Priming for Steganography: Undermining Steganalysis through the Introduction of Superfluous Variation

Rod Hilton, Derek Kern
Department of Computer Science
University of Colorado at Denver
{rod.hilton, derek.kern}@ucdenver.edu

May 13, 2013

Abstract

This paper examines the effects upon steganalysis classifier performance of introducing superfluous variation into digital cover images . By using fixed steganography and steganalysis tools (Steghide 0.5.1 for former and Stegdetect 0.6 for latter), a number of different image filters are individually applied to cover images prior to the introduction of steganographic content; the effect of these filters are then measured in terms of the classifier performance (AUC) of steganalysis software Stegdetect.

1 Introduction

Steganography is the art of hiding information. A steganographer succeeds if the presence of a secret message is never discovered. In the past, this amounted to hiding text under wax, microdots, and invisible ink. Today, it generally means embedding data into digital media, like images and audio/video files. Steganalysis is the art of uncovering steganography. A steganalyst succeeds if the presence of a secret message is discovered, even if the content of the message is never known. In the past, it was nothing more than detective work. Today, it is a computational and statistical science in which digital media are searched for anomalies that might be the result of an embedded message.

In this paper, the only digital media to be considered are digital images (in JPEG format). Let *cover image* refer to an image that contains no message embedded through some steganographic method. Conversely, Let *stego image* refer to an image that contains a hidden message embedded through some steganographic method.

Variation in cover images is well recognized as a factor in steganalysis [11, 6, 9]. Variation in digital images can arise from a number of different sources. In digital photography, for example, varying lighting conditions, temperatures, and humidities can affect the final image in little and, sometimes, imperceptible ways. Digital images created by scanning

analog negatives are notorious for being noisy and often held up as paradigmatic cases of noisy digital data [11]. Let these sources of variation be known as *non-deliberate* or *natural* sources of image variation.

Of course, given the limitations of human vision, most of these variations are of little importance. Often, they increase the space needed to store images with little other effect. However, in the ongoing battle between steganographers and steganalyzers, image variation could be a key participant. Given that it is already a variable that must be accounted for in most steganalytic research, it is reasonable to suspect that its impact is more than negligible.

In past studies, natural variation has been treated almost exclusively as variable to be accounted for. Yet, this singular treatment appears shortsighted, especially given the ability of modern Internet users to quickly and easily modify image properties through the use of filters, like on the social media website Instagram™[3]. This means that users themselves can introduce interesting image variations without being experts, unlike in the past. The effects of these variations upon steganalysis is not well studied. Let sources of image variation like filters be known as *deliberate* sources of image variation. If these types of deliberate image variations can be shown to impact steganalysis, then they must be accounted for in future steganalytic research. After all, given the popularity of image filtering, the mere presence of an image that has been filtered is not sufficient to declare that image as a stego image.

In this paper, the deliberate introduction of superfluous variations will be examined. Let a *superfluous variation* be any modification of an image that leaves its subject visually intact; typically, these types of variations are introduced for aesthetic reasons; Instagram filters are good examples of these types of variations.[1] Specifically, in this experiment, the effects of different image filters upon the ability to steganalyze putative stego images will be studied. The claim under study is the claim that the deliberate introduction of superfluous variations, through filters, will negatively affect steganalysis.

This research is valuable for a number of reasons. From the steganographic perspective, it offers a possible avenue to improving existing steganographic techniques. If superfluous variations can be shown to negatively affect steganalysis, then, perhaps, many steganographic methods should begin by priming cover images in order to be better secure and hide steganographic content. From the steganalysis perspective, its value is twofold. First, introduction of superfluous variation could enable the construction of many and more varied image databases with which to test steganalyzers; these databases would, with the addition of pre-filtered images, in turn be more representative of the types of images that Internet users are sharing and viewing everyday. Second, by precisely controlling the variations that are introduced, the types of variations that most affect steganalysis might be better understood.

2 Background and Previous Work

Most of the research into steganalytic and steganographic techniques incorporate a number of images from a variety of sources into the experiment design [9, 10, 11, 12, 8]. The images

[1]The adjective *superfluous* is used because these types of variations are usually introduced for exclusively aesthetic purposes.

for these experiments often come from publicly available image databases, like ImageNet [7]. There exist many publicly available image databases that can be used to evaluate steganographic and steganalytic techniques. These databases usually contain images taken from a wide variety of sources (camera models, negative scanners, etc) with a wide variety of subjects and lighting situations. Consequently, much of the research in this area [9, 10, 11, 12, 8] is confined natural variation only.

The research in [11] uses a number of different image sources to evaluate a new classifier called 'WAM'. In its first experiment, a single source, a Olympus C7652 camera, is used the classifier. In the next experiment, digital images are used from 20 different digital cameras with a wide variety of subjects and settings. In the final experiment, digital images created by scanning analog negatives are used to evaluate WAM. The upshot of this is that nowhere in [11] is the impact of deliberate image variations upon WAM explored. This is indicative of most steganalytic research.

