Image and Video Processing Lab
Lab 7:Image Transforms

Aruna Shaju Kollannur (SC21M111)

Sub: Image and Video Processing Lab
Date of Lab sheet: November 30, 2021
Department: Avionics(DSP)
Date of Submission: December 21, 2021

Question 1 Transforming image using DCT and performing reconstruction.

Aim

Transforming an image using Discrete Cosine Transform and reconstructing image from transformed image

Discussion

Transform-based compression systems are based on the insight that the de-correlated coefficients of a transform can be coded more efficiently than the original image pixels. The transform typically results in some energy compaction (i.e., an energy redistribution of the original image into a smaller set of coefficients). Even though the energy is compacted into fewer coefficients, the total energy is conserved, resulting in a significant number of coefficients with values of zero or near zero.
The discrete cosine transform (DCT) helps separate the image into parts (or spectral sub-bands) of differing importance (with respect to the image's visual quality). The Trnasformation is representd as:

$$\mathrm{X}_{k_1,k_2} = \sum_{k_1=0}^{N_1-1} \left(\sum_{k_2=0}^{N_2-1} x_{n_1,n_2} cos \left[\frac{\pi}{N_2} \left(n_2 + \frac{1}{2} \right) k_2 \right] \right) cos \left[\frac{\pi}{N_1} \left(n_1 + \frac{1}{2} \right) k_1 \right]$$

2D DCT transformed image can be represented in matrix form as :

$$\mathrm{x} = \mathrm{C}\ \mathrm{XC}^T$$

The Reconstructed Image can be obtained by :

$$\mathrm{x} = \mathrm{C}^T XC$$

Algorithm

```
Step 1: Start
Step 2: Declare the size of DCT transform matrix
Step 3: Obtain the number of DCT computations required row-wise and column-wise.
Step 4: Declare zero matrix to contain transformed DCT image.
Step 5: Apply DCT on each block of image.
Step 6: Display transformed image.
Step 7: Appy IDCT on transformed image.
Step 8: Display Reconstructed Image
```

Program Code

```
import cv2
import numpy as np

# Blocksize of DCT matrix used, thus image divided into 8*8 sizes to
#perform transformation
B = 8
Image_Cameraman = cv2 . imread ( "Cameraman.png",0)
h ,w= np . array ( Image_Cameraman . shape [:2]) /B * B
h= int (h );w= int (w) ;
Image_Cameraman = Image_Cameraman [:h ,: w]

Vertical_blocks =int (h/B)
Horizontal_blocks =int (w/B)
Image_Temp = np . zeros ((h ,w) , np . float32 )
Image_DCT = np . zeros ((h ,w ) , np . float32 )
Image_Temp [:h , : w] = Image_Cameraman

# Applying DCT Blockwise

for row in range ( Vertical_blocks ):
 for col in range ( Horizontal_blocks ):
   Current_Block = cv2 . dct ( Image_Temp [ row *B :( row +1) *B , col *B :( col +1) *B ])
   Image_DCT [ row *B :( row +1) *B , col *B :( col +1) *B ]= Current_Block

cv2.imwrite ('DCT Transformed.png', Image_DCT )
back0 = np . zeros ((h ,w ) , np . float32 )

# Applying IDCT Blockwise to get reconstructed image
for row in range ( Vertical_blocks ):
  for col in range ( Horizontal_blocks ):
    Current_Block = cv2 . idct ( Image_DCT [ row *B :( row +1) *B , col *B :( col +1) * B ])
    back0 [ row *B :( row +1) *B , col *B :( col +1) *B ]= Current_Block

cv2.imwrite("DCT_Reconstructed_Image.jpg", back0 )
```

Result

Figure 1: Cameraman : (a)Original Image (b) DCT Transformed Image (c) Reconstructed Image

