

# Multiple-View Methods

CS6240 Multimedia Analysis

Leow Wee Kheng

Department of Computer Science  
School of Computing  
National University of Singapore

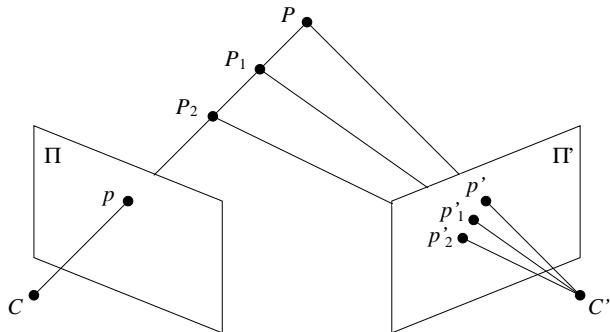


# Outline

- 1 Introduction
- 2 Absolute Orientation
- 3 Camera Orientation
- 4 Triangulation
- 5 Epipolar Geometry
- 6 Estimation of Essential Matrix
- 7 Two-frame Structure From Motion
- 8 Summary
- 9 Exercise
- 10 Further Reading
- 11 Appendix
- 12 References

# Introduction

Multiple 3D scene points can project onto to the same 2D image point.



With two or more views, actual 3D scene point can be recovered.

Perspective camera projection is given by the equation:

$$\rho \tilde{\mathbf{x}} = \mathbf{K}(\mathbf{R}\mathbf{X} + \mathbf{T}) \quad (1)$$

- $\mathbf{X} = [X, Y, Z]^T$ : coordinates of 3D point  $P$  in **world coordinate frame**.
- $\tilde{\mathbf{x}} = [x, y, 1]^T$ : homogeneous coordinates of 2-D image point of  $P$ .
- $\rho$ : parameter related to  $Z$ .
- $\mathbf{K}$ : camera's intrinsic parameter matrix,

$$\mathbf{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \quad (2)$$

- $\mathbf{R}$ ,  $\mathbf{T}$ : rotation and translation matrices that describe the world coordinate frame in camera coordinate frame.

Let  $\mathbf{X}'$  denote the coordinates of  $P$  in **camera coordinate frame**.

Then,

$$\mathbf{X}' = \mathbf{R}\mathbf{X} + \mathbf{T}. \quad (3)$$

From the view point of the world frame,

$$\mathbf{X} = \mathbf{R}^{-1}\mathbf{X}' - \mathbf{R}^{-1}\mathbf{T}. \quad (4)$$

- $\mathbf{R}^{-1}$ : rotation matrix describing camera frame in world frame, i.e., orientation of camera in world frame.
- $\mathbf{C} \equiv -\mathbf{R}^{-1}\mathbf{T}$ : position of camera center ( $\mathbf{X}' = \mathbf{0}$ ) in world frame.

Then,

$$\rho \tilde{\mathbf{x}} = \mathbf{K}(\mathbf{R}\mathbf{X} - \mathbf{R}\mathbf{C}) = \mathbf{K}\mathbf{R}(\mathbf{X} - \mathbf{C}). \quad (5)$$

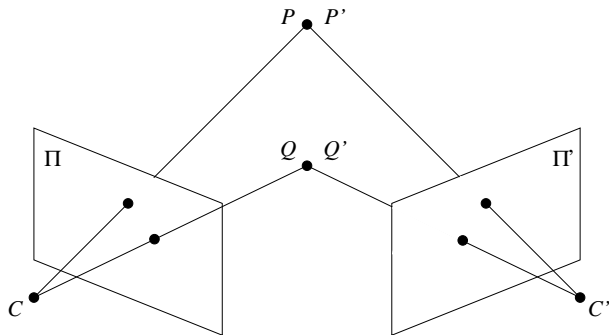
## Related Problems

- **Homography:** (CS4243, CS5240, Appendix C)  
Recover projective transformation between two image frames from corresponding image points  $\mathbf{x}_{ki}$ .
- **Absolute Orientation:**  
Recover rigid transformation  $\mathbf{R}, \mathbf{T}$  between two coordinate systems from corresponding coordinates  $\mathbf{X}_i$  and  $\mathbf{X}'_i$  of 3D points.
- **Camera Orientation:**  
Recover camera's orientation  $\mathbf{R}^{-1}$  and position  $\mathbf{C}$  from corresponding scene points  $\mathbf{X}_i$  and image points  $\mathbf{x}_i$ .
- **Camera Calibration:** (CS4243, Appendix D)  
Recover camera's intrinsic parameters  $\mathbf{K}$  from corresponding scene points  $\mathbf{X}_i$  and image points  $\mathbf{x}_{ki}$  in different views  $k$ .
- **Recovering 3D Structure:**  
Recover 3D scene points  $\mathbf{X}_i$  from corresponding image points  $\mathbf{x}_{ki}$ .

# Absolute Orientation

Recover rigid transformation  $\mathbf{R}$ ,  $\mathbf{T}$  between two coordinate systems.

**Known:** Corresponding coordinates  $\mathbf{X}_i$  and  $\mathbf{X}'_i$  of 3D scene points in the two coordinate systems.



Two forms of rigid transformation:

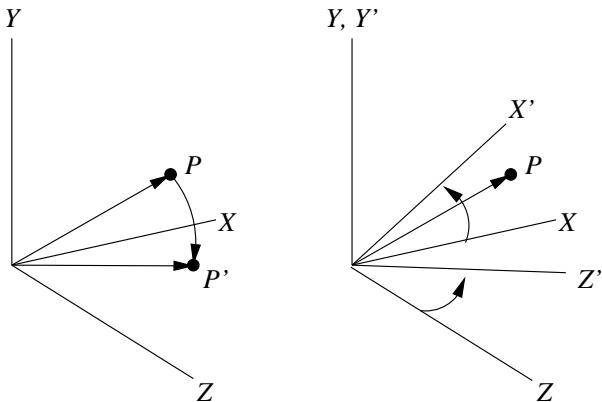
- Transform an object within a coordinate frame.
- Transform a coordinate frame to align with another frame.

Both forms can be described by the same equation:

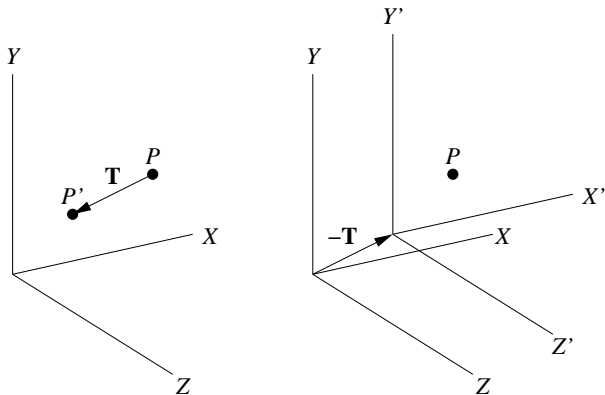
$$\mathbf{X}' = \mathbf{R}\mathbf{X} + \mathbf{T}. \quad (6)$$

The difference is the meaning of  $\mathbf{R}$  and  $\mathbf{T}$ .