Of course, this means that there is some degree of novelty to this research. It also means that depending upon its results, this omission will either be seen as an oversight or as altogether proper. As it turns out, the results of this experiment indicate that this is an oversight.

The experiment in this paper is also novel because no specific steganographic nor steganalytic is being studied. Rather, the steganographic and steganalytic approaches, Steghide and Stegdetect respectively, were chosen as fixed points from which to study deliberate image variation through the application of filters. They are not the target of this research. Instead, they are the means to test its hypothesis.

2.1 Steghide and Stegdetect

Since the topic of this paper is the effect of pre-embedding image filters upon steganalysis, the steganographic and steganalytic methods used in this paper are fixed, while different image filters are applied. As a matter of fact, both Steghide [5] and Stegdetect [4] are quite old. The last version of Steghide was produced in 2003 while the last version of Stegdetect was produced in 2004. These tools were chosen because they were easy to acquire and simply understood.

Given a cover image, a hidden message and a password, Steghide first compresses (and encrypts) the message. It then uses the password to generate a sequence of bit positions within the cover image; it will embed the secret message in these positions; it only flips the bits at these positions if they need flipping. Steghide also does some additional analysis in order to avoid flipping bits (it attempts to exchange instead). Essentially, this makes Steghide a fairly smart LSB (Least Significant Bit) algorithm.

Given a putative stego image, a sensitivity value (a floating point number greater than or equal to 1.0) and a steganographic method to look for, Stegdetect examines the image for statistical anomalies. As the sensitivity value is increased, Stegdetect becomes more likely to determine that the putative stego image has steganographic content. Note that Stegdetect uses a fixed statistical approach that varies only as the sensitivity is adjusted; therefore, it

requires no additional training. Basically, it is designed for use by a sophisticated user, but not necessarily an expert.

3 Experiment Design

The 1000 images used in the image databases of this experiment are from ImageNet [7]. They are all 500x375 pixels in size and they are in the JPEG format. They represent are a cross-section of the kinds of images that one might find on a typical social networking website. They contain a wide variety of lighting situations and subjects (e.g. persons, landscapes, flora, fauna). Therefore, these images, before any image filters have been applied, already contain a representative swath of the kinds of natural image variation one would find on the Internet.

Before any image filters were applied, a baseline image database was created. Like all of the image databases used in this experiment, it contained 1000 images. Half of the images (500) in this database had a message embedded and the other half were cover-only (i.e. no message). A record of the images containing embedded messages was kept so that the same images, within each of the image filter databases, would also carry a message. Thus, if image x carries a message in database A, then image x will also carry a message in database B. The converse is also true; if image y does not carry a message in database A, then image y will not carry a message in database B.

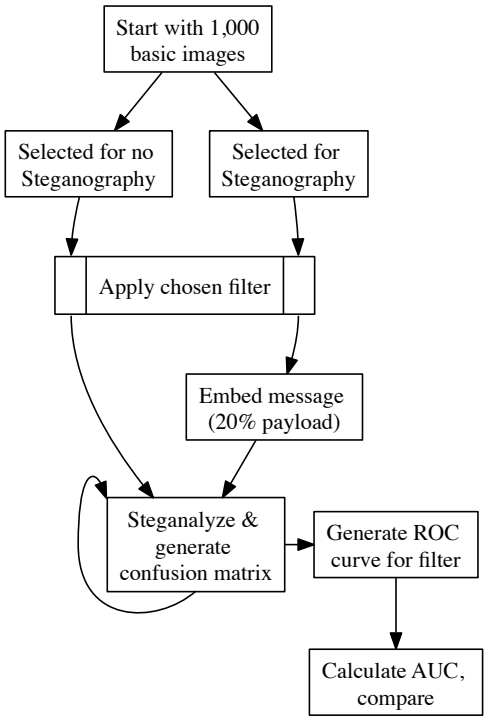

Figure 1: Flowchart for the image filtering, embedding and steganalyzing

When a message was embedded within an image, it was always the same message. The secret message was always the same size. The embedding rate for the stego images was ∼0.2 bpp (bits per pixel).[2] This embedding rate is fairly standard and neither particularly large nor small. The text of the secret message came from *Project Gutenberg*. It was the first few paragraphs of *War and Peace* [13]. See Figure 1 for more information on image filtering and embedding.

After all of the databases were created, the steganalysis began. For each image filter database, Stegdetect, using its jphide detector, was run for 50 iterations; each iteration steganalyzed all of the images in the database; Stegdetect's binary response for each image was recorded.[3] For the first iteration Stegdetect's sensitivity threshold was 1.0; during each subsequent iteration this sensitivity threshold was increased by 0.1. At the end of each iteration, FPR (False Positive Rate) and TPR (True Positive Rate) values were calculated and recorded. After capturing all of this data for each database, ROC graphs were generated and AUC values calculated that accurately characterize Stegdetect's performance over each image filter database. AUC is the primary means evaluating the impact of the image filter

[2]Given this rate and the fact that the images are all 500x375 pixels in size, 36,000 bits were embedded into each stego image.

