수치해석
HW9

2016025478 컴퓨터전공 서경욱

개발 환경

- Python 3.8.0

- 사용한 Library

- Numpy (cosine 계산, 행렬 계산, 원주율, 제곱근 등)
- Opencv (이미지 파일 처리, RGB split, merge, image 출력)

DCT 구현

```
def dct(Color):
    F = [[0 for _ in range(16)] for _ in range(16)]
    for v in range(16):
        for u in range(16):
            _sum = 0
            for x in range(16):
                for y in range(16):
                    _sum += float(Color[x][y]) * np.cos((v*np.pi) * (2*x + 1) / (2 * float(N))) * \
                        np.cos((u*np.pi) * (2*y + 1) / (2 * float(N)))
            F[v][u] = _sum * getC(u,v) * (1/8)

    F = np.array(F)
    tmp = np.ravel(list(map(abs,F)))
    tmp = np.sort(tmp)[::-1]
    cut = tmp[15]

    cnt = 0
    for i in range(16):
        for j in range(16):
            if cnt == 16:
                F[i][j] = 0
            elif abs(F[i][j]) < cut:
                F[i][j] = 0
            else:
                cnt += 1

    return F
```

- F_{vu}를 구하는 함수입니다.
- $C_v * C_u$ 는 getC라는 함수로 구해주고 16*16 block이므로 (1/8)로 scale 해줍니다.
- F를 다 구한뒤 np.ravel 함수로 1차원 배열로 만들어 준뒤 큰 순서로 sort 해줍니다. 그 후 16번째 factor를 기준으로 16*16 block을 순회하며 해당 factor보다 **작은 값들을 0으로** 만들고 0이 아닌 값이 **16개가 되었다면 나머지를 모두 0으로** 만들어줍니다.(Energy Compaction)

IDCT 구현

```
def idct(F):
    S = [[0 for _ in range(16)] for _ in range(16)]
    for x in range(16):
        for y in range(16):
            _sum = 0
            for v in range(16):
                for u in range(16):
                    _sum += getC(u,v) * float(F[v][u]) * np.cos((v*np.pi) * (2*x + 1) / (2 * float(N))) * \
                        np.cos((u*np.pi) * (2*y + 1) / (2 * float(N)))
            S[x][y] = _sum * (1/8)

    return S
```

- S_{yx} 를 구하는 과정입니다. (다만 코드에서는 x,y를 바꿔서 구현했습니다.)
- 이번에는$\sum_{v=0}^{N-1}\sum_{u=0}^{N-1}$ 식 안에 $C_v{*}C_u$ 가 있으므로 **안쪽 for문 안에 getC로 $C_v{*}C_u$** 를 구해 곱해줍니다.

처리 과정

```
for j in range(0,max_j,16):
    for k in range(0,max_k,16):
        tmp = dct(R[j:j+16,k:k+16])
        tmp = idct(tmp)
        for p in range(16):
            for q in range(16):
                R[j+p][k+q] = tmp[p][q]
        tmp = dct(G[j:j+16,k:k+16])
        tmp = idct(tmp)
        for p in range(16):
            for q in range(16):
                G[j+p][k+q] = tmp[p][q]
        tmp = dct(B[j:j+16,k:k+16])
        tmp = idct(tmp)
        for p in range(16):
            for q in range(16):
                B[j+p][k+q] = tmp[p][q]
        print("%d is done!!" % k)
    print("[j]%d is done!!" % j)
for j in range(max_j):
    for k in range(max_k):
        if R[j][k] < 0: R[j][k] = 0
        elif R[j][k] >255: R[j][k] = 255
        if G[j][k] < 0: G[j][k] = 0
        elif G[j][k] >255: G[j][k] = 255
        if B[j][k] < 0: B[j][k] = 0
        elif B[j][k] >255: B[j][k] = 255
merged = cv2.merge([B,G,R])
```

```
img = cv2.imread(join(img_path,files[i]),1)
(B, G, R) = cv2.split(img)
```

- image 파일을 imread로 flag 1을 줘서 color(RGB) 값으로 얻어옵니다. 그리고 split을 통해 **R,G,B 각각의 pixel값**들을 행렬로 받아옵니다.

- 해당 image의 크기만큼 16단위로 for 문을 돕니다. 그러면서 image의 모든 **16*16 block에 R,G,B 각각 dct, idct처리를** 해줍니다.

