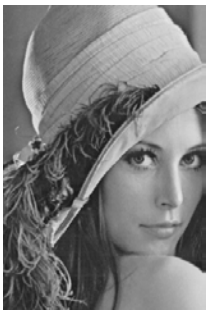
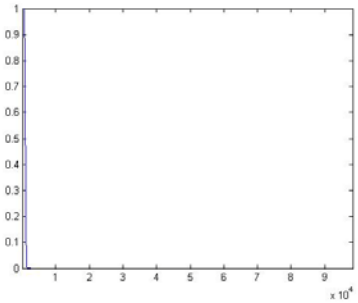
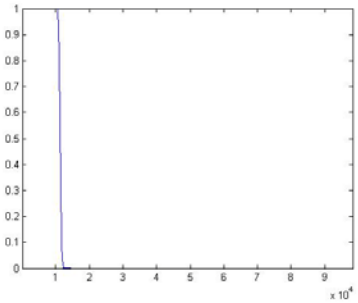
Secure Information Hiding

Portfolio

Ruoxi Li
6075338

Chi-Square Test and Steganalysis

This section uses two scenarios to evaluate the chi-square test function and discusses several factors that may affect its performance in steganalysis. In the first scenario, the function uses image “Lena” as a cover. Its thumbnail is shown in figure 1(a). The image has the pixel dimensions of 256 by 384, which provides a capacity of storing 98304 bits in total. For standard ASCII coding scheme, where each character is coded using 7 bits, the cover image is able to hold 98304 / 7 = 14043 characters; if, on the other hand, extended ASCII codes are used, the cover image will be able to store 98304 / 8 = 12288 characters.

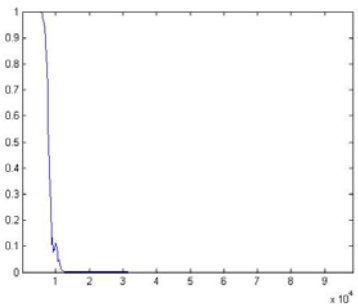
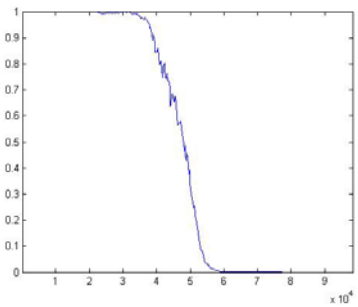




(a) (b) (c)

Figure.1 Lena

Without embedding anything into the image, almost all p-values are equal to zero, as shown in figure 1(b), which precisely reflexes the fact that the image is natural; while figure 1(c) indicates that certain amount of pixels are consecutively used to hide certain messages. According to the figure, about 10% of capacity is used for embedding; hence, the message should be 1228-1404 characters long. Once again, the figure matches the reality: there are 1200 characters embedded in the image.

Image “boat” is used in the second scenario. It has the same size as “Lena” but it behaves quite differently in chi-square test. Figure 2(b) suggests that about 5% of pixels in the image seem to be embedded with some messages, but the truth is there is nothing in there. In figure 2(c), p-values start to drop when more than a third of the pixels are taken into account for analysis, which correctly reveals that around 30% of the pixels are used for embedding. It is noticeable that p-values in figure 2(c) drop gradually while in figure 1(c), they dive to bottom right after the turning point.

(a) (b) (c)

Figure.2 Boat

The two figures above show that the function “chi2p” works properly but it should not always be trusted for steganalysis. Certain types of images, such as the “boat”, may affect the chi-square test, because the fundamental principle of the test is figuring out how likely the observed data is under the null hypothesis (H_0), which in this case, is “for a random message steganogram, a 50-50 even colour values and odd colour values distribution is expected”. Natural images could look like anything and the image used in figure 2(a) happens to look like a steganogram, which causes the function to produce a false positive.

