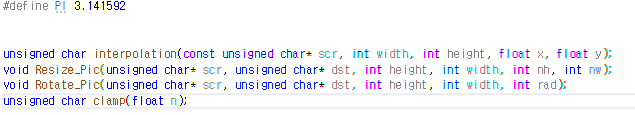
보고서

2020203037 최재민

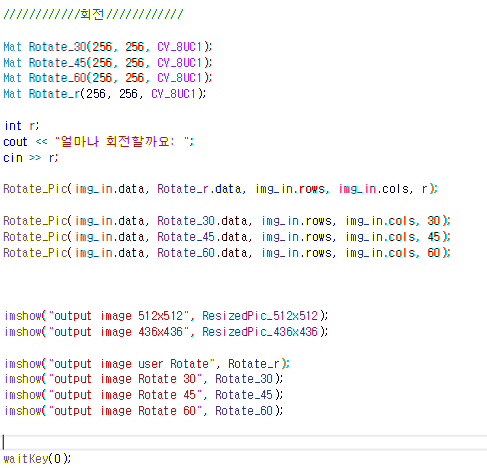


크기를 변환하거나 회전하기 위해 위와 같은 함수들을 사용했습니다.

1. Interpolation 함수는 input 영상을 받아 해당 좌표(x, y)의 interpolation을 한 결과를 반환해주는 함수입니다.
2. Resize\_Pic 함수는 input 영상을 통해 크기를 조절해주는 함수입니다. 인자로 input 영상의 data, 바뀐 영상을 저장할 배열, input 영상의 width와 height, Resize 할 영상의 width와 height을 받습니다.
3. Rotate\_Pic 함수는 input 영상을 rad도 만큼 회전시켜주는 함수입니다. 인자로 input 영상의 data, 바뀐 영상을 저장할 배열, input 영상의 width와 height, 그리고 Rotate 할 각도를 입력 받습니다.
4. Clamp 함수는 픽셀 값이 0이하 혹은 255초과를 할 경우를 대비하여 사용됩니다.

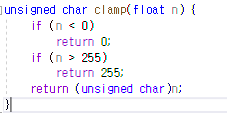


메인 함수에서 Resize를 실행하기 위한 코드입니다. 변수 h,w은 각각 조절될 크기를 저장하고 있습니다. 436\*436크기로 저장할 Mat 변수인 ResizedPic\_436x436와 512\*512 크기로 저장할 Mat 변수인 ResizedPic\_512x512를 만들어줬습니다.

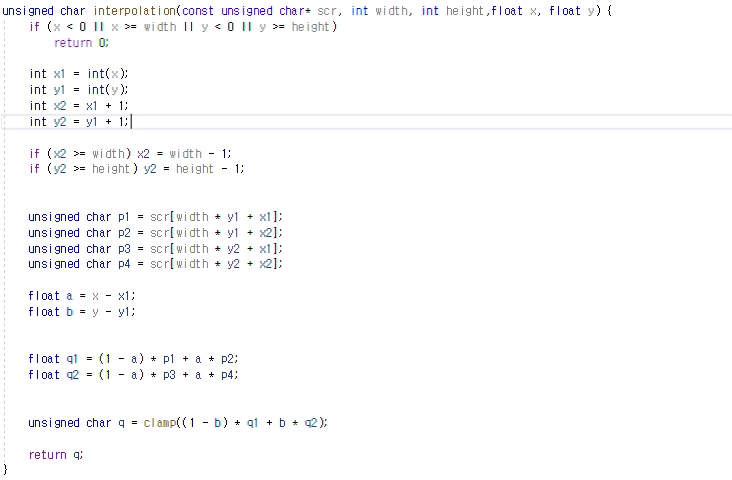


메인 함수에서 회전을 하기 위한 코드입니다. 회전할 각도를 입력 받아 r에 저장을 하고 Rotate\_Pic을 통해 input 영상을 회전시킨 output데이터를 Rotate\_r에 저장했습니다.

그리고 imshow를 통해 output 영상들을 시각화했습니다.