Program Code - Without Inbuilt Function

```
from PIL import Image, ImageOps
import numpy as np

Cameraman_Image = Image.open("Cameraman.png")
Cameraman_gray =  ImageOps.grayscale(Cameraman_Image)
Image_copy = Cameraman_gray.copy()
Image_DCT =  np.array(Cameraman_gray.copy())
Image_Reconstructed =  np.array(Cameraman_gray.copy())

height, width = Cameraman_gray.size
L = 8
C = np.zeros((L,L))
for k in range(L):
    for n in range(L):
        if k == 0:
            C[k,n] = np.sqrt(1/L)
        else:
            C[k,n] = np.sqrt(2/L)*np.cos((np.pi*k*(1/2+n))/L)

for i in range(0,height - L + 1,L):
    for j in range(0,width -L+ 1, L):

        submatrix = np.array(Image_copy)[i:i+L,j:j+L];
        temp = C.dot(submatrix)
        temp1 = temp.dot(C.transpose())

        Image_DCT[i:i+L,j:j+L] = temp1

for i in range(0,height - L + 1,L):
    for j in range(0,width -L+ 1, L):
        submatrix = np.array(Image_DCT)[i:i+L,j:j+L];
        temp = (C.transpose()).dot(submatrix)
        temp1 = temp.dot(C)
        Image_Reconstructed[i:i+L,j:j+L] = temp1
```

Result - Without Inbuilt Function

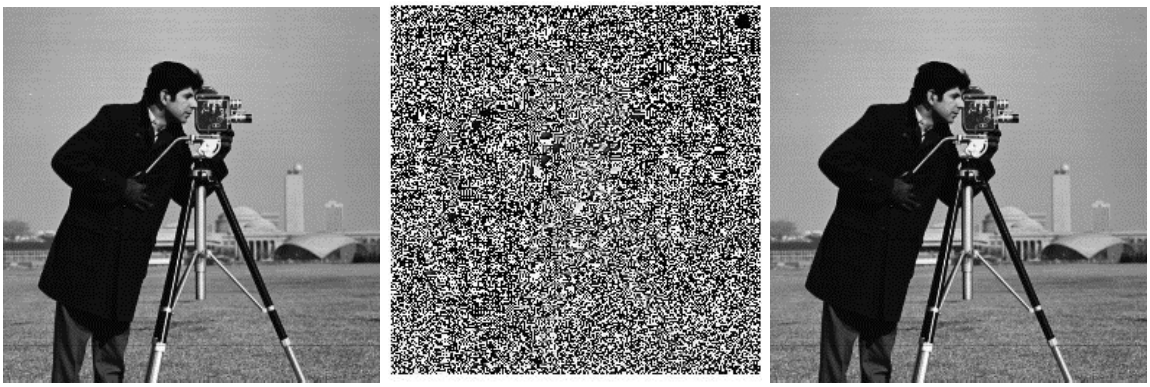


Figure 2: Cameraman : (a)Original Image (b) DCT Transformed Image (c) Reconstructed Image

Inference

DCT transformed image can be reconstructed using Basis matrices.The transformation is orthogonal (inverse is transpose and energy is preserved), fast algorithms can be used for computation, and the output for (near) constant matrices generally consists of a large number of (near) zero values.

Question 2: Transforming image using Haar Transform and performing re-construction.

Aim

Transforming an image using Haar Transform and reconstructing image from transformed image

Discussion

The Harr wavelet transform decomposes the original image into a sum of spatially and frequency localized functions, in a way that is similar to subband decomposition. The most important visual information tends to be concentrated into a reduced number of components (coefficients); therefore, the remaining coefficients can be quantized coarsely or truncated to zero with little image distortion. Compression methods based on wavelets avoid the block artifacts that occur in compression methods based on DCT. This is one of the reasons why wavelet-based compression schemes tend to produce superior image quality.

Algorithm

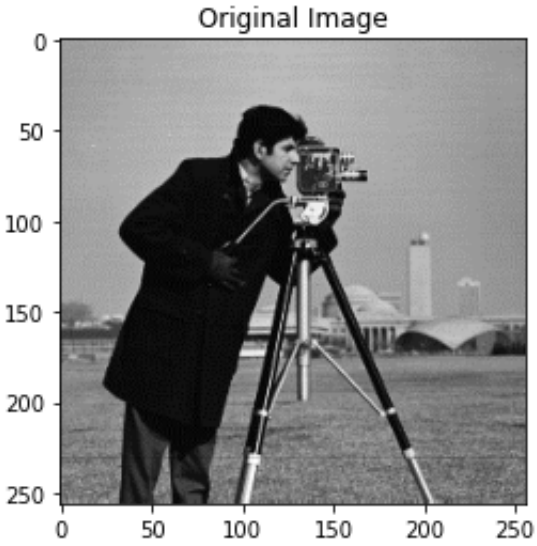
```
Step 1: Start
Step 2: Read the image
Step 3: Obtain transformed image in the form of Haar components using function.
Step 4: Display the components.
Step 5: Apply inverse Haar transform on obtained Haar
        components in Step 3 to get reconstructed image.
Step 6: Display Reconstructed Image
```