Rotating a point by  $\mathbf{R}$  is equivalent to rotating the frame by  $\mathbf{R}^{-1}$ .



Translating a point by  $\mathbf{T}$  is equivalent to translating the frame by  $-\mathbf{T}$ .



# Similarity Transformation

Similarity transformation includes scaling  $s$ :

$$\mathbf{X}' = s\mathbf{R}\mathbf{X} + \mathbf{T}. \quad (7)$$

We look at an algorithm for computing the best fitting  $s$ ,  $\mathbf{R}$  and  $\mathbf{T}$  given  $n$  pairs of corresponding points  $\mathbf{X}_i$  and  $\mathbf{X}'_i$ .

# Computing Similarity Transformation

Algorithm for computing similarity transformation is given in [HHN88].

**Step 1:** Remove translation by moving object's centroid to origin of coordinate system:

$$\mathbf{r}_i = \mathbf{X}_i - \bar{\mathbf{X}}, \quad \mathbf{r}'_i = \mathbf{X}'_i - \bar{\mathbf{X}}' \quad (8)$$

where

$$\bar{\mathbf{X}} = \frac{1}{n} \sum_{i=1}^n \mathbf{X}_i, \quad \bar{\mathbf{X}}' = \frac{1}{n} \sum_{i=1}^n \mathbf{X}'_i. \quad (9)$$

Now,  $\mathbf{r}_i$  and  $\mathbf{r}'_i$  are bundles of vectors.

**Step 2:** Determine scaling factor by comparing mean vector length:

$$s^2 = \frac{\sum_{i=1}^n \|\mathbf{r}'_i\|^2}{\sum_{i=1}^n \|\mathbf{r}_i\|^2} \quad (10)$$

For rigid transformation,  $s = 1$ , so this step can be skipped.

**Step 3:** Compute rotation matrix as follows:

Form matrix  $\mathbf{M}$  from sum of outer product:

$$\mathbf{M} = \sum_{i=1}^n \mathbf{r}'_i \mathbf{r}_i^\top. \quad (11)$$

The rotation matrix  $\mathbf{R}$  is given by

$$\mathbf{R} = \mathbf{M}\mathbf{Q}^{-1/2} \quad (12)$$

where  $\mathbf{Q} = \mathbf{M}^\top \mathbf{M}$ .

Perform **eigen decomposition** of  $\mathbf{Q}$  to obtain

$$\mathbf{Q} = \lambda_1 \mathbf{v}_1 \mathbf{v}_1^\top + \lambda_2 \mathbf{v}_2 \mathbf{v}_2^\top + \lambda_3 \mathbf{v}_3 \mathbf{v}_3^\top \quad (13)$$

where  $\mathbf{v}_i$  and  $\lambda_i$  are the eigenvectors and eigenvalues.

The inverse square root of  $\mathbf{Q}$  can be easily computed as

$$\mathbf{Q}^{-1/2} = \frac{1}{\sqrt{\lambda_1}} \mathbf{v}_1 \mathbf{v}_1^\top + \frac{1}{\sqrt{\lambda_2}} \mathbf{v}_2 \mathbf{v}_2^\top + \frac{1}{\sqrt{\lambda_3}} \mathbf{v}_3 \mathbf{v}_3^\top. \quad (14)$$

Eq. 13 can also be written as

$$\mathbf{Q} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top \quad (15)$$

where

$$\mathbf{V} = [\mathbf{v}_1 \ \mathbf{v}_2 \ \mathbf{v}_3], \quad \mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \lambda_3). \quad (16)$$

Then,

$$\mathbf{Q}^{-1/2} = \mathbf{V} \text{diag} \left( \frac{1}{\sqrt{\lambda_1}}, \frac{1}{\sqrt{\lambda_2}}, \frac{1}{\sqrt{\lambda_3}} \right) \mathbf{V}^\top \quad (17)$$

**Step 4:** Given  $s$  and  $\mathbf{R}$ , we can now compute  $\mathbf{T}$ .

$$\mathbf{T} = \overline{\mathbf{X}}' - s \mathbf{R} \overline{\mathbf{X}}. \quad (18)$$

The  $s$ ,  $\mathbf{R}$ , and  $\mathbf{T}$  obtained are the best-fit solutions.

# Eigen Decomposition

Eigenvalue equation has this form

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v} \quad (19)$$

- $\mathbf{A}$  is a matrix.
- $\mathbf{v}$  is a column vector called **eigenvector**.
- $\lambda$  is a scalar called **eigenvalue** of  $\mathbf{A}$  corresponding to  $\mathbf{v}$ .
- If  $\mathbf{A}$  is  $n \times n$  matrix, then there are  $n$  eigenvectors and eigenvalues.
- Solving for the  $n$   $\mathbf{v}_i$  and  $\lambda_i$  is called **eigen decomposition**.
- There are standard algorithms for eigen decomposition.

## Useful Properties

The eigenvectors  $\mathbf{v}_i$  are orthonormal vectors

$$\mathbf{v}_i^\top \mathbf{v}_j = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \quad (20)$$

Trace of  $\mathbf{A}$  is

$$\text{tr}(\mathbf{A}) = \sum_{i=1}^n \lambda_i. \quad (21)$$

Determinant of  $\mathbf{A}$  is

$$\det(\mathbf{A}) = \prod_{i=1}^n \lambda_i. \quad (22)$$

Eigenvalues of  $\mathbf{A}^k$  are  $\lambda_1^k, \dots, \lambda_n^k$ .

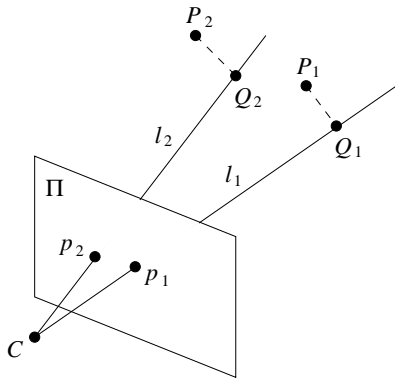
◀ return

# Camera Orientation

Recover camera's orientation  $\mathbf{R}^{-1}$  and position  $\mathbf{C}$ .

**Known:** Camera parameters  $\mathbf{K}$ , 3D scene points  $\mathbf{X}_i$  and corresponding 2D image points  $\mathbf{x}_i$ .

Need only single view.



