Steganalysis Using Color Wavelet Statistics and One-Class Support Vector Machines

Siwei Lyu and Hany Farid

Department of Computer Science, Dartmouth College, Hanover, NH 03755, USA

ABSTRACT

Steganographic messages can be embedded into digital images in ways that are imperceptible to the human eye. These messages, however, alter the underlying statistics of an image. We previously built statistical models using first-and higher-order wavelet statistics, and employed a non-linear support vector machines (SVM) to detect steganographic messages. In this paper we extend these results to exploit color statistics, and show how a one-class SVM greatly simplifies the training stage of the classifier.

Keywords: Steganalysis

1. INTRODUCTION

Over the past few years increasingly sophisticated techniques for information hiding (steganography) have been developing rapidly (see[1–4] for general reviews). These developments, along with high-resolution digital images as carriers, pose significant challenges to detecting the presence of hidden messages. There is, nonetheless, a growing literature on steganalysis. While many of these techniques target specific embedding programs or algorithms,[5–7] techniques for universal steganalysis have also begun to emerge.[8, 9] A review of these and other steganalysis techniques may be found in.[10]

In previous work,[8, 11, 12] we showed that a statistical model based on first-and higher-order wavelet statistics could discriminate between images with and without hidden messages. In this earlier work, we only considered grayscale images in order to simplify the computations. There is no doubt that strong statistical regularities exist between the color channels, and in this paper we extend our earlier statistical model to capture some of these regularities. In our previous work we used a (linear and non-linear) two-class support vector machine (SVM) to discriminate between the statistical features extracted from images with and without hidden messages. This classifier required training from both cover and stego images. From the point of view of universal steganalysis, this training had the drawback of requiring exposure to images from broad range of stego tools. In this paper we employ a one-class SVM that obviates the need for training from stego images, thus making the training easier, and making it more likely that our classifier will be able to contend with novel and yet to be developed stego programs.

We will first present the basic statistical model, and then describe the construction of a one-class support vector machine. We then show the effectiveness of these tools in detecting hidden messages in tens of thousands of images, and from five different stego programs. We believe that this work brings us closer to realizing a robust tool for universal steganalysis.

2. STATISTICAL MODEL

The decomposition of images using basis functions that are localized in spatial position, orientation and scale (e.g., wavelet) have proven extremely useful in image compression, image coding, noise removal and texture synthesis. One reason is that such decompositions exhibit statistical regularities that can be exploited. The image decomposition employed here is based on separable quadrature mirror filters (QMFs).[13–15] As illustrated in Figure 1, the decomposition splits the frequency space into multiple orientations and scales. For a grayscale image, the vertical, horizontal and diagonal subbands at scale i are denoted as $V_i(x, y)$, $H_i(x, y)$, and $D_i(x, y)$, respectively. For a color (RGB) image, the decomposition is applied independently to each color channel. The resulting subbands are denoted as $V_i^c(x, y)$, $H_i^c(x, y)$, and $D_i^c(x, y)$, where $c \in \{r, g, b\}$.

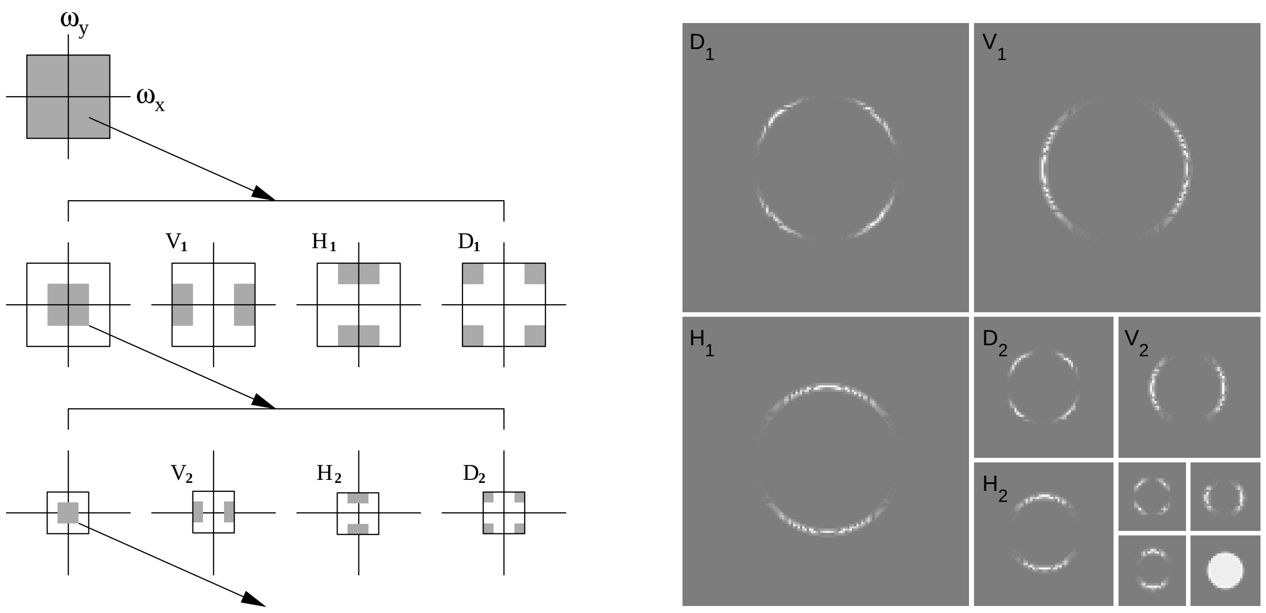

Figure 1. Shown on the left is an idealized multi-scale and orientation decomposition of frequency space. Shown, from top to bottom, are levels 0, 1, and 2, and from left to right, are the lowpass, vertical, horizontal, and diagonal subbands. Shown on the right is the magnitude of a multi-scale and orientation decomposition of a "disc" image.