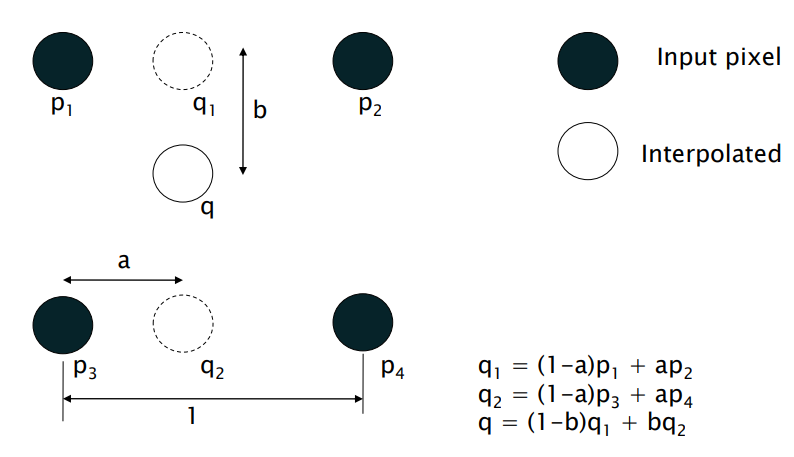
위 함수는 입력 받은 픽셀 값이 0보다 작으면 0으로 255보다 크면 255로 그 외에는 픽셀 값을 unsigned char형태로 변환하여 반환해주는 함수입니다.



위 함수는 인자로 float 형 좌표를 받아 input 영상에서 linear interpolation을 한 픽셀 값을 반환해주는 함수입니다.

Linear interpolation을 진행할 좌표가 input 영상 크기보다 클 경우 오류가 발생하므로 이때 픽셀값을 0을 반환하게 했습니다.

x1, y1, x2, y2는 각 p1, p2, p3, p4의 좌표를 구하기 위해 사용되었습니다. 그리고 x2, y2의 값은 input 영상의 크기인 width, height보다 커진다면 오류가 나기 때문에 조건문을 통해 크기를 제한했습니다.



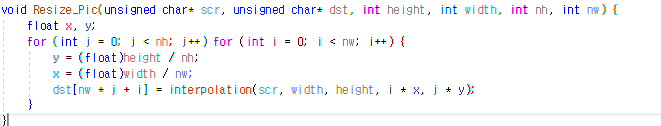
(x1, y1)은 함수 인자로 받은 float형 좌표에서 p1의 좌표를 뜻합니다.

(x2, y2)은 함수 인자로 받은 float형 좌표에서 p4의 좌표를 뜻합니다.

(x1, y2)은 함수 인자로 받은 float형 좌표에서 p3의 좌표를 뜻합니다.

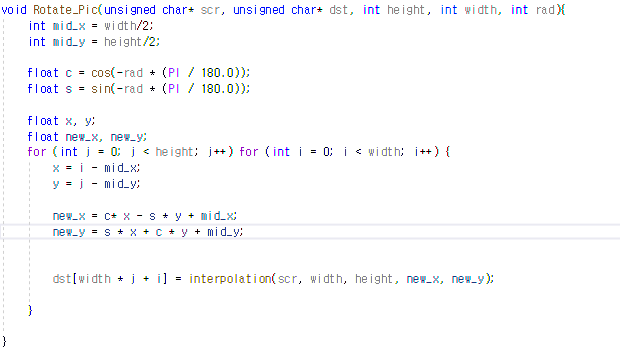
(x2, y1)은 함수 인자로 받은 float형 좌표에서 p2의 좌표를 뜻합니다.

따라서 이 좌표를 이용하여 input 영상의 데이터 픽셀 값에 접근했습니다. 그리고 위 사진처럼 공식을 대입하여 q1, q2 그리고 q를 구했습니다. 혹시 값이 255보다 크거나 0 보다 작으면 안되므로 clamp 함수를 통해 이를 방지해줬습니다.



위 함수에서 Resize를 할 크기인 nh, nw만큼 2중 반복문을 돌며 각 픽셀의 좌표가 기존 input 영상의 어디 좌표에 있는지 파악했습니다. 이때 사용된 수식은 height/nh 과 width/nw입니다. 이를 통해 가로와 세로가 어떤 비율로 크기가 조절되는지 알 수 있습니다.

각 픽셀 좌표에 이 비율(height/nh 과 width/nw)을 곱하게 된다면 input 영상에서의 좌표를 구할 수 있게 됩니다. 그 좌표를 interpolation 함수에 대입을 하여 픽셀 값을 가져와 저장했습니다.



