BPCS Steganography

Steve Beaullieu, Jon Crissey, Ian Smith
University of Texas at San Antonio

Abstract

Our project implemented the BPCS (Bit Plane Complexity Segmentation) technique to embed data into bitmap files. The ultimate goal is to embed as much data as possible into a cover image without detection by human perception or statistical analysis. Our first attempt to implement this hiding technique was on 8-bit grayscale images as our cover object. After accomplishing that version, we manipulated it into a second version that was also capable of using 24-bit color images.

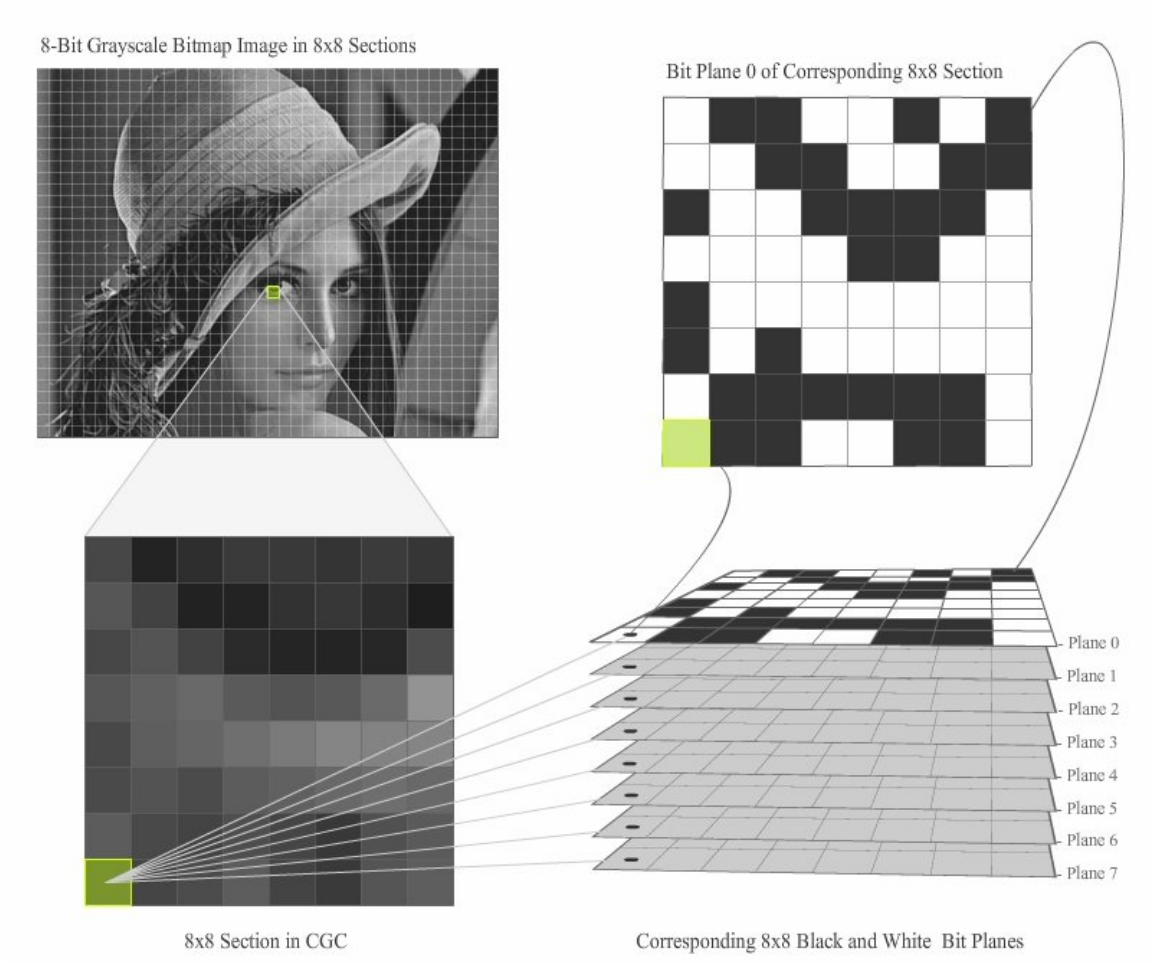
This paper will first outline the BPCS embedding and extraction technique for grayscale images and explain the subtle differences in the color version. It will also compare and contrast the results of embedding data at different thresholds and capacities for both grayscale and color images.

