Visual Cryptography for colour images using multilevel thresholding

Team members:

D.Vishal- S20190020210

B.Chaitanya Dattu S20190020204

1. Motivation

In this project we use Visual Cryptographic Scheme(VCS) as a technique to encrypt the image by dividing it into individual shares, and where decryption is performed by overlapping allthese shares using Human Visual System (HVS). Implementing VCS technique for colour images, by using multilevel thresholding.

We know the importance of the digital security and its demsnd to transfer the sensitive data also which not allowing the unauthorized recipients to change the original data. This Visual cryptography is basically to solve the problem of secret sharing by dividing it into n different shares.

And decryption is done by only overlapping all the divided n shares and no less than n shares retrieve any information about the original image. Making this data hiding technique to transfer message securely.

2. WorkPlan

From the Original image every individual colour component image is formed with colour decomposition of RGB model and coverted into its respective Halftoned image also considering n different levels pixel values.

Considering Halftoned triplet (RHij, GHij ,BHij), there would be (n^3) possible combinations of the triplet if the level in multilevel halftoning is taken as n.

And based on each pixel triplet of the red, green and blue halftones there forms different shares. (Building shares).

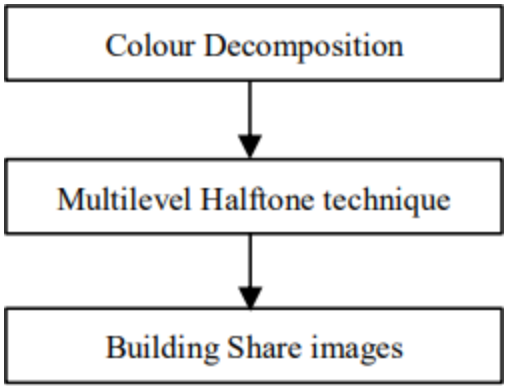


Fig: Flow diagram for Encryption

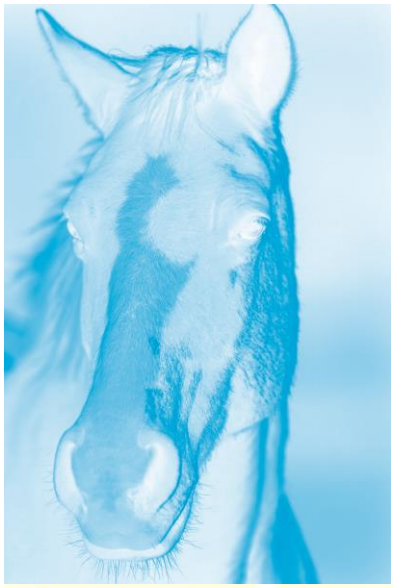
Fig: Original image

3. Implementation and progress of the work

This Technique uses colour decomposition and multilayer halftoning technique to construct shares based on the halftones obtained.

Colour decomposition is done basically by dividing the image into respective primary colour components based on RGB colour model. Making the three red, blue, green components of a secret image respectively.

3.1 RGB colour component images:

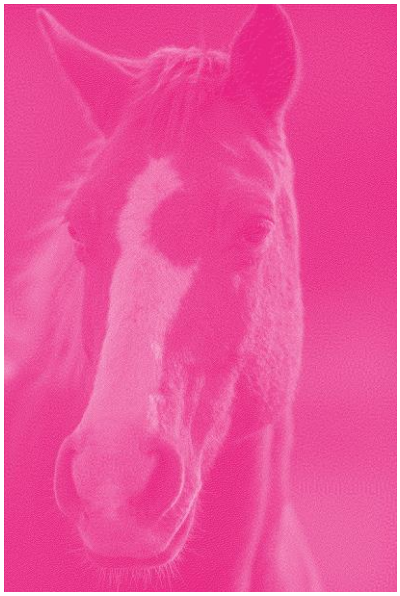


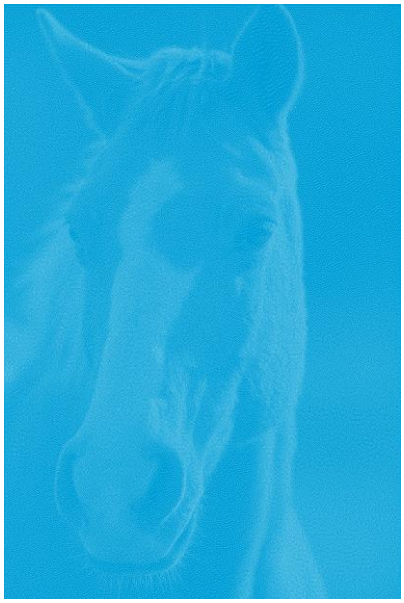
And every individual color component image is converted into respective Halftoned image such as RH, GH and BH respectively. Here we add multilevel halftoning considering n different levels of pixel values.
These halftoned images RH,GH & BH are then used for building up the shares.

