<모바일응용 소프트웨어 설계>

# 12주차 실습 과제

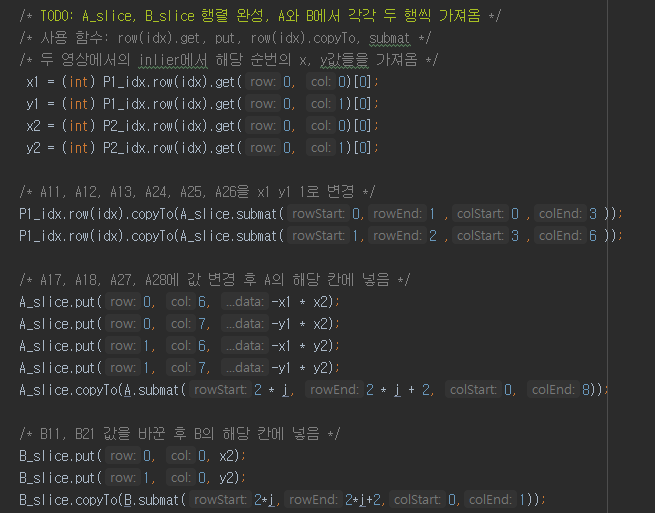
정보통신공학과

12161774 이 채 은

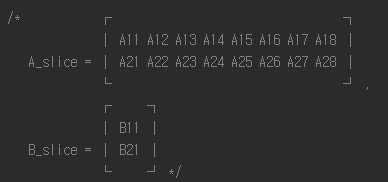
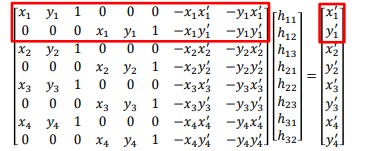
**[소스 코드]**



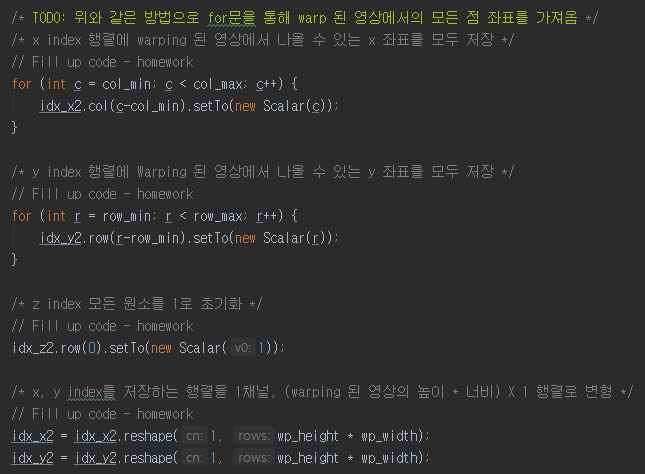
1. MainActivity.java : Homography\_RANSAC( )



=> Homography 행렬을 구하기 위한 과정으로, 8x8 행렬로 나타난 Mat A 와 8x1 행렬로 나타난 Mat B에서 각 두 줄씩을 뽑아낸다. 한 점 x, y 에서 각 좌표가 한 줄씩 표현 되어있기 때문에 A\_slice, B\_slice로 나누는 것이다. 나누는 과정에는 submat 을 이용했는데, (r.s, r.e, c.s, c.e) = (이상, 미만, 이상, 미만) 으로 행렬에서 필요한 부분만 뽑아올 수 있다.