The implementation of “chi2p” also introduces interferences to steganalysis. In order to detect the steganogram, pixels in an image must be processed in the exact same order in both function “chi2p” and the embedding function “p_embed”. For example, the latter function converts pixels into a 1-demensinal array before embedding using the following code in Matlab:

```
imgOneDm = img(:)';
```

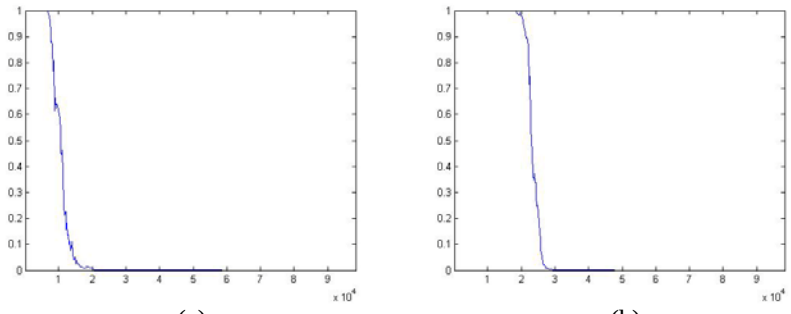
This means that if function “chi2p” needs to convert a 2D pixel matrix into a 1D array, it has to code like this:

```
X = X(:)';
```

The following codes also convert a 2D pixel matrix into 1D but the function will not work properly:

```
X = X';
X = X(:);
```

Perhaps the easiest way to mess up chi-square test is to randomly embed the secret message across the image. The test may still be able to detect the steganogram but it will fail to estimate the length of the message. Figure 3(a) shows how the chart will look like in such situation; it is compared with figure 3(b), where the secret message is embedded consecutively. The contrast will be significantly sharper if the image is large enough.

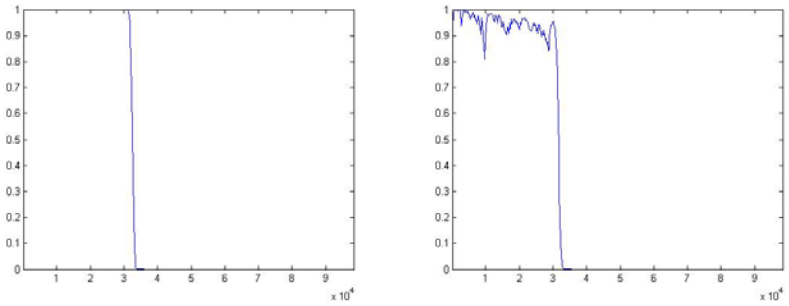


(a) (b)

Figure.3 Embedding Method: Random and Consecutive

Further tests show that another factor that affects chi-square test is the degree of freedom, but the impact is relatively minimal. Figure 4(a) is produced using a fixed degree of freedom v = 127 and 4(b) is plotted using dynamic degrees of freedom.

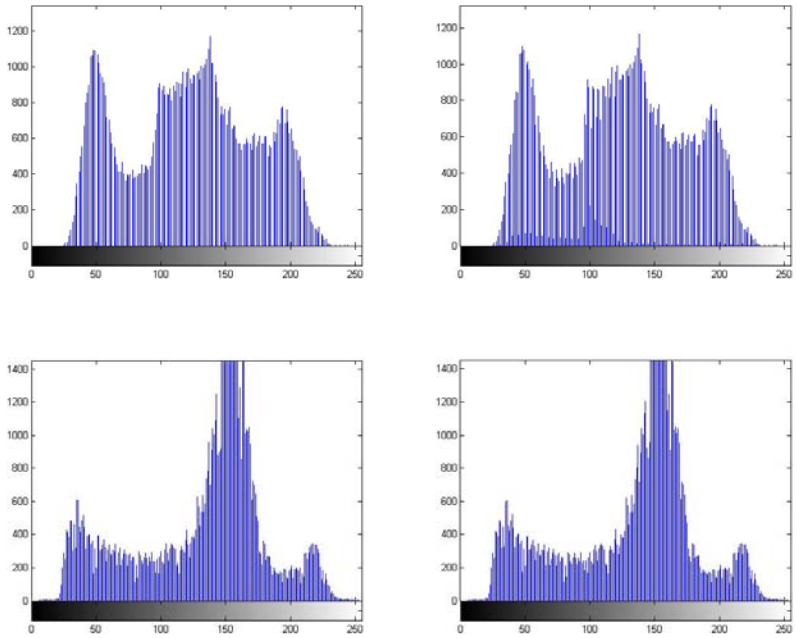
They both correctly indicate the length of the message; the only difference is that the latter chart looks ragged.



