Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных средств

**СИСТЕМЫ ОБРАБОТКИ МЕДИАДАННЫХ**

Лабораторная работа № 4

Обработка изображений в частотной области

Вариант № 14

группа № 850702

| Выполнил: | Турко В.Д. |
| --- | --- |
| Проверил | Рыбенков Е.В. |

Минск 2021

**1 ЦЕЛЬ РАБОТЫ**

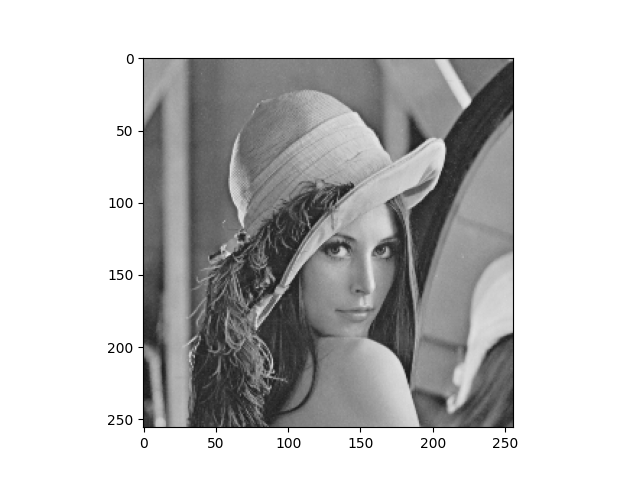
Научиться использовать дискретное косинусное преобразование для обработки изображений.

**2 ЗАДАНИЕ**

1. Загрузить изображение.
2. Выполнить анализ изображения с помощью ДКП.
3. Синтезировать изображение с помощью ДКП.
4. Сравнить собственную реализацию ДКП с реализацией Matlab.

**3 ХОД РАБОТЫ**

Для выполнения лабораторной работы был использован ЯП Python (v3.9), а в качестве исходного изображения было использовано Lena grayscale image.



**3.1 Использованные функции**

Для разбиения изображения на блоки была использована функция split\_in\_blocks, которая принимает исходное изображение, размер блока и функцию, которую необходимо применить к каждому блоку.

def split\_in\_blocks(img, bloc\_shape, block\_processor) -> np.ndarray:

h, w = img.shape

result = np.zeros(img.shape)

bloc\_h, bloc\_w = bloc\_shape

# size of resulting 2d array of block

result\_h = h // bloc\_h

result\_w = w // bloc\_w

for i in range(result\_h \* result\_w):

# coords of a block in 2d array of blocks

y = i // result\_w

x = i - y \* result\_w

# coresponding coords in original array

orig\_y = y \* bloc\_h

orig\_x = x \* bloc\_w

y\_slice = slice(orig\_y, orig\_y + bloc\_h)

x\_slice = slice(orig\_x, orig\_x + bloc\_w)

result[y\_slice, x\_slice] = block\_processor(img[y\_slice, x\_slice])

return result

Для перетасовки пикселей была использована функция shuffle:

def shuffle(img, bloc\_shape) -> np.ndarray:

h, w = img.shape

result = np.zeros(img.shape)

bloc\_h, bloc\_w = bloc\_shape

# size of resulting 2d array of block

result\_h = h // bloc\_h

result\_w = w // bloc\_w

y\_step = result\_h

x\_step = result\_w

for i in range(result\_h \* result\_w):

# coords of a block in 2d array of blocks

y = i // result\_w

x = i - y \* result\_w

# coresponding coords in original array

orig\_y = y \* bloc\_h

orig\_x = x \* bloc\_w

y\_slice = slice(orig\_y, orig\_y + bloc\_h)

x\_slice = slice(orig\_x, orig\_x + bloc\_w)

result[y\_slice, x\_slice] = img[y::y\_step, x::x\_step]

return result

Функции, реализующие прямое и обратное ДКП:

def a(x, n):

return math.sqrt((1 if x == 0 else 2)/n)

def dct2(bloc) -> np.ndarray:

"""Applies dct to the selected bloc of image"""

