High quality content reconstruction in images using the BPCS Steganography algorithm

Hendrik J Kolver

August 17, 2014

Abstract

Abstract of this paper and what I wrote in it. This will give the reader a short overview of what they can expect to find in this paper.

1 Introduction

This section gives a brief description of the concepts used in this report. The reasoning behind the proposed method will also be explained in 1.5.1 and 1.5.2. This will serve to to show the applicability of the proposed method as well as the shortcomings it attempts to address.

1.1 Overview

Self recovering images (described in 1.4) poses an interesting problem. There is an inherent trade-off (described in 1.5) between the quality of the image after embedding and the quality of the image after recovery. This report proposes a method in which a high capacity embedding algorithm (described in 1.5.2) is applied to the problem of self-recovering images to attempt to increase the quality of the images (described in 1.5.1) both after embedding and after recovery.

1.2 Steganography

Steganography is the process of hiding information in such a way that it does not get detected [3]. Steganography, derived from Greek means "covered writing". This means it differs from cryptography which does not attempt to hide information but merely scrambles it so it cannot be understood. Steganography does not scramble the information but rather relies on the information to remain hidden.

1.3 Fragile Watermarking

Fragile watermarking is a variation of traditional watermarking where a watermark is embedded into an image such that subsequent alterations to the watermarked image can be

detected with high probability [5]. This means that if modifications are made to the watermarked image it would be possible to detect that modifications were made as well as what parts of the image were modified. This is a necessary component in self recovering images because in order to be able to recover image data it first needs to be determined what data needs to be recovered. It thus needs to be know what data was modified. A fragile watermark is used for this purpose.

1.4 Self embedding and recovery

Self embedding is a scheme where image content is embedded into the image itself. This embedded content can then later be used to recover damaged or modified parts of the image without accessing the original image itself [2]. A self embedding and recovery scheme such as the proposed method contains a couple of high level steps. Firstly the authentic image content is embedded into the original image along with a fragile watermark. This watermark will help determine later on which parts of the image was modified if any (see 1.3). When the authentic image needs to be recovered, after modification, the fragile watermark is used to determine which content was modified. An attempt is then made to extract the original content for the areas that were modified from the modified image. If this process is successful the user is left with a high quality reconstruction of the original image.

1.5 Quality Trade-off

There is an inherent quality trade-off present within self embedding schemes. The trade-off exists between the image after embedding and the image after recovery. If an embedding algorith such as LSB(Least significan bit) embedding is used (as in 2.1.1) the actual bit values of the original image are overwritten. Naturally the more bit values are overwritten the more noise is introduced into the image. This effectively reduces the quality of the image after embedding.

The solution to this would be to simply embed a small amount of information into the image to keep the noise low. However this presents another problem. The quality of the image after recovery is dependent on the amount of authentic image data embedded into the image.

To achieve maximum quality after embedding the embedded information must be as little as possible. To achieve maximum quality after recover the embedded information must be as large as possible to be as close as possible to the original image. There exists then a trade-off between the two.

1.5.1 Increasing quality with capacity

It might be possible to increase the quality of the image after recovery while still keeping the quality after embedding high. It is proposed that applying an embedding algorithm designed for high capacity while retaining quality (as discussed in 1.5.2) to the problem of self embedding could possibly increase the quality of the image after embedding as well as

after recovery. This is because such an algorithm should provide higher embedding capacity while producing less noise in the image.

1.5.2 High capacity embedding algorithms

The proposed method in this paper attempts to apply the algorithm for BPCS Steganography [1] to the problem of self recovering images, specifically to increase overall quality.

The goal of the BPCS algorithm is to embed as much data as possible into a cover image without detection [1]. This refers to detection by human perception as well as statistical analysis although the latter is not the focus of this paper. The BPCS algorithm uses an adaptive scheme to classify blocks that are suitable for embedding (similar to 2.1.3). BPCS steg differs however in the fact that the blocks can be conjugated in a checker board pattern to increase complexity and decrease detectability if needs be.

The BPCS algorithm can achieve a very high capacity while still remaining undetectable through human perception. This makes the BPCS algorithm a possible good choice to increase the capacity of a self recovering image scheme and possible increase the quality through this process.

2 Current self embedding and recovery schemes and algorithms

This section compares current Content reconstruction algorithms using self embedding. This serves to give an overview of what is currently available and to possibly highlight shortcomings of the current algorithms. This would be useful to determine if the proposed method does indeed improve on existing methods. It would also help highlight strengths and shortcomings in the proposed method. The methods analyzed were chosen because they all provide good quality cover and reconstructed images. They also each offer an unique approach or aspect to the image recovery problem and would thus provide a good idea of the different approaches that have already been tested.