Hiding and Extracting Data

We start off by converting a sample 8-bit grayscale image into CGC (Canonical Gray Coding) form. CGC allows us to manipulate each bit plane without affecting the other bits that represent each grayscale value.

8x8 pixel blocks are segmented within the image and each of the bits (8 bits per pixel) in CGC form will have their own corresponding 8x8 plane. Visually, this would be like slicing the 8x8 planes into 8 8x8 black and white bit planes (*see CGC diagram*). Each bit plane will be measured for complexity, which is determined by the number of borders (transitions between black and white in each pixel plane) present in an 8x8 bit plane versus the maximum borders possible. If a region is complex enough, we will embed our data into the cover image, which is broken up into appropriately sized 8x8 blocks for each bit plane.

If the data to embed (8x8 blocks at a time) in the cover file is statistically complex, it can be embedded into the complex blocks of the image. If not, we will conjugate (exclusive or) the data with a checkerboard pattern (the most complex pattern possible) to ensure complexity. There will be a conjugation bit in each plane that will show whether the data was conjugated with a checkerboard pattern. This technically gets rid of 1 bit of embedding capacity per 8x8 region giving 63 bits to embed per 8x8 bit plane.



CGC Diagram

Once the data has been embedded, the image is converted back into the original format from CGC and saved.

Extraction is basically the same as embedding, except if a bit plane is determined to be complex, it will then look at the conjugation bit and extract the data accordingly. Because the embedded data in the complex regions has to be complex, the complex regions before and after embedding data will remain complex.

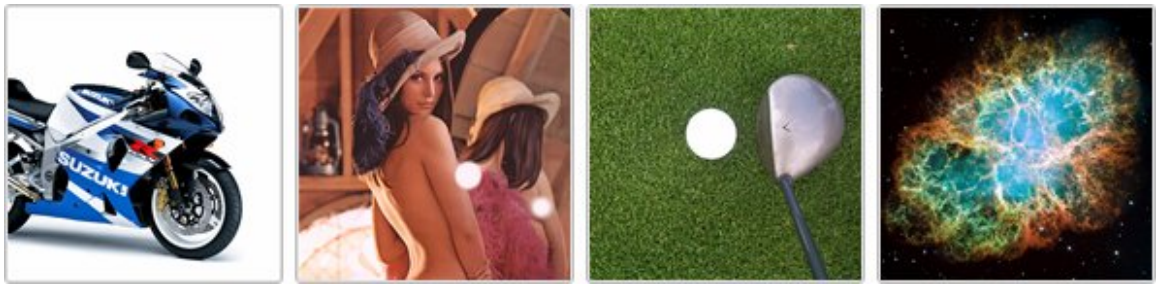
Color is basically the same process. However, it will have 3 8-bit grayscale values that represent each color, thus giving approximately three times the file size and three times the embedding capacity (to its corresponding grayscale version). A subtle other difference is that the color file has a slightly different file structure that does not contain a palette for the pixel values.

Results

Overall, the results went fairly as expected. Increasing the threshold at which bit planes are determined to be complex decreased the embedding capacity, but also decreased the distortion. Embedding at full capacity (based upon the threshold) of the image including every bit plane proved to add distortion (although typically worse at lower thresholds) because the higher bit planes are visually much less tolerant to change.

An interesting observation is that grayscale images had a slight advantage over the color images in the sense that only grayscale values could be changed as opposed to a combination of values for each color plane. When pushed to higher limits, the grayscale images could look altered compared to the original, but still appear unaltered without comparison to the original. In color images, when pushing the limits of threshold and capacity, noticeable color distortions occurred that clearly indicated some kind of change to the original image.

The cover images chosen for analysis behaved expectedly as well. We carefully chose images that were increasingly more complex to see if our expected capacity change and threshold affects would behave as planned. Our images included a sharp-edged motorcycle image with a solid background, the classic Lena, grass with a golf ball and driver, and a nebula.



