Lab04-Image Processing and Analysis

19110522 – Bùi Thị Thanh Xuân

I/ Nội dung bài làm:

Làm đầy đủ các yêu cầu đề bài cho.

II/ Làm thêm:

III/ Bài làm:

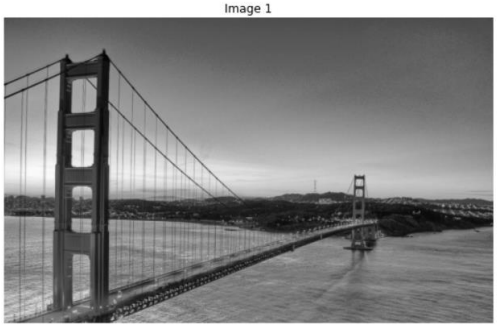
Entrée [1]:
```
import numpy as np
import pandas as pd
import cv2
from matplotlib import pyplot as plt
from pylab import imread
from skimage.color import rgb2gray
```

Entrée [2]:
```
def ShowImage(ImageList, nRows = 1, nCols = 2, WidthSpace = 0.00, HeightSpace = 0.00):
    from matplotlib import pyplot as plt
    import matplotlib.gridspec as gridspec

    gs = gridspec.GridSpec(nRows, nCols)
    gs.update(wspace=WidthSpace, hspace=HeightSpace) # set the spacing between axes.
    plt.figure(figsize=(20,10))
    for i in range(len(ImageList)):
        ax1 = plt.subplot(gs[i])
        ax1.set_xticklabels([])
        ax1.set_yticklabels([])
        ax1.set_aspect('equal')

        plt.subplot(nRows, nCols,i+1)

        image = ImageList[i].copy()
        if (len(image.shape) < 3):
            plt.imshow(image, plt.cm.gray)
        else:
            plt.imshow(image)
        plt.title("Image " + str(i))
        plt.axis('off')

    plt.show()
```

1/ Kiếm một ảnh trên internet và thực hiện các bước sau : ¶

Chuyển đổi thành ảnh xám

Entrée [3]:
```
# Read Image
image_color = imread("sample01.jpg")
# Convert Image into Gray
image_gray = cv2.cvtColor(image_color, cv2.COLOR_RGB2GRAY)

# Display Image
ShowImage([image_color, image_gray], 1, 2)
```



Biến đổi DFT và hiển thị ảnh ở miền tần số

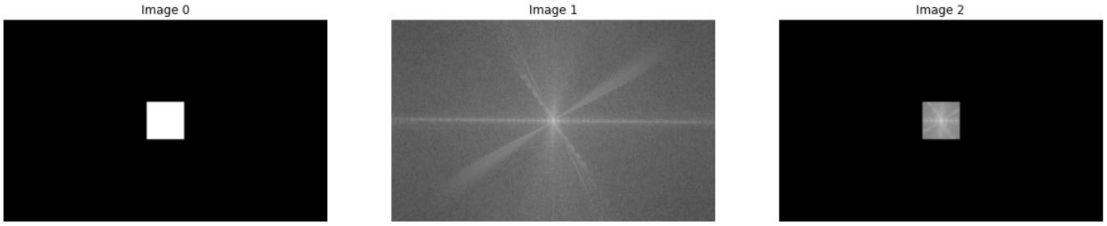
Entrée [4]:
```
def Image3Dto2D(image):
    if(len(image.shape) >= 3):
        image_2D = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
    else:
        image_2D = image.copy()
    return image_2D
```

Entrée [5]:
```
def DFT_Transformation(image):
    img = Image3Dto2D(image)

    img_float32 = np.float32(img)
    dft = cv2.dft(img_float32, flags = cv2.DFT_COMPLEX_OUTPUT)
    dft_shift = np.fft.fftshift(dft)
    magnitude_spectrum = 20*np.log(cv2.magnitude(dft_shift[:,:,0],dft_shift[:,:,1]))

    return magnitude_spectrum, dft_shift
```