2.1 Overview

2.1.1 Erasure channel model utilizing the remaining authentic content

The method proposed in [4] uses an erasure channel as a model for the content reconstruction problem. The method uses LSB embedding to embed the reference data into the image itself. The method also uses a global spreading technique to spread the reference data across the image. They also propose that by using the remaining authentic content in the image it is possible to have a high tamper rate while at the same time achieving good quality images before and after recovery.

The method proposed in [4] achieves good quality cover images with a PSNR >35dB. The

method also achieves an image recovery quality of 35dB <PSNR >40dB and 40dB <PSNR with taper rates of 50% and 33% respectively. The method does not let the restoration quality of the image deteriorate much if the tampering rate is increased up to a value of 50%. They do however note that they can only achieve minimal reconstruction performance increases by decreasing the amount of information in the reconstruction reference. By using only 50% of the available capacity the maximal tampering rate increases form 50% to 59%.

This method [4] is thus quite robust since 50% of the image may be tampered with before recovery starts to deteriorate. The security for this method is also very good since the quality of the cover image is not very susceptible to visual checks. The authors did not do any statistical analysis on the image. The embedding capacity of this method is also acceptable, but because the method uses some of the authentic image data to aid in the recovery the quality of the image, before and after recovery, is still very good even without a very high embedding capacity.

2.1.2 Method using difference expansion

[9] Proposed a method that uses difference expansion and generalized LSB embedding. Difference expansion works by exploiting the high redundancy that are present in images. With difference expansion the payload is embedded in the difference of pixel values. For a pair of pixel values (x,y). [7] It should however be noted that image quality reduces rapidly as the payload size is increased when using difference expansion. The method uses these two techniques in combination to achieve a high embedding capacity while keeping distortion relatively low. The method achieves a PSNR >35dB after embedding. The method achieves an embedding capacity of 1.78bpp when using up to the 4th LSB on a 512x512 8bit gray-scale version of the Lena image.

The method's [9] restoration quality is acceptable at roughly 50%. The difference expansion this method uses provides extra space for embedding. The authors did thus not implement compression on the image data because of the extra space the difference expansion provides. The embedding capacity of this method could thus be further improved by compressing the embedded data. This could possibly lead to better quality than what their experimental results achieved at the expense of complexity. Difference expansion thus seems a good solution to increase the embedding capacity while still keeping the distortion low.

The authors do not mention the image tamper rate that this method [9] allows.

2.1.3 Dynamic block allocation for self embedding

(Qian et al) [6] proposes a method for fragile watermarking with good restoration capabilities using self embedding. The proposed method differs from the other methods mention in 2.1.1 and 2.1.2 due to the fact that the proposed method does not embed the information into all blocks uniformly. The proposed method classifies different blocks of the image according to the block smoothness. The less smooth the block is the more information will be embedded into it. The authors argue that the current methods that use a fixed embedding size for each

block are inadequate since less information should be embedded into the very smooth blocks and more should be embedded into the rough blocks [6].

The advantage of using this dynamic scheme is that there would be less visual distortion to the cover image. This is because if the specific block is already very rough (very busy visually) the human eye would not notice small changes. If however the specific block is very smooth (very uniform visually) the human eye is more likely to notice small changes. The proposed method thus creates less distortion in the cover image than fixed size methods while still retaining good restoration quality [6].

The method proposed by (Qian et al) [6] uses the 3 LSB bit planes to embed the needed information in. The method achieves good results in experimentation with a cover image PSNR >37dB as well as a reconstructed image PSNR = 35dB. The method also allows for a tamper rate <35%. The authors do not analyze the security of the algorithm.

2.2 Comparison

Each of the methods described in 2.1.3, 2.1.1 and 2.1.2 provide good quality cover images as well as good quality reconstructed images. There are however important differences. The method described in 2.1.1 offers a very good tamper rate of 50% compared to the method described in 2.1.3 which has a tamper rate of 35%. This means that a larger of the cove image can be tampered with while still being able to restore the image with good quality.

The method described in 2.1.2 achieves a cover image with a PSNR >35dB, the method described in 2.1.1 achieves similar results, however the method described in 2.1.3 achieves a cover image PSNR >37dB. This means that the method described in 2.1.3 would have less noise in the cover images after embedding and would thus have better overall quality of the cover images.

