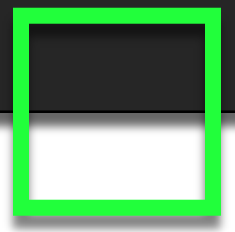
#수치 해석
#과제 9

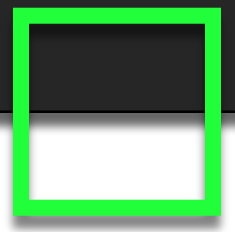
Discrete Cosine Transform

컴퓨터소프트웨어학부
2018008395 박정호



#1

구현



설명에 앞서…

이번 과제는 JPEG에 이용된 DCT에 대한 이해를 위한 과제로, 구현할 모델은 이미 과제 명세에 제공되어 있었다. 따라서, 문제를 해결하기 위한 모델링에 대해서는 설명하지 않겠다. 후술할 내용은 단순히 필자가 구현한 코드의 개요이다.

또한 코드의 개요를 설명하기 전에 사용한 라이브러리를 밝히겠다.
DCT와 IDCT는 scipy의 fftpack에 포함된 dct, idct라는 메소드를 사용했고, 이미지 프로세싱을 위해 PIL을, 기본적인 수학적 연산을 위해 numpy를 사용했다. 모두 오픈 라이브러리이다.

2차원 DCT와 IDCT의 구현

앞에서 DCT와 IDCT를 라이브러리 함수로 사용했다고 했지만, scipy에서 제공하는 메소드는 1차원의 DCT였다.
따라서 필자는 이 라이브러리를 이용해서 2차원 DCT와 IDCT를 구현해야 했다.

```
def dct2(block):
    return dct(dct(block.T, norm = 'ortho').T, norm = 'ortho')

def idct2(block):
    return idct(idct(block.T, norm = 'ortho').T, norm = 'ortho')
```

이렇게 DCT(IDCT)를 두 번 중첩해서 사용하는 것으로, 간단하게 2차원 DCT(IDCT)를 구현했다.
수식이 주어져 있으므로, 직접 계산했어도 되겠지만, 라이브러리를 이용하는 것이 더 연산이 빠를 것이라고 생각해서 이렇게 진행했다.

여기서 block이란, 이미지에서 16*16픽셀의 블록을 의미한다.

Quantize의 구현

과제 명세대로 가장 큰 coefficient 16개를 선택하도록 했다.

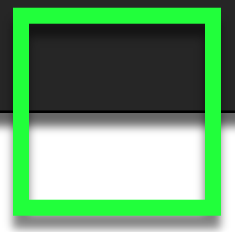
다만, 만약 중복되는 값이 존재해서 coefficient의 대소 비교로는 16개를 고를 수 없을 경우에는 다음과 같은 기준을 적용했다.

1. X축 방향 주파수가 작은 순으로 coefficient를 남긴다.
2. X축 방향 주파수가 같은 경우, Y축 방향 주파수가 작은 순으로 coefficient를 남긴다.

엄밀하게 말해서는 저주파항만 남기는 것은 아니다. 아마 coefficient matrix 상에서 첫번째 row에 non-zero인 값이 몰리게 될 것이다. 다만, for loop의 방향에 있어서 이 방식이 더 구현이 간단했기에 필자는 이 방식을 채택했다.

#2

결과와 비교



첫번째 이미지의 결과

원본 이미지

복원한 이미지

배경을 제외하고는 거의 검은색이 전부인 사진이다.
중앙과, 우측에 위치한 나뭇가지에는 어느 정도의 왜곡이 발생했으나, 하늘이나 여타 영역에는 왜곡의 정도가 그리 크지 않은 것을 볼 수 있다.

두번째 이미지의 결과

원본 이미지

복원한 이미지

배경이 어느정도 일정한 색을 가진 사진이다.
이로 인해 앞의 사진보다는 배경에서 보이는 블록의 경계가 옅은 것을 확인할 수 있다. 다만, 다양한 색상이 포함된 중앙의 다리나, 명암 차이가 많은 우측 전면의 식물에는 왜곡이 심한 것을 확인할 수 있다.

세번째 이미지의 결과

원본 이미지

복원한 이미지

배경이 좁고, 복잡한 영역이 더 많은 사진이다.
역시나 16개로 양자화된 coefficient로 인해 왜곡이 심한 것을 볼 수 있다. 특히 경계에 위치한 인물들의 경우 마치 모자이크처럼 일그러져 있는 것을 확인할 수 있다.

이미지를 통한 결과 분석

각 블록에서 coefficient 를 16개씩 남겼으니, 당연하게도 각 블록에서 약간의 왜곡이 일어났고, 이로 인해서 블록 사이의 경계가 명확하게 보였다. 블록 간의 차이가 명확할 수록 그 경계도 명확하게 보이는 것을 알 수 있었다.

또한 세번째, 두번째, 첫번째 이미지 순으로 이미지에서 복잡한 부분이 차지하는 범위가 넓었는데, 육안으로 보이는 원본과 복원본의 차이도 동일한 순서대로 높은 것을 볼 수 있었다.

이러한 왜곡의 원인은 모든 블록에 대해서 quantize할 범위를 동일한 16으로 잡았기 때문이다. 각 블록마다 복잡한 정도가 다르고, 복잡한 블록은 당연히 주파수가 클 수밖에 없는데, 양자화로 인해 저주파항의 coefficient만 남게 되기 때문에 복원된 블록의 복잡도가 감소할 수밖에 없어진 것이다.

실제로 각 블록의 복잡도에 따른 왜곡의 정도를 다음 슬라이드에서 살펴보자.

블록의 복잡도에 따른 왜곡의 정도


외관상 단순한 블록들과 복잡한 블록들의 복원 전후 비교, 오른쪽의 이미지가 복원본이다.

첫번째 이미지에서 같은 부분을 잘라서 확대했다.

단순한 블록들의 경우, 거의 차이를 못 느끼고, 경계도 흐린 반면에, 복잡한 블록들의 경우 한눈에 봐도 차이가 확실하게 보이며, 블록들 사이의 경계도 명확함을 확인할 수 있었다.

실제로 JPEG에서 사용하는 것과 같이 각 블록에 따라 quantize하는 정도를 다르게 한다면, 좀 더 나은 복원본을 얻을 수 있었으리라 생각한다.