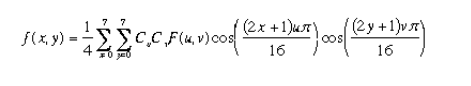
Alex Johnson

Final Project Paper

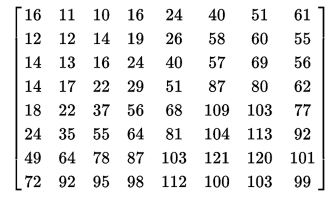
For the final project, I decided to implement image-based steganography to hide plain-text files in .png image files. Image steganography is a very similar concept to cryptography. Cryptography uses a secret key to encrypt messages and in this way, an eavesdropper knows that a secret is being distributed, but has no way of recovering the original message without having the secret key. With Steganography, an image (or video, or audio file) is used to encode the secret message. If implemented properly, the human visual system will not be able to notice any difference between the original (cover) image, and the encoded (stego) image. In this way, an eavesdropper will not be able to determine whether or not a secret message is being transmitted by simply looking at the image. I chose to do my research on image steganography because I found the idea of hiding data in an image to be a very cool and interesting technique for transmitting secrets.

After doing some preliminary research, I found that there are two forms of image steganography. The more simple or basic form is referred to as the spatial domain. In this form of steganography, the bits of the secret message replace the least significant bits of each pixel value. In this way, since only the least significant bits are being changed, the stego output image will look identical to the original. This is known as the LSB substitution method, and is probably one of the most straightforward ways to implement image steganography. With this method, I used a color image, where most sources on the Internet used a black and white image, so there was only one value associated with each pixel. Using a color image made the implementation slightly more challenging, as I had to split up each character of the message into bit values to encode in each of the three color values for each pixel. I ended up putting the three most significant bits in the blue color field, three in the green color field, and two in the red color field. As long as the decoder knows how the secret message is embedded, they will be able to decode the image, and retrieve the original message.

The other form of steganography is referred to as the frequency domain. In the frequency domain, the image is split up into 8x8 blocks of pixels. Each block is then converted using a Direct Cosine Transform into the DCT domain, using the following equation:



Where C­u and Cv are equal to 1/(21/2) when u or v are equal to 0, and equal to 1 otherwise. Once this is done, the each element of the block is divided by the corresponding element in a quantization matrix, and then rounded to the nearest integer. The matrix I used is below:



Once this is done, we go through each element of the resulting matrix, and if the value is not equal to 0 or 1, we replace the LSB of that value with the next bit of the secret message. Once we reach the end of the block, or we reach the end of the secret data, the block is multiplied by the quantization matrix and the inverse DCT equation is used to get the block to look like normal pixel values again. If this is all done properly, the resulting stego image will look identical to the cover to the human visual system, and a proper decoding algorithm will be able to retrieve the original message.

I was able to complete the first half of this project fairly easily, as there is not a lot of thought that goes into the first method. As can be seen, the LSB substitution method only used about 70 lines of code for the embedding, compared to about 200 lines used for the DCT method. With the DCT method, I had a lot of trouble figuring out what algorithm to use, how to implement it, and how to make sure everything was working. For some reason, I could not find a complete algorithm describing the entire process from start to finish of how to embed a secret message into an image using a DCT transform, and how to retrieve the message from the stego image. I had to gather parts of the process from many different sites, and adapt the parts so they all matched up.

After completing the algorithm and testing my code various times, I noticed a problem that seemed inevitable for the DCT method. I noticed that for some images, and some secret message inputs, the retrieved output is not correct. This was the case when I entered about 100 characters as the input, and two characters in the output were incorrect. I thought about it, and realized that when we round to the nearest integer in the quantization step, we are losing some information about the image. This will cause the values of the pixels to be slightly altered when we convert back to the spatial domain. In addition to this, when we convert back to the pixel values, we may get values that are outside of the [0,255] range for accepted color values. If this happens, the OpenCV library automatically takes the remainder over 256 and uses that value. This is problematic because if we have a blue value of 256, then the pixel should have a lot of blue in it, but when the library takes the remainder, the blue pixel value will be 0. To prevent this from happening, I checked the values and if they were greater than 255, I set them to 255, and if they were less than 0, I set them to 0. This way we minimize the error from rounding, but also lose some data, which may contribute to the infrequent error in the decoding process.

After completing the implementation for both steganographic methods, I analyzed the two for efficiency and security. There is no question that the LSB substitution method is far more efficient than the DCT conversion method, however the difference in runtime for the two methods is almost unnoticeable. As far as security goes, both methods are secure as long as any eavesdropper does not know how the message is encoded. If the encoding algorithm is discovered, then the security falls apart. However, I found that there is one major benefit to the DCT method, and that is that it is resistant to a compression attack. A compression attack is where an eavesdropper intercepts the image, compresses it, and forwards it to its destination. Assuming data is lost in this compression, the LSB substitution method would most likely fall apart here because the pixel values would not necessarily be the same as before the compression, so the embedded message would be lost. If this were done to an image that encodes a message using the DCT conversion method, then assuming the compression is done using DCT, which most compression is done with, the secret message will still be attainable. This is because the message is encoded in the image after the image has already been compressed by a DCT conversion and quantization. Assuming that the compression is not done on an extreme level, the secret message will still be retrievable by the receiver. In conclusion, the DCT conversion method in my opinion is the more practical method for image based steganography because the runtime is still very fast, and it is resistant to a compression attack.

This project introduced me to a whole new field of computer security. I enjoyed working on steganography over the past couple of months, and found that a vast amount of research has already been done in the field. In the future, I would like to do more research in the field, particularly with watermarking images and/or videos. This application intrigued me because I see it in use all over the Internet, whether it is a professional photography claiming ownership of a photo, or a news source claiming ownership over some video or footage. I like to think that steganography very similar in concept to public keys, in that the encoding and decoding steps are different, and that proof of ownership or authenticity is achievable. Overall I enjoyed this project, and hope I get the opportunity to do more in the field of steganography in the future.

Bibligraphy

1. <http://www.sciencedirect.com/science/article/pii/S0165168409003648?np=y>
2. <http://www.csie.ntu.edu.tw/~b89044/home/IPR/2DINVERSE.pdf>
3. <http://www.computing.surrey.ac.uk/teaching/2006-07/csm25/Chapter6/jsteg-h.pdf>
4. http://www.svcl.ucsd.edu/courses/ece161c/handouts/DCT.pdf