# 수치해석 HW #9

## Numerical analysis

JaeHoon KANG
강 재 훈

# HW #4-2

Curve fitting using Least squares

## CONTENTS

HW #9

# 01 코드 설명

```
import cv2
import numpy as np
```

## 사용된 library

Numpy = 행렬의 transpose와 inverse를 구하기 위해 사용된 lilbrary

Opencv = img의 R,G,B를 나누고 DCT&IDCT의 결과를 출력하기 위해 사용

HW #9

# 01 코드 설명

```
def aa(u,v):
    result = 1

    if u == 0 and v == 0:
        result = 1/16

    elif (u == 0 and v > 0) :
        result = 2/16
    elif (u > 0 and v == 0) :
        result = 2/16

    else:
        result = 4/16

    return result
```

$$u,v=0,1,\ldots,N-1$$

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & for \quad u = 0 \\ \sqrt{\frac{2}{N}} & for \quad u \neq 0. \end{cases}$$

## 사용된 함수 설명

aa = a(u)a(v)를 계산해주는 함수입니다.

우리의 코드에선 N은 16이기 때문에

u, v == 0 이면 aa는 1/16

u == 0, v > 0이면 aa는 2/16

u > 0 1, v == 0이면 aa는 2/16

u, v >0 이면 aa는 4/16 입니다.

HW #9

# 01 코드 설명

```
def dct(block):
    c = np.zeros((N,N),np.float32)

    for u in range(N):
        for v in range(N):
            tmp = 0
            for x in range(N):
                for y in range(N):
                    tmp += block[x][y] * np.cos( ( np.pi*(2*x+1)*u ) / (2*N) ) * np.cos( ( np.pi*(2*y+1)*v ) / (2*N) )

            tmp = aa(u,v)*tmp
            c[u][v] = tmp

    rank = np.ravel(np.abs(c))
    rank = np.sort(rank)[::-1]
    idx = []

    for i in range(N):
        if len(idx) == 16 :
                    break
        for j in range(N):
            if len(idx) == 16 :
                    break
            for k in range(N):
                if len(idx) == 16 :
                    break
                if c[j][k] == rank[i] :
                    idx.append([j,k,rank[i]])

    c = np.zeros((N,N),np.float32)
    for x,y,v in idx :
        c[x][y] = v

    return c
```

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

x, y: pixel 좌표 in the block  u, v: 주파수 크기 in the block

## 사용된 함수 설명

DCT 함수로 원래 이미지에 16 by 16 block 만큼 DCT를 진행합니다.

총 256개의 coefficients가 나오겠지만 우리는 그중 절대값이 큰 16개의 값만 이용할 것입니다. (high frequency 성부들은 대부분 사라진다.)

따라서 구해진 coefficient matrix를 탐색하며 절대값이 큰 16곳을 제외하고선 모두 0으로 만들어줍니다.

HW #9

# 01 코드 설명

```
def idct(c):
    f = np.zeros((N,N),np.float32)
    for x in range(N):
        for y in range(N):
            tmp = 0
            for u in range(N):
                for v in range(N):
                    tmp += aa(u,v) * c[u][v] * np.cos( ( np.pi*(2*x+1)*u ) / (2*N) ) * np.cos( ( np.pi*(2*y+1)*v ) / (2*N)
            f[x][y] = tmp

    return f
```

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

## 사용된 함수 설명

IDCT 함수로 16 by 16 block만큼을 원래 이미지로 복원합니다.

DCT로 구한 coefficient를 basis function에 곱하고 linear combination을 해서 원래 이미지를 만듭니다.

IDCT의 overflow는 main함수에서 처리하였습니다.

HW #9

# 01 코드 설명

```
file = "test1.jpg"
img = cv2.imread(file,cv2.IMREAD_COLOR)

h, w = np.array(img.shape[:2])
(b, g, r) = cv2.split(img)

for colors in range(3):
    color = np.zeros((h,w),np.float32)
    if colors == 0 :
        color[:h,:w] = r
    elif colors == 1 :
        color[:h,:w] = g
    else :
        color[:h,:w] = b

    print(color)
    print(color.shape)

    for row in range(h//N):
        for col in range(w//N):
            currentblock = dct(color[row*N:(row+1)*N, col*N:(col+1)*N])
            currentblock = idct(currentblock)

            for i in range(N):
                for j in range(N):
                    tmp = currentblock[i][j]
                    if tmp < 0 :
                        tmp = 0
                    if tmp > 255 :
                        tmp = 255

                    if colors == 0 :
                        r[row*N+i][col*N+j] = tmp
                    elif colors == 1 :
                        g[row*N+i][col*N+j] = tmp
                    else :
                        b[row*N+i][col*N+j] = tmp

    print("color : ",colors, " DCT & IDCT is finish")

result = cv2.merge([b,g,r])
cv2.imshow("result", result)
cv2.imwrite("result1.jpg",result)
cv2.waitKey(0)
```

