does not reference other classes), it is not known if the extra attribute space enclosed by the looser model properly increases the model generality (lowering the probability of false negative errors) or intrude into the class non-self space (increasing the probability of false positive errors). Due to this unpredictability, increasing the generality of a blindly-created geometric model is of dubious utility and should be considered only when diverse training data is unavailable for an application with low tolerance for false negative errors. Thus, a good, blindly-created hyper-geometric class model requires extensive knowledge of the class being modeled.

4. *Tight Fit*: An enclosing geometric structure or structures that provide a tight fit around the training points. Again, when the training diversity condition does not hold, a looser fitting model may or may not yield better results. A tightly-fitting geometric enclosure models well that portion of the attribute space that is well-represented by the training points, but is far less likely to enclose *non-self* space and is therefore generally preferable to a loose-fitting model.

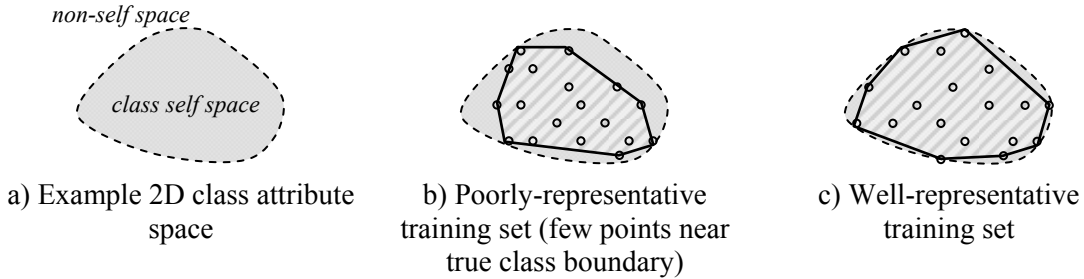


Figure 8. The Importance of a Diverse Training Set

5. *Granularity*: A method, such as the k-means algorithm, to divide the training data into as many clusters as needed to properly model non-convex and disjunctive attribute spaces.

6. *Noise Tolerance*: A way to discard statistically extreme points from the training set. If such noisy points are included, then the resulting model may enclose too much of the attribute space and be prone to declaring false-positive matches. Since different data sets have different noise levels, the filtering of extreme points is best controlled by a user-selected tolerance parameter.

3.2.4 J Score

As stated above, good performance requires attributes that individually and/or collectively discriminate well between classes. Determining collective performance requires hyper-dimensional analysis. However, individual performance can be estimated before applying the attributes to the hyper-geometric classifier. Such estimation can be beneficial in situations that call for attribute prioritization or reduction. Individual attribute performance when discriminating between classes A and B is given by computing the attribute J score. The J score is an implementation of the Fisher criterion from the Fisher Linear Discriminant [19] and is given by

$$J = \frac{(\mu_A - \mu_B)^2}{\sigma_A^2 + \sigma_B^2} \quad (2)$$

where μ_A and σ_A^2 are the mean and variance, respectively, of an attribute for all instances of class A . A high J score indicates that an attribute individually discriminates well between classes A and B . On the other hand, a low J score does not necessarily guarantee poor collective performance when combined with other attributes.

3.3 Classifier Full Implementation

For illustrative purposes, Figure Figure 9 displays a simple two-class problem. Each instance is described by two attributes, which allows the topology of the attribute space to be displayed on a two-dimensional graph. Class1 is disjunctive, as its attribute space is comprised of two well-separated clusters, in between which lies the attribute space of Class2. These two classes are used to illustrate the behavior of the different geometric classification paradigms explained in the following sections. The same basic concepts apply as the complexity of a problem increases to arbitrarily high dimensions.

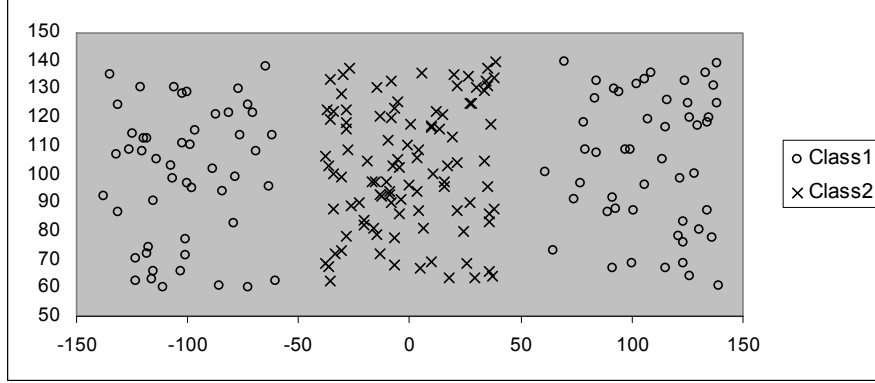


Figure 9. Two-Class Problem in Two Dimensions

3.3.1 Convex Polytope Model

Once the training instances for a particular class C have been mapped to a set T of d -vectors, the convex hull of T is computed:

$$\text{ConvexHull}(T) \equiv H \subseteq T$$

The convex hull of T is represented as the set of vertices, H , of the smallest convex d -polytope that encloses T . This model is blindly-created because it is derived solely from instances of class C . It knows nothing about the attribute values of other classes. A distinct test point p is declared to be a match (member of class C) iff it is bounded by the polytope defined by H , i.e.,

$$\begin{aligned} \text{ConvexHull}(H \cup \{p\}) &\equiv D \\ p \notin D &\Leftrightarrow \text{match}(p, C) \end{aligned}$$

If p is not bounded by the H polytope, then D represents a larger polytope of which p is a vertex. If p is bounded by the polytope, then $D \equiv H$. An alternate method is to compare the hyper-volumes of the polytopes represented by D and H . If $\text{volume}(D) > \text{volume}(H)$, then p does not match the model and is declared anomalous with respect to class C .

3.3.1.1 Convex Hull Tolerance Parameter β

To compensate for attribute space disjunctions and noisy training instances, a tolerance feature controlled by a parameter $0 \leq \beta \leq 1$ is added. It divides the training points into groups of smaller convex polytopes and provides a tighter fit around the training data, as follows:

1. Select values MIN and MAX such that $MIN < MAX$.
2. Scale each dimension of the vertices in T between MIN and MAX so that all points in T lie inside a d -hypercube with opposing extreme points at $\{MIN^d\}$ and $\{MAX^d\}$. The distance squared between these two extreme points

$$\text{dist}(\{MIN^d\}, \{MAX^d\})^2 = d(MAX - MIN)^2$$

provides the upper bound on the distance squared between any two points in T

$$\forall p, q \in T : \text{dist}(p, q)^2 \leq d(MAX - MIN)^2$$

3. Let G be an undirected, unweighted graph with a node for each vertex in T . Create an edge between each distinct pair of nodes, p and q , where

$$\text{dist}(p, q)^2 \leq \beta^2 d(MAX - MIN)^2$$

4. Partition G into unconnected sets such that in each set every node is connected by at least one edge and no edges cross between sets. Eliminate any set with fewer than $d+1$ non-coplanar points (the minimum needed to create a simplex in d -space; thus statistical outliers are discarded).

