

Steganography project

by Axl West

FILE	17796_AXL_WEST_STEGANOGRAPHY_PROJECT_78196_959842484.D OC (5.49M)		
TIME SUBMITTED	21-APR-2016 01:06PM	WORD COUNT	11235
SUBMISSION ID	56197003	CHARACTER COUNT	55299



PRIFYSGOL CYMRU

Y Drindod Dewi Sant

UNIVERSITY OF WALES

Trinity Saint David

SWANSEA - ABERTAWE

Investigation Into LSB Steganography
and its use in a Games Environment

By
Axl West

Supervisor: Gordon Dickers
Second Marker: Carlene Campbell

Project submitted as part of the requirements for the award of
Bsc(Hons) Computer Games Development
April 2016

I, Axl West declare that I am the sole author of this Project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

This work has not previously been accepted as part of any other degree submission.

Signed :

Date :

I, Axl West hereby consent that my Project, submitted in candidature for the B.Sc.(Hons) Computer Games Development, if successful, may be made available for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

Signed :

Date :

Abstract

This project researches the affect that least significant bit steganography has on bitmap images. The projects aim is to find out how many bits of data can be manipulated before it makes the changes visually detectable to the human eye and if the colour if the bit that is manipulated influences the speed of this detection. To realise this aim several objectives must be met. Researching least significant bit steganography, developing a program to preform said steganography, using the program to produce a series of stego images, conduct an experiment using the images and finally to analyse the results of the experiment to determine how the steganography affected the image. This project produces a cross platform steganography program written in C++ and using QT. This program produces sixteen stego images using different types of least significant bit steganography. The results from the experiment that uses these images answer the primary aim of the project but cannot confirm that that red bit manipulation is easier to notice that green and green easier to notice that blue.

Contents Page

Introduction.....	Page 5
Literature Review.....	Page 6
Steganography.....	Page 6
Steganalysis.....	Page 7
Least Significant Bit.....	Page 9
Image File Types.....	Page 11
Watermarking.....	Page 13
The Human Eye.....	Page 13
Steganography and Games.....	Page 14
Current Programs.....	Page 15
Experiment.....	Page 15
Results.....	Page 17
Program Design and Implementation.....	Page 23
Program Requirements.....	Page 23
Design.....	Page 24
Implementation.....	Page 24
Testing.....	Page 25
Evaluation.....	Page 26
Conclusion.....	Page 27
Future Work.....	Page 28
Reflection.....	Page 29
Figures.....	Page 30
References.....	Page 30
Appendix 1.....	Page 33
Appendix 2.....	Page 51
Appendix 3.....	Page 52

Introduction

This project will look into Least Significant Bit (LSB) steganography and how it affects images.

Aim: to find out many bits of RGB data can be manipulated before it is possible visually to see the difference between the new image and the original and does the colour of the bit that is manipulated affect how quickly the change becomes noticeable.

This aim will be realised by completing the following objectives.

Objectives

- 1) Research LSB steganography
- 2) Develop a program that is able to perform LSB steganography
- 3) Use the developed program to hide data using differing RGB bit combinations
- 4) Conduct an experiment using the resulting images
- 5) Analyse results

The project will start with a review of current literature around the subject. This section will explain what steganography is in more detail. It will also talk about some of the research already done in the area. It will also talk about ways in which steganography can be used in games. Current steganography software will also be discussed. The next section is about the experiment. This section will talk about the experiment and how it will be done and the question it's trying to answer. It will also talk about and analyse the results of the experiment. The following section is the development and implementation of the program that makes the stego images for the experiment. This section also talks about any issues with the code while it was being implemented and about the tests done on the finished program. The next section is evaluation. This section will talk about how the project went and if it managed to fulfil the aims and objectives of the project as outlined in this section.

Literature Review

Steganography

Steganography is the technique of hiding information in plain sight, this is sometimes confused with cryptography because they both appear to do similar things [1]. The aim of steganography is to hide the very presence of a message, whereas, cryptography's aim is to make the message unreadable to anyone but its intended recipient. Stegonographic techniques have been used since the times of ancient Greece to convey messages in secret. The word Steganography comes from the Greek words for 'covered writing' [1]. Stegonography has been a favourite method of private communications for the military and political leaders [2].

Steganography has changed a lot over the years. In the early years of Steganography the techniques were simple but very imaginative. Mary Queen of Scots used steganography to hide letters, the letters were hidden in the bunghole of a beer barrel allowing the letters to be passed in and out of her prison freely [3]. Another example is when Histaiacus shaved the hair off of a messenger and then tattooed a message to the messenger's head, after the hair had grown back he was sent off to deliver the message [3], [4]. Giovanni Porta was an Italian scientist born in 1535. He described a method using ink made by one ounce of alum and a pint of vinegar that enabled you to hide a message within a hard-boiled egg. The ink penetrated the shell of the egg and left the message on the hardened surface of the egg, this made the message unreadable until the shell of the egg had been removed [5].

The present day the focus of steganography has been on digital formats by manipulating the way we store information [3]. There are three ways that you can hide a digital message within a digital cover: injection, substitution and generation [5]. The injection method hides the message within the parts of a file that are ignored by the processing application. For example hiding a message within the comments of a C++ file will not change how the program looks after it is built but the message can still be found by looking at the code. Using this method does not change any of the data in the file so in that way it is hard to detect, however depending on the size of the message hidden it can become easier to detect [5], [6]. The substitution method replaces the least significant bits of information in the cover file out for the new information from the message that is being hidden.

This is usually done in the way that causes the least amount to degradation to the original cover file. This method does however limit the size of the message that can be hidden and if too much data is hidden it can degrade the cover file so much that the messages existence becomes easier to detect [5], [6]. The final method is generation. This method requires only the message that you want to be hidden, the program will then generate you a cover file that has the message hidden within it. This eliminates one of the main flaws of both the injection method and the substitution method this flaw is that when the resulting file is compared with its original cover file it can become obvious that it has been tampered with. With the generation technique because the resulting file is also the cover file there is no other existence of the file that it can be compared to making it immune to comparison tests [5], [6].

Steganalysis

Because hiding information using steganography can cause some degradation, or unusual characteristics in the file the message is hidden within, it is possible to analyse files in an attempt to detect these flaws and discover the presence of steganography. This process is called steganalysis. Just as cryptography has cryptanalysis steganography has steganalysis.

One basic but very effective method of steganalysis is known as the known-cover attack [7]. This attack as its name suggests only works if you know and have the original cover before it had the steganography preformed on it. Using this method you can discover any differences between the two covers, during this process you can find patterns that are signatures of steganography the program. These signatures could even be enough to imply the use of steganography on images that you don't have the cover for if they display the same signatures [8].

Another form of attack is known as the stego-only attack. This is one of the hardest ways that a steganalyst could attack a steganographic cover. In this attack the steganalyst has only the stego-medium or cover available. The only possible way that this can be attacked is to just use every possible known attack and hope you get a result [9].

The next method of steganalysis is known as the known-message attack. In this attack the message and the cover are known and the steganalyst attempts to find the method of steganography used to embed the message [5]. Even though the message is known the difficulty of this attack is very high

and could be considered the same as the stego-only attack [10].

The next steganalysis attack is the chosen-stego attack. In this attack both the steganography tool/algorithm along with the cover is known, this makes the attack easier. The last type of attack is the chosen-message attack. In this form of attack the steganalyst creates a cover using a steganography tool or algorithm looking for signatures, these signatures can then be used to help detect future stego-media [5]. However in [9] they state that the above categorizations are primarily theoretical as the only goal of steganalysis is to find out whether or not a message is present.

This leads into the explanation to a less theoretical and more practical way of categorizing steganalysis attacks. This category is made up of Targeted steganalysis, blind steganalysis and semi-blind steganalysis. Targeted steganalysis works on specific stego-systems and works by looking at a specific steganography algorithm and analysing it this way you can find out the statistics that change on the cover after steganography. The results from this method are generally very accurate, however the technique is inflexible with little room to move into other algorithms [9].

