**Images Steganography using Pixel Value Difference**

MAJOR PROJECT REPORT

Submitted in fulfillment of the requirements for the award of the degree of **M**.**Sc**. **Computer Science**

**Submitted by**

KULPREET KOUR (1893904)

MONI TRIPATHI (1893908)

**Under the Supervision of**

Prof. Sunil Kumar Muttoo



Department of Computer Science, Faculty of Mathematical Sciences, University of Delhi

**ACKNOWLEDGEMENT**

It was an intellectually enriching experience to undertake this major project as a part of our MSc. Computer science 4th semester curriculum at Department of Computer Science, University of Delhi.

We sincerely express our gratitude towards our project supervisor **Mr**. **Sunil Kumar Muttoo** , Associate Professor, Department of Computer Science, University of Delhi, who guided us at every step in the conceptual undertaking and the implementation of the project work

We also express our thanks to the Head of Department of Computer Science, **Prof**. **Neelima Gupta**, for providing us the facilities to carry out the project work.

We also acknowledge the cooperation received from the entire laboratory and office staffs.

**DECLARATION**

We hereby declare that the project work entitled Image

Steganography being submitted in fulfilment of the

requirements for the award of the degree of M.Sc. (Computer

Science), is a record of original and bona-fide work carried out

by the undersigned in the Department of Computer Science,

Faculty of Mathematical Sciences, University of Delhi, New

Delhi, India. The work presented in this Project has not been

submitted to any other Institute or University for the award of

any degree or diploma.

**----------------- ------------------ Kulpreet Kour Moni Tripathi**

**TABLE OF CONTENTS**

Contents Page no.

* ACKNOWLEDGEMENT 2
* DECLARATION 3
* TABLE OF CONTENTS 4
* INTRODUCTION 6
* IMAGE STEGANOGRAPHY 9
* STEAGANALYSIS 10
* ADVANTAGES AND DISADVANTAGES 13
* IMAGE STEGANOGRAPHY USING PIXEL 15 VALUE DIFFERENCE

Implementation 17

Tools 17

Phases

Data Embedding 18

Data Extraction 20

* EXECUTION

Image Steganography On Color Image 21

Results 42

Filters on the Image 43

Histogram of the Image 55

Log tranform and Gamma Correction 57

* CONCLUSION & FUTURE SCOPE 60
* REFRENCE 61

**ABSTRACT**

A new data hiding method is proposed in this project , which can increase the steganographic security of a data hiding scheme. In the process of embedding a secret message,a cover image is partitioned into non-overlapping blocks of two consecutive pixels. A difference value is calculated from the values of the two pixels in each block. All possible difference values are classified into number of ranges. As human eye has limited tolerance when it comes to texture and edge areas than in smooth areas , and as the difference between the pixel pairs in those areas are larger. The difference value is then replaced by a new value to embed the value of a substream of the secret message.

The number of bits which can be embedded in a pixel pair is decided by the width of the range that the difference value belongs to. The method is designed in such a way that the modification is never out of range interval.

This method provides an easy way to produce a more imperceptible result than those yielded by simple least significant bit replacement methods.

The embedded secret message can be extracted from the resulting stego-image without refrencing the original cover image

**INTRODUCTION**

With the change in the world and advent of digital media , there are new methods which are evolved and adopted for communication . The advancement in the digital world lead to increase in the requirement of digital security and protection of digital data . As the time progressed it has been observed that digital media and information/data transfer suffered a lot from the lack of digital security through cyber attacks , hacking and became prey to cyber crimes. Criminals and technicians developed software which can hack into other’s system and can extract information through that . On the other hand many of the methods are also invented to make the network secure and prevent these type of hacking .

Among many of the things which are invented , it has been been observed that stegnography and cryptography are one of the finest and easiest method which can be used for protection of digital data . It has also been detected that , both of them work together to maximize the protection level or security of the data compared to the cases when they areused solely . Cryptography , basically scramble the data and twist the data in such a way that it gets turned into a cryptic message . While steganography do the hiding of digital data ,the cryptography make it impossible to decrypt and get the original value of the message.

In greek , the word “ Stego “ stands for “ roof “ or “ covered/hidden “ and “graphia “ means writing . So the literal meaning of steganography is hiding a data in another data ,message or information . The cover may be in the form of any other digital image , video or audio and this is an ancient method which is dated back to ancient roman times where a message is concealed in several other cover equipments just to hide that and make it undetectable . In some of the cases the messages are tattooed on the heads of the messenger and then the hairs of the messenger are allowed to grow so as to conceal the message .

