Steganography: Concealing Data Within an Image.

David A. Sevilla Cazes, LicEng

December 2015

Department of Electrical and Computer Engineering
Northeastern University

Abstract

Throughout history, humans have look for ways to protect communications by utilizing different techniques to assure that some messages can only be read by the intended receiver. Steganography is used when the objective is to conceal information within a medium without revealing any sign of the hidden information. This project explores two of the most common steganography methods in imaging: LSB in the spatial domain and Jsteg in discrete cosine transform domain. Throughout this project a steganography graphical user interface was implemented in MATLAB. The resulting images from the steganography processes implemented were analyes to fined any traces of the concealed information.

Contents

1 Introduction

The oxford dictionary defines Steganography as: “The practice of concealing messages or information within other non-secret text or data”. The etymology of steganography comes from the greek *steganos*, “cover or roof” and *grafia*, ‘writing’ defining it as covered writing.

Throughout history, humans have look for ways to protect communications by utilizing different techniques to assure that some messages can only be read by the intended receiver.

When talking about stenography we cannot stop talking about cryptography. Since both techniques are used to conceal information. The main difference between this two concealing approaches is that when looking at a cryptographic messages we can immediately tell that there is hidden information.

On the other hand, steganography does not show any signs of hidden information [1]. Therefore no suspicion of hidden information will motivate any attacks. Image steganography consists of embedding data into a cover image in such a way that the data can only be access following a specific decryption process. The image steganography objective is to create a new image that looks identical to the cover image without leaving any trace of the concealed information.

Steganography can be classified into fragile and robust. Steganography is said to be fragile if the concealed information will be lost after the stego-image is modified. It is said that steganography is robust if the concealed information is not destroyed after the stego-image is modified as is the case of storing that image in JPEG format.

According to the method in which steganography is achieve it can also be classified in spatial domain and transfrom domain. Spatial domain method tend to be more fragile and transform Domain method tend to be more robust.

There are many application of steganography and different techniques are used according to the intended goal of the hidden message. Some applications mention in [[1]] are: Copyright Control, which is used to concealing secret copyright information; Covert Communication, that uses non-standard method to transmit communication with the goal of hiding the fact that the communication is happening; Smart ID’s, where confidential information is concealed into people ID image; Printers, in which information is hidden inside very small ink dots.

The objective of this project is to research different steganography techniques in the Spatial Domain and in the Transform Domain, compare the performance of different techniques by analyzing resulting stego-image and create a graphical user interface that allows the user to hide data within an image and retrieve the conceal data from *stego-images*.

For this project the data used in the implementation is limited to text or images.

The structure of this report is as following: Section 2: Explores spatial domain steganography techniques, focusing in Less Significant Bit Substitution technique. Section 3: Explores Transform Domain techniques, focusing in the DCT - JSTEG. Section 4: Describes the GUI implementation and section 5: Analysis and comparison of the two techniques that were implemented is explained. Section 6: concludes and summarizes the results that were obtain through the development of this project.

2 Spatial Domain Steganography

Spatial Domain Steganography consists of concealing data in the image domain of the cover image. Spatial domain stenography method tend to be fragile and easier to implement. The most popular technique that is used in spacial domain is the Least Significant Bit (LSB) substitution technique.

2.1 Less Significant Bit Substitution

The Less significant bit (LSB) substitution method comes from the fact that an image visual information is mostly contained in the More Significant Bits (MSB) and that the LSBs contain more subtle details. So if we change elements in the LSBs of each pixels the changes will not be visible in the modified image.

Figure 1 shows an original 8-bit fractal image and Figure 2 shows the same 8 bit fractal image divided into bit planes where the top left plane has the Figure 1 MSB (Bit 7) and the lower right image has Figure 1 LSB (Bit 0) [4].

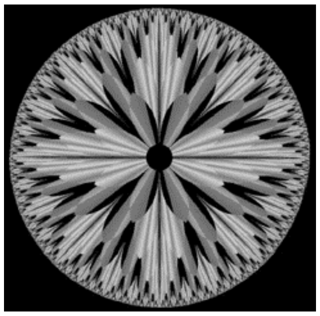


Figure 1: 8-bit Fractal Image [[4]]

When using the LSB substitution, the sequence in which the hidden message will be embedded into the cover image has to be set and known by the embedding and extracting parties. This includes the order in which the secret message is embedded into each pixel and what bits of the original image pixels will have hidden information.

Since the extracting party knows the order in which the message was embedded into the LSB there is no need for a secret key or the original image to extract the hidden information form the stego-image.

This method has high storage capacity. If the cover image is a 250×250 gray scale picture and the secret data will only be stored in the less significant bit of each pixel (Bit 0), then would be able to store a maximum of 62500 bits. Since most of the messages that we would like to hide uses bytes as its unit (e.g. Ascii text or 8-bit level images) we have to use full bytes to store the hidden message. For this reason in a 250×250 pixel image we would be able to up to store 7812 bytes that can be translated to 7812 characters string or an 88 x 88 image with 256 gray levels.

The amount of complexity in the hidden data will grow, if we use color images and treat each RGB component as a grayscale image multiplying by three the amount of data that can be concealed within a cover image.