[3]Jphide is name of the algorithm that StegHide uses for message embedding.

upon steganalysis.

3.1 Filters (Image Databases)

Twelve image filters were chosen for testing. Some were applied using Imagemagick [2], which is a popular open-source image modification package. Others were applied using Instagram-like filters that can be run using a PHP class called Instagraph [1]. The filters were chosen to be the types that Internet users themselves would apply (for aesthetic reasons). In other words, the image appearance, post filter, must not, itself, reveal the fact that a message is embedded; it must look like the kind of image that a user would upload to Instagram and similar websites.

Again, for each image filter, a database is created with the filter applied to the 1000 images from ImageNet [7]. Afterwards, half these images are embedded with the secret message using Steghide. These are the image filter databases.

In the Section 4, the performance of Stegdetect is examined upon 13 image databases; one database, baseline, contains images where no filter has been applied; the other 12 each have a single specific chosen image filter applied. Interestingly, it will be shown that these filters can affect steganalysis positively, negatively or not at all.

3.1.1 Baseline

The 1000 images within the baseline dataset have had no image filters applied. They are the same as when they were sourced from ImageNet [7]. As described above, 500 of the images in this database have the secret message embedded at $\sim$0.2 bpp (these are the *baseline stego images*) and half do not (these are the *baseline cover images*). Stegdetect's performance, in terms of AUC, upon this database will serve as the baseline for comparison of its performance against the other databases.

3.1.2 Blur

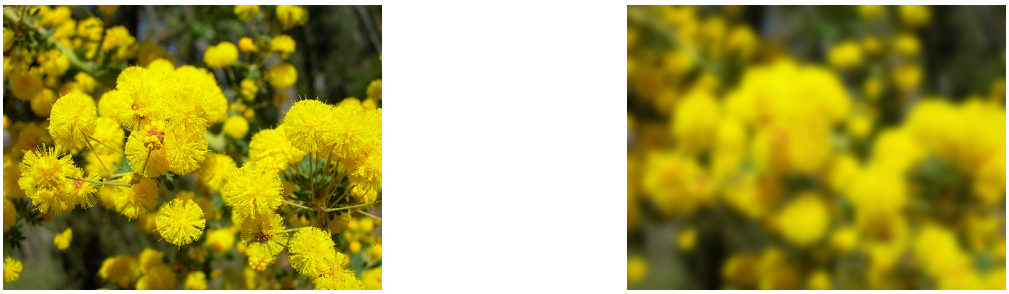

Figure 2: Comparison of baseline image (on the left) to blur filtered image (on the right)

This ImageMagick filter, shown in Figure 2, blurs the image so that the resulting image looks as though it were taken by an out-of-focus camera [2]. The blur database contains

1000 images, sourced from ImageNet [7], with this filter applied. Like the baseline dataset, 500 of the images in this database have the secret message embedded at ∼0.2 bpp (these are the *blur stego images*) and half do not (these are the *blur cover images*).

3.1.3 Contrast

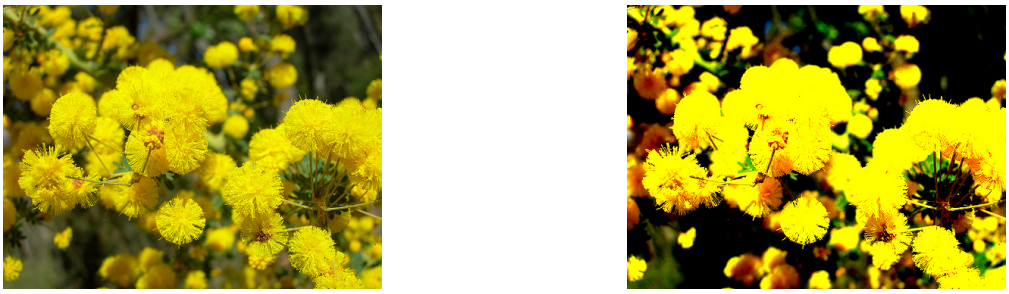

Figure 3: Comparison of baseline image (on the left) to contrast filtered image (on the right)

This ImageMagick filter, shown in Figure 3, increases overall contrast and the degree to which colors differ from one another. The resulting image may become overall darker or lighter, depending on the black/white levels of the original image [2]. The contrast database contains 1000 images, sourced from ImageNet [7], with this filter applied. Like all of the other datasets, 500 of the images in this database have the secret message embedded at ∼0.2 bpp (these are the *contrast stego images*) and half do not (these are the *contrast cover images*).