Given this image decomposition, the statistical model is composed of the mean, variance, skewness and kurtosis of the subband coefficients at each orientation, scale and color channel. While these statistics characterize the basic coefficient distributions, they are unlikely to capture the strong correlations that exist across space, orientation, scale and color. For example, edges tend to extend spatially and across multiple scales. As such, if a large coefficient is found in a horizontal subband, then it is likely that its left and right spatial neighbors in the same subband will also have a large value. Similarly, a large coefficient at scale i might indicate a large value for its "parent" at scale $i+1$.

In order to capture some of these higher-order statistical correlations, we collect a second set of statistics that are based on the errors in a linear predictor of coefficient magnitude.[16] For the purpose of illustration, consider first a vertical band of the green channel at scale i, $V_i^g(x,y)$. A linear predictor for the magnitude of these coefficients in a subset* of all possible spatial, orientation, scale, and color neighbors is given by:

$$\begin{aligned} |V_i^g(x,y)| &= w_1|V_i^g(x-1,y)| + w_2|V_i^g(x+1,y)| + w_3|V_i^g(x,y-1)| + w_4|V_i^g(x,y+1)| \\ &+ w_5|V_i^g(x/2,y/2)| + w_6|D_i^g(x,y)| + w_7|D_{i+1}^g(x/2,y/2)| + w_8|V_i^r(x,y)| + w_9|V_i^b(x,y)|, \end{aligned} \tag{1}$$

where $|\cdot|$ denotes absolute value and w_k are the weights. This linear relationship can be expressed more compactly in matrix form as:

$$\vec{v} = Q\vec{w}, \tag{2}$$

where $\vec{v}$ contains the coefficient magnitudes of $V_i^g(x,y)$ strung out into a column vector (to reduce sensitivity to noise, only magnitudes greater than 1 are considered), the columns of the matrix Q contain the neighboring coefficient magnitudes as specified in Equation (1), and $\vec{w} = (w_1 \ \ldots \ w_9)^T$. The weights $\vec{w}$ are determined by minimizing the following quadratic error function:

$$E(\vec{w}) \quad = \quad [\vec{v} - Q\vec{w}]^2. \tag{3}$$

*The particular choice of neighbors was motivated by the observations of [16] and modified to include non-casual neighbors.

This error function is minimized by differentiating with respect to $\vec{w}$:

$$\frac{dE(\vec{w})}{d\vec{w}} = 2Q^T(\vec{v} - Q\vec{w}), \tag{4}$$

setting the result equal to zero, and solving for $\vec{w}$ to yield:

$$\vec{w} = (Q^TQ)^{-1}Q^T\vec{v}. \tag{5}$$

Given the large number of constraints (one per pixel) in only nine unknowns, it is generally safe to assume that the 9×9 matrix Q^TQ will be invertible.

Given the linear predictor, the log error between the actual coefficient and the predicted coefficient magnitudes is:

$$\vec{p} = \log(\vec{v}) - \log(|Q\vec{w}|), \tag{6}$$

where the $\log(\cdot)$ is computed point-wise on each vector component. It is from this error that additional statistics are collected, namely the mean, variance, skewness and kurtosis. This process is repeated for scales $i = 1, ..., n-1$, and for the subbands V_i^r and V_i^b, where the linear predictors for these subbands are of the form:

$$\begin{aligned} |V_i^r(x,y)| &= w_1|V_i^r(x-1,y)| + w_2|V_i^r(x+1,y)| + w_3|V_i^r(x,y-1)| + w_4|V_i^r(x,y+1)| \\ &+ w_5|V_i^r(x/2,y/2)| + w_6|D_i^r(x,y)| + w_7|D_{i+1}^r(x/2,y/2)| + w_8|V_i^g(x,y)| + w_9|V_i^b(x,y)|, \end{aligned} \tag{7}$$

and

$$\begin{aligned} |V_i^b(x,y)| &= w_1|V_i^b(x-1,y)| + w_2|V_i^b(x+1,y)| + w_3|V_i^b(x,y-1)| + w_4|V_i^b(x,y+1)| \\ &+ w_5|V_i^b(x/2,y/2)| + w_6|D_i^b(x,y)| + w_7|D_{i+1}^b(x/2,y/2)| + w_8|V_i^r(x,y)| + w_9|V_i^g(x,y)|. \end{aligned} \tag{8}$$