Entrée [6]:
```
image_dft_frequency, dft_shift = DFT_Transformation(image_gray)
image_dft_frequency1, dft_shift1 = DFT_Transformation(image_color[:,:,0])
image_dft_frequency2, dft_shift2 = DFT_Transformation(image_color[:,:,1])
image_dft_frequency3, dft_shift3 = DFT_Transformation(image_color[:,:,2])
ShowImage([image_color, image_gray,  image_dft_frequency, image_dft_frequency1, image_dft_frequency2, image_dft_frequency3], 2, 3
```

Thực hiện tạo ảnh mask để bỏ miền tần số cao và hiển thị ảnh texture

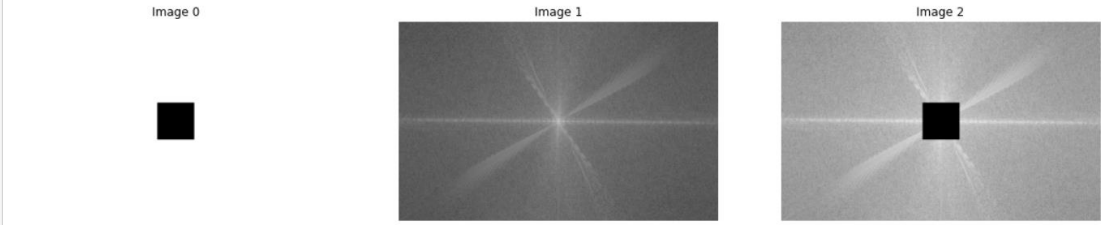
Entrée [7]:
```
rows, cols = image_gray.shape
crow,ccol = (int)(rows/2) , (int)(cols/2)

# create a mask first, center square is 1, remaining all zeros
mask = np.zeros((rows, cols, 2), np.uint8)
size = 50
mask[crow-size:crow+size, ccol-size:ccol+size] = 1
image_dft_frequency_crop = image_dft_frequency* mask[:,:,0]

# apply mask and inverse DFT
fshift = dft_shift*mask
f_ishift = np.fft.ifftshift(fshift)
img_inverse = cv2.idft(f_ishift)
image_inverse = cv2.magnitude(img_inverse[:,:,0],img_inverse[:,:,1])

ShowImage([mask[:,:,0], image_dft_frequency, image_dft_frequency_crop], 1, 3)
ShowImage([image_gray, image_inverse], 1, 2)
```



Thực hiện tạo ảnh mask để bỏ miền tần số thấp và hiển thị ảnh edge

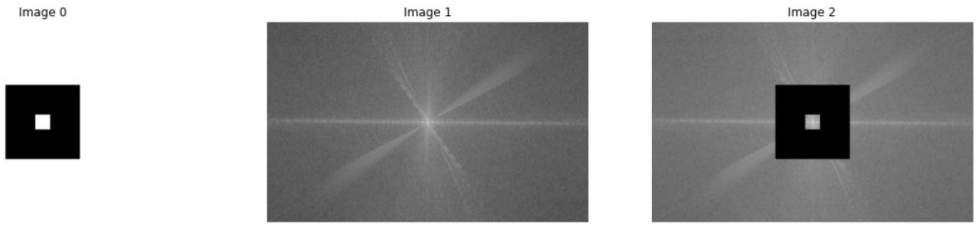
Entrée [8]:

```
rows, cols = image_gray.shape
crow,ccol = (int)(rows/2) , (int)(cols/2)

# create a mask first, center square is 1, remaining all zeros
mask = np.zeros((rows, cols, 2), np.uint8)
size = 50
mask[crow-size:crow+size, ccol-size:ccol+size] = 1
mask = 1 - mask
image_dft_frequency_crop = image_dft_frequency* mask[:,:,0]

# apply mask and inverse DFT
fshift = dft_shift*mask
f_ishift = np.fft.ifftshift(fshift)
img_inverse = cv2.idft(f_ishift)
image_inverse = cv2.magnitude(img_inverse[:,:,0],img_inverse[:,:,1])

ShowImage([mask[:,:,0], image_dft_frequency, image_dft_frequency_crop], 1, 3)
ShowImage([image_gray, image_inverse], 1, 2)
```