At almost equal tamper rates of around 33%-35% the method described in 2.1.1 achieves a restoration PSNR >40dB whereas the method described in 2.1.3 only achieves a restoration PSNR = 35dB. The method described in 2.1.2 only achieves a restoration quality of about 50% the original image. This means that the method described in 2.1.1 would generally produce better quality restored images than the methods described in 2.1.3 and 2.1.2.

This means that the method described in 2.1.1 provides the best tamper rate and the best restored image quality, and the method described in 2.1.3 provides the best cover image quality. The method described in 2.1.2 provides decent quality of the cover image as well as the restored image.

All three methods thus provide good results and would serve as a good benchmark for comparing the method described later in this paper.

3 Implementation

3.1 Embedding

3.1.1 BPCS Steg

In order to be able to recover the image after it has been tampered with the image needs to be embedded into itself. For this embedding the proposed method used BPCS Steganography [1].

This method works with the following steps:

1. Get the binary representation of the image
2. Convert the binary representation into Gray code
3. Divide the image into bit planes
4. Divide the bit planes into 8x8 blocks
5. Calculate the complexity for each block //TODO find reference for how this is done
6. Divide the image content reference to be embedded into similar binary 8x8 blocks
7. Calculate the complexity for each image reference block and conjugate if not complex enough
8. for each image block that is complex enough, replace that block with the next image reference block
9. Convert the blocks back to binary representation
10. reconstruct the image from the processed blocks

The proposed method works with colour images which means that for each pixel there is a red, green and blue value. This means that each block effectively store 3 bits for each pixel.

BPCS Steg has an inherent property that if any part of the image is modified the embedded message will be lost. This would obviously be a problem for the proposed method since the image will definitely be modified.

As a solution to this problem each block used for embedding is treated as a separate embedding entity. There is thus no overlap between two different blocks used for embedding. The image reference is first process using the method explained later in this section. Each block has 12 pixels (with a bit depth of 16 bits) of the reference image embedded into it. An embedding map containing the location of the first pixel to be embedded in the block is also created. Since the reference image pixels are embedded in order it is only necessary to know the location of the first pixel. This embedding map is then embedded into the LSB layer of that specific 8x8 block of pixels This allows the extractor to be able to identify where all the

extracted pixels belong in the image even if the image has been modified. This serves to give random access to the embedded content which is an essential property of self embedding as defined by (Korus et al. [4])

Total space available for embedding per block = (8*8)*3 = 64*3 = 192
Total embedded bits = 12*16 = 192

Image resizing The image that is embedded into the cover image cannot be embedded with perfect quality due to space restrictions caused by the embedding algorithm. The proposed method however does not utilize compression as it is beyond the scope of this paper. The proposed method rather reduces the bit rate of the image from 24 bits to 16 which reduces the size of each pixel by 6 bits. The image's size is also reduced according to the space available for embedding. When extracting the image reference data again the image is scaled up to it's original size of 512x512 pixels before restoration

Block Spread It is very important how the embedded content is spread across the image because this affects the tamper rate
//TODO put reference here
The proposed method uses a simple method for spreading the image reference content across the image. The image reference content to be embedded is first divided into blocks as per the BPCS steg algorithm. These blocks are subsequently stored in a list. The created list is then reversed before embedding. Reversing the list means that the image reference content that contains the bottom right part of the original image will now be embedded into the top left part of the cover image. This creates an effect where the embedded reference content is far away from the content that it is referencing. This is important because if a piece of the cover image is destroyed it would not be able to recover it if that destroyed piece's reference content was destroyed with it.

Conjugation //TODO Look into references for this
If a image content reference block is found to not be complex enough that block needs to be conjugated before embedding. This is important because if a block is not complex enough it cannot be detected when extracting and it may cause visual distortions to the image since it is replacing a complex block.

The block complexity is calculated by using a border complexity method. That is for each binary bit that has an adjacent binary bit that is different that it the border count is increased. The final border count is then divided with the theoretical maximum border count, which would be a grid with a checkerboard pattern of 1's and 0's.

To conjugate a block each value in the grid needs to be changed to the XOR value of the original grid value and the corresponding value in a checkerboard grid (starting with a 1 in the top left hand corner). To reverse the conjugation this process simply needs to be repeated.

This method of conjugation is a simple method that makes a block more complex while still allowing the original values of the block to be retrieved.

