Advanced Digital Signal Processing
Final Project - Python Code Description

B08901155 田祐行

1 JPEG Compression

Main function usage:

```
data, code, dim, mode = JPEG_compress(img)
img = JPEG_extract(data, code, dim, mode)
```

- `img`: 3D Numpy array
  Numpy array of image file. Dimension should be (xxx, xxx, 3).
- `data`: Python list
  The compressed data of the image.
- `code`: Python dictionary
  The mapping rule for Huffman coding. Maps integers to strings of binary numbers.
- `dim`: Python list
  Dimension informations required for JPEG extraction.
- `mode`: integer
  YCbCr compression mode (decided in the function). Possible values are 444, 422, 420.

Figure 1, 2 are the original image and the image that is compressed and recovered. The compression rate cannot be controlled yet, so figure 2 looks blurry due to high compression rate.

Figure 3 shows the compressed data size. For the image in figure 1 with size 599×800, the data after compression (`data` only, not including other outputs) contains about half the number of numbers in the original image array. Figure 3 also shows the data size in bits and bytes. Thanks to Huffman coding, the number of bits is the same order of the number of data numbers.

Because I do not know how to write a standard JPEG file, the compressed data and informations are saved in a json file. All bits are represented by strings of 0, 1, so the json file size is larger than the original image file. For the case of figure 1, the sizes of the original image (PNG file) and the json file are 443.3 kB and 3.6 MB respectively.

The two main functions `JPEG_compress`, `JPEG_extract` are written in the file `JPEG.py`. The result in figure 3 is generated by `jpeg_data.py`.

Figure 1: original image

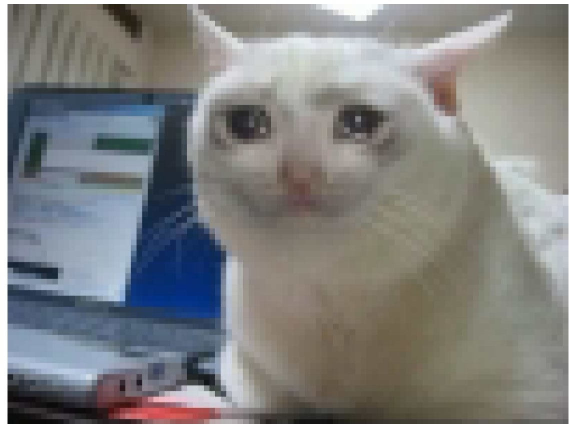

Figure 2: compressed and recovered image

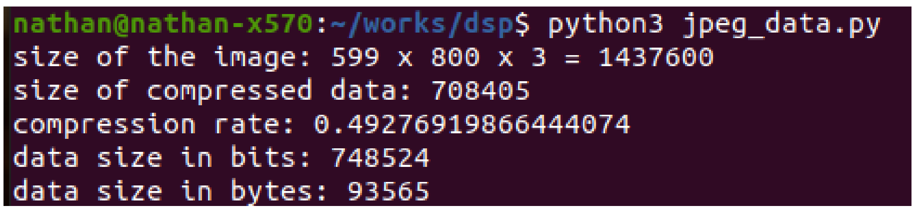

Figure 3: data after JPEG compression

2 Prime Factor Algorithm

Main function usage:

```
X = prime_factor_dft(x)
```

- `x`: 1D Numpy array
  The target signal for DFT.
- `X`: 1D Numpy array
  The result of the DFT on `x`.

Figure 4 is the result of my program. For a signal with length 3500, it takes about 0.08 seconds to finish the DFT. Computing the DFT directly take about 0.41 seconds, which is roughly 5 times the time using prime factor algorithm.

The average of the absolute error between two methods has an order of 10^{-9}, so I think it is presice enough in most cases.

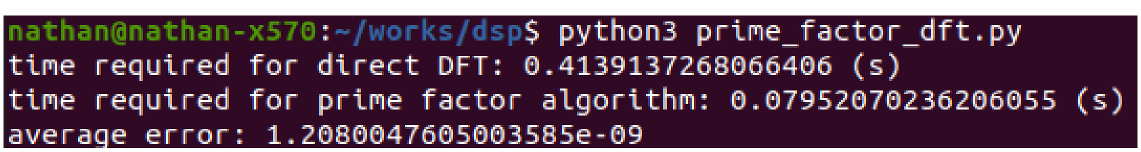

Figure 4: result of the DFT on a signal with length 3500.