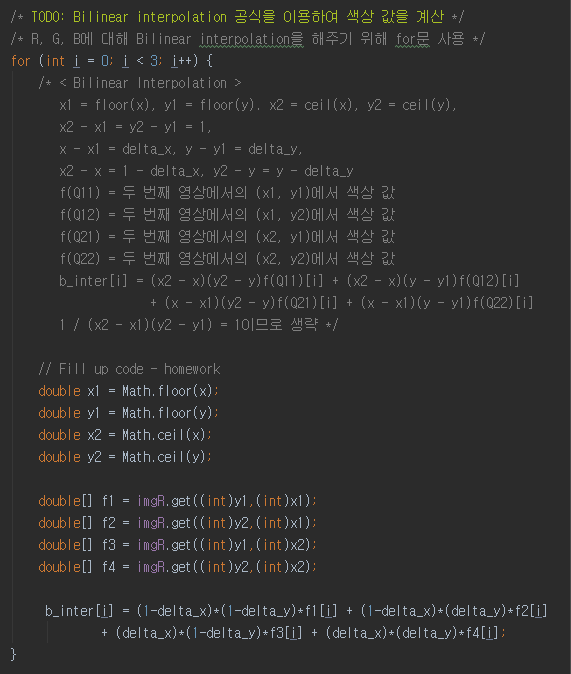
=> 왼쪽의 행렬은 A \* H = B 행렬을 나타낸 것이다. 두 줄은 하나의 점의 x, y 좌표를 나타낸다. 두 줄을 따로 뽑아서 오른쪽 행렬과 같이 A\_slice, B\_slice 를 만들어서 계산에 이용하는 원리이다.

2. MainActivity.java : backward\_warping( )



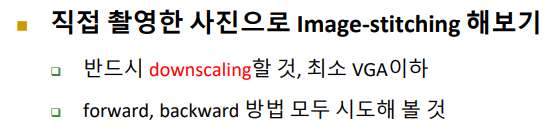
=> Forward warping 과 같은 과정을 진행한 뒤에 warping 된 이미지에서 나온 x, y 좌표를 모두 저장하고, z 좌표는 1로 초기화해준다. Col, row 모두 0부터 시작하도록 c – col\_min, r – row\_min으로 설정하고, setTo(new Scalar(c))로 새로운 스칼라 값 c 로 저장한다. 그리고 이 x, y index를 저장하는 행렬을 reshape 함수를 이용해서 cn(채널 값) 1, warping 된 영상의 높이\*너비를 행의 개수로 갖는 행렬로 변형시켜준다.