A similar process is repeated for the horizontal and diagonal subbands. As an example, the predictor for the green channel takes the form:

$$\begin{aligned} |H_i^g(x,y)| &= w_1|H_i^g(x-1,y)| + w_2|H_i^g(x+1,y)| + w_3|H_i^g(x,y-1)| + w_4|H_i^g(x,y+1)| \\ &+ w_5|H_i^g(x/2,y/2)| + w_6|D_i^g(x,y)| + w_7|D_{i+1}^g(x/2,y/2)| + w_8|H_i^r(x,y)| + w_9|H_i^b(x,y)|, \end{aligned} \tag{9}$$

$$\begin{aligned} |D_i^g(x,y)| &= w_1|D_i^g(x-1,y)| + w_2|D_i^g(x+1,y)| + w_3|D_i^g(x,y-1)| + w_4|D_i^g(x,y+1)| \\ &+ w_5|D_i^g(x/2,y/2)| + w_6|H_i^g(x,y)| + w_7|V_i^g(x,y)| + w_8|D_i^r(x,y)| + w_9|D_i^b(x,y)|. \end{aligned} \tag{10}$$

For the horizontal and diagonal subbands, the predictor for the red and blue channels are determined in a similar way as was done for the vertical subbands, Equations (7)-(8). For each oriented, scale and color subband, a similar error metric, Equation(6), and error statistics are computed.

For a multi-scale decomposition with scales $i = 1, ..., n$, the total number of basic coefficient statistics is $36(n-1)$ ($12(n-1)$ per color channel), and the total number of error statistics is also $36(n-1)$, yielding a grand total of $72(n-1)$ statistics. These statistics form the feature vector to be used to discriminate between images with and without hidden messages.

3. CLASSIFICATION

In earlier work we showed the effectiveness of using multi-class classification schemes to detect hidden messages.[8, 11, 12] Specifically, we employed both linear discrimination analysis[17] and non-linear support vector machines (SVM).[18] While these techniques afforded good classification accuracy, they required training with both cover and stego images. Since there are numerous stego programs for which these techniques might need to be trained, it would be advantageous to build a classifier from only the more easily obtained cover images. Shown in Figure 2(a) is a toy 2-D example where a non-linear SVM was trained on black dots (cover) and white

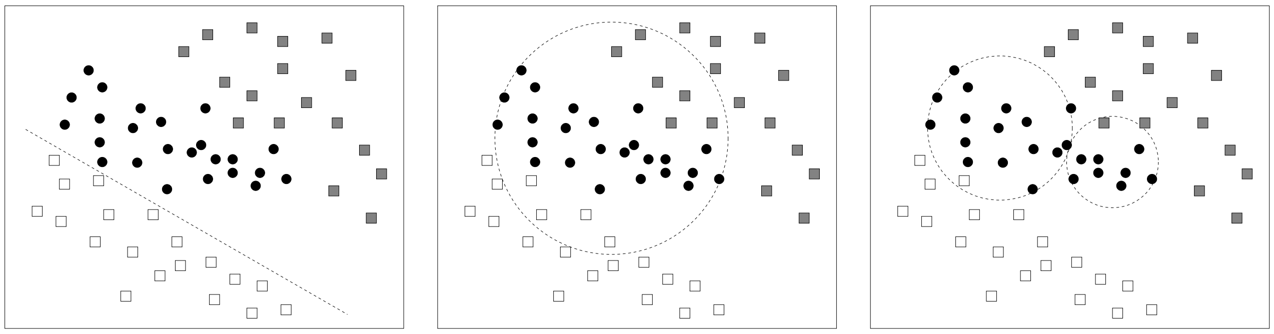
(a) (b) (c)

Figure 2. Shown are toy examples of (a) two-class SVM, (b) one-class SVM with one-hypersphere, and (c) one-class SVM with two hyperspheres. In each case, the dotted line or circle represents the classifier. The two-class SVM is trained on the black dots (cover) and white squares (stego) - notice that the gray squares (also stego) will be incorrectly classified as they were not including in the training. The one-class SVMs are trained on only the black dots - notice that in these cases the classifier is better able to generalize as both the white and gray squares generally fall outside the support of the bounding circle(s).

squares (stego), and where the dashed line corresponds to the separating surface [†]. In this same figure, the gray squares correspond to previously unseen images from a different stego program. Notice that without explicit training on the gray squares, the classifier is unable to correctly classify them. To contend with this problem, we employ one-class support vector machines (OC-SVM).[19]

An OC-SVM is trained on data from only one class by computing a bounding hypersphere (in the projected high-dimensional space) that encompasses as much of the training data as possible, while minimizing its volume. For example, shown in Figure 2(b) is an OC-SVM trained on the black dots. Note that, unlike the two-class SVM shown in panel (a), this classifier is able to classify, reasonably well, both types of stego images (white and gray squares). We describe below the details behind the construction of such OC-SVMs.

3.1. One-class Support Vector Machines

Consider n training data points in a d-dimensional space denoted as $\{\vec{x}_1, ..., \vec{x}_n\}$. An OC-SVM first projects these data into a higher, potentially infinite, dimensional space with the mapping: $\phi : \mathcal{R}^d \rightarrow \mathcal{F}$. In this space, a bounding hypersphere is computed that encompasses as much of the training data as possible, while minimizing its volume. This hypersphere is parameterized by a center, $\vec{c}$, and a radius, r. Described below is how these parameters are computed from the training data, and then how classification is performed given this bounding hypersphere.

