PDI_Trabalho2

December 8, 2020

0.1 Introdução ao Processamento Digital de Imagens - PDI: Módulo 2 do Trabalho Prático

0.1.1 Alunos: Caio Albuquerque, Epitácio Neto, Guilherme Moreira, Jordan Elias e Kaio Moura

0.1.2 Professor: Leonardo Vidal

0.2 0. Descrição do Trabalho

Apresentar a imagem resultante do seguinte processamento sobre uma imagem I em tons de cinza: 1. Obtenção da DCT bidimensional de I: Id = DCT2(I) 2. Preservação do nível DC e dos n coeficientes AC mais importantes de Id, e anulação de todos os demais coeficientes, produzindo uma matriz Idp. Atenção: Em geral, essa operação não equivale a um filtro passa-baixas! 3. Obtenção e exibição da imagem Ip resultante da DCT inversa de Idp, Ip = round(IDCT2(Idp)). Lembre-se que a faixa permitida para os níveis de cinza de Ip é [0, L-1]; os valores devem ser truncados para esse intervalo. As operações DCT direta e inversa devem ser desenvolvidas diretamente das equações vistas em aula (sem usar funções prontas), de dois modos: a) utilizando a separabilidade; e b) sem utilizar a separabilidade. Apresente resultados para n = 0, 1, 2, 3, e 10; n = 25% do total de coeficientes de Id, n = 50% do total de coeficientes de Id, n = 75% do total de coeficientes de Id, e n = 100% do total de coeficientes de Id, bem como os tempos de processamento para cálculo da DCT direta e da inversa, com e sem separabilidade.

0.3 1. Introdução

Neste trabalho de cunho avaliativo, temos como propósito a aplicação dos fundamentos teóricos da disciplina de Introdução ao Processamento Digital de Imagens, ministradas à distância pelo professor Leonardo Vidal. Ao longo destas aulas, aprendemos métodos de conversão de números do domínio do tempo para o domínio da frequência, em específico a representação numérica de sinais (de imagem ou áudio) por meio de transformadas, como a Transformada de Fourier e a DCT (Direct Cosine Transform).

Em específico, focaremos na DCT e sua inversa (IDCT, ou Inverse Direct Cosine Transform). Assim, aplicaremos a seguinte forma de trabalhar com as DCTs e IDCTs: com separabilidade (a abordagem sem separabilidade será abordada posteriormente), ou seja, utilizaremos esta propriedade de forma a simplificar o problema caso o mesmo fosse abordado sem ela, evitamos, então, a complexidade de trabalhar com as equações da DCT 2D com os recursos aprendidos quando, durante as aulas, trabalhamos com as DCTs 1D.

Portanto, partiremos de uma operação de uma DCT 1D e, usando o fundamento da separabilidade, demonstraremos a aplicação destas DCTs 1D para criarmos uma DCT 2D e, esta mesma situação

se repetirá na volta, ou seja, quando estivermos utilizando as IDCTs. O intuito é aplicar este conhecimento em uma imagem e, desta forma, aplicamos a DCT 2D para levarmos-a ao domínio da frequência e a IDCT para trazê-la de volta ao domínio do tempo.

Avaliaremos o tempo de processamento destas operação bem como o resultado final da IDCT durante nossos casos de teste, específicados pelo professor, onde alteraremos os coeficientes que fazem parte da análise e a influência juntamente como a diferença da imagem final quando estes coeficientes são diferentes.