Over the years vast range of steganography methods have been used which have been proven successful and failure according to their ability to hold and conceal the message .

A lot of research is still going on to develop more security and reduce the detectability of the concealed message. In the same direction , the computers and networks use several methods to hide data . Some of these include , hiding text within the web pages , hiding data in plain pages , using ciphers and encrypted messages to conceal more complicated single message which may contain the direction to find out the real hidden message in a cryptic way .Therefore in modern times , a large amount of data is stored in images and a lot of efforts are put in to increase the capacity of data hiding in various available covers.

Protection against detection can be needed if someone wants to ensure that the embedded message is not detected by a third unauthorized party. For example, a user wants to prevent others from finding out that an image contains a secret message hidden by the "Least-bit insertion method". This aim of data hiding is achieved by using schemes that do not modify the original object in a visible way; all changes should be indiscernible to the human eye or the computer. Protection against removal, on the other hand, tries to prevent the removal of hidden data without making it useless or degrading its quality. There are incomprehensible and therefore attracts the attention of observers and hackers . And this isone of the biggest difference which helps steganography to become one of the best method for data transfer secretly and securely.

**Need Of Information Security**

In modern times , technology security is one of the biggest concern . As the cyber crime is on rise , not only a secured network is needed but also it is necessary to provide security to images like blue print of company projects and their product designs , confidential data and images related to army or country’s security , using image steganography .As the messages are encrypted by several algorithms , its difficult to find them in the image.

**IMAGE STEGANOGRAPHY**

When the principles of steganography are applied over the images for getting an output which have data/ information concealed in them , then its called image steganography. Here the data is hidden in the form of text in the images which generally camouflage the secret data.



**STEGANALYSIS**

It has been observed that , digital watermarking and the steganography are one of the most secure method for digital data communication . This has led to development of methods which can find out the digital data which is hidden in the audio or image or video . Steganalysis is one of the method which has played an important role in detection if such hidden data which has been sent using steganography. As terrorists or other criminal organization use this method to have their covert exchange of data and even for communication purpose , the method of steganalysis is used to detect and extract the hidden messages from the cover .

In the method of steganalysis it is priory assumed that , the data is hidden in the images and the entire method is designed on the basis of theory of statistics .The process works in such a way that it narrows down to file which has hidden data from the original set of files . Due to abundant amount of images and a large variety of encoding algorithms , steganalysis is a challenge. An algorithm named general steganalysis method don’t depend on cover image or the encoding algorithm type , thus many obstacles incurred in obtaining the original cover image and the method used are avoided .

**Data hiding techniques**

Lossless data hiding techniques is a method of compressing data in a way such that the original data is retrieved without any data loss. The image obtained is the exact replica of the original image. The quality of the data that is compressed is not degraded and the exact data is obtained without any loss. Lossless compression methods may be categorized according to the type of data they are designed to compress. Some main types of targets for compression algorithms are text, executables, images, and sound. Whilst, in principle, any general-purpose lossless compression algorithm (general-purpose means that they can handle all binary input) can be used on any type of data, many are unable to achieve significant compression on data that is not of the form that they are designed to deal with.

Sound data, for instance, cannot be compressed well with conventional text compression algorithms. Most lossless compression programs use two different kinds of algorithms: one which generates a statistical model for the input data, and another which maps the input data to bit strings using this model in such a way that "probable" (e.g. frequently encountered) data will produce shorter output than "improbable" data. Often, only the former algorithm is named, while the second is implied (through common use, standardization etc.) or unspecified.

**Categories of Steganography**

****

**ADVANTAGES OF STEGANOGRAPHY**

1. It is robust method .
2. It provide a secret and safe method for transmission of data .
3. As the file show no physical change after steganography , it go undetected .
4. The method is also used in holding the digital data in images , not discretly but purposely and is being used in medical technology to.
5. It has larger data hiding capacity .

**DISADVANTAGES OF STEGANOGRAPHY**

1.Staganalysis can be used to extract the data and it gets easy when the image is subjected to staganalysis attacks .

2. The extracted data can be easily be understood as the data is not be in the cryptic form and the important information which is supposed to be hidden can be easily perceived by the person who checks that .

3. Message is hard to recover if image is subject to attack such as translation and rotation.

4. Significant damage to picture appearance. Message difficult to recover.

5. Image is distorted. Message easily lost if picture subject to compression such as JPEG.

**IMAGE STEGANOGRAPHY USING PIXEL-VALUE DIFFERENCING (PVD)**

The pixel-value differencing (PVD) scheme uses the difference value between two consecutive pixels in a block to determine how many secret bits should be embedded.

