수치해석
HW9

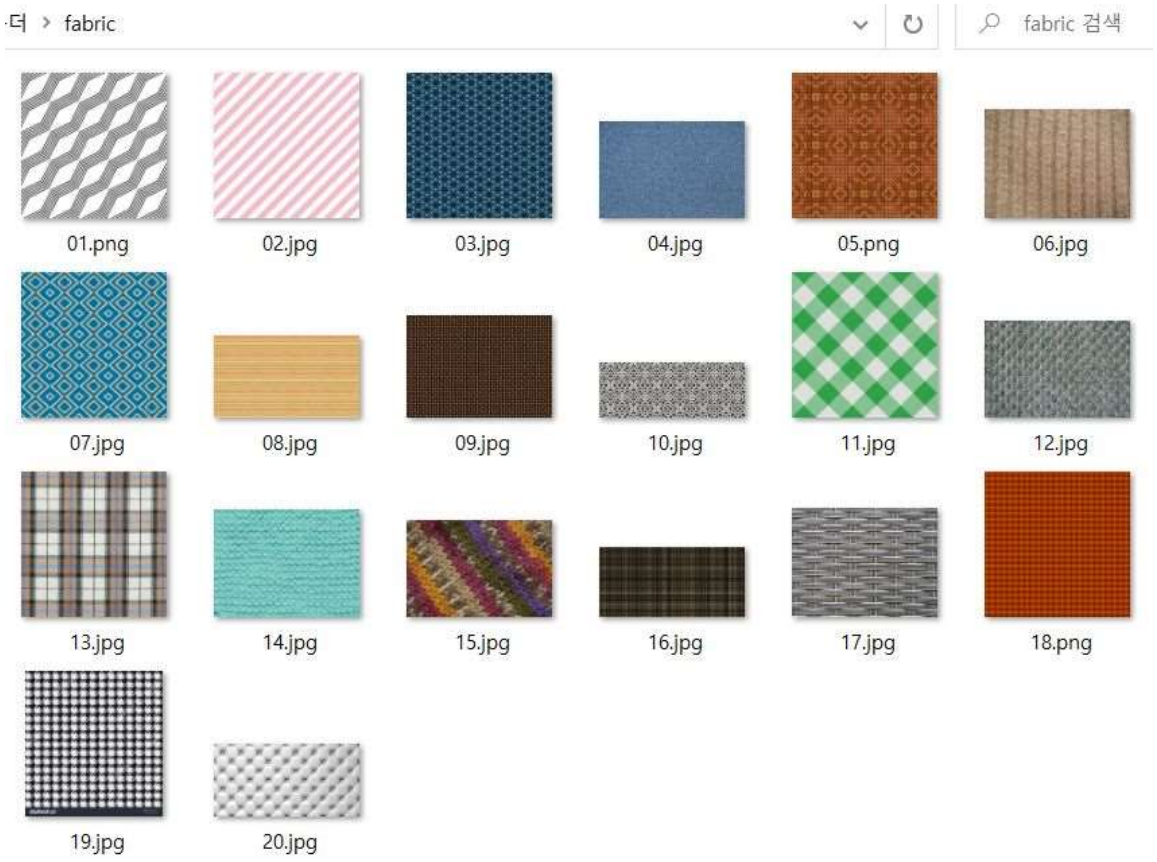
2016025478 컴퓨터전공 서경욱

개발 환경

- Python 3.8.0

- 사용한 Library
- Numpy (Fast Fourier Transform)
- Opencv (이미지 파일 처리, image resize)
- matplotlib (이미지 출력, magnitude 출력)

DataSet



- 반복적인 패턴을 가지는 20장의 사진을 준비하였습니다.
- 사진들은 color image이지만 opencv의 imread 함수에 0 flag를 주어서 gray image로 다루었습니다.

