Steganography

4th November 2016

TEAM:

1. Riya Bubna - 13CS10041
2. Shrey Garg - 13CS10045

WHAT IS STEGANOGRAPHY?

Steganography is the practice of concealing a file, message, image, or video within another file, message, image, or video. The advantage of steganography over cryptography is that the intended secret message does not attract attention to itself as an object of scrutiny. Plainly visible encrypted messages—no matter how unbreakable—arouse interest, and may in themselves be incriminating in countries where encryption is illegal. Thus, whereas cryptography is the practice of protecting the contents of a message alone, steganography is concerned with concealing the fact that a secret message is being sent, as well as concealing the contents of the message.

Various Existing approaches of Image Steganography

- **LSB Insertion Image**: Least significant bit insertion (LSB) is a simple common method to hide data into a cover image. In an image the 8th bit of a particular byte or all of the bytes is replaced with a bit of secret image or message. In a color image there are Red, Green and Blue color components. So, 3 LSBs can be used to store in a pixel.
- **RGB Intensity based Image Steganography:** For insubstantial colors, expressively additional bits can be altered for every byte channel of an RGB color image. The basic concept of this algorithm is that, as compared to higher value, the lower color-value of a channel has a smaller amount of influence change on the overall color of the pixel. As a result, additional bits can be altered in a low value channel than high value channel.
- **Spread Spectrum Image Steganography:** Spread spectrum communication defines the procedure of distribution the bandwidth of a narrowband frequency over a wideband of frequencies. We can achieve this by spreading the narrowband waveform over a wideband waveform, for example white noise. Once spreading done, the energy or strength of the narrowband (or weak) signal in every frequency band is small and as a result hard to detect or perceive.
- **DCT based Image Steganography:** This method gives higher capacity as compared with earlier methods. If we consider stego image superiority, image quality is greater than the other methods. The capacity of message which is being embedded is increased, but the image degrades as it changes all DCT coefficients of every block.
- **DWT Image Steganography:** The DWT (Discrete Wavelet Transform) divides the image in frequency components. High frequency components are detailed coefficients which hold the additional information about image, used for embedding secret image. Low frequency components are approximate coefficients that hold almost the original image.

METHODOLOGY

Taking the above into consideration, we decided on using a modified version of bit insertion.

- Data is not directly encoded. We compress data using a compression library called zlib and then encoded that.
- Zlib helped reducing the size of the data by compressing it and hence enabling us to encode more data.
- We used 3 bits from each colour pixel to encode the data. We realized that modifying 3 bits per color pixel does not cause much distortion in the image.
- This enabled us to use 9 bits per image pixel to encode our data. Three bits for each colour Red, Green and Blue (as we used the RGB image format).
- Each character took 8 bits to encode. The zlib format was RFC 1950 with ASCII values in the range -128 to 127.
- The encoded string was followed by a terminating character which was predefined by us and a zlib specific number needed to decode the image.
- The pixels were filled one after the other row by row till all the image pixels were exhausted.
- The code was written in OpenCV and the language used was C++.
- No code fragment was copied and the entire code was written by us.

Example:

Input string: This is a sample test string. [Size: 29]

Compressed string: xÚVTÉÈ,V [Size: 7]

Encoded format: <compresssed_string><terminating_string><zlib_specific_compressed_size>

- Each character in the compressed string requires 8 bits to be stores. Hence each character is encoded by 8/9th of a bit.
- The terminating string that we use is “##” and this requires 16 bits. Finding this pattern tells us when out encoded test has ended.
- We save 24 bits to store the zlib specific size. This needs to be retrieved in order to decode the compressed text.
- Hence we take a total of 24+12 = 36 = 4 image pixels extra to store the parameters needed to decode the image. This is very less in comparison to the compression that zlib provides and hence allows us to encode more data.

