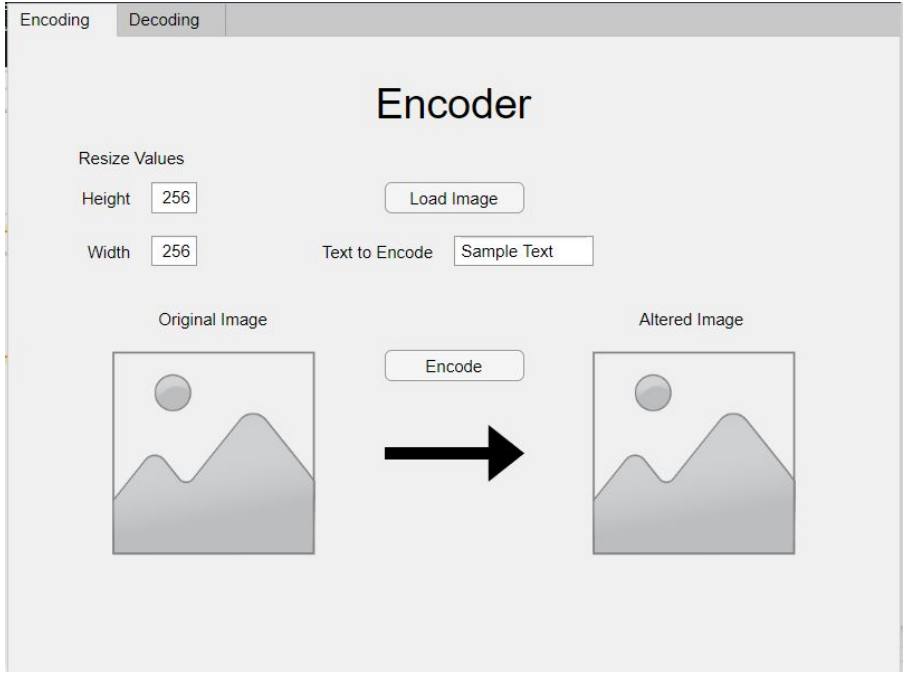
Steganography App

Sameer Narendran

Purpose

- Converts the image to a grayscale square
- Encodes a string of text inside of an image to prevent detection by others
- Decodes the text from the image
- Allows user specification of image height and width
- Uses Least Significant Bit Steganography



Least Significant Bit Steganography

- LSB Steganography is typically done using grayscale images
- The secret message is converted to binary
- The last value of each pixel is changed based on the binary message
    - If only the last value of the pixel is changed, it will not be too noticeable to viewers.

Encoding

- Message converted to ASCII and then to binary
- Loop through pixels in the image
    - For each compare the last bit of the pixel and the message bit
    - Add the message bit to the output picture pixel

```
function EncodeButtonPushed(app, event)
    message = app.messageProp;
    ascii = uint8(message);
    bin = dec2bin(ascii, 8)';
    bin = bin(:);
    l = length(bin);

    if (l ~= 0)
        im = app.imProp;
        h = app.hProp; w = app.wProp;
        imout = im;

        bin_num = str2num(bin);

        count = 0;

        for i = 1:h
            for j = 1:w
                if(count < l)
                    leastbit = mod(double(im(i, j)), 2);
                    added = double(xor(leastbit, bin_num(count+1)));
                    imout(i, j) = im(i, j)+added;
                    count = count + 1;
                end
            end
        end

        imout(h,w) = l;
```

Decoding

- Length of the message is gotten
- The last bit of each pixel in the image is taken
- Every 8 bits is converted into a character
- The characters are added together to find the original string

```
function DecodeButtonPushed(app, event)

    im = app.imProp;
    h = app.hProp; w = app.wProp;

    l = im(h,w);

    count = 0;
    for i = 1:h
        for j = 1:w
            if (count < l)
                bitarr(count+1, 1) = mod(double(im(i, j)), 2);
                count = count + 1;
            end
        end
    end


    binvals = [ 128 64 32 16 8 4 2 1 ];
    binmat = reshape(bitarr, 8, l/8);

    textString = char(binvals*binmat);
```