0.3.1 Materiais e métodos

Neste projeto, utilizaremos a linguagem de programação Python 3, juntamente com as funcionalidades básicas com suas bibliotecas, portanto, utilizaremos funções com intuito de:

I. Abrir e ler a imagem lena265.jpg, disponibilizada por Surya Prasath no site Research Gate, originalmente baixada e disponibilizada por World Scientific para propósitos acadêmicos;

II. Comandos para leitura da imagem, nas quais foram alteradas e trabalhadas, com intuito de análise do conteúdo da disciplina;

III. Comando para salvar as imagens em uma máquina local;

IV. Comandos para analisar o tempo de processamento em cada caso de teste que serão abordados posteriormente.

0.3.2 Bibliotecas

[34]:
```
from math import cos, pi, sqrt
import numpy as np
import cv2
import config
import time, random, datetime
from PIL import Image
import os
from matplotlib import pyplot as plt
from matplotlib.pyplot import imshow
```

0.3.3 Declaração de funções

[2]:
```
def dct_2d(image, numberCoefficients):
    start_time = time.time()

    nc = numberCoefficients
    height = image.shape[0]
    width = image.shape[1]
    imageRow = np.zeros_like(image).astype(float)
    imageCol = np.zeros_like(image).astype(float)

    for h in range(height):
        imageRow[h, :] = dct_1d(image[h, :], nc)
```

```
    for w in range(width):
        imageCol[:, w] = dct_1d(imageRow[:, w], nc)

    processing_time = time.time() - start_time
    print("\nProcessing time: %s seconds \n" % "{0:.3f}".
 ↪format(processing_time))

    return imageCol

def dct_1d(image, numberCoefficients):

    nc = numberCoefficients
    n = len(image)
    newImage= np.zeros_like(image).astype(float)


    for k in range(n):
        sum = 0
        for i in range(n):
            sum += image[i] * cos(2 * pi * k / (2.0 * n) * i + (k * pi) / (2.0␣
 ↪* n))
        ck = sqrt(0.5) if k == 0 else 1
        newImage[k] = sqrt(2.0 / n) * ck * sum

    # salvando os N maiores numeros e zerandos todos os outros
    if nc > 0:
        newImage.sort()
        for i in range(nc, n):
            newImage[i] = 0

    return newImage # retorno de um VETOR

def idct_2d(image):
    start_time = time.time()

    height = image.shape[0]
    width =  image.shape[1]
    imageRow = np.zeros_like(image).astype(float)
    imageCol = np.zeros_like(image).astype(float)


    for h in range(height):
        imageRow[h, :] = idct_1d(image[h, :])
    for w in range(width):
        imageCol[:, w] = idct_1d(imageRow[:, w])

    processing_time = time.time() - start_time
```

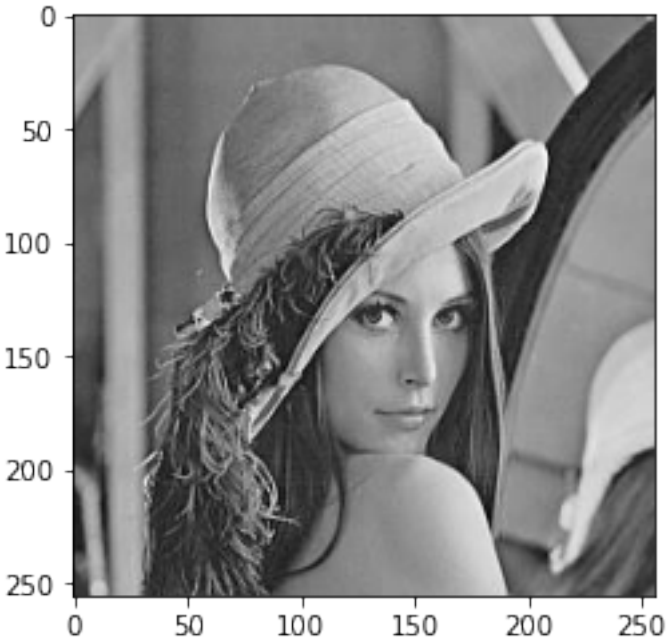
```
    print("\nProcessing time: %s seconds \n" % "{0:.3f}".
↪format(processing_time))

    return imageCol

def idct_1d(image):

    n = len(image)
    newImage = np.zeros_like(image).astype(float)

    for i in range(n):
        sum = 0
        for k in range(n):
            ck = sqrt(0.5) if k == 0 else 1
            sum += ck * image[k] * cos(2 * pi * k / (2.0 * n) * i + (k * pi) /␣
↪(2.0 * n))

        newImage[i] = sqrt(2.0 / n) * sum

    return newImage

def abrir_imagem(path):
    filename, file_extension = os.path.splitext(path)

    imagem = Image.open(path)

    return imagem
```

