IMAGE STEGANOGRAPHY USING BLOCK – DCT

SUMMARY:

This project involves the extraction of hidden audio data from an image that has been embedded using block Discrete Cosine Transform (DCT) of the image. The hidden data is retrieved by scanning through the DCT-transformed image and sequentially collecting coefficients starting from a specified position, and decrypting the message using the encrypted key which will be embedded in the image, and it is retrieved to extract the audio.

PRINCIPLE:

Steganography is the technique of hiding secret information within a carrier medium such as an image, audio, or video.

In this project, the block Discrete Cosine Transform (DCT) is used, which transforms spatial-domain 8 x 8 block pixel values into frequency-domain coefficients. Data is embedded in the high frequency coefficients, as slight changes to these coefficients would not be perceivable to the human eyes. The key which has the information about where it is embedded in the image (length and width) is also embedded in the image at a particular position.

ALGORITHM:

1. Taking block DCT with 8 x 8 pixels values gives you the appropriate DCT coefficients.
2. Sperate out particular blocks of pixel values from the image of a particular width and height(low frequency components) for reconstruction, and set the remaining DCT coefficients(high frequency coefficients) to 0.
3. Replacing those 0 values with the values of the audio signal, without embedding the audio signal values into the separated block, by analyzing the borders of the separated block.
4. Embed the length of the audio signal, which is converted into equivalent binary digits, as part of key in the last 30 bits of the DCT converted image, by finding the last 30 columns in the last row.
5. After doing the embedding process, reconstruct the image by taking the inverse DCT.
6. Now the audio signal is embedded inside the image.
7. The key to identify at which position in the image the audio is embedded is nothing but the length and width at which the blocks are to be separated, so that the remaining values in the image can be read and displayed, which will give the audio output.
8. This key has to be known to the receiver in advance to decode the output in a proper format.
9. After receiving the stego image, DCT is taken for the image, to get the DCT coefficients of the image.

10 Using the key, first identify which part of the image is to be separated, and read the values of the image up to the length of the audio signal.

11. This reading has to be done such that the separated block should not be red, only the remaining rows should be red.

12. Once the values are retrieved, the audio output is ready to be used.

CODE:

```
%Audio Generating Code%
[audio, fs] = audioread("jan1.wav");
dn = downsample(audio, 2);
du = upsample(dn, 2);
stft(audio, fs)
%[b, t, info] = stftmag2sig(abs(x), size(x,1));
%b = abs(x);
%c = reshape(b, [], 1);
%norm = c/max(c);
length(audio)
length(du)
length(dn)   % -> To be used for embedding in the image
```

The Encoder

```
% Encoder & Embedder
```

```
function imagevec = stegano(image, encoded)
   [len1, len2] = size(image);
   bits = 1;
   done = false;

   for i = 1:len1
      for j = 1:len2
         if bits <= length(encoded) && image(i,j) == 0
            image(i,j) = encoded(bits);
            bits = bits + 1;
         end
         if bits > length(encoded)
            done = true;
            break;
         end
      end
      if done
         break;
      end
   end

   imagevec = image;
end

% Decoder and DeEmbedder
function answer = decsteg(image,x,y,len)
```

```
    [len1,len2] = size(image);
    answer = [];
    count = 0;
    flag = 0;
    for i = 1:len1
       for j = 1:len2
          if i < x
             if j > y
                answer = [answer,image(i,j)];
                count = count + 1;
             end
          end
          if i > x
             answer = [answer,image(i,j)];
             count = count+1;
          end
          if count == len
             flag = 1;
             break;
          end
       end
       if flag == 1
          break;
       end
    end
end
```

```
% Converts binary to decimal number
function dec = binary2dec(binary)
   newbin = [];
   for i = 1:length(binary)
      newbin = [newbin,binary(i)];
   end
   len = length(newbin);
   dec = 0;
   for i = 1:length(newbin)
      dec = dec + newbin(i)*(2^(len-i));
   end
end

% Reading the image
image = imread("mon.jpg");
image = im2gray(image);    % Convert to grayscale if needed
image = double(image);     % Convert to double for block dct
[x,y] = size(image);
mask = zeros(x,y);
mask(1:100,1:300) = 1;     % Low pass mask (Private keys)
dct2D = @(block) dct(dct(block')');  % Dct computing object
idct2D = @(block) idct(idct(block')');  %Idct computing object
dct_coeffs = dct2D(image);
dct_coeffs = mask.*dct_coeffs;         %Masking the dct coefficients
binary = dn;
```

```
dct_coeffs = stegano(dct_coeffs,binary); % Embedding the audio
len = length(binary);
newbinary = dec2bin(len);
newbin = [];
for i = 1:length(newbinary)
   newbin = [newbin,str2num(newbinary(i))];
end
dct_coeffs(end,end-30+1,end) = 0;      %Embedding the length of the signal
for i = 1:length(newbinary)
   dct_coeffs(end,end-length(newbinary)+i) = str2num(newbinary(i));
end
reconstructed = idct2D(dct_coeffs);   % Reconstruct the stegano image

% Show the original image
imshow(uint8(image));

% Show the reconstructed image
new = uint8(reconstructed);
imshow(uint8(reconstructed));

% Save the image
t = Tiff('stegano.tiff', 'w');

tagstruct.ImageLength = size(reconstructed,1);
tagstruct.ImageWidth = size(reconstructed,2);
```

```
tagstruct.SampleFormat = Tiff.SampleFormat.IEEEFP;  % Set sample
format to floating point
tagstruct.Photometric = Tiff.Photometric.MinIsBlack;
tagstruct.BitsPerSample = 32; % 32 bits for float
tagstruct.SamplesPerPixel = 1;
tagstruct.PlanarConfiguration = Tiff.PlanarConfiguration.Chunky;
t.setTag(tagstruct);

t.write(single(reconstructed));  % Write as single-precision float
t.close();
%{
newdct = dct2D(reconstructed);
new1binary = dct_coeffs(end,end-30+1:end);
dec = binary2dec(new1binary)
answer = decsteg(newdct,150,300,new1binary);
sound(answer,8000);
%}
```

The Decoder

```
% The decoding function
function answer = decsteg(image,x,y,len)
   [len1,len2] = size(image);
   answer = [];
   count = 0;
   flag = 0;
   for i = 1:len1
      for j = 1:len2
```

```
            if i < x
               if j > y
                  answer = [answer,image(i,j)];
                  count = count + 1;
               end
            end
            if i > x
               answer = [answer,image(i,j)];
               count = count+1;
            end
            if count == len
               flag = 1;
               break;
            end
         end
         if flag == 1
            break;
         end
      end
   end

% Dct and Idct computing blocks
dct2D = @(block) dct(dct(block')');
idct2D = @(block) idct(idct(block')');
newphoto = imread("stegano.tiff");
imshow(uint8(newphoto))
```

```
% Retrieve the data
newdct1 = dct2D(newphoto);
answer = decsteg(newdct1,150,300,length(binary));
```

REAL LIFE APPLICATIONS:

1. Platforms like Shutterstock or NFT markets embed ownership information invisibly inside the image using steganography rather than visible watermarks.
2. Military intelligence operations use image steganography to encode mission-critical data into harmless images shared over public channels.
3. Satellite images or social media posts could secretly carry embedded maps, instructions, or surveillance information.