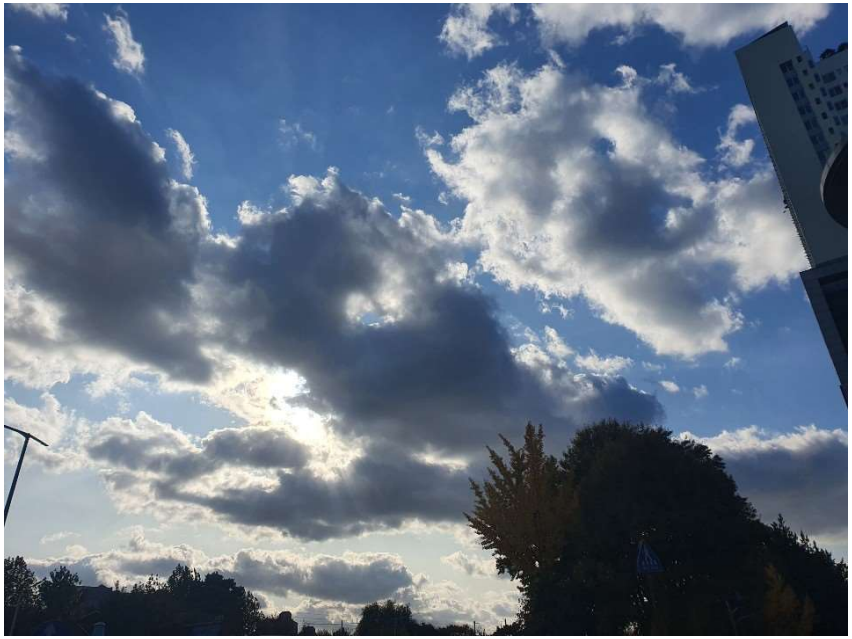
- 연산이 다 끝난 뒤에는 0보다 작은 값은 0으로 255보다 큰 값은 255로 보정을 실시하고 merge를 통해 하나의 배열로 합쳐줍니다.

- 이후 사진을 출력해봄으로써 기존 사진과의 비교를 진행했습니다.

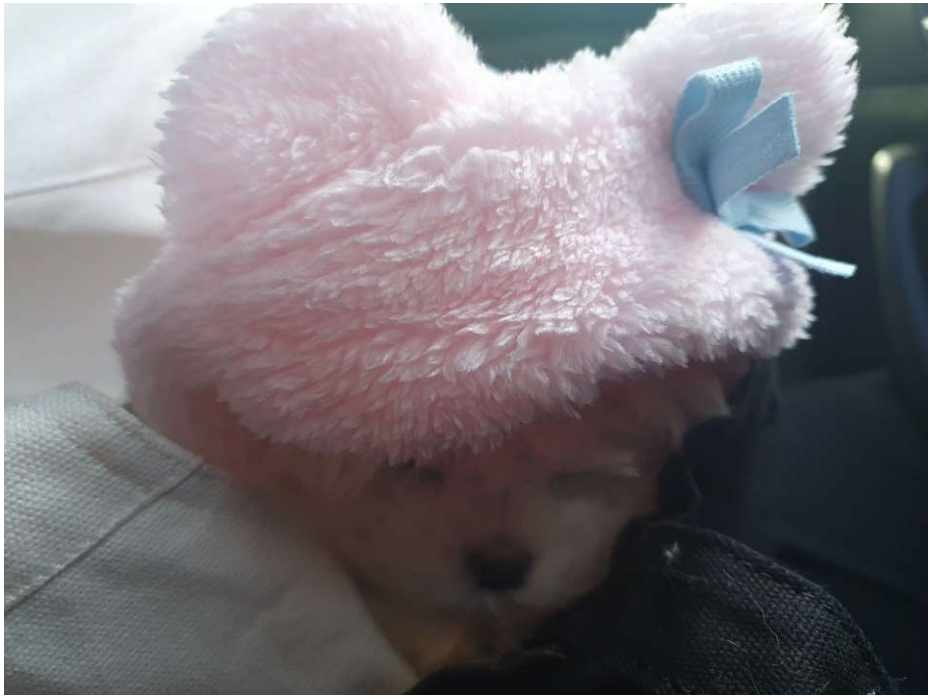
사용한 3 images

-

1.jpg(256*256)

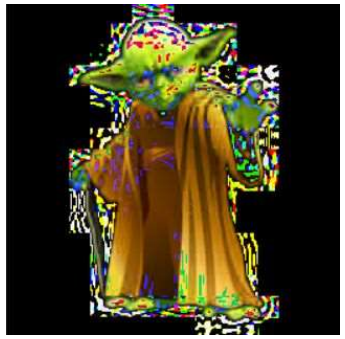


2.jpg(720*960)