(a) (b)

Figure.4 Degree of Freedom: Fixed and Variable

Figure 5 shows the histograms plotted from two sets of images, those ones in group (a) are natural and the ones in group (b) are their stego images. Histograms of stego images are expected to be more ragged and every other bar in the histogram is expected to stick out. These characters do appear in the histograms of stego images and it is easier see them if there is something to compare with, but it is fairly hard to tell whether an image is a stego image by observing just one histogram.

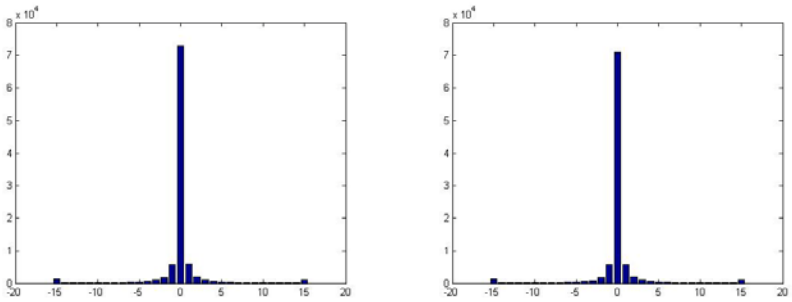


(a) (b)

Figure.5 Histograms: Innocent and Guilty

JPEG Calibration and Steganalysis

JPEG compression offers a fairly large space for embedding secret messages; steganography algorithms such as JSteg and OutGuess are well-known for exploiting redundant data in DCT domain. Similar to chi-square test, which reveals the existence of secret messages in spacial domain, calibration is used in transform domain for steganalysis. This section shows how steganograms can be identified by comparing histograms of the original image and the calibrated image.

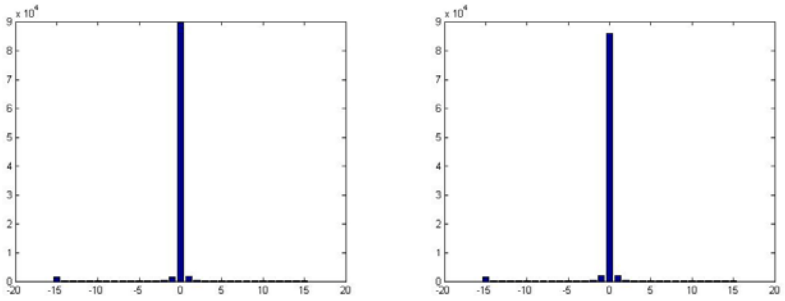


(a) (b)

Figure.6 JPEG AC Histograms for Original and Calibrated Image-a

Figure 6(a) and figure 6(b) are plotted from the original "image-a.jpg" and its calibrated version using function "bar(*X*, *h*)", where *X* is defined as *X* = [-15:1:15] and *h* is a vector constructed by function *h* = hist(*Y(:)*, *X*), where *Y* is the coefficients matrix in a JPEG image.