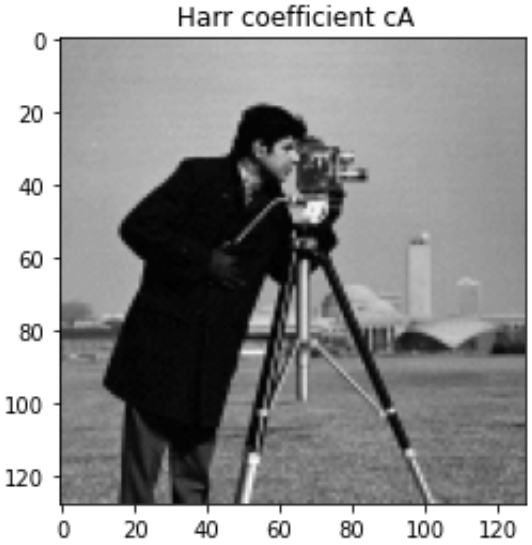
Program Code

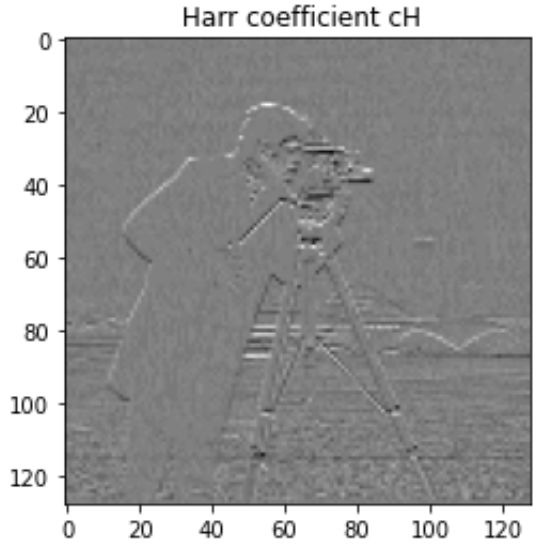
```
import cv2
from matplotlib import pyplot as plt
from pywt import dwt2 , idwt2

img = cv2 . imread ( "Cameraman.png",0)
cA , (cH , cV , cD ) = dwt2 ( img , 'haar')

plt . figure ()
plt . imshow (cA , cmap = "gray")
plt . title (" Harr coefficient cA")

plt . figure ()
plt . imshow (cH , cmap = "gray")
plt . title (" Harr coefficient cH")

plt . figure ()
plt . imshow (cV , cmap = "gray")
plt . title (" Harr coefficient cV")

plt . figure ()
```

```
plt . imshow (cD , cmap = "gray")
plt . title (" Harr coefficient cD")

plt . figure ()
plt . imshow ( img , cmap = "gray")
plt . title (" Original Image ")

recovered_image = idwt2 (( cA , (cH , cV , cD )) , "haar")

plt . figure ()
plt . imshow ( recovered_image , cmap = "gray")
plt . title (" Reconstructed Image ")
```