3.1.4 Emboss

Figure 4: Comparison of baseline image (on the left) to emboss filtered image (on the right)

This ImageMagick filter, shown in Figure 4, applies a simple emboss to an image. The resulting image will have changes in colors between adjacent pixels, with gray representing no change [2]. The emboss database contains 1000 images, sourced from ImageNet [7], with this filter applied. Again, 500 of the images in this database have the secret message embedded at ∼0.2 bpp (these are the *emboss stego images*) and half do not (these are the *emboss cover images*).

3.1.5 Gotham

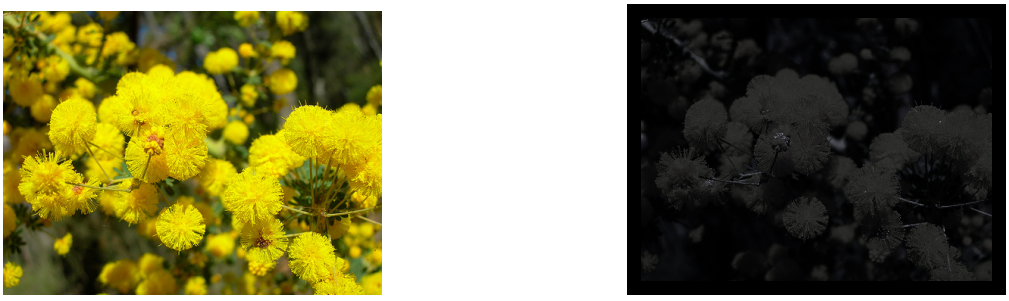

Figure 5: Comparison of baseline image (on the left) to gotham filtered image (on the right)

This Instagraph filter, shown in Figure 5, converts an image to black and white with high contrast and a blue tint. It does this by increasing brightness slightly, de-saturating the image nearly completely, adding the purple tone to the highlights, then darkening the image and increasing the contrast substantially. It also adds a small black border around the image. This filter is meant to emulate pictures taken with a Holga camera and Ilford X2 film [1]. Of all of the chosen image filters, this one has the most stark effect on the resulting image. The gotham database contains 1000 images, sourced from ImageNet [7], with this filter applied. Like the other datasets, 500 of the images in this database have the secret message embedded at ∼0.2 bpp (these are the *gotham stego images*) and half do not (these are the *gotham cover images*).

3.1.6 Kelvin

Figure 6: Comparison of baseline image (on the left) to kelvin filtered image (on the right)

This Instagraph filter, shown in Figure 6, applies an extremely strong peach-colored overlay to the image, de-saturating the image substantially so that peach is the dominant color. It also adds a washed-out frame to the edges, making the image look like a deteriorated photograph. To do this, the filter enhances brightness, de-saturates by half, and then creates an overlay with RGB values (255, 153, 0) multiplied at 50% [1]. The resulting image looks as though it had been taken decades ago and been washed out by time. The kelvin database

contains 1000 images, sourced from ImageNet [7], with this filter applied. Again, 500 of the images in this database have the secret message embedded at ∼0.2 bpp (these are the *kelvin stego images*) and half do not (these are the *kelvin cover images*).

3.1.7 Lomo

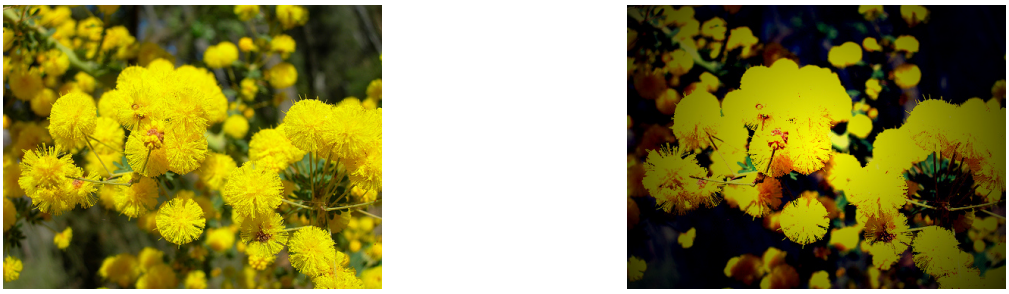

Figure 7: Comparison of baseline image (on the left) to lomo filtered image (on the right)

This Instagraph filter, shown in Figure 7, emulates cameras with plastic lenses (toy cameras), which tend to add a soft focus and shadowy border. This is done by increasing the red channel contrast substantially and applying a vignette that darkens/burns the edges of the resulting image [1]. The lomo database contains 1000 images, sourced from ImageNet [7], with this filter applied. Like the other datasets, 500 of the images in this database have the secret message embedded at ∼0.2 bpp (these are the *lomo stego images*) and half do not (these are the *lomo cover images*).