3. MainActivity.java : backward\_warping( ) – Bilinear interpolation

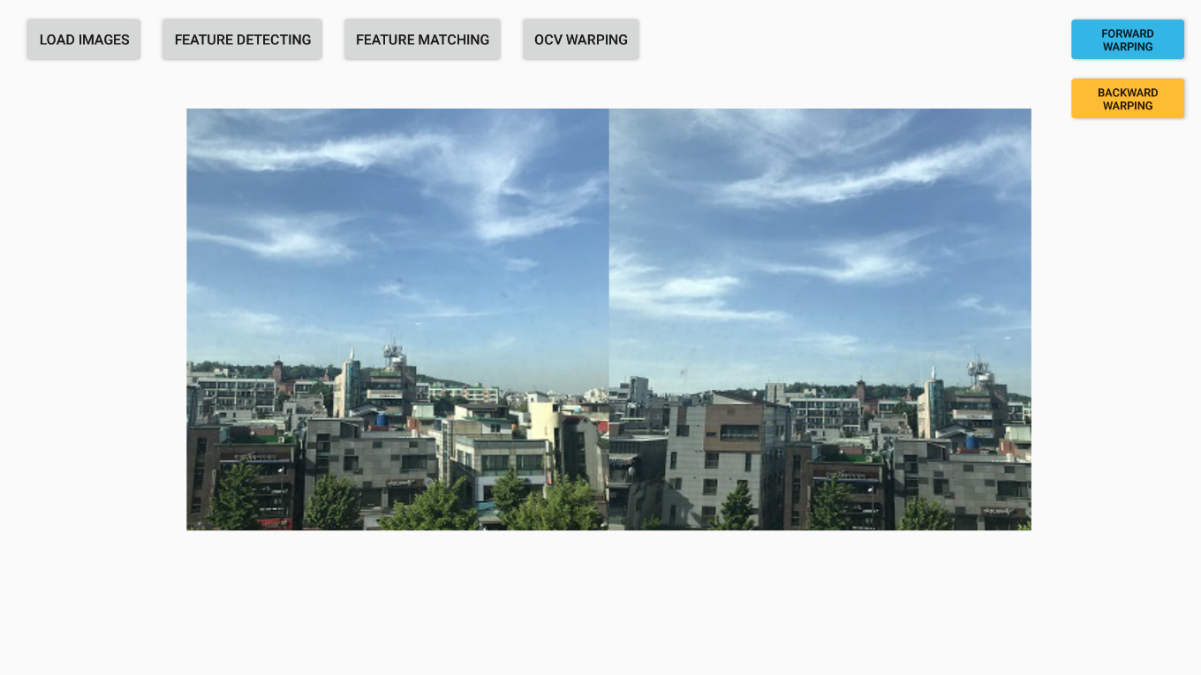


=> Backwarding 과정 중 Bilinear interpolation에 대한 코드이다. 선택한 좌표는 x, y 이고 Math.floor 와 Math.ceil은 내림과 올림이다. 좌표가 소수이기 때문에 정수 값으로 나타내는 과정이다. 따라서 정수 좌표 x1, y1 과 x2, y2로 나타낼 수 있다. 그리고 imgR.get 으로 두번째 영상에서 색상 값을 받아온다. 그 값을 f1, f2, f3, f4에 저장하고 interpolation 계산식에 대입해준다.