Three shares are build in such a way that share 1 has red and green component pixel details of the secret image, share 2 contains the green and blue component details and share 3 has the red and blue components as well.

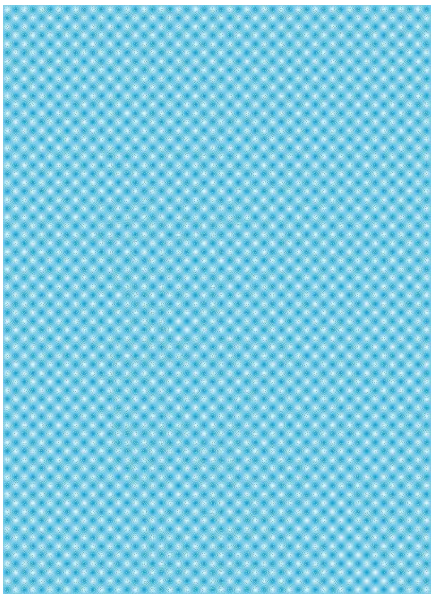
Considering n different levels of pixel values (n^3) possible combinations of the triplet(RH,GH,BH) are possible.

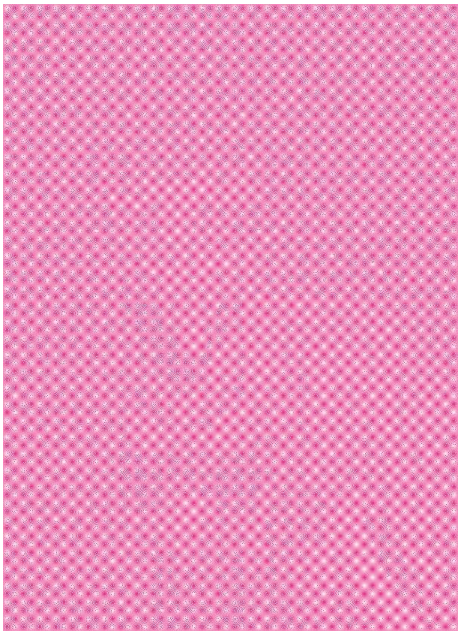
3.2 Halftoned images:





3.3 Building Shares :





From the respective halftones share building is done in such a way that, different combinations are choosen among these three. Corresponding pixels in share1 will have pixel values of red and blue and share2 will have pixel values of green and blue and share1 will have pixel values of red and green out of which 2 remaing are black for each of the shares.

They are placed in such a way that black pixels of a share coincides with the black pixels of the other share.

Here to be noted that, this is the reason why there will be difference with final decrypted and the original image with the same content.

3.4 Decrypting the image:

Decryption of the image is done in such a way overlapping the three share builded pictures will produce the original image context with in same pixels. If Intruder try overlapping any two images doesn't produce any meaning full image rather produces something with no meaning at all.(i.e also capable to provide security).

Even if one share is missing while reconstructing the image, its doesn't produce and reveal any information of the original image.Which it can be only reconstructed when all the three share are true and overlapped.

4.Conclusion

After this for each possible combination of halftoned images, four pixels are taken in share1, share2 and share3 each to build corresponding shares for that particular pixel triplet of original secret image.

And corresponding pixels in share1 will have two pixels of RH and GH out of four and other two will be black as same to other each of the shares. And these black pixels of any one share coincides with the black pixels of rest other two shares.

which is why also noted that the size of the shared image is twice the size of original image (2Mx2N).

While decrypting no single share could reveal any information of the secret image which provides secrecy of all the shares.So as on decrypting only by overlapping all the images original image can be retried.

- ➔ Colour details were based on levels in multi thresholding forming more detailed reconstructed image.
- ➔ Algorithm works for binary and greyscale images as well
- ➔ Secrecy of the shares is maintained
- ➔ One pixel of secret image is given by four pixels of each shared image(2 lengthwise x 2 breadthwise) so thus the reconstructed image has the contrast reduction of 50 %.