The multiple convex hulls constructed around the partitioned sets of G comprise the model of class C . Test point p is then scaled by the same factors used on T in step 2 and is declared a match iff it resides within any of the convex polytopes modeling class C . When $\beta = 1$, the algorithm behaves as the unmodified version and creates a single convex polytope around all points. As β decreases the potential number of smaller polytopes increases and their combined hyper-volume in the attribute space decreases. The decreasing hyper-volume results in a lower probability of false positives at the expense of a higher probability for false negatives, or a loss of generality. At $\beta = 0$, no convex hull models are created and all test points are rejected. Figure 10 and Figure 11 show how proportional hyper-volume and number of polytopes, respectively, for Class1 of Figure 9 are affected by β .

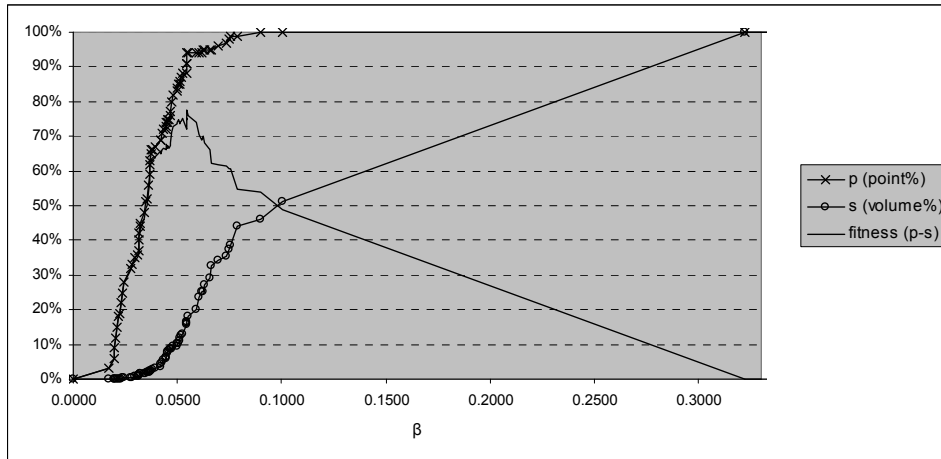


Figure 10. β Estimation: β Fitness Scores for Class1

3.3.1.2 Choosing β

The selection of an appropriate β value is critical to achieving good classification accuracy. If instances of all possible testing classes are available when creating the class model, then the β value which best fits the training data (i.e., provides an appropriate balance between false positives and false negatives) can be found through experimentation. This process, however, would not be a blindly-created model because it requires testing against opposing classes. When such domain knowledge does not exist, then an appropriate value for β must be estimated.

To provide guidance for the selection of β , another layer of abstraction is added to estimate a value for β that yields the desired classification sensitivity. First, the finite number of β values that produce unique partitionings of graph G are determined as follows:

1. Compute D , the n -by- n upper-triangular distance-squared matrix for all n instances of training class C , such that

$$\forall i, j \in C : D_{ij} = D_{ji} = \text{dist}(i, j)^2$$

- Let L be an ordered list containing the unique upper values (i.e., column index $>$ row index) of matrix D sorted in ascending order, such that

$$|L| \leq \frac{n(n-1)}{2} \text{ and } L_k < L_{k+1} \text{ for } 1 \leq k \leq |L|-1.$$

- Map each squared distance in L to a distinct β^2 value, as follows

$$\forall L_i \in L : \beta_i^2 = \frac{L_i}{d(\text{MAX} - \text{MIN})^2}.$$

- Form the ordered list $B = (0, \beta_1^2, \beta_2^2, \dots, \beta_{|L|}^2)$.

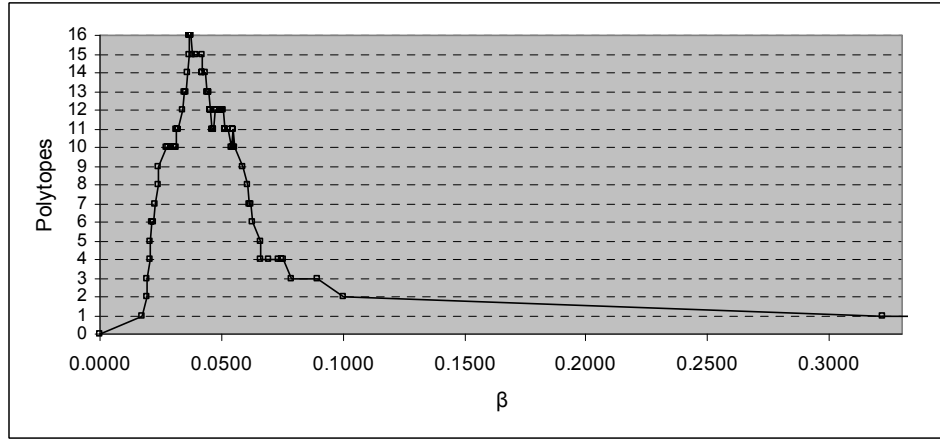


Figure 11. Number of Disjunctive Polytopes for Class 1

Note that the sorted ascending property of L also applies to B . Although there exists an uncountably-infinite number of possible β^2 values (i.e., the real numbers between 0 and 1), only those contained in list B yield distinct graphs. That is, the graph created by any β^2 value greater than B_i but less than B_{i+1} is identical to the graph created by B_i . Once all distinct β^2 values are identified, one must be selected for the β estimate by evaluating the properties of the class model derived from each.

Define $G_i = (V_i, E_i)$ as the graph created when $\beta^2 = B_i$, where V_i and E_i are the sets of vertices and edges, respectively, of G_i . Note that $E_1 \subseteq E_2 \subseteq \dots \subseteq E_{|B|}$ and $V_1 \equiv V_2 \equiv \dots \equiv V_{|B|}$. These two observations allow the set of all possible G graphs to be created incrementally by a stepwise application of the ordered values of B . The convex hulls of the unconnected partitions of each G_i are computed. Recall that partitions without enough non-coplanar points are discarded. Let $p_i = (|C| - \text{num_discarded}) / |C|$, or the percentage of points not discarded when $\beta^2 = B_i$. Further, let vol_i = the cumulative hyper-volume of the polytopes computed from G_i and let $s_i = \text{vol}_i / \text{vol}_{|B|}$, or the percentage of the total possible hyper-volume when $\beta^2 = B_i$. Once these p and s values are computed, list B is pruned by discarding all B_i where $p_i = p_{i+1}$ and $s_i = s_{i+1}$.

Both p_i and s_i provide important information about the generality of the class model produced by B_i . For example, if $p_i = 75\%$ then the resulting model fails to declare a match with 25% of the training points. If the training set is a representative sample of class C , a similar percentage of testing points from class C does not match correctly. An ideal model only discards extreme training points (statistical outliers, noise-corrupted, etc). Depending on the training data, the percentage of extreme points is usually small, so a good estimate for β favors higher values of p_i to maximize the probability of correct testing matches. On the other hand, a model produced

using noisy extreme points has a larger value for s_i and a higher probability for false positive matches. Also, dividing disjunctive data into smaller clusters reduces s_i . A good β estimate should therefore maximize p_i and minimize s_i .