## Main Ideas [LCH08]

- Compute the projection line  $l_i$  through image point.
- Ideally, scene point  $\mathbf{X}_i$  should lie on projection line  $l_i$ .
- In practice, there is an error:  $\mathbf{X}_i$  is a distance from  $l_i$ .
- So, find the rotation  $\mathbf{R}$  and camera center  $\mathbf{C}$  that minimize the error.
- Apply **iterative algorithm** to iteratively refine  $\mathbf{R}$  and  $\mathbf{C}$ .

Start with camera projection equation:

$$\rho_i \tilde{\mathbf{x}}_i = \mathbf{K}(\mathbf{R}\mathbf{X}_i - \mathbf{R}\mathbf{C}) \quad (23)$$

where  $\tilde{\mathbf{x}}_i = [x_i, y_i, 1, ]^\top$ .

# Camera Orientation Algorithm

Initialize  $\mathbf{R} = \mathbf{I}$  and  $\mathbf{C} = \mathbf{0}$ .

Repeat until convergence:

- (1) Compute 3D points in camera coordinate frame:

$$\mathbf{X}'_i = \mathbf{R} (\mathbf{X}_i - \mathbf{C}). \quad (24)$$

- (2) Compute projection lines:

$$\mathbf{l}_i = \mathbf{K}^{-1} \tilde{\mathbf{x}}_i. \quad (25)$$

- (3) Compute ideal position  $\mathbf{Q}_i$  of  $\mathbf{X}'_i$ :

Ideal position  $\mathbf{Q}_i = \rho_i \mathbf{l}_i$  lies on  $\mathbf{l}_i$ , i.e.,  $\mathbf{Q}_i = \rho_i \mathbf{l}_i$  for some  $\rho_i$ .

$\rho_i$  can be obtained by substituting Eq. 24 and 25 into Eq. 23:

$$\rho_i \mathbf{l}_i = \mathbf{X}'_i, \quad \rho_i = \frac{\mathbf{l}_i^\top \mathbf{X}'_i}{\mathbf{l}_i^\top \mathbf{l}_i}. \quad (26)$$

- $\rho_i \mathbf{l}_i$  is the projection of  $\mathbf{X}'_i$  on  $\mathbf{l}_i$ . Eq. 26 is the best-fit solution.

Mean distance of  $\mathbf{X}'_i$  to  $\mathbf{l}_i$  is:

$$E_Q = \frac{1}{n} \sum_{i=1}^n \|\mathbf{Q}_i - \mathbf{X}'_i\|. \quad (27)$$

- (4) Compute best-fitting rotation matrix  $\mathbf{R}$  that maps  $\mathbf{X}_i$  to  $\mathbf{Q}_i$  using **absolute orientation** algorithm.
- (5) Compute new camera center  $\mathbf{C}$  by substituting the means of the point sets into Eq. 24:

$$\mathbf{C} = \bar{\mathbf{X}} - \mathbf{R}^{-1}\bar{\mathbf{Q}} \quad (28)$$

where  $\bar{\mathbf{X}}$  and  $\bar{\mathbf{Q}}$  are the means of  $\mathbf{X}_i$  and  $\mathbf{Q}_i$ .

Note:

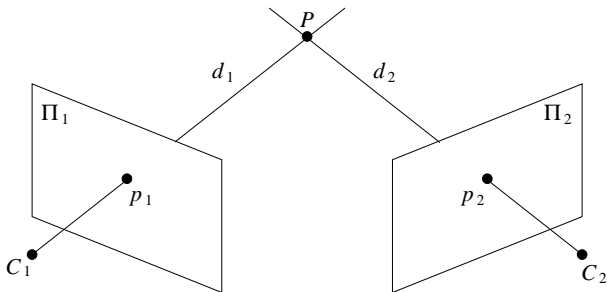
- This algorithm iteratively moves  $\mathbf{X}'_i$  onto  $\mathbf{l}_i$ , thereby minimizing the error, and yielding the best-fit  $\mathbf{R}$  and  $\mathbf{C}$ .
- Another algorithm is given in [SS01].

# Triangulation

Recover 3D scene points  $\mathbf{X}_i$  by **triangulation**.

**Known:** Camera parameters  $\mathbf{K}_k$ , orientation  $\mathbf{R}_k$  and centers  $\mathbf{C}_k$ , and corresponding image points  $\mathbf{x}_{ki}$ .

**Idea:** Project from image points and compute intersection of projection lines.



Perspective projection equation gives

$$\rho_k \tilde{\mathbf{x}}_k = \mathbf{K}_k(\mathbf{R}_k \mathbf{X} - \mathbf{R}_k \mathbf{C}_k) \quad (29)$$

where  $\tilde{\mathbf{x}}_k = [x_k, y_k, 1]^\top$ ,  $k = 1, \dots, m$ , for  $m$  camera views. So,

$$\rho_k \mathbf{R}_k^{-1} \mathbf{K}_k^{-1} \tilde{\mathbf{x}}_k = \mathbf{X} - \mathbf{C}_k. \quad (30)$$

- $\mathbf{R}_k^{-1} \mathbf{K}_k^{-1} \tilde{\mathbf{x}}_k$  is the projection line.

Let

$$\mathbf{v}_k = \frac{\mathbf{R}_k^{-1} \mathbf{K}_k^{-1} \tilde{\mathbf{x}}_k}{\|\mathbf{R}_k^{-1} \mathbf{K}_k^{-1} \tilde{\mathbf{x}}_k\|}. \quad (31)$$

Then,  $\mathbf{v}_k$  is a **unit vector** along the projection line.

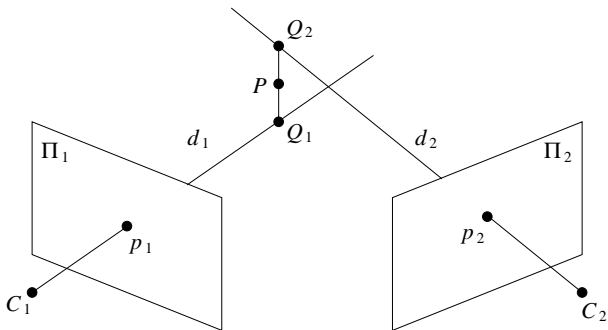
Let  $d_k$  denote the distance of point  $P$  from camera center  $C_k$  along the projection line. Then,

$$d_k \mathbf{v}_k = \mathbf{X} - \mathbf{C}_k \quad (32)$$

That is,

$$d_k = \mathbf{v}_k \cdot (\mathbf{X} - \mathbf{C}_k). \quad (33)$$

In practice, can't find exact intersection point due to numerical error and noise.



So, compute **closest point** to the lines,  
i.e., mid-point of the shortest line between projection lines.

Shortest line is perpendicular to projection lines.