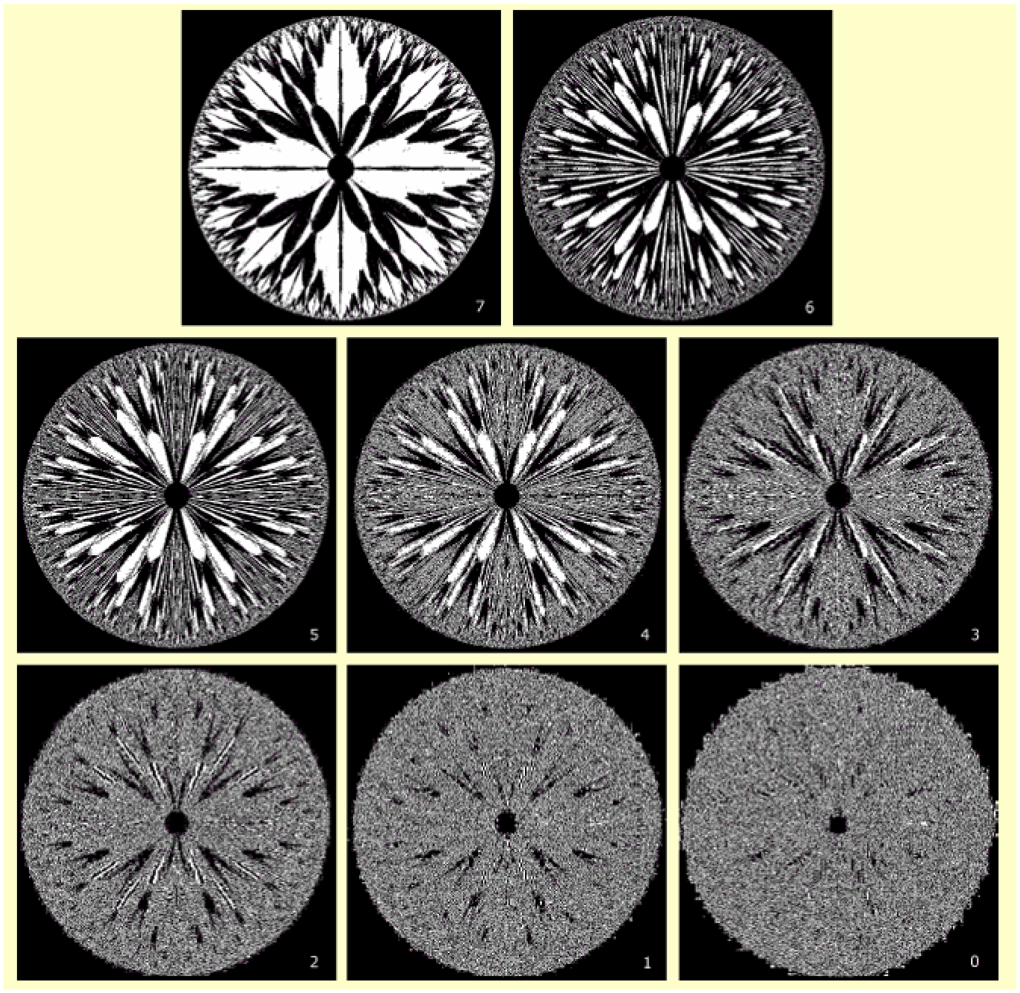


Figure 2: Bit Planes [[4]]

2.2 Algorithm

2.2.1 Embedding Algorithm

Input: Cover image, secret message

a: The message data is measured and reshaped into a single row
b: The cover image is measured.
c: Determine if the message will fit in the cover image.
d: Concatenate an arbitrary (0) end of message byte to the message.
c: Convert the message into a bit stream
e: Convert cover image pixels to binary byte matrix
f: Substitute LSBs of the cover image with the secret message
g: Convert the resulting matrix to decimal and reshape to the cover image size

Output: Stego-image

2.2.2 Extracting Algorithm

Input: Stego-image

a: Convert stego-image pixels to binary byte matrix
b: Extract LSB stream
c: Reshape bit stream as a binary byte matrix
d: Convert the resulting matrix to decimal
c: The secret message is contain in the resulting decimal string up to the end of message byte.

Output: Secret message

2.2.3 Special considerations

In the implementation of this algorithm, the end of message was arbitrarily represened with a value of zero (ascii null).

If the message that is going to be embedded is a 256 level image it is recommended to check if the lowest pixel level is zero. If the lower pixel level is zero it is recommended to manipulate the original image message so that the zero will be only designated for the end of message deliminator.

It is also important to take note on the shape of the hidden image for an easy reconstruction of the secret message.

For simplicity the implementation of this method is limited to monochromatic cover images. The messages to be embedded should be restricted to text or to monochromatic square images.

3 Transform Domain Steganogrpahy

Transform Domain steganography consists on concealing data in the frequency domain. Because of the popularity and advantages of JPEG compression standard, most of the transform domain steganography is implemented in the discrete cosine transform (DCT) domain.

3.1 JPEG Overview

JPEG standard was developed by the 'Joint Photographic Experts Group'. This standard is based in lossy compression, meaning that through the compression algorithm that forms a JPEG file, redundant data is evaluated and eliminated. The the information that was eliminated is lost with the benefit of reducing the storage requirement of the image. An image that was compressed using the JPEG format can be reconstructed with a good quality.

3.1.1 JPEG Compression Algorithm

Input: Original Image
 a: The original image is divided in 8 x 8 blocks
 b: The 2D DCT is applied to each block independently to get the cosine coefficients
 c: Quantize data by dividing each cosine coefficient by the quantization matrix.
 e: Encode the data of each block indicating where the zeros are located.
Output: Compressed Image

3.2 Jsteg

The Jsteg steganography method applies the LSB substitution method in JPEG compression process. The Jsteg method implementation is more complex and is more robust than the simple LSB substitution process.

Because of the way Jsteg images are built we are able to compress the stegoimage into a JPEG format giving all the advantage of the JPEG format. On the other hand, since the Jsteg algorithm is designed to be compressed the length of the hidden message is fully dependent on the properties of the original image and the compression quality of the JPEG file.

3.3 Algorithm

3.3.1 Embedding Algorithm

Input: Original Image, secret message
 a: The original image is divided in 8 x 8 blocks
 b: The 2D DCT is applied to each block independently to get the cosine coefficients
 c: Quantize data by dividing each cosine coefficient by the quantization matrix.
 d: Determine if the message will fit in the DCT coefficients with value different than -1, 0 and 1
 f: Convert message into a bit stream
 g: Selected DCT coefficients are converted into a binary

h: Message stream is inserted into selected DCT coefficients and stream is converted to decimal

i: Data is dequantified

j: Inverse Discrete Cosine Transform is Applied and data is reshaped into image

k: Key is generated containing the DCT selected coefficient index and compression quality.

Output: Stego-Image, Key

3.3.2 Extracting Algorithm

Input: Stego-Image, Key

a: The original image is divided in 8 x 8 blocks

b: The 2D DCT is applied to each block independently to get the cosine coefficients

c: Quantize data by dividing each cosine coefficient by the quantization matrix.

f: Convert coefficients to binary

g: Use key to retrieve the LSB stream from selected coefficients.

h: Reshape bit stream as a binary byte matrix and convert the resulting matrix to decimal

output: Secret Message

3.3.3 Special Considerations

In the implementation of this algorithm, the end message was arbitrarily represende with a value of zero (ascii null).

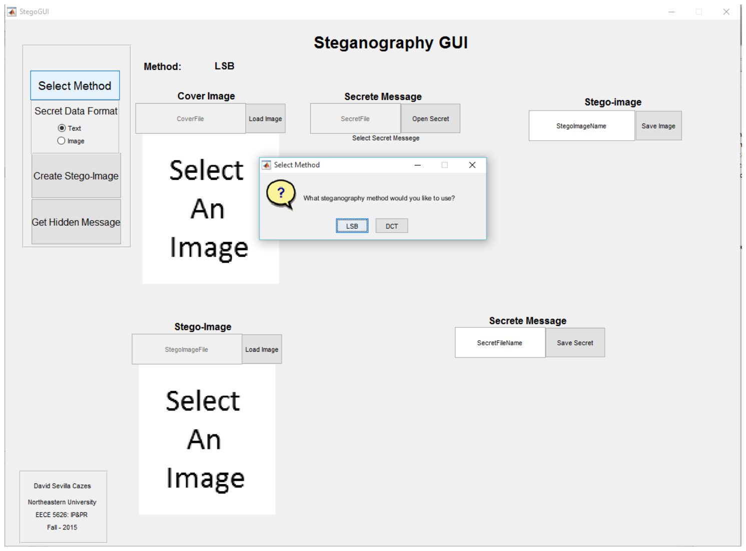
Because of the storage capacity of this method, the messages that are embedded using this method should be limited to text files.