The blind steganalysis technique works on all different kinds of steganography techniques. It works by learning the difference between the statistical properties of both stego images and non-stego in order to tell the difference between the two. This technique is less accurate than the targeted steganalysis technique but is a lot more expandable.

The final technique is semi-blind steganalysis this works on a range of stego-systems this range can depend on the domain they are embedded on [11]. Several papers refer to and use these categorizations [12]–[16].

The aim of steganalysis is to detect the presence of steganography however there is another form of attack that can be used on steganography and that is destruction [7]. Discovering the presence of steganography is not always necessary sometimes it is possible to remove steganography this way you only need to suspect the presence of steganography. In [7] there is a simple formula that can destroy image steganography and watermarks. There are several other forms of attacks you can use. Blur smooth transitions between sections that have significant differences in pixel colour values. This will help remove or destroy any message embedded in these regions. The opposite of blur is sharpen. This increases the differences instead of dampening them but causes the same effect to the

embedded message. Noise and noise reduction are opposite types of attacks. In a noise attack the aim is to insert random coloured pixels into an image that closely resemble the colour of the pixel they replace this has the effect of destroying any data embedded in those pixels. However in noise reduction the aim is to rearrange out pixel values.

Histograms are a graphical representation of a distribution of numerical data. As images stored on a PC are stored as a set of numerical Red, Green and Blue

(RGB) values you can create an image histogram. An image histogram is the graphical representation of the pixel intensity values (so for a grey scale 8 bit image there are 256 intensity values, and for a 24 bit image there 3 sets of intensity values each of the RGB values). Fig 3 shows an example of a grayscale image and its corresponding histogram. The



Fig3: Lena and histogram

The histogram shows the frequency that each intensity value occurs, the far left being 0 and the far right being 255. A difference image histogram can be used as a statistical analysis tool for the detection of least significant bit steganography due to the footprints it leaves behind. Where the intensity value of an image I at position (i, j) is $I(i, j)$ the difference image is $D(i, j) = I(i + 1, j) - I(i, j)$ the difference image histogram is defined as the histogram of the difference image D . The difference image histogram has been shown to be an excellent tool in the detection of LSB steganography [17], [18].

This causes some obstacles that steganography has to try and overcome. A good steganography technique should try to remain undetected even under close scrutiny and, if it can, resist destruction techniques.

Least Significant Bit

A common technique for digital steganography is the least significant bit (LSB) approach. This is a form of substitution steganography. In a 24 bit image each pixel has a different red, green and blue value that is stored in a byte (8 bits). If the colour of one of the pixels is white then each byte for that pixel is equal to 255 or in binary 11111111 each 1 being a bit. The final 1 is what is known as the least significant bit as it has less impact on the final colour of the pixel. For example if the red and green values were set to 11111111 (255) but the blue value was set to 11111110 (254) the pixel

would still appear to be white but would have the smallest tinge of blue that the human eye can't detect. Using this method of changing the LSB it is possible to hide a message within the cover file that cannot be detected by simply looking at the message. This method does have its disadvantages though, the size of the message you can hide is limited by the size of the cover file. If you have a 400 x 400 24 bit image you have 480000 bytes in your image but one character in a message takes up 1 byte so $480000 / 8$ leaves you with room to store only 60000 characters. This disadvantage can be overcome by increasing the number of least significant bits that you use this however causes its own problems as after a time it becomes noticeable just by looking at the image that it has been tampered with. Another way round that problem is to simply use a bigger cover image. That is not the only issue that LSB steganography has though. Once the message has been hidden the if the cover file is change in any way you run the risk of destroying the message hidden within, and that is a lot harder disadvantage to overcome.

In [19], [20] they talk about an improved method of using LSB steganography on 24 bit images. In [20] they decide to use a 1-2-3 ratio of LSBs. This means that for the red pixel value they modify only the LSB, for the green pixel value they modify the last 2 LSBs, and finally for the blue pixel value they modify the last 3 LSBs. However in [19] they decide upon a 1-3-4 ratio. Both papers use the same justification for the increased use of LSBs which, as mentioned above, distort the quality of the image. They can use increased LSBs on the green and blue colour values because of the way that the human visual system works, our eyes are more sensitive to the colour read as 65% of the cones in our eyes are sensitive to red, 33% are sensitive to green and 2% are sensitive to blue. This improvement has the effect of dramatically increasing the storage capacity of the cover image. In fact if you use the method in [19] you can store one character per pixel as $1 + 3 + 4 = 8$ bytes, this also make the calculation for how large a message you can store in an image as it is now a 1:1 ratio. E.g. a 400 x 400 image can store 160000 characters.

LSB steganography is done by modifying the least significant bits of data. But how can only the LSBs be modified without affecting the more significant bits. To do this you

use a bitmask. A bitmask is a value of your choosing that you can use to

compare with your original value. Let's say that we need to set the second

LSB of a value to 1 first we use a bitwise AND operation. The AND

operation allows us to query the value of a single bit. As you can see in fig 4 using a bitwise AND if the value we are checking is not equal to 1 we get a result of 0 and if it is equal to 1 we get a non 0

```

11010110 11010100
AND 00000010 00000010
= 00000010 00000000

```

Fig 4: Bitwise AND

result. At this stage if the value we were comparing was 11010110 then we would not have to do anything as the second LSB is already set to 1, however what if we were using the value 11010100. In this case we can use a bitwise XOR operation. The XOR operator Allows for a bit to be toggled from 0 – 1 or 1 – 0. To use this to toggle any bit we want we do this value $\text{value} \oplus 1 \ll X$ where value is our 11010100 value, $\oplus$ is the XOR operator, $=$ sets the value to the new value with the toggled bit, 1 is our bitmask (00000001) and $\ll X$ is the bit shift operator that will shift the bit to the left by the value of X. in this example case X would be set to 1 because we want to change the bitmask to 00000010. This will toggle the second LSB (or seventh bit) to a 1 making our value 110101100.

Image File Types

This project will use the 24 bit bitmap (.bmp) file type. This file type stores each pixel of an image as an RGB value using 1 byte per colour (1 byte = 8 bits $8 * 3 = 24$ hence 24 bit). Another type of bitmap is the 32 bit bitmap, this stores each pixel of an image with a RGBA value (the addition of the A component increases the size by 1 byte increasing it from 24 bit to 32 bit). The A stands for alpha which stores the transparency of the pixel. 32 bit bitmaps are less common than the 24 bit bitmap.

Image file compression come in two types lossy and lossless. With lossless compression when the image is uncompressed it is an exact copy of the original uncompressed image. Generally lossless files are larger than their lossy counterparts. Lossy compression may give the impression that it is a perfect copy of the original uncompressed image but it is not. Most of the time lossy compressed images have file sizes smaller than their lossless counterparts. Some lossy compression algorithms allow for different levels of compression, however the more you compress the image the more it degrades the image you are compressing. This affects steganography in a number of ways. Sending uncompressed image file looks suspicious as the majority of all image communications are done in compressed image formats [21], [22]. If LSB steganography was used on a large image it may draw suspicion if the lossless file format was still very large, this would go against the aim of steganography by drawing attention to itself. However if the image is instead compressed in a lossy manner you could potentially remove parts of or all of the data embedded into the image. If the majority of communications you have are in lossy formats and then you change to lossless in order to use steganography this change could be considered suspicious and the images looked into, again going against the aim of steganography. This meant that steganography techniques that worked on

lossy file formats had to be developed.

The JPEG format is a lossy file format for the storage images. It is the most commonly used image format for storage and transmission is the JPEG format [23] created by the Joint Photographic Experts Group committee. It stores image data as quantised frequency coefficients. The JPEG compressor works by first splitting the uncompressed image up into blocks of 8×8 pixels. The discrete cosine transformation (DCT) then transfers all of the 8×8 blocks into blocks of 8×8 frequency coefficients [24]. The next step is quantization. This is the lossy step in the process. This step works by dividing each component in the frequency domain by a constant for that component, then the results are rounded to the nearest whole number. This step can be controlled by what is known as the Q-factor. A higher Q-factor will produce a smaller file size but the image will be of a worse quality. A Q-factor of zero will skip this step making the resulting file have lossless compression [25]. The final step is to use Huffmans coding (a method of lossless data compression) to compress the results of the last step. (This is a very simplified explanation for more detail read [25]).