Each B_i is assigned a fitness score $f(B_i) = p_i - s_i$. The highest-scoring B_i is selected and its square root becomes the blind β estimate. For the 72 distinct β values possible for Class1, Figure 10 shows the fitness scores and Figure 11 displays the total number of convex polytopes. The pivotal events from these two figures are summarized in Table 3.

Table 3. Pivotal Class 1 β Values

β	p	s	Fitness ($p-s$)	Polytopes	Comment
0	0	0	0	0	Each point is in its own group. Since all groups are smaller than $d+1=3$, all are discarded and no polytopes are created.
0.0376	0.6600	0.0254	0.6345	16	Number of polytopes peaks.
0.0547	0.9100	0.1626	0.7474	10	Global fitness score peaks. Accepting 91% of the training points produces 16% of the possible hyper-volume.
0.1004	1.0000	0.5134	0.4866	2	Model uses 100% of the training points, but divides them into two polytopes. Assuming no noise content, this model is ideal and is achieved if the no-noise assumption is expressed as $\gamma=1$. This model is illustrated in Figure 12a.
0.3222	1.0000	1.0000	0	1	All points are enclosed by a single large polytope, which nearly doubles the hyper-volume and results in every instance of Class2 being incorrectly declared a member of Class1.

An additional tolerance parameter, $0 \leq \gamma \leq 1$, constrains the β estimation process by defining a lower threshold on p for an acceptable β value. Through γ the user specifies a limit on the amount of training data that can be discarded, which may be visualized as an imaginary vertical bar on Figure 10 that occurs where $p = \gamma$. β values to the left of the bar are not considered. In practice, γ is a measure of confidence in the training data. If a user believes the training data is relatively free of noisy instances and other statistical outliers, then a higher γ value is appropriate. If the user is unsure about data noise levels, a lower γ allows the β estimator to search a wider range and possibly discard a larger percentage of noisy training data.

There are many freely available programs that compute convex hulls in high dimensions [20]. The qhull program [21] is used with this convex polytope classifier because it computes hyper-volumes and produces easily-parsed output. Qhull has a worst-case time complexity of $O(n^{\lfloor d/2 \rfloor})$ for n input points in d -space [22]. The time complexities of all other parts of creating a convex polytope model (including β estimation) are dominated by the exponential qhull complexity. Therefore, model creation and point testing have an overall time complexity of $O(n^{\lfloor d/2 \rfloor})$.

While a convex polytope has the flexibility to create tight fits around training data, its exponential-in- d time complexity limits its feasibility to classification problems containing a relatively small number of attributes. Therefore, simpler geometric constructs are evaluated as possible replacements for the convex polytope. An ideal geometric construct for a practical hyper-geometric classifier has lower computational complexity without sacrificing too much of the convex polytope flexibility to create tight fits.

3.3.2 Hyper-Sphere

A hyper-sphere may also be used to enclose the set, T , of class C training points. A single hyper-sphere is too simple to create a good fit around diverse attribute space topologies. To compensate for this lack of flexibility, multiple hyper-spheres are created. First, the k -means clustering algorithm partitions T into k different clusters (k acts as a classification tolerance parameter by controlling the partitioning of T .) Each cluster centroid produced by the k -means algorithm forms the center point, c , of a hyper-sphere. The radius, r , of each hyper-sphere is given by the distance between the centroid and the most distant cluster point. Test point p is declared a member of class C iff $dist(p,c) \leq r$ for any of the k hyper-spheres.

Testing a point for inclusion in the k hyper-spheres of S takes $O(kd)$ time. The obvious advantage the hyper-sphere model has over a convex polytope is that its time complexity is linear, not exponential, in d . Thus, a hyper-sphere may create a model with much higher dimensionality than is feasible with a convex polytope. The next hyper-geometric shape attempts to strike a balance between these two paradigms and leverage their relative strengths (i.e., the tighter fit of a convex polytope and the computational feasibility of a hyper-sphere).

3.3.3 Hyper-Ellipsoid

A hyper-ellipsoid, as observed by [23], can be used to approximate a convex polytope. Hyper-ellipsoids have been used to classify high-dimensional data in previous work. Specifically, [24] makes use of a special kind of ellipsoid, the Minimum Volume Ellipsoid (MVE), in which the size of the ellipsoid, s , is equal to the dimensionality of the space and the shape of the ellipsoid, Σ^{-1} , is a scatter matrix of points. This research differs from the MVE ellipsoid definition in that Σ^{-1} is instead an inverse covariance matrix of points and s is defined by the location properties of the points in the training set, as explained below.

Like the hyper-sphere model, the hyper-ellipsoid model divides the training set T into k clusters using the k -means algorithm. Since an ellipsoid is meant to approximate a convex polytope, all clusters containing fewer than $(d + 1)$ points are discarded. And, ellipsoids are formed around each cluster:

1. Let μ = the cluster centroid.
2. Define M as an n -by- d matrix where n is the number of points in the cluster, d is the dimensionality of the attribute space, and each row of M is a training instance.
3. Subtract μ from each row of M :

$$1 \leq i \leq n, \quad 1 \leq j \leq d : \quad Q_{ij} = M_{ij} - \mu_j$$

4. Compute the d -by- d covariance matrix of M :

$$\Sigma = \frac{Q^T Q}{n}$$

5. Compute Σ^{-1}
6. Select a value for s , as follows:

Recall that a hyper-ellipsoid is given by Equation 1. At this stage, Σ^{-1} and μ are computed but s is still an unknown quantity. The size of each cluster ellipsoid must be chosen carefully, as it affects the fit and generality of the resulting class model.

Define L as the sorted-ascending list of s values that results from computing Equation 1 with each cluster point as x , where $L_{|L|}$ defines the smallest ellipsoid size that encloses all cluster points. If the cluster contains extreme points (statistical outliers), then setting s equal to $L_{|L|}$ creates an ellipsoid that encloses too much of the attribute space and that has a high probability of declaring false-positive matches.