Cover Images (640x640 – 1.2MB)

When embedding at reasonable (< 30% of capacity) levels for these images, the solid areas did not appear to have any distortion, although under strenuous conditions, the transitions between solid and textured areas were affected in their corresponding 8x8 sections. This was important to prove that data was only being embedded in complex regions. With the golf ball and nebula, embedding on every bit plane proved to be acceptable at a certain threshold. This gave us a capacity near 50% without noticeable human perception.

There are several factors that heavily influence what data output is possible for our analysis. The alpha threshold that determines whether regions are complex enough to embed in is important, but the amount of data that is embedded versus the embedding capacity at that threshold is important as well. Ultimately, one of the largest factors that

influences the appearance of the final image with the embedded file is how many significant bit planes have been embedded into.

For data analysis, embedding a certain percentage of the file's embedding capacity initially seemed like a good idea, but because each image would have a differing amount of bit planes that are affected based upon a set percentage of their embedding capacity, it was determined not to be a fair comparison.

Human analysis is fairly subjective, and by using it on top of embedding percentages that affected images differently proved to be an apples and orange comparison. So we slightly altered the program to determine a capacity based upon the number of significant bit planes that we would like to embed in. This way, when determining whether a file looked significantly altered, we added the number of bit planes for embedding use as a command line argument. This made it easier for the group to subjectively decide on whether an image appeared to be altered.

We used two different alpha thresholds for complexity that were chosen because of their dramatic clarity difference between the two for most images at a set capacity. In addition to these thresholds, the number of significant bit planes to embed in were used for our human and consecutive statistical analysis. The number of bit planes in which to embed changed for each image based upon human perception. At each complexity, when changing the number of significant bit planes to embed in, we left the embedding amount at full capacity of the file based upon those parameters. While this still may not be perfect as far as scientific data analysis goes, human perception is one of the most crucial parts of the project.

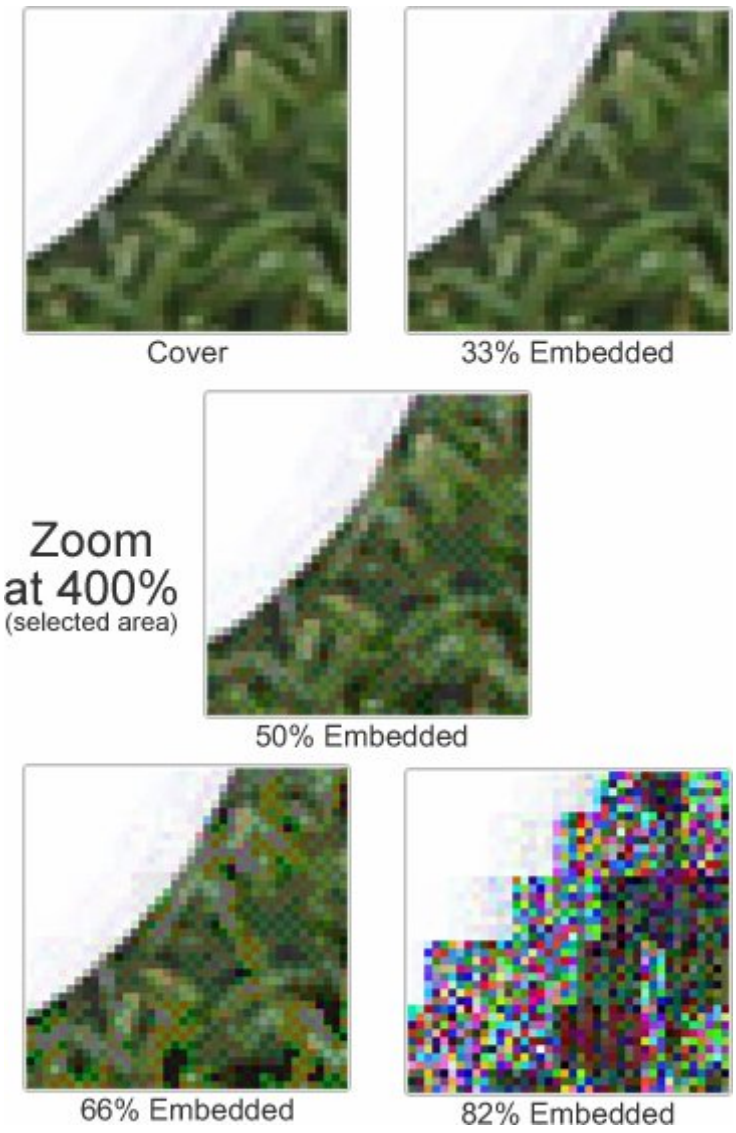
The entropy of the original cover image and corresponding standard deviation based upon the histogram were spared from human perceptual judgment. However the compression ratio and standard deviation of the embedded file had to be based upon our subjective determination as to when the image looked one bit plane less than altered.