0.4 2. Leitura da imagem

[3]:
```
imageToRead = 'lena256.jpg'

img = cv2.imread(imageToRead)
```

[35]:
```
imshow(img)
```

[35]:
```
<matplotlib.image.AxesImage at 0x20431655790>
```



0.5 3. Casos de teste

0.5.1 Para n = 0

[4]:
```
numberCoefficients = 0

imgResult = dct_2d(img,numberCoefficients)
cv2.imwrite('dct256.jpg',imgResult)
```

```
Processing time: 71.242 seconds
```

[4]:
```
True
```

[5]:
```
idct_img = idct_2d(imgResult)
cv2.imwrite('idct256.jpg',idct_img)
```

```
Processing time: 94.005 seconds
```

[5]:
```
True
```

[6]:
```
path1 = 'dct256.jpg'
path2 = 'idct256.jpg'
```

```
imagem = abrir_imagem(path1)
imagem = np.array(imagem, dtype='int')
imagem  = np.clip(imagem, a_min = 0, a_max = 255)

imagem1 = abrir_imagem(path2)
imagem1 = np.array(imagem1, dtype='int')
imagem1  = np.clip(imagem1 , a_min = 0, a_max = 255)

fig, (ax_orig, ax_conv) = plt.subplots(1, 2)

ax_orig.imshow(imagem)
ax_orig.set_title('DCT')
ax_orig.set_axis_off()

ax_conv.imshow(imagem1)
ax_conv.set_title('IDCT')
ax_conv.set_axis_off()

fig.show()
```

```
<ipython-input-6-832a679c09f5>:22: UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot
show the figure.
  fig.show()
```

0.5.2 Para n = 1

[7]:
```
numberCoefficients = 1

imgResult = dct_2d(img,numberCoefficients)
cv2.imwrite('dct256n1.jpg',imgResult)
```

```
Processing time: 71.833 seconds
```