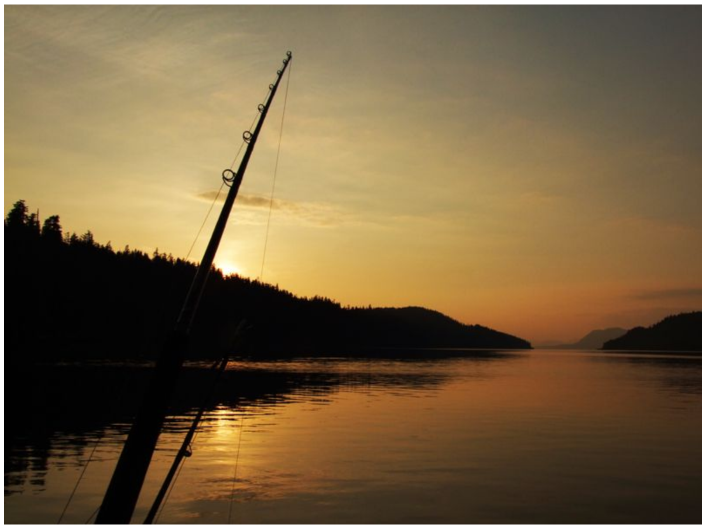
TEST CASES

Test Case 1:

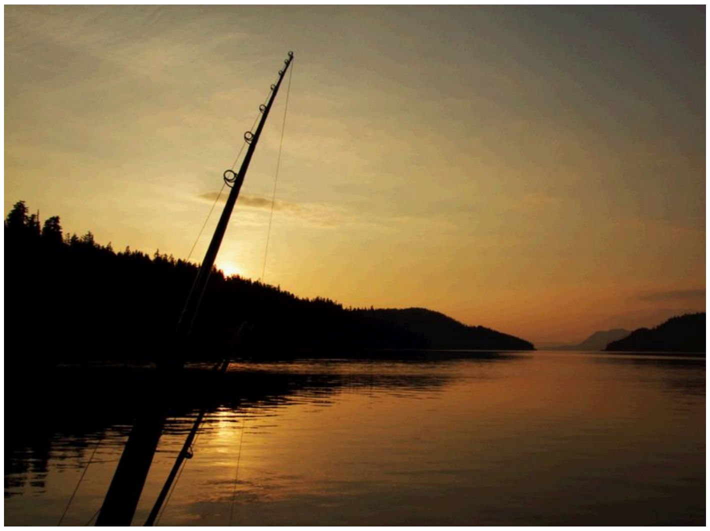
Encoded Text: If ever there is tomorrow when we're not together.. there is something you must always remember. You are braver than you believe, stronger than you seem, and smarter than you think. But the most important thing is, even if we're apart.. i'll always be with you

[Uncompressed string size: 260]

Original Image



Encoded Image



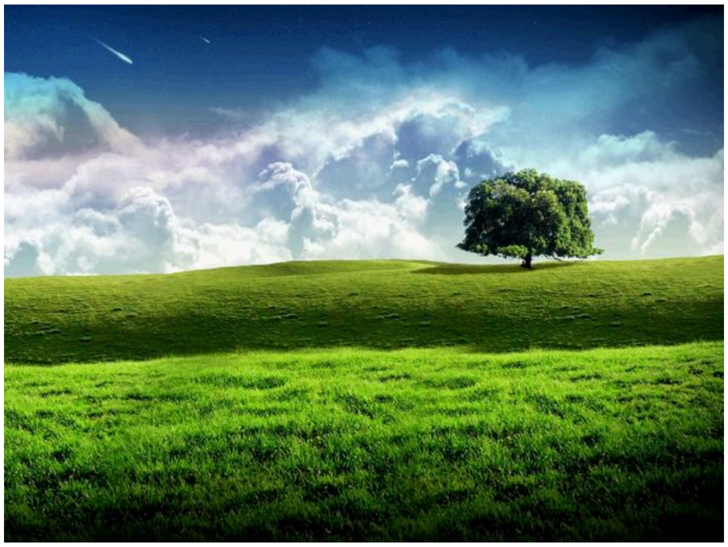
[Compressed string size: 164]

Test Case 2:

Encoded Text: I just read the first page of Lolita, and it was just so good that I put it down and appreciated it.“Lo-lee-ta: the tip of the tongue taking a trip of three steps down the palate to tap, at three, on the teeth. Lo. Lee. Ta. She was Lo, plain Lo, in the morning, standing four feet ten in one sock. She was Lola in slacks. She was Dolly at school. She was Dolores on the dotted line. But in my arms she was always Lolita.”I can’t think of anything else that is this beautiful... Maybe the start of Fahrenheit 451. [Uncompressed string size: 518]

