# 수치해석 HW#5

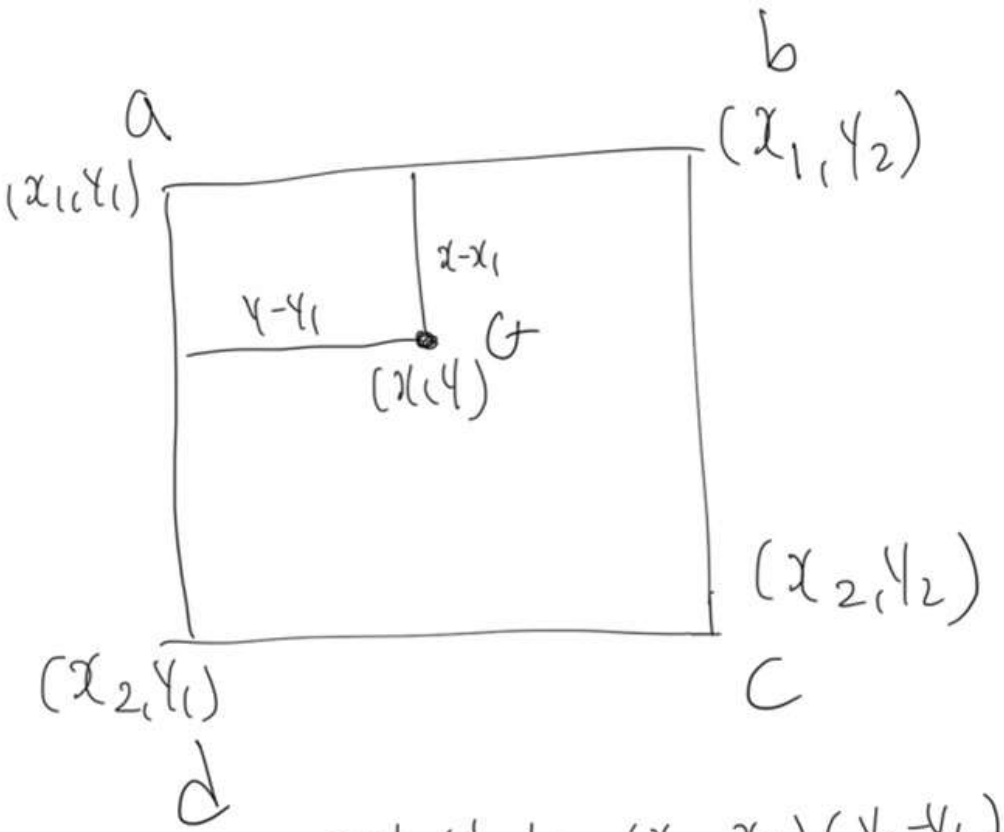
2018008559 컴퓨터소프트웨어학부 신상윤

# 1. Introduction

## 1. 목적

bilinear interpolation을 이용하여 임의의 이미지의 크기를 재설정한다.

## 2. bilinear interpolation을 이용한 resampling




전체 넓이 : $(x_2-x_1)(y_2-y_1)$

bilinear interpolation

$$f = a \times \frac{(x_2-x)(y_2-y)}{(x_2-x_1)(y_2-y_1)} + b \times \frac{(y-y_1)(x_2-x)}{(x_2-x_1)(y_2-y_1)}$$

$$+ c \times \frac{(x-x_1)(y-y_1)}{(x_2-x_1)(y_2-y_1)} + d \times \frac{(x-x_1)(y_2-y)}{(x_2-x_1)(y_2-y_1)}$$

## 2. Process

```
from PIL import Image
import numpy as np
import math

im = Image.open("simple.jpg")
pix = np.array(im)
im.show()
h, w = 3067, 5452
#resize
n, m = 100, 100
answer = np.zeros([n, m, 3], dtype=np.uint8)
```

목표 이미지인 simple.jpg를 읽어온다. 목표이미지의 크기는 3067 * 5452 픽셀이고, resampling을 원하는 이미지의 크기는 n, m을 임의로 설정해주면 된다.
resampling이후 결과는 answer에 저장된다.