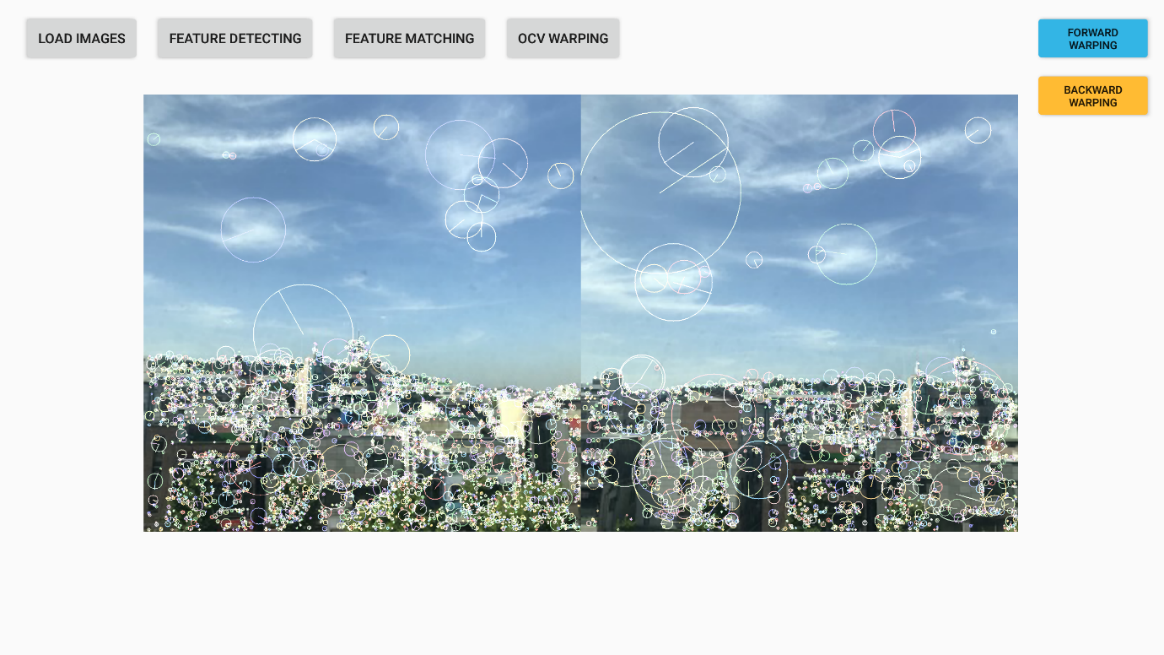
**[ 실습 결과 및 고찰 ]**



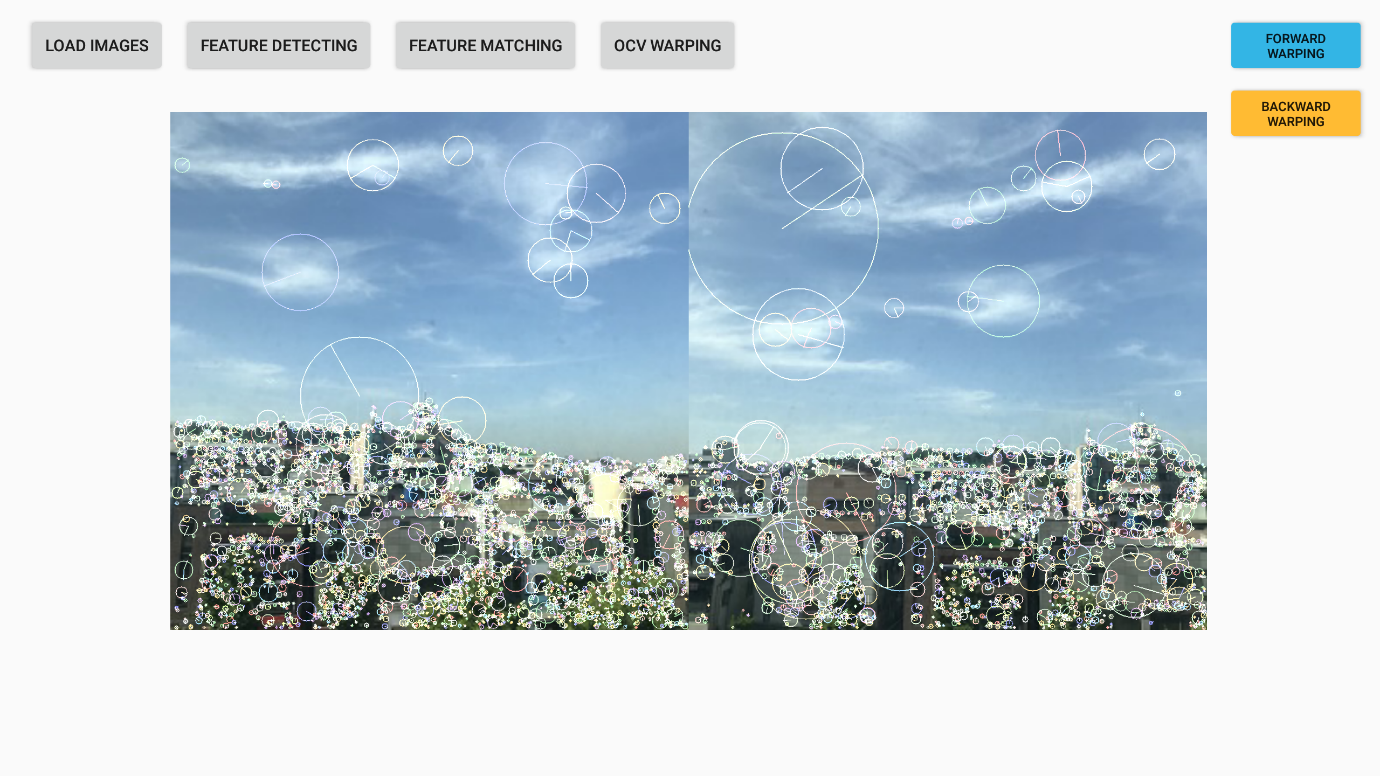
1. LOAD IMAGES



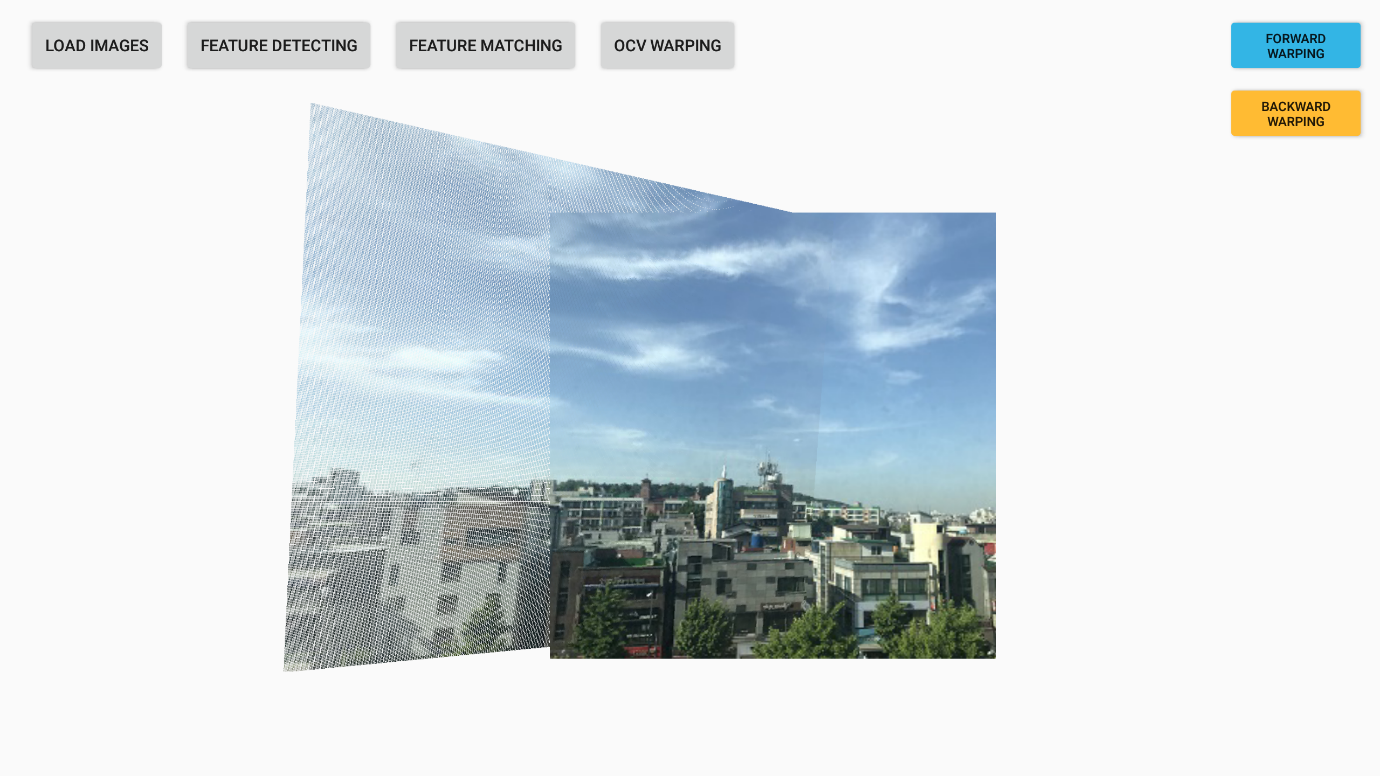
2. FEATURE DETECTING



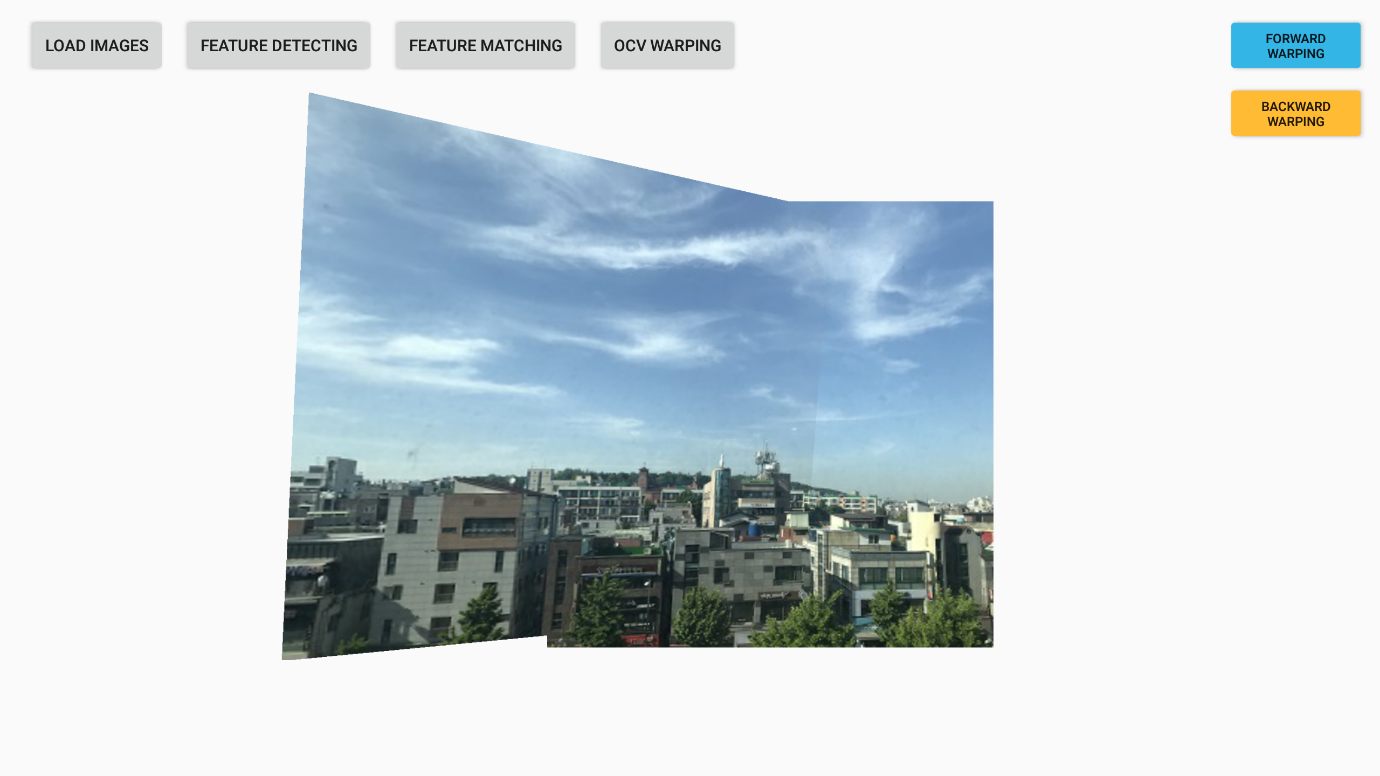
3. FEATURE MATCHING

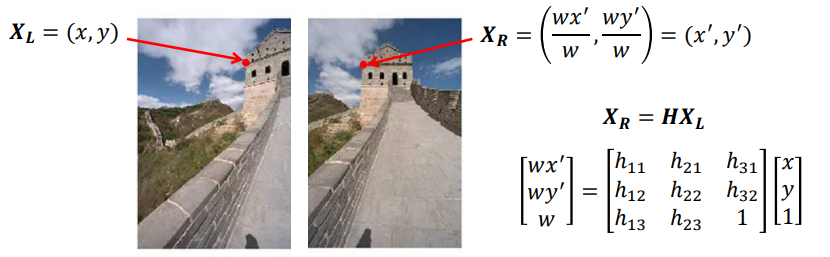


4. FORWARD WARPING



5. BACKWARD WARPING



=> 이번 과제를 통해서 image stitching 에 대해서 공부할 수 있었다. Image stiching 은 두 장 이상의 이미지를 붙여서 파노라마와 같이 한 장의 이미지로 만드는 방법이다. 저번 실습시간에 진행한 feature detecting과 feature matching 을 차례로 진행하면 이미지 및 특징점이 등록된다. 그 후 Homography 를 이용해서 image warping 과 blending을 하고 그 결과 파노라마 이미지가 나오게 된다. 