Student: Li Hongguang (A0233309L)

Implementation: transform_homography()

1. Convert Euclidean coordinate to homogeneous representation.
2. Apply homography to homogeneous coordinates.
3. Convert transformed homogeneous coordinates to Euclidean coordinates.
4. Return list of transformed euclidean coordinates

Implementation: warp_image()

1. Create coordinate matrix ($H_{dst}W_{dst}$*2) with the shape of the destination image.
2. Use inverted homography to transform each destination coordinate to its corresponding source coordinate using **transform_homography()**
3. Map the pixel values of the destination image at every coordinate to the corresponding source coordinate of the source image computed in step 3.

Implementation: compute_affine_rectification() (Fig 1)

1. Find intersection of two pairs of parallel lines x_0 and x_1 in the distorted image using inbuilt **intersection_point()** function
2. Construct parameters of l_∞ in distorted image by taking cross product of x_0 and x_1
3. With parameters of l_∞, we find H_p^T and thus H_p^{-1}.
4. We apply H_p^{-1} to the image using **warp_image()** to complete the affinity rectification

Implementation: compute_metric_rectification_step2() (Fig 2)

1. Create 2x3 constraint matrix M using the parameters of the 2 pairs of orthogonal lines.
2. Perform SVD on M to get U, $\sum$ and V^T.
3. Extract the last row of V^T to get the solution (up to scale) (s_{11}, s_{12}, s_{22}) to the right null space of M.
4. Construct symmetrical positive-definite S (= KK^T) matrix using parameters computed from step 3.
5. Perform Cholesky decomposition on S to get K.
6. Use computed K to define H_A and thus H_A^{-1}
7. Apply H_A^{-1} to image that has been affinely rectified image to complete metric rectification.

Implementation: compute_metric_rectification_step1() (Fig 3)

1. Create 5x6 constraint matrix M using the parameters of the 5 pairs of orthogonal lines.
2. Perform SVD on M to get U, $\sum$ and V^T.
3. Extract the last row of V^T to get the solution (up to scale) (a, b, c, d, e, f) to the right null space of M.
4. Using parameters compute from step 3, construct the parameterizing matrix C of C_∞^* on the perspective image.
5. Perform SVD on C to get U_C, $\sum_C$ and V^T_C.
6. Replace last diagonal element of $\sum_C$ with 1
7. Take the inverse of $U_C \bullet \sum_C$ to get the inverse homography H^{-1}
8. Apply H^{-1} to the image using **warp_image()** to complete metric rectification.

Implementation: compute_homography_error() (Omitted as trivial to implement)

Implementation: compute_homography_ransac()

1. Randomly select sample of 4 correspondences
2. Using selected sample, estimate the homography
3. Using **compute_homography_error()** and given threshold value, compute number of inliers for dataset under estimated homography
4. Choose the homography estimated from the sample with highest number of inliers.
5. Return the homography from step 4 and the mask computed using the homography.

Student: Li Hongguang (A0233309L)

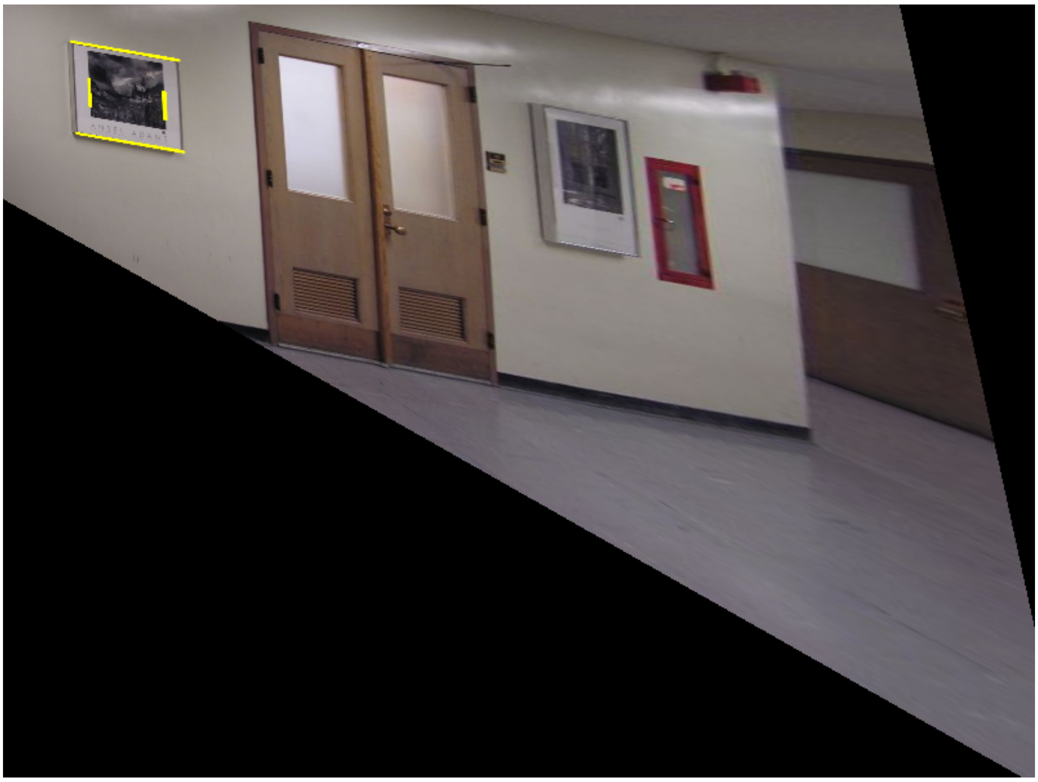

Fig 1. Result from affine rectification (Scaled to fit)

Fig 2. Result from metric rectification on image from Fig 1 (Scaled to fit)

Fig 3. One-step metric rectification result (Scaled, Rotated & Translated to fit)