**SVKM’s NMIMS**

**Mukesh Patel School of Technology Management & Engineering**

**Department of Electronics and Telecommunication Engineering**

**Subject: Image and Video Processing Program: B.Tech/BTI/MBA**

**Sem: VII/IX/V ACAY: 2020-21**

**EXPERIMENT NO. 7**

**Aim:**

To write a program in PYTHON to implement Hadamard transform on an image

**Software:**  PYTHON.

**Prerequisite:**

|  |  |
| --- | --- |
| Sr. No | Concepts |
| 1. | Transforms- Hadamard |

**Outcome:**

After successful completion of this experiment students will be able to:

1. Understand the significance of transform in image processing
2. Implement forward and inverse Hadamard transform on a 2D matrix

**Theory:**

If is the transformation matrix and is its transpose, forward transform of any 2D matrix is given by

Similarly, inverse transform is given by

**Hadamard Transform**

Hadamard matrix of order 2 is given by

Hadamard Matrix higher order 4 is given by

|  |
| --- |
| Name of the Experiment: To implement Hadamard Transform on an image. |
| Roll No. C018 Name: Dhruvit Jain |
| Program: B.Tech ExTC Semester : VII |
| Date of Performance:28/08/2020 Date of Submission: 28/08/2020 |

**CODE:**

from scipy.linalg import hadamard

import numpy as np

import random

# generate Hadamard matrix of order N

N = int (input('Enter the order of Hadamard matrix: '))

H = hadamard(N)

print(H)

#original matrix

mat = np.random.randint(9, size=(N,N))

mat

#transposed image F = H\*f\*Ht

temp = np.dot(H,mat)

trans\_mat = np.dot(temp, np.transpose(H))

trans\_mat

#original image f = Ht\*F\*H

temp2 = np.dot(np.transpose(H),trans\_mat)

og\_image = np.dot(temp2,H)

og\_image = og\_image//(N\*N)

og\_image

#Hadamar transform for image

import numpy as np

from skimage import io

import matplotlib.pyplot as plt

from scipy.linalg import hadamard

from skimage.color import rgb2gray

def had2 (f):

H= hadamard (8)

F = np.dot(np.dot(H,f),np.transpose(H))

return F

def had2inv (F):

H= hadamard (8)

f = (np.dot(np.dot(np.transpose(H),F),H))//8

return f

image\_ori=io.imread('roof.tif')

image=rgb2gray(image\_ori)

sh= image.shape

row = sh[0]

col = sh[1]

#hadmard of image

image\_hada = np.zeros ((row,col),dtype=int)

for r in range (0,row//8):

for c in range (0,col//8):

image\_hada[r\*8:(r+1)\*8, c\*8:(c+1)\*8]=had2(image[r\*8:(r+1)\*8, c\*8:(c+1)\*8])

#inv hadmard of image

image\_inv = np.zeros ((row,col),dtype=int)

for r in range (0,row//8):

for c in range (0,col//8):

image\_inv[r\*8:(r+1)\*8, c\*8:(c+1)\*8]=had2(image\_hada[r\*8:(r+1)\*8, c\*8:(c+1)\*8])

plt.figure(figsize=(12,12))

plt.subplot (1,3,1)

io.imshow (image\_ori,cmap='gray')

plt.title ("Original Image")

plt.subplot (1,3,2)

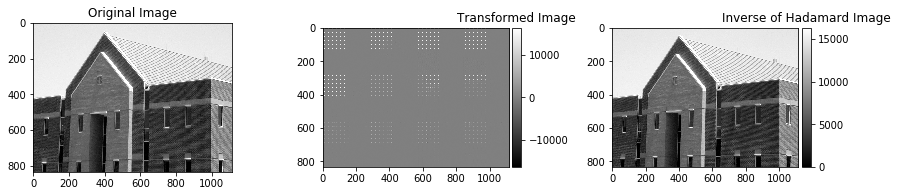
io.imshow (image\_hada,cmap='gray')

plt.title ("Transformed Image")

plt.subplot (1,3,3)

io.imshow (image\_inv,cmap='gray')

plt.title ("Inverse of Hadamard Image")



**CONCLUSIONS:**

1. Elements of Hadamard transform matrix are either +1 or -1.

2. Hadamar matrix is symetric and has real values.

3. It can be applied on matrices of sizes which are multiples of two.

4. It is observed that original image and inverse of transformed image is same, and hence, hadamard matrix can be used for signal processing.