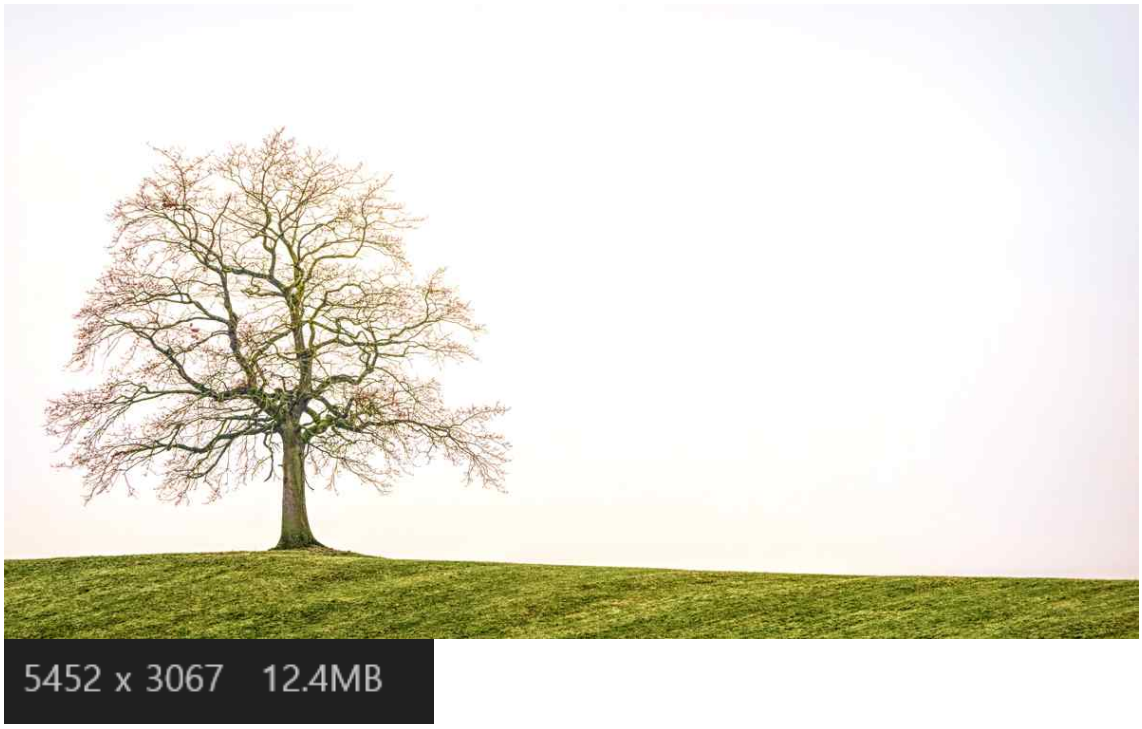
```
for i in range(n):
    for j in range(m):
        x, y = (h-1)*i/(n-1), (w-1)*j/(m-1)
        x1, x2 = math.floor(x), math.ceil(x)
        y1, y2 = math.floor(y), math.ceil(y)

        xx = (x-x1)/(x2-x1) if x1 != x2 else 0
        yy = (y-y1)/(y2-y1) if y1 != y2 else 0

        a = pix[x1][y1]
        b = pix[x1][y2]
        c = pix[x2][y2]
        d = pix[x2][y1]

        answer[i][j] = (1-xx)*(1-yy)*a + (1-xx)*yy*b + xx*yy*c + xx*(1-yy)*d

image = Image.fromarray(answer,"RGB")
image.show()
```

재설정 이후 이미지의 픽셀을 구해본다. 재설정 이후 이미지의 (i, j)는
원래 이미지의 크기 비율만큼의 좌표에서 온 것으로 해석할수 있으므로 원본이미지 좌표에서 interpolating을 하고 그 값으로 설정해주면 된다.

(x, y)는 비율대로라면 어디서 왔을지 추측한 좌표이고
x1, x2, y1, y2는 그 주변의 가장 가까운 격자점이다.
이를 이용하여 1-2를 적용하고, interpolating하고 값을 구한다.
최종적으로 answer 배열을 다시 이미지로 변환하여 출력한다.

# 3. Result

## 3.1 원본이미지 (n=3067, m=5452)



3.2 n = m = 2000

3.3 n = 1234, m = 5678

5678 x 1234 6.8MB

3.4 n = 500, m = 500

500 x 500 373.4KB

# 4. Discussion

## 4.1 linear interpolation

코드를 보면 x1, x2가 같을 때와 y1, y2가 같을 때 예외 처리를 했다. 이는 (x, y)에서 한 좌표가 격자 위에 존재하는 경우로 엄밀하게 처리를 하려면 점 2개에 대하여 lenear interpolation을 해주어야 한다. 즉, 실제로 경우가 4가지 존재한다.
하지만 주제가 bilinear interpolation이고, 넓이에 가중치를 주어 값을 결정하는 방법을 쓰고 있으므로 선일 때 가중치를 0으로 주는것과 같다고 생각한다면 크게 오차가 생기지는 않는다. 실제로 결과에서도 크게 문제가 될만한 부분은 없다.

# 5. Reference

[1] http://numerical.recipes/book/book.html