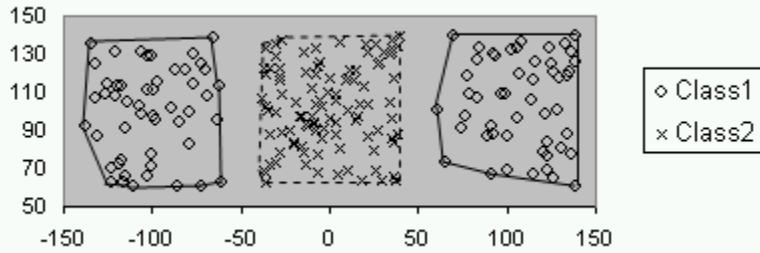
Therefore, a tolerance parameter, $0 \leq \delta \leq 1$, is applied to tweak the size of the hyper-ellipsoid. A preliminary cluster ellipsoid size is

$$s = L_{[\delta|L]}.$$

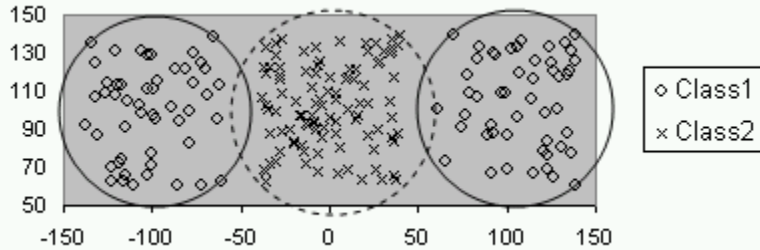
Thus, if $\delta = 0.9$ then the upper-tenth percentile of cluster points (the 10% that create the largest s values) are not enclosed by the hyper-ellipsoid, which prevents the most extreme points from affecting the size of the hyper-ellipsoid model. To purge their influence from the ellipsoid shape and location parameters as well, Σ^{-1} and μ must be recomputed for a new M containing only the bottom δ -percentile of cluster points, then L is recomputed for the new hyper-ellipsoid parameters and the remaining cluster points. Now that the effects of the discarded points are completely purged, the final cluster s value is set to $L_{|L|}$.

Once s values are selected for each cluster, test point p is declared to be a member of class C iff $(p - u)^T \Sigma^{-1} (p - u) \leq s$ for any of the k ellipsoids of C . The time complexity of multiplying together two matrices of sizes $(a \times b)(b \times c)$ is $O(abc)$ and produces a matrix of size $(a \times c)$. The ellipsoid equation multiplies three matrices of sizes $(1 \times d)(d \times d)(d \times 1)$. Thus, testing a point for inclusion in the k clusters of C takes $O(k[d^2 + d]) \approx O(kd^2)$ time, while creating the k ellipsoid models has a time complexity of $O(kn^2d^2)$.

a) Convex Polytope
 Class1: $\beta=0.1$ or $\gamma=1.0$
 Class2: $\beta=1.0$ or $\gamma=1.0$



b) Hyper-Sphere
 Class1: $k=2$
 Class2: $k=1$



c) Hyper-Ellipsoid
 Class1: $k=2, \delta=1.0$
 Class2: $k=1, \delta=1.0$

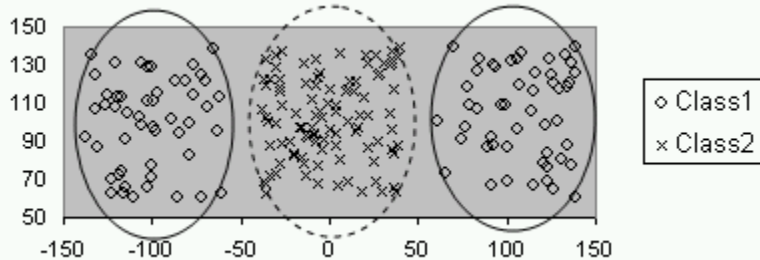


Figure 12. Three Geometric Enclosures Applied to Figure Figure 9

Table 4. Geometric Model Comparison

Shape	Tight Fit	Complexity in d	Model Creation	Point Test
Convex Polytope	Yes	Exponential	$O(n^{\lfloor d/2 \rfloor})$	$O(n^{\lfloor d/2 \rfloor})$
Hyper-Sphere	No	Linear	$O(knr)$	$O(kd)$
Hyper-Ellipsoid	Moderate	Polynomial	$O(kn^2d^2)$	$O(kd^2)$

3.3.4 Model Comparison

Figure 12 illustrates how the two classes of Figure 9 are enclosed by a) convex polytope, b) hyper-sphere, and c) hyper-ellipsoid models. The qualities of the different shapes are summarized in Table 4.

3.3.5 Hybrid Anomaly/Signature Classifier

If instances from multiple classes are available for training, then hyper-geometric models, or signatures, may be created for each class. A test instance is compared against each of the models and classified as an anomaly if no matches are found. This classifying algorithm thus becomes a hybrid signature/anomaly detector. Classification accuracy and specificity increase with the level of training data available. When the classifier contains signatures for more than one class, model overlapping is possible, as seen in Figure 12b. This overlapping is likely when poorly discriminating attributes are used or when a model has too much generality. Multiple matches may be resolved using a tie-breaking protocol.

The distance between the test point and a matching model center or nearest boundary is measured. The test point is then declared a match with whichever model best encloses the point. The complex structure of a convex hyper-polytope makes measuring the distance from the nearest boundary difficult. Instead, tie-breaking distance is computed from the mean of the polytopes vertices. For a hyper-sphere the distance between test point p and the boundary of the hyper-sphere is used ($dist(p, \mu) - r$). For a hyper-ellipsoid, a measure of the distance between p and the boundary of the hyper-ellipsoid can be derived by computing the size of an ellipsoid with p as a boundary point and then subtracting the true ellipsoid size, as follows:

$$[(p - \mu)^T \Sigma^{-1} (p - \mu)] - s$$

A purely signature-based classifier is created by also applying tie-breaking to test points that are not enclosed by any class models. The flexibility of this classification paradigm allows for uses in many possible domains. The next section describes the testing regimen used to evaluate the hyper-geometric classifier.

4 Testing Methodology

In a previous publication [25], the hyper-geometric classifier was tested against several standardized databases to validate the general classification concept and evaluate its strengths and weaknesses. The next phase of testing uses data collected from previous work on blind steganalysis.

The CIS classifier uses wavelet coefficient statistics derived from a database of 1,100 images converted to grayscale. The CIS operates on the first three of 36 coefficients extracted from each image. In addition to clean images, the database includes statistics for stego images created by Jsteg with the grayscale switch on, Jsteg color images converted to grayscale in Matlab, Outguess without statistical correction, and Outguess with statistical correction. For each of these four steganography methods, images are created using 100%, 50%, 25%, and 12.5% of the cover image's embedding capacity. Thus, the database contains 17 different classes (1 clean + 16 stego)

with 1,100 instances for each as summarized in Table 5. Each instance of the 16 stego classes represents a cover image from the clean class with steganography applied. When the i -th instance of the clean class is selected for training, the i -th instance from each of the 16 stego classes is precluded from inclusion in the testing set.

Table 5. CIS Steganography Database Classes

Class Name	Steg Tool	Statistical Correction	Embedding Capacity Used	Instances Per Class
CleanImage	-	-	0%	1100
JstegGraySwitch100	Jsteg	-	100%	
JstegGrayMatlab100	Jsteg	-		
OutGuessGrayNoStat100	Outguess	No		
OutGuessGrayStat100	Outguess	Yes		
JstegGraySwitch50	Jsteg	-	50%	
JstegGrayMatlab50	Jsteg	-		
OutGuessGrayNoStat50	Outguess	No		
OutGuessGrayStat50	Outguess	Yes		
JstegGraySwitch25	Jsteg	-	25%	
JstegGrayMatlab25	Jsteg	-		
OutGuessGrayNoStat25	Outguess	No		
OutGuessGrayStat25	Outguess	Yes		
JstegGraySwitch12	Jsteg	-	12.5%	
JstegGrayMatlab12	Jsteg	-		
OutGuessGrayNoStat12	Outguess	No		
OutGuessGrayStat12	Outguess	Yes		

4.1 Phase I- CIS Comparison

The first testing phase duplicates the CIS testing parameters. Only the first three image features are used. Ten iterations of training and testing are performed. For each iteration, 18% of the clean image class is randomly selected for training and a random 9% of each class is used for testing.

