1. transform_homography()
    a. Given the src coordinates and homography matrix, matrix multiply h_matrix with each coordinate (appended with 1) transpose.
    b. Scale the transformed coordinates back down to form [x, y, 1]

2. warp_image()
    a. Given the dst, src and homography matrix, construct a matrix of coordinate points in dst.
    b. Transform each coordinate point in dst back to src image coordinates using transform_homography using inv(homography).
    c. Remap the src image over the dst image.

3. compute_affine_rectification()
    a. Given 2 pairs of parallel line, find the vanishing point of each pair and cross these 2 vanishing point to get the vanishing line at infinity.
    b. Parameterize this vanishing line and construct Hp.
    c. Calculate the new size of transformed image and warp the src image onto this new size.

4. compute_metric_rectification_step2()
    a. Given 2 pairs of parallel lines, construct the constraint for each pair of parallel line and stack them to form a 2x3 matrix.
    b. Use svd to get the null vector (last row of Vh) and contruct S from the null vector.
    c. Using Cholesky decomposition to get the matrix K and scale it down by its determinant.
    d. Convert K to a 3x3 matrix in form [[K.Transpose, 0], [0, 1]]
    e. Calculate the new size of transformed image and warp the src image onto this new size.

5. compute_metric_rectification_one_step()
    a. Given 5 pairs of orthogonal lines, construct the constraint for each pair and stack them to form a 5x6 matrix.
    b. Use svd to get the null vector (last row of Vh) and construct the conic in form of 3x3 matrix [[a, b/2, d/2], [b/2, c, e/2], [d/2, e/2, f]].
    c. Use svd again to get the D = sqrt(∑) and Homography matrix = UD
    d. Calculate a similarity/translation matrix to scale/translate the warped image back into view window.
    e. Warp the src image into the dst using H = (Hs*H)

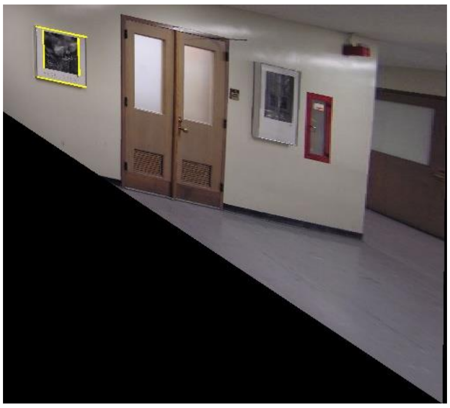

Figure 1 Affinely Rectified Image

Figure 2 Final Rectified Image 1

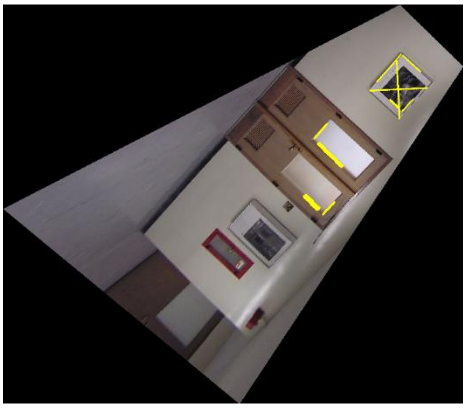

Figure 3 Final Rectified Image 2

6. compute_homography_error()
    a. Given the src(x), dst(x') and homography matrix, calculate the estimated dst(Hx) and src (inv(H)x') coordinates using the homography matrix.
    b. Calculate the distance error using formula, d(x,x') = ||x - inv(H)x'||^2 + ||x' - Hx||^2

7. compute_homography_ransac()
    a. Given 2 sets of corresponding points from 2 images, randomly choose 4 pairs of corresponding points.
    b. Compute the homography matrix using these 4 pairs of points.
    c. Count number of inliers according to distance error below threshold
    d. If number of inliers > current max inliers, save the homography matrix and inliers mask.
    e. Repeat for the specified number of tries to find max inliers, homography and mask
    f. Recalculate homography with the inliners.