HW 02 – REPORT

소속 : 정보컴퓨터공학부

학번 : 201824523

이름 : 안혜준

1. 서론

이번 실습의 목표는 aligned images를 각 low frequency와 high frequency로 filtering하여 두 이미지를 hybrid시키는 것이다. 진행 순서는 다음과 같다.

1. Gaussian Filtering 구현
    1.1. Box filter 구현
    1.2. 1D – Gaussian filter 구현
    1.3. 2D – Gaussian filter 구현
    1.4. Gaussian filter를 사용하여 이미지에 convolution 적용
2. Hybrid Image 생성
    2.1. low frequency image 생성
    2.2. high frequency image 생성
    2.3 두 이미지 합치기

Hybrid Image는 합치기에 충분하도록 정렬된 두 이미지가 필요하다(이는 사전 제공되었다). 정렬된 두 이미지 중 하나는 Blurring하여 low frequency로 만들고, 하나는 sharpening하여 high frequency로 만든다. 두 이미지를 합치면 크게 보았을 때 high frequency로 만든 이미지가 강조되어 보이고 작게 보았을 때 low frequency로 만든 이미지가 강조되어 보이는 hybrid 이미지가 만들어진다.

Blurring과 sharpening은 box filter를 적용해도 되지만 여기서는 gaussian filter를 사용하여 수행된다. Gaussian filter를 적용함으로써 blurring이 수행되고, 원본 이미지에서 Gaussian filter를 적용한 이미지를 뺌으로써 sharpening이 수행된다. 또한, cross-correlation이 아닌 convolution을 적용할 것이기에 filter를 뒤집는 과정이 필요할 것이다.

실습에서 사용될 이미지는 다음과 같다. 먼저 dog에 gaussian filter를 적용해 볼 것이고, tower와 eiffel을 각각 low, high frequency로 만들어 hybrid 이미지로 만들 것이다.



실습에 사용될 이미지들

2. 본론

Part 1: Gaussian Filtering

1. boxfilter(n)

n이 홀수가 아니면 assert error를 발생하는 (n x n) box filter를 생성해주는 함수이다.

```
boxfilter(3)
```
[241] ✓ 0.0s

```
array([[0.11111111, 0.11111111, 0.11111111],
       [0.11111111, 0.11111111, 0.11111111],
       [0.11111111, 0.11111111, 0.11111111]])
```

```
boxfilter(4)
```
[52] ⊗ 0.0s

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
Cell In[52], line 1
----> 1 boxfilter(4)

Cell In[48], line 3, in boxfilter(n)
      1 def boxfilter(n):
      2     # n이 odd가 아니면 assert error
----> 3     assert n%2==1, "Dimension must be odd"
      4     return np.ones((n, n)) / (n * n)

AssertionError: Dimension must be odd
```

```
boxfilter(5)
```
[242] ✓ 0.0s

```
array([[0.04, 0.04, 0.04, 0.04, 0.04],
       [0.04, 0.04, 0.04, 0.04, 0.04],
       [0.04, 0.04, 0.04, 0.04, 0.04],
       [0.04, 0.04, 0.04, 0.04, 0.04],
       [0.04, 0.04, 0.04, 0.04, 0.04]])
```

실행 결과는 위와 같다.

2. gauss1d(sigma)

sigma에 맞는 1차원 gaussian filter를 생성해주는 함수이다.

```
[245]  gauss1d(0.3)
       ✓ 0.0s
...    array([0.00383626, 0.99232748, 0.00383626])

[246]  gauss1d(0.5)
       ✓ 0.0s
...    array([0.10650698, 0.78698604, 0.10650698])

[247]  gauss1d(1)
       ✓ 0.0s
...    array([0.00443305, 0.05400558, 0.24203623, 0.39905028, 0.24203623,
              0.05400558, 0.00443305])

[248]  gauss1d(2)
       ✓ 0.0s
...    array([0.0022182 , 0.00877313, 0.02702316, 0.06482519, 0.12110939,
              0.17621312, 0.19967563, 0.17621312, 0.12110939, 0.06482519,
              0.02702316, 0.00877313, 0.0022182 ])
```

실행 결과는 위와 같다.

3. gauss2d(sigma)

gauss1d를 외적하여 만든 2차원 gaussian filter를 생성해주는 함수이다.