Thực hiện tạo ảnh mask như sau và thực hiện thay đổi miền tần số sau đó xuất ảnh kết quả

Entrée [9]:
```
rows, cols = image_gray.shape
crow,ccol = (int)(rows/2) , (int)(cols/2)

# create a mask first, center square is 1, remaining all zeros
mask1 = np.zeros((rows, cols, 2), np.uint8)
mask2 = np.zeros((rows, cols, 2), np.uint8)

size = 20
mask1[crow-size:crow+size, ccol-size:ccol+size] = 1

size = 100
mask2[crow-size:crow+size, ccol-size:ccol+size] = 1


mask = np.zeros((rows, cols, 2), np.uint8)
mask[(mask1 == 1) | (mask2 == 0)] = 1

image_dft_frequency_crop = image_dft_frequency* mask[:,:,0]

# apply mask and inverse DFT
fshift = dft_shift*mask
f_ishift = np.fft.ifftshift(fshift)
img_inverse = cv2.idft(f_ishift)
image_inverse = cv2.magnitude(img_inverse[:,:,0],img_inverse[:,:,1])

ShowImage([mask[:,:,0], image_dft_frequency, image_dft_frequency_crop], 1, 3)
ShowImage([image_gray, image_inverse], 1, 2)
```



2/ Kiếm một ảnh trên internet và thực hiện các bước sau:

Chuyển đổi thành ảnh xám

Entrée [10]:
```
# Read Image
image_color = imread("Sample02.jpg")
# Convert Image into Gray
image_gray = cv2.cvtColor(image_color, cv2.COLOR_RGB2GRAY)

# Display Image
ShowImage([image_color, image_gray], 1, 2)
```

Image 0

Image 1

Biến đổi DCT và hiển thị ảnh DCT

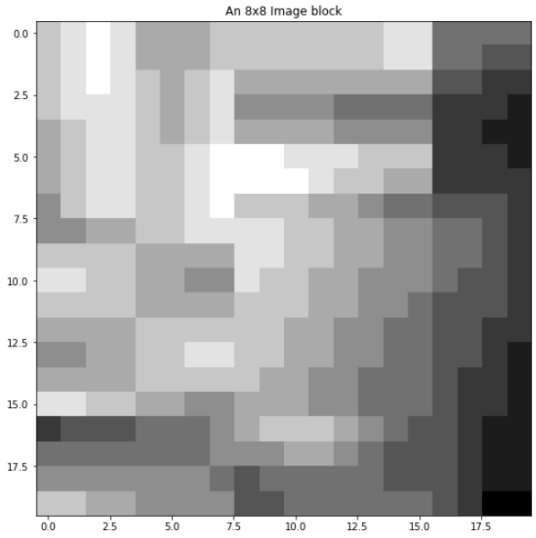
Entrée [29]:
```
import scipy.fftpack
from matplotlib import cm
```

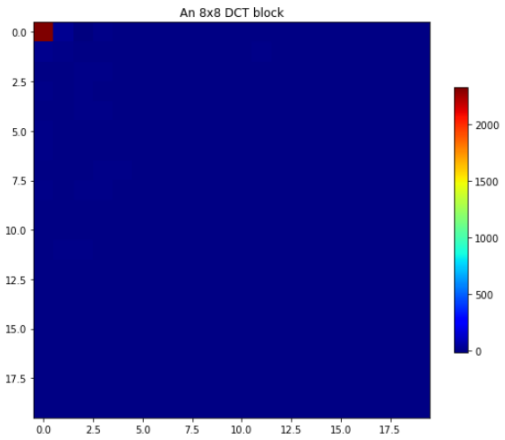
Entrée [26]:
```
def dct2(a):
    import scipy
    return scipy.fftpack.dct( scipy.fftpack.dct( a, axis=0, norm='ortho' ), axis=1, norm='ortho' )

def idct2(a):
    import scipy
    return scipy.fftpack.idct( scipy.fftpack.idct( a, axis=0 , norm='ortho'), axis=1 , norm='ortho')
```

