1. 개요

<요구사항>

구현사항		구현여부
Billinear Interpolation	영상을 임의의 크기로 바꾸기	O
영상 회전	입력 영상을 x도 만큼 회전	O

2. 구현

2.1 Bilinear Interpolation

```
int main() {
	Mat img_in;
	double x_size = 0;
	double y_size = 0;
	cout << "x_size(128~1024) : ";
	cin >> x_size;
	cout << "y_size(128~1024) : ";
	cin >> y_size;
	//image 읽고 gray로 바꾸기
	img_in = imread("Lena_256x256.png");
	cvtColor(img_in, img_in, cv::COLOR_RGB2GRAY);
	imshow("Original img", img_in);

	double x_scale = (double)x_size/(double)255;
	double y_scale = (double)y_size/(double)255;

	int h = img_in.rows; //256
	int w = img_in.cols; //256

	int imgcstate =  CV_8UC1 ; //흑백
	Mat outimage((h-1)*y_scale, (w-1)*x_scale, imgcstate, Scalar(0));

	Bilinear_Interpolation(img_in, outimage, x_scale, y_scale);

	imshow("Output Image", outimage);
	waitKey(0);
	return 0;
}

void Bilinear_Interpolation(Mat &image1, Mat &image2, double x_rate, double y_rate) {
	for (int y = 0; y < image2.rows; y++) {
		for (int x = 0; x < image2.cols; x++) {
			int px = (int)(x / x_rate);
			int py = (int)(y / y_rate);
```

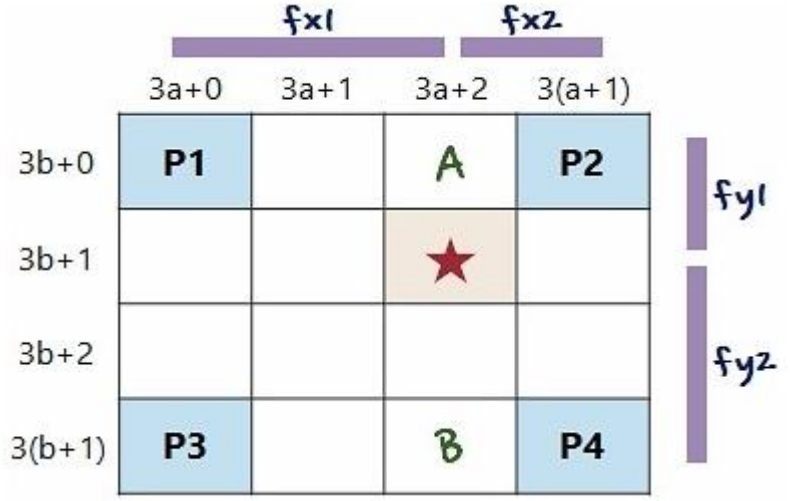
```
                double fx1 = (double)x / (double)x_rate - (double)px;
                double fx2 = 1 - fx1;
                double fy1 = (double)y / (double)y_rate - (double)py;
                double fy2 = 1 - fy1;

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

                uchar P1 = image1.at<uchar>(py, px);
                uchar P2 = image1.at<uchar>(py, px + 1);
                uchar P3 = image1.at<uchar>(py + 1, px);
                uchar P4 = image1.at<uchar>(py + 1, px + 1);
                image2.at<uchar>(y, x) = w1 * P1 + w2 * P2 + w3 * P3 + w4 * P4;


            }
        }
}
```

- 영상의 확대 비율은 rate = 입력된_size / 원본_size



- 값을 구하고자 하는 점과 인접한 4개 지점을 원 영상(image1)으로부터 찾는다.(P1, P2, P3, P4)

- P1의 좌표 : px = x/x_rate, py = y/y_rate

- P1을 기준으로 P2, P3, P4좌표를 구함.

- 네 점으로부터의 거리비 fx, fy를 구함.

- 네 점과 거리비로 구하고자 하는 점의 값을 구한다.

2.2영상 회전

```
int main() {
	Mat img_in;
	int degree;
	cout << "input degree : ";
	cin >> degree;
	//image 읽고 gray로 바꾸기
	img_in = imread("Lena.png");
	cvtColor(img_in, img_in, cv::COLOR_RGB2GRAY);
	imshow("Original img", img_in);


	int h = img_in.rows;
	int w = img_in.cols;
	int imgcstate = CV_8UC1; //흑백
	Mat outimage(h, w, imgcstate, Scalar(0));
	Geo_Rotate(img_in, outimage, degree);

	imshow("Output Image", outimage);
	waitKey(0);
	return 0;
}



void Geo_Rotate(Mat &image1, Mat &image2, int degree) {
	double h = image2.rows;
	double w = image2.cols;

	int centerY = h / 2;
	int centerX = w / 2;
	double seta = -degree * 3.14 / 180.0;

	for (int y = 0; y < h; y++) {
		for (int x = 0; x < w; x++) {
			int newY = (x - centerX)*sin(seta) + (y - centerY)*cos(seta) +
centerY;
			int newX = (x - centerX)*cos(seta) - (y - centerY)*sin(seta) +
centerX;


			if ((newX < 0.0) || (newX >= w) || (newY < 0.0) || (newY >= h)) {
				image2.at<uchar>(y, x) = 0;
			}
			else {
				uchar data = image1.at<uchar>(newY, newX);
				image2.at<uchar>(y, x) = data;
			}

		}


	}
}
```

- 회전 이후, 새로운 x, y좌표는 입력된 각 seta에 따라 다음과 같은 식을 만족함.