The hypersphere center $\vec{c}$ and radius r are computed by minimizing:

$$\min_{\vec{c},r,\xi_1,...,\xi_n} r^2 + \frac{1}{n\nu} \sum_{i=1}^{n} \xi_i, \tag{11}$$

where $\nu \in (0, 1)$ is a parameterized constant that controls the fraction of training data that fall outside of the hypersphere, and ξ_is are the "slack variables" whose values indicate how far these outliers deviate from the surface of the hypersphere. This minimization is subject to:

$$\|\phi(\vec{x}_i) - \vec{c}\|^2 \leq r^2 + \xi_i, \quad \xi_i \geq 0, \quad i = 1, ..., n, \tag{12}$$

where $\| \cdot \|$ is the Euclidean norm. The objective function of Equation (11) embodies the requirement that the volume of the hypersphere is minimized, while simultaneously encompassing as much of the training data as

[†]The 2-D space shown in Figure 2 corresponds to the result of projecting the original data into a higher-dimensional space. As such, the non-linear separating surface is linear.

possible. Equation (12) forces the training data to lie within the hypersphere (the slack variables ξ_i allow the loosening of this constraint for specific data points).

To determine $\vec{c}$ and r, the quadratic programming problem of Equations (11)-(12) are transformed into their dual form[20]:

$$\min_{\alpha_1,\ldots,\alpha_n} \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j\phi(\vec{x}_i)^T\phi(\vec{x}_j) - \sum_{i=1}^{n}\alpha_i\phi(\vec{x}_i)^T\phi(\vec{x}_i), \tag{13}$$

subject to:

$$\sum_{i=1}^{n}\alpha_i = 1, \qquad 0 \leq \alpha_i \leq \frac{1}{n\nu}, \qquad i = 1,\ldots,n, \tag{14}$$

where α_i's are Lagrange multipliers. Note that in this dual formulation the constraints (Equation (14)) are now linear, and both the objective function and constraints are convex. Standard techniques from quadratic programming can be used to solve for the unknown Lagrange multipliers.[20] The center of the hypersphere, is then given by:

$$\vec{c} \;=\; \sum_{i=1}^{n}\alpha_i\phi(\vec{x}_i). \tag{15}$$

In order to computed the hypersphere's radius, r, we first use the Karush-Khun-Tucker (KKT) condition[20] to find the data points that lie exactly on the surface of the optimal hypersphere. Such points, $\vec{x}_i$, satisfy the condition $0 < \alpha_i < 1/(n\nu)$. Any such data point $\vec{y}$ that lies on the surface of the optimal hypersphere satisfies the following:

$$r^2 \;=\; \|\phi(\vec{y}) - \vec{c}\|^2. \tag{16}$$

Substituting the solution of Equation (15) into the above yields a solution for the hypersphere radius:

$$r^2 \;=\; \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j\phi(\vec{x}_i)^T\phi(\vec{x}_j) - 2\sum_{i=1}^{n}\alpha_i\phi(\vec{x}_i)^T\phi(\vec{y}) + \phi(\vec{y})^T\phi(\vec{y}). \tag{17}$$

With the hypersphere parameters, the decision function, $f(\vec{x})$, which determines whether a data point lies within the support of the hypersphere, is defined as:

$$f(\vec{x}) \;=\; r^2 - \|\phi(\vec{x}) - \vec{c}\|^2, \tag{18}$$

such that, if $f(\vec{x})$ is greater than or equal to zero, then $\phi(\vec{x})$ lies within the hypersphere, otherwise it lies outside. Note that this decision function requires an explicit evaluation of $\phi(\vec{x})$. This is computationally costly if $\phi(\cdot)$ maps the data into a very high dimensional space, and is problematic when that space is infinite dimensional. Fortunately, the evaluation of $\phi(\cdot)$ can be avoided entirely. Substituting the solution of Equation (15) into the above decision function yields:

$$f(\vec{x}) \;=\; r^2 - \left(\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j\phi(\vec{x}_i)^T\phi(\vec{x}_j) - 2\sum_{i=1}^{n}\alpha_i\phi(\vec{x}_i)^T\phi(\vec{x}) + \phi(\vec{x})^T\phi(\vec{x})\right). \tag{19}$$

The inner products between two projected data points in the above equation, in the computation of r of Equation (17), and the objective function of Equation (13) are replaced with an appropriate kernel function[21]:

$$k(\vec{x},\vec{y}) \;=\; \phi(\vec{x})^T\phi(\vec{y}), \tag{20}$$

to yield:

$$f(\vec{x}) \quad = \quad \left(-2\sum_{i=1}^{n}\alpha_i k(\vec{x}_i,\vec{y}) + k(\vec{y},\vec{y})\right) - \left(-2\sum_{i=1}^{n}\alpha_i k(\vec{x}_i,\vec{x}) + k(\vec{x},\vec{x})\right), \tag{21}$$

where the re-formulated objective function takes the form:

$$\min_{\alpha_1,\dots,\alpha_n} \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j k(\vec{x}_i,\vec{x}_j) - \sum_{i=1}^{n}\alpha_i k(\vec{x}_i,\vec{x}_i). \tag{22}$$

Commonly used kernel functions are polynomials and radial basis functions (see[21] for more examples). Note that this objective function is defined in the original d-dimensional space. In the following section we show how this basic framework can be extended to allow for coverage of the data with multiple hyperspheres.