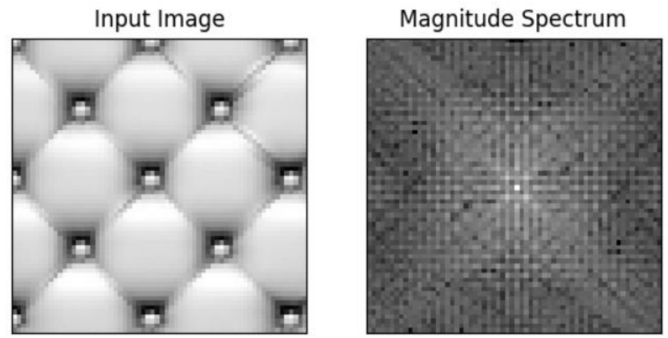
Image Rescaling

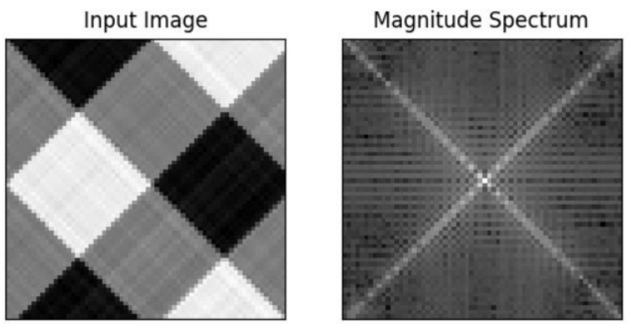
```
if i==0 or i==1:
    img = cv2.resize(img,dsize=(160,160),interpolation=cv2.INTER_AREA)
elif i==2:
    img = cv2.resize(img,dsize=(320,320),interpolation=cv2.INTER_AREA)
elif i==4 or i==12:
    img = cv2.resize(img,dsize=(130,130),interpolation=cv2.INTER_AREA)
elif i==6 or i==10:
    img = cv2.resize(img,dsize=(200,200),interpolation=cv2.INTER_AREA)
elif i==5 or i==9 or i==14 or i==16 or i==19:
    img = cv2.resize(img,dsize=(0,0),fx=0.4,fy=0.4,interpolation=cv2.INTER_AREA)
elif i==7:
    img = cv2.resize(img,dsize=(0,0),fx=0.7,fy=0.7,interpolation=cv2.INTER_AREA)
elif i==18:
    img = cv2.resize(img,dsize=(360,360),interpolation=cv2.INTER_AREA)
```

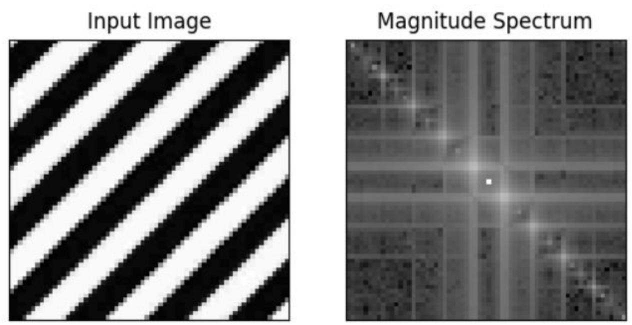
- Image scaling은 크기가 서로 다른 Image들을 64*64 DFT를 진행함에 있어서 패턴의 반복이 block에 들어갈 수 있도록 하는 과정입니다.

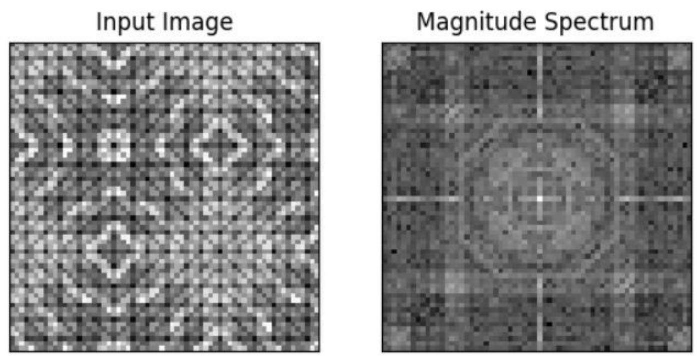
- Scaling의 구현은 64*64 dft한 magnitude와 원본을 출력해보면서 패턴이 block안에 잘 들어갈 때까지 **하나 하나 임의로 기준을** 정하게 되었습니다.

Magnitude 확인









- Rescaling한 Image를 **원본과 magnitude**를 함께 출력해 본 결과입니다.

- 여기서 가로 중간 세로 중간을 선으로 긋는다고 생각하면 **4분면끼리의 대칭성을** 볼 수 있었고 이를 이용하여 64*64 magnitude 전체를 이용하는 것이 아닌 1사분면의 32*32 magnitude를 이용할 것입니다.