3.1.2 Fragile Watermark

In order to be able to detect which parts of the image have been tampered with a fragile watermark can be used.

An initial attempt was made to apply the method proposed in [8]. This method is a reversible fragile watermark which means that the watermark can be removed to restore the original content bit by bit before watermarking if the image was found to be authentic.

The method needed to be adapted to authenticate the image on a block level, the image is divided into 8x8 blocks. The method requires compression of bit streams in order to embed the watermark and to be able to extract the watermark again. The bit streams are rather small in the proposed method because they are only applicable to a specific 8x8 pixel block. This results in the fact that there is not always enough redundancy to be able to compress the bit streams enough for embedding.

Another problem with this method was the large amount of noise that this method of watermarking introduces into the cover image since it also uses difference expansion as mentioned in [7].

The fragile watermark used in the proposed method embeds a calculated hash into the image that is used to authenticate the content when the image is processed on the receiving end.

For each of the 8x8 pixel section in the image a hash is calculated by combining the binary of each of the 7 Most significant bit layers of that section. This binary stream is then used as the input to the SHA-256 hash function the output is a 256 bit hash string for that image block. This hash string is then truncated to a length of 57 bits. This is done to enable the conjugation map as well as the embedding map that was generated when embedding the image content (3.1.1) to be stored at the end of the stream.

//TODO - Find reference for truncation of hash security, also explain why a 57 bit hash is sufficient

The conjugation map is appended to the truncated hash string to create a binary string of 192 bits. These 192 bits are finally embedded into the least significant bit plane of that specific section of the cover image effectively replacing all the bits in that bit plane with the generated bit stream.

3.2 Extraction and Restoration

3.2.1 Image authentication

The first step for the image extraction and restoration process is to determine which, if any, parts of the image has been tampered with. This is accomplished by first completing steps 1-4 of the BPCS steg algorithm (3.1.1). for each block (block refers here to 8x8 pixel blocks, thus a block here contains all 8 bit planes) the authentication hash string is calculated as described in 3.1.2. The embedded hash is then extracted from the Least significant bit plane of that block. The extracted hash is then compared with the calculated hash and if they match the block is authentic.

3.2.2 Image restoration

For each authentic block the 7 most significant bit planes have their complexity calculated. If a bit plane block is found to be complex enough it is assumed that there is image information embedded into it. The embedded conjugation map is extracted at this point and if it is found that any of the complex enough bit planes were conjugated the conjugation algorithm is applied again to reverse the conjugation.

The embedded image content is extracted at this point from the processed bit planes and the embedded image is reconstructed. In order to know the location of the extracted pixels the embedding map that was embedded into the LSB bit plane of that image section is consulted 3.1.1. The reconstructed image is then scaled up back to it's original size of 512x512 pixels. The reconstructed image is then also split into pixel blocks. The original pixel blocks now get iterated and each block that was found not to be authentic is replaced by the corresponding reconstructed pixel block.

This process restores the tampered areas of the image if their matching pixel blocks could be extracted from the image. That is if the embedded blocks were not destroyed in the image modification. Obviously if too large a part of the image has been modified it would not be possible to accurately reconstruct the image because too much of the embedded image content would be lost.

4 Results

In testing the proposed method a simple experiment was set up in which the image was processed using the embedding process described in 3.1. The results of that process can be seen in the following two images, showing the original Image and the image after the embedding process. Note that in this case the Image Reference content was only embedded once

//TODO get reference for PSNR

After the image after embedding was generated the PSNR of the embedded image in relation

Figure 1: Embedding Once. Left: Original Image, Right: Image After embedding

to the original image was calculated. In this instance the PSNR after embedding was: 34.92dB

The next step in the experiment conducted was to modify the Image after embedding (shown in figure 1). After the Image was modified it was processed using the method described in 3.2. The results of this process can be seen in the following two images, showing the Image after Tampering and the Image after Restoration.

Figure 2: Embedding Once. Left: Image After Tampering, Right: Image After Restoration

After the image (shown in figure 2) was restored the PSNR of the Image After Restoration was calculated in relation to the Original Image (shown in figure 1). In this instance the PSNR after restoration was: 34.89dB.