M, N = bloc.shape

result\_bloc = np.zeros(bloc.shape)

for coord, v in np.ndenumerate(result\_bloc):

p, q = coord

for coord, v in np.ndenumerate(bloc):

m, n = coord

cos\_mp = math.cos(math.pi \* (2 \* m + 1) \* p / (2 \* M))

cos\_nq = math.cos(math.pi \* (2 \* n + 1) \* q / (2 \* M))

result\_bloc[p][q] += bloc[m][n] \* cos\_mp \* cos\_nq

result\_bloc[p][q] \*= a(p, M) \* a(q, N)

return result\_bloc

def idct2(bloc) -> np.ndarray:

"""Applies idct to the selected bloc of image"""

M, N = bloc.shape

result\_bloc = np.zeros(bloc.shape)

for coord, v in np.ndenumerate(result\_bloc):

m, n = coord

for coord, v in np.ndenumerate(bloc):

p, q = coord

cos\_mp = math.cos(math.pi \* (2 \* m + 1) \* p / (2 \* M))

cos\_nq = math.cos(math.pi \* (2 \* n + 1) \* q / (2 \* M))

result\_bloc[m][n] += a(p,M) \* a(q,N) \* bloc[p][q] \* cos\_mp \* cos\_nq

return result\_bloc

**3.2 Использование реализации ДКП из scipy**

def lambda\_dct2(x): return dctn(x, norm='ortho')  
def lambda\_idct2(x): return idctn(x, norm='ortho')

img\_dct = split\_in\_blocks(img,(8, 8), lambda\_dct2)

shuffeled = shuffle(img\_dct,(32, 32))  
img\_idct = split\_in\_blocks(shuffle(shuffeled, (8, 8)), (8, 8), lambda\_idct2)

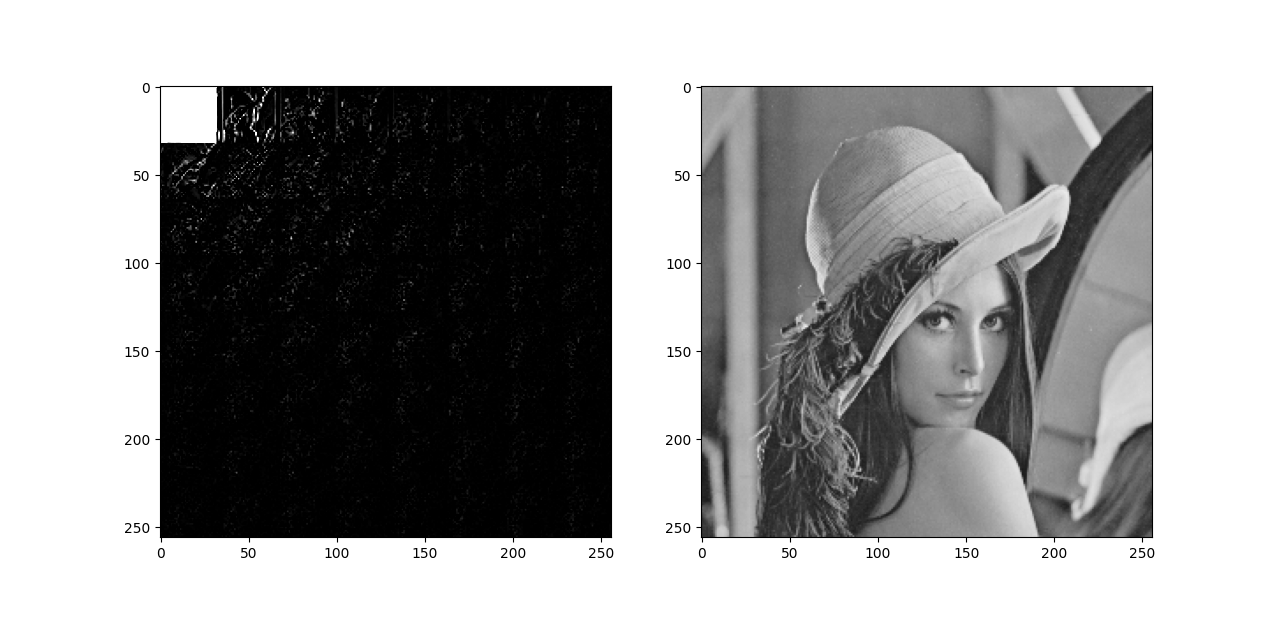
plt.subplot(1, 2, 1)

plt.imshow(shuffeled, cmap='gray', vmin=0, vmax=255)

plt.subplot(1, 2, 2)

plt.imshow(img\_idct, cmap='gray', vmin=0, vmax=255)