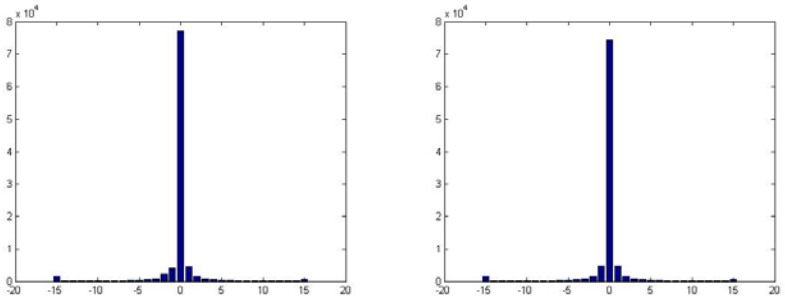
These graphs show that the original image and its calibrated image have very similar statistics, which indicates that image-a is unlikely to be a steganogram. Another observation is that the image might be highly compressed because the bar that represents 0 is significantly higher than others.



(a) (b)

Figure.7 Histograms for Image-b and its Calibrated Image

The similar statistics appear again in histograms for image-b and its calibrated version, hence, it is safe to assume that the image is innocent. The histograms do have some minor differences, for example, it is noticeable that the calibrated image has fewer 0 coefficients. It could be the result of calibration; because the image is incredibly dark, removed coefficients may mostly be equal to zero. A lack of bright pixels also explains the excessive appearance of zero coefficients over all.



(a) (b)

Figure.8 Histograms for Image-c and Its Calibrated Image

At a glance, the histograms still look similar but compare with figure 8(b), figure 8(a) does not seem to have a convincing symmetric structure. The presence of the difference between the original image and its calibrated image indicates image-c may be a steganogram.

Image-c could be produced by JSteg, because its histogram shows asymmetry, while stego images produced by improved algorithms such as F3, F4 are able to maintain the symmetric structure in the histogram.

Pairs Analysis

Pairs Analysis was original designed for analyzing palette images (i.e. GIF); the algorithm is used here for analyzing grayscale images. In this section, function "pairs_estimate" and its auxiliary functions are used to estimate the length of the secret messages in some sample images.

The table below shows experiments results of using Pairs Analysis to detect the length of secret message in stego-images produced by "p_embed", a simple steganography algorithm that hide messages in the LSB of an image. Grayscale BMP images "Lena" and "boat" are used here again as a cover, but their size is reduced to 128 by 128 pixels; the embedding algorithm uses 8 bits to represent characters in the secret message.

% of Capacity Used	0%	30%, Consecutive Embedding	30%, Random Embedding	70%, Consecutive Embedding	70%, Random Embedding	100%, Consecutive Embedding	100%, Random Embedding
Detected Length, Lena	q = 0.0101 L = 2.02%	q = 0.0380 L = 7.60%	q = 0.0651 L = 13.0%	q = 0.1366 L = 27.3%	q = 0.2507 L = 50.1%	q = 0.2080 L = 41.6%	q = 0.3703 L = 74.0%
Detected Length, Boat	q = -0.3429 L = N/A	q = -0.1152 L = N/A	q = 0.2460 L = 49.2%	q = -0.0163 L = N/A	q = 0.3226 L = 64.5%	q = 0.0000 L = 0.00%	q = 0.5000 L = 100%

**q: The smaller root of the quadratic equation: $4D(0.5)q^2 - 4D(0.5)q + D(q) = 0$*

***L: The approximation to the length of the message, L = 2q* 100*

Figure.9 Pairs Analysis, Experiment 1

Estimations made by Pairs Analysis are accurate in some cases but they do not precisely reflex the reality overall. Somehow it works better on image "boat". Using more test images may reveal the true capability of Pairs Analysis, after all it shows great accuracy in the paper <Quantitative steganalysis of digital images: estimating the secret message length> by J. Fridrich et al. The reason that it does not perform so well here could be:

Firstly, Pairs Analysis was originally designed for detecting stego-images generated by EzStego, which is a steganography algorithm that embeds secret message in GIF images; the concepts of colour cuts and colour cuts for "shifted" colour pairs in Pairs Analysis specifically target the bit flipping mechanism in EzStego. As "p_embed" works completely differently, Pairs Analysis may fail to identify some typical stego-images.