For final clarification, a complexity threshold and the number of significant bits available for embedding were used as command line arguments along with the cover image. Based upon these values, a maximum capacity was determined for embedding. Random data as mentioned in the previous section of this paper was made to the full embedding capacity of the image based upon the parameters and embedded. To check that this was working correctly, the random data was compared before and after extraction and proved to be a success.

The following examples were made using an alpha threshold that would provide just enough capacity to embed the percentage of data into the cover image relative to the cover image file size. The higher bit planes are effected in the higher embedding percentages, which is much more noticeable than the lower bit planes. The example with 82% shows how in order to get that much data into the image, the threshold had to be lowered to a point where most blocks are considered complex.

When using the 6th and 7th bit planes (*see CGC diagram*) it is clear to see that data has been embedded. It is also visible how the embedding goes from lower left to upper right and then upward when looking at the extreme cases where the contrast between embedding and not embedding data in two different higher CGC bit planes is noticeable.

Golf Images at Relative Embedding Capacities



Golf Images at Relative Embedding Capacities, Zoomed

BPCS Embedding Human Perception Comparison

Grayscale	Entropy	Full Capacity		Human Perception		Compression Ratio		$SDev_{orig}$	$SDev_{embed}$	
		α = 0.3	α = 0.4	α = 0.3	α = 0.4	α = 0.3	α = 0.4		α = 0.3	α = 0.4
golf	7.13	229KB,0.573	193KB,0.482	5	8	0.9808	0.9866	1787	1698	1729
lena	7.35	154KB,0.384	117KB,0.292	6	8	0.9477	0.9689	1568	1543	1555
nebula	7.01	155KB,0.388	125KB,0.314	8	8	0.9767	0.9911	3479	3535	3500
suz	4.98	76KB,0.191	50KB,0.127	4	5	0.9027	0.9423	12006	11871	11924

Color	Entropy	Full Capacity		Human Perception		Compression Ratio		$SDev_{orig}$	$SDev_{embed}$	
		α = 0.3	α = 0.4	α = 0.3	α = 0.4,	α = 0.3	α = 0.4		α = 0.3	α = 0.4
golf	7.44	917KB,0.764	726KB,0.606	4	8	0.9913	0.982	4230	4192	3866
lena	7.61	816KB,0.682	462KB,0.385	4	4	0.9127	0.9373	3409	3188	3281
nebula	7.09	702KB,0.585	484KB,0.404	4	5	0.8577	0.9444	9574	8385	9762
suz	4.90	346KB,0.289	226KB,0.189	4	5	0.7953	0.8382	36573	35921	36120

// NOTES

α is the threshold at which blocks are considered complex

Entropy is the amount of randomness in the cover file

Full Capacity is the embedding capacity of the cover image per corresponding α and the ratio of the embedding capacity/cover size

Human Perception is the maximum bit plane at which distortion is no longer noticeable based upon the corresponding α

Compression Ratio is the ratio between the compressed cover file and compressed stego file (Cover file)/(Stego File)

