Problem 7

Digital Image Processing

肖逸群 116033910112

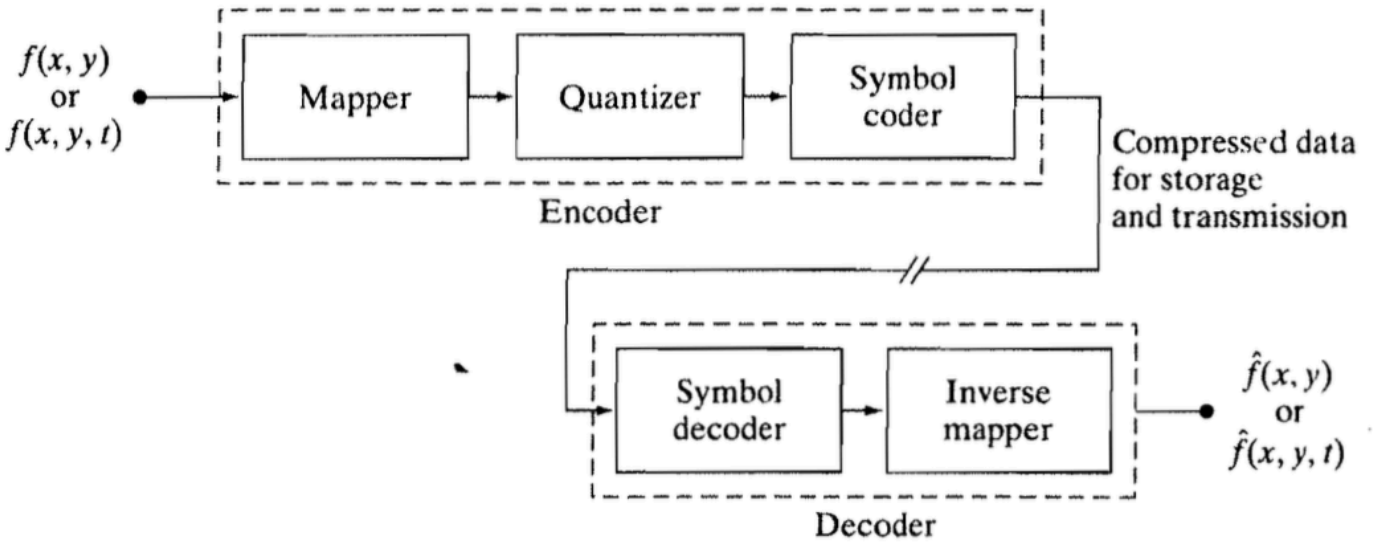
Transform image compression

问题描述

> (a) Investigate image compression based on DCT. Divide the image into 8-by-8 subimages, compute the two- dimensional discrete cosine transform of each subimage, compress the test image to different qualities by discarding some DCT coefficients based on zonal mask and threshold mask and using the inverse discrete cosine transform with fewer transform coefficients. Display the original image, the reconstructed images and the difference images.
> (b) Investigate image compression based on wavelets. Consider four types of wavelets: Haar, Daubechies, Symlet, The biorthogonal Cohen-Daubechies-Feauveau.
> Decompose the test image by wavelets to 3 levels, truncate the wavelet coefficients to 0 below some threshold. And reconstruct image from the left coefficients. Display the wavelet transforms of the image, the reconstructed images and the difference images.
> (Consult Chapter 7 for more technique details).