8. Code

Image color decomposition:

```
from PIL import Image as im

import random
import sys


image = im.open("D:\Monsoon 2022- 4\Crypto_gpy\proj\color_image.jpg")
color_image = image.convert('CMYK')
bw_image = image.convert('1')



output1 = im.new("CMYK", [dmn for dmn in image.size])

output2 = im.new("CMYK", [dmn for dmn in image.size])

output3 = im.new("CMYK", [dmn for dmn in image.size])



for i in range(0, image.size[0], 1):  # decomposition of the color image
    for j in range(0, image.size[1], 1):
        sourcepixel = image.getpixel((i, j))

        output1.putpixel((i, j),(sourcepixel[0],0,0,0))

        output2.putpixel((i, j),(0,sourcepixel[1],0,0))

        output3.putpixel((i, j),(0,0,sourcepixel[2],0))


output1.save('D:\Monsoon 2022- 4\Crypto_gpy\proj\out1.jpg')
```

```
output2.save('D:\Monsoon 2022- 4\Crypto_gpy\proj\out2.jpg')
output3.save('D:\Monsoon 2022- 4\Crypto_gpy\proj\out3.jpg')
```

RBG images to halftones:

```
from PIL import Image as im
import random
import sys

im1 = im.open("D:\Monsoon 2022- 4\Crypto_gpy\proj\out1.jpg") # image 1
im2 = im.open("D:\Monsoon 2022- 4\Crypto_gpy\proj\out2.jpg") # image 2
im3 = im.open("D:\Monsoon 2022- 4\Crypto_gpy\proj\out3.jpg") # image 3

im1 = im1.convert('1')
im2 = im2.convert('1')
im3 = im3.convert('1')

hlf_tn1 = im.new("CMYK", [dmn for dmn in im1.size]) # half-tone image 1
hlf_tn2 = im.new("CMYK", [dmn for dmn in im1.size]) # half-tone image 2
hlf_tn3 = im.new("CMYK", [dmn for dmn in im1.size]) # half-tone image 3

for i in range(0, im1.size[0]):      # reconstructing half-tone image
    for j in range(0, im1.size[1]):
        pxl_clr1 = im1.getpixel((i, j))
        pxl_clr2 = im2.getpixel((i, j))
        pxl_clr3 = im3.getpixel((i, j))

        if pxl_clr1 == 255:
            hlf_tn1.putpixel((i, j),(255,0,0,0))
        else:
            hlf_tn1.putpixel((i, j),(0,0,0,0))

        if pxl_clr2 == 255:
            hlf_tn2.putpixel((i, j),(0,255,0,0))
        else:
            hlf_tn2.putpixel((i, j),(0,0,0,0))

        if pxl_clr3 == 255:
            hlf_tn3.putpixel((i, j),(0,0,255,0))
        else:
            hlf_tn3.putpixel((i, j),(0,0,0,0))



hlf_tn1.save('D:\Monsoon 2022- 4\Crypto_gpy\proj\hf1.jpg')
hlf_tn2.save('D:\Monsoon 2022- 4\Crypto_gpy\proj\hf2.jpg')
hlf_tn3.save('D:\Monsoon 2022- 4\Crypto_gpy\proj\hf3.jpg')
```