Projection line passes through  $Q_k$  instead of  $P$ .

$$\begin{aligned}
 \mathbf{Q}_k &= \mathbf{C}_k + d_k \mathbf{v}_k \\
 &= \mathbf{C}_k + \mathbf{v}_k (\mathbf{v}_k \cdot (\mathbf{X} - \mathbf{C}_k)) \\
 &= \mathbf{C}_k + (\mathbf{v}_k \mathbf{v}_k^\top) (\mathbf{X} - \mathbf{C}_k).
 \end{aligned} \tag{34}$$

Distance  $e_k$  between  $P$  and  $Q_k$  is

$$\begin{aligned}
 e_k &= \|\mathbf{X} - \mathbf{C}_k - (\mathbf{v}_k \mathbf{v}_k^\top) (\mathbf{X} - \mathbf{C}_k)\| \\
 &= \|(\mathbf{I} - \mathbf{v}_k \mathbf{v}_k^\top) (\mathbf{X} - \mathbf{C}_k)\|.
 \end{aligned} \tag{35}$$

Then, the optimal  $\mathbf{X}$  is the one that minimizes the sum-squared error

$$E = \sum_k e_k^2 = \sum_k \|(\mathbf{I} - \mathbf{v}_k \mathbf{v}_k^\top) (\mathbf{X} - \mathbf{C}_k)\|^2. \tag{36}$$

So, find solution of  $\partial E / \partial \mathbf{X} = 0$ .

$$\frac{\partial E}{\partial \mathbf{X}} = 2 \sum_k (\mathbf{I} - \mathbf{v}_k \mathbf{v}_k^\top)^\top (\mathbf{I} - \mathbf{v}_k \mathbf{v}_k^\top) (\mathbf{X} - \mathbf{C}_k) = 0. \quad (37)$$

That is,

$$\sum_k \mathbf{M}_k^\top \mathbf{M}_k \mathbf{X} = \sum_k \mathbf{M}_k^\top \mathbf{M}_k \mathbf{C}_k \quad (38)$$

where

$$\mathbf{M}_k = \mathbf{I} - \mathbf{v}_k \mathbf{v}_k^\top. \quad (39)$$

Then,

$$\mathbf{X} = \left[ \sum_k \mathbf{M}_k^\top \mathbf{M}_k \right]^{-1} \sum_k \mathbf{M}_k^\top \mathbf{M}_k \mathbf{C}_k. \quad (40)$$



- Lines  $l_1, l_2$  are **epipolar lines**.
- All scene points on the epipolar plane project onto the corresponding epipolar lines.

Without loss of generality,

- Set world coordinate frame at  $C_1$  and aligned with camera frame 1, i.e.,  $\mathbf{C}_1 = \mathbf{0}$ ,  $\mathbf{R}_1 = \mathbf{I}$ .
- Camera frame 1 is related to camera frame 2 by  $\mathbf{R}$  and  $\mathbf{T}$ .

# Calibrated Cameras

Let's consider the case where camera parameters  $\mathbf{K}_k$  are known.

Then, given image points  $\tilde{\mathbf{x}}_k$ , can obtain normalized image points  $\bar{\mathbf{x}}_k$  by

$$\bar{\mathbf{x}}_k = \mathbf{K}_k^{-1} \tilde{\mathbf{x}}_k. \quad (41)$$

So, can regard  $\Pi_1$  and  $\Pi_2$  as normalized image planes (Appendix B).

Let  $\mathbf{X}_1$  and  $\mathbf{X}_2$  denote the coordinate vectors of  $P$  in camera frame 1 and camera frame 2. Then,

$$\rho_2 \bar{\mathbf{x}}_2 = \mathbf{X}_2 = \mathbf{R} \mathbf{X}_1 + \mathbf{T} = \mathbf{R}(\rho_1 \bar{\mathbf{x}}_1) + \mathbf{T}. \quad (42)$$

Multiply both side by cross product with  $\mathbf{T}$  gives

$$\rho_2 \mathbf{T} \times \bar{\mathbf{x}}_2 = \rho_1 \mathbf{T} \times \mathbf{R} \bar{\mathbf{x}}_1 \quad (43)$$

since  $\mathbf{T} \times \mathbf{T} = \mathbf{0}$  for any vector  $\mathbf{T}$ .

Writing vector cross product in matrix multiplication form (Exercise):

$$\rho_2 [\mathbf{T}]_{\times} \bar{\mathbf{x}}_2 = \rho_1 [\mathbf{T}]_{\times} \mathbf{R} \bar{\mathbf{x}}_1 \quad (44)$$

where

$$[\mathbf{T}]_{\times} = \begin{bmatrix} 0 & -T_Z & T_Y \\ T_Z & 0 & -T_X \\ -T_Y & T_X & 0 \end{bmatrix}. \quad (45)$$

Taking dot product with  $\bar{\mathbf{x}}_2$  yields

$$\rho_1 \bar{\mathbf{x}}_2^{\top} [\mathbf{T}]_{\times} \mathbf{R} \bar{\mathbf{x}}_1 = \rho_2 \bar{\mathbf{x}}_2^{\top} [\mathbf{T}]_{\times} \bar{\mathbf{x}}_2 = 0. \quad (46)$$

Exercise: Show that  $\mathbf{x}^{\top} [\mathbf{T}]_{\times} \mathbf{x} = 0$  for any  $\mathbf{x}$ .

Denoting  $[\mathbf{T}]_{\times} \mathbf{R}$  by  $\mathbf{E}$  yields the **epipolar constraint**:

$$\bar{\mathbf{x}}_2^{\top} \mathbf{E} \bar{\mathbf{x}}_1 = 0. \quad (47)$$

$\mathbf{E}$  is called the **essential matrix**.

- $\bar{\mathbf{x}}_2$  is on the epipolar line  $\mathbf{l}_2$ . So,  $\bar{\mathbf{x}}_2^{\top} \mathbf{l}_2 = 0$  (see Eq. 68).
- So,  $\mathbf{E} \bar{\mathbf{x}}_1 = \mathbf{l}_2$ .
- Essential matrix  $\mathbf{E}$  maps a point  $\bar{\mathbf{x}}_1$  in image 1 into epipolar line  $\mathbf{l}_2$  in image 2.
- The converse is also true:  $\mathbf{E}^{\top} \bar{\mathbf{x}}_2 = \mathbf{l}_1$ .

For any scalar  $s$ ,

$$\bar{\mathbf{x}}_2^{\top} (s\mathbf{E}) \bar{\mathbf{x}}_1 = 0. \quad (48)$$

That is,  $s\mathbf{E}$  is also a valid essential matrix describing the two points.

So, essential matrix is defined only **up to scale**.

The actual scale is unknown.

# Uncalibrated Cameras

When the camera parameters  $\mathbf{K}_k$  are unknown, can't convert to normalized image points.

