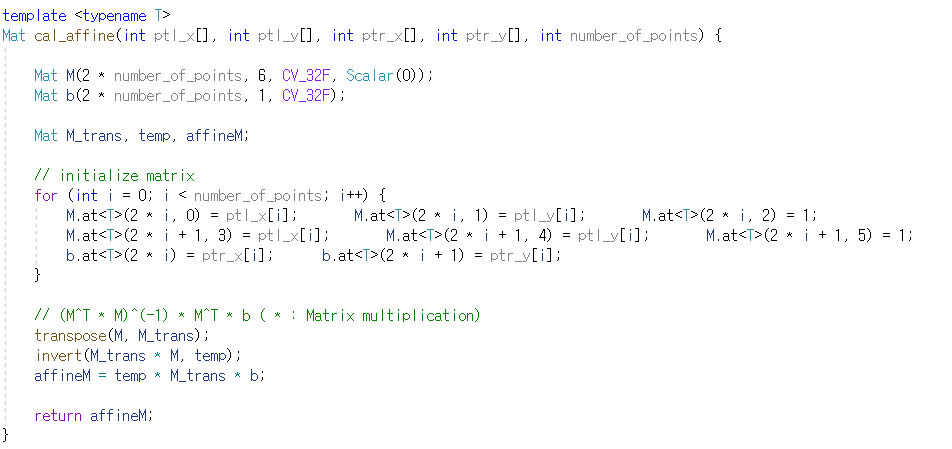
**2376029 김문경**

**동작 원리**

해당 코드는 두 이미지 L1, L2를 각각 불러와 서로 대응되는 픽셀들을 이용하여 파노라마 뷰처럼 이어 붙이기 위한 코드이다. 원하는 이미지를 목표 이미지의 좌표계로 바꾸기 위해 Affine transform에 필요한 미지수 6개를 cal\_affine 함수에서 구하고, 구한 matrix 값을 이용해 전체 이미지의 사이즈를 계산한 다음 inverse warping과 bilinear interploation을 한다. 추가로 blend stitching 과정도 거친다.

**함수 설명**

**1. cal\_affine() 함수**



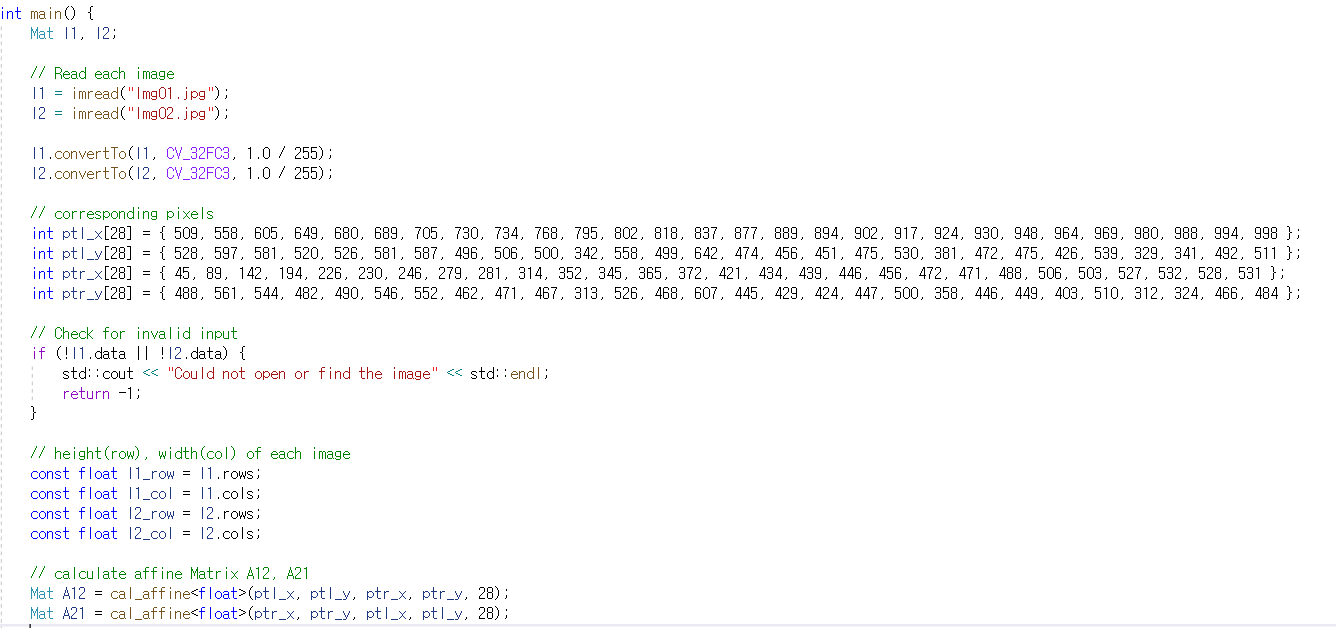
이미지 스티칭을 하기 위해, affine matrix를 계산하는 과정이다.

