Lab 5: Steganography

Background

Steganography is the practice of hiding information in some data or message. Contrasted with encryption, the goal of stenography is to produce a final message, which to the casual observer is unaltered, but actually contains a secret message.

In this lab you will create two MATLAB scripts: one to embed a message in a grayscale image and another to extract the message using the original image and the coded image.

A grayscale image is a 2-d array with pixels that vary from 0 (black) to 255 (white). Values in between are shades of grey becoming darker as the pixel value decreases.

A. Converting a Message to Binary

In this lab, you will be converting the ASCII values associated with the letters, numbers, and symbols to 8 bit binary numbers (i.e., each text character will be converted to a unique string of 8 ones and zeros).

As an example, consider the capital letter, H. The ASCII value for 'H' is 72. The corresponding 8-bit binary code ('01001000') is shown in the table below.

8-bit Binary Code for 'H'	0	1	0	0	1	0	0	0
Bit Values (Powers of 2)	128	64	32	16	8	4	2	1

Each bit represents a power of 2, starting on the right with 2^0 and ending on the left with 2^7. If we add up the bit values corresponding to the bits with 1s, we have: 64 + 8 = 72 (the ASCII code for letter 'H').

Another example, consider the small letter, i. The ASCII value for 'i' is 105. The corresponding 8-bit binary code ('01101001') is shown in the table below.

8-bit Binary Code for 'i'	0	1	1	0	1	0	0	1
Bit Values (Powers of 2)	128	64	32	16	8	4	2	1

Adding up the bit values corresponding to the bits with 1s, we have: 64 + 32 + 8 + 1 = 105 (the ASCII code for letter 'i').

Clearly the smallest number you can make with this 8-bit format is 0. **What is the largest number you can store with this 8-bit format? ANS:255**

The ***dec2bin*** function in MATLAB can be used to convert a message to a string of bits. At command prompt, type each of the following commands. Paste in the results.

```
>> message = 'Hi!'
```

Results: message =

```
'Hi!'
```

```
>> message = double(message)
```

Results: message =

```
 72   105    33
```

```
>> bin_message = dec2bin(message,8)
```

Results: bin_message =

3×8 char 数组

```
'01001000'
'01101001'
'00100001'
```

You can see that bin_message is currently a 2-d array with each text character in the original message forming a row in the array. It would be more convenient to have a 1-d array of all the bits for our encoding scheme. Type each of the following commands into MATLAB. Paste in the results and an explanation of what each command does.

```
>> bin_message = bin_message'
```

Results: bin_message =

8×3 char 数组

```
'000'
'110'
'011'
'000'
'110'
'000'
'000'
'011'
```

What did this command do?

It exchanges the rows and columns of matrix "message".

```
>> bin_message = bin_message(:)

   Results: bin_message =

  24×1 char 数组

    '0'
    '1'
    '0'
    '0'
    '1'
    '0'
    '0'
    '0'
    '0'
    '1'
    '1'
    '0'
    '1'
    '0'
    '0'
    '1'
    '0'
    '0'
    '1'
    '0'
    '0'
    '0'
    '0'
    '1'
```

What did this command do?
It expresses every number in matrix "message" in one column.

```
>> bin_message = bin_message'

   Results: bin_message =

    '010010000110100100100001'
```

What did this command do?
It expresses every number in matrix "message" in one row.

B. Importing Images, Displaying Images and Understanding the uint8 Format.

Download the cat.png image file from Blackboard under Lab 5 and save it in your Current MATLAB folder.

Type the following commands into MATLAB:

```
>> Pic = imread('Cat','png');
>> imshow(Pic)
```

Now you know how to import an image into MATLAB and how to display the image.

Look in your workspace window at the variable Pic and answer the following questions:

What is the size of Pic? ANS: 168 300

What is the datatype for the values in Pic? ANS: uint8

uint8 stands for unsigned 8 bit integer. In Part A, you learned that unsigned 8-bit integers can range from 0 to 255. Try the following commands.

```
>> a = uint8(255)

>> a + 1

Results: ans =

  uint8

   255

>> b = uint8(0)

>> b - 5

Results: ans =

  uint8

   0
```

Important Note: Since uint8 values can only range from 0 to 255, any computations you do with these values will be forced to stay in this range – a very important thing to remember for the next part of this lab!

C. Hiding a message in an Image

Here is how the coding scheme will work. Suppose the user inputs the message: 'Hi'. You will convert the message to a binary string using the commands from part A. The following table illustrates how the message will be encoded into CodedPic.

bin_message	0	1	0	0	1	0	0	0	0	1	1	0	1	0	0	1
Pic	254	255	75	75	75	73	73	74	74	72	72	72	72	73	73	73
CodedPic	254	254	75	75	76	73	73	74	74	73	73	72	73	73	73	74

- If a bit in bin_message is 0, then the corresponding pixel value for CodedPic is exactly the same value as the pixel value in Pic.
- If the bit in bin_message is a 1, then the corresponding pixel value for CodedPic is Pic + 1 unless the original pixel value is 255 in which case we subtract 1. (Why? Go back and look at the math exercise at the end of Part B).

Write a script file to do the following:

1. Clean up your workspace by making the first command: `clear; close all;`

2. Prompt the user for a message and convert it to 1-d binary string. *Remember the input command in MATLAB looks a bit different when you are asking for text rather than a number.*

3. Import the cat image and save it under some variable such as Pic.

4. Create a new variable for the coded image (CodedPic) and set it initially equal to Pic. Do not overwrite the values in Pic. In order to decode the message (Part D), you will need both the original image and the coded image.

