컴퓨터그래픽스 4주차 과제 보고서
201300995 이상건 물리학과

구현 내용
이미지를 업 케일링 할 때 중간 좌표의 값을 구해서 업스케일링 하는거랑, 이미지 피라미드에서 가우시안 피라미드와 라플라시안 피라미드를 이용해 업스케일링 하는거다.

이유(구현 방법)
이번 과제는 코드를 어떻게 이해하였는지 위주로 작성하겠다.

일단 Interpolation을 보자.

edgeDetector.py | my_filtering.py | my_bilinear.py | my_pyramids.py

```
def my_bilinear(img, scale):
    h, w, c = img.shape
    print(img.shape)
    print(h,w,c)

    reslmg = np.zeros((int(h*scale), int(w*scale), c), dtype = np.uint8)# scale 만큼 크기를 키운 배열

    rh, rw, rc = reslmg.shape

    for i in range(0, rh):
        for j in range(0, rw):
            px = int(j // scale)# 원래 이미지의 x 좌표값
            py = int(i // scale)# 원래 이미지의 y 좌표값

            s = (i/scale) - py# u가 s, 중간값 - 원래 가로 축 좌표값 = s가 나옴.
            t = (j/scale) - px# 람다가 t

            zz = (1-t) * (1-s)# zz*f(m,n)
            zo = (1-s) * t# zo*f(m+1,n)
            oz = (1-t) * s# oz*f(m,n+1)
            oo = t * s# f(m+1,n+1)

            if px == h-1 and px == w-1:
                reslmg[i,j,:] = zz*img[py,px,:] + zo*img[py,px-1,:] + oz*img[py-1, px,:] + oo * img[py-1,px-1,:]
            elif py == h-1:
                reslmg[i,j,:] = zz*img[py,px,:] + zo * img[py, px+1,:] + oz * img[py-1, px, :] + oo * img[py-1, px+1, :]
            elif px == w-1:
                reslmg[i,j,:] = zz*img[py,px,:] + zo * img[py, px-1,:] + oz * img[py+1, px, :] + oo * img[py+1, px-1, :]
            else:
                reslmg[i,j,:] = zz*img[py,px,:] + zo*img[py, px+1,:] + oz*img[py+1, px, :] + oo*img[py+1, px+1, :]


    return reslmg
```

ppt에 나와있는 그대로의 수식을 이용한 모습이다. μ가 s이고 λ가 t라는 것을 이해하는데 조금 시간이 걸렸다. (뒤에 if문에서 px == h-1과 px == w-1 같이 조건을 거는 부분이 이해가 안갑니다.. 그리고 그 조건이 나올 때 왜 저렇게 하는지도..)

다음 이미지 피라미드를 보자.

```python
def my_pyramids(src):
    '''
    :param src: image pyramids를 생성할 이미지 원본.
    :return: Gaussian pyramids 이미지 list, Laplacian pyramids 이미지 list
    '''
    gList = []
    lList = []
    gList.append(src)

    for i in range(3):
        gaus = cv2.GaussianBlur(src,(5,5),1)
        lList.append(cv2.subtract(src,gaus))
        half = gaus[1::2, 1::2, :]
        gList.append(half)
        src = half # 계속 반으로 줄임

    end = cv2.GaussianBlur(src, (5,5), 1)
    lList.append(cv2.subtract(src,end))

    return gList, lList
```

일단 사전 단계인데 리스트에 이미지를 반으로 재귀로 계속 줄여서 저장한 리스트를 반환한다. 줄이기 전 라플라시안을 저장하고 줄인다.

```python
def my_recon(gList, lList):
    '''
    :param gList: gaussian pyramids 이미지 List
    :param lList: Laplacian pyramids 이미지 List
    :return: Laplacian pyramids로 복원된 이미지 List.
    '''
    # 복원된 이미지 List는 3개입니다.
    recon = []

    for i in range(2,-1, -1):
        #cv2.resize 함수 사용 금지.
        #gList의 이미지를 확대해, lList의 이미지를 더하세요.
        upImg = np.zeros(gList[i].shape, dtype = np.uint8) # 업스케일링 할 이미지 만큼의 크기

        # 아까 가우시안으로 계속 줄여서 저장했던 것들을 일단 upImg에 넣어놓고 append로 라플라시안과 더함.
        upImg[0::2, 0::2, :] = gList[i+1] # gList는 가우시안 리스트, 4차원 배열인데 0번째 인덱스는 (512,512,3)짜리, 1번째 인덱스는 (256,256,3) 짜리 ...
        upImg[0::2, 1::2, :] = gList[i+1]
        upImg[1::2, 0::2, :] = gList[i+1]
        upImg[1::2, 1::2, :] = gList[i+1]
        recon.append(cv2.add(upImg, lList[i])) # 여기서 가우스와 라플라시안을 더해준다.

    return recon # recon은 4차원 배열, 0번째 인덱스는 3차원 배열 크기 (128,128,3) 짜리, 1번째 인덱스는 (256,256,3), 2번째 인덱스는 (512,512,3).
```

위에서 계속 줄여서 저장한 리스트를 가져온다. 원래 업스케일링으로 키운 뒤 라플라시안과 더한다. ([0::2, 1::2, :] 부분을 잘 모르겠음..)

느낀 점

파이썬 문법 자체에 고전 할 줄은 몰랐음. [0::1]이 뭐지..

과제 난이도
어렵다.