It was found in the experimentation process that if the Image reference content is only embedded once that the quality of the Image after embedding as well as the quality of the

Image after restoration was relatively high. It was however found that the tamper rate possible was quite low at <3%. To attempt to remedy this problem another instance of the experiment was run with the only independent variable being the fact that the Image reference content now being embedded twice. The results can be seen below

Figure 3: Embedding Twice. Left: Original Image, Right: Image After embedding

Figure 4: Embedding Twice. Left: Image After Tampering, Right: Image After Restoration

In order to embed the Image reference content twice the content needs to be half the size that is used when embedding once. This reduces the quality of the Image after Restoration from the previous 34.89dB to 32.52 dB. The tamper rate however is now increased from <3% to <35%

This experiment was also completed on the following image. First Embedding the Image reference content once:

The PSNR of the Image after embedding (show in figure 5) was 27.94dB.

Figure 5: Embedding Once. Left: Original Image, Right: Image After embedding

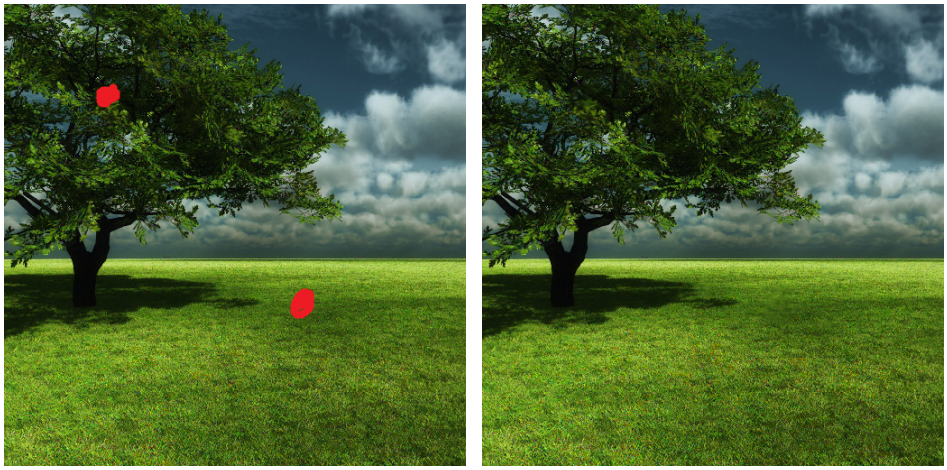

Figure 6: Embedding Once. Left: Image After Tampering, Right: Image After Restoration

The PSNR of the Image after restoration (show in figure 6) was 27.72dB. The tamper rate, as was the case with the lena image, was also <3%. The experiment was also conducted by embedding the Image reference content twice. As is shown below.

The PSNR of the Image after embedding (show in figure 7) was 27.93dB.

The PSNR of the Image after restoration (show in figure 6) was 26.05dB. The tamper rate, as was the case with the lena image, was increased from <3% to <35% with a similar reduction in quality experienced.

The second image used thus behaved in a similiar manner to the first image with the only significant difference being the much lower PSNR values produced

Figure 7: Embedding Twice. Left: Original Image, Right: Image After embedding

Figure 8: Embedding Twice. Left: Image After Tampering, Right: Image After Restoration

5 Conclusion

//TODO Write the conclusion

References

[1] Steve Beaullieu, Jon Crissey, and Ian Smith. Bpcs steganography. *University of Texas at San Antonio*.

[2] Jiri Fridrich and Miroslav Goljan. Images with self-correcting capabilities. In *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on*, volume 3, pages 792–796. IEEE, 1999.

[3] Neil F Johnson and Sushil Jajodia. Exploring steganography: Seeing the unseen. *Computer*, 31(2):26–34, 1998.

[4] Pawel Korus and Andrzej Dziech. Efficient method for content reconstruction with self-embedding. *Image Processing, IEEE Transactions on*, 22(3):1134–1147, 2013.

[5] Eugene T Lin and Edward J Delp. A review of fragile image watermarks. In *Proceedings of the Multimedia and Security Workshop (ACM Multimedia'99) Multimedia Contents*, pages 25–29. Citeseer, 1999.

[6] Zhenxing Qian, Guorui Feng, Xinpeng Zhang, and Shuozhong Wang. Image self-embedding with high-quality restoration capability. *Digital Signal Processing*, 21(2):278–286, 2011.

[7] Jun Tian. Reversible watermarking by difference expansion. In *Proceedings of workshop on multimedia and security*, pages 19–22, 2002.

[8] Jun Tian. Wavelet-based reversible watermarking for authentication. In *Electronic Imaging 2002*, pages 679–690. International Society for Optics and Photonics, 2002.

[9] Jun Tian. High capacity reversible data embedding and content authentication. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 3, pages III–517. IEEE, 2003.