4.2 Phase II- CIS Comparison

Next, the hyper-geometric classifier is released from the CIS testing limitations to 1) train on a full 90% of the clean image instances (testing on 10% of each class) 2) use as many attributes as desired, and selecting those attributes with the highest J scores when using fewer than the full 36 attributes.

4.3 Phase III- Expanded Tests

For the next set of tests, a new database of stego images is created from 1,000 of the CIS clean images. Each is a 512-by-512 pixel, 24-bit color jpeg image. Some are taken outdoors with bright lighting while others are taken indoors with lower lighting. Some are sharp and in focus while other are fuzzy. Some contain many different colors while others are taken through night vision goggles and contain varying shades of green. The diversity of the image database should facilitate the blind creation of a clean image hyper-geometric model. However, this diversity also increases the likelihood of noisy image attributes.

Five different steganography embedding methods, Jsteg, Outguess (with and without statistical correction), JPHide, and F5 are used to embed data into the cover images. As with the CIS, for each steganography method and cover image, four stego images are created using 100%, 50%, 25%, and 12.5% of the maximum embedding capacity. Each embedded payload is

generated from a random stream of bytes to prevent tainting of classification performance being any underlying structure in the embedded messages. Features are extracted from all images to form a database of 21 classes.

The hyper-geometric classifier can incorporate attributes derived from a variety of different sources. Wavelet coefficient statistics are computed for all three color planes of each jpeg image, in the hopes that color-domain analysis will provide better detection accuracy than grayscale. Several wavelet tools are examined.

First, image features are extracted using an implementation of the Daubechies79 wavelet filter [26]. Next, features are derived using a tool written by [27]. It extracts wavelet coefficients with a user-defined wavelet filter from the Matlab wavelet toolbox and then derives error metrics from a linear predictor of coefficient magnitudes [28]. With this tool, four scales of wavelet coefficients are computed using the following Matlab wavelet toolbox filters: Biorthogonal 2.2 and 4.4 (equivalent to Daubechies79), Coiflets 1, 3, and 5, Daubechies 1, 10, and 45, Reverse Biorthogonal 4.4, and Discrete Meyer.

Once all image features are extracted, they are subjected to one-dimensional J-Score analysis to estimate discrimination effectiveness. The best scoring attributes are used by the hyper-geometric classifier. For each classifier testing iteration, 90% of the clean images are used to create the class model. The corresponding instances of the steg classes are precluded from the testing set. The remaining 10% from each class is used for testing. Ten iterations of random selection, training, and testing are performed.

The first set of tests creates a single blind clean image model for anomaly-based classification with convex polytopes, hyper-spheres, and hyper-ellipsoids. The best-performing model is retained for use in the second set of tests, in which a model for a single steg class is added to test hybrid anomaly/signature and signature-only based classification.

5 Test results

To evaluate the overall performance of each geometric model compared to the CIS, a scoring mechanism is used that places equal weight on both false positive and false negative errors, which requires averaging the mean test instance accuracy scores of the 16 stego classes. The steg average is then averaged with the clean image mean accuracy to produce the overall score.

5.1 Phase I Results- CIS Comparison

The scores of the best performing models for all geometric shapes are listed in Table 6, alongside approximate mean scores of the CIS. The convex polytope and hyper-ellipsoid models both achieve slightly better scores than the CIS, even when limited to the same testing constraints.

Some steganalysis applications may require a lower false positive rate than offered by the best-scoring hyper-ellipsoid model of Table 6. Therefore, Table 7 displays the highest scoring models constrained to have a false positive rate similar to that of the CIS. All models are similar in performance by this measure, with the convex polytope performing best and the hyper-sphere worst. These results show that the hyper-geometric classifier performs similarly to, and slightly better than, the CIS classifier when limited to the same testing constraints.

Table 6. CIS versus HGC: Best Mean Detection Rates

Class	CIS	Convex Polytope	Hyper- Sphere	Hyper- Ellipsoid
		$\gamma = 0.95$	$k = 40$	$k = 1$ $\delta = 0.85$
CleanImage	92.7%	83.0%	87.5%	71.6%
JstegGraySwitch100	65.0%	86.6%	69.8%	94.7%
JstegGrayMatlab100	65.0%	85.4%	70.7%	93.4%
OutGuessGrayNoStat100	42.0%	64.3%	47.6%	80.3%
OutGuessGrayStat100	24.0%	39.8%	29.1%	63.1%
JstegGraySwitch50	35.0%	60.0%	43.5%	76.3%
JstegGrayMatlab50	35.0%	57.7%	42.5%	77.0%
OutGuessGrayNoStat50	22.0%	37.5%	30.2%	60.5%
OutGuessGrayStat50	18.0%	26.4%	18.7%	47.8%
JstegGraySwitch25	25.0%	19.7%	14.7%	37.5%
JstegGrayMatlab25	25.0%	35.3%	23.8%	54.7%
OutGuessGrayNoStat25	15.0%	27.3%	21.7%	44.4%
OutGuessGrayStat25	13.0%	19.3%	14.4%	37.8%
JstegGraySwitch12	12.0%	18.3%	14.0%	36.4%
JstegGrayMatlab12	12.0%	18.4%	17.2%	41.8%
OutGuessGrayNoStat12	11.0%	19.3%	15.2%	36.9%
OutGuessGrayStat12	9.0%	16.7%	13.5%	35.4%
OVERALL SCORE	59.72%	61.24%	58.96%	64.49%

Table 7. CIS versus HGC: Best Constrained Mean Detection Rates

Class	CIS	Convex Polytope	Hyper- Sphere	Hyper- Ellipsoid
		$\gamma = 1$	$k = 20$	$k = 10$ $\delta = 1$
CleanImage	92.7%	94.1%	93.4%	91.4%
JstegGraySwitch100	65.0%	69.3%	48.1%	70.9%
JstegGrayMatlab100	65.0%	68.1%	52.5%	68.2%
OutGuessGrayNoStat100	42.0%	44.8%	30.9%	46.6%
OutGuessGrayStat100	24.0%	26.5%	15.4%	25.4%
JstegGraySwitch50	35.0%	38.8%	26.9%	42.4%
JstegGrayMatlab50	35.0%	40.5%	25.8%	38.5%
OutGuessGrayNoStat50	22.0%	24.6%	16.9%	24.6%
OutGuessGrayStat50	18.0%	16.8%	12.5%	15.8%
JstegGraySwitch25	25.0%	11.3%	7.9%	9.5%
JstegGrayMatlab25	25.0%	19.0%	14.9%	21.6%
OutGuessGrayNoStat25	15.0%	14.5%	11.3%	12.8%
OutGuessGrayStat25	13.0%	11.2%	7.3%	10.8%
JstegGraySwitch12	12.0%	9.1%	7.4%	8.8%
JstegGrayMatlab12	12.0%	11.9%	9.0%	13.8%
OutGuessGrayNoStat12	11.0%	8.7%	9.4%	10.8%
OutGuessGrayStat12	9.0%	9.4%	8.6%	8.7%
OVERALL SCORE	59.72%	60.33%	56.22%	59.11