3.2. One-class Support Vector Machines with Multiple Hyperspheres

An OC-SVM with a single hypersphere, as described in the previous section, obviates the need for training a classifier on a fully labeled training set. In our case, an OC-SVM need only be trained on cover images. In the testing stage, images that are not in the support of the bounding hypersphere are considered to contain a hidden message. One potential drawback of using only a single hypersphere is that it may not provide a particularly compact support for the training data, Figure 2(b).

To alleviate this problem, we propose to cover the training data with several hyperspheres, where each hypersphere encompasses a non-intersecting subset of the training data. Shown in Figure 2(c), for example, is the result of using two hyperspheres to cover the same data as shown in panel (b). Note that, in this case, the support is significantly more compact, thus leading to improved classification. In choosing the number of hyperspheres, however, we need to balance between the compactness of the support and the generalizability of the classifier. Specifically, if too many hyperspheres are used, then it is likely that the classifier will be tuned only to the training data, and will perform poorly when presented with novel data.

With a specified number of hyperspheres, M, the training data are first automatically segmented into M non-intersecting subsets. Specifically, a standard K-means clustering algorithm[17] is employed to cluster the original data into M groups. An OC-SVM, using a single hypersphere, is then independently trained on each of the M groups. We next compute the distance between each data point and the center of each OC-SVM's hypersphere. For each data point, the hypersphere to whose center it is closest is determined. Each data point is then re-assigned to this group, regardless of its previous assignment. And finally, a new set of M OC-SVMs are trained using the new group assignments. This process is repeated until no single data point is re-assigned. The convergence of this algorithm can be proven in a fairly straight-forward way similar to that used in proving the convergence of K-means clustering.

In the testing stage, a novel image is tested against each of the M OC-SVMs. It is classified as a cover image if it falls within the support of any OC-SVM's hypersphere, otherwise it is classified as a stego image. In the following section we show the efficacy of using an OC-SVM with multiple hyperspheres for distinguishing between cover and stego images.

4. RESULTS

Shown in Figure 3 are thirty two cover images taken from a database of $40,000$ natural images[‡]. These color images span a range of indoor and outdoor scenes, are JPEG compressed with an average quality of 90%, and typically are 600×400 pixels in size (on average, 85.7 kilobytes). To contend with slight variations in image size, only the central 256×256 region of each image was considered in the analysis. Statistics, as described in Section 2, were collected by constructing, for each color channel, a four-level, three-orientation QMF pyramid. For each image a 216-D feature vector (72 per color channel) of coefficient and error statistics was then computed. Of

[‡]All natural images were downloaded from www.freefoto.com - all images were photographed with a range of different films, cameras, and lenses, and digitally scanned.