The data hiding techniques is carried out in three domains , namely spatial domain, compress domain, and frequency domain. The domains specified has their own advantages and disadvantages when it comes to the hiding capacity , execution time and the payload capacity. Recently in the research field in this domain , many researchers are more enthusiastic to improve the embedding efficiency and decrease the possibility of detection. Least-significant-bit matching is the conventional effective steganography method, and it is proved much more problematic to detect than simple LSB replacement

.

* Pixel Value Differencing (PVD) is able to provide a high quality stego image in spite of the high capacity of the concealed information.
* That is, the number of insertion bits is dependent on whether the pixel is an edge area or smooth area.
* In edge area the difference between the adjacent pixels is more, whereas in smooth area it is less.
* This method hides the data in the

target pixel by finding the

characteristics of four pixels

surrounding it, indicated in the table



**Implementation**

Tools

**Language used**: python

**Platform used** : Anaconda

**IDE** : Jupyter notebook

**Packages used:**

* PIL
* SYS
* OS
* NUMPY
* CV2
* MATPLOTLIB
* IPYTHON.DISPLAY
* IMAGEIO
* MATH

**Phases**

This project is divided into 2 phases:

1. Data Embedding

2. Data Extraction

**Embedding Algorithm**

Step 1.We simply divide the cover image into a number a non overlapping two pixel blocks.

Step 2.Each block is categorized according to difference of the gray values of the two pixels in the block.

Step 3.A smooth difference value indicates that the block is in smooth area a larger one indicates that it is in edged area.

Step 4.The pixels in edged areas may, as mentioned previously , tolerate larger changes of pixel values than those in the smooth areas.

Step 5. Read secret bits from the secret bit stream, and

transform it into decimal value .

Repeat Steps 1–5 until all secret bits are embedded and the stego image is produced.

Step 6. Classify pixels based on the difference in pixel value to the number of bits to be substituted to LSB.

Step 7. Calculate embedding capacity of the given cover image using pixel value difference.

Step 8. Calculate the embedded bits in the stego image.

Step 8.1 If the number of bits required is less than the number of bits in the data(char.) to be Embedded.

Step 8.2 If the number of bits required is more than the number of bits in the data(char.) to be Embedded

**Extraction Algorithm**

Step 1. Check if a new character in embed log is reached.

Step 2. Check if embedded pixel is red.

Step 3. Check if embedded pixel is green.

Step 4. Check if embedded pixel is blue.

Step 5.For checking if character has changed in embed file

**EXECUTION**

**IMAGE STEGANOGRAPGY ON COLOR IMAGE:**

* **WE HAVE TAKEN FIRST IMAGE**

****

ORIGINAL IMAGE STEGO IMAGE

* **WE HAVE TAKEN SECOND IMAGE**

** **

ORIGINAL IMAGE STEGO IMAGE

* **WE HAVE TAKEN THIRD IMAGE**

** **

ORIGINAL IMAGE STEGO IMAGE

* **WE HAVE TAKEN FOURTH IMAGE**

** **

ORIGINAL IMAGE STEGO IMAGE

**EMBEDDING PROCESS**

from PIL import Image

import sys

import os

# File Objects creation

input = open("data1.txt", "r") #opening the txt file in which we are having the data

im = Image.open("lady.jpg") #opening the image

lg = open("embedlog.log", "w")

# Initialisation

pix = im.load() # loading the pixels values of the image

hi, wi = im.size # getting the size of the image i.e. its height and width

print(hi,wi)

completed = 0 #initialize the embedding flag to 0 and when embedded is completed it is changed to 1

retrieved = "" #to check whether the data embedded is reached its end or not

count = 0

paddbits = "0000000"

#reading the whole txt file line by line till end

binval = input.read(1)

#print(binval)

charNum = 1 #Increment the char count of embedded data

if len(binval) == 0: #for checking the empty file

print("\nEmpty i/p File!")

sys.exit("Exiting...")

