# 영상 처리
# - JPEG Compression -

| | |
|---|---|
| **제출일자** | 2021.05.25 |
| **분 반** | 01 |
| **이 름** | 강인한 |
| **학 번** | 201701969 |

| 1 | 구현 코드 및 설명 |
|---|---|

```
def img2block(src, n=8):
    blocks= []
    h,w = src.shape
    for i in range(h//8):
        for j in range(w//8):
            blocks.append(src[n*i:(n*i+8),j*n:(j*n+8)])
    return np.array(blocks).astype(np.int32)
```

encoding 하는 과정에서 이미지를 각각의 블록으로 쪼개 리스트에 append하고 np.array형태로 변환하는 과정이다.

```
def DCT(block, n=8):

    dst = np.zeros((n,n))
    y, x = np.mgrid[0:n, 0:n]
    for v_ in range(n):
        for u_ in range(n):
            tmp = block * np.cos(((2*x+1)*u_*np.pi)/(2*n)) * np.cos(((2*y+1)*v_*np.pi)/(2*n))
            dst[v_,u_] = np.sum(tmp) * C(v_,n=n) * C(u_,n=n)
    return np.round(dst)
```

DCT함수는 9주 차에 실습했던 것과 같다. 4중 for문 대신 sum 함수를 이용하여 구현하였다.

```
def C(w, n = 8):
    if w == 0:
        return (1/n)**0.5
    else:
        return (2/n)**0.5
```

C함수는 다음과 같다.

```
else:
    z=[]
    for k in range(8):
        i=0
        j=k
        if k%2==0:
            j=0
            i=k
            for n in range(k+1):
                z.append(block[i,j])
                i-=1
                j+=1
        else:
            for n in range(k+1):
                z.append(block[i,j])
                i+=1
                j-=1
    for k in range(7):
        i=7
        j=k+1
        if (k+1)%2==0:
            i=k+1
            j=7
            for n in range(7-k):
                z.append(block[i,j])
                i+=1
                j-=1
        else:
            for n in range(7-k):
                z.append(block[i,j])
                i-=1
                j+=1
```

my_zigzag_scanning함수에서 mode가 encoding일 경우의 코드는 위와 같다. 대각선을 차례로 z에 append하는 방법으로 이루어진다. 위에 있는 for문을 통해 왼쪽 위의 삼각형에 해당하는 배열을 append하고 아래의 for문을 통해 오른쪽 아래 있는 삼각형에 해당하는 배열을 append하게 된다.

```
        k=63
        while (z[k]==0):
            z.pop()
            k-=1
            if k==0:
                break
```

뒤부터 탐색하여 0인 값을 pop하는 과정이다.

zigzag함수에서 decoding의 경우도 enconding의 경우와 비슷하다.

윗삼각형과 아래 삼각형을 구분해서 차례로 block에 들어있는 값을 반대로 배열에 넣어주는 작업을 하게 된다.

```
def DCT_inv(block, n = 8):
    ##################################################
    # TODO                                           #
    # DCT_inv 완성                                    #
    # DCT_inv 는 DCT와 다름.                           #
    ##################################################
    dst = np.zeros((n,n))
    v_, u_ = np.mgrid[0:n, 0:n]

    for i in range(n):
        for j in range(n):
            block[i,j] = block[i,j] * C(i,n=n) * C(j,n=n)
    for y in range(n):
        for x in range(n):
            tmp = block *np.cos(((2*x+1)*u_*np.pi)/(2*n)) * np.cos(((2*y+1)*v_*np.pi)/(2*n))
            dst[x,y] = np.sum(tmp)
    return np.round(dst)
```

DCT_inv함수는 위와 같다.