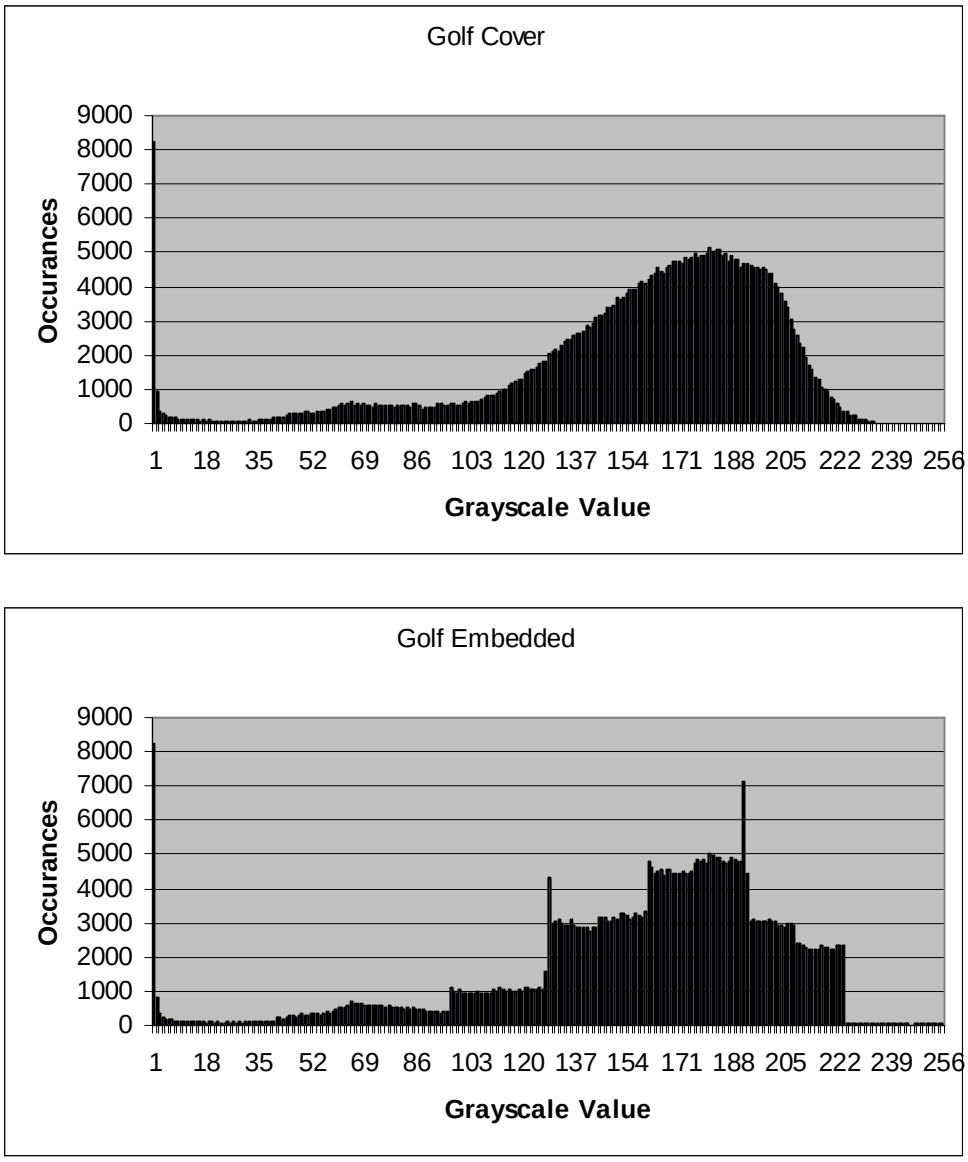
Standard Deviation is based upon the difference in grayscale values from 0 - 255

$SDev_{orig}$ is the original standard deviation before embedding the data

$Sdev_{embed}$ is the standard deviation after embedding data at the respective α complexity

Detection and Destruction

The histogram values varied significantly between images of different textures. Some images with more even textures that covered most of the image ended up having a smooth looking histogram, but after embedding data, the histogram became more rough and/or plateau like. Others that had sharper transitions or more contrast had very little difference in the histograms. Surprisingly, the standard deviation based upon the histogram values before and after embedding data did not vary significantly. However, a standard deviation did give somewhat of a clue to the image's contrast or large areas of flat color. If an image had high contrast or areas of flat color, there were extreme spikes at certain points of the histogram that led to higher standard deviation values.



Golf histogram (1 color plane) before and after embedding 50% capacity

As far as the destruction of our embedded data goes, there were not any measures put in to prevent an active attacker from destroying our data. Any bit change or affine transformation would completely destroy our embedded data and render it useless. While this may seem like a drawback, the BPCS technique is typically used for capacity as opposed to robustness, such as in watermarking.

