3D Reconstruction

camera model

- camera matrix

  K - intrinsic

  R - rotation

  t - transformer

  P - camera matrix

  $$Z^c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K[R|t] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

  $P = K[R|t]$

- camera pose -> extrinsic

  C - camera location

  R_C - pose rotation

  $R = R_c^T$

  $t = -RC$

  more detail extrinsic

Essential & Fundamantal Matrix

- Essential Matrix 

  $R = R_c, t = C$ in this case is camera pose.

  $$E = [t\times]R = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} R$$

  $F = K'^{-T} E K^{-1}$

  $l' = Ex, l = E^T x'$

  $x'^T E x = 0$

  $e'^T E = 0, Ee = 0$

- Fundamental matrix

  8 点法计算Fundamental matrix, Given p' p

  $A = [x'x, x'y, x', y'x, y'y, y', x, y, 1]_{N\times 9}$

  $U_{N\times N}, S, V_{9\times 9} = svd(A)$

  $F1 = V[-1].reshape(3,3)$ 最小二乘结果

  $U_{3*3}, S_3, V_{3\times 3} = svd(F1)$

  $S[-1] = 0$ 因为F矩阵秩为2

$$F = U \begin{bmatrix} S[0] & 0 & 0 \\ 0 & S[1] & 0 \\ 0 & 0 & 0 \end{bmatrix} V$$

$$E = U \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} V$$

Camera Matrix P from Essential E

- create 4 possible camera matrices

  $U, S, V = svd(E)$

  $W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

  假设$P_1 = [I|0]$，P_2四种可能参数

  $P_2 = [UWV^T|U[:,2]]P_2 = [UWV^T|-U[:,2]]P_2 = [UW^TV^T|U[:,2]]P_2 = [UW^TV^T|-U[:,2]]$

  more detail in Hartley p 258

  Note: $P_2 = [R_c|C]$是camera pose, 转成extrinsic借助公式camera pose->extrinsic

- Find the correct camera parameters

  4个camera parameter对应四种投影情况，找到能够使得点x和x'的$Z_c > 0$的camera parameter.

  $$\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} \times P_1 \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} \times P_2 \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = 0$$

```
def reconstruct_one_point(pt1, pt2, m1, m2):
  """
      pt1 and m1 * X are parallel and cross product = 0
      pt1 x m1 * X  =  pt2 x m2 * X  =  0
  """
  A = np.vstack([
      np.dot(skew(pt1), m1),
      np.dot(skew(pt2), m2)
  ])
  U, S, V = np.linalg.svd(A)
  P = np.ravel(V[-1, :4])
  return P / P[3] # P = [X,Y,Z,1]
#
for i, P2 in enumerate(P2s):
  # Find the correct camera parameters
  d1 = structure.reconstruct_one_point(points1n[:, 0], points2n[:, 0], P1, P2)
  P2_homogenous = extrinsic_from_camera_pose(P2)
  d2 = np.dot(P2_homogenous[:3, :4], d1) # d2 = [R|t]d1 is [X_c, Y_c, Z_c]

  if d1[2] > 0 and d2[2] > 0:
      ind = i
```

或者可以直接从基础矩阵F推出P

$P = [I|0]$

$P' = [[e'_\times]F|e']$

Triangulation

- find the 3D point X

  Linear triangulation (Hartley ch 12.2 pg 312) to find the 3D point X.

  $x = PX => x \times PX = 0$

  $$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} -p_1^T- \\ -p_2^T- \\ -p_3^T- \end{bmatrix} \begin{bmatrix} | \\ X \\ | \end{bmatrix} = \begin{bmatrix} p_1^T X \\ p_2^T X \\ p_3^T X \end{bmatrix}$$

  $P_1 = [I|0], P_2 = [R_c|C]$

  $$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \times \begin{bmatrix} p_1^T X \\ p_2^T X \\ p_3^T X \end{bmatrix} = \begin{bmatrix} yp_3^T X - p_2^T X \\ p_1^T X - xp_3^T X \\ xp_2^T X - yp_1^T X \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

  $$\begin{bmatrix} yp_3^T - p_2^T \\ p_1^T - xp_3^T \\ y'p'^T_3 - p'^T_2 \\ p'^T_1 - x'p'^T_3 \end{bmatrix} X = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} => AX = 0$$

  to solve $AX = 0$ using SVD

  $X = V[-1, :]/V[-1, 3]$