Пыхтин. М К3242

Импортируем библиотеки

```
In [5]: import cv2
        import numpy as np
        import math
```

Считываем изображение и выполняем DCT, применяем квантование (стандартная матрица jpeg) и кодируем. И эти данные записываем в текстовый файл(txt).

```
# Размер блоков
block_size = 8

# Матрциа Квантования
QUANTIZATION_MAT = np.array([[16,11,10,16,24,40,51,61],[12,12,14,19,26,58,60,55],[14,13,16,24,40,57,69,56 ],[14,17,22,29

img = cv2.imread('14.jpg', cv2.IMREAD_GRAYSCALE)

[h , w] = img.shape
height = h
width = w
h = np.float32(h)
w = np.float32(w)

nbh = math.ceil(h/block_size)
nbh = np.int32(nbh)

nbw = math.ceil(w/block_size)
nbw = np.int32(nbw)
```

```
# Начинаем делить изображение на блоки, и применяем коэффиценты матрицы в зиг-заг порядке
for i in range(nbh):
        row_ind_1 = i*block_size
        row_ind_2 = row_ind_1+block_size

        for j in range(nbw):
            col_ind_1 = j*block_size
            col_ind_2 = col_ind_1+block_size

            block = padded_img[ row_ind_1 : row_ind_2 , col_ind_1 : col_ind_2 ]
            DCT = cv2.dct(block)
            DCT_normalized = np.divide(DCT,QUANTIZATION_MAT).astype(int)
            reordered = zigzag(DCT_normalized)
            reshaped= np.reshape(reordered, (block_size, block_size))
            padded_img[row_ind_1 : row_ind_2 , col_ind_1 : col_ind_2] = reshaped

cv2.imshow('encoded image', np.uint8(padded_img))

arranged = padded_img.flatten()

# записываем все в txt файл, чтобы можно было бы раскодировать в дальнейшем
bitstream = get_run_length_encoding(arranged)

bitstream = str(padded_img.shape[0]) + " " + str(padded_img.shape[1]) + " " + bitstream + ";"

file1 = open("image.txt","w")
file1.write(bitstream)
file1.close()

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Затем мы читаем записанный нами же текстовый файл, для того чтобы записать наше новое сжатое изображение.

```
#декодинг
QUANTIZATION_MAT = np.array([[16,11,10,16,24,40,51,61],[12,12,14,19,26,58,60,55],[14,13,16,24,40,57,69,56 ],[14,17,22,29
block_size = 8
# читаем наш txt файл
with open('image.txt', 'r') as myfile:
    image=myfile.read()
details = image.split()
h = int(''.join(filter(str.isdigit, details[0])))
w = int(''.join(filter(str.isdigit, details[1])))
array = np.zeros(h*w).astype(int)

k = 0
i = 2
x = 0
j = 0
```

```python
# Новый размер изображения

while k < array.shape[0]:
    if(details[i] == ';'):
        break
    if "-" not in details[i]:
        array[k] = int(''.join(filter(str.isdigit, details[i])))
    else:
        array[k] = -1*int(''.join(filter(str.isdigit, details[i])))
    if(i+3 < len(details)):
        j = int(''.join(filter(str.isdigit, details[i+3])))
    if j == 0:
        k = k + 1
    else:
        k = k + j + 1
    i = i + 2
array = np.reshape(array,(h,w))
i = 0
j = 0
k = 0
padded_img = np.zeros((h,w))

while i < h:
    j = 0
    while j < w:
        temp_stream = array[i:i+8,j:j+8]
        block = inverse_zigzag(temp_stream.flatten(), int(block_size),int(block_size))
        de_quantized = np.multiply(block,QUANTIZATION_MAT)
        padded_img[i:i+8,j:j+8] = cv2.idct(de_quantized)
        j = j + 8
    i = i + 8

# 8бит
padded_img[padded_img > 255] = 255
padded_img[padded_img < 0] = 0
cv2.imwrite("compressed_image.jpg",np.uint8(padded_img))
```

Мы записали наше изображение и теперь нам необходимо оценить результат нашей работы. Для этого воспользуемся функциями энтропии и psnr(Пиковое отношение сигнала к шуму).

In [9]:
```python
def entropy(img):
    frcy = np.array([0 for i in range(256)])
    for row in img:
        for px in row:
            frcy[px] += 1

    n = len(img) * len(img[0])
    frcy = frcy / n
    ent = -np.sum([p * np.log2(p) for p in frcy if p != 0])
    return ent
def psnr(img1, img2):
    rmse = np.sqrt(np.sum(np.power(img1 - img2, 2)) / img1.shape[0] / img1.shape[1])
    if rmse == 0:
        return 100
    pxmax = 255.0
    return 20 * math.log10(pxmax / math.sqrt(rmse))
```

In [11]:
```python
entropy(a) # исходник
```

Out[11]: 7.634483223644974

In [12]:
```python
entropy(c) # итог
```

Out[12]: 7.5159972039605698

In [26]:
```python
d = cv2.imread("compressed_image.jpg")
psnr(img, c) # сравниваем обработанный и исходный
```

Out[26]: 42.39518804707859

In [25]:
```python
psnr(a,d)# png сравниваем исходные и сохранившую
```

Out[25]: 40.010095149422035

Вывод: Таким образом, показатели несколько отличаются, но не очень сильно. Из этого следует, что алгоритм справился со своей задачей и почти не уступает по качеству работы простому встроенному алгоритму сохранения изображения в формат JPEG. Я научился применять вручную алгоритм JPEG, который смог сжать изображение, немного ухудшив его качество.

Изображения привоу ниже.

Исходник