From Eq. 47, we have

$$\begin{aligned}\bar{\mathbf{x}}_2^\top \mathbf{E} \bar{\mathbf{x}}_1 &= 0 \\ \tilde{\mathbf{x}}_2^\top \mathbf{K}_2^{-\top} \mathbf{E} \mathbf{K}_1^{-1} \tilde{\mathbf{x}}_1 &= 0\end{aligned}\tag{49}$$

where  $\mathbf{K}^{-\top} = (\mathbf{K}^{-1})^\top$ .

Denoting  $\mathbf{K}_2^{-\top} \mathbf{E} \mathbf{K}_1^{-1}$  by  $\mathbf{F}$  yields the constraint:

$$\tilde{\mathbf{x}}_2^\top \mathbf{F} \tilde{\mathbf{x}}_1 = 0.\tag{50}$$

$\mathbf{F}$  is called the **fundamental matrix**.

Structure from motion in this case is much more complex.

# Estimation of Essential Matrix

**Known:** Camera parameters  $\mathbf{K}_1$ ,  $\mathbf{K}_2$  and  $n$  pairs of corresponding image points  $\mathbf{x}_{1i}$  and  $\mathbf{x}_{2i}$ ,  $i = 1, \dots, n$ .

From image points  $\mathbf{x}_{ki}$ , form homogeneous coordinates

$$\tilde{\mathbf{x}}_{ki} = [x_{ki}, y_{ki}, 1]^\top, \quad k = 1, 2.$$

Then, the normalized image points can be computed as  $\bar{\mathbf{x}}_{ki} = \mathbf{K}_k^{-1} \tilde{\mathbf{x}}_{ki}$ .

Let  $\mathbf{E}$  be

$$\mathbf{E} = \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix}. \quad (51)$$

Expanding Eq. 47 gives

$$\begin{aligned}
 \bar{x}_{1i} \bar{x}_{2i} e_{11} &+ \bar{y}_{1i} \bar{x}_{2i} e_{12} + \bar{x}_{2i} e_{13} + \\
 \bar{x}_{1i} \bar{y}_{2i} e_{21} &+ \bar{y}_{1i} \bar{y}_{2i} e_{22} + \bar{y}_{2i} e_{23} + \\
 \bar{x}_{1i} e_{31} &+ \bar{y}_{1i} e_{32} + e_{33} = 0
 \end{aligned} \tag{52}$$

where  $\bar{x}_{ki}$  and  $\bar{y}_{ki}$  are the components of  $\bar{\mathbf{x}}_{ki}$ .

With  $n$  points, can form a system of linear equations (Exercise):

$$\mathbf{A} \mathbf{e} = 0. \tag{53}$$

- $\mathbf{A}$  is a matrix that contains the products of  $\mathbf{x}_{1i}$  and  $\mathbf{x}_{2i}$ .
- $\mathbf{e}$  is a vector that contains the  $e_{ij}$  parameters.

So, can solve for  $\mathbf{e}$  using linear least square method.

Need at least 8 corresponding points, called **8-Point Algorithm**.

Why 8?

# Improved 8-Point Algorithm

Improve accuracy of 8-point algorithm [Har97].

All scaled versions of  $\mathbf{A}$  can satisfy Eq. 53.

To avoid trivial solution of  $\mathbf{e} = \mathbf{0}$ , add a constraint:

**Option 1:**  $e_{33} = 1$ . Solve in the same way as before.

**Option 2:** Require  $\|\mathbf{e}\| = 1$ .

With noise and measurement error,  $\mathbf{A}\mathbf{e}$  cannot be exactly  $\mathbf{0}$ .

So, find  $\mathbf{e}$  that minimizes  $\|\mathbf{A}\mathbf{e}\|^2$  subject to the constraint  $\|\mathbf{e}\|^2 = 1$ .

This is equivalent to minimizing

$$C = \|\mathbf{A}\mathbf{e}\|^2 + \lambda(1 - \|\mathbf{e}\|^2) \quad (54)$$

This is called **Lagrange multiplier method**,  $\lambda$  is the Lagrange multiplier.

To minimize  $C$ , find solution of  $\partial C / \partial \mathbf{e} = \mathbf{0}$ .

$$\frac{\partial C}{\partial \mathbf{e}} = 2\mathbf{A}^\top \mathbf{A} \mathbf{e} - 2\lambda \mathbf{e} = 0. \quad (55)$$

That is

$$\mathbf{A}^\top \mathbf{A} \mathbf{e} = \lambda \mathbf{e}. \quad (56)$$

The eigenvector  $\mathbf{e}$  of  $\mathbf{A}^\top \mathbf{A}$  with the smallest eigenvalue  $\lambda$  minimizes  $C$  (Eq. 54).

## Other Properties of $\mathbf{E}$

- $\mathbf{E}$  is singular, i.e.,  $\mathbf{E}$  has no inverse, it's determinant is 0.
- Of the 3 singular values of  $\mathbf{E}$ , the smallest is 0.

Methods of estimating  $\mathbf{E}$  discussed don't guarantee that  $\mathbf{E}$  is singular.

To enforce singularity constraint, replace  $\mathbf{E}$  by singular  $\mathbf{E}'$  such that  $\|\mathbf{E} - \mathbf{E}'\|_F$  is minimized.

- For a  $m \times n$  matrix  $\mathbf{M}$ , the **Frobenius norm**  $\|\mathbf{M}\|_F$  is

$$\|\mathbf{M}\|_F^2 = \sum_{i=1}^m \sum_{j=1}^n |m_{ij}|^2. \quad (57)$$

The minimization method required is **singular value decomposition** (SVD).

Apply SVD on  $\mathbf{E}$  to obtain

$$\mathbf{E} = \mathbf{U} \operatorname{diag}(s_1, s_2, s_3) \mathbf{V}^\top \quad (58)$$

with  $s_1 \geq s_2 \geq s_3$ .

Then, required  $\mathbf{E}'$  is given by

$$\mathbf{E}' = \mathbf{U} \operatorname{diag}(s_1, s_2, 0) \mathbf{V}^\top. \quad (59)$$

## Summary of Improved 8-Point Algorithm

- (1) Given image points  $\mathbf{x}_{ki}$ ,  $k = 1, 2$ , form homogeneous coordinates  $\tilde{\mathbf{x}}_{ki}$ .
- (2) Compute normalized image points  $\bar{\mathbf{x}}_{ki} = \mathbf{K}_k^{-1} \tilde{\mathbf{x}}_{ki}$ .
- (3) Form matrix  $\mathbf{A}$  as in Eq. 53.
- (4) Find eigenvector  $\mathbf{e}$  of  $\mathbf{A}^\top \mathbf{A}$  with the smallest eigenvalue.
- (5) Reassemble  $\mathbf{e}$  into  $\mathbf{E}$  according to Eq. 53.
- (6) Perform SVD on  $\mathbf{E}$  and obtain singular  $\mathbf{E}'$  that minimizes  $\|\mathbf{E} - \mathbf{E}'\|_F$ .