Result







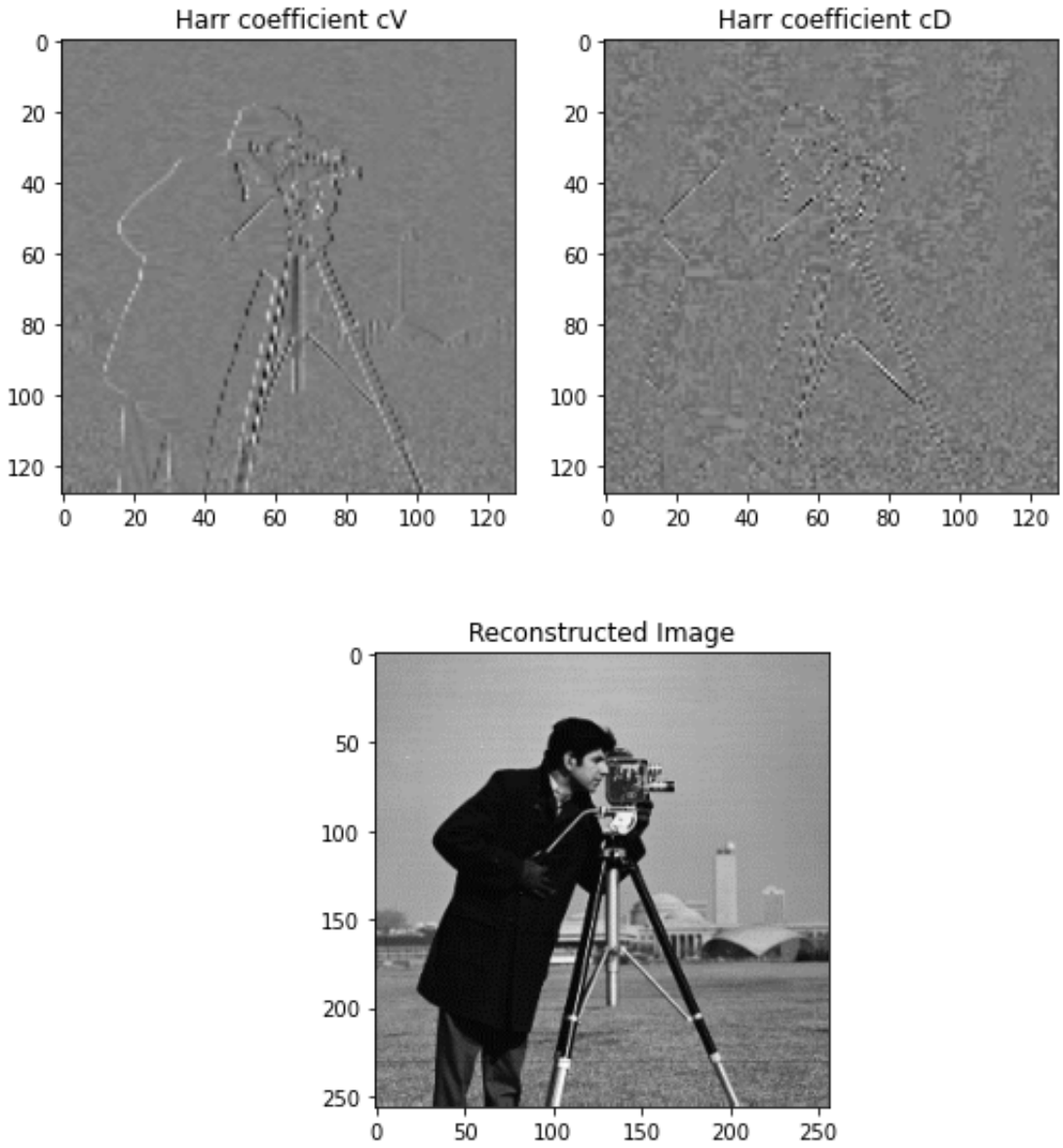


Figure 3: (a)Original Image (b) Haar Components (c) Reconstructed Image

Inference

Haar transform does not apply multiplications. It requires only additions and there are many elements with zero value in the Haar matrix, so the computation time is short. It can be used to analyse the localized feature of signals. Due to the orthogonal property of the Haar function, the frequency components of input signal can be analyzed.

Question 3 : Transforming image using Hadamard Transform and performing re-construction.

Aim

Transforming an image using Hadamard Transform and reconstructing image from transformed image.

Discussion

The Hadamard transform is an example of a generalized class of Fourier transforms. It performs an orthogonal, symmetric, involutive, linear operation on 2m real numbers (or complex, or hypercomplex numbers, although the Hadamard matrices themselves are purely real). The Hadamard transform can be regarded as being built out of size-2 discrete Fourier transforms (DFTs), and is in fact equivalent to a multidimensional DFT of size $2 \times 2 \times \times 2 \times 2$.It decomposes an arbitrary input vector into a superposition of Walsh functions.