Since the objective of this project is oriented to the analysis of the stego-image. The implementation of the embedding and extracting algorithm require a key because after embedding the image using the Jsteg method the resulting image is not stored in JPEG format, but instead a 2D IDCT is applied. If the stego-image was sotred as a JPEG file after step **i.** of the embedding algorithm there would be no need of having a key to extract the message.

4 GUI Implementation

A graphic user interface (GUI) using MATLAB was developed to study LSB and Jsteg stenganography methods discussed previously. The implemented GUI is shown in Figure 3. In this GUI the user select between LSB method and the Jsteg method is refereed in the GUI as DCT.

To simplify the GUI, with the notion that color images are formed by combining three monochromatic images, this GUI was designed to work with gray scale images. i.e. if a color cover image is chosen the GUI will convert it into a gray scale cover image. On the other hand this GUI only works with monochromatic secret messages.



What steganography method would you like to use?

Figure 3: GUI

4.1 Embedding Process

The LSB method allows the user to hide a square image or a text message within a cover image. Due to the limited storing capacity of the DCT (JSTEG) method, it was determine through experimentation that a cover image of size 1024 $times$768. Would only allow to hide images with size 32x32 or smaller. It was decided that limiting the DCT method to hidden text would have a more practical use.

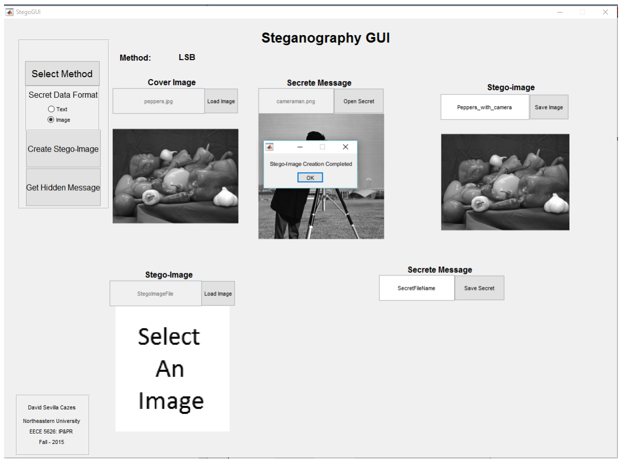
After selecting the desired steganography method and secret data format. The user can load the cover image and secret message to the interface. The user can then press the "Create Stego-Image" button to embed the secret message into the cover image to obtain the Stego-Image that can then be stored as a 'bmp' format file with termination "._LSB_.bmp" if using LSB method, or as a 'JPEG' format file with termination "_DCT_.jpg' if using DCT. When using the DCT method a key file will also be generated with the same name as the JPEG file but with termination '.key' instead of '.jpg'. This key file will be needed when extracting the hidden message.

4.2 Extracting Hidden Message

To extract the hidden message from a stego-image it is assumed that the user has knowledge of the method that was used to create the stego-image as well as the data format of the hidden message (i.e. image or text). The user should set the appropriate method and data format and load the stego-image.

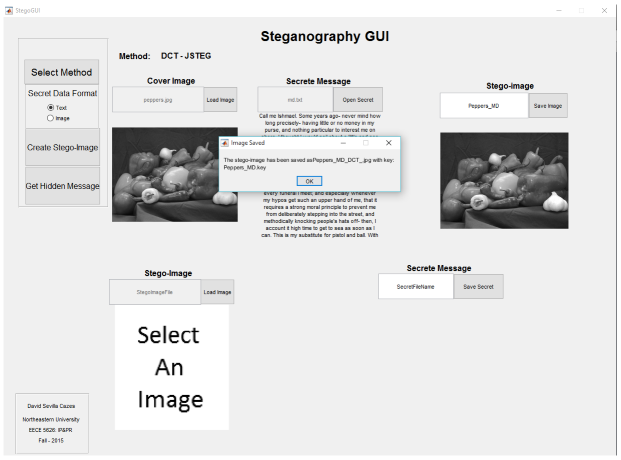
After the stego-image is loaded the user can extract the hidden message by clicking the "Get Hidden Message" button.

After extracting the hidden message, the message can be stored as a '.bmp' or '.txt' file. Depending on the data format selection



Stego-Image Creation Completed

Figure 4: Creating LSB Stego-Image



Call me Ishmael. Some years ago- never mind how long precisely- having little or no money in my purse, and nothing particular to interest me on

every funeral I meet; and especially whenever my hypos get such an upper hand of me, that it requires a strong moral principle to prevent me from deliberately stepping into the street, and methodically knocking people's hats off- then, I account it high time to get to sea as soon as I can. This is my substitute for pistol and ball. With

The stego-image has been saved asPeppers_MD_DCT_.jpg with key: Peppers_MD.key

Figure 5: Creating JSTEG Stego-Image

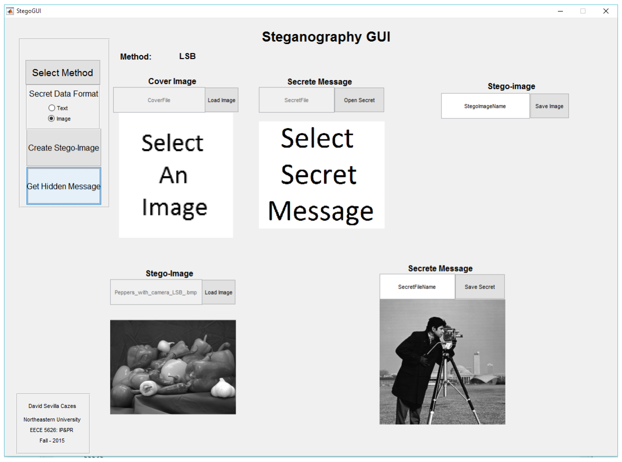
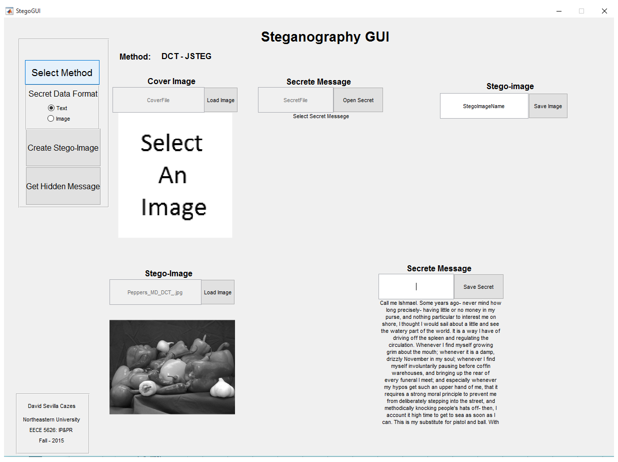


Figure 6: Retriveing LSB Message



Call me Ishmael. Some years ago- never mind how long precisely- having little or no money in my purse, and nothing particular to interest me on shore, I thought I would sail about a little and see the watery part of the world. It is a way I have of driving off the spleen and regulating the circulation. Whenever I find myself growing grim about the mouth; whenever it is a damp, drizzly November in my soul; whenever I find myself involuntarily pausing before coffin warehouses, and bringing up the rear of every funeral I meet; and especially whenever my hypos get such an upper hand of me, that it requires a strong moral principle to prevent me from deliberately stepping into the street, and methodically knocking people's hats off- then, I account it high time to get to sea as soon as I can. This is my substitute for pistol and ball. With

Figure 7: Retriveing JSTEG Message

5 Technique Comparison

The human eye cannot detect the difference between a normal image and one that has hidden information. This section explores some methods that can be used to detect if an image has concealed information.