이 때 Homography 는 여러 이미지 사이의 관계를 나타내는데, 계산을 위해 matrix H로 이용한다. 계산 시에는 오른쪽과 같은 식으로 나타낼 수 있는데, 첫 번째 이미지의 좌표를 x, y 라고 하면 새로 붙일 두 번째 이미지의 좌표를 x’, y’ 이라고 할 수 있다. 따라서 Homography 행렬 H를 이용해서 첫 번째 이미지에서 두 번째 이미지로 넘어가는 좌표를 구할 수 있다. H에는 h11 부터 h32까지 총 8개의 인자가 있다. 이 인자는 2D 좌표 4개를 이용해서 구할 수 있다. 이 4개의 좌표는 Homopraphy 행렬 H를 구하기 위한 후보군인데, 구한 H를 이용해서 model에 적절한 점인지 아닌지를 파악하기 위한 과정을 거쳐야 한다. 이 과정을 통해서 우리가 사용하고자 하는 적절한 점인 inliers와 필요하지 않은 outlier 이 나눠지게 된다. 따라서 좀 더 정확한 modeling 이 가능해지는 것이다. 따라서 이 때 보다 정확한 inlier을 뽑아 내기 위해 사용하는 것이 RANSAC 알고리즘이다. 이전의 feature detecting에서 이용된 KNN 알고리즘의 결과로 good\_match 된 점들이 matching 된 특징점으로 이용된다. 