[7]:
```
True
```

[8]:
```
idct_img = idct_2d(imgResult)
cv2.imwrite('idct256n1.jpg',idct_img)
```

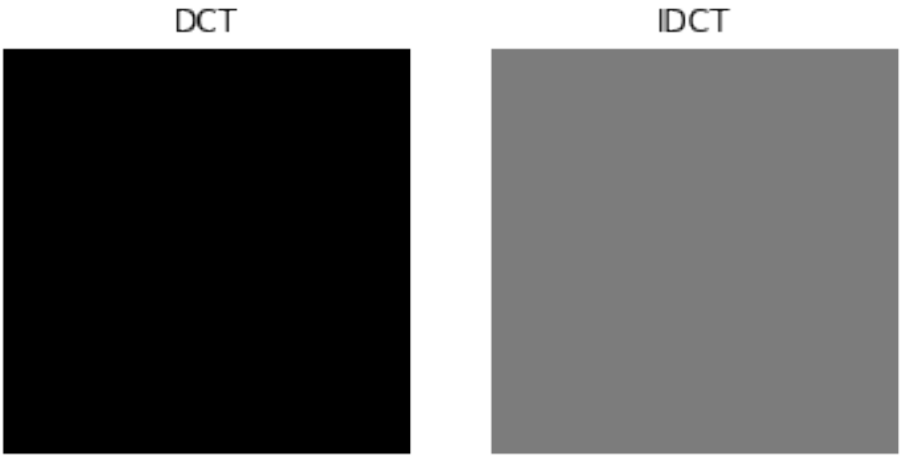
```
Processing time: 93.622 seconds
```

[8]:
```
True
```

[9]:
```
path1 = 'dct256n1.jpg'
path2 = 'idct256n1.jpg'

imagem = abrir_imagem(path1)
imagem = np.array(imagem, dtype='int')
imagem  = np.clip(imagem, a_min = 0, a_max = 255)

imagem1 = abrir_imagem(path2)
imagem1 = np.array(imagem1, dtype='int')
imagem1  = np.clip(imagem1 , a_min = 0, a_max = 255)

fig, (ax_orig, ax_conv) = plt.subplots(1, 2)

ax_orig.imshow(imagem)
ax_orig.set_title('DCT')
ax_orig.set_axis_off()

ax_conv.imshow(imagem1)
ax_conv.set_title('IDCT')
ax_conv.set_axis_off()

fig.show()
```

```
<ipython-input-9-55625e19539c>:22: UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot
show the figure.
```

```
fig.show()
```



0.5.3 Para n = 2

[10]:
```
numberCoefficients = 2

imgResult = dct_2d(img,numberCoefficients)
cv2.imwrite('dct256n2.jpg',imgResult)
```

```
Processing time: 70.577 seconds
```

[10]:
```
True
```

[11]:
```
idct_img = idct_2d(imgResult)
cv2.imwrite('idct256n2.jpg',idct_img)
```

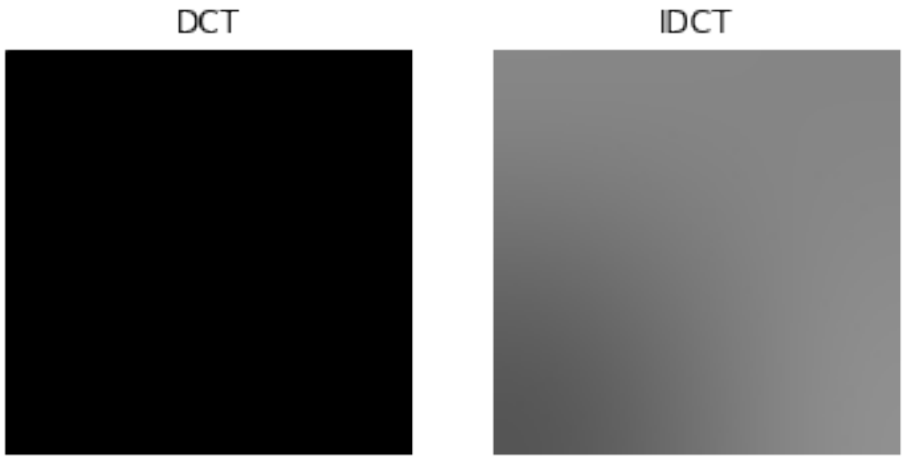
```
Processing time: 93.144 seconds
```

[11]:
```
True
```

[12]:
```
path1 = 'dct256n2.jpg'
path2 = 'idct256n2.jpg'

imagem = abrir_imagem(path1)
imagem = np.array(imagem, dtype='int')
imagem  = np.clip(imagem, a_min = 0, a_max = 255)
```

```
imagem1 = abrir_imagem(path2)
imagem1 = np.array(imagem1, dtype='int')
imagem1  = np.clip(imagem1 , a_min = 0, a_max = 255)

fig, (ax_orig, ax_conv) = plt.subplots(1, 2)

ax_orig.imshow(imagem)
ax_orig.set_title('DCT')
ax_orig.set_axis_off()

ax_conv.imshow(imagem1)
ax_conv.set_title('IDCT')
ax_conv.set_axis_off()

fig.show()
```

```
<ipython-input-12-635861f4aa89>:22: UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot
show the figure.
  fig.show()
```



0.5.4 Para n = 3

[13]:
```
numberCoefficients = 3

imgResult = dct_2d(img,numberCoefficients)
cv2.imwrite('dct256n3.jpg',imgResult)
```

```
Processing time: 70.759 seconds
```