Criterion for decision

```
rows,cols = img.shape
r = rows // 64
c = cols // 64

coefficients = []

for j in range(r):
    for k in range(c):
        _img = img[j*64:(j*64)+64,k*64:(k*64)+64]
        f = np.fft.fft2(_img)
        fshift = np.fft.fftshift(f)
        m_spectrum = 20*np.log(np.abs(fshift))

        quadrant = m_spectrum[:32,32:64]

        tmp = np.ravel(quadrant)
        tmp = np.sort(tmp)[::-1]
        cut = tmp[9]

        I = []
        x = 0
        for p in range(32):
            for q in range(32):
                if quadrant[p][q] >= cut:
                    if x == 10: break
                    I.append([p,q])
                    x += 1
            if x==10: break

        coefficients.append(I)
dominant.append(coefficients)
```

- 이번 HW의 목적은 image recognition을 하기 위함이고 이를 위한 기준이 필요합니다.
- 저는 우선 image마다 64*64 block들을 dft처리한 후 처리 값의 4분의 1만 잘라서(32*32) 해당 **magnitude의 가장 큰 10개 값의 index**를 구해서 저장했습니다.
- 그 후 Image의 모든 64*64 block들의 1사분면에서 가장 큰 10개의 Index들을 저장하고 **분석해보았습니다**.

Criterion for decision

1

[[0, 16], [2, 16], [2, 18], [14, 30], [16, 30], [28, 0], [28, 2], [30, 0], [30, 2], [30, 4]]

[[0, 16], [2, 16], [2, 18], [14, 30], [16, 30], [28, 0], [28, 2], [30, 0], [30, 2], [30, 4]]

[[0, 16], [2, 16], [2, 18], [14, 30], [16, 30], [28, 0], [28, 2], [30, 0], [30, 2], [30, 4]]

c:₩Users₩user₩Desktop₩새 폴더₩HW10.py:49: RuntimeWarning: divide by zero encountered in log

m_spectrum = 20*np.log(np.abs(fshift))

[[0, 16], [2, 16], [2, 18], [14, 30], [16, 30], [28, 0], [28, 2], [30, 0], [30, 2], [30, 4]]

2

[[28, 0], [28, 1], [28, 4], [29, 0], [29, 1], [29, 2], [29, 3], [30, 3], [31, 3], [31, 4]]

[[28, 0], [28, 1], [28, 4], [29, 0], [29, 1], [29, 2], [29, 3], [30, 3], [31, 3], [31, 4]]

[[28, 0], [28, 1], [28, 4], [29, 0], [29, 1], [29, 2], [29, 3], [30, 3], [31, 3], [31, 4]]

[[28, 0], [28, 1], [28, 4], [29, 0], [29, 1], [29, 2], [29, 3], [30, 3], [31, 3], [31, 4]]

- 몇 안되는 **모든 block들의 <u>가장 큰값 10개의 Index가 다 같은 경우</u>**입니다. 이 경우 recognition을 하려는 block의 1사분면의 10개의 가장 큰 magnitude를 뽑고 이 Index가 모두 들어있는지 확인하여 구분하였습니다.

#13

[[16, 0], [21, 0], [24, 0], [25, 0], **[26, 0], [27, 0], [28, 0], [29, 0], [30, 0], [31, 0]]**

[[8, 0], [16, 0], [21, 0], [24, 0], [26, 0], [27, 0], [28, 0], [29, 0], [30, 0], [31, 0]]

[[8, 0], [16, 0], [21, 0], [24, 0], [26, 0], [27, 0], [28, 0], [29, 0], [30, 0], [31, 0]]

[[8, 0], [16, 0], [21, 0], [24, 0], [26, 0], [27, 0], [28, 0], [29, 0], [30, 0], [31, 0]]

- 하나의 Index만 다르고 나머지는 다 공통적인 같은 Index를 가지는 case입니다.

Criterion for decision

#8

[[7, 0], [9, 0], [12, 0], [21, 0], **[22, 0], [24, 0], [25, 0], [27, 0],** [30, 0], [31, 0]]

[[7, 0], [9, 0], [12, 0], [21, 0], [22, 0], [24, 0], [25, 0], [27, 0], [30, 0], [31, 0]]

[[7, 0], [9, 0], [12, 0], [21, 0], [22, 0], [24, 0], [25, 0], [27, 0], [30, 0], [31, 0]]

[[7, 0], [9, 0], [12, 0], [21, 0], [22, 0], [24, 0], [25, 0], [27, 0], [30, 0], [31, 0]]

[[3, 0], [7, 0], [9, 0], [21, 0], [22, 0], [24, 0], [25, 0], [27, 0], [30, 0], [31, 0]]

[[7, 0], [9, 0], [12, 0], [21, 0], [22, 0], [24, 0], [25, 0], [27, 0], [30, 0], [31, 0]]