5.2 Phase II Results- Expanded CIS Comparison

The J scores are computed for all 36 image attributes and each steg tool at 100% embedding. For example, Figure 13 shows the J scores for Jsteg. Each attribute’s average among all steg tools is computed. The hyper-geometric classifier operates on the attributes in descending J score order. Only the first four attributes, 14, 1, 3 and 2, appear to be significantly disturbed by the application of steganography. The remaining attributes with their low J scores do not individually discriminate well between clean and stego images. However, multi-dimensional analysis in the hyper-geometric classifier is required to assess their collective discriminatory abilities.

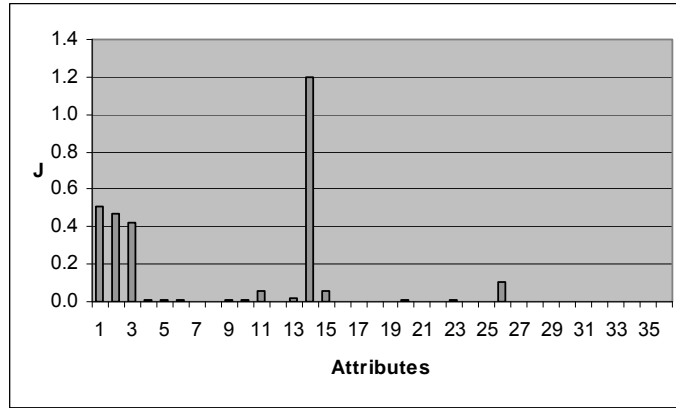


Figure 13. J Scores for Jsteg (gray switch)

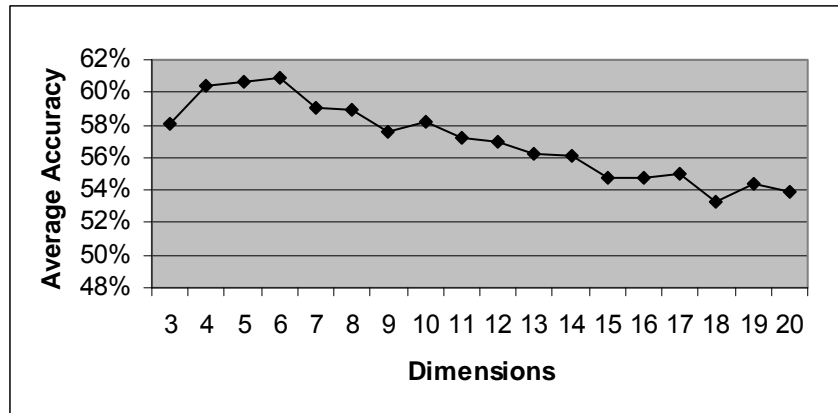


Figure 14. CIS Database- Accuracy versus Dimensionality

The hyper-ellipsoid model is tested with increasing dimensionality, as illustrated in Figure 14. After about six attributes, the added dimensions actually diminishes overall accuracy. Thus the inclusion of poorly-discriminating attributes (as determined by J score analysis) has a negative, rather than merely ambivalent, effect on performance. The two best-performing expanded models are summarized in Table 8. The first model more closely matches the false positive rate of the CIS. The second model produces the global best score.

Table 8. CIS versus expanded HGC: Best Mean Detection Rates

Class	CIS	Hyper-Ellipsoid	
		$k=1$ $\delta=0.95$ $dim=6$	$k=1$ $\delta=0.90$ $dim=5$
Clean	92.7%	90.6%	78.0%
JstegGraySwitch100	65.0%	83.4%	92.4%
JstegGrayMatlab100	65.0%	80.8%	92.6%
OutGuessGrayNoStat100	42.0%	55.5%	76.4%
OutGuessGrayStat100	24.0%	28.1%	48.3%
JstegGraySwitch50	35.0%	51.9%	73.5%
JstegGrayMatlab50	35.0%	48.1%	70.2%
OutGuessGrayNoStat50	22.0%	28.2%	51.5%
OutGuessGrayStat50	18.0%	13.9%	35.5%
JstegGraySwitch25	25.0%	14.4%	24.9%
JstegGrayMatlab25	25.0%	22.3%	42.7%
OutGuessGrayNoStat25	15.0%	16.8%	38.4%
OutGuessGrayStat25	13.0%	14.3%	26.4%
JstegGraySwitch12	12.0%	10.3%	26.7%
JstegGrayMatlab12	12.0%	16.5%	30.5%
OutGuessGrayNoStat12	11.0%	12.8%	29.3%
OutGuessGrayStat12	9.0%	10.5%	25.2%
OVERALL SCORE	59.72%	61.2%	63.5%

With an increase of only 2.1% in the false positive rate, the expanded ellipsoid model increases mean detection rates for 100% embedded Jsteg, OutGuess (no stat), and OutGuess (stat) by 17.1%(average), 13.5%, and 4.1%, respectively. These tests show that the hyper-dimensional geometric classifier outperforms the CIS when operating on the same data set. The next set of tests further examine and apply a wider range of image attributes.

5.3 Phase III Results- New Steganography Database

The highest J scores for each steg tool (at 100% embedding) and each wavelet filter are summarized in Table 9. No one filter stood out as clearly superior to the others. All wavelet statistics achieve very low J scores for stego images created by the F5 and JPHide programs, which indicates that the wavelet statistics used do not individually discriminate well between clean images and these two types of stego images. The overall ineffectiveness of the wavelet-derived features is confirmed if the multi-dimensional analysis of the hyper-geometric classifier shows that the attributes also do not discriminate well collectively.

The Daubechies79-produced attributes are selected for inclusion in the Hyper-Geometric Classifier. Figure 15 shows Daubechies79 J scores applied to Jsteg-embedded images. It shows J-scores significantly higher than those achieved by grayscale analysis. The attributes are used in descending J score order.

The best classification scores for each geometric shape are summarized in Table 10. The hyper-sphere models consistently encloses too much of the attribute space while the convex polytope overtrains on the data. As with the CIS database, hyper-ellipsoid detection is only aided to a point by increasing the number of dimensions. The best-scoring model occurs at ten dimensions, as shown in Figure 16. These ten attributes are used for the hyper-sphere and hyper-ellipsoid models, whereas the convex polytope model uses six dimensions.