3.1.8 Nashville

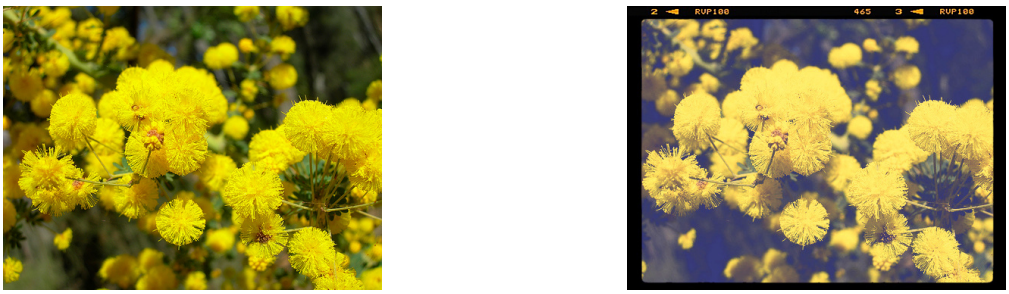

Figure 8: Comparison of baseline image (on the left) to nashville filtered image (on the right)

This Instagraph filter, shown in Figure 8, emulates old fashion photos from the 80's. The resulting image has a peach tint and a frame around it that makes it look like a slide. This is accomplished by changing shadows to an indigo-esque color, changing highlights to a peach-esque color, increasing overall image contrast and saturation, and then correcting the brightness level [1]. The nashville database contains 1000 images, sourced from ImageNet

[7], with this filter applied. Again, 500 of the images in this database have the secret message embedded at ∼0.2 bpp (these are the *nashville stego images*) and half do not (these are the *nashville cover images*).

3.1.9 Noise

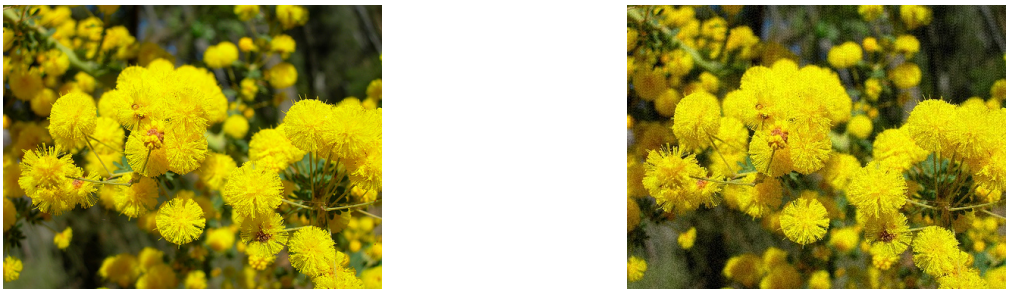

Figure 9: Comparison of baseline image (on the left) to noise filtered image (on the right)

This ImageMagick filter, shown in Figure 9, adds monochromatic noise to the image, blending it over the image at 50%. The resulting image has an appearance that looks somewhat like film grain [2]. Interestingly, the size of the resulting image can, at times, increase dramatically while the visual appearance may change very little. The noise database contains 1000 images, sourced from ImageNet [7], with this filter applied. Like the other datasets, 500 of the images in this database have the secret message embedded at ∼0.2 bpp (these are the *noise stego images*) and half do not (these are the *noise cover images*).

3.1.10 Saturate

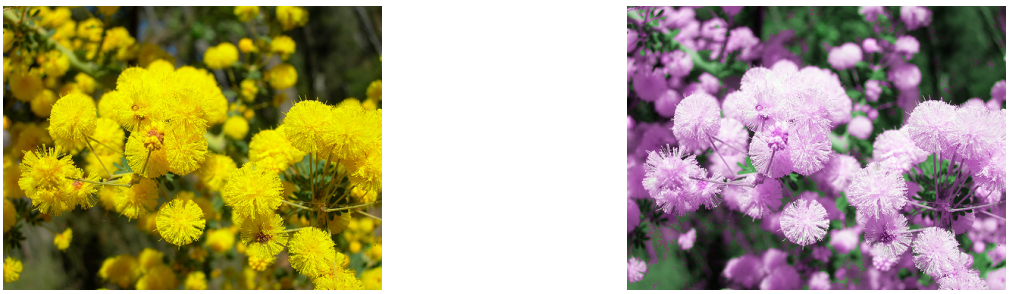

Figure 10: Comparison of baseline image (on the left) to saturate filtered image (on the right)

This ImageMagick filter, shown in Figure 10, deepens the colors of an image, with colors with a grayish hue being converted to the same colors without the grayish hue. The resulting image tends to have a lower point of brightness, and a higher level of contrast between colors [2]. The saturate database contains 1000 images, sourced from ImageNet [7], with this filter applied. Again, 500 of the images in this database have the secret message embedded at

∼0.2 bpp (these are the *saturate stego images*) and half do not (these are the *saturate cover images*).