[[7, 0], [9, 0], [21, 0], [22, 0], [24, 0], [25, 0], [27, 0], [28, 0], [30, 0], [31, 0]]

[[7, 0], [9, 0], [19, 0], [21, 0], [22, 0], [24, 0], [25, 0], [27, 0], [29, 0], [31, 0]]

#9

[[22, 0], [25, 2], [25, 3], **[27, 0],** [28, 0], [29, 2], [29, 7], **[30, 2]**, [30, 3], [30, 7]]

[[22, 0], [24, 2], [25, 2], [27, 0], [28, 0], [29, 2], [29, 7], [30, 2], [30, 3], [30, 7]]

[[22, 0], [25, 2], [27, 0], [27, 5], [29, 2], [29, 3], [29, 7], [30, 2], [30, 3], [30, 7]]

[[22, 0], [25, 2], [27, 0], [27, 5], [29, 2], [29, 7], [30, 2], [30, 3], [30, 7], [31, 5]]

[[22, 0], [25, 2], [25, 3], [27, 0], [27, 5], [29, 2], [29, 7], [30, 2], [30, 3], [30, 7]]

[[22, 0], [25, 2], [25, 3], [27, 0], [28, 0], [29, 0], [29, 7], [30, 2], [30, 3], [30, 7]]

[[22, 0], [25, 2], [25, 3], [27, 0], [27, 5], [28, 0], [29, 3], [30, 2], [30, 3], [30, 7]]

[[22, 0], [25, 2], [27, 0], [27, 5], [28, 0], [29, 2], [29, 7], [30, 2], [30, 3], [30, 7]]

- 10개의 큰 Index가 **공통되는 것이 적은 경우**입니다.
- 이 경우에도 모든 case에 있는 Index만 추려도 **Image마다 서로 다른 set의 Index**를 가짐을 파악할 수 있고 이를 이용해 recognition하였습니다.

Criterion for decision

case 1(다 일치) – 1,2

case 2(많이 일치) – 3,5,8,10, 11,13 ,18, 19 ,20

case 3(조금 일치) – 4, 9, 14, 15

case 4(일치 x) – 6, 12

-dataset을 분석한 결과 20장의 Image의 특성은 다음과 같았습니다. image recognition을 할 시에 단순히 공통된 Index의 존재여부뿐만 아니라 **공통된 순서(몇 번째 큰지)가 존재한다면 추가 조건으로 이용해 분류가 더 잘되도록 했습니다**. 또한 특성이 강한 case1부터 2,3,4 순으로 recognition을 진행했습니다.

- 다만 여기서 **case 4**와 같이 10개의 큰 Index가 모든 block에 대해 공통된 경우가 없는 경우에 대한 조건이 추가로 필요했습니다.

Criterion for decision

#6↵

[[20, 4], [20, 6], [20, 7], [20, 8], [20, 11], [26, 3], [28, 0], [29, 1], [31, 0], [31, 1]]↵

[[20, 1], [20, 6], [20, 9], [20, 11], [21, 11], [24, 1], [27, 1], [30, 2], [30, 5], [31, 2]]↵

[[20, 8], [20, 9], [20, 11], [20, 14], [26, 0], [26, 3], [27, 0], [27, 7], [29, 0], [30, 2]]↵

[[20, 1], [20, 8], [20, 9], [20, 11], [20, 13], [20, 14], [20, 16], [26, 2], [29, 0], [31, 0]]↵

[[20, 4], [20, 6], [20, 8], [20, 11], [21, 6], [21, 7], [21, 11], [27, 0], [29, 1], [30, 2]]↵

[[20, 4], [20, 6], [20, 7], [20, 9], [20, 11], [20, 12], [21, 7], [26, 1], [26, 4], [31, 0]]↵

[[20, 9], [21, 6], [21, 8], [21, 11], [26, 1], [26, 3], [27, 5], [29, 0], [30, 0], [31, 0]]↵

[[20, 6], [20, 8], [20, 10], [20, 15], [21, 1], [21, 3], [21, 8], [28, 0], [31, 0], [31, 17]]↵

[[20, 6], [20, 8], [20, 10], [21, 6], [21, 8], [21, 10], [21, 11], [21, 13], [28, 1], [31, 0]]↵

[[20, 4], [20, 6], [20, 7], [20, 9], [20, 10], [20, 11], [20, 12], [20, 14], [30, 2], [31, 0]]↵

[[20, 0], [20, 11], [20, 14], [21, 6], [26, 1], [26, 6], [27, 0], [28, 2], [30, 0], [31, 0]]↵

[[20, 10], [21, 3], [21, 5], [21, 10], [25, 0], [29, 0], [30, 0], [30, 2], [31, 0], [31, 14]]↵

- 6번 Image의 10개의 큰 magnitud의 Index를 조사한 결과 **1,2,3번째 값들의 y좌표가 20 or 21**이라는 특징을 발견하여 기준으로 사용하였습니다.