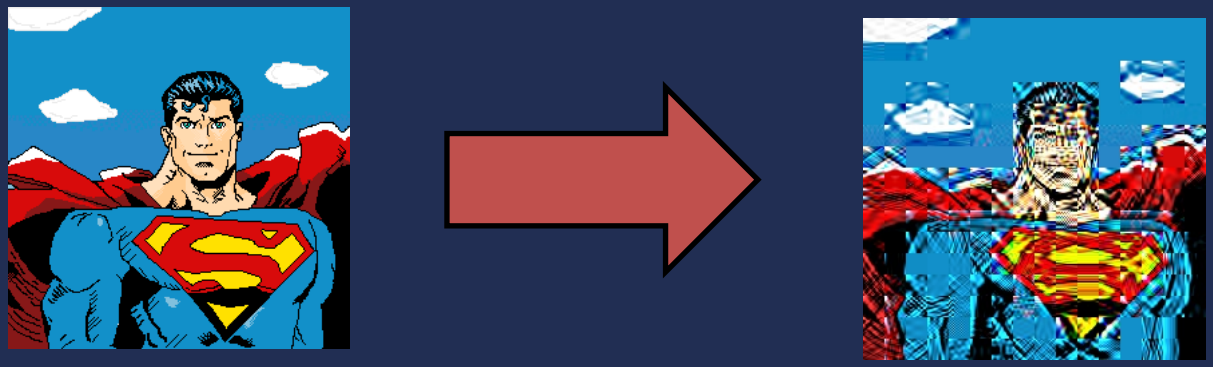
## main 함수 설명

사용할 이미지의 이름을 file 변수에 저장 후 Img를 R,G,B로 각각 나눈 후 16 by 16 block으로 나눠서 DCT, IDCT를 진행하며 이미지를 압축하고 복원합니다.

만약 idct를 하고 난 후의 pixel 값이 0보다 작다면 0으로 만들어주고 255보다 크다면 255로 만들어줍니다.

HW #9

# 02 실행결과



Case1에선 256 by 256 이미지가 사용되었습니다.

16 by 16 block을 이용했더니

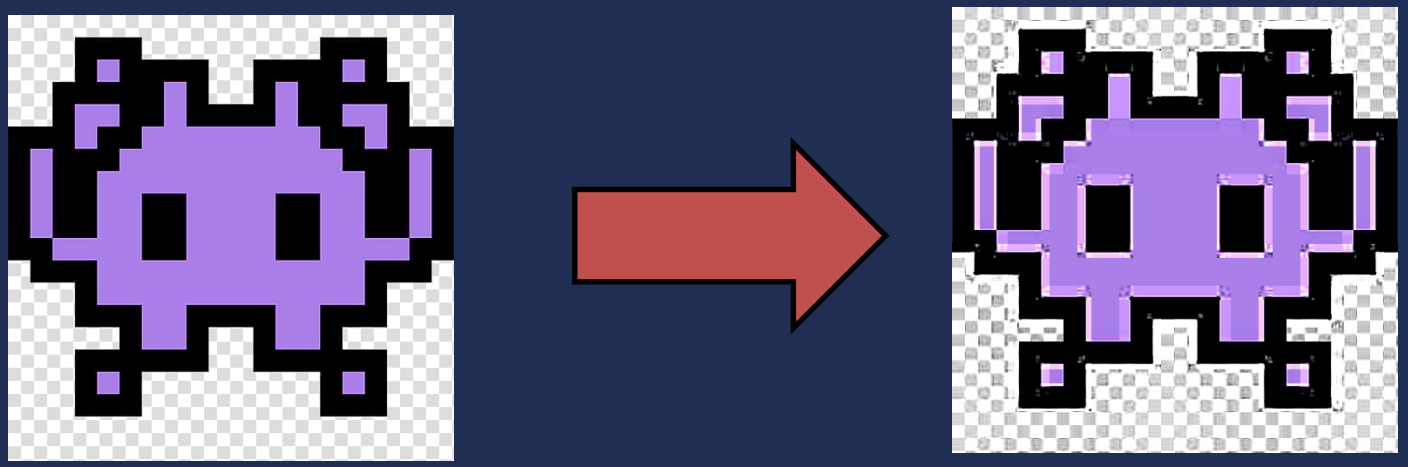
그 크기만큼 미세한 칸이 나눠지는걸 볼 수 있었습니다.

또한 0보다 작은 값은 0으로, 255보다 큰 값은 255로 설정하였더니

영상의 부드러움이 조금 깨지는 것을 확인 할 수 있었습니다.

HW #9

# 02 실행결과



Case2에선 512 by 512 이미지가 사용되었습니다.

16 by 16 block을 이용했더니

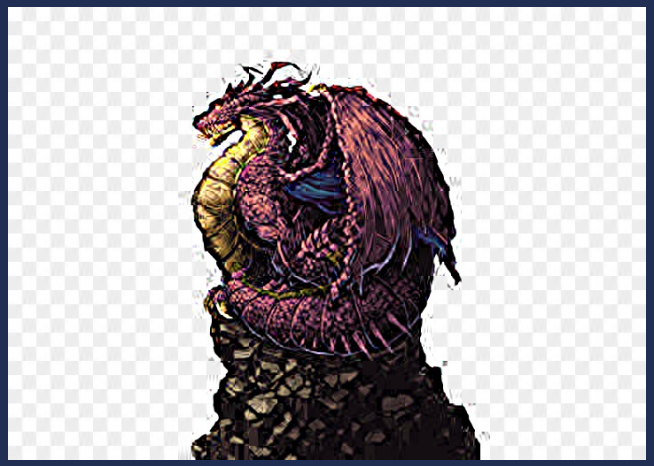
그 크기만큼 미세한 칸이 나눠지는걸 볼 수 있었습니다.

또한 0보다 작은 값은 0으로, 255보다 큰 값은 255로 설정하였더니

영상의 부드러움이 조금 깨지는 것을 확인 할 수 있었습니다.

HW #9

# 02 실행결과



Case3에선 720 by 512 이미지가 사용되었습니다.

16 by 16 block을 이용했더니

그 크기만큼 미세한 칸이 나눠지는걸 볼 수 있었습니다.

또한 0보다 작은 값은 0으로, 255보다 큰 값은 255로 설정하였더니

영상의 부드러움이 조금 깨지는 것을 확인 할 수 있었습니다.

# 감사합니다

## THANK YOU

JaeHoon KANG
강 재 훈