算法思想

从 Rafael C.Gonzalez 的书中我们可以看到图像压缩模型如下：



由上图内容可知，图像压缩系统是由编码器和解码器这两个不同的功能部分所组成的，编码器执行压缩的操作，解码器执行解压的互补操作。
在本题中，我们进行的是对静止图片的压缩和解压，故编码器的输入和输出分别是 f(x, y) 和 f_hat(x, y)。其中，如果 f_hat(x, y) 是 f(x, y) 的精确复制，则压缩系统被称为无损的，否则为有损的。

本题的 (a) 部分是基于 DCT 来对图像进行压缩和解压的。 DCT 函数定义如下：

$$F(0,0)=\frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)$$

$$F(0,v)=\frac{\sqrt{2}}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)\cdot\cos\frac{(2y+1)v\pi}{2N}$$

$$F(u,0)=\frac{\sqrt{2}}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)\cos\frac{(2x+1)u\pi}{2N}$$

$$F(u,v)=\frac{2}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)\cos\frac{(2x+1)u\pi}{2N}\cdot\cos\frac{(2y+1)v\pi}{2N}$$

iDCT 函数的定义如下：

$$f(x,y)=\frac{1}{N}F(0,0)+\frac{\sqrt{2}}{N}\sum_{v=1}^{N-1}F(0,v)\cos\frac{(2y+1)v\pi}{2N}$$
$$+\frac{\sqrt{2}}{N}\sum_{u=1}^{N-1}F(u,0)\cos\frac{(2x+1)u\pi}{2N}$$
$$+\frac{2}{N}\sum_{u=1}^{N-1}\sum_{v=1}^{N-1}F(u,v)\cos\frac{(2x+1)u\pi}{2N}\cdot\cos\frac{(2y+1)v\pi}{2N}$$

由于DCT变换高度的对称性，在使用Matlab进行相关的运算时，我们可以使用更简单的矩阵处理方式：

$$F=AfA^T$$

$$A(i,j)=c(i)\cos\left[\frac{(j+0.5)\pi}{N}i\right]$$

在本题中，我为了简化计算，采用的就是该方法求出矩阵 A，再来做 DCT。
在得到 DCT 变换后的结果后，我们需要对图像进行掩模运算，掩模处理的定义如下：
当 F(u, v) 符合掩模的切割标准时，其灰度值 r(u, v) 被设为 0，否则设为 1。本题中要求我们使用 zonal 掩模和阈值掩模。

本题要求以 Haar, Daubechies, Symlet, The biorthogonal Cohen-Daubechies-Feauveau 四种小波变换方法来对图像进行压缩，具体四种小波变换我直接使用了 pywt 里的小波变换函数，首先使用不同的小波函数对原始图像进行编码，然后根据所设阈值对变换后的矩阵进行截断，达到压缩图像的目的，最后再使用相应的小

波函数对原始图像进行解码，得到压缩图像。

源码分析

- 第一部分的实验过程如下：

    a. 读取原始图片并将其转为灰度图像。
    b. 对原始图片分别使用 zonal mask 和 threshold mask 进行压缩，在压缩之前需要先把图像切割成为 8 * 8 的子图。
    c. 对子图进行离散余弦变换，在此基础上根据所用掩模对子图进行截断，返回阶段后的子图列表。
    d. 对得到子图列表进行离散余弦逆变换，最后拼接各个子图，得到压缩后的图像。
    e. 求出原始图像与压缩图像的灰度值差别，得到差异图像。
    f. 显示原始图像，压缩图像与差异图像。

- 第二部分的实验过程如下：

    a. 对原始图像分别进行四种小波编码变换。
    b. 对变换后得到的图像根据所设定阈值进行截断。
    c. 对截断后的图像进行相应的小波解码变换，返回压缩后的图片。
    d. 求出原始图像与压缩图像的灰度值差别，得到差异图像。
    e. 显示原始图像，压缩图像与差异图像。