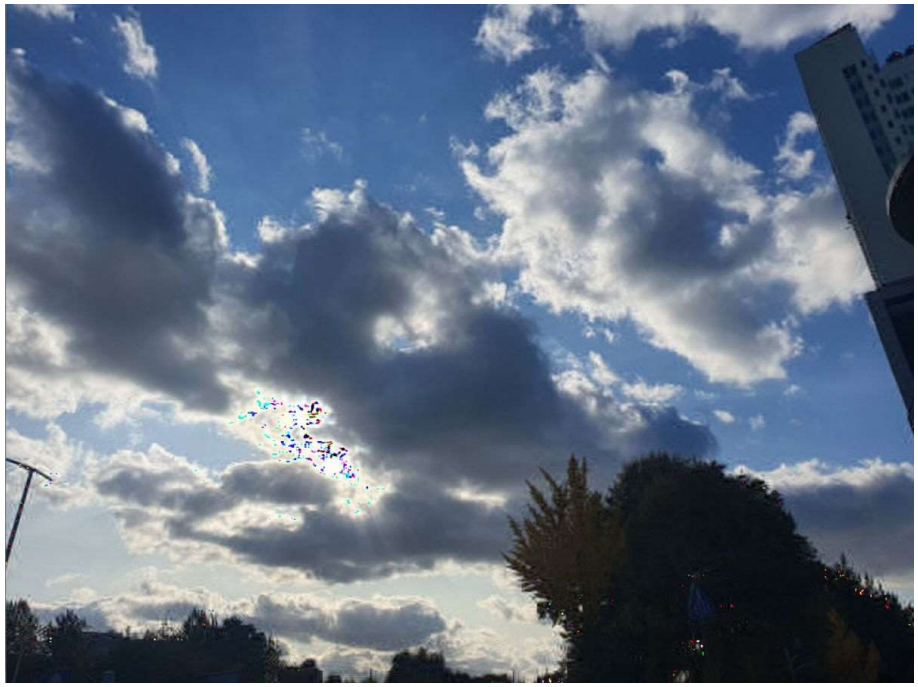


3.jpg(720*960)

변환된 3 images

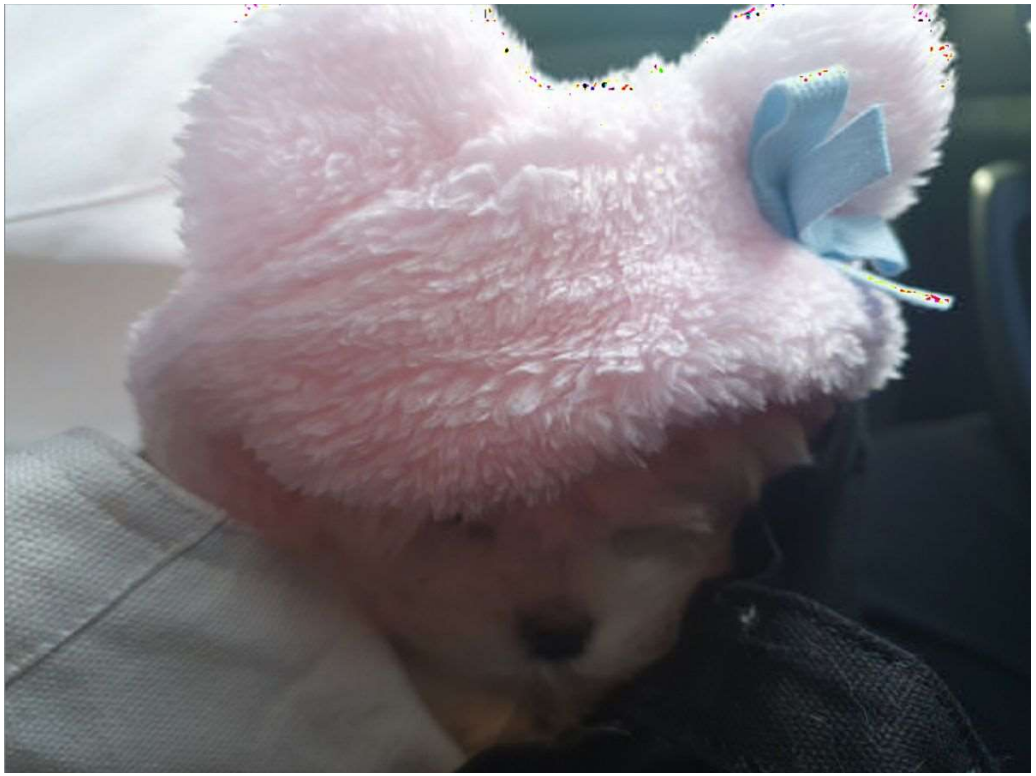


1.jpg(256*256)



2.jpg(720*960)

-



3.jpg(720*960)