Entrée [27]:
```
im = image_gray
imsize = im.shape
dct = np.zeros(imsize)

# Do 8x8 DCT on image (in-place)
for i in np.r_[:imsize[0]:8]:
    for j in np.r_[:imsize[1]:8]:
        dct[i:(i+8),j:(j+8)] = dct2( im[i:(i+8),j:(j+8)] )
```

Entrée [30]:
```
pos = 128
SampleBlock = im[pos:pos+20,pos:pos+20]
SampleDCT = dct2(SampleBlock)

plt.figure(figsize=(20,10))
plt.subplot(1,2,1)
plt.imshow(SampleBlock,cmap='gray')
plt.title( "An 8x8 Image block")

# Display the dct of that block
plt.subplot(1,2,2)
plt.imshow(SampleDCT,cmap=cm.jet,interpolation='nearest')
plt.colorbar(shrink=0.5)
plt.title( "An 8x8 DCT block")
```

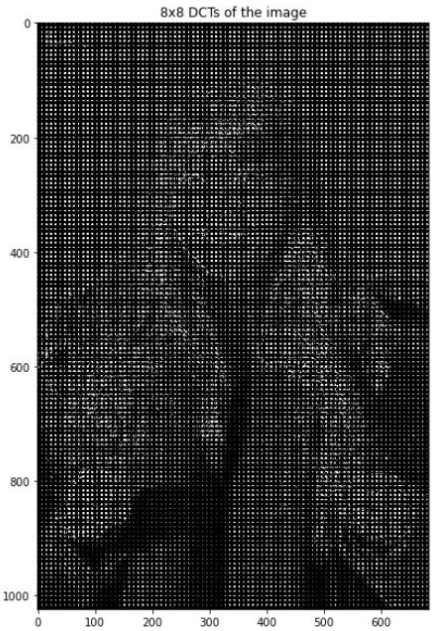
Out[30]: Text(0.5, 1.0, 'An 8x8 DCT block')





Entrée [31]:
```
# Display entire DCT
plt.figure(figsize=(20,10))
plt.subplot(1,2,1)
plt.imshow(image_gray, cmap = 'gray')
plt.title( "Original Image")
plt.subplot(1,2,2)
plt.imshow(dct,cmap='gray',vmax = np.max(dct)*0.01,vmin = 0)
plt.title( "8x8 DCTs of the image")
```

