HW 01 – REPORT

소속 : 정보컴퓨터공학부

학번 : 2 0 2 0 5 5 5 3 6

이름 : 민 예 진

1. 서론

1) 실습 목표

실습을 통한 이미지 데이터의 값을 직접 바꿔 봄으로써 이미지 데이터에 대한 이해를 돕는 것을 목표로 진행한다.

2) 이론적 배경 기술

① PIL 라이브러리

PIL란 Python Imaging Library의 약자로 파이썬 인터프리터에 다양한 이미지 파일 형식을 지원하고 이미지 처리 및 그래픽 기능을 제공하는 오픈 소스 소프트웨어 라이브러리이다. PIL를 통해 이미지를 픽셀 단위로 조작할 수 있고, 이미지 필터, 선명도, 색 보정 등의 화상 조정 등 여러가지 이미지 작업이 가능하다.

② Numpy 라이브러리

Numpy란 행렬이나 일반적으로 대규모 다차원 배열을 쉽게 처리할 수 있도록 지원하는 파이썬 오픈 소스 소프트웨어 라이브러리이다.

③ Luminance grayscale conversion

Grayscale이란 검은색과 흰색의 중간색인 회색의 색상으로 이미지를 표현하는 것으로 빛의 강도(Intensity)를 기준으로 하며 0부터 255까지의 값으로 이미지를 표현한다. 이때 검정색을 0, 흰색을 255로 표현한다.

PIL 라이브러리에서 'L' mode로 convert method를 실행하여 수행할 수 있다. convert method는 ITU-R 601-2 luma transform을 통해 RGB color space에서 YCbCr color space로 이미지를 변환한다. 이때 Y는 밝기 성분, Cb, Cr은 색차 성분을 뜻한다.

$$L = R * 299/1000 + G * 587/1000 + B * 114/1000$$

ITU-R 601-2 luma transform

④ Numpy의 tile 함수

tile(A, reps) : A라는 array를 reps의 수 만큼 반복하는 함수이다.

(A : input array / reps : A를 각 축으로 반복할 횟수)

2. 본론

1) 이미지 Open 및 이미지 정보 출력

```
im = Image.open('chipmunk.png')
print(im.size, im.mode, im.format)      # (750, 599) RGB JPEG)
im.show()                               # chipmunk.png 출력
```

print(im.size, im.mode, im.format)을 통해 chipmunk.png 파일의 사이즈가 (width, height) = (750, 599)임을 알 수 있다. 또한, 출력 이미지 모드는 RGB이며 이미지 형식은 JPEG임을 알 수 있다.

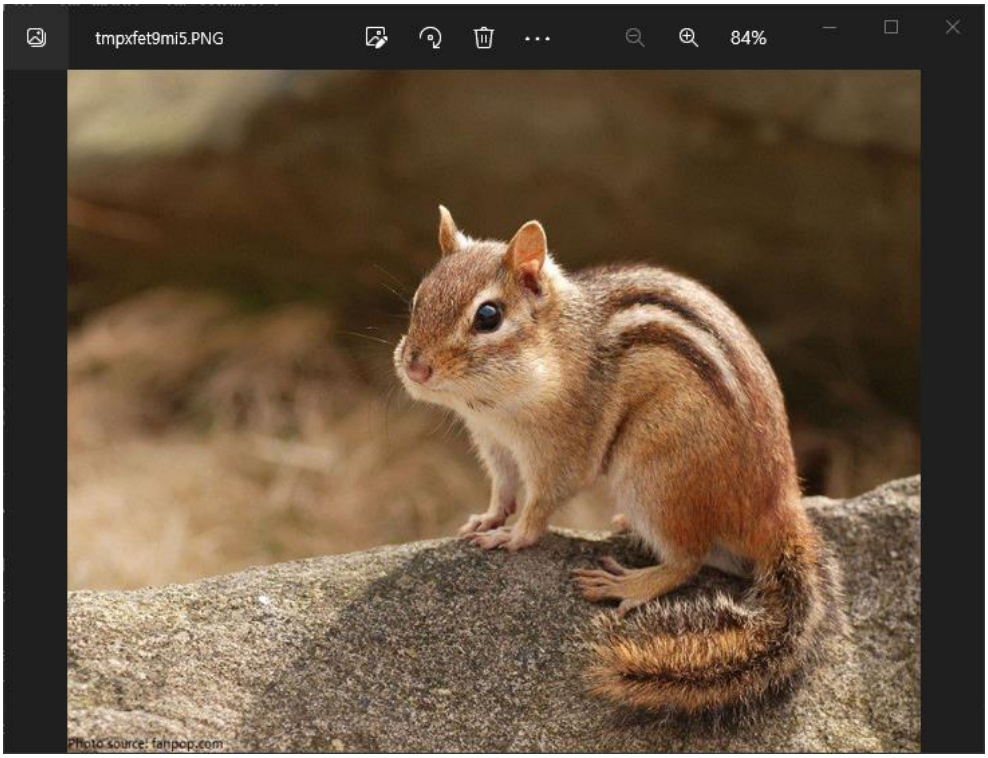

그림 1 chipmunk.png

im.show()를 통해 chipmunk.png 이미지를 볼 수 있다. 이때, 파일명으로 보아 show method는 이미지를 임시파일로 저장한 후 이미지를 보여줌을 알 수 있다.