#12↵

[[16, 11], [19, 11], [26, 0], [26, 3], [26, 8], [29, 0], [30, 1], [30, 2], [31, 0], [31, 3]]↵

[[23, 9], [24, 7], [29, 0], [30, 2], [30, 3], [31, 0], [31, 1], [31, 3], [31, 4], [31, 6]]↵

[[18, 10], [21, 6], [23, 3], [26, 1], [28, 1], [29, 0], [29, 4], [30, 3], [30, 7], [31, 0]]↵

[[24, 2], [26, 3], [27, 0], [28, 1], [29, 0], [29, 2], [30, 2], [30, 5], [31, 1], [31, 4]]↵

[[26, 3], [26, 4], [27, 2], [30, 1], [30, 2], [30, 3], [30, 4], [31, 0], [31, 1], [31, 2]]↵

[[24, 5], [28, 0], [28, 3], [28, 6], [28, 9], [29, 5], [31, 0], [31, 1], [31, 2], [31, 4]]↵

[[23, 8], [25, 6], [27, 5], [27, 12], [28, 4], [29, 0], [30, 0], [30, 2], [31, 4], [31, 7]]↵

[[24, 2], [26, 7], [27, 0], [29, 7], [30, 1], [30, 3], [31, 0], [31, 1], [31, 4], [31, 6]]↵

[[23, 13], [27, 0], [28, 3], [30, 0], [30, 3], [31, 1], [31, 2], [31, 4], [31, 5], [31, 7]]↵

[[28, 3], [29, 0], [29, 4], [30, 0], [30, 2], [31, 0], [31, 1], [31, 2], [31, 4], [31, 5]]↵

[[26, 4], [27, 7], [27, 10], [28, 10], [29, 0], [30, 0], [30, 2], [30, 4], [31, 0], [31, 3]]↵

[[16, 9], [28, 2], [28, 5], [29, 4], [30, 3], [30, 5], [31, 0], [31, 1], [31, 2], [31, 4]]↵

[[23, 10], [24, 2], [24, 3], [26, 2], [27, 0], [27, 1], [27, 4], [29, 1], [30, 1], [31, 1]]↵

[[23, 5], [24, 10], [25, 0], [28, 0], [29, 2], [30, 2], [30, 3], [31, 1], [31, 2], [31, 4]]↵

[[27, 0], [27, 1], [29, 0], [29, 5], [30, 3], [30, 5], [31, 2], [31, 3], [31, 4], [31, 6]]↵

(block 개수가 너무 많아서 짤림)

- 12번 Image의 10개의 큰 magnitud의 Index를 조사한 결과 10번째 값들의 y좌표가 31이고 9번째 값들의 y좌표가 31 or 30 or 29임을 파악할 수 있었습니다.

- 다른 case에 비해 기준이 크게 seperable 하지 않으므로 12에 대한 검사는 제일 마지막에 진행하도록 했습니다.

실행 결과

```
this picture is 1!
this picture is 2!
this picture is 3!
this picture is 4!
this picture is 5!
this picture is 6!
this picture is 7!
this picture is 8!
this picture is 9!
this picture is 10!
this picture is 11!
this picture is 12!
this picture is 13!
this picture is 14!
this picture is 14!
this picture is 16!
this picture is 17!
this picture is 18!
this picture is 19!
this picture is 20!
```

- 반복문을 돌며 1~20까지의 Image를 recognition 한 결과입니다.
- 성공률은 15번째 Image를 14번째 Image로 인식하여 20개 중에 19개를 성공하였고 **95%의 성공률**을 보였습니다.

```
while True:
    if is_1(I) or is_2(I): break
    elif is_3(I) or is_5(I) or is_7(I) or is_8(I) or is_10(I) or is_11(I) or is_13(I) or \
        is_18(I) or is_19(I) or is_20(I): break
    elif is_4(I) or is_9(I) or is_14(I) or is_15(I) or is_17(I): break
    elif is_16(I): break
    elif is_6(I): break
    elif is_12(I): break
```

- Test 실행 우선순위