Algorithm

```
Step 1: Start
Step 2: Read the image
Step 3: Resize image into a square form with new dimension being the height of original image.
Step 4: Create Hadamard transform of size height of image.
Step 5: Perform Hadamard transform on image using Hadamard tranform and its transpose.
Step 6: Display Transformed Image.
Step 7: Obtain reconstructedimage using hadamard matrices.
Step 8: Display reconstructed image.
```

Program Code

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
from scipy . linalg import hadamard
import math

img = cv2 . imread ("Cameraman.png",0)

l= np . ceil ( math . log ( img . shape [0] ,2) ) ;
m= np . ceil ( math . log ( img . shape [1] ,2) ) ;
img1 = cv2 . resize ( img , [ int ( pow (2 , max (l , m))) ,int( pow (2 , max(l ,m) )) ])
M , N = img1 . shape [:2]
H = hadamard (M )
print (H)
txmed = np . dot ( np . dot (H , img1 ) , np . transpose (H))
print ( txmed )

plt.figure()
plt . imshow ( img1 , cmap ="gray")
plt . title ('Original Image')

txmednor = np . real ( txmed ) / np . max ( np . real ( txmed ))

plt.figure()
plt . imshow ( np . log (1+( np . abs ( txmednor ))) , cmap ="gray")
plt . title ('Hadamard Transform Image')
```

```
inv_img = np . dot ( np . dot ( np . linalg . inv (H) , txmed ) , np . linalg . inv (H) )

plt.figure()
plt . imshow ( inv_img , cmap ="gray")
plt . title ('Reconstructed Image')
plt . show ()
```

Result

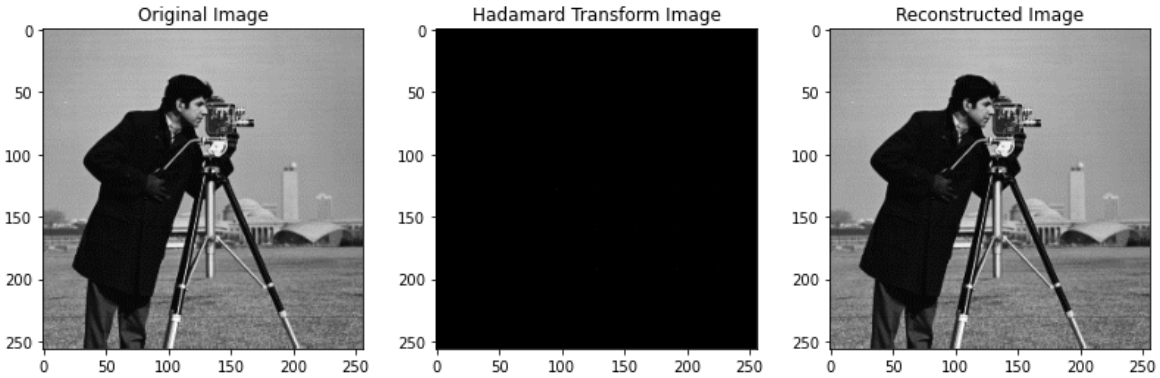


Figure 4: Cameraman : (a)Original Image (b) Hadamard Transformed Image (c) Reconstructed Image