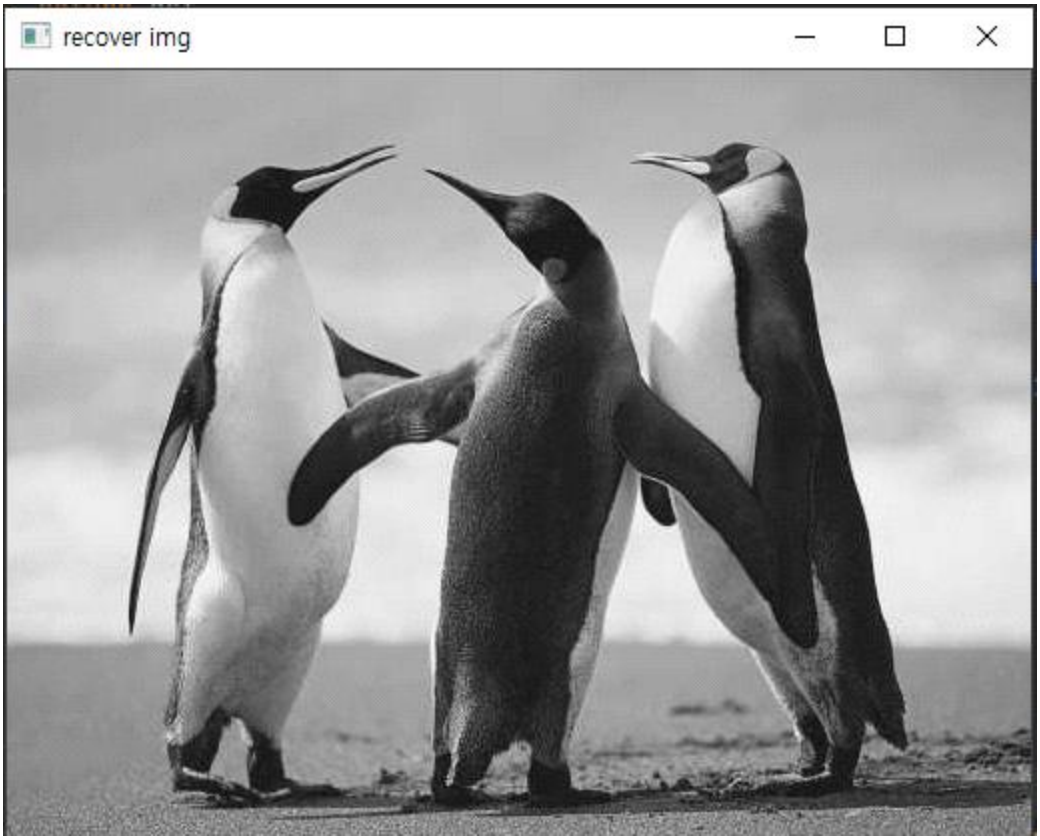
C(i)와 C(j)를 block에 사전에 곱해주는 작업이 필요하다.

```python
def block2img(blocks, src_shape, n = 8):
    ###################################################
    # TODO                                            #
    # block2img 완성                                   #
    # 복구한 block들을 image로 만들기                    #
    ###################################################
    h,w = src_shape
    dst= np.zeros((h,w))

    k=0
    for i in range(h//8):
        for j in range(w//8):
            dst[n*i:(n*i+8),j*n:(j*n+8)]= blocks[k]
            k+=1
    return dst.astype(np.int32)
```

블록들을 다시 이미지로 복원하는 과정이 필요하다. 블록들의 index를 차례로 알맞은 위치에 넣어준다.

| 2 | 결과 이미지 |
|---|---|




| 3 | 느낀 점 및 과제 난이도 |
|---|---|

jpeg의 압축과 복원 방법이 재밌었고 어려울 것 같았지만 단계별로 차근차근 강의해주셔서 잘 따라올 수 있었던 것 같습니다. 함수작성에 있어 높은 난도를 요구하진 않았지만, 지그재그 함수를 구현하기가 조금 어려웠던 것 같습니다. 알고리즘 실력이 부족해서 그런 것 같습니다.