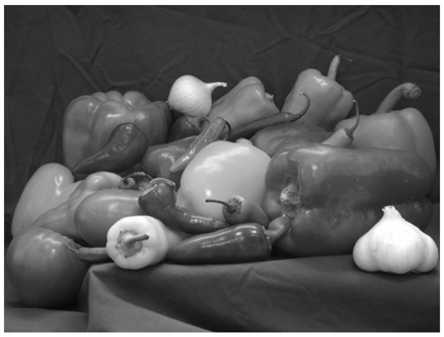


Figure 8: Cover Image

5.1 LSB analysis

For the LSB substitution method Figure 8 was used as a cover image and Figure 9 was embedded into the cover image resulting in the stego-image shown in Figure 10.

When observed the cover image and the resulting stego-image obtained through this method they seem indistinguishable. Figure 11 shows the error that exists between the original image and the stego-image. Note that the difference between the original image and the stego-image lies purely in the less significant bit of each pixel and closely resembles random noise in the pixels where the message is hidden.

Figure 12 and Figure 13 show the cover image and the stego-image histogram respectively. Both histograms have a similar distribution. With a close examination of the histogram it

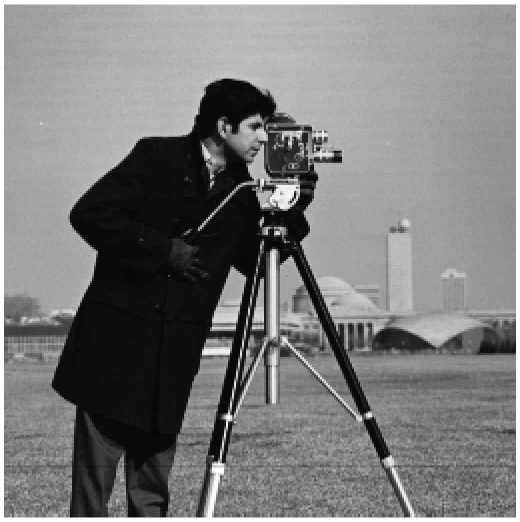


Figure 9:

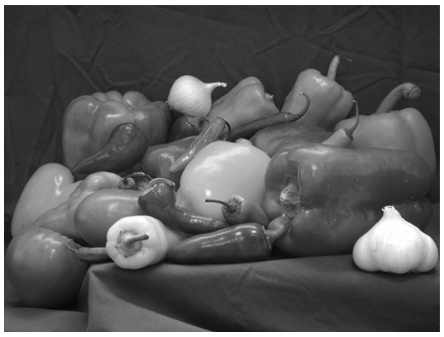


Figure 10: LSB Stego-Image

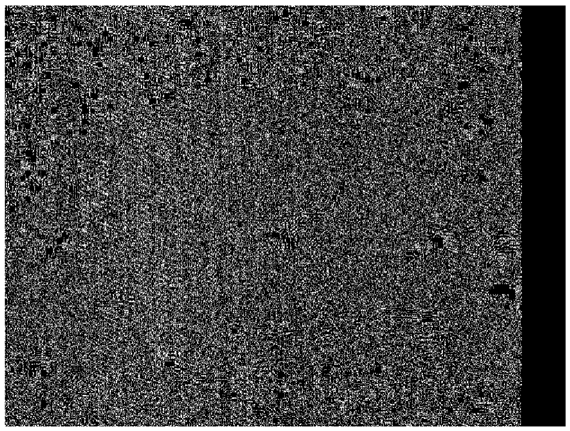


Figure 11: Difference between Cover Image and Stego-Image

is possible to see that the stego-image histogram seems noisier in the sense that spikes in the histogram can be observed, while the original image histogram is smoother. The mean square error (MSE) and peak signal to noise ration (PSNR) were calculated between the stego image and the cover image. Obtaining the following results.

MSE=0.4603

PSNR=51.5005 dB

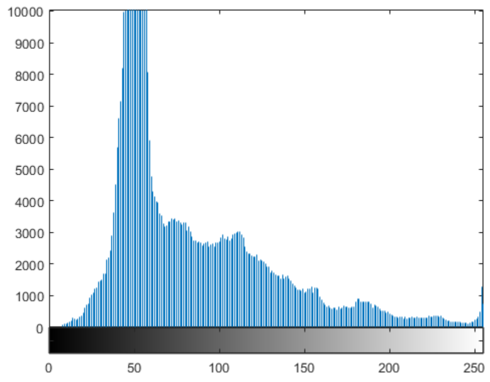


Figure 12: Cover Histogram

The stego-image was saved as a jpeg, with 75% quality compression, when passed through the message extraction algorithm, the hidden image was lost. When comparing the MSE and PSNR between the original image and the jpeg reconstruction of the stego-image. The MSE and PSNR where caluclated with a result similat to what would be expected from a 75% JPEG compression of the cover image.

MSE=0.0274

PNSR=63.7472

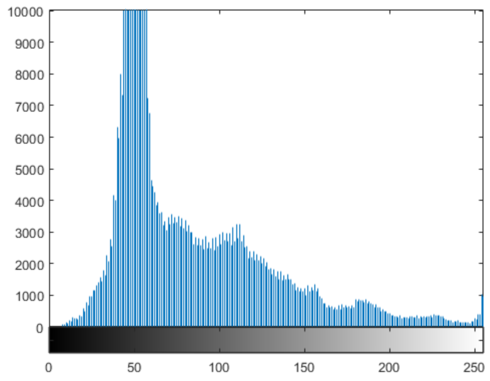


Figure 13: Stego-Image Histogram

5.2 DCT-JSTEG Analysis

For the Jsteg method analysis, Figure 14 was used as a cover image of size 256 x 256, which is a JPEG format image that was compressed with 75% quality from Figure 9 and the following extract form Herman Melville, Moby Dick was set as a hidden message.

```
   ‘‘Call me Ishmael.  Some years ago- never mind how long precisely- having little
or no money in my purse, and nothing particular to interest me on shore, I thought
I would sail about a little and see the watery part of the world.  It is a way
I have of driving off the spleen and regulating the’’
```

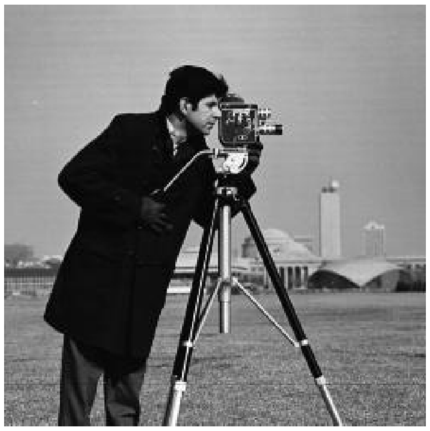


Figure 14: DCT Cover Image

Figure 15 shows the resulting stego-image.