代码如下：

```
def main():
    originalImage = np.array(Image.open('./resource/lenna.tif'))
    row, col = originalImage.shape
    plt.subplot(2, 3, 1)
    plt.imshow(originalImage, cmap = plt.get_cmap('gray'))
    plt.title('Original')

    zonalMaskSubImg = DiscreteCosineTransform.imageCompress(originalImage, "zonal mask")
    zonalMaskRestoreImage = DiscreteCosineTransform.imageRestore(zonalMaskSubImg, row, c
ol)
    plt.subplot(2, 3, 2)
    plt.imshow(zonalMaskRestoreImage, cmap = plt.get_cmap('gray'))
    plt.title('Restore Image(zonal mask)')

    zonalMaskDifferenceImage = originalImage - zonalMaskRestoreImage
    plt.subplot(2, 3, 3)
    plt.imshow(zonalMaskDifferenceImage, cmap = plt.get_cmap('gray'))
    plt.title('Difference Image(zonal mask)')

    thresholdMaskSubImg = DiscreteCosineTransform.imageCompress(originalImage, "threshol
d mask")
    thresholdMaskRestoreImage = DiscreteCosineTransform.imageRestore(thresholdMaskSubImg
, row, col)
    plt.subplot(2, 3, 5)
    plt.imshow(thresholdMaskRestoreImage, cmap = plt.get_cmap('gray'))
    plt.title('Restore Image(threshold mask)')

    thresholdDifferenceImage = originalImage - thresholdMaskRestoreImage
    plt.subplot(2, 3, 6)
    plt.imshow(thresholdDifferenceImage, cmap = plt.get_cmap('gray'))
    plt.title('Difference Image(threshold mask)')

    plt.show()

```

```
    haarTransformImage, haarReconstructImage = WaveletsCompression.waveTransform('haar', originalImage, threshold = 0)
    plt.subplot(1, 3, 1)
    plt.imshow(haarTransformImage, cmap = plt.get_cmap('gray'))
    plt.title('Haar Transform Image')

    plt.subplot(1, 3, 2)
    plt.imshow(haarReconstructImage, cmap = plt.get_cmap('gray'))
    plt.title('Haar Reconstruct Image')

    haarDifferenceImage = originalImage - haarReconstructImage
    plt.subplot(1, 3, 3)
    plt.imshow(haarDifferenceImage, cmap = plt.get_cmap('gray'))
    plt.title('Difference Image(Haar)')

    plt.show()

    daubechiesTransformImage, daubechiesReconstructImage = WaveletsCompression.waveTransform('db3', originalImage, threshold = 0)
    plt.subplot(1, 3, 1)
    plt.imshow(daubechiesTransformImage, cmap = plt.get_cmap('gray'))
    plt.title('Daubechies Transform Image')

    plt.subplot(1, 3, 2)
    plt.imshow(daubechiesReconstructImage, cmap = plt.get_cmap('gray'))
    plt.title('Daubechies Reconstruct Image')

    daubechiesDifferenceImage = originalImage - daubechiesReconstructImage
    plt.subplot(1, 3, 3)
    plt.imshow(daubechiesDifferenceImage, cmap = plt.get_cmap('gray'))
    plt.title('Difference Image(Daubechies)')

    plt.show()

    symletTransformImage, symletReconstructImage = WaveletsCompression.waveTransform('sym3', originalImage, threshold = 0)
    plt.subplot(1, 3, 1)
    plt.imshow(symletTransformImage, cmap = plt.get_cmap('gray'))
    plt.title('Symlet Transform Image')

    plt.subplot(1, 3, 2)
    plt.imshow(symletReconstructImage, cmap = plt.get_cmap('gray'))
    plt.title('Symlet Reconstruct Image')

    symletDifferenceImage = originalImage - symletReconstructImage
    plt.subplot(1, 3, 3)
    plt.imshow(symletDifferenceImage, cmap = plt.get_cmap('gray'))
    plt.title('Difference Image(Symlet)')

    plt.show()

    biorthogonalCDFTransformImage, biorthogonalCDFReconstructImage = WaveletsCompression.waveTransform('bior1.1', originalImage, threshold = 0)
    plt.subplot(1, 3, 1)
    plt.imshow(biorthogonalCDFTransformImage, cmap = plt.get_cmap('gray'))
    plt.title('biorthogonalCDF Transform Image')

    plt.subplot(1, 3, 2)
    plt.imshow(biorthogonalCDFReconstructImage, cmap = plt.get_cmap('gray'))
    plt.title('biorthogonalCDF Reconstruct Image')