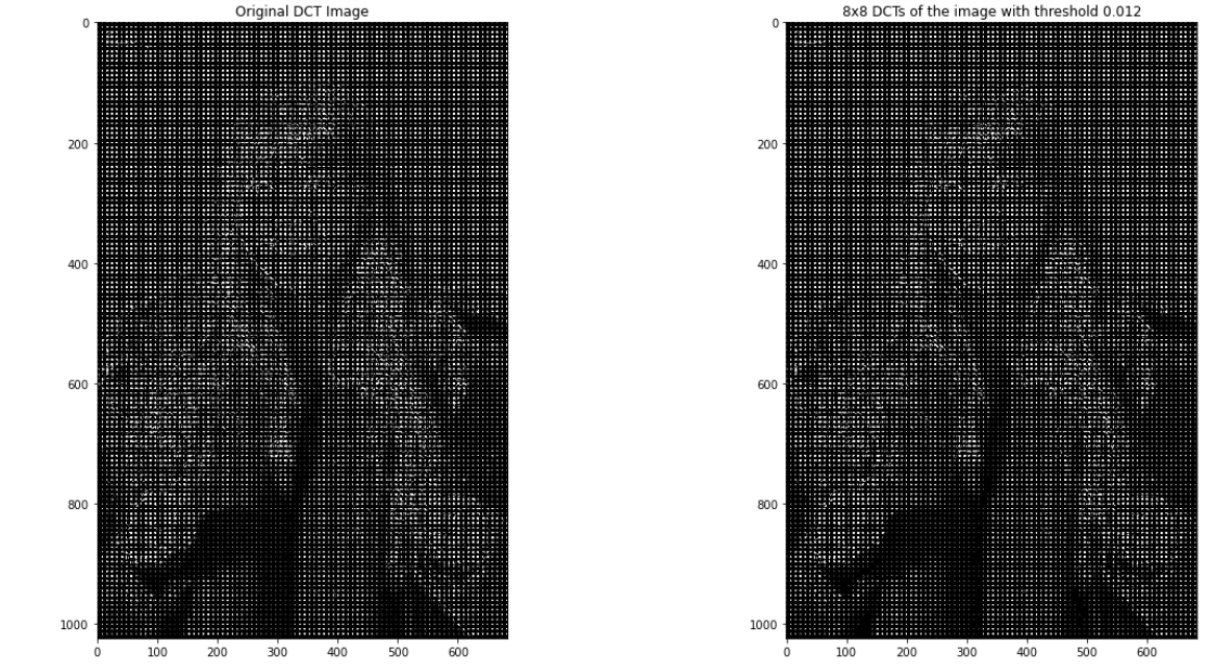
Out[31]: Text(0.5, 1.0, '8x8 DCTs of the image')



Đặt ngưỡng giữ lại khoảng 5% hệ số DCT và hiển thị ảnh nén kết quả

Entrée [32]:
```
# Threshold
thresh = 0.012
dct_thresh = dct * (abs(dct) > (thresh*np.max(dct)))

plt.figure(figsize=(20,10))
plt.subplot(1,2,1)
plt.imshow(dct,cmap='gray',vmax = np.max(dct)*0.01,vmin = 0)
plt.title( "Original DCT Image")
plt.subplot(1,2,2)
plt.imshow(dct_thresh,cmap='gray',vmax = np.max(dct)*0.01,vmin = 0)
plt.title( "8x8 DCTs of the image with threshold " + str(thresh))


percent_nonzeros = np.sum( dct_thresh != 0.0 ) / (imsize[0]*imsize[1]*1.0)
print("Keeping only %f%% of the DCT coefficients" % (percent_nonzeros*100.0))

im_dct = np.zeros(imsize)
for i in np.r_[:imsize[0]:8]:
    for j in np.r_[:imsize[1]:8]:
        im_dct[i:(i+8),j:(j+8)] = idct2( dct_thresh[i:(i+8),j:(j+8)] )

print("Comparison between original and DCT compressed images" )
ShowImage([im, im_dct])
```

```
Keeping only 4.467023% of the DCT coefficients
Comparison between original and DCT compressed images
```



3/ Kiếm một ảnh màu trên internet và thực hiện nén các ảnh này dùng DFT và DCT bằng cách giữ lại 5% hệ số. Sau đó khôi phục lại ảnh màu và lưu xuống thư mục cũng như xuất dung lượng ảnh xem giảm được bao nhiêu dung lượng (Gợi ý dùng không gian HSV nén V nhưng vẫn giữ được màu, thử nén H và S xem sao)

Entrée [17]:
```
image_color = imread("Sample02.jpg")
image_hsv = cv2.cvtColor(image_color, cv2.COLOR_BGR2HSV)
```

Entrée [18]:
```
image_dft_frequency_v, dft_shift_v = DFT_Transformation(image_hsv[:,:,2])
```