Table 9. Highest J scores For Each Wavelet Filter

Wavelet Filter	Steganography Tools				
	F5	JPHide	Jsteg	OutGuess (No Stat)	OutGuess (Stat)
daub79	0.012	0.063	2.81	1.31	0.468
bior2.2	0.012	0.074	2.79	1.27	0.471
bior4.4	0.012	0.069	2.75	1.26	0.470
coif1	0.012	0.076	2.81	1.27	0.471
coif3	0.012	0.071	2.85	1.30	0.472
coif5	0.011	0.074	2.89	1.33	0.471
db1	0.013	0.087	0.88	0.78	0.470
db10	0.012	0.100	2.88	1.28	0.470
db45	0.008	0.124	0.88	0.79	0.465
Dmey	0.011	0.074	2.76	1.26	0.471
rbio4.4	0.012	0.076	2.72	1.25	0.467

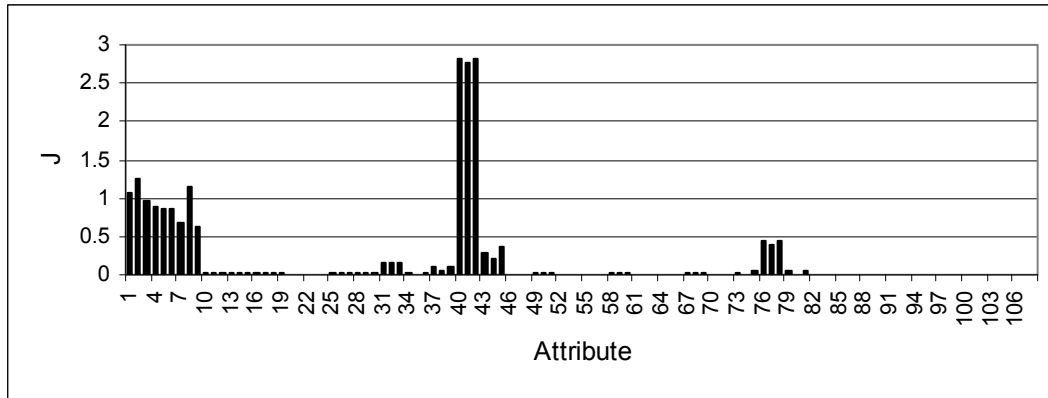


Figure 15. J Scores for Jsteg (Daubechies79 filter, color)

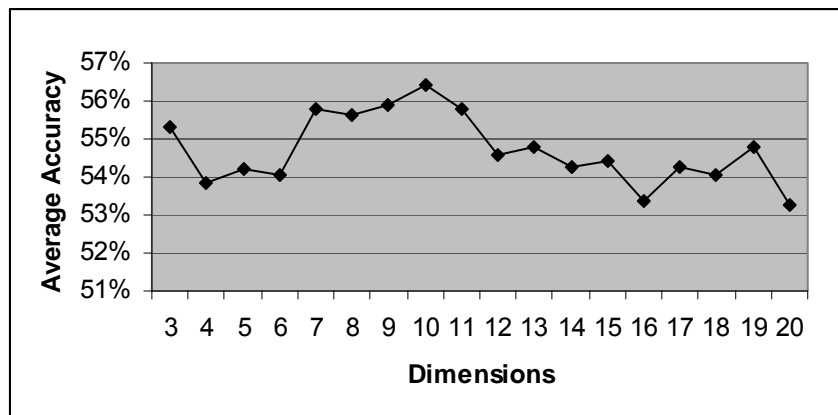


Figure 16. Effect of Dimensionality on Accuracy (Ellipsoid model)

Table 10. Best Mean Detection Rates

Class	Convex Polytope	Hyper-Sphere	Hyper-Ellipsoid
	$\beta = 0.5$	$k = 200$	$k = 1$ $\delta = 0.95$
CleanImage	47.0%	86.0%	92.9%
F5_100	55.7%	8.8%	5.1%
JPHide100	61.9%	11.5%	23.7%
Jsteg100	91.1%	71.1%	85.4%
OutGuessNoStat100	72.0%	40.2%	50.3%
OutGuessStat100	59.8%	20.9%	24.6%
F5_50	52.8%	9.2%	5.3%
JPHide50	54.8%	8.8%	9.6%
Jsteg50	68.1%	40.9%	51.7%
OutGuessNoStat50	59.2%	24.1%	24.4%
OutGuessStat50	56.4%	12.4%	13.6%
F5_25	54.8%	8.0%	4.0%
JPHide25	54.1%	8.3%	7.8%
Jsteg25	61.8%	20.2%	21.7%
OutGuessNoStat25	55.9%	14.6%	12.4%
OutGuessStat25	51.8%	11.6%	10.4%
F5_12	53.6%	8.0%	3.8%
JPHide12	51.1%	6.2%	8.5%
Jsteg12	55.8%	13.8%	11.9%
OutGuessNoStat12	52.3%	7.9%	10.8%
OutGuessStat12	57.9%	8.3%	7.8%
OVERALL SCORE	53.0%	51.9%	56.3%

Compared to the CIS, the Hyper-Geometric classifier has a slightly better false positive rate while achieving significantly better Jsteg and OutGuess detection. As expected, classification accuracy degrades as the embedded percentage shrinks. The Hyper-Sphere and Hyper Ellipsoids classifiers fail to detect F5 steganography even at maximum embedding rates. From the J score analysis, it is apparent that the attributes do not discriminate well individually between clean and F5 stego images. Testing makes clear that they also do not discriminate well collectively. Detection accuracy would certainly improve with the inclusion of image attributes that are disturbed by F5 [11], which illustrates a significant concern: Any classifier, blind or otherwise, is no better than the attributes on which it trains and tests. Novel steganography that does not perturb these attributes is not detected by an anomaly-based classifier.

The detection rate for JPHide is better. In fact, detection accuracy for 100% embedding is better than the extremely poor J scores predict, which indicates that the attributes discriminate between clean and JPHide images collectively better than they do individually. Even so, at 50% embedding the lower JPHide detection scores fall to just above the false positive rate. As with F5, detection accuracy may be improved by applying better attributes. Wavelet statistics, while useful for detecting certain types of steganography, are clearly no silver bullet. Better classification can be achieved by incorporating a wider range of image features.

Using the best-performing ellipsoid model, the next set of tests measures anomaly/signature and signature only classification. The results are summarized in Table 11. A single model of Jsteg100, being the most easily detectable steganography class, is created along with the clean image class. The wavelet attributes do not discriminate well enough between the various classes for steganography program-specific classification. Therefore, the Jsteg model represents a general

class of images containing DCT steganography. Creating a steg class model significantly improves detection rates for Jsteg and both forms of OutGuess, especially at the higher embedding rates.

Table 11. Mean Detection After Adding Jsteg Model (Ellipsoid)

Class	Anomaly	Hybrid Anomaly/Signature	Signature Only
CleanImage	92.9%	89.1%	92.5%
F5_100	5.1%	5.0%	4.9%
JPHide100	23.7%	23.6%	12.1%
Jsteg100	85.4%	94.2%	95.0%
OutGuessNoStat100	50.3%	73.7%	72.2%
OutGuessStat100	24.6%	34.9%	35.2%
F5_50	5.3%	7.3%	5.1%
JPHide50	9.6%	12.1%	8.2%
Jsteg50	51.7%	71.1%	71.0%
OutGuessNoStat50	24.4%	39.6%	35.4%
OutGuessStat50	13.6%	21.6%	16.9%
F5_25	4.0%	7.7%	4.7%
JPHide25	7.8%	10.1%	7.2%
Jsteg25	21.7%	35.2%	32.2%
OutGuessNoStat25	12.4%	18.6%	15.8%
OutGuessStat25	10.4%	14.1%	11.0%
F5_12	3.8%	7.4%	3.7%
JPHide12	8.5%	13.2%	7.0%
Jsteg12	11.9%	17.8%	17.0%
OutGuessNoStat12	10.8%	13.5%	10.9%
OutGuessStat12	7.8%	10.9%	8.8%
OVERALL SCORE	56.3%	57.8%	58.1%

6 Conclusion

This paper presents a blind hyper-dimensional geometric classification paradigm and applies it to detecting steganographic images. A class model is created without reference to opposing classes using one or more convex shapes, including convex polytopes, hyper-spheres, and hyper-ellipsoids. Creating a single class model yields an anomaly-based classifier. If more than one class is modeled, then a hybrid anomaly/signature or signature-only based classifier may be created.

The signature-only version (no anomaly detection) detects, on average, 95% of Jsteg images (at 100% embedding) with a false positive rate of 7.5%. Even the anomaly-only version detects 85.4% with 7.1% false positives. In comparison, previous work on blind steganography detection using a Computational Immune System only detects 65% with a false positive rate of 7.3%.

Acknowledgments

The work on this paper was supported by the Digital Data Embedding Technologies group of the Air Force Research Laboratory, Information Directorate. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation there on. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of Air Force Research Laboratory, or the U.S. Government.

References

- [1] Cummings, Roger. “Nothing New. An Introduction to the History of Cryptography” Storage Networking Industry Association, 2002. http://www.snia.org/apps/group_public/download.php/1627/History_of_Cryptography_Introduction.pdf
- [2] Katzenbeisser, Stefan and Fabien A. P. Petitcolas, editors. “Information Hiding Techniques for Steganography and Digital Watermarking”, Boston: Artech House, 2000.
- [3] Fridrich, Jessica, Miroslav Goljan, and Dorin Hoge. “New methodology for breaking steganographic techniques for JPEGs”, Proc. EI SPIE, Santa Clara, CA, pp. 143-155, Jan 2003.
- [4] “Jpeg Tutorial” Society for Imaging Science and Technology, Springfield, VA. <http://www.imaging.org/resources/jpegtutorial/index.cfm>. Accessed on 15 Jan 2004.
- [5] Provos, Neils and Peter Honeyman. “Hide and Seek: An Introduction to Steganography”, *IEEE Security & Privacy Magazine*, May/June 2003. <http://niels.xtdnet.nl/papers/practical.pdf>
- [6] Upham, Derek *Jsteg*. Version 4, IBM. 92k. Computer software. 1997.
- [7] Provos, Neils. *Outguess*. Version 0.2, Linux. 215k Computer software. 2001.
- [8] Provos, Neils and Peter Honeyman. “Detecting Steganographic Content on the Internet”, *CITI Technical Report 01-11*, 2001.
- [9] Latham, Allan. *JPHide*. Version 0.51, IBM, 158k. Computer software. 1999.
- [10] Westfeld, Andreas. *F5*. Version 1.2 beta, Java, 48k, Computer software, 2001.
- [11] Fridrich, Jessica, Miroslav Goljan, and Dorin Hoge. “Steganalysis of JPEG Images: Breaking the F5 Algorithm”, *5th Information Hiding Workshop*, Noordwijkerhout, The Netherlands, October 2002.
- [12] Jackson, Jacob T. “Targeting Covert Messages: A Unique Approach For Detecting Novel Steganography”, MS Thesis, Air Force Institute of Technology, Wright Patterson Air Force Base, Ohio, 2003.
- [14] Coxeter, H.S.M. *Regular Polytopes*, 3rd ed. New York: Dover, 1973.
- [13] Somayaji, Anil, Steven Hofmeyr, and Stephanie Forrest, “Principles of a Computer Immune System,” *New Security Paradigms Workshop*, pp.75-82, Langdale, UK, 23-26 Sept. 1997.
- [15] O'Rourke, J. [*Computational Geometry in C, 2nd ed.*](#) Cambridge, England: Cambridge University Press, 1998.
- [16] Lambert, Tim. “Convex Hull Algorithms applet”, UNSW School of Computer Science and Engineering, 1998. <http://www.cse.unsw.edu.au/~lambert/java/3d/hull.html>

- [17] Wong C., C. Chen, and S. Yeh. "K-Means-Based Fuzzy Classifier Design", *The Ninth IEEE International Conference on Fuzzy Systems*, vol. 1, pp. 48-52, 2000.
- [18] Witten, I.H. and E. Frank. WEKA, version 3.2.3, Java Programs for Machine Learning University of Waikato, Hamilton, New Zealand, 2002. www.cs.waikato.ac.nz
- [19] Brown, Michael. "Fisher's Linear Discriminant", web document, 1999. <http://www.cse.ucsc.edu/research/compbio/genex/genexTR2html/node12.html>. Accessed 24 Jan 2004.
- [20] Avis, D., D. Bremner, and R. Seidel. "How Good are Convex Hull Algorithms?", *ACM Symposium on Computational Geometry*, 20-28, 1997.
- [21] Barber, C.B and H.T. Huhdanpaa. *Qhull*, Version 2002.1. 283k, Computer Software, 2002. <http://www.thesa.com/software/qhull/>
- [22] Barber, C.B., D.P. Dobkin, and H.T. Huhdanpaa. "The Quickhull Algorithm For Convex Hulls", *ACM Trans. on Mathematical Software*, 22, 469-483, 1997.
- [23] Nguyen, Hoa, Melnik, Ofer, and Nissim, Kobbi. "Explaining High-Dimensional Data", unpublished undergraduate presentation. <http://dimax.rutgers.edu/~hnguyen/GOAL.ppt>. Accessed 10 Jan 2004.
- [24] Melnik, O. "Decision Region Connectivity Analysis: A method for analyzing high-dimensional classifiers", *Machine Learning*, 48:(1/2/3), 2002.
- [25] McBride, B. and Peterson G., "Blind Data Classification using Hyper-Dimensional Convex Polytopes", In *Proceedings of the 17th International FLAIRS Conference*, 2004.
- [26] Claypoole, Roger. Lift_daub79.m. Computer Software (Matlab function), Rice University, 29 Sep 1999.
- [27] Dostal, Jonathan. "jonpred.m". Computer Software (Matlab function), Air Force Institute of Technology, 2003.
- [28] Faird, Hany and Siwei Lyu. "Higher-order Wavelet Statistics and their Application to Digital Forensics", *IEEE Workshop on Statistical Analysis in Computer Vision*, Madison, Wisconsin, June 2003.