Secondly, GIF images rely on a palette to display colours rather than using the actual colour values, hence, a more sophisticated algorithm is needed for manipulating GIF images in order to preserve the fidelity. The dissimilarity between GIF and BMP could result in a different behaviour of Pairs Analysis.

The second reason is unlikely to cause any problem in experiment 1, though, because the sample images are greyscale, the pixel matrices of a BMP image and a GIF image are identical.

The following table shows how Pairs Analysis reacts to stego-images produced by JSteg. Greyscale images “Lena” and “boat” are converted to JPEG images for this experiment.

% of Capacity Used	0%	30%	70%	100%
Detected Length, Lena	q = 0.0275 L = 5.50%	q = 0.0194 L = 3.88%	q = 0.0081 L = 1.62%	q = -0.0354 L = N/A
Detected Length, boat	q = 0.1392 L = 27.8%	q = 0.0084 L = 1.68%	q = 0.5000-0.0909i L = 100%	q = 0.5000-0.4263i L = 100%

**q: The smaller root of the quadratic equation: $4D(0.5)q^2 - 4D(0.5)q + D(q) = 0$*

***L: The approximation to the length of the message, L = 2q* 100*

Figure.10 Pairs Analysis, Experiment 2

Pairs Analysis seems to be insensitive to jpeg stego images. However, this is expected because it is supposed to work in spacial domain. Embedding messages in DCT domain cause different kind of distortion in spacial domain, which is completely unlike what Pairs Analysis expects. In addition, image format handling may introduce more interference. For example, JSteg needs the coefficients matrix from the JPEG structure, which is constructed by function “jpeg_read” from JPEG Toolbox; while Pairs Analysis has to read a JPEG image using Matlab function “imread”.

Generally speaking, Pairs Analysis does a lousy job for steganalysis, because it is specifically designed for EzStego. Since EzStego is no longer downloadable from its official website, it will be difficult to find out how powerful Pairs Analysis really is.

Pairs Analysis also suffers from performance issue. One of its auxiliary functions, “rpp5”, runs way too slow due to a large amount of calculation. Matlab functions “tic” and “toc” precisely report how much time is spent on running a chunk of code. For processing an image that has 128 * 128 pixels, rpp5 requires 2 seconds; for a 256 by 384 image, it takes up to 270 seconds, which is equivalent of 4 and half minutes. A look-up table may be a solution here, but it would not be very practical.

Grammar-Based Steganography

Grammar-based steganography is a very interesting technique for information hiding; it converts bits into segments of human language and put all pieces together to form innocent-looking sentences. Compare with other algorithm, grammar-based steganography is sort of unique, because it does not necessarily require a cover-text.

The idea of such algorithm is straightforward. For example, a simple version of this algorithm could use nouns to represent 1 and verbs to represent 0; a more complex version would include more grammar rules and a large vocabulary is also essential, it needs to avoid the output-text repeats itself over and over again.

A grammar-based steganography system could be more successful if its designer is careful enough with the form of the output text. A simple system shouldn't attempt to generate news articles as output texts because their readers tend to be more critical. Poetry seems to be the suitable format since poems are usually grammatically flexible. Sophisticated systems may try to make the output-text looks like children's dairy; similar to the previous example, the purpose of doing so is to persuade the readers to tolerant more mistakes in the text.

Choosing the right words and putting them into a sentence is not hard for human but it is difficult for computers. The programme that is able to generate reasonable sentences will need a linguistic database. Complex grammar and a large vocabulary is helpful for encoding the secret message but not necessarily helpful for decoding it. The most challenging implementation issue is to generate a paragraph of text with a clear subject; although the algorithm could employ context-free grammar, the stego-text still needs some kind of structure.