After a visual inspection and comparison is performed between the cover image and the stego-image obtain using the Jsteg method both images seem the same image. When

Figure 15: DCT Stego-Image

plotting the error between the cover image and the stego-image it is possible to detect DCT manipulation, this is more evident at the higher frequencies of the image as can be seen in Figure 16

If we compare the histograms form the cover image (Figure 17) and the stego-image (Figure 18). It can be observed that both histograms have a close distribution, but in the stego-image histogram we can observe spikes similar to those observed in the LSB substitution method.

Finally a statistical comparison between the cover image and the stego image by calculating the mean square error and the peak signal to noise ratio with the following results:

MSE=6.57

PSNR=39.9552

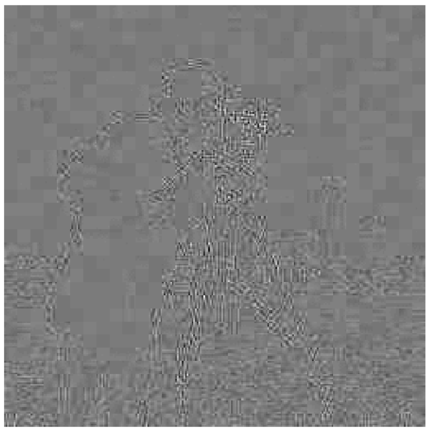


Figure 16: Jsteg error between cover image and stego-image

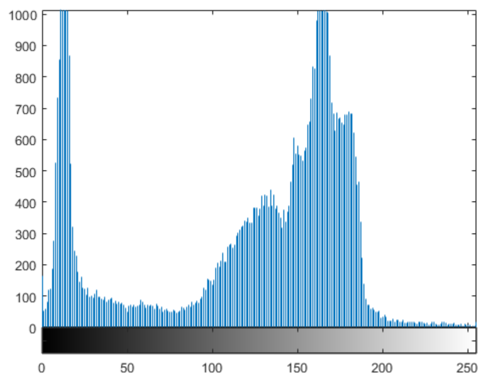


Figure 17: Jsteg Cover Histogram

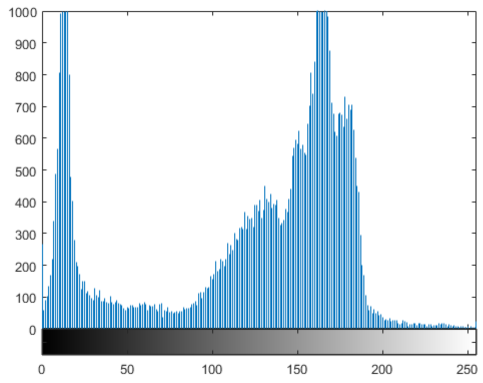


Figure 18: Jsteg Stego-Image Histogram

6 Conclusion

This report explored two different methods that are commonly used in image steganography. The first method is based on LSB substitution in the spatial domain while the second method, Jsteg, consisted on LSB substitution in the DCT domain. A user interface was developed that is capable of generating stego-images in which the steganography method can be chosen. Resulting images that where obtain through both steganography methods where analyzed.

The LSB substitution method is fragile since any further manipulation to the stego-image will erase the hidden information. In this method the mean square error is significantly small and the signal to noise ratio is significantly large and the concealed information might be confused with subtle noise.

It is possible to detect if an image has a hidden message through this method by looking at the histogram since the intensity distribution shows of an stego-images is not as smooth as a real world image. When the stego-image is compared with the cover image it is possible to confuse the error with noise and overlook that it is hidden information. If the error is analyzed it may show that the noise has a multimodal distribution. This problem can be avoided by adding bits with a desired distribution in the pixels where data will not be hidden. Also, changing the order of the information bits, or using a secondary codification strategy, will make the extraction of the message close to impossible.

The Jsteg method is more robust is more robust than the LSB substitution method since it is compatible with JPEG format, but it is still a fragile method, since the information will be lost if the image is stored with a different quality is change or if the image is run through though further processing. Because the hidden information is embedded in the DCT coefficients that are different of -1, 0 and 1 the size of the information that can be embedded using this method is significantly lower than using the LSB substitution method. Like, in the LSB substitution method, the histogram of the stego-image might suggest the hidden content.

If it is possible to compare the original cover image with the stego-image. The mean square error will be larger and the signal to noise ration will be lower than when using the LSB method. If the error of the original image is plotted it is apparent that there has been some DCT manipulation since DCT matrix pattern will be highly noticeable in the higher frequency elements. Even though the existence of a hidden message is more apparent using the Jsteg method extracting the hidden information without the knowledge of the embedding process might be more challenging than using the LSB substitution since the DCT transform of the stego-image would have different coefficients than the DCT of the original image.

Finally some properties that could be a key decision factor when deciding which steganography to use are the following:

For LSB substitution: The quality of the stego-image will be close to the cover image; large data can be concealed with this process; simple method to implement; fragile method; hidden information can be detected through histogram analysis; requires large storing space, effects are not detectable by human eye; the message will be destroyed after any image processing attempt.

For the Jsteg: Because of compression, a lower quality image might suggest traces of steganography; the data that can be stored is limited; more robust than LSB substitution; histogram is less reviling than with LSB substitution process; allows JPEG compression

References

[1] R. Poornima and R.J. Iswarya, “An Overview of Digital Image Steganography”, *International Journal of Computer Science & Engineering Survey*,4(1),pp. 23-31, (February 2013)

[2] N. Provos and P. Honeyman, “Hide and seek: an introduction to steganography”, *Security & Privacy, IEEE* [On-line],1(3), pp. 32-44, (June 2003). Available: `http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=1203220&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D1203220`

Cees Dekker. “Carbon Nanotubes as Molecular Quantum Wires”. *: Physics Today* [On-line]. 52(5), pp. 22-29 (May, 1999). Available: `http://scitation.aip.org/content/aip/magazine/physicstoday/article/52/5/10.1063/1.882658` [Apr. 2,2014]

[3] A. Stevenson, Oxford Dictionary of English, Oxford University Press, 2010.

[4] Vinay Ingle, EECE 5626. Class Lecture. “Image Processing and Pattern Recognition”. Northeastern University, Boston, MA. Fall 2015

7 Vita

David Sevilla has a Licenciatura in Manufacturing and Mechatronics Engineeringdegree from the Universidad Iberoamericana in Mexico City, is currently pursuing a M.Sc. in Electrical and Computer Engineering at Northeastern University, and is a Gordon Engineering Leadership Fellow.

David has experience and solid knowledge in automation, instrumentation, robotics,electronics and programming and has taken courses that cover the areas of electrical,computer, instrumentation, mechanical, control, and automation engineering. David also has a broad knowledge of physics and hands on experience building and debugging electrical engineering projects that involve different programming languages.