Affine 변환을 위해 6개의 미지수를 찾기 위해서, 를 이용해서 계산한다.

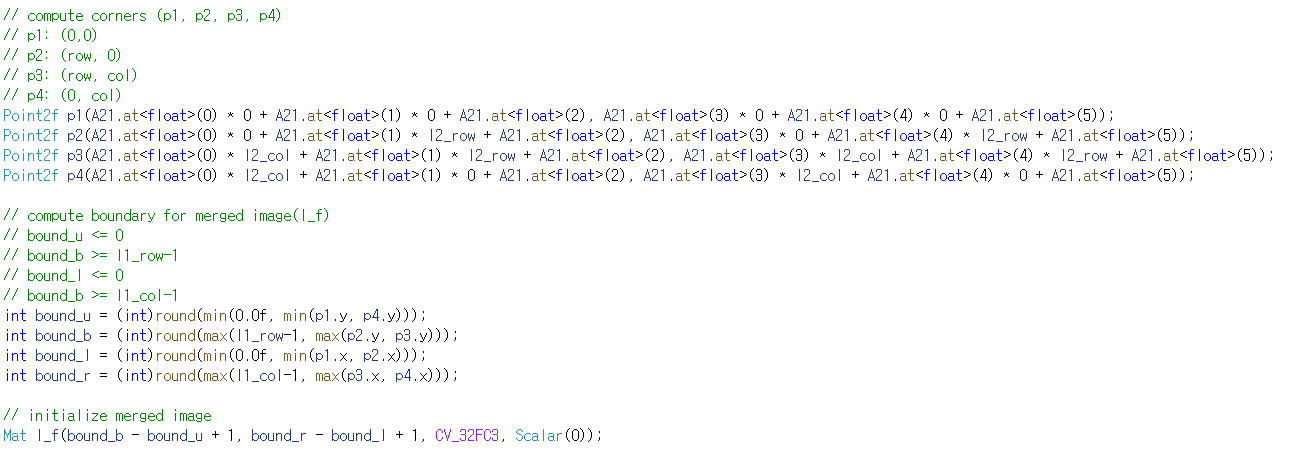
위의 함수는 left 이미지에서 right 이미지로 affine transform하는 과정으로, M 행렬에 left 이미지의 픽셀들을 넣고, b 행렬에 right 이미지의 픽셀들을 넣는다.

최소자승 해 공식을 이용해서 affine matrix 값을 구하고, return 한다.

**2. main() 함수**



선언된 Corresponding pixels의 배열들은 각각 left, right 이미지의 x좌표와 y좌표이다. 이미지 스티칭을 위해 해당 값들을 affine 함수를 이용해서 affine matrix를 구한다.