```
[249]  gauss2d(0.5)
       ✓ 0.0s
...    array([[0.01134374, 0.08381951, 0.01134374],
              [0.08381951, 0.61934703, 0.08381951],
              [0.01134374, 0.08381951, 0.01134374]])

[76]   gauss2d(1)
       ✓ 0.0s
...    array([[1.96519161e-05, 2.39409349e-04, 1.07295826e-03, 1.76900911e-03,
              1.07295826e-03, 2.39409349e-04, 1.96519161e-05],
             [2.39409349e-04, 2.91660295e-03, 1.30713076e-02, 2.15509428e-02,
              1.30713076e-02, 2.91660295e-03, 2.39409349e-04],
             [1.07295826e-03, 1.30713076e-02, 5.85815363e-02, 9.65846250e-02,
              5.85815363e-02, 1.30713076e-02, 1.07295826e-03],
             [1.76900911e-03, 2.15509428e-02, 9.65846250e-02, 1.59241126e-01,
              9.65846250e-02, 2.15509428e-02, 1.76900911e-03],
             [1.07295826e-03, 1.30713076e-02, 5.85815363e-02, 9.65846250e-02,
              5.85815363e-02, 1.30713076e-02, 1.07295826e-03],
             [2.39409349e-04, 2.91660295e-03, 1.30713076e-02, 2.15509428e-02,
              1.30713076e-02, 2.91660295e-03, 2.39409349e-04],
             [1.96519161e-05, 2.39409349e-04, 1.07295826e-03, 1.76900911e-03,
              1.07295826e-03, 2.39409349e-04, 1.96519161e-05]])
```

4. Gaussian Filtering

A. convolve2d(array, filter)

filter로 zero-padding의 크기를 계산 후 padding을 추가한 이미지를 생성하였다. Convolution을 위해 각 차원에 대해 뒤집고, 이미지의 모든 pixel에 대해 filter를 적용하였다.

B. gaussconvolve2d(array, sigma)

gauss2d(sigma)로 gaussian filter를 생성하여 convolve2d(array, filter)를 실행하여 리턴한다.

C. Apply to dog image.

PIL을 이용하여 개 이미지를 열고 gray scale을 적용한다. Filter 적용을 위해 numpy array로 변경해주고 Gaussian convolution을 수행한다. 이때 sigma는 3으로 설정하였다. Numpy.clip과 astype을 통해 이미지형식에 맞춰주고 저장한다.

D. Show original and filtered images.

PIL을 통해 생성한 이미지를 출력한다.

Gaussian Filtering이 적용된 개 이미지

Part 2: Hybrid Images

Hybrid 이미지를 위해 Tower 이미지와 Eiffel 이미지 쌍을 선택하였다.

1. Make low frequency image.

개 이미지와 다르게 gray scale이 아닌 각 rgb 채널에 gaussian convolution을 수행하여 low frequency image를 만든다. Gaussian의 sigma는 3으로 설정하였다.