Perhaps the system will be more practical and efficient on personal computers, since computers could have enough processing power to generate cover-texts as academic papers or personal emails. Another potential application of grammar-based steganography is hiding secret messages using SMS on mobile phones, but it may be unrealistic for the following two reasons. First of all, the maximum length of a text message is 160 characters, it is too small in many cases; long messages can be split and re-joint but the complexity of the steganography system and the cost to users will rise. Secondly, mobile phones have very limited processing power, they cannot handle heavyweight programmes.

Steganalysis for grammar-based steganography using mathematical and statistical methods will be very hard because the cover-text can perfectly match statistics models. In this circumstance, steganalysis will have to rely on human but consequently it will suffer from objectiveness problems.

It is possible to adopt this technique and use it on images, where the goal of algorithm is to mimic abstract expressionism in art. The figure on the left is called *No.5 1948*, painted by Jackson Pollock, an American painter known for his contributions to the abstract

expressionist movement. There is some kind of structures in the image but it looks pretty random over all. The steganography system could define certain groups of colours, stripes or blocks as 1 or 0, and construct an image based on the secret message. Similar to grammar-based steganography in text, the system does not require a cover-image in advance to embed the secret message; in addition, the image could be statistically perfect and therefore defeat most of steganalysis algorithms. It will be hard for humans to analysis and identify stego-image as well, because unlike texts, an image does not have to make any sense.

Conclusion

The first three sections in this portfolio evaluate three steganalysis techniques. Their abilities of detecting steganograms reveal weaknesses in many steganography systems and provide valuable thoughts for designing new steganography algorithms.

An in-depth understanding of statistics would greatly help comprehending these steganalysis techniques since they can all be viewed as statistic models at some level. However, heavily relying on statistics to detect steganogram has consequences. First of all, statistical models can be fooled; Outguess 0.2 is a classic example of using "statistics-aware" embedding to mimic innocent statistical structure; secondly, statistics gives people probabilities and possibilities, it hardly gives a very firm answer. But even if it does, how can we be sure that we are not dealing with an exception? In other words, how can we eliminate natural images that happen to look like steganogram?

There does not seem to be an answer for this question and that leads to the third problem of statistics: it shows confidence but not evidence. For example, assume 99% of the population in X country is terrorists, but it does not mean that anyone from X can be accused of being terrorist without hard evidences; the same rule applies to steganalysis, assume steganalysis finds two prisoners are exchanging suspicious pictures, the warden cannot punish them unless he finds the content of the secret message that is hidden in the picture. Unfortunately, most steganalysis model can only detect the existence of the secret message, maybe the length of the message as well, but not the content (steganalysis does not care about the content).

Another observation of steganalysis is that it is insensitive to short messages. It seems that the less secret message uses the capacity of the cover, the harder it will be detected, just like the old saying: the best place to hide a tree is the forest. From the steganography point of view, it means that the size of the cover should be significantly larger than the length of the secret message, in order to achieve higher security; on the other hand, such strategy increases redundant data and difficulties of sending them.

In reality, steganography algorithms do not always have to deal with steganalysis models, they make more effort to deal with human. For most of the time, stego-objects will be safe as long as they look innocent. This is especially true in grammar-based steganography because there is no effective approach to identify its stego-text automatically. Grammar-based steganography is also an exception of the "message-should-be-much-shorter-than-the-cover" law, because it does not require a cover-text, the length of the stego-text is determined by the secret message.

Compare with steganography techniques that use the combination of texts and images, those ones that manipulate text directly seem to be more practical (e.g. grammar-based steganography). Perhaps the only widely-used steganography technique is using abbreviations and acronyms to hide information in text messages on mobile phones. Symbols such as "ctn" or "d8" do not look like words but they actually mean "can't talk now" and "date". It is reported that kids are especially enthusiastic to use those lingos to avoid parents or teachers reading their messages. Some programmes, such as

“TransL8it” and “Acronyms Teen Chat Decoder”, are made in the hope of revealing the real meaning of the strange-looking symbols. But their accuracy is doubtful, because making up new symbols is extremely easy and there’s no standard way of using them.