Dùng DFT bằng cách giữ lại 5% hệ số

Entrée [19]:
```
dft = np.zeros(imsize,dtype='complex');
im_dft = np.zeros(imsize,dtype='complex');

# 8x8 DFT
for i in np.r_[:imsize[0]:8]:
    for j in np.r_[:imsize[1]:8]:
        dft[i:(i+8),j:(j+8)] = np.fft.fft2( im[i:(i+8),j:(j+8)] )

# Thresh
thresh = 0.013
dft_thresh = dft * (abs(dft) > (thresh*np.max(abs(dft))))


percent_nonzeros_dft = np.sum( dft_thresh != 0.0 ) / (imsize[0]*imsize[1]*1.0)
print("Keeping only %f%% of the DCT coefficients" % (percent_nonzeros*100.0))
print("Keeping only %f%% of the DFT coefficients" % (percent_nonzeros_dft*100.0))

# 8x8 iDFT
for i in np.r_[:imsize[0]:8]:
    for j in np.r_[:imsize[1]:8]:
        im_dft[i:(i+8),j:(j+8)] = np.fft.ifft2( dft_thresh[i:(i+8),j:(j+8)] )

print("Comparison between original, DCT compressed and DFT compressed images")
ShowImage([im, im_dct, abs(im_dft)], 1, 3)
```

```
Keeping only 4.467023% of the DCT coefficients
Keeping only 4.868657% of the DFT coefficients
Comparison between original, DCT compressed and DFT compressed images
```

Keeping only 4.467023% of the DCT coefficients
Keeping only 4.868657% of the DFT coefficients
Comparison between original, DCT compressed and DFT compressed images

Entrée [20]:
```
rows, cols = image_gray.shape
crow,ccol = (int)(rows/2) , (int)(cols/2)

# create a mask first, center square is 1, remaining all zeros
mask = np.zeros((rows, cols, 2), np.uint8)
size = 50
mask[crow-size:crow+size, ccol-size:ccol+size] = 1
image_dft_frequency_crop = image_dft_frequency_v* mask[:,:,0]

# apply mask and inverse DFT
fshift = dft_shift_v*mask
f_ishift = np.fft.ifftshift(fshift)
img_inverse = cv2.idft(f_ishift)
image_inverse = cv2.magnitude(img_inverse[:,:,0],img_inverse[:,:,1])
```

Sau đó khôi phục lại ảnh màu

Entrée [21]:
```
image_hsv[:,:,2]=image_inverse
image_hsv=cv2.cvtColor(image_hsv, cv2.COLOR_HSV2RGB)
ShowImage([image_hsv])
```

Image 0

lưu xuống thư mục cũng như xuất dung lượng ảnh xem giảm được bao nhiêu dung lượng

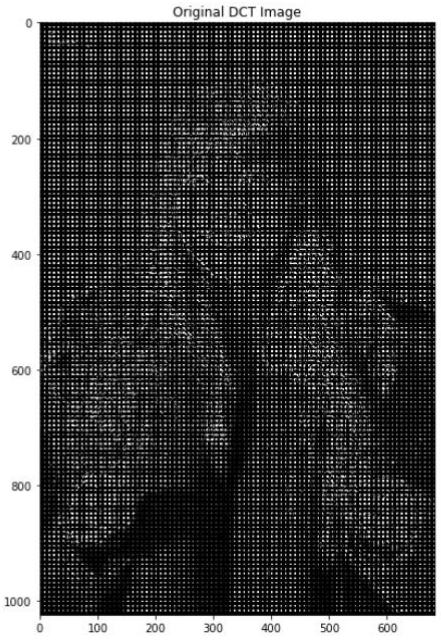
Entrée [22]:
```
cv2.imwrite(f'image_dft_v.jpg',image_hsv)
#Xuất kích thước
import os
print ("Kích ảnh khi nén kênh v là:",os.stat('image_dft_v.jpg').st_size,"byte")
```

