NUS CS5477: 3D Computer February 8, 2024

Assignment 1

Lecturer: Lee Gim Hee *Student: Zhang Rongqi A0276566M*

In `transform_homography()`, I convert the coordinates into $\mathbb{P}^2$ to multiply the projective matrix H and then convert them back to $\mathbb{R}^2$.

In `warp_image()`, `cv2.remap()` receives an image `src`, and a map from `dst` to `src` to map the `src` to `dst` backward. To get the map from `dst` to `src`, I multiplied the inverse of the transformation matrix H with the coordinates of the `dst` image.

In `compute_affine_rectification()`, I used 2 pairs of parallel lines to find the intersection point and found I_∞, and further calculated H'_p with $H_A = I$. To view the output image properly, I implemented a function called `construct_hs()` to generate the similarity transformation H_s which can rescale and relocate the output image.

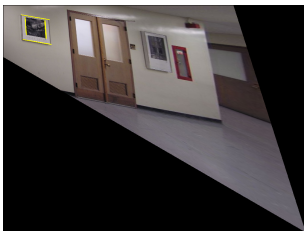
In `compute_metric_rectification_step2())`, I construct the A with 2 pairs of orthogonal lines, get KK^T by SVD , and get K by Cholesky decomposition. And further construct the required H_a. Using the inverse of H_a, I get the final rectified image.

In `compute_metric_rectification_onestep()`, I used 5 pairs of orthogonal lines to construct A. Solving it by SVD, I get the C^*_∞ with noisy. Letting singular value $S_3 = 0$ and $S_2 = S_1$, I get the approximated $C^*_\infty{}'$. Decomposing $C^*_\infty{}'$ again by SVD, left singular vector matrix U is obtained, which is exactly H_aH_p.

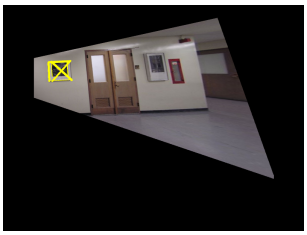
In `construct_hs()`, I generate the image rectangle by tracking the four corner points after transformation and move the whole image to the center of the output canvas. To avoid changing angles, the scale of x-axis should be the same with the scale of y-axis.

In `compute_homography_error()`, I simply compute the 2-norm of the differences between points and transformed points.

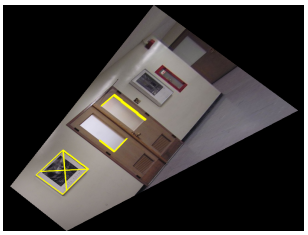
In `compute_homography_ransac()`, I randomly choose 4 pairs of points which are not collinear with each other. Helper function `compute_homography` is used to compute homogeneous transformation H. I check the number of inliners under H, and update `mask` and `h_matrix` which store the state corresponding to the H_{max} which has the most inliners.



(a) affinely rectified image



(b) final rectified image (two step)



(c) final rectified image (one step)