3.1.11 Sigmoidal

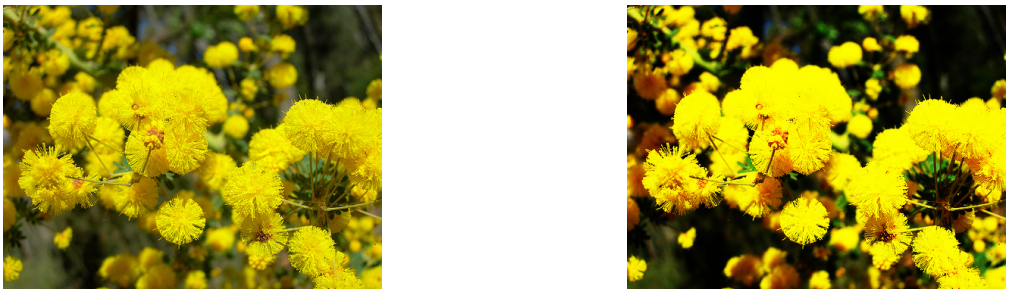

Figure 11: Comparison of baseline image (on the left) to sigmoidal filtered image (on the right)

This ImageMagick filter, shown in Figure 11, increases the contrast of the image using a sigmoidal transfer function without saturating highlights or shadows [2]. The sigmoidal database contains 1000 images, sourced from ImageNet [7], with this filter applied. Again, 500 of the images in this database have the secret message embedded at ∼0.2 bpp (these are the *sigmoidal stego images*) and half do not (these are the *sigmoidal cover images*).

3.1.12 Tiltshift

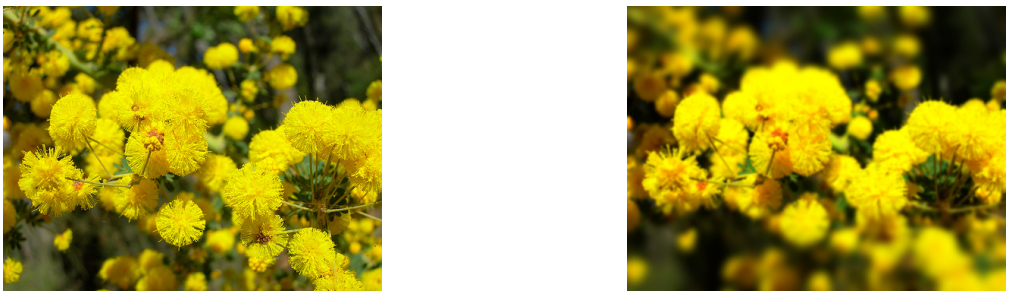

Figure 12: Comparison of baseline image (on the left) to tiltshift filtered image (on the right)

This Instagraph filter, shown in Figure 12, is meant to emulate "tilt shift" photography, in which the camera lens is rotated relative to the image plane. The effect tends to blur the top and bottom of the resulting image, focusing on the center subject being photographed. This is accomplished by increasing overall brightness and contrast and applying a blur effect to the top and bottom of the image at a bit of an angle [1]. The tiltshift database contains 1000 images, sourced from ImageNet [7], with this filter applied. Like the other datasets, 500 of the images in this database have the secret message embedded at ∼0.2 bpp (these are the *tiltshift stego images*) and half do not (these are the *tiltshift cover images*).

3.1.13 Toaster

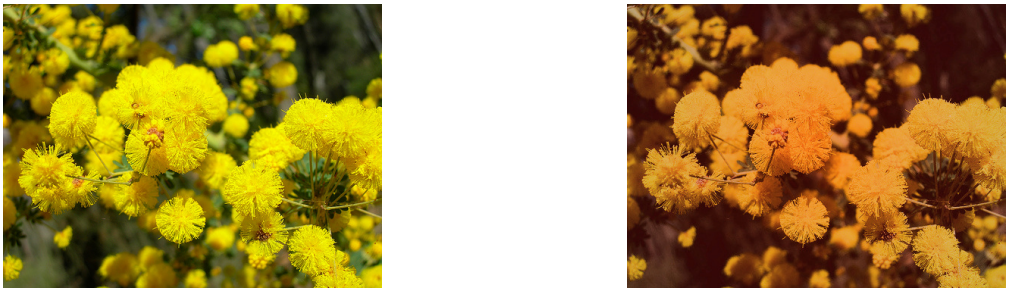

Figure 13: Comparison of baseline image (on the left) to toaster filtered image (on the right)

This Instagraph filter, shown in Figure 13, is meant to emulate old Polaroids. It gives the resulting image a pinkish glow with increased color saturation. It converts blacks to dark red, increases brightness, de-saturates a bit, and increases the contrast. This filter also adds a vignette, a de-saturation around the edges of the resulting image giving its borders a somewhat "burnt" look [1]. The toaster database contains 1000 images, sourced from ImageNet [7], with this filter applied. Like all of the other datasets, 500 of the images in this database have the secret message embedded at $\sim$0.2 bpp (these are the *toaster stego images*) and half do not (these are the *toaster cover images*).