Decoding

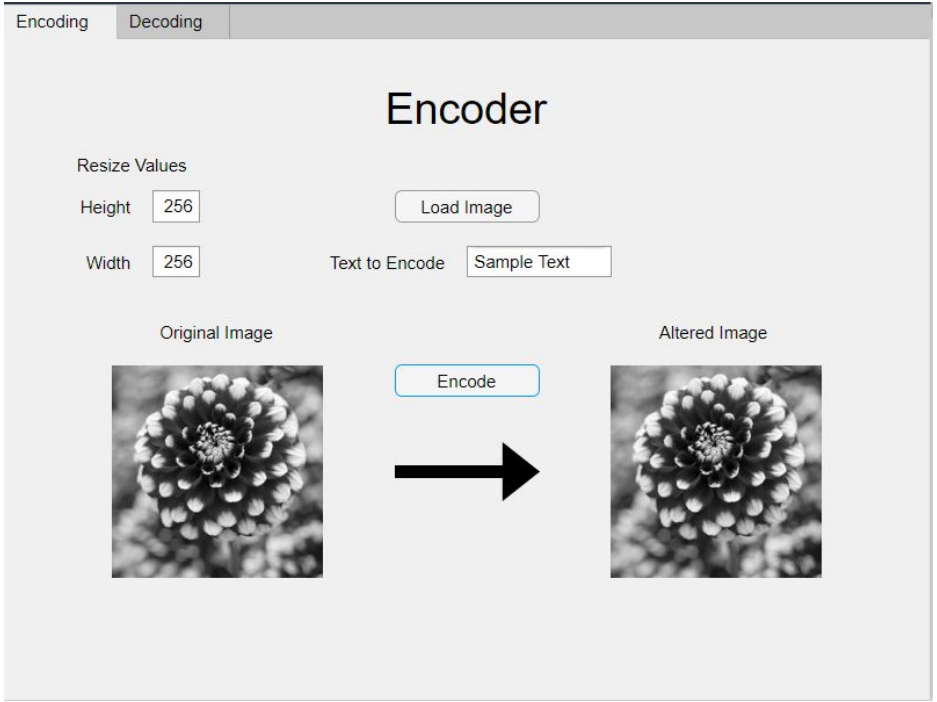
- Length of the message is gotten
- The last bit of each pixel in the image is taken
- Every 8 bits is converted into a character
- The characters are added together to find the original string

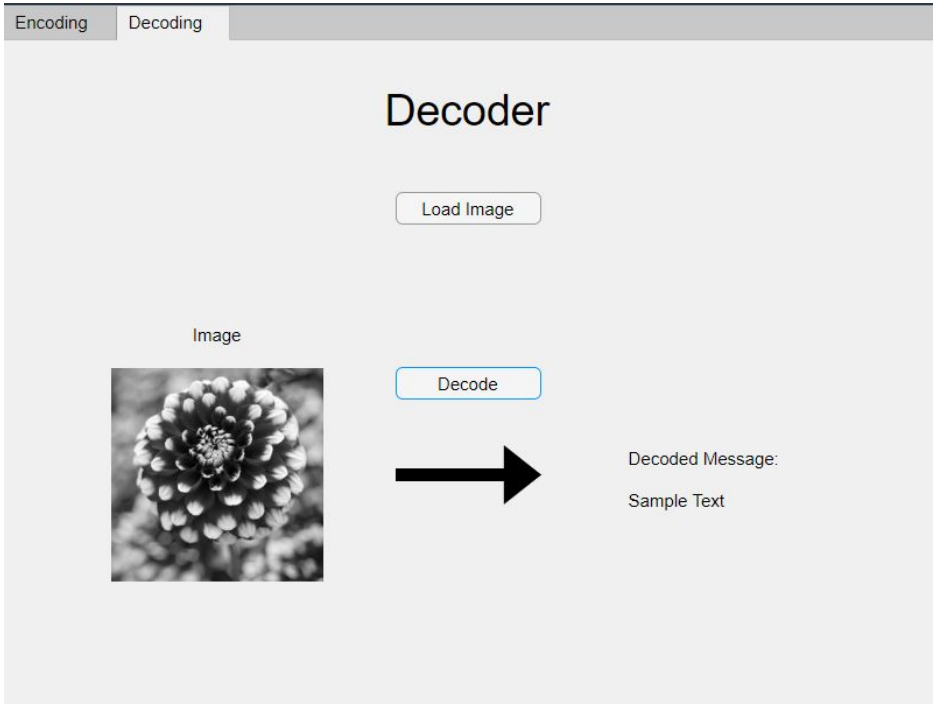
```
function DecodeButtonPushed(app, event)

    im = app.imProp;
    h = app.hProp; w = app.wProp;

    l = im(h,w);

    count = 0;
    for i = 1:h
        for j = 1:w
            if (count < l)
                bitarr(count+1, 1) = mod(double(im(i, j)), 2);
                count = count + 1;
            end
        end
    end


    binvals = [ 128 64 32 16 8 4 2 1 ];
    binmat = reshape(bitarr, 8, l/8);

    textString = char(binvals*binmat);
```

Conclusions

- Drawbacks
  - The method of steganography is weak and can be easily decoded by other people
  - Use of it may cause blurring or distortion of the images
- Advantages
  - Comparatively easy to create images with text encoded in them compared to other types of steganography

Figures


Encoding
Decoding
Encoder
Resize Values
Height
256
Load Image
Width
256
Text to Encode
Sample Text
Original Image
Encode
Altered Image


Encoding
Decoding
Decoder
Load Image
Image
Decode
Decoded Message:
Sample Text

What was Challenging

- Creating the binary matrix size for different messages
- Getting the least significant bit from the values
- Storing global variables in the App Designer
- Using the reshape() function