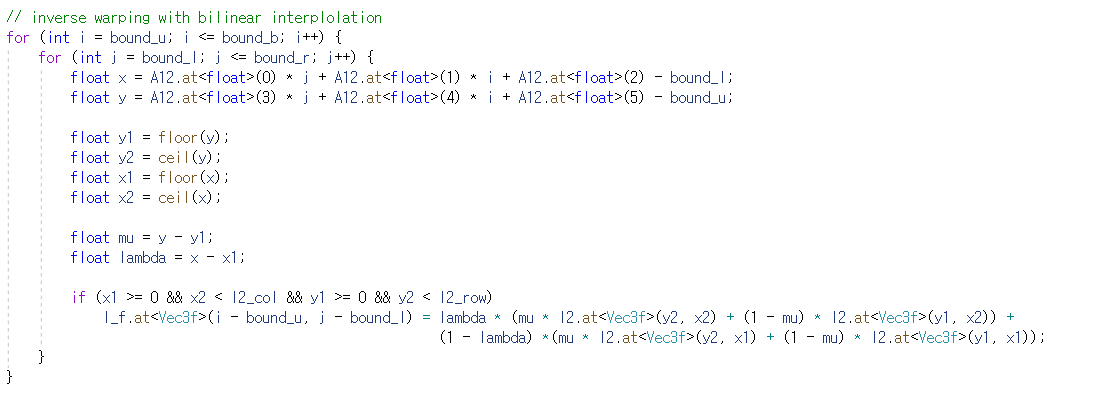


Merged image인 l\_f의 사이즈를 구하는 과정이다.

Point2f p3(A21.at<float>(0) \* I2\_col + A21.at<float>(1) \* I2\_row + A21.at<float>(2), A21.at<float>(3) \* I2\_col + A21.at<float>(4) \* I2\_row + A21.at<float>(5));

부분을 자세히 보자. A21.at<float>(숫자인덱스)의 의미는 위에서 계산한 A21 affine matrix의 파라미터를 말한다. 따라서, L2 이미지의 col, row 좌표들을 이용해서 새로운 이미지의 꼭짓점 좌표를 계산한다. (L2에서 L1으로)

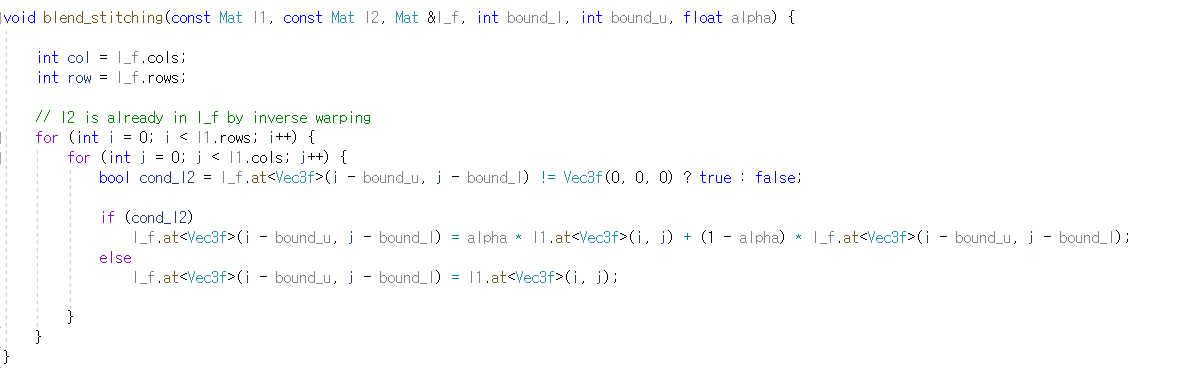
위에서 구한 꼭짓점 좌표를 이용해서 최종 이미지의 경계를 계산한다.



Inverse warping과 bilinear interpolation하는 과정이다.

원본 좌표(L1)의 위치를 A12 affine matrix (A21의 역행렬)를 이용해서 구한 float x, float y로 둔 다음, if문을 통해 L2 이미지의 점을 bilinear interpolation했다.

**3. blend\_stitching() 함수**



왼쪽 이미지의 모든 점들을 훑어본다.