The final estimated essential matrix is  $\mathbf{E}'$ .

Fundamental matrix  $\mathbf{F}$  can be estimated in the same way.

- Skip Steps 1 and 2.
- Form matrix  $\mathbf{A}$  from image points  $\mathbf{x}_{ki}$ .

# Singular Value Decomposition

The singular value decomposition of a  $m \times n$  real matrix  $\mathbf{M}$  is

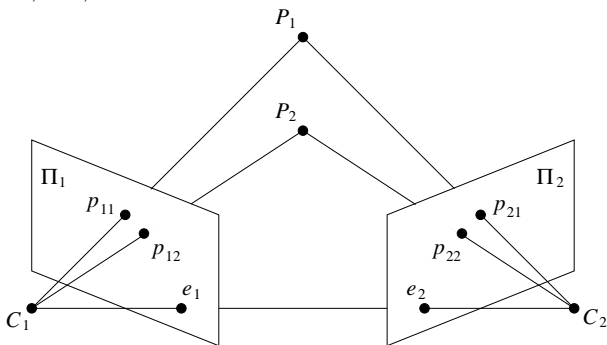
$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\top} \quad (60)$$

- $\mathbf{U} = [\mathbf{u}_1 \cdots \mathbf{u}_m]$  is a  $m \times m$  matrix.
- $\mathbf{u}_i$ 's are **left singular vectors** and are orthonormal
- $\mathbf{V} = [\mathbf{v}_1 \cdots \mathbf{v}_n]$  is a  $n \times n$  matrix.
- $\mathbf{v}_i$ 's are **right singular vectors** and are orthonormal.
- $\mathbf{\Sigma}$  is a diagonal matrix.
- Diagonal values of  $\mathbf{\Sigma}$  are **singular values** sorted in decreasing order.
- A  $m \times n$  matrix  $\mathbf{M}$  has at least 1 and at most  $p = \min(m, n)$  distinct singular values.
- There are standard algorithms for SVD.

# Two-frame Structure From Motion

Recover scene points  $\mathbf{X}_i$  from image points  $\mathbf{x}_{ki}$  in two views.

**Known:** Camera parameters  $\mathbf{K}_k$ , corresponding image points  $\mathbf{x}_{ki}$ ,  
 $k = 1, 2, i = 1, \dots, n$ .



Without loss of generality,

- Set world coordinate frame at  $C_1$  and aligned with camera frame 1, i.e.,  $\mathbf{C}_1 = \mathbf{0}$ ,  $\mathbf{R}_1 = \mathbf{I}$ .
- Camera frame 1 is related to camera frame 2 by  $\mathbf{R}$  and  $\mathbf{T}$ .

Basic steps in recovering 3D scene points:

- Estimate essential matrix  $\mathbf{E}$  from corresponding image points  $\mathbf{x}_{ki}$ .
- Estimate rotation  $\mathbf{R}$  and translation  $\mathbf{T}$  from  $\mathbf{E}$ .
- Recover 3D scene points  $\mathbf{X}$  using triangulation or other methods.

# Estimation of Rotation and Translation

Essential matrix  $\mathbf{E}$  is singular. Perform SVD on  $\mathbf{E}$ :

$$\mathbf{E} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \mathbf{v}_3^T \end{bmatrix}. \quad (61)$$

- In noise-free case, smallest singular value is 0. Otherwise, non-zero.
- In general, the other two singular values are not 1 because  $\mathbf{E}$  is computed up to unknown scale.
- $\mathbf{u}_3$  gives the direction of translation  $\mathbf{t} = \mathbf{T}/\|\mathbf{T}\|$ .
- Absolute distance  $\|\mathbf{T}\|$  cannot be estimated because  $\mathbf{E}$  is computed up to unknown scale.

So, can only recover  $\mathbf{E} = [\mathbf{t}]_{\times}\mathbf{R}$  instead of  $[\mathbf{T}]_{\times}\mathbf{R}$ .

The vectors  $\mathbf{u}_1$ ,  $\mathbf{u}_2$ ,  $\mathbf{u}_3 = \mathbf{t}$  are orthonormal, and  $\mathbf{t} = \mathbf{u}_1 \times \mathbf{u}_2$ .

The cross product operator  $[\mathbf{t}]_{\times}$  can be expanded as (Exercise):

$$\begin{aligned}
 [\mathbf{t}]_{\times} &= [\mathbf{u}_1 \quad \mathbf{u}_2 \quad \mathbf{t}] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{u}_1^{\top} \\ \mathbf{u}_2^{\top} \\ \mathbf{t}^{\top} \end{bmatrix} \\
 &= \mathbf{U}\mathbf{\Sigma}\mathbf{R}_{90^\circ}\mathbf{U}^{\top}.
 \end{aligned} \tag{62}$$

$[\mathbf{t}]_{\times}$  projects a vector onto orthonormal basis vectors  $(\mathbf{u}_1, \mathbf{u}_2, \mathbf{t})$ , zeros out the  $\mathbf{t}$  component ( $\mathbf{\Sigma}$ ), and rotates  $\mathbf{u}_1, \mathbf{u}_2$  by  $90^\circ$  ( $\mathbf{R}_{90^\circ}$ ).

From Eq. 61 and 62, we get

$$\mathbf{E} = [\mathbf{t}]_{\times}\mathbf{R} = \mathbf{U}\mathbf{\Sigma}\mathbf{R}_{90^\circ}\mathbf{U}^{\top}\mathbf{R} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\top}. \tag{63}$$

So,

$$\mathbf{R}_{90^\circ} \mathbf{U}^\top \mathbf{R} = \mathbf{V}^\top \quad (64)$$

and

$$\mathbf{R} = \mathbf{U} \mathbf{R}_{90^\circ}^{-1} \mathbf{V}^\top. \quad (65)$$

The correct signs of  $\mathbf{E}$ ,  $\mathbf{t}$ ,  $\mathbf{U}$  and  $\mathbf{V}$  are not known.

So, have to generate all 4 possible matrices

$$\mathbf{R} = \pm \mathbf{U} \mathbf{R}_{\pm 90^\circ}^{-1} \mathbf{V}^\top \quad (66)$$

and keep the two whose determinants  $|\mathbf{R}| = 1$ .

To determine the final rotation matrix,

- pair the two with both signs of translation  $\pm \mathbf{t}$ ,
- select the combination for which the largest number of reconstructed 3D points is seen in front of both cameras.

# Summary

We have learned some useful methods for solving problems:

- Linear least square: triangulation
- Eigen-decomposition: absolute orientation, camera orientation, estimating essential matrix
- Iterative algorithm: camera orientation
- Lagrange multiplier method: estimating essential matrix
- Singular value decomposition: estimating essential matrix
- All the above: two-frame structure from motion

# Exercise

- (1) Let  $\mathbf{a} = [a_1, a_2, a_3]^\top$  and  $\mathbf{b} = [b_1, b_2, b_3]^\top$  be vectors, and

$$[\mathbf{a}]_{\times} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}.$$

Show that cross product  $\mathbf{a} \times \mathbf{b}$  is equal to the matrix product  $[\mathbf{a}]_{\times} \mathbf{b}$ .