5. Determine the size of CodedPic using the **size** command (don't hardcode the number of rows and columns).

6. Write a conditional statement that checks to see if the message is too long to fit in the image. If it is too long, use the ***error*** function to tell the user that the message is too long and terminate the script.

7. Create a nested loop with the ***outer loop going through the rows*** of CodedPic and the ***inner loop going through the cols*** of CodedPic. Within your loops, modify the pixels of CodedPic using the coding scheme described above.

    - You will need an additional variable to track your way through bin_message starting at entry 1, incrementing by 1 within the loops, and ending when you run out of bits (***break***). This is not an additional loop!
    - bin_message is a string of 1s and 0s. So, to determine if an entry is a 1, **DO NOT USE:** `if bin_message(n) ==1.`
      Use the strcmp function: `if strcmp(bin_message(n),'1').`

8. After the loops, use the imshow command to display the original image and the coded image. Remember, the command ***figure*** will open a new figure window so you won't lose the first image. Add titles to both indicating which is the original and which is the coded image.

9. Finally, use the imwrite command to save CodedPic as a png file:

```
imwrite(CodedPic,'CodedCat.png');
```

Test your code using the following message:

The more that you read, the more things you will know. The more that you learn, the more places you'll go. Dr. Seuss
If you get errors, debug away. Otherwise, if you get two images, run the following two commands in the command window and verify with your TA that your encoding scheme is indeed working properly:

```
>> Pic(1:4,1:8)
```

```
Results: ans =

  4×8 uint8 矩阵

   5   1   1   1   1   0   0   1
   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0
```

```
>> CodedPic(1:4,1:8)
```

```
Results: ans =

  4×8 uint8 矩阵

   5   2   1   1   2   0   0   1
   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0
```

Once you have verified that all is correct, paste you two images in the space below.

Can you see any differences between the two images?
No

Paste Script Here:

```
%% LAB5 Problem C
% Name: Horace
% Date: 26 Feb 2019

%% Code
%clear processor
clear; clc;close all;
message = input('What meassage do you want to
make?','s');
message = double(message) ;
bin_message = dec2bin(message,8);
bin_message = bin_message';
bin_message = bin_message(:);
bin_message = bin_message';
Pic = imread('Cat','png');

figure(1)
imshow(Pic);
hold on
title('Pic')

CodedPic = Pic;
```

```
[a,b] = size(CodedPic);
if length(bin_message)>a*b
    error('The message is too long and terminate the
script.')
end

n=1;
while n<length(bin_message)
for i = 1:a
   for j = 1:b
      if
n<length(bin_message)&&strcmp(bin_message(n),'1') &&
Pic(i,j)~=255
           CodedPic(i,j)=Pic(i,j)+1;
           n=n+1;
      elseif
n<length(bin_message)&&strcmp(bin_message(n),'1') &&
Pic(i,j)==255
           CodedPic(i,j)=Pic(i,j)-1;
           n=n+1;
      elseif
n<length(bin_message)&&strcmp(bin_message(n),'0')
           CodedPic(i,j)=Pic(i,j);
           n=n+1;
      else
          break
      end
       end
   end
end

figure(2)
imshow(CodedPic);
hold on
title('CodedPic')
imwrite(CodedPic,'CodedCat.png');
```

D. Extracting the Message

To recover the message, we need both the coded image and the original image because the message is actually the absolute value of the difference between the two images.

Write a second script file to do the following:

1. Clean up your workspace by making the first command: `clear; close all;`

2. Use the ***imread*** command to import Cat.png and CodedCat.png

3. Convert both image arrays, Pic and CodedPic, to doubles.

4. Create a new array, Difference = abs(Pic – CodedPic). This array is now filled with 1s and 0s corresponding to the original binary message.

5. Create a nested loop with the **outer loop going through the rows** of Difference and the **inner loop going through the columns** of Difference. Within the loops, extract the original binary message. Example: if Difference(row,col) is a 1, then bin_message(n) = '1'.

6. Now that you have the original binary message string, you will need to grab 8 bits at a time and convert each set of 8 bits back to the original message character:

```
message(1) = char(bin2dec(bin_message(1:8)))
message(2) = char(bin2dec(bin_message(9:16)))
```

   Obviously, you want to do this with a for loop!

Run your script and see if you successfully extract the original Dr. Seuss quote using the Cat.png and CodedCat.png images.

Paste Message Recovery Script Here:

```
%% LAB5 Problem D
% Name: Horace
% Date: 26 Feb 2019

%% Code
%clear processor
clear; clc;close all;
Pic = imread ('Cat.png');
CodedPic = imread ('CodedCat.png');
Pic = double(Pic);
CodedPic = double(CodedPic);
Difference = abs(Pic - CodedPic);
[a,b]=size(Difference)
n=1;
while n<17
for i=1:a
    for j=1:b
        if n<17 && Difference(i,j)==1
```

```
        bin_message(n) = '1'
        n=n+1
        elseif n<17 && Difference(i,j)==0
        bin_message(n) = '0'
        n = n+1
        else
            break
        end
    end
end
end
message(1) = char(bin2dec(bin_message(1:8)));
message(2) = char(bin2dec(bin_message(9:16)));
```

Questions:

1. How many letters could you hide in a 2736 x 3648 grayscale image using the encryption scheme from this lab?
   1247616
2. Assuming the typical English word in text is five letters long and we want the words separated by spaces, approximately how many words could you hide in a 2736 x 3648 grayscale image?
   207936
3. According to Amazon's Text Stats, the median length of a novel is 64,000 words. How many novels could you hide in a 2736 x 3648 grayscale image?
   3.249

To be turned in: This document with results, scripts, and answers to the questions.