Input Image

Encoded Image

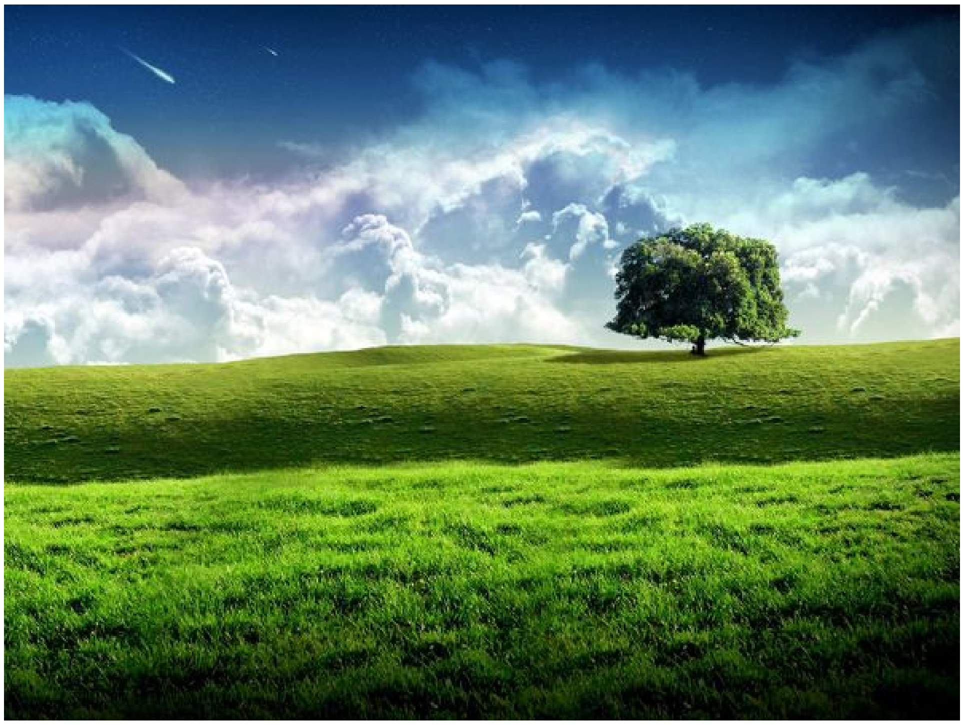


[Compressed string size: 316]

Test Case 3:

Encoded text: The discoveries people made about each other. The only thing that didn't bore me, obviously enough, was how much money Tim Price made, and yet in its obviousness it did. There wasn't a clear, identifiable emotion within me, except for greed and possibly, total disgust. I had all the characteristics of a human being–flesh, blood, skin, hair–but my depersonalization was so intense, had gone so deep, that the normal ability to feel compassion had been eradicated, the victim of a slow, purposeful erasure. I was simply imitating reality, a rough resemblance of a human being, with only a dim corner of my mind functioning. Something horrible was happening and yet I couldn't figure out why–couldn't put my finger on it. The only thing that calmed me was the satisfying sound of ice being dropped into a glass of J&B. Eventually I drowned the chow, which Evelyn didn't miss she didn't even notice its absence, not even when I threw it in the walk-in freezer, wrapped in one of her sweaters from Bergdorf Goodman. We had to leave the Hamptons because I would find myself standing over our bed in the hours before dawn, with an ice pick gripped in my fist, waiting for Evelyn to open her eyes. At my suggestion, one morning over breakfast, she agreed, and on the last Sunday before Labor Day we returned to Manhattan by helicopter. [Uncompressed string size: 1347]

Input image:



Encoded Image:

[Compressed string size: 753]

CONCLUSION

Hence we can safely conclude that it is possible to encode a large amount of data in images effectively without drastically affecting the image. There is always a tradeoff between the amount of data that can be compressed with the amount of distortion that we can allow in the image. If we wanted the image to look a bit less distorted, we could have used less than 3 bits per colour pixel to encode data. But that would have resulted in less carrying capacity.

As is visible from the above examples, the compression format reduces the data size, encrypts it in a special format and encoding the image with this data doesn't modify the image much. Hence, the change is hardly visible and our data is safely hidden in the image.

REFERENCES

- Zlib-compression
- Steganography
- Work done on Steganography