[13]: True

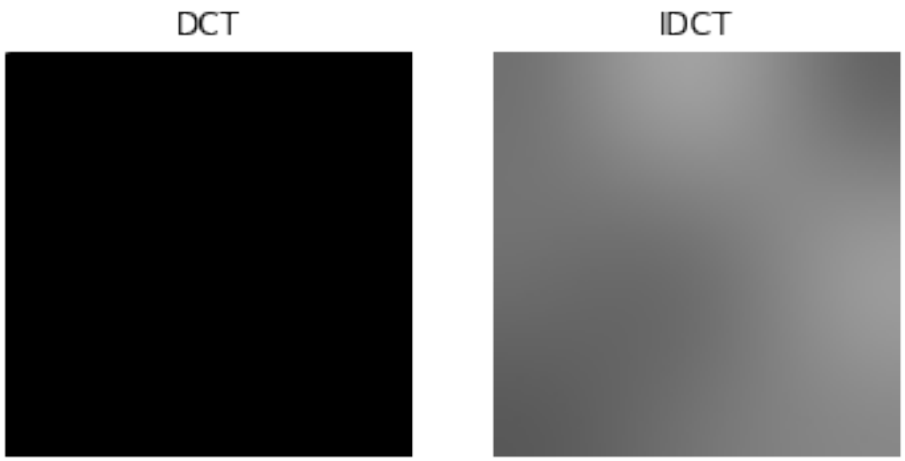
[14]:
```
idct_img = idct_2d(imgResult)
cv2.imwrite('idct256n3.jpg',idct_img)
```

```
Processing time: 92.434 seconds
```

[14]: True

[15]:
```
path1 = 'dct256n3.jpg'
path2 = 'idct256n3.jpg'

imagem = abrir_imagem(path1)
imagem = np.array(imagem, dtype='int')
imagem  = np.clip(imagem, a_min = 0, a_max = 255)

imagem1 = abrir_imagem(path2)
imagem1 = np.array(imagem1, dtype='int')
imagem1  = np.clip(imagem1 , a_min = 0, a_max = 255)

fig, (ax_orig, ax_conv) = plt.subplots(1, 2)

ax_orig.imshow(imagem)
ax_orig.set_title('DCT')
ax_orig.set_axis_off()

ax_conv.imshow(imagem1)
ax_conv.set_title('IDCT')
ax_conv.set_axis_off()

fig.show()
```

```
<ipython-input-15-7c47bc57115f>:22: UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot
show the figure.
  fig.show()
```



0.5.5 Para n = 10

[16]:
```
numberCoefficients = 10

imgResult = dct_2d(img,numberCoefficients)
cv2.imwrite('dct256n10.jpg',imgResult)
```

```
Processing time: 72.687 seconds
```

[16]:
```
True
```

[17]:
```
idct_img = idct_2d(imgResult)
cv2.imwrite('idct256n10.jpg',idct_img)
```