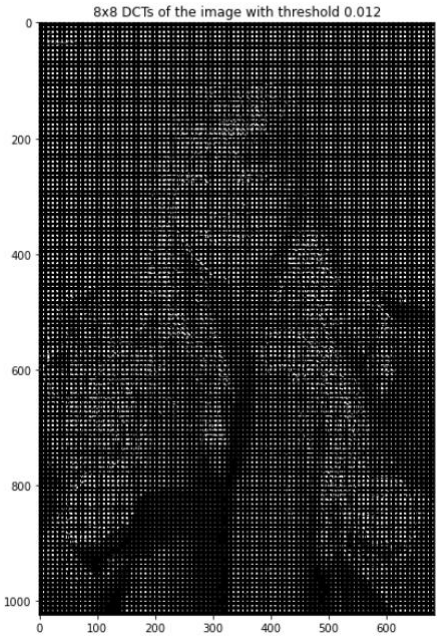
```
Kích ảnh khi nén kênh v là: 739521 byte
```

Dùng DCT bằng cách giữ lại 5% hệ số

Entrée [23]:
```
# Threshold
thresh = 0.012
dct_thresh = dct * (abs(dct) > (thresh*np.max(dct)))

plt.figure(figsize=(20,10))
plt.subplot(1,2,1)
plt.imshow(dct,cmap='gray',vmax = np.max(dct)*0.01,vmin = 0)
plt.title( "Original DCT Image")
plt.subplot(1,2,2)
plt.imshow(dct_thresh,cmap='gray',vmax = np.max(dct)*0.01,vmin = 0)
plt.title( "8x8 DCTs of the image with threshold " + str(thresh))


percent_nonzeros = np.sum( dct_thresh != 0.0 ) / (imsize[0]*imsize[1]*1.0)
print("Keeping only %f%% of the DCT coefficients" % (percent_nonzeros*100.0))

im_dct = np.zeros(imsize)
for i in np.r_[:imsize[0]:8]:
    for j in np.r_[:imsize[1]:8]:
        im_dct[i:(i+8),j:(j+8)] = idct2( dct_thresh[i:(i+8),j:(j+8)] )

print("Comparison between original and DCT compressed images" )
ShowImage([im, im_dct])
```

```
Keeping only 4.467023% of the DCT coefficients
Comparison between original and DCT compressed images
```



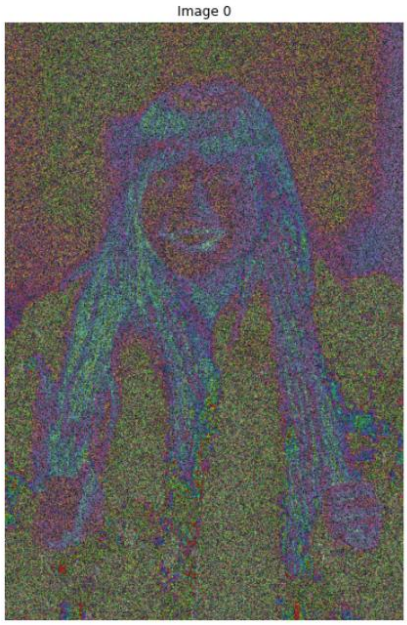


Entrée [24]:
```
im = image_hsv[:,:,2]
imsize = im.shape
dct = np.zeros(imsize)

# Do 8x8 DCT on image (in-place)
for i in np.r_[:imsize[0]:8]:
    for j in np.r_[:imsize[1]:8]:
        dct[i:(i+8),j:(j+8)] = dct2( im[i:(i+8),j:(j+8)] )
```

Khôi Phục lại ảnh màu

Entrée [25]:
```
image_hsv[:,:,2]=image_inverse
image_hsv=cv2.cvtColor(image_hsv, cv2.COLOR_HSV2RGB)
ShowImage([image_hsv])
```

Entrée [25]:
```
image_hsv[:,:,2]=image_inverse
image_hsv=cv2.cvtColor(image_hsv, cv2.COLOR_HSV2RGB)
ShowImage([image_hsv])
```



Entrée [26]:
```
cv2.imwrite(f'image_dct_v.jpg',image_hsv)
#Xuất kích thước
import os
print ("Kích ảnh khi nén kênh v là:",os.stat('image_dft_v.jpg').st_size,"byte")
```

```
Kích ảnh khi nén kênh v là: 739521 byte
```

Trong Thư mục có: