

# Image Processing

## 실습 9.

2021. 05. 02.

# 실습 수업 소개

- 과목 홈페이지
  - 충남대학교 사이버 캠퍼스 ( <http://e-learn.cnu.ac.kr> )
- TA 연락처
  - 신준호
  - wnsgh578@naver.com
- 튜터 연락처
  - 한승오
  - so.h4ns@gmail.com
- 실습 중 질문사항
  - 수업중 질문 or 메일을 통한 질문
  - 메일로 질문할 때 [IP] 를 제목에 붙여주세요

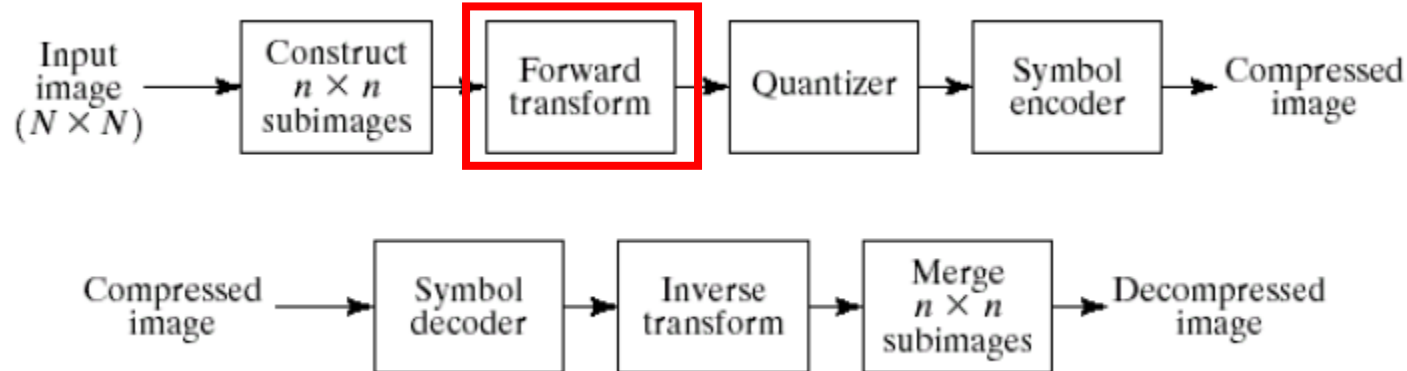
# 목 차

- 과제
  - DCT 수행 및 mask 시각화

# Discrete cosine transform (DCT)

- DCT (이산 코사인 변환)

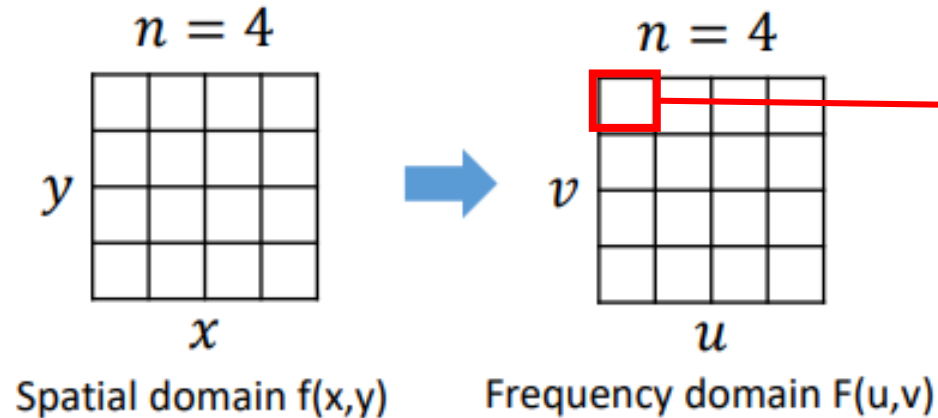
- 공간도메인에서 주파수 도메인으로 바꿀 때 JPEG에서는 DCT를 이용해서 바꾼다.



# Discrete cosine transform (DCT)

## • DCT (이산 코사인 변환)

- 공간도메인에서 주파수 도메인으로 바꿀 때 JPEG에서는 DCT를 이용해서 바꾼다.



$$F(0, 0) = \frac{1}{4} \sum_{y=0}^{n-1} \sum_{x=0}^{n-1} f(x, y)$$

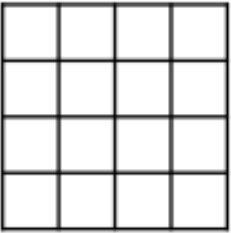
$F(0,0)$ 의 주파수 값 : 전반적인 밝기(명도)  
 나머지 주파수 값 : 전반적인 밝기를 기준으로 얼마나 변화하는지

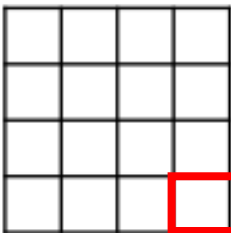
$$F(u, v) = C(u)C(v) \sum_{y=0}^{n-1} \sum_{x=0}^{n-1} f(x, y) \cos\left(\frac{(2x+1)u\pi}{2n}\right) \cos\left(\frac{(2y+1)v\pi}{2n}\right)$$

$$C(w) = \begin{cases} \sqrt{1/n} & \text{if } w = 0 \\ \sqrt{2/n} & \text{otherwise} \end{cases}$$

# Discrete cosine transform (DCT)

- DCT (이산 코사인 변환)

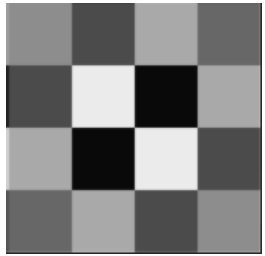
$n = 4$   
  
 Spatial domain  $f(x,y)$

$n = 4$   
  
 Frequency domain  $F(u,v)$

$C(u) * C(v) * np.sum$

$$F(u, v) = C(u)C(v) \sum_{y=0}^{n-1} \sum_{x=0}^{n-1} f(x, y) \cos\left(\frac{(2x+1)u\pi}{2n}\right) \cos\left(\frac{(2y+1)v\pi}{2n}\right)$$

$C(w) = \begin{cases} \sqrt{1/n} & \text{if } w = 0 \\ \sqrt{2/n} & \text{otherwise} \end{cases}$



dct mask

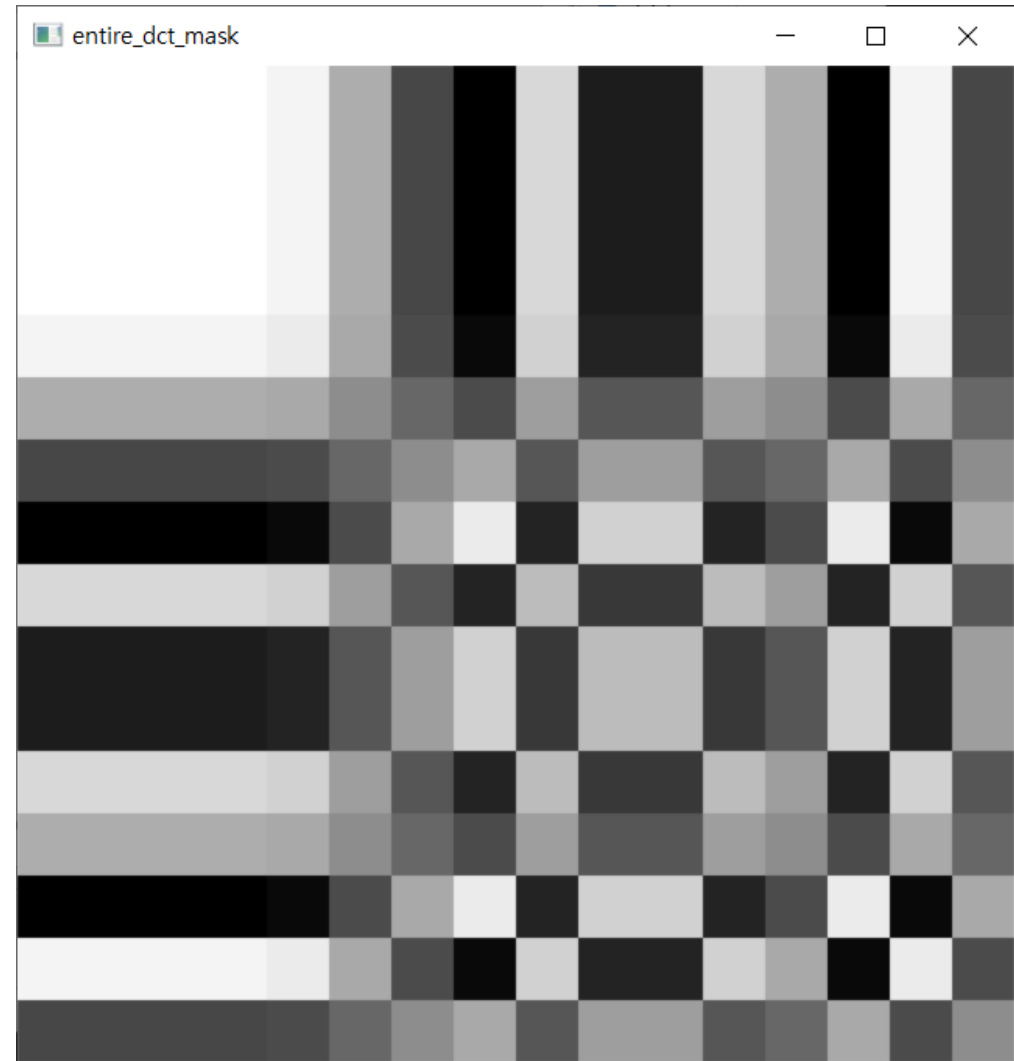
# 과제

- DCT mask 완성 및 수행

```
[[0.53731403 0.47336688 0.50497433 1.      ]  
 [0.60318716 0.71475572 0.60758565 0.663362 ]  
 [0.62448659 0.62549253 0.51641143 0.      ]  
 [0.80377561 0.1795282  0.3361258  0.77435916]]
```



```
[[ 2.2412  0.0466  0.2621  0.0086]  
 [ 0.2491 -0.3091 -0.102  -0.1901]  
 [ 0.0635 -0.3806  0.5487 -0.0029]  
 [-0.2116  0.2147 -0.2189  0.0621]]
```



# 과제

- dct mask들을 별도로 리턴

```
def dct_block(block, n=8):
    dst = np.zeros(block.shape)

    v, u = dst.shape
    y, x = np.mgrid[0:u, 0:v]

    # dct_mask를 시각화 하기 위한 리스트
    # 본 과제에서만 사용되며, 다음 주 jpeg 과제에서는 사용되지 않을 예정
    # 각 단계의 mask를 list에 추가해서 리턴
    dct_mask_list = list()

    for v_ in range(v):
        for u_ in range(u):
            mask = ???

            dct_mask_list.append(mask)

            dst[v_, u_] = ???

    return dst, dct_mask_list
```



# 과제

- dct mask들을 한번에 시각화

```
# dct_mask 전체 한번에 시각화
tmp_list = list()
for i in range(block_size):
    row = np.concatenate(dct_mask_list[i*block_size:(i+1)*block_size], axis=1)
    tmp_list.append(row)
dct_mask_show = np.concatenate(tmp_list, axis=0)

dct_mask_show = (dct_mask_show - dct_mask_show.min()) / (dct_mask_show.max() - dct_mask_show.min())
dct_mask_show = (dct_mask_show * 255).astype('uint8')

print(dct_mask_show)

cv2.imshow('entire_dct_mask', cv2.resize(dct_mask_show, dsize=(512, 512), interpolation=cv2.INTER_NEAREST))
cv2.waitKey(0)
cv2.destroyAllWindows()
```

# 과제

- **dct mask**
  - 약간의 오차(1,2정도?)가 있는 정도는 감점X
  - 참고용으로 mask의 값을 첨부

[255	255	255	255	244	173	71	0	216	28	28	216	173	0	244	71]
[255	255	255	255	244	173	71	0	216	28	28	216	173	0	244	71]
[255	255	255	255	244	173	71	0	216	28	28	216	173	0	244	71]
[255	255	255	255	244	173	71	0	216	28	28	216	173	0	244	71]
[244	244	244	244	235	169	75	9	209	35	35	209	169	9	235	75]
[173	173	173	173	169	141	103	75	158	86	86	158	141	75	169	103]
[ 71	71	71	71	75	103	141	169	86	158	158	86	103	169	75	141]
[ 0	0	0	0	9	75	169	235	35	209	209	35	75	235	9	169]
[216	216	216	216	209	158	86	35	188	56	56	188	158	35	209	86]
[ 28	28	28	28	35	86	158	209	56	188	188	56	86	209	35	158]
[ 28	28	28	28	35	86	158	209	56	188	188	56	86	209	35	158]
[216	216	216	216	209	158	86	35	188	56	56	188	158	35	209	86]
[173	173	173	173	169	141	103	75	158	86	86	158	141	75	169	103]
[ 0	0	0	0	9	75	169	235	35	209	209	35	75	235	9	169]
[244	244	244	244	235	169	75	9	209	35	35	209	169	9	235	75]
[ 71	71	71	71	75	103	141	169	86	158	158	86	103	169	75	141]]



# 과제

- **dct 결과 값 확인**

- 랜덤 시드 고정 후, 정해진 4x4 행렬 생성
- src와 dst가 다음과 같은 값이 나와야 함

```
if __name__ == '__main__':  
    block_size = 4  
  
    np.random.seed(2022)  
    src = np.random.randn(4, 4)  
    src = (src - src.min()) / (src.max() - src.min())  
    print(src)  
  
    dst, dct_mask_list = dct_block(src, n=block_size)  
    print(np.round(dst, 4))
```

```
[[0.53731403 0.47336688 0.50497433 1.        ]  
 [0.60318716 0.71475572 0.60758565 0.663362   ]  
 [0.62448659 0.62549253 0.51641143 0.        ]  
 [0.80377561 0.1795282  0.3361258  0.77435916]]
```



```
[[ 2.2412  0.0466  0.2621  0.0086]  
 [ 0.2491 -0.3091 -0.102  -0.1901]  
 [ 0.0635 -0.3806  0.5487 -0.0029]  
 [-0.2116  0.2147 -0.2189  0.0621]]
```

# 과제

- **dct.py**
  - dct\_block 함수 완성
  - 보고서에 dct\_block 함수의 결과물 포함
    - dct mask를 한번에 시각화한 이미지
    - 정해진 행렬의 dct 수행 전과 후의 값

# 과제

## • 제출 방법

- 코드 파일
  - 구현 결과가 포함된 python 파일(.py)
- 보고서
  - [IP]201900000\_홍길동\_2주차\_과제.pdf
  - 보고서 양식 사용
  - PDF 파일 형식으로 제출(pdf가 아닌 다른 양식으로 제출시 감점)
- 제출 파일
  - [IP]201900000\_홍길동\_2주차\_과제.zip
  - .py 파일과 pdf 보고서를 하나의 파일로 압축한 후, 양식에 맞는 이름으로 제출

**QnA**