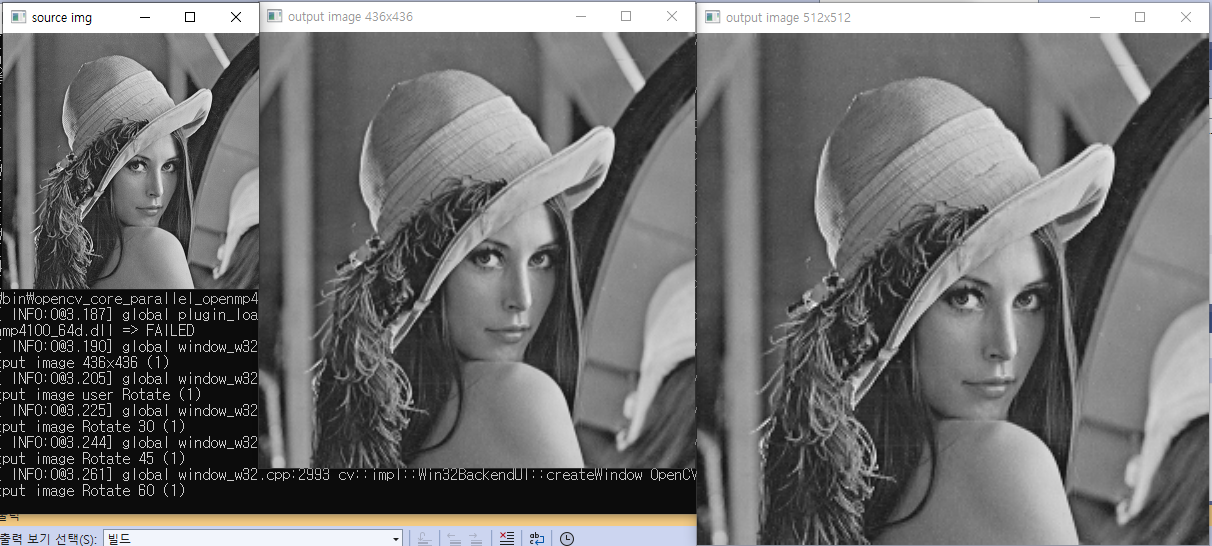
위 함수는 rad도 만큼 input 영상을 회전하는 함수입니다. 일단 input 영상의 중심을 기준으로 회전하기 때문에 가운데 좌표인 mid\_x 와 mid\_y를 구해줬습니다. 그리고 Image Rotation 을 위한 행렬계산을 위해 cos(-rad도), sin(-rad도)를 미리 구했습니다. (-rad 인 이유는 역으로 계산해야하기 때문)

반복문을 돌며 output 영상의 픽셀 좌표에 해당하는 픽셀 값을 찾기 위해 해당 픽셀 값에 해당하는 input 영상의 좌표를 구하기 위해 역으로 계산하는 과정을 거쳤습니다.

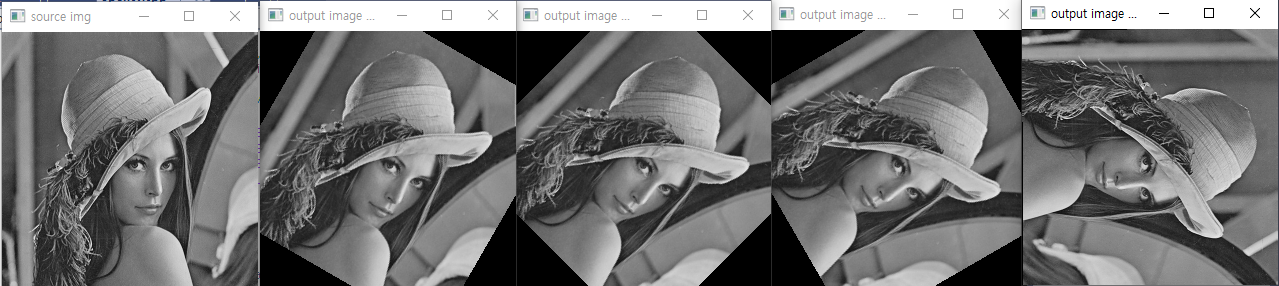
즉 input 영상을 회전하기 위해 각 좌표를 중심좌표만큼 빼고 rad도 만큼 회전한 후 다시 중심좌표만큼 더하는 과정이 있다면, 이를 역으로 계산함으로써 output 영상의 픽셀 값이 input 영상의 어떤 좌표에 해당하는지 알 수 있습니다.

이렇게 계산한 좌표를 interpolation 함수에 대입하여 interpolation 한 픽셀 값을 저장했습니다.

결과



436\*436 크기와 512\*512크기의 이미지

각각 원본, 30, 45, 60, 90(사용자 입력)을 회전한 이미지