3.2 Hypotheses

The null hypothesis being tested for this experiment is: The introduction of superfluous variation into cover images does not negatively affect the performance of Stegdetect. The alternative hypothesis being tested for this experiment is: The introduction of superfluous variation into cover images can negatively affect the performance of Stegdetect. The alternative hypothesis is confirmed if a significant number of the image filters being used in this experiment are shown to significantly, and negatively, affect the performance of Stegdetect.

4 Results

Image Database	Stegdetect AUC
Kelvin	0.692
Tiltshift	0.660
Saturate	0.603
Noise	0.597
Baseline	**0.576**
Lomo	0.572
Sigmoidal	0.539
Contrast	0.525
Blur	0.503
Random	*0.500*
Nashville	0.500
Toaster	0.498
Gotham	0.485
Emboss	0.468

Figure 14: Stegdetect AUCs over each image filter database

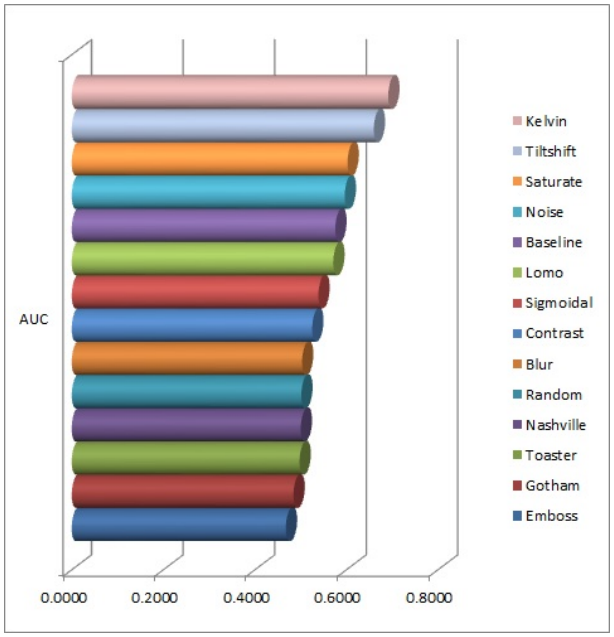

Figure 15: Comparison of Stegdetect AUCs over each image filter database

As can be seen in Figures 14 and 15, the alternative hypothesis was confirmed. The performance of Stegdetect was significantly diminished over 7 different image filter databases. Against the blur, nashville and toaster databases, Stegdetect's performance was no better than the random classifier. More impressively, its performance was worse than the random classifier against the gotham and emboss databases.[4] Over these two databases, it would be better to decide whether an image has an embedded message by flipping a coin than by using Stegdetect. Stegdetect's performance over these 7 image filter databases represents a significant degradation from its performance over the baseline database and, therefore, is confirmation of the alternative hypothesis. The ROC graphs of Stegdetect's performance over the image filter databases where it was degraded are displayed in Figure 17.

However, somewhat unexpectedly, the results of this experiment also seem to indicate that the performance of Stegdetect can be improved through the application of certain image filters. The performance of Stegdetect appears to have significantly improved over 4 databases (kelvin, tiltshift, saturate and noise) and very significantly so over kelvin and tiltshift. Of course, this unexpected result does not confirm a hypothesis posited in this experiment. However, it does indicate the need for more work in this area. The ROC

[4]In normal classification circumstances, if an AUC is less than 0.5 the classification determination can be flipped, thereby yielding a classifier with an AUC greater than 0.5. In the case of Stegdetect, this would be very difficult, if not impossible, since it would require prior knowledge of applied image filters, i.e. one would have to identify, exactly, which filter was applied and then flip Stegdetect's determination accordingly.

graphs of Stegdetect's performance over the databases where it was improved are displayed in Figure 16.