```
r = gaussconvolve2d(tower[:,:,0], sigma)
g = gaussconvolve2d(tower[:,:,1], sigma)
b = gaussconvolve2d(tower[:,:,2], sigma)
tower = np.dstack([r, g, b])
```

위 코드를 통하여 각 rgb 채널에 gaussian convolution을 적용하였다. 그 후 numpy.dstack을 사용하여 생성한 rgb 채널을 하나로 합쳤다.

low frequency image

2. Make high frequency image.

Tower와 같은 방식으로 eiffel에 gaussian convolution을 수행한다. Eiffel의 high frequency를 얻어야 하므로 원본에서 low frequency 이미지를 제거한다. 이때 high frequency의 값 범위는 0 ~ 255(uint8)이 아닌 -128 ~ 127이 되야 함을 유의해야 한다. 따라서 numpy.clip(highFreqEiffel, -128, 127)을 수행하고 int형식으로 저장해준다.

현재 high frequency는 음수 값을 가지고 있으므로 시각화를 위해 128을 더하여 이미지를 확인한다.

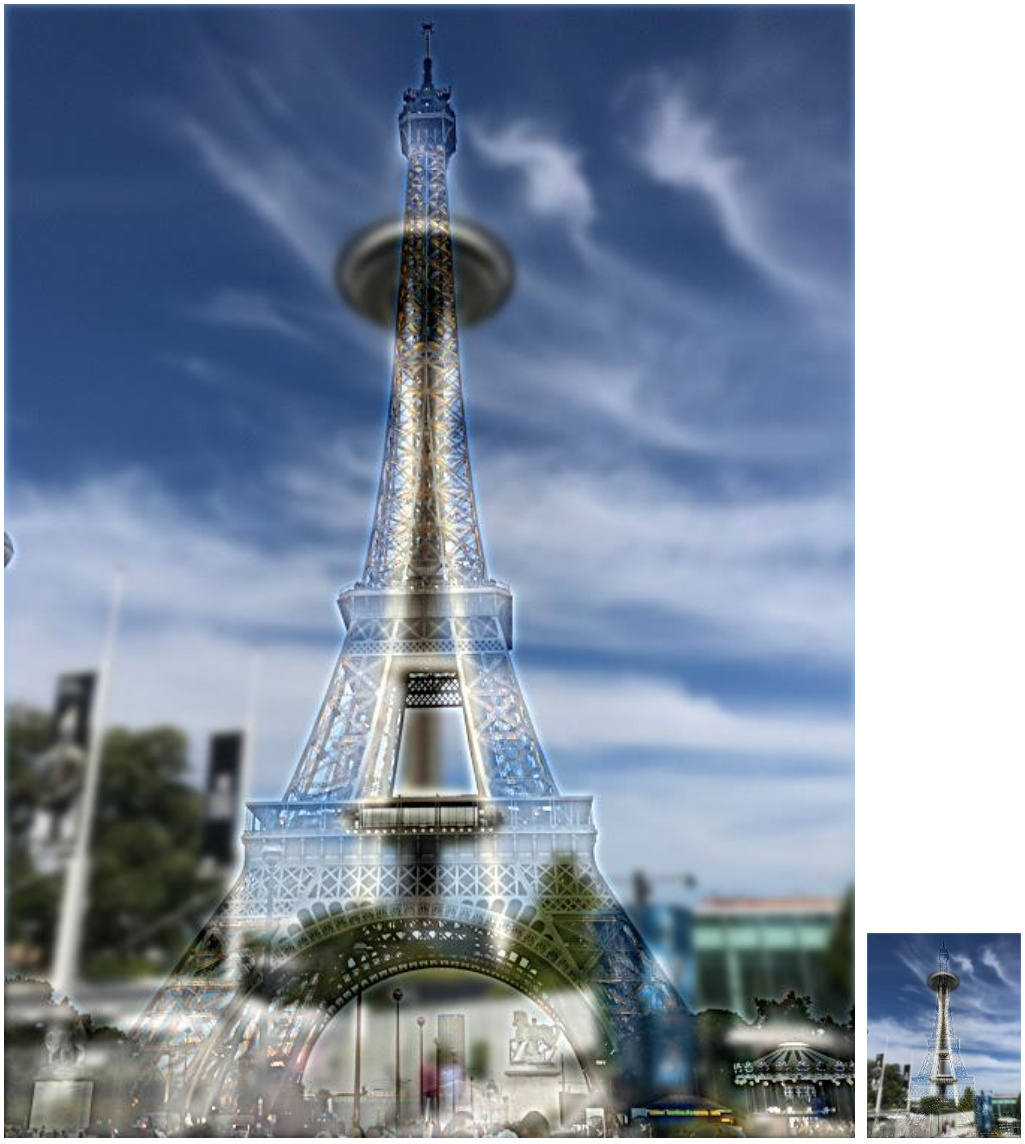
High frequency image

3. Make hybrid image.

이제 이미지를 합칠 차례이다. tower에 highFreqEiffel을 더하고 0 ~ 255를 벗어난 게 있을 수 있으니 값을 맞춰주며 타입도 uint8로 바꾼다.

```
hybridImage = tower + highFreqEiffel
hybridImage = np.clip(hybridImage, 0, 255).astype(np.uint8)
```

이로써 hybrid 이미지가 만들어졌다.



최종 hybrid 이미지(좌우 같은 이미지)

큰 이미지에서는 에펠탑이 강조되어 보이고, 작은 이미지에서는 타워가 보이는 것을 확인할 수 있다. 작은 이미지에서 high frequency 이미지가 보이지 않는 이유는 이미지가 resize되면서 sub-sampling 후 주변 pixel과 blur가 되기 때문이다. 이로 인해 high frequency pixel이 주변의 low frequency pixel에 영향을 받아 사라지게 된다.

3. 결론

이번 실습을 통해 filtering 및 convolution, blurring, sharpening의 기능을 직접 만들어 볼 수 있었다. Gaussian convolution을 사용하여 low frequency image와 high frequency image를 만들 수 있다. 먼저 기초적인 box filter와 gaussian filter를 직접 만들고 Gaussian filter를 통해 blurring을 수행하였다. 그리고 2개의 정렬된 이미지를 각각 low pass와 high pass를 통하여 hybrid 이미지를 만들었다. 만들어진 hybrid 이미지로부터 이미지가 작아졌을 때 resize 되면서 이미지가 blur가 되고 high frequency가 사라지게 됨을 확인할 수 있었다.