b = ord(binval) #calculating the ordinal value of the character

bitstring = bin(b) #calculating the binary value of the character

bits = bitstring[2:] #slicing the binary value of the character to get the exact binary value

capacity = 0 #to calculate the embedding capacity

lix = hi // 3

liy = wi // 3

# Classify pixels based on the difference in pixel value to the number of bits to be substituted to LSB

def classify(pvd):

nbits = 0

if pvd < 16:

nbits = 2

elif 16 < pvd < 32:

nbits = 3

else:

nbits = 4

return nbits

# Calculate embedding capacity of the given cover image

def calcCapacity():

global capacity

# Divide pixels to [3 x 3] matrix

for i in range(0, lix \* 3, 3):

for j in range(0, liy \* 3, 3):

# Obtain pixel values of ref. pixel

rref, gref, bref = pix[i + 1, j + 1]

# For all pixels in the matrix

for k in range(i, (i + 3)):

if k >= hi:

break

for l in range(j, (j + 3)):

if k == i + 1 and l == j + 1:

continue

if l >= wi:

break

# Calculate the difference in pixel values

r, g, b = pix[k, l]

rdif = r - rref

gdif = g - gref

bdif = b - bref

rdif = abs(rdif)

gdif = abs(gdif)

bdif = abs(bdif)

# Cumulative capacity

capacity = (

capacity + classify(rdif) + classify(gdif) + classify(bdif)

)

# Return capacity

return capacity

# Function to embed data to pixel

def embedbits(i, j, pixel, diff, colorpixel):

global bits, count, bitstring, paddbits, binval, completed, retrieved, input, charNum

# Initialise

pad = 0

nb = diff

# If the number of bits required is less than the number of bits in the data(char.) to be Embedded

if nb < len(bits):

# Initialise

newbits = bits[:nb]

bits = bits[nb:]

val = colorpixel

data = newbits

bival = bin(val)

bival = bival[2:]

newbival = bival[: (len(bival) - len(data))] + data

# Write data to log File for extraction

lg.write("%s %s %s %s %s %s %s" % (i, j, pixel, diff, pad, charNum, "\n"))

# Return new pixel value after embedding

return int(newbival, 2)

# If the number of bits required is greater than the number of bits in the data(char.) to be Embedded

else:

# Apply padding

newbits = bits + paddbits[: (nb - len(bits))]

#print(newbits)

pad = nb - len(bits)

#print(pad)

val = colorpixel

data = newbits

bival = bin(val)

bival = bival[2:]

newbival = bival[: (len(bival) - len(data))] + data

count += 1

#print(count)

# Write data to log File for extraction

lg.write("%s %s %s %s %s %s %s" % (i, j, pixel, diff, pad, charNum, "\n"))

# Read new char. for embedding

binval = input.read(1)

# Check if file containing data to embed reached its end

if len(binval) == 0:

print("Embedding Completed")

# Close input file object

input.close()

# Activate complete flag

completed = 1

# Return new pixel value after embedding

return int(newbival, 2)

# Check if file containing data to embed havent reached its end

b = ord(binval)

bitstring = bin(b)

bits = bitstring[2:]

retrieved = ""

# Increment the char count of embedded data

charNum += 1

# Return new pixel value after embedding

return int(newbival, 2)

# Main Function

def main():

# Initialise counter containing num of bits embedded till embedding ends

embedded = 0

# Print total Embedding capacity

print("Total Embd. Capacity: ", calcCapacity())

# Divide pixels to [3 x 3] matrix

for i in range(0, lix \* 3, 3):

for j in range(0, liy \* 3, 3):

# Obtain pixel values of ref. pixel

rref, gref, bref = pix[i + 1, j + 1]

# For all pixels in the matrix

for k in range(i, (i + 3)):

if k >= hi:

break

for l in range(j, (j + 3)):

if k == i + 1 and l == j + 1:

continue

if l >= wi:

break

# Calculate pixel value difference

r, g, b = pix[k, l]

rdif = r - rref

gdif = g - gref

bdif = b - bref

rdif = abs(rdif)

gdif = abs(gdif)

bdif = abs(bdif)

# Till embedding gets completed

if completed == 0:

newr = embedbits(k, l, "r", classify(rdif), r)

if completed == 0:

newg = embedbits(k, l, "g", classify(gdif), g)

if completed == 0:

newb = embedbits(k, l, "b", classify(bdif), b)

# Embedding completed

if completed == 1:

# Assign modified pixel values

pix[k, l] = (newr, newg, newb)

# Save embedded image

im.save("protest.png")

# Close log file

lg.close()

print("Embedded:", embedded, "bits")

# Exit program

sys.exit("Done..Exiting main prog.")