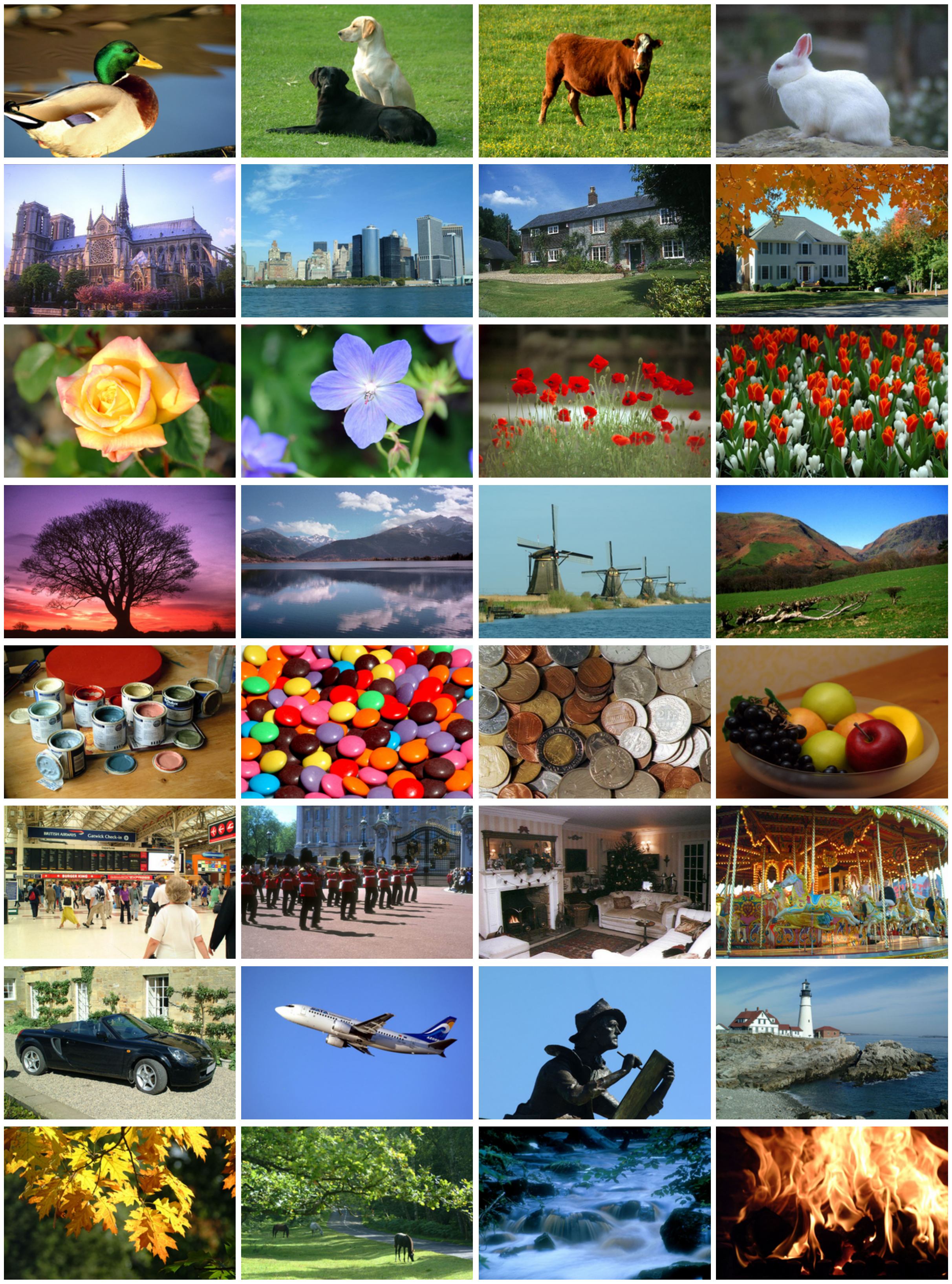

Figure 3. Examples from a database of 40,000 color images.

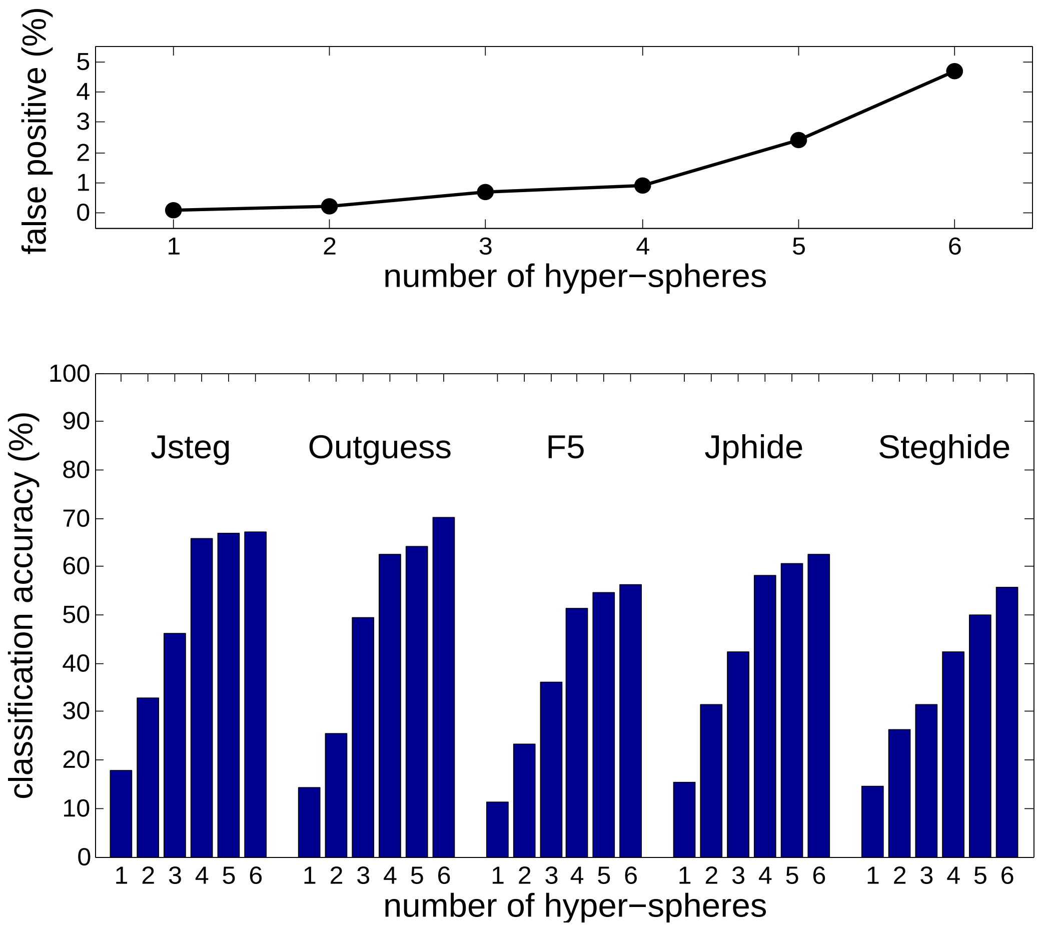

Figure 4. Shown in the bottom panel is the classification accuracy, for five different stego programs, as a function of the number of hyperspheres used in the training of the OC-SVM. Shown in the top panel is the false-positive rate (a cover image incorrectly classified as stego) as a function of the number of hyperspheres. Note that although classification accuracy improves with more hyperspheres, the false-positive rate also begins to increase.

the $40,000$ collected feature vectors, $32,000$ were used to train a one-class support vector machine (OC-SVM) [§]. For comparison sake, multiple OC-SVMs were trained with N hyperspheres, $N = 1, 2, ..., 6$. In each case, the false-positive rate was fixed at less than 1% (i.e., a cover image being incorrectly classified as a stego image).