In the past David designed and built a pneumatic robot that was used to test detergents in a Quality Control Laboratory that operates in a hood under corrosive and volatile conditions for a cleaning industry leader company. He led a project that processed on LabVIEW data that was acquired from a meteorological station and turned it into a colloquial language weather report on twitter. He built a robot emulator GUI on MATLAB that ran the mathematical model of a manipulator and could save the robot motions into a program that could be loaded into an actual robot without stopping the production line.

He also have experience working with: PLC and microcontrollers; CAD tools such as AutoCAD and NX (Unigraphix), which he has used to design andmanufacture simple mechanical parts; Experience designing and implementing many instrumentation solutions and electronic circuits.

A LSB Code

A.1 LSBHide

```
function [ stego ] = LSBHide( I,M )
%LSBHIDE This function embeds a messaged M into an image I with the LSB
%process.
  c = I;
 [h,w]=size(c);

 %secret Messege is asigned
 message = M;% 'This is a secret, do not tell anyone';
 [hm,wm]=size(message);
 message=reshape(message,1,hm*wm);

 % validate that the message will fit in the image
 m = length(message) * 8;
 if m>(h*w)
 stego=[];
 return;
 %error('The image is to small, choose a larger image')
 end

 %convert the secret message into bit stream (of lenthg m)
    Ascii = [uint8(message),0];  %  concatenates the end of message characeter
    binStr = (dec2bin(Ascii,8));
    binStr = reshape(binStr',1,m+8);
    b=binStr;
    L=length(b); % Size of message

    %compute image size
    h = size(c,1);
    w = size(c,2);
    s=dec2bin(c,8);
    m=length(s);

    % add message
    s(1:L,8)=b;

    %recontruct stegoimage
    s=bin2dec(s);
    s=reshape(s,h,w);
    s=uint8(s);
    stego=s;
end
```

A.2 LSBretrive

```
function [ message ] = LSBretrive( stego )
%LSBRETRIVE This function retrives a hidden message from a stego-image that was encoded

   s = stego;
  [h,w] = size(s);
   s=reshape(s,h*w,1);
   s=dec2bin(s);
   m=s(:,8);
   k=mod(length(m),8);

    if k~=0
    m=str2num(m);
    m=cat(1,m,zeros(8-k,1));
    m=num2str(m);
    end

   m=reshape(m',8,length(m)/8);
   m=m';

   i = find(bin2dec(m)==0);  %looks for end of message
   if isempty(i)
   message=char(bin2dec(m))';
   warning('No Message');
   else
   message=char(bin2dec(m(1:i-1,:)))';

   end
end
```

A.3 DCTHide

```
function [ stego,key ] = DCThide( I,M,quality )
%DCThide This function embeds a messaged M into an image I with the DCT
%JSTEG process.

message=M;

I1=I;
[row coln]= size(I);
I= double(I);
% Subtracting each image pixel value by 128
I = I - (128*ones(size(I)));
```

```
% Quality Matrix Formulation
Q50 =stdQ;

 if quality > 50
     QX = round(Q50.*(ones(8)*((100-quality)/50)));
     QX = uint8(QX);
 elseif quality < 50
     QX = round(Q50.*(ones(8)*(50/quality)));
     QX = uint8(QX);
 elseif quality == 50
     QX = Q50;
 end

% Formulation of forward DCT Matrix and inverse DCT matrix
DCT_matrix8 = dctmtx(8);
iDCT_matrix8 = inv(DCT_matrix8);

% Jpeg Compression
dct_restored = zeros(row,coln);
QX = double(QX);

% Jpeg Encoding

% Forward Discret Cosine Transform
for i1=[1:8:row]
    for i2=[1:8:coln]
        zBLOCK=I(i1:i1+7,i2:i2+7);
        win1=dct2(zBLOCK);
        dct_domain(i1:i1+7,i2:i2+7)=win1;
    end
end
% Quantization of the DCT coefficients
for i1=[1:8:row]
    for i2=[1:8:coln]
        win1 = dct_domain(i1:i1+7,i2:i2+7);
        win2=round(win1./QX);
        dct_quantized(i1:i1+7,i2:i2+7)=win2;
    end
end

icoef=find(abs(dct_quantized)>1); % index of available coefficients for message

% validate that the message will fit in the image
n=numel(icoef);
```

```
[hm,wm]=size(message);
message=reshape(message,1,hm*wm);
L = length(message) * 8;
L=L+8;
 if (L)>n
 stego=[];
 key=[];
 return
 %error('The image is to small, choose a larger image')
 end

% message is converted to bit stream

    Ascii = [uint8(message),0];  %  concatenates the end of message characeter
    binStr = (dec2bin(Ascii,8));
    binStr = reshape(binStr',1,L);

% Quantizatoin index are converted to binary
indmes =  icoef(1:L); % index that will be changed
modified = dec2bin(dct_quantized(indmes)+128,8);
modified(1:length(binStr),8)=binStr;
modified=bin2dec(modified)-128;

dct_quantized(indmes)=modified;

% Jpeg Decoding
%  DCT Coefficients Dequantization
for i1=[1:8:row]
    for i2=[1:8:coln]
        win2 = dct_quantized(i1:i1+7,i2:i2+7);
        win3 = win2.*QX;
        dct_dequantized(i1:i1+7,i2:i2+7) = win3;
    end
end
% Inverse DCT
for i1=[1:8:row]
    for i2=[1:8:coln]
        win3 = dct_dequantized(i1:i1+7,i2:i2+7);
        win4=idct2(win3);
        dct_restored(i1:i1+7,i2:i2+7)=win4;
    end
end
I2=dct_restored+128;
```

```
stego=uint8(I2);
key=[quality;indmes];
%imwrite(stego,[filename,'.jpg'],'jpg','quality',quality);

end
```

A.4 DCTretrive

```
function [ message ] = DCTretrive( stego,key)
%DCTretireve This function retrives a hidden message from a stego-image that
%was encoded using DCTHide function.
quality=key(1);
indmes=key(2:end);
I = stego;
I1=I;
[row coln]= size(I);
I= double(I);
% Subtracting each image pixel value by 128
I = I - (128*ones(size(I)));

% Quality Matrix Formulation
Q50 = stdQ;

 if quality > 50
     QX = round(Q50.*(ones(8)*((100-quality)/50)));
     QX = uint8(QX);
 elseif quality < 50
     QX = round(Q50.*(ones(8)*(50/quality)));
     QX = uint8(QX);
 elseif quality == 50
     QX = Q50;
 end

% Formulation of forward DCT Matrix and inverse DCT matrix
DCT_matrix8 = dctmtx(8);
iDCT_matrix8 = inv(DCT_matrix8);

% Jpeg Compression
dct_restored = zeros(row,coln);
QX = double(QX);

% Jpeg Encoding

% Forward Discret Cosine Transform
for i1=[1:8:row]
```

```
    for i2=[1:8:coln]
        zBLOCK=I(i1:i1+7,i2:i2+7);
        win1=DCT_matrix8*zBLOCK*iDCT_matrix8;
        dct_domain(i1:i1+7,i2:i2+7)=win1;
    end
end
% Quantization of the DCT coefficients
for i1=[1:8:row]
    for i2=[1:8:coln]
        win1 = dct_domain(i1:i1+7,i2:i2+7);
        win2=round(win1./QX);
        dct_quantized(i1:i1+7,i2:i2+7)=win2;
    end
end

% index that can contain part of the message
icoef=find(abs(dct_quantized)>1); % index of available coefficients for message

s=dct_quantized(indmes)+128;
s=dec2bin(s);
m=s(:,8);
k=mod(length(m),8);
    if k~=0
    m=str2num(m);
    m=cat(1,m,zeros(8-k,1));
    m=num2str(m);
    end

m=reshape(m,8,numel(m)/8)';
message=char(bin2dec(m))';

end
```