The Jsteg algorithm is a steganography algorithm that was developed by Derek Upham. It was the first publicly available solution to using steganography in JPEG images [26]. This algorithm uses LSB steganography to hide information in the JPEG file format. As stated above if LSB steganography is used in normal image space (pixel space) also known as the spatial domain, then lossy compression will destroy any message that has been embedded. Because of this Jsteg takes advantage fo the way the file format works and embeds the message in the LSBs of the frequency coefficients that you get after the quantisation step of the JPEG compression. This allows Jsteg to avoid the lossy step of the JPEG compression algorithm, meaning that the LSB embedded data is safe. The Jsteg algorithm will not replace the LSBs of the values 1 or 0. The resulting image is, by the human eye indistinguishable from the original JPEG image. Because of the cover images indistinguishably from a normal JPEG and its use of the most common image file format it is able to hide in plain sight as is the aim of steganography. Jsteg is by some forms of steganalysis for example the statistical attack described in this paper [27].

Watermarking

Watermarking comes in two types viable and invisible [5], invisible watermarking is the same as steganography in any respects for this reason it is treated as a subset of steganography [7].

Steganography and watermarks differ in a few ways. Watermarking is designed (in most cases) to be robust. This robustness applies to both visible and invisible watermarking. Fragile watermarking is not used to validate the owner or creator of the medium but instead to detect if the medium has been tampered with, for more information read [5], [28], [29]. Robust watermarking is completely opposite the watermark should be incredibly difficult or impossible to remove or damage. A robust watermark should resist file processing such as cropping, scaling, rotation, translation and compression [5]. Watermarking does not always try to hide the fact that it has hidden information [5] e.g. a visible watermark. The purpose of a watermark is also different. Watermarks are applied to specific covers in order to provide copyright protection. Steganography on the other hand chooses its cover arbitrarily.

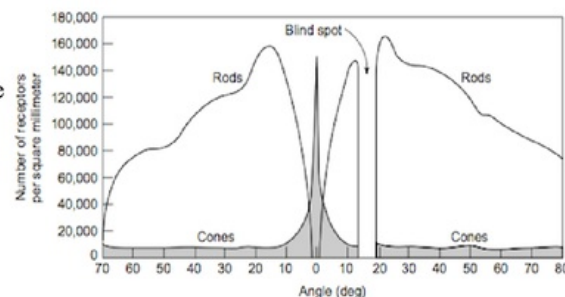
The Human Eye

Eyes can see light thanks to photosensitive cells called rods and cones. These cells are located at the back of the eye on the retina. Rods are more numerous in the eye as shown in Fig 1, with there being around 120 million rod cells compared to around 7 million cone cells [30]. Rods are designed to give vision in low light levels (e.g. 1cd/m² or less). They also come in only one type. Rods respond best to light waves with a wavelength of around 520nm. At light levels greater than 100cd the rod cells become saturated and only the cones function. Cones are less numerous than rods and are focused mainly in the fovea, as shown in Fig

1. Unlike rods cones come in three different kinds, these kinds are called L, M and S. They are named this way because L cones are more sensitive to longer wavelengths, M cones to the middle wavelengths and S to the shorter wavelengths. The types of cones are also

sometimes referred to as R, G and B cones, this is *Fig 1: Rod/Code density graph*

because of the spectral responsiveness of the three types of cone cells. The L cone cells responds best to light wavelengths around 560nm. The M cone cell responds best to light wavelengths around



530nm. The S cone cell responds best to light wavelengths of around 420nm. Although as shown in Fig 2 the responsiveness of the cone types overlap. It is because of the three types of cone cells that we can see in colour, as there is only one type of rod cell, rod system is not capable of coloured vision. The L, M and S cones are not equal in the eye in both number and distribution. The ratio of L:M:S cones is approximately 12:6:1 with reasonable estimates as high as 40:20:1 [30]. Due to this the S cones are sparsely distributed throughout the retina and completely absent from the most central part of the fovea. This shows that L cones or the red cones are most abundant allowing our eyes to perceive the colour red the best, S cones or B cones are by far the least abundant and absent from the centre of our eyes making the eye much poorer at perceiving the colour blue, and the M cones or G cones fall somewhere in the middle.

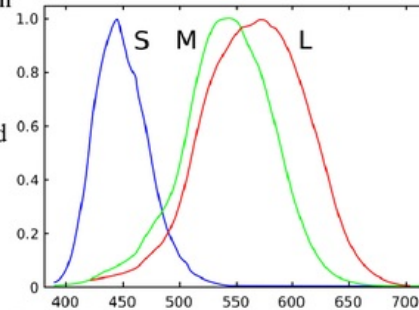


Fig 2: Normalized responsivity spectra of human cone cells

Steganography and Games

Steganography can be used in many ways, the only limit to its use is the imagination of its user. An example of its use in a video games is in the game Monaco: What's Yours Is Mine developed by Pocketwatch Games they used LSB steganography to embed all the information about a level of the game in a side-view of the level itself [31]. This makes passing around levels that you had created a lot easier and more interesting as the image of the level was the level. This was not the first use of steganography in games however, the game Spore that was developed by maxis and released on September the 4th 2008 is the first example of a major video game using steganography. In spore you can evolve a creature from a single celled organism all the way into interstellar travelling civilisations of complex organisms. Each creature you create is has an image of it saved as a .png image and it is in this image that steganography is used, all of the information for the creature is embedded within the image. This allows them to share information about your creatures and worlds with every other player of them game without passing around needless files.

Current Programs

There are many steganography programs available which give a wide variety of ways to hide different types of information. StegoShare is 3:2:3 (RGB) LSB steganography program that will allow you to hide any file within a lossless .png image. It provides about 40% hiding capacity and can hide files up to 2GB in size and is undetectable to the human eye [32]. QuickStego is a basic steganography program that allows you to hide text within a .bmp image using LSB steganography [33]. QuickCrypto is a program made by the same people as QuickStego (Cybernecence) but is not free. It does offer more features however, it allows you to hide files of text within images or sound files using LSB steganography that is undetectable to human vision or hearing. It also allows you to hide files within folders so that they are undetectable to the normal windows applications (e.g. windows explorer). Aside from steganography as its name suggests QuickCrypto also allows your files to be encrypted and then hidden using steganography [34].

Experiment

Question: How many bits of RGB data can be manipulated before it is possible visually to see the difference between the new image and the original and does the colour that is manipulated affect how quickly the change becomes noticeable?

Null hypothesis 1: Regardless of how many bits of data are manipulated the ability to identify a change in the image will be no different.

Hypothesis 1: As the number of bits of data manipulated increases it becomes easier to visually detect the difference.

Null hypothesis 2:

The manipulation of the RGB elements will all be detected at the same stage.

Hypothesis 2:

The manipulation of R, G and B bits will be easier to notice respectively.

The software used for this experiment was developed as a part of this project and is available on git hub <https://github.com/AxlWest/Steganography>. This software allows text files to be hidden within images by changing the LSBs of 24 bit bitmap images. LibreOffice Writer was also used to create the word document questionnaire that is the experiment.

The experiment will be conducted via a questionnaire, this will be given to a random sample of people. The reason for using a questionnaire over sitting down and asking for peoples answers is due to time constraints and access to test subjects. The randomization will include people with colour blindness and those with glasses and other eyesight conditions to increase the validity of the experiment. To add further validity one of the questions asks you to compare to unmodified images, any agreement that this image had been manipulated will cause that questionnaire to be discarded. This increases the experiments validity by removing any questionnaires that have been filled out randomly or without accurately comparing images. To increase the reliability of the experiment the questionnaire will be given to as many people as possible. The reliability of this experiment is also increased by its repeatability as each section can be replicated or the questionnaire can be handed out again. This experiment is generalisable to an extent as any image file format that stores images as colour values will provide the same kind of results. However not all file formats store image data in the same way so it would not be possible to repeat the experiment under all conditions.