- (2) Show that  $\mathbf{x}^\top [\mathbf{T}]_{\times} \mathbf{x} = 0$  for any vector  $\mathbf{x}$ .
- (3) Assemble the epipolar constraint equation (Eq. 47, 52) for  $n$  points into a system of linear equations of the form given in Eq. 53.
- (4) Expand the cross product operator  $[\mathbf{t}]_{\times}$  into Eq. 62.

# Further Reading

- Representation of 2D lines, 3D planes, and 3D lines (Appendix A).
- Camera model:
- Absolute orientation: Jain [JKS95], Section 12.7.
- Camera orientation: [LCH08], Szeliski [Sze10] Section 6.2.
- Triangulation: Szeliski [Sze10], Section 7.1.
- Epipolar geometry: Szeliski [Sze10], Section 7.2.
- Two-frame structure from motion: Szeliski [Sze10], Section 7.2.
- Camera calibration: Zhang [Zha00], Bouquet [Bou]

## A.1 2D Lines

First, let's look at how to represent lines and planes mathematically.

A 2D line  $l$  in  $(x, y)$  coordinate frame is given by the equation

$$ax + by + c = 0. \quad (67)$$

- Let  $\tilde{\mathbf{l}}$  denote the homogeneous coordinate vector  $[a, b, c]^T$ .
- Let  $\tilde{\mathbf{x}}$  denote the homogeneous coordinate vector  $[x, y, 1]^T$ .

Then, the line  $l$  can be represented as

$$\tilde{\mathbf{x}} \cdot \tilde{\mathbf{l}} = 0. \quad (68)$$

- $\tilde{\mathbf{l}}$  is called **line equation vector**.

Using homogeneous coordinates, the intersection  $\tilde{\mathbf{x}}$  of two 2D lines  $\tilde{\mathbf{l}}_1$  and  $\tilde{\mathbf{l}}_2$  is simply their cross product (Exercise):

$$\tilde{\mathbf{x}} = \tilde{\mathbf{l}}_1 \times \tilde{\mathbf{l}}_2. \quad (69)$$

Similarly, the joining of two points  $\tilde{\mathbf{x}}_1$  and  $\tilde{\mathbf{x}}_2$  forms a line  $\tilde{\mathbf{l}}$  given by the cross product (Exercise):

$$\tilde{\mathbf{l}} = \tilde{\mathbf{x}}_1 \times \tilde{\mathbf{x}}_2. \quad (70)$$

## A.2 3D Planes

A 3D plane  $\Pi$  in  $(x, y, z)$  coordinate frame is given by the equation

$$ax + by + cz + d = 0. \quad (71)$$

- Let  $\tilde{\mathbf{m}}$  denote the homogeneous coordinate vector  $[a, b, c, d]^\top$ .
- Let  $\tilde{\mathbf{x}}$  denote the homogeneous coordinate vector  $[x, y, z, 1]^\top$ .

Then, the plane  $\Pi$  can be represented as

$$\tilde{\mathbf{x}} \cdot \tilde{\mathbf{m}} = 0. \quad (72)$$

- $\tilde{\mathbf{m}}$  is called **plane equation vector**.

Let  $\mathbf{x} = [x, y, z]^T$ ,  $\mathbf{a} = [a, b, c]^T$ .

Then, the plane equation becomes

$$\mathbf{a} \cdot \mathbf{x} + d = 0. \quad (73)$$

Suppose  $\mathbf{x}_1$  and  $\mathbf{x}_2$  are two points on the plane. Then,

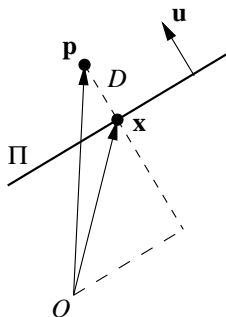
$$\begin{aligned} \mathbf{a} \cdot \mathbf{x}_1 + d &= 0 \\ \mathbf{a} \cdot \mathbf{x}_2 + d &= 0. \end{aligned} \quad (74)$$

That is,

$$\mathbf{a} \cdot (\mathbf{x}_1 - \mathbf{x}_2) = 0. \quad (75)$$

Since the vector  $\mathbf{x}_1 - \mathbf{x}_2$  is on the plane,  $\mathbf{a}$  must be normal to the plane. Then, the plane can also be represented by the unit normal vector  $\mathbf{u} = \mathbf{a}/\|\mathbf{a}\|$  of the plane.

Let  $\mathbf{p}$  be a general 3D point and  $\mathbf{x}$  be its normal projection on  $\Pi$ .



Then, the perpendicular distance  $D$  of  $\mathbf{p}$  from  $\Pi$  is

$$\begin{aligned} D &= \mathbf{p} \cdot \mathbf{u} - \mathbf{x} \cdot \mathbf{u} \\ &= \frac{\mathbf{p} \cdot \mathbf{a} - \mathbf{x} \cdot \mathbf{a}}{\|\mathbf{a}\|} = \frac{\mathbf{a} \cdot \mathbf{p} + d}{\|\mathbf{a}\|}. \end{aligned} \quad (76)$$

Formulae of unit normal vector and perpendicular distance apply to 2D lines also.

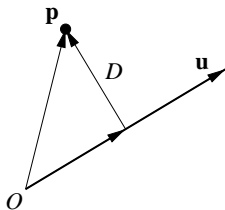
## A.3 3D Lines

There is no elegant representation of lines in 3D.  
One possible representation is

$$\mathbf{p} + \lambda \mathbf{u} \quad (77)$$

where  $\mathbf{p}$  is a point on the line,  $\mathbf{u}$  is the unit direction vector of the line, and  $\lambda$  is a parameter.

A 3D line  $l$  that passes through the origin  $O$  can be represented as  $\lambda \mathbf{u}$ .



Projection of 3D point  $\mathbf{p}$  on the line is  $\mathbf{p} \cdot \mathbf{u}$ .