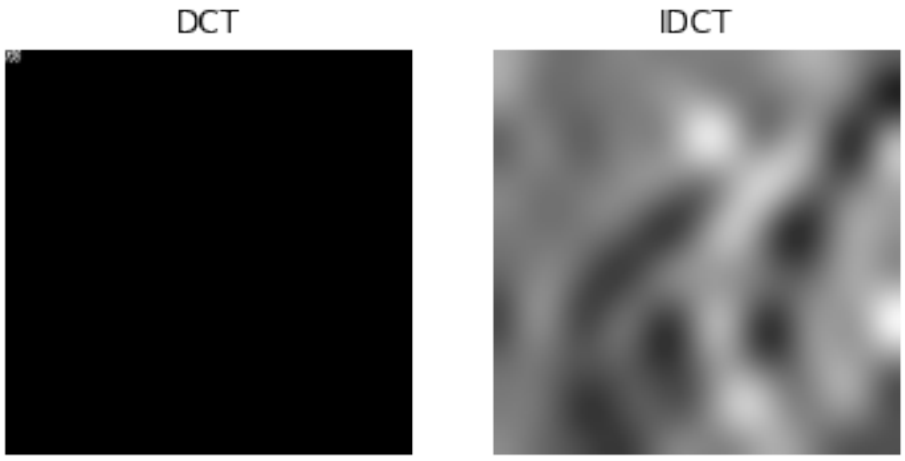
```
Processing time: 97.006 seconds
```

[17]:
```
True
```

[18]:
```
path1 = 'dct256n10.jpg'
path2 = 'idct256n10.jpg'

imagem = abrir_imagem(path1)
imagem = np.array(imagem, dtype='int')
imagem  = np.clip(imagem, a_min = 0, a_max = 255)

imagem1 = abrir_imagem(path2)
imagem1 = np.array(imagem1, dtype='int')
```

```
imagem1  = np.clip(imagem1 , a_min = 0, a_max = 255)

fig, (ax_orig, ax_conv) = plt.subplots(1, 2)

ax_orig.imshow(imagem)
ax_orig.set_title('DCT')
ax_orig.set_axis_off()

ax_conv.imshow(imagem1)
ax_conv.set_title('IDCT')
ax_conv.set_axis_off()

fig.show()
```

```
<ipython-input-18-e7a643528e38>:22: UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot
show the figure.
  fig.show()
```



0.5.6 Para n = 25% da imagem

[21]:
```
numberCoefficients = round(len(img)*0.25)

imgResult = dct_2d(img,numberCoefficients)
cv2.imwrite('dct256n25.jpg',imgResult)
```

```
Processing time: 72.116 seconds
```

[21]: True

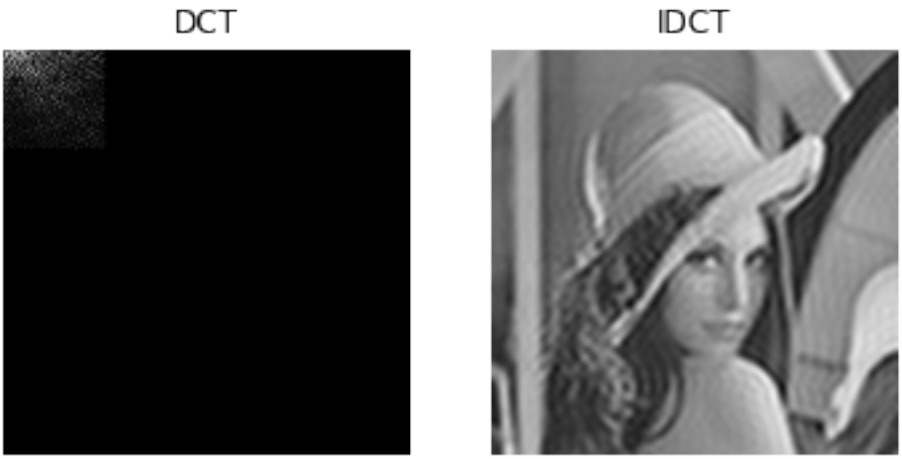
[22]:
```
idct_img = idct_2d(imgResult)
cv2.imwrite('idct256n25.jpg',idct_img)
```

```
Processing time: 93.902 seconds
```

[22]:
```
True
```

[23]:
```
path1 = 'dct256n25.jpg'
path2 = 'idct256n25.jpg'

imagem = abrir_imagem(path1)
imagem = np.array(imagem, dtype='int')
imagem  = np.clip(imagem, a_min = 0, a_max = 255)

imagem1 = abrir_imagem(path2)
imagem1 = np.array(imagem1, dtype='int')
imagem1  = np.clip(imagem1 , a_min = 0, a_max = 255)

fig, (ax_orig, ax_conv) = plt.subplots(1, 2)

ax_orig.imshow(imagem)
ax_orig.set_title('DCT')
ax_orig.set_axis_off()

ax_conv.imshow(imagem1)
ax_conv.set_title('IDCT')
ax_conv.set_axis_off()

fig.show()
```

```
<ipython-input-23-cb2be09a2510>:22: UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot
show the figure.
  fig.show()
```



