1.**Fundamental Matrix Estimation from Point Correspondence**

(a)

(1) For 46 pairs of point correspondences (u_i, v_i), (u'_i, v'_i), use them to create a 46×9 matrix M. Each row of M is equal to:

$$[u_iu'_i \quad u_iv'_i \quad u_i \quad v_iu'_i \quad v_iv'_i \quad v_i \quad u'_i \quad v'_i, 1],\ i = 1, 2, \ldots, 46$$

(2) Use SVD to decompose M into three matrixes: U (46×46), D (46×9 diagonal), V (9×9)

(3) Reshape the last row (which yields the smallest residual) of V to a 3×3 matrix, use SVD to decompose it into : U', D', V', force D[3, 3] to be 0, then multiply U', D' and V' to form a rank two fundamental matrix F.

(4) for each point (u, v) on img1, use $[a, b, c]^T = F^T \times [u, v, 1]^T$ to compute the corresponding epipolar line for (u, v) should be drawn on img2 and draw it. Similarly, for each point (u', v') on img2, use $[a', b', c']^T = F \times [u', v', 1]^T$ to compute the corresponding epipolar line for each point (u', v') should be drawn on img1 and draw it.

(b) The only difference between (a) and (b) is that points used to compute the fundamental matrix in (b) should be normalized in advance. To normalize a point (u_i, v_i, 1), by defining $s = \sqrt{2}/\mathrm{STD}(\{u_i, v_i\})$, $m1 = \mathrm{mean}\{u_i\}$, $m2 = \mathrm{mean}\{v_i\}$, for i = 1, 2, …, 46, we can obtain a matrix T:

$$\begin{matrix} s & 0 & -s \times m1 \\ 0 & s & -s \times m2 \\ 0 & 0 & 1 \end{matrix}$$

$T \times [u, v, 1]^T = [u \times s - s \times m1,\ v \times s - s \times m2, 1]^T$, which is a normalized coordinate and the mean squared distance between the origin and all normalized coordinates $[u \times s - s \times m1,\ v \times s - s \times m2]$ is equal to 2.

(c) The average distance between the feature points and their corresponding epipolar lines computed from the linear least-squares 8-point algorithm using the fundamental matrix generated from the last row of V computed from SVD = 12.134833

The average distance between the feature points and their corresponding epipolar lines computed from the normalized 8-point algorithm using the same fundamental matrix as above = 0.892465

The average distance computed from the normalized 8-point algorithm is considerably smaller since the normalized 8-point algorithm eliminates the noises resulted from the linear least-squares algorithm by bounding the value of (u, v) and (u', v') to prevent the products of two value of x or y coordinates from being too sparse.

2. **Homography transform**

(a)

(1) Select several points in the original image and assign their ideal target points after transform. All selected points are shown below.

Predefining dX, dY, offsetX, offsetY, the target point of the leftmost point (x, y) of each line segment should be **(x – offsetX, y + offsetY – (max{y} - y)×dY)**, and the target point of the rightmost point (x', y') of each line segment should be **(x + (x' - x)×dX - offsetX, y + offsetY – (max{y} - y)×dY)**.

(2) Set the number of iterations we want to use to compute the homography matrix. In each iteration, four pairs of correspondences (x, y) and (x', y') will be randomly selected and used to compute the homography matrix. To compute a homography matrix with 4 correspondences, we should create a 8 ×9 matrix M where each two row of M is equal to

$$[x \quad y \quad 1 \quad 0 \quad 0 \quad 0 \quad -x'x \quad -x'y \quad -x']$$
$$[0 \quad 0 \quad 0 \quad x \quad y \quad 1 \quad -y'x \quad -y'y \quad -y']$$

By using SVD to decompose M into U, S, V and reshaping the last row of V (which leads to the minimal $||AH||^2$) to a 3×3 matrix, we can obtain a homography matrix H. Then, we need to compute the distance between the original points and the points transformed with H in each iteration and choose the H leading to the least distance among all homography matrix.

Note that the source points in this process are the target points we assign in (a) and the target points are actually the original points in (a). Thus, the homography matrix computed in this process

is a backward homography matrix.

(b) After getting the backward homography matrix, we should use backward warping and bilinear interpolation to fetch the value of pixel for each point in the transformed image from the original image.

For each points (i, j) on the transformed image, let (x, y, 1) = H×(u, v, 1) / H×(u, v, 1)[3], (x1, y1) = (floor(x), floor(y)), (x2, y2) = (ceil(x), ceil(y)). Define P(x, y) = the value of pixel of point (x, y) on the original image. The ideal value of points (i, j) on the transformed image should be:

(x2-x)×(y2-y)×P(x1,y1) + (x-x1)×(y2-y)×P(x2,y1) + (y-y1)×(x2-x)×P(x1,y2) + (x-x1)×(y-y1)×P(x2,y2)

If x or y is out of range of image, use 0 to fill the corresponding points.