```

```
    biorthogonalCDFDifferenceImage = originalImage - biorthogonalCDFReconstructImage
    plt.subplot(1, 3, 3)
    plt.imshow(biorthogonalCDFDifferenceImage, cmap = plt.get_cmap('gray'))
    plt.title('Difference Image(biorthogonalCDF)')

    plt.show()
```

- 子图切割函数：

```
def divideImage(originalImage):
    row, col = originalImage.shape
    row8 = row // 8
    col8 = col // 8
    resList = []
    for i in range(row8):
        for j in range(col8):
            resList.append(originalImage[i * 8 : (i + 1) * 8, j * 8 : (j + 1) * 8])
    return resList
```

- 离散余弦变换函数：

```
def DCT(subImage):
    row , col = subImage.shape
    A = np.zeros((row, col))
    for i in range(row):
        for j in range(col):
            c = np.sqrt(1 / 8) if i == 0 else np.sqrt(2 / 8)
            A[i, j] = c * np.cos(np.pi * (j + 0.5) * i / 8)
    res = np.dot(A, subImage)
    res = np.dot(res, A.T)
    return res
```

- 图像压缩函数：

```
def imageCompress(orginalImage, method):
    divImg = divideImage(orginalImage)
    DCTImg = []
    for img in divImg:
        DCTSubImg = DCT(img)
        if method == 'zonal mask':
            DiscardImg = coefficientDiscard(DCTSubImg, zonalMask, 0)
            DCTImg.append(DiscardImg)
        elif method == 'threshold mask':
            DiscardImg = coefficientDiscard(DCTSubImg, DCTSubImg / t, 1 / 2 / 255)
            DCTImg.append(DiscardImg)
    return DCTImg
```

- 阈值截断函数：

```
def coefficientDiscard(img, mask, threshold):
    resImg = np.zeros((8, 8))
    for i in range(8):
        for j in range(8):
            if mask[i][j] > threshold:
                resImg[i, j] = img[i, j]
    return resImg
```

- 离散余弦逆变换函数：

```
def coefficientDiscard(img, mask, threshold):
    resImg = np.zeros((8, 8))
```

```
    for i in range(8):
        for j in range(8):
            if mask[i][j] > threshold:
                resImg[i, j] = img[i, j]
    return resImg
```

- 子图合并函数：

```
def imageRestore(transformedImg, row, col):
    restoreImg = []
    for img in transformedImg:
        restoreImg.append(np.round(IDCT(img)))
    row8 = row // 8
    col8 = col // 8
    temp = []
    for i in range(row8):
        temp.append(np.column_stack(restoreImg[i * col8 : (i + 1) * col8]))
    res = np.row_stack(temp)
    return res
```