- newX = (x - centerX)*cos(seta) - (y - centerY)*sin(seta) + centerX

- newY = (x - centerX)*sin(seta) + (y - centerY)*cos(seta) + century

- 화면을 벗어나는 경우나 값이 없는 경우 0으로 처리 (newX < 0.0) || (newX >= w) || (newY < 0.0) || (newY >= h)

- 나머지 정상적인 경우 새로바뀐 newX, newY로 출력.

3. 결과

3.1 Bilinear Interpolation

- 왼쪽부터 원본(256x256), 436x436, 512x512

3.2 영상 회전

- 왼쪽부터 원본, 30, 45, 60도 회전

4. 논의

Bilinear Interpolation 문제점 해결.

위 코드는 값을 구할 때, P1 기준점이 항상 왼쪽 위(0,0)로부터 기준으로 오른쪽과 아래쪽으로 나머지 점을 기준으로 잡기 때문에, 마지막으로 배가 아닌 상수 x_scale만큼의 픽셀(오른쪽 끝, x_scale = 2인경우, 오른쪽 끝 2픽셀), y_scale만큼의 픽셀(아래끝)은 더 이상 오른쪽과 아래쪽을 기준으로 추가 할 수 있는 픽셀값이 원본이미지에 존재하지 않기 때문에 오류가 발생하여 이를 해결하기 위해 다음과 같이 시도하였다.

- 기준점 다르게 잡기.

```
if ((x <= image2.cols / 2) && (y <= image2.rows / 2)) {
                                int px = (int)(x / x_rate);
                                int py = (int)(y / y_rate);

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

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

                                uchar P1 = image1.at<uchar>(py, px);
                                uchar P2 = image1.at<uchar>(py, px + 1);
                                uchar P3 = image1.at<uchar>(py + 1, px);
                                uchar P4 = image1.at<uchar>(py + 1, px + 1);
                                image2.at<uchar>(y, x) = w1 * P1 + w2 * P2 + w3 * P3 + w4 *
P4;
                        }
                        else if ((x > image2.cols / 2) && (y <= image2.rows / 2)) {
                                int px = (int)(x / x_rate);
                                int py = (int)(y / y_rate);

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

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

```
                        uchar P1 = image1.at<uchar>(py, px-1);
                        uchar P2 = image1.at<uchar>(py, px);
                        uchar P3 = image1.at<uchar>(py + 1, px-1);
                        uchar P4 = image1.at<uchar>(py + 1, px);
                        image2.at<uchar>(y, x) = w1 * P1 + w2 * P2 + w3 * P3 + w4 *
P4;
                }
                else if ((x <= image2.cols / 2) && (y > image2.rows / 2)) {
                        int px = (int)(x / x_rate);
                        int py = (int)(y / y_rate);

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

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

                        uchar P1 = image1.at<uchar>(py-1, px);
                        uchar P2 = image1.at<uchar>(py-1, px + 1);
                        uchar P3 = image1.at<uchar>(py, px);
                        uchar P4 = image1.at<uchar>(py, px + 1);
                        image2.at<uchar>(y, x) = w1 * P1 + w2 * P2 + w3 * P3 + w4 *
P4;
                }
                else if ((x > image2.cols / 2) && (y > image2.rows / 2)) {
                        int px = (int)(x / x_rate);
                        int py = (int)(y / y_rate);

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

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

                        uchar P1 = image1.at<uchar>(py-1, px-1);
                        uchar P2 = image1.at<uchar>(py-1, px);
                        uchar P3 = image1.at<uchar>(py, px-1);
                        uchar P4 = image1.at<uchar>(py, px);
                        image2.at<uchar>(y, x) = w1 * P1 + w2 * P2 + w3 * P3 + w4 *
P4;
                }
```

범위를 다음과 같이 4분면 형태로 나누어 P1 기준점 위치를 다르게 잡아보았음.

실행 결과 다음과 같이 사분면의 경계가 눈에 보이는 결과가 나타났고, 이를 해결하지 못함.

- 계산, 출력범위 수정하기

```
double x_scale = (double)x_size/(double)255;
double y_scale = (double)y_size/(double)255;

int h = img_in.rows; //256
int w = img_in.cols; //256

int imgcstate =  CV_8UC1 ; //흑백
Mat outimage((h-1)*y_scale, (w-1)*x_scale, imgcstate, Scalar(0));
```

Scale을 구하는 과정에서 원본의 크기 256으로 나누는 것이 아닌 원본size-1값 255로 나누어주어, out image에서 사용자가 입력한 크기의 size로 출력되도록 바꿈.

기존은 상수(x_scale, y_scale)만큼의 픽셀이 계산도 되지않고, 출력도 되지않게 설정을 했었음.(사용자가 512x512를 입력한경우, scale=2 이므로, 510x510으로 출력되도록 설계했었음.)

수정이후, 입력 size대로 오류없이 정상적으로 출력됨.

5. 참고

https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=dic1224&logNo=220841161411