```
   Hadamard Mtrix =

[[ 1  1  1 ...  1  1  1]
 [ 1 -1  1 ... -1  1 -1]
 [ 1  1 -1 ...  1 -1 -1]
 ...
 [ 1 -1  1 ... -1  1 -1]
 [ 1  1 -1 ...  1 -1 -1]
 [ 1 -1 -1 ... -1 -1  1]]
```

Inference

In hadamard transorm, multiplication calculations are not required.It provids with good energy compaction for highly correlated images.

Question 4: Transforming image using Wavelet Transform (Daubechies-8) and performing re-construction.

Aim

Transforming an image using Wavelet Transform(Daubechies-8) and reconstructing image from transformed image

Discussion

The wavelet transform decomposes the original image into a sum of spatially and frequency localized functions, in a way that is similar to subband decomposition. The most important visual information tends to be concentrated into a reduced number of components (coefficients); therefore, the remaining coefficients can be quantized coarsely or truncated to zero with little image distortion.

The Daubechies wavelets, based on the work of Ingrid Daubechies, are a family of orthogonal wavelets defining a discrete wavelet transform and characterized by a maximal number of vanishing moments for some given support.

Algorithm

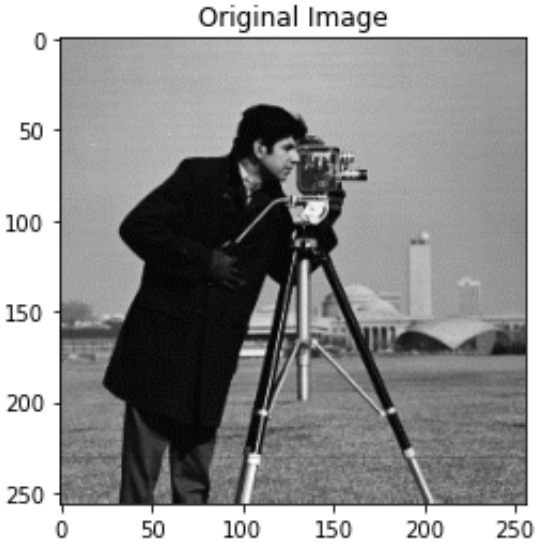
```
Step 1: Start
Step 2: Read the image
Step 3: Obtain transformed image in the form of db8 components using function.
Step 6: Display the components.
Step 7: Apply inverse db8 transform on obtained db8
        components in Step 3 to get reconstructed image.
Step 8: Display Reconstructed Image
```

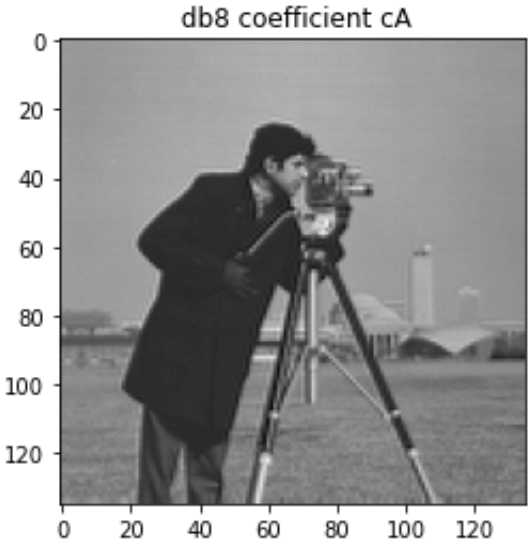
Program Code

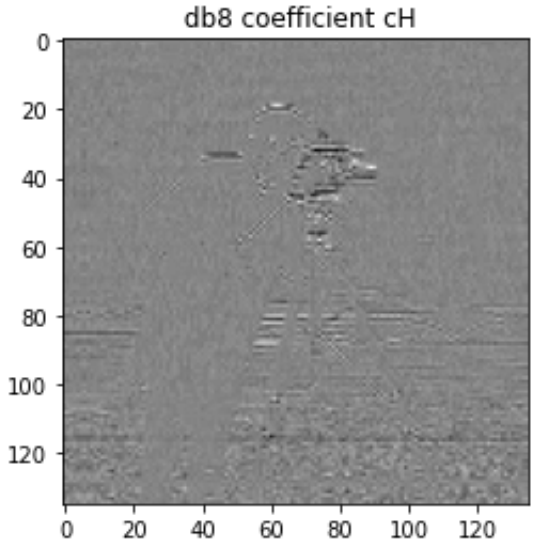
```
import cv2
from matplotlib import pyplot as plt
from pywt import dwt2 , idwt2

img = cv2 . imread ("Cameraman.png",0)
cA , (cH , cV , cD ) = dwt2 ( img , 'db8')

plt . figure ()
plt . imshow (cA , cmap = "gray")
plt . title (" db8 coefficient cA")

plt . figure ()
plt . imshow (cH , cmap = "gray")
plt . title (" db8 coefficient cH")

plt . figure ()
plt . imshow (cV , cmap = "gray")
plt . title (" db8 coefficient cV")

plt . figure ()
plt . imshow (cD , cmap = "gray")
plt . title (" db8 coefficient cD")

plt . figure ()
plt . imshow ( img , cmap = "gray")
plt . title (" Original Image ")
```