Next, $40,000$ stego images were generated by embedding messages of various sizes into the full-resoltuion cover images. The messages were $n \times n$ central regions of randomly chosen images from the same image database, $n = 80, 40, 20$, or, 10. These messages were embedded using Jsteg,[23] Outguess,[24] F5,[25] Jphide,[26] and Steghide.[27] Each stego image was generated with the same quality factor as the original cover image so as to minimize double JPEG compression artifacts. The same statistical feature vector as described above was computed from the central 256×256 region of each stego image.

The trained OC-SVMs were then used to test the previously unseen $8,000$ cover images, and all $40,000$ stego images ($8,000$ per each of five stego embedding programs). Shown in the bottom panel of Figure 4 are the classification results of the testing stage for a message size of 80×80, and for OC-SVMs trained with a varying number of hyperspheres. Shown in the top panel of this same figure are the number of false-positives (a cover image incorrectly classified as a stego image) from the testing stage. Note that even though the overall detection

[§] *LIBSVM*,[22] with a radial basis kernel, was used as the underlying SVM algorithm.

Embedding	Message (pixels)	Message (K)	SVM^a (2 class) (gray)	SVM^b (2 class) (color)	SVM^c (2 class) (color)	OC-SVM (1 HS) (color)	OC-SVM (4 HS) (color)
No-steg	-	-	99.1	99.3	99.1	99.9	99.0
Jsteg	80×80	18.8	76.7	95.2	92.4	17.9	65.7
Jsteg	40×40	4.7	68.2	87.4	79.6	18.2	37.2
Jsteg	20×20	1.2	33.1	57.6	42.3	15.5	21.3
Jsteg	10×10	0.3	4.1	16.1	7.3	9.3	12.3
Outguess	80×80	18.8	71.5	76.3	78.6	14.3	62.5
Outguess	40×40	4.7	49.6	63.4	61.9	16.2	49.3
Outguess	20×20	1.2	20.9	29.7	34.8	12.3	26.6
Outguess	10×10	0.3	1.4	3.9	4.3	7.3	13.8
F5	80×80	18.8	5.1	48.1	69.7	11.2	51.4
F5	40×40	4.7	2.3	22.7	49.2	9.7	33.2
F5	20×20	1.2	0.0	7.1	15.8	9.1	28.3
F5	10×10	0.3	0.0	0.3	2.7	6.4	11.7
Jphide	80×80	18.8	37.8	38.9	64.4	15.3	58.3
Jphide	40×40	4.7	21.6	31.1	60.8	14.5	32.4
Jphide	20×20	1.2	13.3	16.4	25.5	19.0	29.0
Jphide	10×10	0.3	0.8	1.1	3.7	10.6	15.1
Steghide	80×80	18.8	45.4	52.6	79.6	14.6	42.4
Steghide	40×40	4.7	28.9	39.4	55.9	12.9	33.9
Steghide	20×20	1.2	11.2	18.3	33.1	14.1	31.0
Steghide	10×10	0.3	2.1	4.8	8.9	15.3	12.6

Table 1. Classification results from two-class and one-class SVMs. The two-class SVMs are trained only on Jsteg (column *b*) or on Jsteg and F5 (column *a* and *c*). The two-class SVM results shown in column *c* are based on color images, while those in column *a* are based on the same images converted to grayscale. Note that color statistics affords a considerable improvement in overall accuracy. Shown in the last two columns are the results from a one-class SVM. The OC-SVMs were trained with either one or four hyperspheres (HS). Note that the OC-SVMs are better able to generalize as compared to the two-class SVM trained only on Jsteg (column *b*). In all cases, the reported accuracy is from the classification testing stage.

improves with an increasing number of hyperspheres, the false-positive rate begins to increase considerably after four hyperspheres. The reason for this is that while multiple hyperspheres afford a more compact support in the training stage, they lead to poor generalization in the testing stage.

Shown in Table 1 are the complete grayscale and color classification results from our previous two-class SVM[8] trained only on Jsteg stego images (column SVM^b) and trained on Jsteg and F5 images (columns SVM^a and SVM^c). Also shown in this table are the results from an OC-SVM with one and four hyperspheres. In all cases, the reported accuracy is from the testing stage (i.e., images unseen during the training of the SVMs). Note first that color statistics affords a considerable improvement in overall accuracy. The two-class SVM trained on Jsteg and F5 generally performs better on novel stego programs than does the SVM trained only on Jsteg. Note also that an OC-SVM with four hyperspheres affords considerably higher classification accuracy than one hypersphere. And as expected, the OC-SVM (with four hyperspheres) is generally more consistent at generalizing to novel stego programs than the SVM trained only on Jsteg. A two-class SVM fully trained on all stego programs will almost certainly outperform an OC-SVM. The training of such an SVM, however, requires a larger training set, and may not generalize well to stego programs previously unseen by the classifier.