2) Convert Luminance grayscale & Crop a image

그림 2 chipmunk_head.png

```
im = im.convert('L')
im2 = im.crop((280, 150, 430, 300))

im2_array = np.asarray(im2)
average = np.mean(im2_array)          # 125.47813333333333
```

그림 2는 그림 1에 convert('L') method와 crop method를 실행한 결과이다. ITU-R 601-2 luma transform을 통해 color 이미지를 grayscale 이미지로 바꾼 후 (좌, 상, 우, 하) = (280, 150, 430, 300)의 좌표로 이미지를 잘랐다. numpy의 asarray method를 통해 그림 2의 intensity의 평균을 구하면 125.48의 값을 얻을 수 있다.

3) Modify intensity by Numpy

그림 3 chipmunk_head_bright.png

```
im2_array = np.asarray(im2)
im3_array = im2_array.copy()
for x in range(0, 150):
    for y in range(0, 150):
        im3_array[y, x] = min(im3_array[y, x] + 50, 255)
```

그림 3은 chipmunk_head.png의 각 픽셀 값에 50 만큼 값을 더해 밝기를 높인 이미지이다.

그림 4 chipmunk_head_dark.png

```
im2_array = np.asarray(im2)
im4_array = im2_array.copy()
im4_array = im4_array * 0.5
im4_array = im4_array.astype('uint8')
```

그림 4는 각 픽셀의 intensity를 절반으로 줄이고 uint8로 형 변환 후 추출한 이미지이다.

4) Generating a gradient pattern image

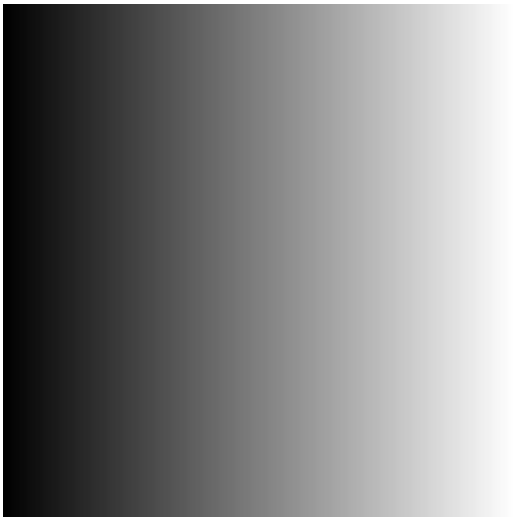

그림 5 gradient.png

그림5는 np.arange(0,256) = [0, 1, 2, ..., 255]이 256개 있는 256x256 이미지이다. 즉, 검은색부터 흰색까지의 색상을 전부 나타낸 이미지라고 볼 수 있다.

3. 결론

위 실습을 통해 이미지 데이터의 값을 조정해 봄으로써 이미지 데이터에 대한 이해를 돕고 Image 라이브러리를 직접 사용해 보는 활동을 할 수 있었다.

convert method를 통해 RGB 값을 Luminance grayscale 값으로 바꿔볼 수 있었다. Image를 numpy array로 바꿔 봄으로써 변환한 이미지 데이터 각 픽셀의 평균값을 더하니 약 125.48의 값으로 회색을 띄는 수치를 얻을 수 있었다. 또한, 각 픽셀에 50의 수치를 더하는 실습과 각 픽셀의 값을 절반으로 조정하는 실습을 통해 이미지의 밝기를 높이는 실습을 통해 값이 0에 가까울수록 검은색을 띄고 255(uint8의 최대값)에 가까울수록 흰색을 나타냄을 알 수 있었다.

마지막으로 np.arange(0, 256)의 array를 256개 만들어 256x256의 array를 Image로 만드는 실습을 통해 0부터 255까지 값의 색상을 직접 볼 수 있었다.

위 실습은 RGB 이미지를 grayscale로 변환하여 진행하였다. 실제로 grayscale를 사용하는 이유는 무엇일까? RGB는 R, G, B 즉, 3차원의 색공간을 가지고 있어 각각을 연산하기 위해서 $2^{8+8+8} = 2^{24}$의 연산량이 있지만 grayscale은 2^8가지의 값만 생각하면 되므로 연산량이 크게 줄어드는 것을 알 수 있다. grayscale은 image binarization(image thresholding)과 함께 이미지 전처리 과정에 사용되어 데이터의 폭을 줄여주는 역할을 한다.