0.5.7 Para n = 50% da imagem

[24]:
```
numberCoefficients = round(len(img)*0.5)

imgResult = dct_2d(img,numberCoefficients)
cv2.imwrite('dct256n50.jpg',imgResult)
```

```
Processing time: 72.354 seconds
```

[24]:
```
True
```

[25]:
```
idct_img = idct_2d(imgResult)
cv2.imwrite('idct256n50.jpg',idct_img)
```

```
Processing time: 94.284 seconds
```

[25]:
```
True
```

[26]:
```
path1 = 'dct256n50.jpg'
path2 = 'idct256n50.jpg'

imagem = abrir_imagem(path1)
imagem = np.array(imagem, dtype='int')
imagem  = np.clip(imagem, a_min = 0, a_max = 255)

imagem1 = abrir_imagem(path2)
imagem1 = np.array(imagem1, dtype='int')
```

```
imagem1  = np.clip(imagem1 , a_min = 0, a_max = 255)

fig, (ax_orig, ax_conv) = plt.subplots(1, 2)

ax_orig.imshow(imagem)
ax_orig.set_title('DCT')
ax_orig.set_axis_off()

ax_conv.imshow(imagem1)
ax_conv.set_title('IDCT')
ax_conv.set_axis_off()

fig.show()
```

```
<ipython-input-26-f9f612527ec5>:22: UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot
show the figure.
  fig.show()
```

0.5.8 Para n = 75% da imagem

[27]:
```
numberCoefficients = round(len(img)*0.75)

imgResult = dct_2d(img,numberCoefficients)
cv2.imwrite('dct256n75.jpg',imgResult)
```

```
Processing time: 72.725 seconds
```

[27]: True

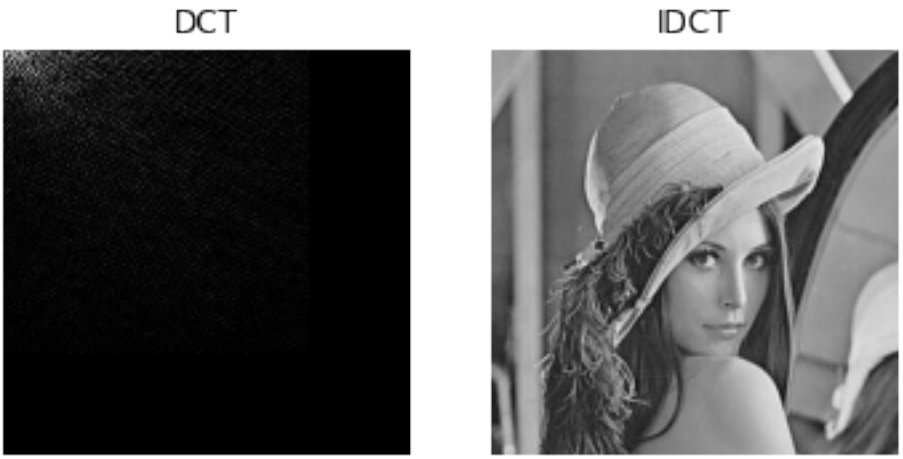
[28]:
```
idct_img = idct_2d(imgResult)
cv2.imwrite('idct256n75.jpg',idct_img)
```

```
Processing time: 95.061 seconds
```

[28]:
```
True
```

[29]:
```
path1 = 'dct256n75.jpg'
path2 = 'idct256n75.jpg'

imagem = abrir_imagem(path1)
imagem = np.array(imagem, dtype='int')
imagem  = np.clip(imagem, a_min = 0, a_max = 255)

imagem1 = abrir_imagem(path2)
imagem1 = np.array(imagem1, dtype='int')
imagem1  = np.clip(imagem1 , a_min = 0, a_max = 255)

fig, (ax_orig, ax_conv) = plt.subplots(1, 2)

ax_orig.imshow(imagem)
ax_orig.set_title('DCT')
ax_orig.set_axis_off()

ax_conv.imshow(imagem1)
ax_conv.set_title('IDCT')
ax_conv.set_axis_off()

fig.show()
```

```
<ipython-input-29-48694e5282ac>:22: UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot
show the figure.
  fig.show()
```