5. DISCUSSION

We have described a universal steganalysis algorithm that exploits the inherent statistical regularities of natural images. The statistical model consists of first- and higher-order color wavelet statistics. A one-class support vector machine (OC-SVM) was employed for detecting hidden messages in digital images. The work presented here builds on our earlier work where we used first- and higher-order grayscale wavelet statistics and a two-class support vector machine. The addition of color statistics provides a considerable improvement in overall detection accuracy. And, while a fully trained two-class SVM is likely to outperform an OC-SVM, the OC-SVM has the advantage that it is more likely to generalize to stego programs not previously seen by the classifier. In addition, the training of the OC-SVM is simplified as it only requires training on the more easily obtained cover (non-stego) images.

Techniques for universal steganalysis, such as that presented here, hold out promise that high-throughput steganography can be detected. There is no doubt, however, that small messages (relative to the cover medium) will be nearly impossible to detect. Finally, it might be possible to alter a cover image such that, after inserting a hidden message, the statistical feature vector which we collect falls within the realm of non-stego images. It is not immediately obvious (to us) how this image manipulation could be performed. We expect, however, that counter-measures will eventually be developed which can foil our detection scheme. The development of such techniques will in turn lead to better statistical models and detection schemes, and so on.

ACKNOWLEDGMENTS

This work was supported by an Alfred P. Sloan Fellowship, an NSF CAREER Award (IIS-99-83806), an NSF Infrastructure Grant (EIA-98-02068), and under Award No. 2000-DT-CX-K001 from the Office for Domestic Preparedness, U.S. Department of Homeland Security (points of view in this document are those of the authors and do not necessarily represent the official position of the U.S. Department of Homeland Security).

REFERENCES

1. D. Kahn, "The history of steganography," in *Proceedings of Information Hiding, First International Workshop*, (Cambridge, UK), 1996.
2. R. Anderson and F. Petitcolas, "On the limits of steganography," *IEEE Journal on Selected Areas in Communications* **16**(4), pp. 474–481, 1998.
3. N. Johnson and S. Jajodia, "Exploring steganography: seeing the unseen," *IEEE Computer* **31**(2), pp. 26–34, 1998.
4. E. Petitcolas, R. Anderson, and M. Kuhn, "Information hiding - a survey," *Proceedings of the IEEE* **87**(7), pp. 1062–1078, 1999.
5. N. Provos and P. Honeyman, "Detecting steganographic content on the internet," Tech. Rep. CITI 01-1a, University of Michigan, 2001.
6. J. Fridrich, M. Goljan, and D. Hogea, "Steganalysis of JPEG images: Breaking the F5 algorithm," in *5th International Workshop on Information Hiding*, (Noordwijkerhout, The Netherlands), 2002.
7. X. Wu, S. Dumitrescu, and Z. Wang, "Detection of lsb steganography via sample pair analysis," in *5th International Workshop on Information Hiding*, (Noordwijkerhout, The Netherlands), 2002.
8. S. Lyu and H. Farid, "Detecting hidden messages using higher-order statistics and support vector machines," in *5th International Workshop on Information Hiding*, (Noordwijkerhout, The Netherlands), 2002.
9. J. Fridrich, M. Goljan, and D. Hogea, "New methodology for breaking steganographic techniques for JPEGs," in *SPIE Symposium on Electronic Imaging*, (Santa Clara, CA), 2003.
10. J. Fridrich and M. Goljan, "Practical steganalysis – state of the art," in *SPIE Symposium on Electronic Imaging*, (San Jose, California), 2002.
11. H. Farid and S. Lyu, "Higher-order wavelet statistics and their application to digital forensics," in *IEEE Workshop on Statistical Analysis in Computer Vision (in conjunction with CVPR)*, (Madison, WI), 2003.
12. H. Farid, "Detecting hidden messages using higher-order statistical models," in *International Conference on Image Processing*, (Rochester, New York), 2002.

13. P. Vaidyanathan, "Quadrature mirror filter banks, M-band extensions and perfect reconstruction techniques," *IEEE ASSP Magazine* **4**(3), pp. 4–20, 1987.
14. M. Vetterli, "A theory of multirate filter banks," *IEEE Transactions on ASSP* **35**(3), pp. 356–372, 1987.
15. E. Simoncelli and E. Adelson, *Subband image coding*, ch. Subband transforms, pp. 143–192. Kluwer Academic Publishers, 1990.
16. R. Buccigrossi and E. Simoncelli, "Image compression via joint statistical characterization in the wavelet domain," *IEEE Transactions on Image Processing* **8**(12), pp. 1688–1701, 1999.
17. R. Duda and P. Hart, *Pattern Classification and Scene Analysis*, John Wiley and Sons, 1973.
18. V. Vapnik, *The nature of statistical learning theory*, Spring Verlag, 1995.
19. B. Scholkopf, J. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution.," in *Neural Computation*, pp. 1443–1471, 2001.
20. R. Fletcher, *Practical Methods of Optimization*, John Wiley and Sons, 2nd ed., 1987.
21. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery* **2**, pp. 121–167, 1998.
22. 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`.
23. D. Upham, *Jsteg*. Software available at `ftp.funet.fi`.
24. N. Provos, *Outguess*. Software available at `www.outguess.org`.
25. A. Westfeld, *F5*. Software available at `wwwwrn.inf.tu-dresden.de/westfeld/f5`.
26. A. Latham, *Jpeg Hide-and-Seek*. Software available at `linux01.gwdg.de/alatham/stego`.
27. S. Hetzl, *Steghide*. Software available at `steghide.sourceforge.net`.