Haltone to Share:

```
from PIL import Image as im
```

```
import random
import sys


im1 = im.open("D:\Monsoon 2022- 4\Crypto_gpy\proj\hf1.jpg")
im1 = im1.convert('CMYK')

im2 = im.open("D:\Monsoon 2022- 4\Crypto_gpy\proj\hf2.jpg")
im2 = im2.convert('CMYK')

im3 = im.open("D:\Monsoon 2022- 4\Crypto_gpy\proj\hf3.jpg")
im3 = im3.convert('CMYK')


shr1 = im.new("CMYK", [dmn * 2 for dmn in im1.size])

shr2 = im.new("CMYK", [dmn * 2 for dmn in im2.size])

shr3 = im.new("CMYK", [dmn * 2 for dmn in im3.size])




for i in range(0, im1.size[0]):
    for j in range(0, im1.size[1]):
        pxlclr = im1.getpixel((i, j))

        if pxlclr[0]+pxlclr[1]+pxlclr[2] == 0:
            shr1.putpixel((i * 2, j * 2), (255,0,0,0))
            shr1.putpixel((i * 2 + 1, j * 2), (0,0,0,0))
            shr1.putpixel((i * 2, j * 2 + 1), (0,0,0,0))
            shr1.putpixel((i * 2 + 1, j * 2 + 1), (255,0,0,0))

        else:
            shr1.putpixel((i * 2, j * 2), (0,0,0,0))
            shr1.putpixel((i * 2 + 1, j * 2), (255,0,0,0))
            shr1.putpixel((i * 2, j * 2 + 1), (255,0,0,0))
            shr1.putpixel((i * 2 + 1, j * 2 + 1), (0,0,0,0))

        pxlclr = im2.getpixel((i, j))

        if pxlclr[0]+pxlclr[1]+pxlclr[2] == 0:
            shr2.putpixel((i * 2, j * 2), (0,255,0,0))
            shr2.putpixel((i * 2 + 1, j * 2), (0,0,0,0))
            shr2.putpixel((i * 2, j * 2 + 1), (0,0,0,0))
            shr2.putpixel((i * 2 + 1, j * 2 + 1), (0,255,0,0))

        else:
            shr2.putpixel((i * 2, j * 2), (0,0,0,0))
            shr2.putpixel((i * 2 + 1, j * 2), (0,255,0,0))
            shr2.putpixel((i * 2, j * 2 + 1), (0,255,0,0))
```

```
            shr2.putpixel((i * 2 + 1, j * 2 + 1), (0,0,0,0))

        pxlclr = im3.getpixel((i, j))

        if pxlclr[0]+pxlclr[1]+pxlclr[2] == 0:
            shr3.putpixel((i * 2, j * 2), (0,0,255,0))
            shr3.putpixel((i * 2 + 1, j * 2), (0,0,0,0))
            shr3.putpixel((i * 2, j * 2 + 1), (0,0,0,0))
            shr3.putpixel((i * 2 + 1, j * 2 + 1), (0,0,255,0))

        else:
            shr3.putpixel((i * 2, j * 2), (0,0,0,0))
            shr3.putpixel((i * 2 + 1, j * 2), (0,0,255,0))
            shr3.putpixel((i * 2, j * 2 + 1), (0,0,255,0))
            shr3.putpixel((i * 2 + 1, j * 2 + 1), (0,0,0,0))



shr1.save('D:\Monsoon 2022- 4\Crypto_gpy\proj\share1.jpg')
shr2.save('D:\Monsoon 2022- 4\Crypto_gpy\proj\share2.jpg')
shr3.save('D:\Monsoon 2022- 4\Crypto_gpy\proj\share3.jpg')
```

Decryption:

```
from PIL import Image as im
import sys

input1 = im.open("D:\Monsoon 2022- 4\Crypto_gpy\proj\share1.jpg")  #share image 1
input2 = im.open("D:\Monsoon 2022- 4\Crypto_gpy\proj\share2.jpg")  #share image 2
input3 = im.open("D:\Monsoon 2022- 4\Crypto_gpy\proj\share3.jpg")  #share image 3

output = im.new('CMYK', input1.size)

for i in range(0,input1.size[0],2): # decrypting the share images
    for j in range(0,input1.size[1],2):

        C = input1.getpixel((i+1, j))[0]
        M = input2.getpixel((i+1, j))[1]
        Y = input3.getpixel((i+1, j))[2]



        output.putpixel((i, j), (C,M,Y,0))
        output.putpixel((i+1, j), (C,M,Y,0))
        output.putpixel((i, j+1), (C,M,Y,0))
        output.putpixel((i+1, j+1), (C,M,Y,0))


output.save("D:\\Monsoon 2022- 4\\Crypto_gpy\\proj\\final.jpg")  # final output of the image
using decryption
```

9. References

1.P. Kashyap and A. Renuka, "Visual Cryptography for colour images using multilevel thresholding," *2019 Third International Conference on Inventive Systems and Control (ICISC)*, 2019, pp. 567-572, doi: 10.1109/ICISC44355.2019.9036432.(link)

2.Moni Naor and Adi Shamir, "A Visual Cryptography", *Advances in cryptology EUROCRYPT 94 Lecture notes in computer science*, 1994.

3. Young-Chang Hou, "Visual Cryptography for colour images", *Department of information Management National Central University*, June 2002.