# Calculate the number of bits embedded

embedded = (

embedded + classify(rdif) + classify(gdif) + classify(bdif)

)

# Assign modified pixel values

pix[k, l] = (newr, newg, newb)

# Exit if Data size greater than embedding capacity

sys.exit("Exiting... Data size greater than embedding capacity!!")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**EXTRACTING PROCESS**

from PIL import Image

import sys

import os

# File Objects creation

im = Image.open("protest.png")

outp = open("abc.txt", "w")

lg = open("embedlog.log", "r")

# Initialisation

pix = im.load()

temp = 1

chrtr = ""

# Main Function

def main():

global chrtr, temp

while True:

# Read each line from log file

st = lg.readline()

# Check if log file reached its end

if len(st) == 0:

# Write extracted data to file

outp.write(chr(int(chrtr, 2)))

break

# Unpack line read from log file to variables

i, j, pixel, diff, pad, charNum = st.split()

# Process variables

i = int(i)

j = int(j)

diff = int(diff)

pad = int(pad)

charNum = int(charNum)

r, g, b = pix[i, j]

# Check if a new character in embed log is reached

if temp != charNum:

outp.write(chr(int(chrtr, 2)))

chrtr = ""

# If embedded pixel is red

if pixel == "r":

binr = bin(r)

binr = binr.replace('0b','')

chrtr += binr[(len(binr) - diff) :

# If embedded pixel is green

if pixel == "g":

binr = bin(g)

binr = binr.replace('0b', '')

chrtr += binr[(len(binr) - diff) :]

# If embedded pixel is blue

if pixel == "b":

binr = bin(b)

binr = binr.replace('0b', '')

chrtr += binr[(len(binr) - diff) :]

# Unpad if padding is done

if pad != 0:

chrtr = chrtr[: (len(chrtr) - pad)]

# For checking if character has changed in embed file

temp = charNum

# Close file objects

outp.close()

lg.close()

print("Extracting to the file:","abc.txt")

print("Extraction completed... Exiting!")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**PEAK SIGNAL TO THE NOISE RATIO**

**PSNR** is defined as follows:

PSNR=10Log10(L-1)^2/MSE=20Log10(L-1)/RMSE  
Here, **L** is the number of maximum possible intensity levels (minimum intensity level suppose to be 0) in an image.

**MSE** is the mean squared error & it is defined as:

MSE=1/mn

Where, **O** represents the matrix data of original image. **D** represents the matrix data of degraded image. **m** represents the numbers of rows of pixels and **i** represents the index of that row of the image. **n** represents the number of columns of pixels and **j** represents the index of that column of the image.  
**RMSE** is the root mean squared error.

"""

Signal-to-noise ratio numbers are all about the strength of the desired signal compared to the unwanted noise.

The larger the number, the more the desired signal “stands out”

in comparison to the noise, which means a clearer transmission of better technical quality.

A negative number means the noise is stronger than the desired signal, which may spell trouble,

such as a cell phone conversation that’s too garbled to understand.

"""

import numpy

import math

import cv2

original = cv2.imread("lady.jpg")

contrast = cv2.imread("protest.png",1)

def psnr(img1, img2):

mse = numpy.mean( (img1 - img2) \*\* 2 )

if mse == 0:

return 100

PIXEL\_MAX = 255.0

return 20 \* math.log10(PIXEL\_MAX / math.sqrt(mse))

d=psnr(original,contrast)

print(d)