이 특징점 중에서 4개를 random 하게 뽑아서 H를 계산한다. 유클리디안 distance threshold 값을 이용해서, 이 threshold 값을 넘을 때까지 계속해서 반복한다. 이 때 threshold 는 dst 인데 오른쪽과 같이 argmin을 이용한다. 그 이유는 Ax-B가 0에 가까울수록 정확한 결과이기 때문이다. 따라서 이 threshold 값보다 작은 distance 값을 갖는다면 inlier 로 간주한다. 그리고 좀 더 정확한 modeling 을 위해서 위 과정을 통해 구한 inlier 를 통해서 한번 더 homography matrix 를 구한다. 이 결과 도출된 최종 행렬을 H로 결정하는 방식이 RANSAC 이다. 그 다음 과정은 구한 Homography 행렬을 이용해 warp 됐을 때 좌표를 저장하고, warp 된 좌표의 색상 값을 채워서 forward warping 을 완료하는 것이다. 이 결과는 위의 4번 사진을 통해서 확인할 수 있었다. Image 를 저장할 때 왼쪽에 있는 이미지를 img\_r로 잘못 저장해서 위와 같이 warping 후의 이미지가 왼쪽으로 가게 됐다. Warping 의 결과를 보면 중간중간 하얗게 비어서 나오는 곳이 있는 것을 볼 수 있었는데, 이는 좌표계 변환 중에 hole 이 생겼기 때문이다. 따라서 빈 공간(hole)을 제공하기 위해 hole 의 RGB 값을 원본 영상에서 다시 가져오는 Backward warping 과정이 필요하다. 이 때 bilinear interpolation 이 사용된다. Backward warping 에서는, 먼저 forward warping과 같은 과정을 진행한 후 Bilinear interpolation으로 hole 부분을 채워준다. Hole이 생기는 부분은 forwarding 변환 후 좌표가 정수가 아닌 소수부분이다. 따라서 이 소수부분은 주변 4개의 정수부분의 좌표를 이용해서 계산한다. 근처에 있는 좌표의 색상 값과 면적을 가중치를 둬서 적당한 값을 구하는 원리이다. 따라서 Hole 부분에 먼저 H의 역행렬을 곱해서 원래 좌표계로 복원시킨 뒤에 위의 Bilinear interpolation 기법을 적용해 hole이 없도록 backwarding 과정을 마친다. 실습 결과 위의 5번 그림과 같이 hole 없이 두 개의 이미지가 합쳐진 것을 확인할 수 있었다.