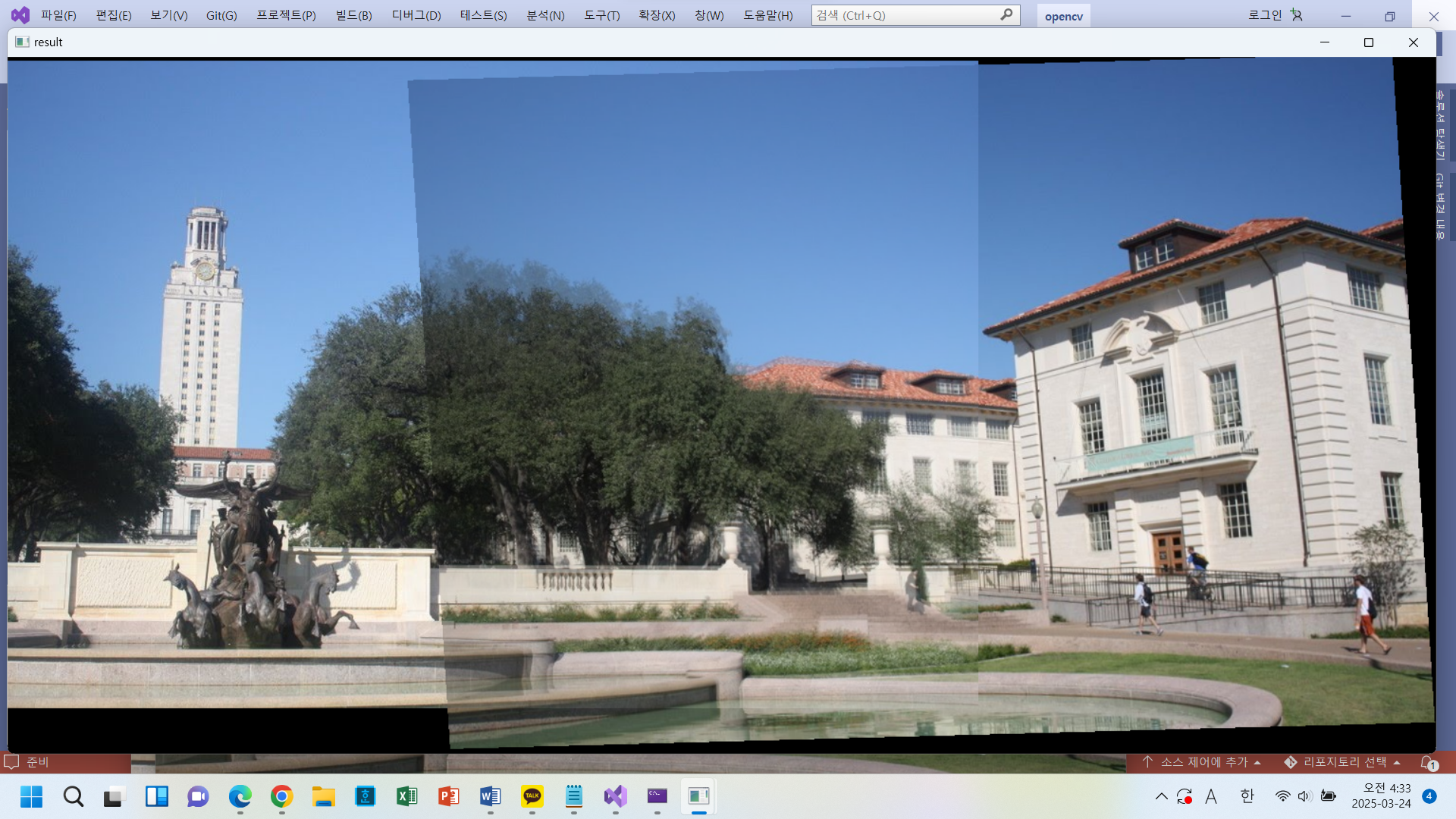
I\_f.at<Vec3f>(i - bound\_u, j - bound\_l) = alpha \* I1.at<Vec3f>(i, j) + (1 - alpha) \* I\_f.at<Vec3f>(i - bound\_u, j - bound\_l);

최종 이미지에 만약 픽셀이 채워져 있다면, alpha 비율(main 함수에서는 0.5로 설정)로 왼쪽 이미지와 final 이미지를 blending 한다.

I\_f.at<Vec3f>(i - bound\_u, j - bound\_l) = I1.at<Vec3f>(i, j);

아닐 시, l1 값을 l\_f 에 넣는다.

**최종 이미지**

****

전체 이미지의 크기가 증가하여 파노라마 뷰처럼 L1이미지와 L2이미지가 붙게 된다. 하지만 겹치는 부분의 이미지가 흐릿하여 잘 보이지 않는다는 단점이 있다.

더 많은 대응점을 구하거나, bilinear interpolation보다 bicubic interpolation 방식을 사용하거나, blend\_stitching 함수에서 alpha 값을 더 크게 한다면 더욱 선명해질 것이다.