A.5 stdQ

```
function [ q ] = stdQ( )
%StdQ returns the standar quantization matrix used in JPEG
%
q= [16 11 10 16 24 40 51 61; 12 12 14 19 26 58 60 55;...
     14 13 16 24 40 57 69 56; 14 17 22 29 51 87 80 62; ...
     18 22 37 56 68 109 103 77; 24 35 55 64 81 104 113 92;...
     49 64 78 87 103 121 120 101; 72 92 95 98 112 100 103 99];
end
```

A.6 StegoGUI

```
function varargout = StegoGUI(varargin)
% STEGOGUI MATLAB code for StegoGUI.fig
%      STEGOGUI, by itself, creates a new STEGOGUI or raises the existing
%      singleton*.
%
%      H = STEGOGUI returns the handle to a new STEGOGUI or the handle to
%      the existing singleton*.
%
%      STEGOGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in STEGOGUI.M with the given input arguments.
%
%      STEGOGUI('Property','Value',...) creates a new STEGOGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before StegoGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to StegoGUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help StegoGUI

% Last Modified by GUIDE v2.5 07-Dec-2015 17:54:43

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @StegoGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @StegoGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```
% End initialization code - DO NOT EDIT


% --- Executes just before StegoGUI is made visible.
function StegoGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to StegoGUI (see VARARGIN)

% Choose default command line output for StegoGUI
handles.output = hObject;




%initilaize path
set(handles.path1, 'string','CoverFile');
set(handles.path2, 'string','SecretFile');
set(handles.StegoName, 'string','StegoImageName');
set(handles.path3, 'string','StegoImageFile');
set(handles.path5, 'string','SecretFileName');

axes(handles.axes1)
handles.f=imread('selectImage.png');
handles.im1=imshow(handles.f);
set(handles.textIn,'String','Select Secret Messege')

axes(handles.axes2)
handles.g=imread('SelectMessage.png');
handles.im2=imshow(handles.g);
axes(handles.axes4)
imshow(handles.f);
%
% axes(handles.axes3)
% handles.stego=imread('StegoImage.png');
% handles.im3=imshow(handles.f);

handles.cover=0;
handles.message=0;
handles.ready=0;
handles.hidden=0;
handles.key=0;
handles.secret=[];
handles.k=[];
handles.stego=[];

handles.method='LSB';
handles.key=[];

% set(handles.StartButton, 'visible','on');
% set(handles.path1, 'visible','off');
% set(handles.path2, 'visible','off');
% set(handles.pushbutton1, 'visible','off');
% set(handles.pushbutton2, 'visible','off');
% set(handles.pushbutton3, 'visible','off');
% set(handles.uibuttongroup1, 'visible','off');
% set(handles.axes1, 'visible','off');
 set(handles.axes2, 'visible','off');
 set(handles.axes3, 'visible','off');
  set(handles.axes5, 'visible','off');
% set(handles.textIn,'visible','off');
% set(handles.im1,'visible','off');
% set(handles.im2,'visible','off');

% Update handles structure
guidata(hObject, handles);


% UIWAIT makes StegoGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = StegoGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



path=get(handles.path1,'string');
[handles.f,m,p,filename]=imagein();
[~,~,dim]=size(handles.f);
if dim==3
    handles.f=rgb2gray(handles.f);
end

set(handles.path1,'string',filename)
axes(handles.axes1)
imshow(handles.f)

handles.cover=1;


guidata(hObject, handles)


function path1_Callback(hObject, eventdata, handles)
% hObject    handle to path1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of path1 as text
%        str2double(get(hObject,'String')) returns contents of path1 as a double


% --- Executes during object creation, after setting all properties.
function path1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to path1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColo
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

path=get(handles.path2,'string');
if get(handles.radioImage,'value')
    [handles.g,m,p,filename]=imagein();
    set(handles.textIn,'visible','off');
    handles.g=handles.g+1;
    [m,n]=size(handles.g);
    if m==n
    set(handles.path2,'string',filename)
    %set(handles.axes2,'visible','on')
    axes(handles.axes2)
    imshow(handles.g)
    handles.message=1;

    else
        errordlg('You should use a square image');
        handles.message=0;

    end
end

if get(handles.radioText,'value')
    [handles.g,p,filename]=textin();
     set(handles.path2,'string',filename)
%     set(handles.axes2,'visible','off')
     set(handles.textIn,'visible','on')
     set(handles.textIn,'String',handles.g);
     handles.message=1;

end

guidata(hObject, handles)


function path2_Callback(hObject, eventdata, handles)
% hObject    handle to path2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of path2 as text
%        str2double(get(hObject,'String')) returns contents of path2 as a double
% --- Executes during object creation, after setting all properties.
function path2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to path2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColo
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


if (handles.cover && handles.message)
    handles.ready=1;

    axes(handles.axes3)
    if strcmp(handles.method,'LSB')
        handles.stego=LSBHide(handles.f,handles.g);
        handles.key=[];
    end
    if strcmp(handles.method,'DCT')
        [handles.stego,handles.key]=DCThide(handles.f,handles.g,75);
    end
    if isempty(handles.stego)
        uiwait(errordlg('The cover image is too small for your secret message'));
    else
    imshow(handles.stego)
    uiwait(msgbox(' Stego-Image Creation Completed'))
    end

else
    uiwait(errordlg('Please load an image a secret message'));
end


guidata(hObject, handles)
```

```
% --- Executes when figure1 is resized.
function figure1_SizeChangedFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

choice = questdlg('Would you like to hide or extract a secret?', ...
'Select Action', ...
'Hide','Extract','No thank you');
% Handle response

if strcmp(choice,'Hide')
    disp(choice)
else
    disp(choice)
end

status='on';

set(handles.path1, 'visible',status);
set(handles.path2, 'visible',status);
set(handles.pushbutton1, 'visible',status);
set(handles.pushbutton2, 'visible',status);
set(handles.pushbutton3, 'visible',status);
set(handles.uibuttongroup1, 'visible',status);
set(handles.axes1, 'visible',status);
set(handles.axes2, 'visible',status);
set(handles.axes3, 'visible',status);

set(handles.StartButton, 'visible','on');
set(handles.StartPanel, 'visible','off');


% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in StartButton.
function StartButton_Callback(hObject, eventdata, handles)
% hObject    handle to StartButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


handles.method = questdlg('What steganography method would you like to use?', ...
'Select Method', ...
'LSB','DCT','LSB');
% Handle response
if strcmp(handles.method,'LSB')
    disp(handles.method)
    status='on';
%     set(handles.path1, 'visible',status);
%     set(handles.path2, 'visible',status);
%     set(handles.pushbutton1, 'visible',status);
%     set(handles.pushbutton2, 'visible',status);
%     set(handles.pushbutton3, 'visible',status);
%     set(handles.uibuttongroup1, 'visible',status);

    set(handles.methodText,'String', 'LSB')
    %set(handles.axes1, 'visible',status);
    %set(handles.axes3, 'visible',status);



end
if strcmp(handles.method,'DCT')
    disp(handles.method)
    set(handles.methodText,'String', 'DCT - JSTEG')
    disp('select method')
    handles.method
end

guidata(hObject, handles);



% --- Executes on button press in radioText.
function radioText_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to radioText (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radioText

set(handles.textIn,'visible','on')
set(handles.textIn,'visible','on')


%file = textread('loremipsum.m', '%c', 'delimiter', '\n', 'whitespace','')'



function StegoName_Callback(hObject, eventdata, handles)
% hObject    handle to StegoName (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of StegoName as text
%        str2double(get(hObject,'String')) returns contents of StegoName as a double


% --- Executes during object creation, after setting all properties.
function StegoName_CreateFcn(hObject, eventdata, handles)
% hObject    handle to StegoName (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColo
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in imageOut.
function imageOut_Callback(hObject, eventdata, handles)
% hObject    handle to imageOut (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if handles.ready
    disp('saving..')
    handles.method
    filename=get(handles.StegoName,'String')
    handles.method
    if strcmp(handles.method,'LSB')
        newname=[filename,'_LSB_.bmp'];
        imwrite(handles.stego,newname)
        msgbox(['The stego-image has been saved as', newname],'Image Saved')
    end
    if strcmp(handles.method,'DCT')
        newname=[filename,'_DCT_.jpg'];
        imwrite(handles.stego,newname,'jpg','quality',75);

        fid=fopen([filename,'_DCT_.key'],'w');
        fwrite(fid,handles.key,'double');
        fclose(fid);
        msgbox(['The stego-image has been saved as', newname, ' with key: ', newname,'.k
    end

end


% --- Executes on button press in radioImage.
function radioImage_Callback(hObject, eventdata, handles)
% hObject    handle to radioImage (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.textIn,'visible','off')
set(handles.textOut,'visible','off')


% Hint: get(hObject,'Value') returns toggle state of radioImage


% --- Executes on button press in pushbutton9.
function pushbutton9_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[handles.s,m,p,filename]=imagein();
set(handles.path3,'string',filename)
axes(handles.axes4)

if strcmp(handles.method,'LSB')
imshow(handles.s)
```

```
end

if strcmp(handles.method,'DCT')
    i=find(filename=='.');
    key=[filename(1:i-1),'.key']
    fid=fopen(key);

    if fid~=-1
        handles.k=fread(fid,'double');
        fclose(fid);
        imshow(handles.s);
    else
        errordlg('The key file associatied with this image was not found');
    end

end
handles.hidden=1;



guidata(hObject, handles)


function path3_Callback(hObject, eventdata, handles)
% hObject    handle to path3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of path3 as text
%        str2double(get(hObject,'String')) returns contents of path3 as a double


% --- Executes during object creation, after setting all properties.
function path3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to path3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColo
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in pushbutton10.
function pushbutton10_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    name=get(handles.path5,'String');

if get(handles.radioText,'value')
    name=[name,'.txt'];
    fid=fopen(name,'w');
    fwrite(fid,handles.secret);
    fclose(fid);
    msgbox(['The hidden messge has been saved as ',name]);

end

if get(handles.radioImage,'value')

    imwrite(handles.secret,[name,'.bmp'])
    msgbox(['The hidden image has been saved as ',name]);
end

function path5_Callback(hObject, eventdata, handles)
% hObject    handle to path5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of path5 as text
%        str2double(get(hObject,'String')) returns contents of path5 as a double


% --- Executes during object creation, after setting all properties.
function path5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to path5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColo
    set(hObject,'BackgroundColor','white');
end
```