Then, the normal vector from line  $l$  to any 3D point  $\mathbf{p}$  is

$$\begin{aligned}
 \mathbf{p} - (\mathbf{p} \cdot \mathbf{u})\mathbf{u} &= \mathbf{p} - \mathbf{u}(\mathbf{u} \cdot \mathbf{p}) \\
 &= \mathbf{p} - (\mathbf{u}\mathbf{u}^\top)\mathbf{p} \\
 &= (\mathbf{I} - \mathbf{u}\mathbf{u}^\top)\mathbf{p}.
 \end{aligned} \tag{78}$$

Perpendicular distance from  $\mathbf{p}$  to  $l$  is

$$\|\mathbf{p} - (\mathbf{p} \cdot \mathbf{u})\mathbf{u}\|. \tag{79}$$

## B Normalized Image

Recall perspective projection equation:

$$\rho \tilde{\mathbf{x}} = \mathbf{K}\mathbf{X}. \quad (80)$$

Let's define another perspective project  $\mathbf{K}' = \mathbf{I}$  such that

$$\rho \bar{\mathbf{x}} = \mathbf{K}'\mathbf{X} = \mathbf{I}\mathbf{X} = \mathbf{X}. \quad (81)$$

This is the same as the pinhole camera model. So,

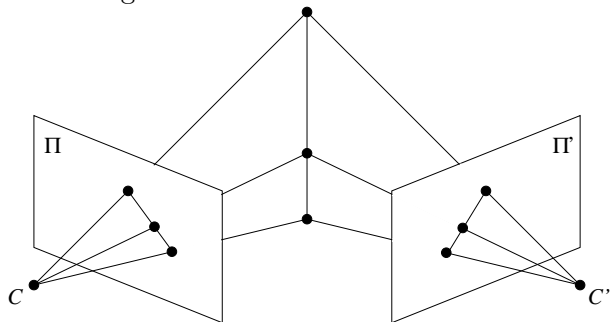
$$\rho \tilde{\mathbf{x}} = \mathbf{K}\mathbf{X} = \mathbf{K}\mathbf{I}\mathbf{X} = \rho\mathbf{K}\bar{\mathbf{x}}, \quad (82)$$

$$\bar{\mathbf{x}} = \mathbf{K}^{-1}\tilde{\mathbf{x}}. \quad (83)$$

- Image plane containing  $\bar{\mathbf{x}}$  is **normalized image plane**.
- Can think of  $\mathbf{X}$  as projecting first to the normalized image plane, then to the real image plane.
- Can derive normalized image from real image if  $\mathbf{K}$  is known.

# C Homography

Homography is a transformation between projective planes that maps straight lines to straight lines.



- Also known as **projective transformation**.
- If camera motion is pure rotation, then the two images are related by a homography.

Let  $\mathbf{x}_i = [x_i, y_i]^\top$  and  $\mathbf{x}'_i = [x'_i, y'_i]^\top$ ,  $i = 1, \dots, n$ , denote the corresponding points in the two images.

Let  $\tilde{\mathbf{x}}_i = [x_i, y_i, 1]^\top$  and  $\tilde{\mathbf{x}}'_i = [w_i x'_i, w_i y'_i, w_i]^\top = [u_i, v_i, w_i]^\top$ , for any  $w_i \neq 0$ , denote their homogeneous coordinates.

Then,

$$\begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} = \tilde{\mathbf{x}}'_i = \mathbf{H} \tilde{\mathbf{x}}_i = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}. \quad (84)$$

Affine transformation is a special case of homography with  $h_{31} = h_{32} = 0$ ,  $h_{33} = 1$ .

Scaling  $\mathbf{H}$  by a constant  $s$  preserves Eq. 84:

$$(s\mathbf{H})\tilde{\mathbf{x}}_i = s\tilde{\mathbf{x}}'_i = \tilde{\mathbf{x}}'_i. \quad (85)$$

So, homography is defined only up to an unspecified scale.

So, can set  $h_{33} = 1$ .

Expanding Eq. 84 with  $h_{33} = 1$  gives

$$h_{11}x_i + h_{12}y_i + h_{13} - h_{31}x_ix'_i - h_{32}y_ix'_i = x'_i, \quad (86)$$

$$h_{21}x_i + h_{22}y_i + h_{23} - h_{31}x_iy'_i - h_{32}y_iy'_i = y'_i. \quad (87)$$

Assembling Eq. 86 and 87 over all points yields

$$\begin{bmatrix}
 x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 \\
 & & & \vdots & & & & \\
 x_n & y_n & 1 & 0 & 0 & 0 & -x_nx'_n & -y_nx'_n \\
 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 \\
 & & & \vdots & & & & \\
 0 & 0 & 0 & x_1 & y_1 & 1 & -x_ny'_n & -y_ny'_n
 \end{bmatrix}
 \begin{bmatrix}
 h_{11} \\
 h_{12} \\
 h_{13} \\
 h_{21} \\
 h_{22} \\
 h_{23} \\
 h_{31} \\
 h_{32}
 \end{bmatrix}
 =
 \begin{bmatrix}
 x'_1 \\
 \vdots \\
 x'_n \\
 y'_1 \\
 \vdots \\
 y'_n
 \end{bmatrix}. \quad (88)$$

This is a system of linear equations which can be solved for least square solution.

## D Camera Calibration

Recover camera's intrinsic parameters  $\mathbf{K}$  from corresponding scene points  $\mathbf{X}_i$  and image points  $\mathbf{x}_{ki}$  in different views  $k$ .

Some algorithms also compute lens distortion parameters  $\kappa_1, \kappa_2$ .

Software is available:





- Zhang's method [Zha00]:  
Use a single plane checkerboard pattern.  
Website: [research.microsoft.com/en-us/um/people/zhang/calib/](http://research.microsoft.com/en-us/um/people/zhang/calib/)
- Bouguet's toolbox [Bou]:  
Based on Zhang's method.  
Website: [www.vision.caltech.edu/bouguetj/calib\\_doc/](http://www.vision.caltech.edu/bouguetj/calib_doc/)  
C version is available in OpenCV library.

Show different views of a calibration image in front of camera.







Click some points in images as required by the program.  
Then, program works out the camera parameters.

# Reference I

-  J.-Y. Bouguet.  
Camera calibration toolbox for Matlab.
-  R. I. Hartley.  
In defense of the eight-point algorithm.  
*IEEE Trans PAMI*, 19(6):580–593, 1997.
-  B. K. P. Horn, H. M. Hilden, and S. Negahdaripour.  
Closed-form solution of absolute orientation using orthonormal matrices.  
*J. Optical Society of America A*, 5(7):1127–1135, 1988.
-  R. Jain, R. Kasturi, and B. G. Schunck.  
*Machine Vision*.  
McGraw-Hill, 1995.

## Reference II

-  W. K. Leow, C.-C. Chiang, and Y.-P. Hung.  
Localization and mapping of surveillance cameras in city map.  
*In Proc. ACM Int. Conf. on Multimedia*, 2008.
-  L. G. Shapiro and G. C. Stockman.  
*Computer Vision*.  
Prentice-Hall, 2001.
-  R. Szeliski.  
*Computer Vision: Algorithms and Applications*.  
Springer, 2010.
-  Z. Zhang.  
A flexible new technique for camera calibration.  
*IEEE Trans. PAMI*, 22(11):1330–1334, 2000.