<그림 1>

<그림 2>

<그림 3>

<그림 1> = original
<그림 2> = original 그림을 가로 1.5배, 세로 2배 확대한 그림
<그림 3> = <그림2>가 캡쳐해서 붙여넣기로 하니 scaling정도가 티가 안나서, <그림 1>크기로 잘랐을때 scaling정도

target_resolution = (1259*1.5, 588*2)

● 생각보다 많이 깨지지않고 잘 되는 것 같다.

Code 설명

Visual Studio 2019에서 OpenCV package를 사용하여 C++로 구현하였다.
사용한 OpenCV 버전은 3.4.16이다.

```
Mat before = imread("sample.png", CV_LOAD_IMAGE_COLOR);
```

위 line을 통해 sample.png를 배열 형태로 저장 할 수 있다.

bilinear_interpolation 함수

```
void bilinear_interpolation(Mat& org_img, Mat& new_img) {
    double x_rate = (double)new_img.cols / org_img.cols;
    double y_rate = (double)new_img.rows / org_img.rows;


    for (int y = 0; y < new_img.rows; y++) {
        for (int x = 0; x < new_img.cols; x++) {

            int px = (int)(x / x_rate);
            int py = (int)(y / y_rate);
            double fx1 = (double)x / x_rate - px;
            double fx2 = 1 - fx1;
            double fy1 = (double)y / y_rate - py;
            double fy2 = 1 - fy1;

            double w1 = fx2 * fy2;
            double w2 = fx1 * fy2;
            double w3 = fx2 * fy1;
            double w4 = fx1 * fy1;

            Vec3b P1 = org_img.at<Vec3b>(py, px);
            Vec3b P2 = org_img.at<Vec3b>(py, px + 1);
            Vec3b P3 = org_img.at<Vec3b>(py + 1, px);
            Vec3b P4 = org_img.at<Vec3b>(py + 1, px + 1);

            new_img.at<Vec3b>(y, x) = w1 * P1 + w2 * P2 + w3 * P3 + w4 * P4;
        }
    }
}
```

* RGB 3 color 채널을 사용하고자 하였기 때문에, Vec3b 형식을 사용하였다.

* 값을 구하고자 하는 점(x,y가 새롭게 mapping되는 곳)과 인접한 4개지점이 (px,py), (px,py+1), (px+1,py), (px+1, py+1)이다.

* 구하고자 하는 점과 인접한 4개의 점과(위에 명시) 의 거리비가 fx1, fx2, fx3, fx4가 된다.

```
new_img.at<Vec3b>(y, x) = w1 * P1 + w2 * P2 + w3 * P3 + w4 * P4;
```

각각의 거리비를 활용하여 구하고자 하는 점 new_img.at<Vec3b>(y,x)를 구한다.