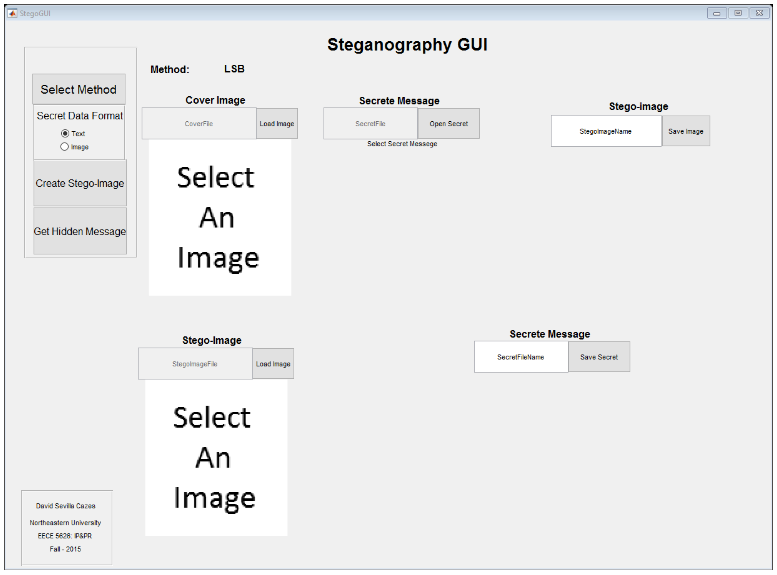
```
% --- Executes on button press in pushbutton11.
function pushbutton11_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if handles.hidden==1
    if strcmp(handles.method,'LSB')
        m=LSBretrive(handles.s);
        if get(handles.radioText,'Value')
            handles.secret=m;
            set(handles.textOut,'visible','on')
            set(handles.textOut,'String',handles.secret)
        end
        if get(handles.radioImage,'Value')
            len=length(m)
            m=reshape(m,sqrt(len),sqrt(len));
            handles.secret=uint8(m);
            set(handles.textOut,'visible','off')
            axes(handles.axes5)
            imshow(handles.secret)
        end

    end

    if strcmp(handles.method,'DCT')
        m = DCTretrive( handles.s,handles.k);
        handles.secret=m;
        set(handles.textOut,'visible','on')
        set(handles.textOut,'String',handles.secret);
    end
else
    errordlg('Please Load a Stego-Image')
end

guidata(hObject, handles)
```


StegoGUI
Steganography GUI
Select Method
Secret Data Format
Text
Image
Create Stego-Image
Get Hidden Message
Method:
LSB
Cover Image
CoverFile
Load Image
Select An Image
Secrete Message
SecretFile
Open Secret
Select Secret Messege
Stego-image
StegoImageName
Save Image
Stego-Image
StegoImageFile
Load Image
Select An Image
Secrete Message
SecretFileName
Save Secret
David Sevilla Cazes
Northeastern University
EECE 5626: IP&PR
Fall - 2015