**RESULTS**

No. Of Characters in txt PSNR

1,218 58.8268

3,654 54.2545

7,308‬ 51.5452

12,180 49.3938

18,270 46.3541

24,360 44.0011

30,450‬ 42.8355

36540 41.9411

42630 41.3420

48720 40.9492

54810 40.6257

60900 40.6257

66990 40.6257

**FILTERS ON THE IMAGE**

**BLURRING OF THE IMAGE**

****ORIGINAL IMAGE **** BLUR IMAGE

**RESLUTS**

#no. of characters in the txt file psnr value

#1,218 60.316

#3,654 55.876

#7,308‬ 53.258

#12,180‬ 51.5680

#18,270 49.42248

#24,360 47.8772

**CONTOURING OF THE IMAGE**

****ORIGINAL IMAGE****CONTOUR IMAGE

**RESULTS**

#no. of characters in the txt file psnr value

#1,218 54.868

#3,654 51.226

#7,308‬ 48.932

#12,180 47.2463

#18,270‬ 45.1648

#24,360 43.5397

**EDGEENHANCE OF THE IMAGE**

****ORIGINAL IMAGE****EDGEENHANCED IMAGE

**RESULTS**

#no. of characters in the txt file psnr value

#1,218 56.584

#3,654 51.027

#7,308‬ 48.436

#12,180 46.7578

#18,270 44.4098

#24,360 42.8495

**EMBOSS OF THE IMAGE**

****ORIGINAL IMAGE****EMBOSS IMAGE

**RESULTS**

#no. of characters in the txt file psnr value

#1,218 57.607

#3,654 54.440

#7,308‬ 52.307

#12,180 50.5942

#18,270 47.9808

#24,360 45.7946

**GAUSSIAN OF THE IMAGE**

****ORIGINAL IMAGE****GUASSIAN BLURR IMAGE

**RESULTS**

#no. of characters in the txt file psnr value

#1,218 59.9855

#3,654 55.206

#7,308‬ 52.765

#12,180 51.0688

#18,270 48.6405

#24,360 47.1403

**LAPLACIAN OF THE IMAGE**

****ORIGINAL IMAGE****LAPLACIAN IMAGE

**RESULTS**

#no. of characters in the txt file psnr value

#1,218 58.185

#3,654 53.727

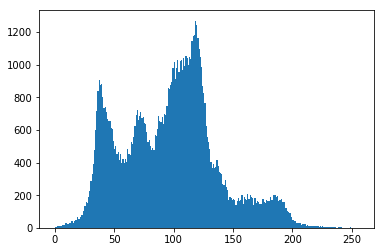
#7,308‬ 50.822

#12,180 48.8420

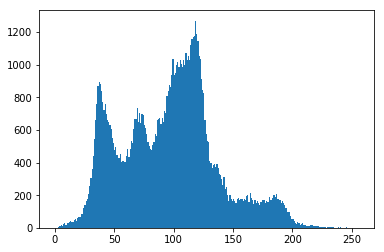
#18,270 46.3883

#24,360 44.14066

**HISTOGRAM OF THE IMAGE**

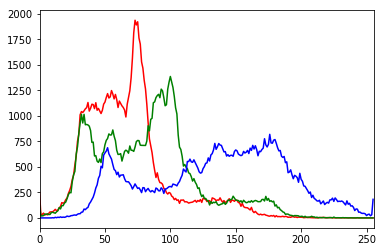
****

COVER IMAGE

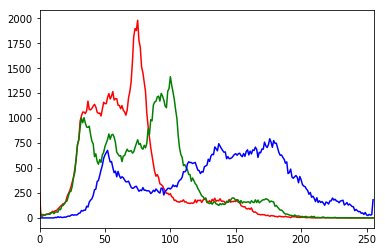


STEGO IMAGE

**HISTOGRAM OF IMAGE IN RGB FORM**

****

COVER IMAGE



STEGO IMAGE

**LOG TRANSFORM AND GAMMA CORRECTION**

**NEGATIVE OF THE IMAGE**

****COVER IMAGE

NEGATIVE IMAGE

**LOG TRANSFORM**

****COVER IMAGE

LOG TRANSFORM

**GAMMA CORRECTION**

****COVER IMAGE

****GAMMA CORRECTED IMAGE

**CONCLUSIONS**

A new and efficient computer based steganographic method for embedding secret messages into images without producing noticeable changes has been proposed. There is no need of referencing the original image when extracting the embedded data from a stego image. The method utilizes the characteristic of the human vision's senstivity to gray value variations. Secret data are embedded into a cover image by replacing the difference values of the two pixel blocks of the cover image with similar ones in which bits of embedded data are included.

The method not only provides a better way for embedding large amounts of data into cover images with imperception , but also offers an easy way to accomplish secrecy. This embedding method can be easily extended to efficiently carry content related messages such as captions or annotations in audios and videos by embedding data in each adjacent pair of signals of the data streams.

**REFERENCES**

1. Research paper : A steganography method for images by pixel value differencing.(by Da Chun Wu Wen Hsiang Tsai)
2. Research Paper : Colour Image Steganography based on pixel value diffrencing in spatial domain.(by J.K.MANDAL AND DEBASHIS DAS)
3. <https://www.geeksforgeeks.org/python-pil-image-filter-with-imagefilter-module/>
4. <https://docs.opencv.org/3.4/d5/db5/tutorial_laplace_operator.html>
5. <https://www.geeksforgeeks.org/opencv-python-program-analyze-image-using-histogram/>
6. <https://www.geeksforgeeks.org/python-intensity-transformation-operations-on-images/>
7. <https://www.geeksforgeeks.org/python-peak-signal-to-noise-ratio-psnr/>