0.5.9 Para n = 100% da imagem

[30]:
```
numberCoefficients = len(img)

imgResult = dct_2d(img,numberCoefficients)
cv2.imwrite('dct256n100.jpg',imgResult)
```

```
Processing time: 72.111 seconds
```

[30]:
```
True
```

[31]:
```
idct_img = idct_2d(imgResult)
cv2.imwrite('idct256n100.jpg',idct_img)
```

```
Processing time: 94.708 seconds
```

[31]:
```
True
```

[32]:
```
path1 = 'dct256n100.jpg'
path2 = 'idct256n100.jpg'

imagem = abrir_imagem(path1)
imagem = np.array(imagem, dtype='int')
imagem  = np.clip(imagem, a_min = 0, a_max = 255)

imagem1 = abrir_imagem(path2)
imagem1 = np.array(imagem1, dtype='int')
```

```
imagem1  = np.clip(imagem1 , a_min = 0, a_max = 255)

fig, (ax_orig, ax_conv) = plt.subplots(1, 2)

ax_orig.imshow(imagem)
ax_orig.set_title('DCT')
ax_orig.set_axis_off()

ax_conv.imshow(imagem1)
ax_conv.set_title('IDCT')
ax_conv.set_axis_off()

fig.show()
```

```
<ipython-input-32-a8d0abe4805c>:22: UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot
show the figure.
  fig.show()
```

0.5.10 4. Discussão

Quanto ao processamento, podemos perceber que as operações, independentemente do caso de teste, o tempo foi bastante próximo, com diferenças de 1 à 2 segundos entre um caso e outro utilizando a mesma imagem e, também podemos dizer que o tempo de processamento foi razoável, dentro das capacidades atuais deste projeto.

Seria interessante uma análise da funcionalidade dct() feita e disponibilizada na biblioteca SciPy, de forma que possamos mensurar a diferença na qualidade da imagem bem como a eficiência da biblioteca quanto ao tempo de processamento em relação ao nosso, que os algoritmos foram baseados em equações fundamentais e introdutórias como aprendizado da disciplina.

A dificuldade foi encontrada durante a elaboração das DCTs e IDCTs 2D da forma tradicional,

sem separabilidade. Devido à sua complexidade, a construção da mesma não conseguiu ser feita, as tentativas de construção foram retiradas e desconsideradas na versão final e não foi implementada neste projeto por problemas quanto ao prazo de entrega.

Desta forma, esta dificuldade tornara-se um objetivo e sugestão para futura atualização deste projeto: utilizar as DCTs e IDCTs 2D de forma tradicional, comparar o processamento em relação com os casos feitos com separabilidade e, também, comparar com a função dct() do SciPy.

0.5.11 5. Conclusão

Enfim, dadas as ressalvas discutidas no tópico anterior, o projeto foi concretizado de maneira satisfatória. Conseguimos perceber as diferenças nos espectros visualizados nas imagens durante a DCT de cada caso, apesar de não haver compreensão da imagem durante este formato; obtivemos, com sucesso, a volta da imagem original em alguns casos, outros conseguimos perceber faltas de detalhes expressados como imagens borradas e, em outros casos (como n = 1 e n = 2), não conseguimos compreender a imagem durante a volta.

Dito isto, também podemos discutir que, durante os casos de teste feitos a partir dos percentuais do tamanho da imagem, a diferença quando utilizamos um n = 50% até 100% do tamanho da imagem é bem pequena, também presente na pequena diferença de processamento entre eles, também. A utilização de uma imagem com tamanho maior poderia expor uma diferença mais significativa durante esta análise, por exemplo.

Portanto, conseguimos expressar, de forma prática, os ensinamentos que nos foram dados durante esta disciplina. Trabalhamos e manipulamos imagens no domínio da frequência, estudamos suas diferentes transformadas e como elas podem ser utilizadas para o ramo do processamento digital, à exemplo deste trabalho.