Technical Difficulties

One of the first problems we encountered is that even though the research papers went over the general direction of the BPCS hiding technique, they conveniently omitted many of the specific details needed for implementation. Figuring out exactly what CGC was and its implementation was time-consuming, but crucial to the BPCS process.

Another detail that was not fully explained was the exact order of embedding the data into the cover image. Instead of going through bit plane 0 for each of the 8x8 blocks and then progressively going to the higher bit planes as necessary, our first working program went through each bit plane of every 8x8 block. Since embedding in the 8x8 blocks goes from lower left to upper right, we noticed that with a lower threshold and half of the embedding capacity of the cover image that the embedded blocks on the lower part of the image were much more distorted than the upper 8x8 blocks that did not have any data embedded. Later, we embedded from least to most significant bit, one bit plane at a time for each 8x8 block to prevent embedding in the higher bits if unnecessary.

A method of creating sample data to embed that approached the actual embedding size of the images at first seemed like an arduous task. We first thought of changing the size of some images to get different sized large files (because we have a high embedding capacity). However, this would not be too accurate and would take forever. The next idea was to manually create large text files because they could be made to the exact byte size that we needed for our embedding capacity. However, this proved to take forever and the data was a lot less random because of the copied lines.

Our solution was to create random data (8-bit ASCII characters) inside the program that was based upon the embedding capacity of the file. This was just for our data analysis and not for the actual demonstration of embedding and extraction capabilities. The randomness of the data may seem to have a significant effect on the visual or statistical results, but even if a region of the embedding file was not very random, it would still be exclusive ored with a random pattern, thus making it complex either way.

Another problem included accidentally converting to CGC twice. The image was converted to CGC when determining the capacity and later in the data hiding function. This caused the images to appear as if they had gone through a contrast increase. The more frustrating part was the fact that the embedding and extraction worked, but produced the distorted output files.

The most unsuspecting problem arose in the data analysis. With such subjective material and endless parameters to work with, it took us days to finally come up with the structure our final report analysis, which could still be considered less than perfect.

Future Direction

Several improvements could be implemented to our program to make the images with embedded data safer from detection or extraction by an unwanted user. One would be to encrypt the data before embedding it into the image. Another improvement would be to randomly distribute the data instead of linearly going through each plane. In addition to this improvement, random data could be inserted in areas between valid data, especially because this technique has a higher embedding capacity.

Run length irregularity or border noisiness could be considered improved methods of determining complexity. A speculative improvement that would be much more complex to implement would be to use areas of other than 8x8 for embedding, or use areas of varying size.

Another measure to improve human perception (or lack thereof) would be to have the definition of complexity increase for each progressive significant bit plane. Having a set definition for complexity is not smart because the higher bit planes suffer more from the same complexity settings. This would take much more testing to determine good values, but would be interesting nonetheless to see what would happen to capacity and perceptibility based upon the adaptive definition of complexity.

References

1. Eiji Kawaguchi and Richard O. Eason
   Principle and Applications of BPCS-Steganography
   Kyushu Institute of Technology, Kitakyushu, Japan – University of Maine, Orono, Maine

2. Niimi, Michiharu, Noda, Hideki and Segee, Bruce
   A study on Visual Attack to BPCS Steganography and Countermeasures
   Kyushu Institute of Technology, Kitakyushu, Japan – University of Maine, Orono, Maine

3. Ross J. Anderson, Fabien A.P. Petitcolas
   On the Limits of Steganography
   IEEE Journal of Selected Areas in Communications, 16(4):474-481, May 1998

4. Hioki Hirohisa
   A Data Embedding Method Using BPCS Principle With New Complexity Measures
   http://www.i.h.kyoto-u.ac.jp/~hioki/research/DH/files/abcde_steg02_revised.pdf

5. Jeremiah Spaulding, Hideki Noda, Mahdad N. Shirazi, Eiji Kawaguchi
   BPCS Steganography Using EZW Lossy Compressed Images
   Pattern Recognition Letters 23 (2002) 1579-1587