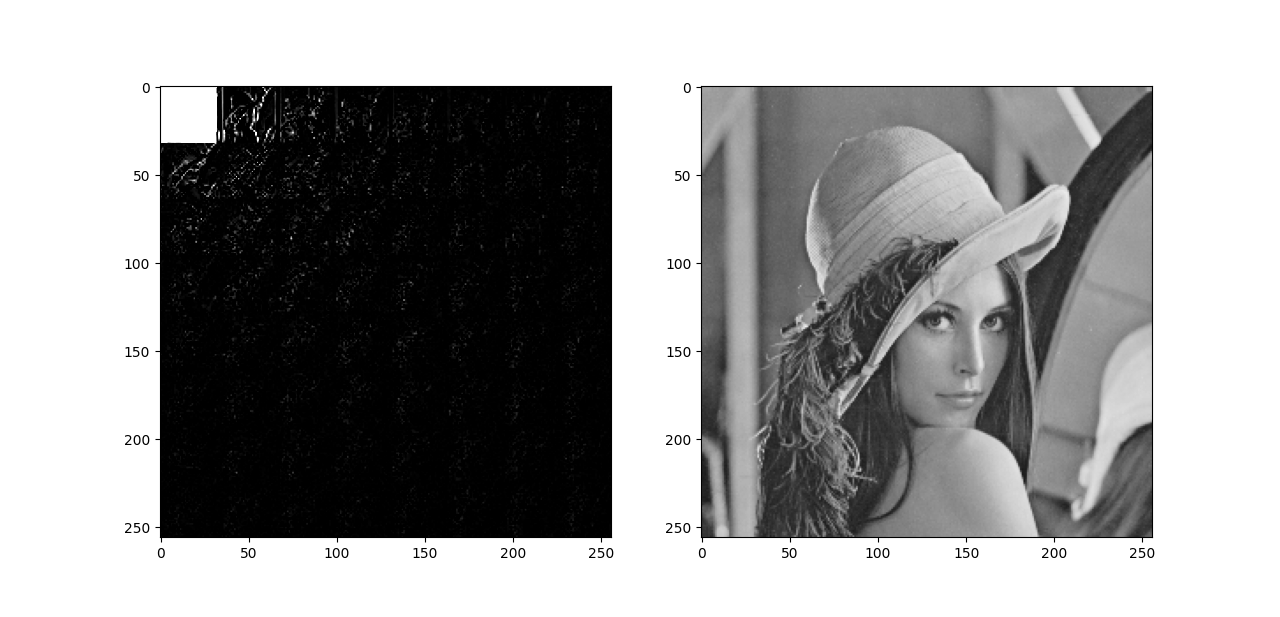
plt.show()



**3.3 Использование собственной реализации ДКП**

img\_dct = split\_in\_blocks(img, (8, 8), dct2)  
shuffeled = shuffle(img\_dct, (32, 32))  
img\_idct = split\_in\_blocks(shuffle(shuffeled, (8, 8)), (8, 8), idct2)

plt.subplot(1, 2, 1)  
plt.imshow(shuffeled, cmap='gray', vmin=0, vmax=255)  
plt.subplot(1, 2, 2)  
plt.imshow(img\_idct, cmap='gray', vmin=0, vmax=255)  
plt.show()



**4 ВЫВОД:**

В результате выполнения лабораторной работы мной был реализовано прямое и обратное дискретное косинусное преобразование на языке программирования Python 3.9, которое пусть и проигрывает в скорости выполнения реализации из пакета scipy, но при этом дает схожий результат.