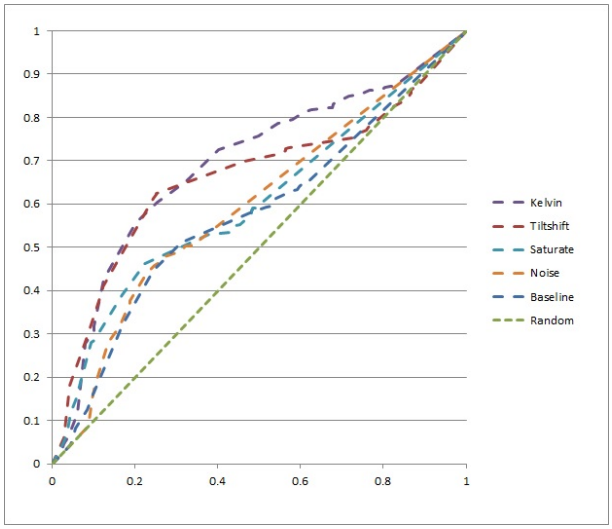

Figure 16: ROC graphs for databases where performance improved over baseline

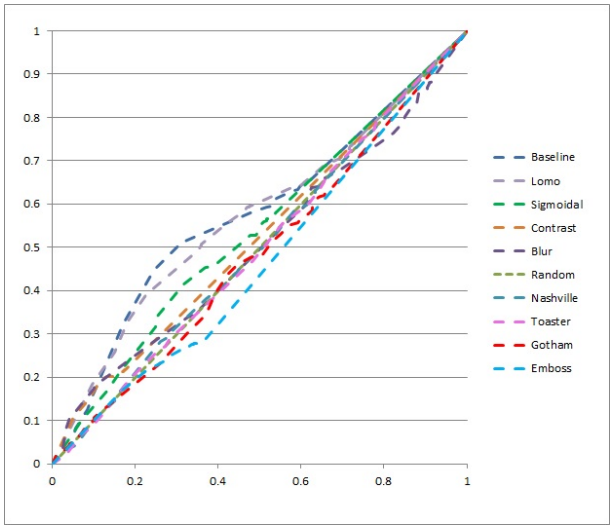

Figure 17: ROC graphs for databases where performance degraded under baseline

The results of this experiment show that the performance of Stegdetect can, indeed, be degraded by the introduction of superfluous variation into cover images. This result is confirmed by Stegdetect's performance, in terms of AUC, over 7 different databases, each constructed by introducing different variations (filters). Moreover, the results of this experiment suggest that the performance of Stegdetect can also be improved by the introduction of superfluous variation into cover images.

5 Future Work

There are a number of ways this research could be improved and expanded. In terms of expansion, more image filtering techniques could be applied and evaluated. Only a small number were chosen for evaluation amongst many, many possibilities. Some of these other possibilities may have an even larger affect upon Stegdetect or other stegnanalyzers.

This research could also be expanded by testing combinations of filters in order to study how combined filtering techniques affect steganalysis. It is reasonable to assume that combination of two image filters, both of which have been separately shown to have a negative effect upon steganalysis, would, combined, have an even more negative effect. However, this can only be shown through research. Also, if combining filters does have additional effect, then how do they combine? Incrementally? Cumulatively? Only further research could reveal this detail.

In terms of improvements, this research would greatly benefit by using more up-to-date steganography and steganalysis techniques. There are many, many possible techniques for

both available. Obviously, the image filters that had such a significant effect on Stegdetect may have less (or more) of an effect upon some other steganalyzer. Furthermore, the image filters may bolster some other steganography method more than Steghide. These are open and interesting areas for new research.

6 Conclusion

In the experiment described in this paper, the claim that the introduction of superfluous variation into cover images can negatively affect the performance of Stegdetect was tested. This claim was confirmed. Seven different variation types (filters) negatively impacted the performance of Stegdetect. The experimental results also suggested that the performance of Stegdetect could be improved by certain variation types.

The results of this experiment point the way to improved steganography and improved steganalysis. By introducing certain image variations into cover images, steganographers can increase the resistance of their techniques to steganalysis. Furthermore, given the ubiquity of the use of image filters, they can do so without making their stego images appear any more suspicious.

Steganalysts can, on the other hand, use the results of this experiment to better tune test databases in order to evaluate steganalytic techniques. Future test image databases should include more than just the natural types of variation that occur through digital photography. They should also include deliberate variations that are introduced through the modern image filters used by Internet users.

References

[1] Create instagram filters with php. http://net.tutsplus.com/tutorials/php/create-instagram-filters-with-php.

[2] Imagemagick, version 6.8.5. http://www.imagemagick.org/script/index.php.

[3] Instagram. http://instagram.com.

[4] Stegdetect, version 0.6. http://www.outguess.org/detection.php.

[5] Steghide, version 0.5.1. http://http://steghide.sourceforge.net.

[6] Rajarathnam Chandramouli, Mehdi Kharrazi, and Nasir Memon. Image steganography and steganalysis: Concepts and practice, 2004.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[8] Jessica Fridrich, Miroslav Goljan A, and Dorin Hogea B. New methodology for breaking steganographic techniques for jpegs, 2012.

[9] Jessica Fridrich and Miroslav Goljan. Practical steganalysis of digital images - state of the art. In *In Proceedings of SPIE*, pages 1–13, 2002.

[10] Jessica Fridrich, Miroslav Goljan, and Dorin Hogea. Steganalysis of jpeg images: Breaking the f5 algorithm. In *in 5th International Workshop on Information Hiding*, pages 310–323. Springer-Verlag, 2002.

[11] Miroslav Goljan, Jessica Fridrich, and Taras Holotyak. New blind steganalysis and its implications. In *in Proc. of the SPIE, Security, Steganography, and Watermarking of Multimedia Contents VI*, pages 1–13, 2006.

[12] Siwei Lyu and Hany Farid. Detecting hidden messages using higher-order statistics and support vector machines. In *In 5th International Workshop on Information Hiding*, pages 340–354. Springer-Verlag, 2002.

[13] Leo Tolstoy. War and peace. http://www.gutenberg.org/cache/epub/2600/pg2600.txt, 1869.