```
recimg = idwt2 (( cA , (cH , cV , cD )) , 'db8')

plt . figure ()
plt . imshow ( recimg , cmap = "gray")
plt . title (" Reconstructed Image ")
```

Result







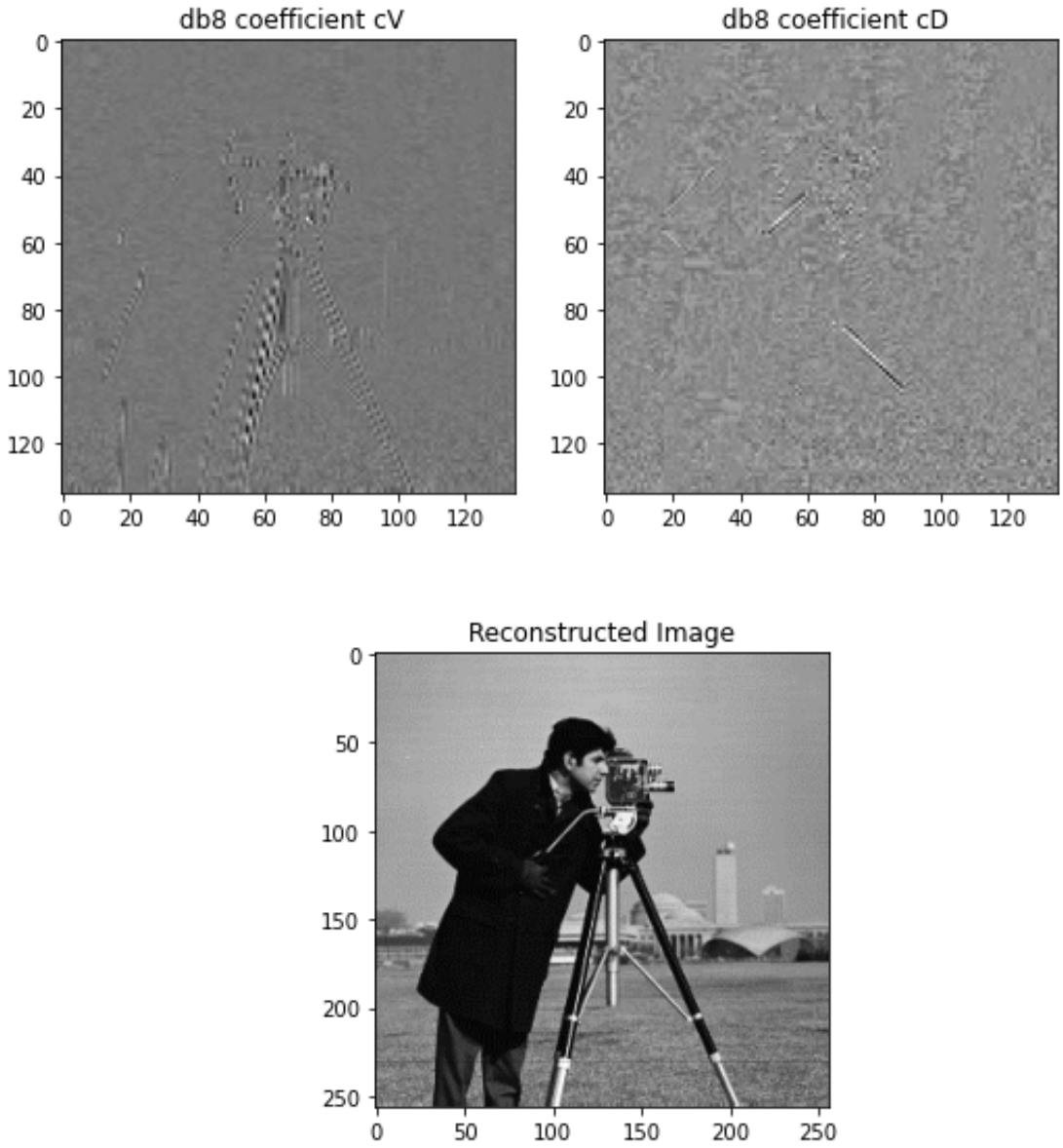


Figure 5: (a)Original Image (b) Daubechies Transform Components (c) Reconstructed Image

Inference

The advantages of wavelet transform is that they offer a simultaneous localization in time and frequency domain. Wavelets have the great advantage of being able to separate the fine details in a signal.