The questionnaire will ask for an age group, gender and eyesight condition (i.e. colour blindness or visual assistance) to see if there are any interesting or unforeseen links. The rest of the questions ask for a comparison of two images both at a quick look and a close inspection. The image will be in a random order so that there is no patterns for people to pick up on, there will also be a control question. The control question will also be in a random place and be a comparison of two unmodified images. The image chosen to be used in the questionnaire can be found here [35], it can also be seen with a link in the questionnaire. This image was chosen as it is a series of tiled game textures, this allows for many styles of image and also helps to show that steganography can be used in a games context without being noticed. A copy of the questionnaire is can be found in appendix 1.

Results

What follows is a table of results. The results tables are filled up with fifteen of the eighteen responses to the questionnaire. Only fifteen of the questionnaires were used because the other three reported seeing a difference between the two images that were identical, at the bottom of this section these results will be shown and discussed. The question column corresponds to an image question of the questionnaire. The next two columns correspond to the first part of the question the "At a quick look can you spot any differences between the original and comparison image?". The columns tally the answers of the question. The fourth and fifth column correspond to the next part of the question "Upon closer inspection can you spot any differences between the original and comparison image?" in the same way as the previous columns. The last column is the type of LSB steganography used, to explain for question one 444 four bits of red were used, four bits of green and 4 bits of blue.

Table of Responders

Age	No.	Sex	No.	Eyesight	No.
-25	6	Male	11	20:20	4
25-50	6	Female	4	Assisted	11
51-75	3			Colour-blind	1
75+	0			Other	0

Results table (Question order)

Question	Could tell the difference on first look	Couldn't tell the difference on first look	Could tell the difference with closer inspection	Couldn't tell the difference with closer inspection	LSB type (RGB)
1	8	7	14	1	444
2	0	15	1	14	100
3	2	13	3	12	400
4	0	15	1	14	001
5	1	14	2	13	010
6	7	8	9	6	333
7	3	12	6	9	020
8	1	14	1	14	003
9	11	4	14	1	040
10	2	13	2	13	004
11	1	14	1	14	200
12	0	15	1	14	002
13	1	14	1	14	222
14	2	13	2	13	111
15	1	14	1	14	300
16	5	10	6	9	030
17	0	15	0	15	000 (Raw image)

Results table (LSB order)

Question	Could tell the difference on first look	Couldn't tell the difference on first look	Could tell the difference with closer inspection	Couldn't tell the difference with closer inspection	LSB type (RGB)
17	0	15	0	15	000 (Raw image)
2	0	15	1	14	100
5	1	14	2	13	010
4	0	15	1	14	001
14	2	13	2	13	111
11	1	14	1	14	200
7	3	12	6	9	020
12	0	15	1	14	002
13	1	14	1	14	222
15	1	14	1	14	300
16	5	10	6	9	030
8	1	14	1	14	003
6	7	8	9	6	333
3	2	13	3	12	400
9	11	4	14	1	040
10	2	13	2	13	004
1	8	7	14	1	444

Results table (Colour order)

Question	Could tell the difference on first look	Couldn't tell the difference on first look	Could tell the difference with closer inspection	Couldn't tell the difference with closer inspection	LSB type (RGB)
17	0	15	0	15	000 (Raw image)
2	0	15	1	14	100
11	1	14	1	14	200
15	1	14	1	14	300
3	2	13	3	12	400
5	1	14	2	13	010
7	3	12	6	9	020
16	5	10	6	9	030
9	11	4	14	1	040
4	0	15	1	14	001
12	0	15	1	14	002
8	1	14	1	14	003
10	2	13	2	13	004
14	2	13	2	13	111
13	1	14	1	14	222
6	7	8	9	6	333
1	8	7	14	1	444

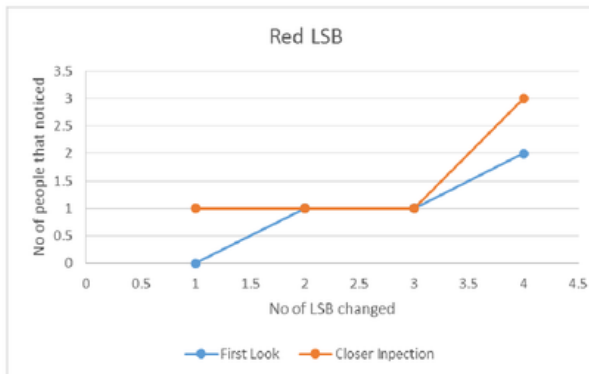


Fig 5: Red LSB Graph

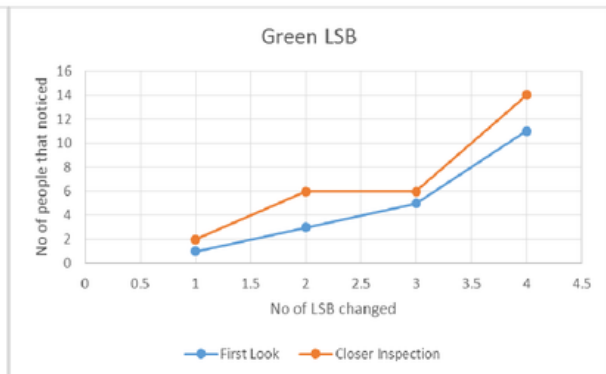


Fig 6: Green LSB Graph

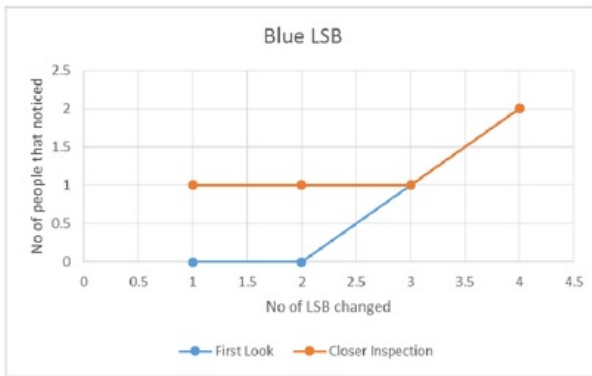


Fig 7: Blue LSB Graph

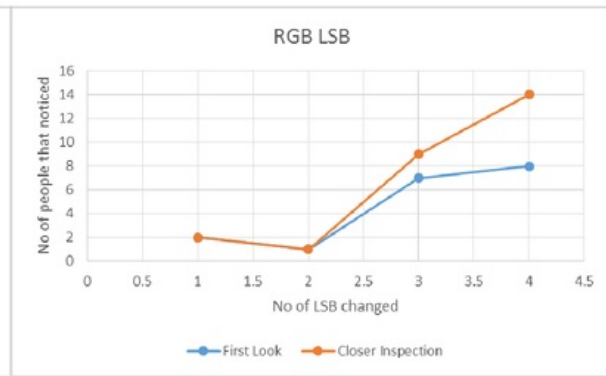


Fig 8: RGB LSB Graph

The overall results are as expected, as the number of LSBs are increased it becomes easier to identify the manipulation of the image. The results for low LSB usage are as expected. For 111, 100, 010 and 001 where only noticed in two of the results. In the results of the 222, 200, 020 and 002 LSB questions the manipulation still goes largely unnoticed. The green bit manipulation for 020 went unnoticed by the majority of people at first glance but the number of people that noticed went up to almost half upon closer inspection. This result goes against expectations and research discovered in the literature review, because the green LSB was noticed before the red. I think however that the research was not wrong but that the image had an unexpected impact. As stated above the image used was made up of multiple game textures tiled together. One of these tiles is green metal with some rust, a large portion of it is a solid green colour. Because this tile was solid green any noise introduced into the image by the steganography program become more evident. Interestingly 222 was only noticed by one person which is much less than 020 even though more bits were manipulated. I think this is because the noise introduced to the image is not concentrated on one colour but is instead spread equally between all colours making the noise less noticeable. The 3 bit manipulation questions have almost the same results as the 2 bit ones. The 333 had more people notice it than the 030 however. The results for 4 bit manipulation are again very similar. At this point for 444 over half of the people noticed the manipulation on their first look, upon closer inspection all but one person could notice the manipulation, same for the green manipulation. An interesting anomaly can be seen in fig 8. This shows that for 222 LSB steganography was noticed by less people than 111 LSB steganography. This may be because as when people are looking for something (in this case differences between two images) they can trick themselves into seeing what is not there. Each of the graphs also show a nice visual representation of the gradual increase in people noticing as the number of LSBs increase.

The results confirm hypothesis 1. This is shown by all of the bit increases in the tables above.

The second hypothesis however could not be confirmed by this experiment. This is because of the image chosen. The data did start to show that it agrees with the hypothesis with red being noticed before blue but due to the image this cannot be confirmed. These results show that the image heavily influences the ability to detect manipulation. All of the tiles except from the green metal tile are quite noisy, due to this the noise introduced by the steganography program is harder to detect.

Table of Unused Responders

Age	No.	Sex	No.	Eyesight	No.
-25	1	Male	3	20:20	1
25-50	2	Female	0	Assisted	2
51-75	0			Colour-blind	0
75+	0			Other	0

Unused responses results table (LSB order)

Question	Could tell the difference on first look	Couldn't tell the difference on first look	Could tell the difference with closer inspection	Couldn't tell the difference with closer inspection	LSB type (RGB)
17	1	2	3	0	000 (Raw image)
2	1	2	2	1	100
5	0	3	1	2	010
4	1	2	2	1	001
14	1	2	1	2	111
11	2	1	2	1	200
7	3	0	3	0	020
12	2	1	3	0	002
13	2	1	2	1	222
15	2	1	2	1	300
16	1	2	3	0	030
8	2	1	2	1	003
6	0	3	2	1	333
3	2	1	2	1	400
9	3	0	3	0	040
10	2	1	2	1	004
1	2	1	3	0	444

These results as you can see were spread over two age groups (both young) and two eyesight groups assisted vision (i.e. Glasses needed for reading etc.) and 20:20. All of the answers were from men. It is impossible to draw a conclusion from this however as three results are not enough to show any kind of trend most likely it is coincidence that all of these responders were male but if the questionnaire reached more test subjects it may be possible to draw a reliable conclusion. In the one bit questions (100, 010, 001 and 111) more of the responders could tell there had been manipulation in the red and blue colours than the green which is the opposite of the other responses. On the larger scale on closer inspection there was not a single question that upon closer inspection the responders did not notice a difference. I think that this is because when looking really closely it's easy to see differences that are not there. For the first look question only one of the questions the one bit green one was not seen at first look this may be because the responders are trying to get "correct" answers instead of answering what they can actually see.

Program Design & Implementation

Program Requirements

The program developed must be able to take a .txt file and using LSB steganography hide the text files contents within a 24 bit bitmap image. The program must be able to do LSB steganography that uses the following RGB LSBs 100, 010, 001, 111, 200, 020, 002, 222, 300, 030, 003, 333, 400, 040, 004, 444 and have an option that allows you to select what type of steganography to use. The program has to be able to do this in order to produce images that can be used in the experiment. The program should also let you dynamically select an image. It should allow you to dynamically select file to save the image to. It should also allow you to choose the name the image will save with. The program should also be able to run multiple times in the same instance. These requirements while not vital to the experiment will allow for faster and a more user friendly production of the stego images.

Design

The first decision was what coding environment to use. For this I decided to use QT Creator. The reason for this is QT Creator allows you to create a professional looking User Interface (UI) with relative speed and ease. QT also offer many more advantages and is fully cross platform. Creating a nice UI will make the program look neater while running and also increase the speed that it can be used. The main menu will have to buttons for that will allow the user to select to preform steganography or extract a message hidden via steganography, two buttons where used to keep the menu neat and to allow for quick option selection. Each button will open a sub window that allows the user to perform the selected function (screen shots of the UI can be seen in appendix 2). The extract function of this program is not in the requirements but has to be developed in order to test that the steganography function works. The program when told to preform steganography will load the raw image into the program as binary data stored in an array of unsigned chars, then it load in the text file as binary data stored in an array of unsigned chars. The data type unsigned char was chosen because it is stored as one byte. After this the program will then loop around the text array hiding each bit of the text byte iteratively into the LSB required for the selected LSB method. The program will then save the modified image array out into a new bitmap with the name specified by the user.

Implementation

The first thing to be implemented was the programs UI. QT Creator come with a design feature that allows you to create windows with ease and a professional look. It took a small amount of time to get use to the tool but as the UI does not need to do very much the implementation did not take too long. The next step was it implement the bitmap loader. The easiest way to implement a bitmap loader it to include the windows.h library that has all of the definitions for the bitmap header file and types hard coded into it. QT is as mentioned above is cross platform so to keep the program cross platform it could not use windows.h as this is only available on a windows computer. To get round this the bitmap definitions where hard coded into the bitmap loader. Although the program is only required to read and save out .bmp files an imageFile class was created and the bitmap data was saved into it this allows for access to the image data (E.g. image size, width, height...) but also allow for changing between image file types. This was done to improve the reusability of the code and would allow for leeway if the program requirements where to change. During testing it was

found that the bitmap file saver corrupted the image if the width and height were not equal. To fix this problem line padding was implemented (see fig 9).

Line padding adds

extra bytes to the end of the width to make the width a multiple of 4. The next implementation

```

98
99
100
101
102
103
104
105

```

```

int add = 4 - ((width * 3) % 4) ; //Work out bytes to add at end of line for line padding
if(add == 4)
{
    add = 0 ;
}
int size = ((width * 3) + add) * height ; //Calculate size of image with padding

```

Fig 9: Bitmap Line Padding

was for 111 LSB steganography. This was the first implementation of LSB steganography and it went smoothly using bit masking and bitwise operations to check and change the LSB of the RGB values. At the start of the proformSteganography method the text file is read and stored in an array of unsigned chars. For each unsigned char in the method does a loop that uses a bitmasks to find out if each bit of text data is a 1 or a 0. Then using an appropriate mask for the LSB to be modified it finds out if the LSB is equal to 1 or 0. If the bit data is the same there is no need to do anything. If they are different the program uses the XOR bitwise operation to flip the bit (see fig 10). The next step of the implementation was to do the rest of the LSB steganography options this includes 100, 010, 001, 111, 200, 020, 002, 222, 300, 030, 003, 333, 400, 040, 004, 444, 123, 134. The only bump in this section of the implementation was when 300 was implemented as the same loop could not be used as it uses an odd number of bits this was easily made to work by increasing the size of the loop to run for 3 unsigned chars instead of one. (Check appendix 3 for a graph of implementation over time)

```

45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66

```

```

if(textData[1] & 128) //Is the first Bit a 1
{
    if(imageData[k] & 1) //RGB LSB is also a 1
    {
        //Already a 1 do nothing
    }
    else //RGB LSB is a 0
    {
        imageData[k] ^= 1 << 0 ;
    }
}
else //Bit is a 0
{
    if(imageData[k] & 1) //RGB LSB is a 1
    {
        imageData[k] ^= 1 << 0 ;
    }
    else //RGB LSB is also a 0
    {
        //Already a 0 do nothing
    }
}

```

Fig 10: LSB

Testing

The UI was tested by using every option and button on the UI and checking its output with an expected output. No errors were found during these test however, some mislabelling was noticed and corrected. Each of the LSB steganography technique by hiding some text then using the extraction to retrieve the text all LSB implementations passed.

Evaluation

The data shows that it is possible to visually identify an image that's RGB bits have been manipulated. The data mostly agrees with the research from the literature review, but it can't conclusively prove anything for that more work will need to be done. Each of the five objectives were met during the course of this project. The overall aim of this project was also met.

Manipulating the LSBs of an image can be visually detected by human eyes, the project results also show that the colour manipulated does contribute to how quickly manipulations are noticed. The results were almost as expected with the exception of the green manipulations. These unexpected results are linked to the image used, more work will need to be done to answer what affects an image has on visually detecting LSB steganography. If I were to repeat this project again I would change two things firstly I would change the image used and secondly I would change the method of the experiment in one of two ways depending on time and funding constraints. The first way of changing it would be the cheapest and fastest. It would be to release the experiment questionnaire earlier and via more distribution channels. This will help to increase the number of responses to the questionnaire and the responders will be more varied (In this project there were no responses from the 75+ age group). This will increase the validity of the experiment. The second way requires a lot more time and money to achieve but the validity of the experiment would again increase. This method would involve interviewing each person separately and showing them the images and asking for their results. This would allow the experiment to reach all groups of people. With a questionnaire there is also the chance that people could just give random answers as they may not care how it affects the research the chances of this happening would be reduced in a one on one environment.

Conclusion

This project looked into steganography and how LSB steganography can be used to hide text within images. It also talked about the effect LSB steganography has on the image and ran an experiment to see how changing the LSB of an image affects the image visually. In the literature review we talked about the history of steganography and how steganography has changed to fit into the modern era. One of the methods of modern steganography is LSB steganography. Researching this helped with my first objective. Using the knowledge gained from the literature review a program was developed that could take a .txt file and using LSB steganography hide its contents within a 24 bit bitmap image, this solved objective two. The next step was to use the program. Sixteen stego images were created from the same raw image but using different LSB ratios. These images were filled with text information so that every available bit was used. Using these images a questionnaire based experiment was developed, this happened in the experiment section. The questionnaire was then handed out to as wide an audience as could be reached with the time that was available. In total eighteen questionnaires were answered and sent back. As explained in the results subsection of the experiment section only fifteen of these were used. The results mostly agree with the research from the literature review however against the research the green manipulation was noticed first. I suspect that the image is responsible for this however to confirm this more research will be needed. The results of the experiment show that very few people can tell an image has been manipulated when only one LSB of the RGB data has been used (only 2 out of 15 said that they noticed a difference) the numbers of people noticing the manipulation starts to increase, this is especially true for the green LSB steganography and for the even ratio LSB steganography (e.g. 222 LSB steganography). The design and implementation section details the requirements for the program both necessary and luxury. It then goes through each step in the design of the program and any specific code related decisions. The implementation subsection explains how these designs were implemented into code and talks about code decisions that were made and the reasons behind them. The testing subsection talks about the program being tested and the labelling issue noticed while doing so.

Future Work

Steganography is a huge subject and this project has only looked at a small section of it LSB steganography. Even within LSB steganography this project has only looked at the effect of changing the LSBs in images and only in the 24 bit bitmap file type. This opens many more areas of research. Each image file type stores its data differently and some file types have lossy compression research can be done on how to use LSB steganography in those types of files and does it affect the image in the same way, if the file types don't store data in the spatial domain (RGB) but instead in the frequency domain like the JPEG file type does that effect the ability to notice LSB steganography. Is this type of steganography noticed earlier or can you hide in more bits before it is noticed. LSB steganography is not limited to just images and the ear like the eye does not do a perfect job and can miss subtle changes. Work should be done here to expand upon the work done in this project as it is asking the same question. How many bits of sound data can be manipulated before it is possible to hear the difference between the new sound and the original? This will also require work into how LSB steganography can be used in audio files. One of the problems with this project is not knowing that the image chosen would have the effect that it did on noticing the effects of LSB steganography. This creates an interesting area of research, this project looked at how LSB steganography effected an image another area of research is how the image effects LSB steganography. The program was built for expiation this could give rise to a development based project that would expand the program to work with more image types, different file types and different methods of steganography.

Reflection

This is the second time I am writing this section as I made an error that lost me a days' worth of work. I mention this not for sympathy but because it taught me a very valuable lesson. Backing up your work and keeping it all together in a single document is incredibly important. Mistakes can be made by both you and a computer that can compromise your work and it is up to you to make sure that this if it does happen does not cost you too dearly. Modern computer will let you get away with a lot if you accidentally delete documents they can normally be recovered or if you are using Microsoft Word or LibreOffice they will keep regular backups for you, this however is I discovered does not mean you should underestimate the value of making backups yourself because these auto-backup methods can fail. This project has also taught me a lot about time management and the importance of Gantt charts to set targets, deadlines and estimates for when work should be completed doing this has helped keep this project up to date and running smoothly. My understanding of bit masking and bitwise operations has improved. Before this project although I knew about bitwise operations I did not understand their value thanks to this project I now know how bitwise operations can be used and way they can improve upon using standard flags or bools. This new knowledge will improve my ability to code and improve my programs as a whole. The programs UI was made in QT Designer this was a new tool to me and required quite the learning curve. After a brief period of time however the UI was completed and it looks quite professional as intended. This has program has helped me master a new set of tools and made my programs neater and more professional at the same time.

Figures

Fig 1: <http://www.uxmatters.com/mt/archives/2010/07/updating-our-understanding-of-perception-and-cognition-part-i.php>

Fig 2: https://upload.wikimedia.org/wikipedia/commons/thumb/1/1e/Cones_SMJ2_E.svg/1280px-Cones_SMJ2_E.svg.png

Fig 3: <http://blog.blockos.org/?p=696>

Fig 4-10: This image was created by Axl West for the purpose of this project

References

- [1] J. Umeh, *The World Beyond Digital Rights Management*, 1st ed. BCS, 2008.
- [2] T. Jamil, "Steganography: the art of hiding information in plain sight," *IEEE Potentials*, vol. 18, no. 1, pp. 10–12, 1999.
- [3] J. Judge, "Steganography: Past, Present, Future," *SANS Inst. Publ.*, 2001.
- [4] N. F. Johnson and S. Jajodia, "Exploring steganography: Seeing the unseen," *Computer (Long. Beach. Calif.)*, vol. 31, no. 2, pp. 26–34, Feb. 1998.
- [5] G. Kipper, *Investigator's guide to steganography*. Auerbach, 2003.
- [6] M. Weiss, "Principles of Steganography." [Online]. Available: <http://www.math.ucsd.edu/~crypto/Projects/MaxWeiss/steganography.pdf>.
- [7] S. Johnson, Neil F. Jajodia, "Steganalysis of Images Created Using Current Steganography Software." [Online]. Available: <http://www.cs.arizona.edu/~collberg/Teaching/620/1999/Handouts/ashwin2.pdf>.
- [8] N. F. Johnson and S. Jajodia, "Steganalysis: the investigation of hidden information," in *1998 IEEE Information Technology Conference, Information Environment for the Future (Cat. No. 98EX228)*, 1998, pp. 113–116.
- [9] "Statistical Steganalysis." [Online]. Available: <http://www.computing.surrey.ac.uk/personal/st/H.Schaathun/projects/Past/leivaditis.pdf>. [Accessed: 24-Nov-2015].
- [10] P. Richer, "Steganalysis: Detecting hidden information with computer forensic analysis," *SANS Inst. Publ.*, 2003.

- [11] S. Bhattacharyya and G. Sanyal, "Steganalysis of LSB Image Steganography using Multiple Regression and Auto Regressive (AR) Model," *Int. J. Comput. Technol. Appl.*, vol. 2, no. 4, pp. 1069–1077, 2011.
- [12] S. K. Jena and G. V. V. Krishna, "Blind Steganalysis: Estimation of Hidden Message Length," *Int. J. Comput. Commun. Control*, vol. 2, no. 2, pp. 149–158, 2007.
- [13] T. Pevny, J. Fridrich, and A. D. Ker, "From Blind to Quantitative Steganalysis," *IEEE Trans. Inf. Forensics Secur.*, vol. 7, no. 2, pp. 445–454, Apr. 2012.
- [14] S. Tan and B. Li, "Targeted steganalysis of adaptive pixel-value differencing steganography," in *2012 19th IEEE International Conference on Image Processing*, 2012, pp. 1129–1132.
- [15] M. Sharma and S. Bera, "A REVIEW ON BLIND STILL IMAGE STEGANALYSIS TECHNIQUES USING FEATURES EXTRACTION AND PATTERN CLASSIFICATION METHOD," *Int. J. Comput. Sci. Eng. Inf. Technol.*, vol. 2, no. 3, p. 2012, 2012.
- [16] S. Tan and B. Li, "Targeted Steganalysis of Edge Adaptive Image Steganography Based on LSB Matching Revisited Using B-Spline Fitting," *IEEE Signal Process. Lett.*, vol. 19, no. 6, pp. 336–339, Jun. 2012.
- [17] T. Zhang and X. Ping, "A new approach to reliable detection of LSB steganography in natural images," *Signal Processing*, vol. 83, no. 10, pp. 2085–2093, Oct. 2003.
- [18] Tao Zhang and Xijian Ping, "Reliable detection of LSB steganography based on the difference image histogram," in *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03).*, 2003, vol. 3, pp. III–545–8.
- [19] M. Juneja and P. S. Sandhu, "An Improved LSB Based Steganography Technique for RGB Color Images," *Int. J. Comput. Commun. Eng.*, vol. 2, no. 4, 2013.
- [20] D. Rawat and V. Bhandari, "A Steganography Technique for Hiding Image in an Image using LSB Method for 24 Bit Color Image," *Int. J. Comput. Appl.*, vol. 64, no. 20, pp. 15–19.
- [21] A. Rutherford, "Steganography in Lossy and Lossless Images," 2014.
- [22] J. J. Eggers, R. Bauml, and B. Girod, "A Communications Approach to Image Steganography," in *Proceedings of SPIE Vol. 4675 Security and Watermarking of Multimedia Contents IV*, 2002.
- [23] Linjie Guo, Jiangqun Ni, and Yun Qing Shi, "Uniform Embedding for Efficient JPEG Steganography," *IEEE Trans. Inf. Forensics Secur.*, vol. 9, no. 5, pp. 814–825, May 2014.
- [24] A. Westfeld, "F5—A Steganographic Algorithm," in *Information Hiding: 4th International Workshop*, 2001.
- [25] A. C. Hung, "PVRG-JPEG CODEC 1.1," Stanford, 1993.
- [26] H. G. Schaathun, "Steganography in JPEG," *University of Surrey*, 2007. [Online]. Available: <http://www.computing.surrey.ac.uk/teaching/2006-07/csm25/Chapter6/jsteg-h.pdf>.

[Accessed: 27-Nov-2015].

- [27] A. Westfeld and A. Pfitzmann, "Attacks on Steganographic Systems," in *Information Hiding. Third International Workshop*, 2000, pp. 61–76.
- [28] P. MeenakshiDevi, M. Venkatesan, and K. Duraiswamy, "A Fragile Watermarking Scheme for Image Authentication with Tamper Localization Using Integer Wavelet Transform," *J. Comput. Sci.*, vol. 5, no. 11, pp. 831–837, 2009.
- [29] H. Lu, R. Shen, and F.-L. Chung, "Fragile watermarking scheme for image authentication," *Electron. Lett.*, vol. 39, no. 12, p. 898, 2003.
- [30] M. D. Fairchild, *Color Appearance Models*, 2nd ed. John Wiley & Sons, 2005.
- [31] "Steganography - The Secret Data behind the Level Images," 2010. [Online]. Available: https://www.facebook.com/note.php?note_id=437497056995. [Accessed: 22-Jan-2016].
- [32] "How it works." [Online]. Available: <http://stegoshare.sourceforge.net/howitworks.html>. [Accessed: 16-Feb-2016].
- [33] "Free Steganography Software - QuickStego." [Online]. Available: <http://www.quickcrypto.com/free-steganography-software.html>. [Accessed: 16-Feb-2016].
- [34] "Steganography Software." [Online]. Available: <http://www.quickcrypto.com/steganography-software.html>. [Accessed: 16-Feb-2016].
- [35] C. Sikkenga, "6 Free Texture Packs," 2016. [Online]. Available: <https://episcura.com/blog/2016/03/14/a-list-of-our-free-texture-packs-at-episcura/>. [Accessed: 06-Apr-2016].

Appendix 1: Questionnaire

Steganography is the practice of concealing a file, message, image, or video within another file, message, image, or video.

Thank you for taking part in my final year project by completing the following questionnaire. The following questions will help determine if the human eye can distinguish between small changes in colour. There are no right or wrong answers so please fill out honestly and without collaboration, please note the original and comparison image may also be identical.

Name: (Optional)		
Age (Optional)	Under 25	Please indicate your age range
	25 to 50	
	51 to 75	
	Over 75	
Gender (Optional)	Male	Please indicate your gender
	Female	
Eyesight	20:20 Vision	Please indicate any that apply
	Assisted Vision (i.e. Glasses needed for reading etc)	
	Colour blind	
	Other	
	(Please detail)	

Image Comparison 1

Following questions can simply be answered with Yes or No opinion.	Yes	No
At a quick look can you spot any differences between the original and comparison image?		
Upon closer inspection can you spot any differences between the original and comparison image? (For closer inspection you may any zoom features the application you are using to view the image has)		

Original Image



Comparison Image



Image Comparison 2

Following questions can simply be answered with Yes or No opinion.	Yes	No
At a quick look can you spot any differences between the original and comparison image?		
Upon closer inspection can you spot any differences between the original and comparison image? (For closer inspection you may any zoom features the application you are using to view the image has)		

Original Image



Comparison Image



Image Comparison 3

Following questions can simply be answered with Yes or No opinion.	Yes	No
At a quick look can you spot any differences between the original and comparison image?		
Upon closer inspection can you spot any differences between the original and comparison image? (For closer inspection you may any zoom features the application you are using to view the image has)		

Original Image



Comparison Image



Image Comparison 4

Following questions can simply be answered with Yes or No opinion.	Yes	No
At a quick look can you spot any differences between the original and comparison image?		
Upon closer inspection can you spot any differences between the original and comparison image? (For closer inspection you may any zoom features the application you are using to view the image has)		

Original Image

Comparison Image



Image Comparison 5

Following questions can simply be answered with Yes or No opinion.	Yes	No
At a quick look can you spot any differences between the original and comparison image?		
Upon closer inspection can you spot any differences between the original and comparison image? (For closer inspection you may any zoom features the application you are using to view the image has)		

Original Image



Comparison Image



Image Comparison 6

Following questions can simply be answered with Yes or No opinion.	Yes	No
At a quick look can you spot any differences between the original and comparison image?		
Upon closer inspection can you spot any differences between the original and comparison image? (For closer inspection you may any zoom features the application you are using to view the image has)		

Original Image



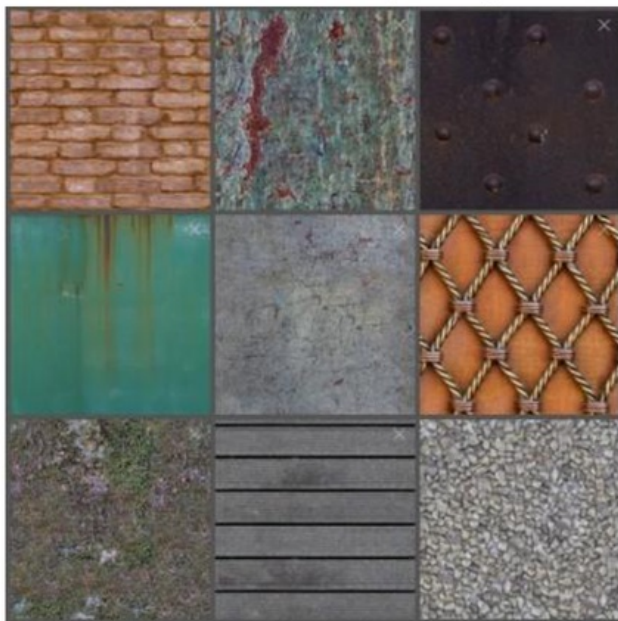
Comparison Image



Image Comparison 7

Following questions can simply be answered with Yes or No opinion.	Yes	No
At a quick look can you spot any differences between the original and comparison image?		
Upon closer inspection can you spot any differences between the original and comparison image? (For closer inspection you may any zoom features the application you are using to view the image has)		

Original Image



Comparison Image



Image Comparison 8

Following questions can simply be answered with Yes or No opinion.	Yes	No
At a quick look can you spot any differences between the original and comparison image?		
Upon closer inspection can you spot any differences between the original and comparison image? (For closer inspection you may any zoom features the application you are using to view the image has)		

Original Image



Comparison Image



Image Comparison 9

Following questions can simply be answered with Yes or No opinion.	Yes	No
At a quick look can you spot any differences between the original and comparison image?		
Upon closer inspection can you spot any differences between the original and comparison image? (For closer inspection you may any zoom features the application you are using to view the image has)		

Original Image

Comparison Image



Image Comparison 10

Following questions can simply be answered with Yes or No opinion.	Yes	No
At a quick look can you spot any differences between the original and comparison image?		
Upon closer inspection can you spot any differences between the original and comparison image? (For closer inspection you may any zoom features the application you are using to view the image has)		

Original Image



Comparison Image



Image Comparison 11

Following questions can simply be answered with Yes or No opinion.	Yes	No
At a quick look can you spot any differences between the original and comparison image?		
Upon closer inspection can you spot any differences between the original and comparison image? (For closer inspection you may any zoom features the application you are using to view the image has)		

Original Image



Comparison Image



Image Comparison 12

Following questions can simply be answered with Yes or No opinion.	Yes	No
At a quick look can you spot any differences between the original and comparison image?		
Upon closer inspection can you spot any differences between the original and comparison image? (For closer inspection you may any zoom features the application you are using to view the image has)		

Original Image



Comparison Image

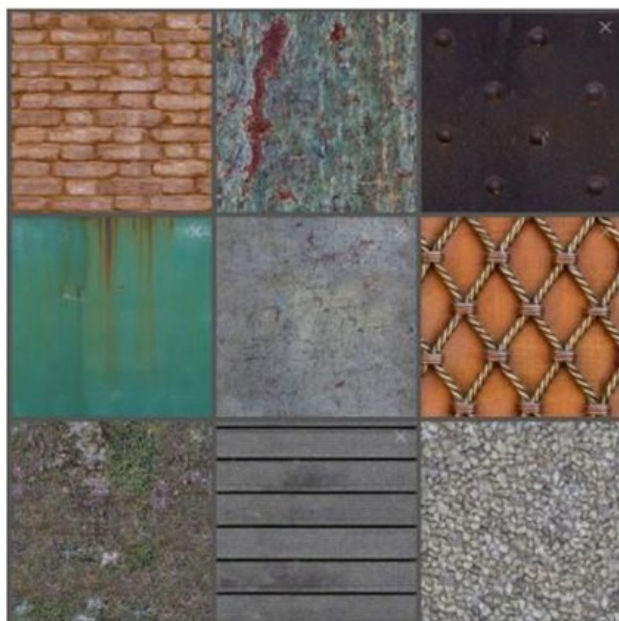


Image Comparison 13

Following questions can simply be answered with Yes or No opinion.	Yes	No
At a quick look can you spot any differences between the original and comparison image?		
Upon closer inspection can you spot any differences between the original and comparison image? (For closer inspection you may any zoom features the application you are using to view the image has)		

Original Image



Comparison Image



Image Comparison 14

Following questions can simply be answered with Yes or No opinion.	Yes	No
At a quick look can you spot any differences between the original and comparison image?		
Upon closer inspection can you spot any differences between the original and comparison image? (For closer inspection you may any zoom features the application you are using to view the image has)		

Original Image



Comparison Image



Image Comparison 15

Following questions can simply be answered with Yes or No opinion.	Yes	No
At a quick look can you spot any differences between the original and comparison image?		
Upon closer inspection can you spot any differences between the original and comparison image? (For closer inspection you may any zoom features the application you are using to view the image has)		

Original Image



Comparison Image



Image Comparison 16

Following questions can simply be answered with Yes or No opinion.	Yes	No
At a quick look can you spot any differences between the original and comparison image?		
Upon closer inspection can you spot any differences between the original and comparison image? (For closer inspection you may any zoom features the application you are using to view the image has)		

Original Image



Comparison Image



Image Comparison 17

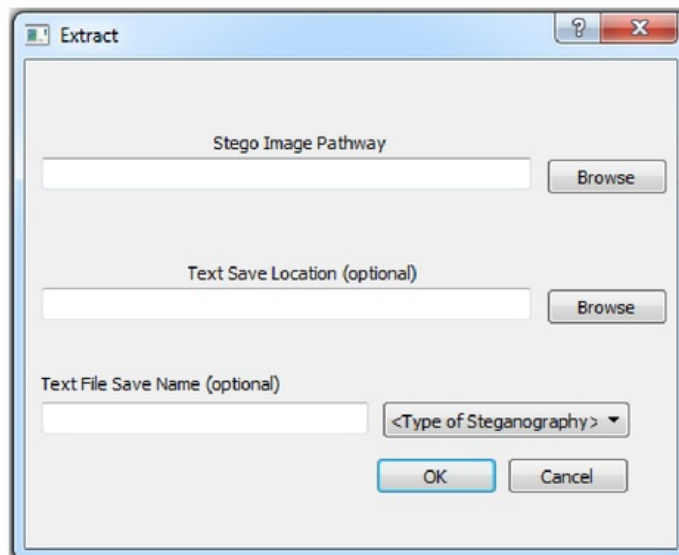
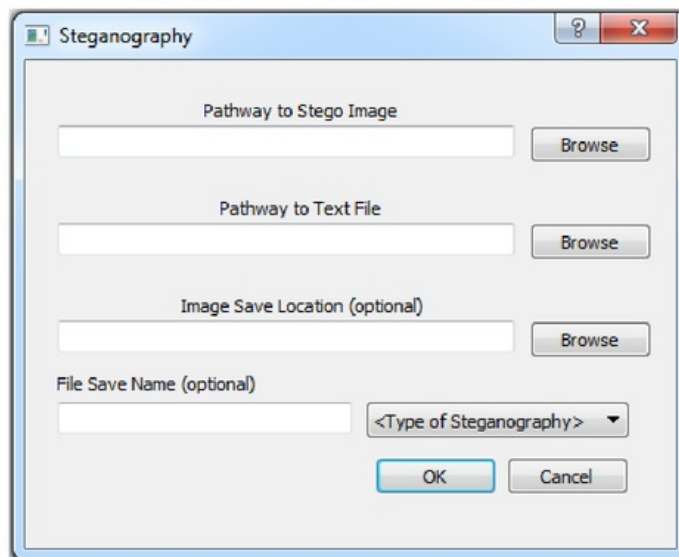
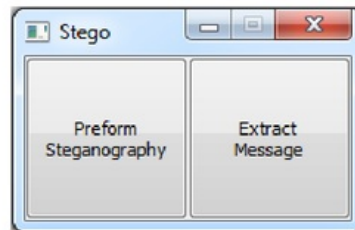
Following questions can simply be answered with Yes or No opinion.	Yes	No
At a quick look can you spot any differences between the original and comparison image?		
Upon closer inspection can you spot any differences between the original and comparison image? (For closer inspection you may any zoom features the application you are using to view the image has)		

Original Image

Comparison Image



Appendix 2: UI Screen Shots



Appendix 3: Implementation over Time

Oct 25, 2015 – Apr 20, 2016

Contributions: **Commits** ▾

Contributions to master, excluding merge commits