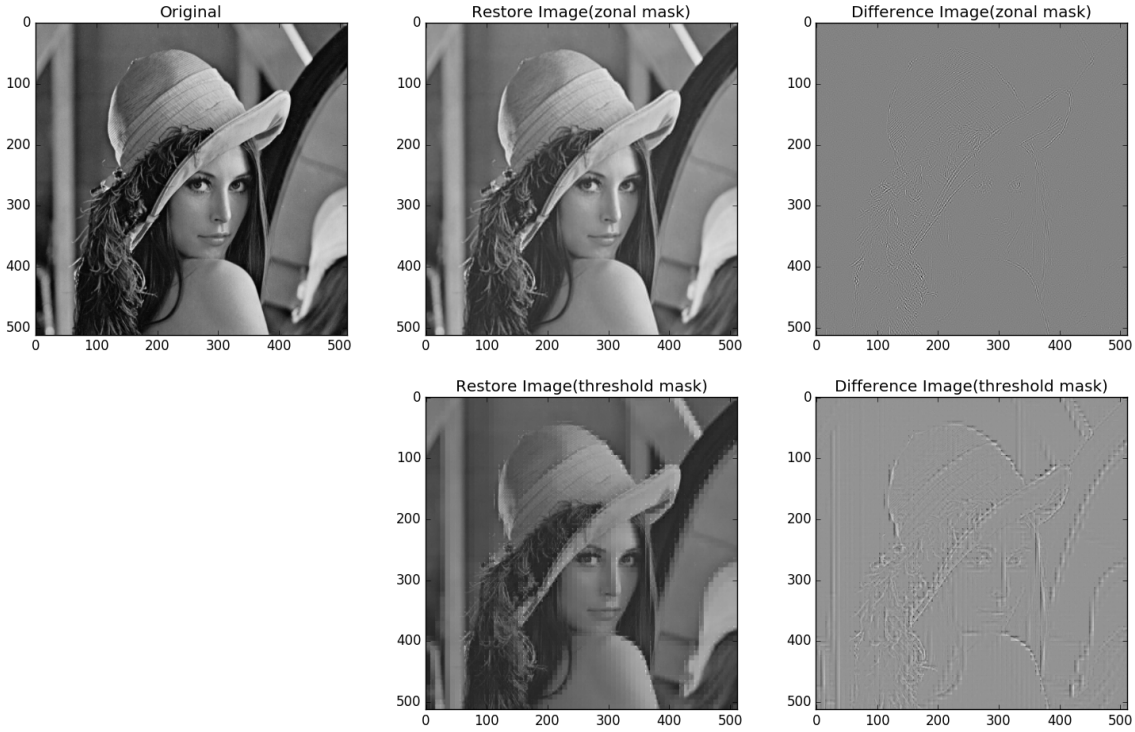
- 小波变换函数：

```
def waveTransform(wavelet, originalImage, threshold):
    coeffs = pywt.wavedec2(originalImage, wavelet = wavelet, level = 3)
    cA3, (cH3, cV3, cD3), (cH2, cV2, cD2), (cH1, cV1, cD1) = coeffs
    cDList = [cD1, cD2, cD3]
    for i in range(3):
        cD = cDList[i]
        for j in range(len(cD)):
            if cD[j].any() < threshold:
                cD[i][j] = 0
    transformedImage = pywt.waverec2(coeffs, wavelet = wavelet)
    return cA3, transformedImage
```

实验结果

下列图片为不同压缩算法下的实验结果：

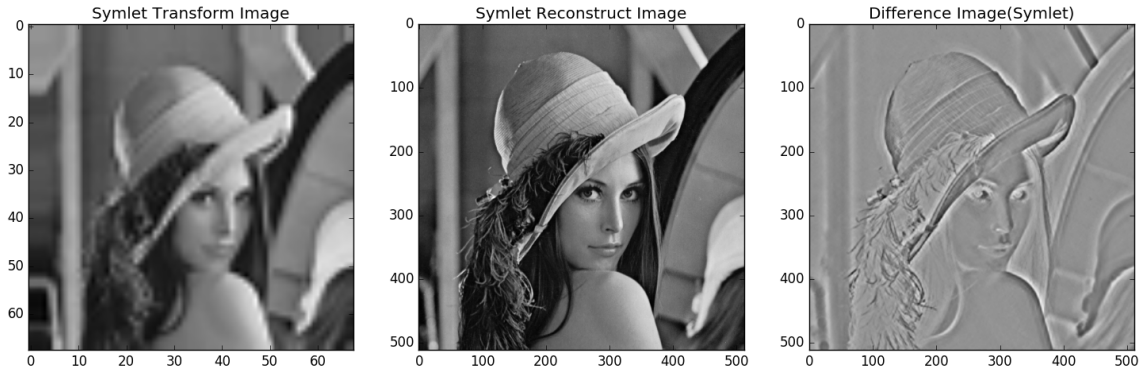
- zonal 掩模与阈值掩模实验结果



- Haar 压缩变换实验结果：

- Daubechies 压缩变换实验结果：

- Symlet 压缩变换实验结果：



- The biorthogonal Cohen-Daubechies-Feauveau 压缩变换实验结果：


biorthogonalCDF Transform Image
biorthogonalCDF Reconstruct Image
Difference Image(biorthogonalCDF)