计算机视觉
Computer Vision

-- Reconstruction 1

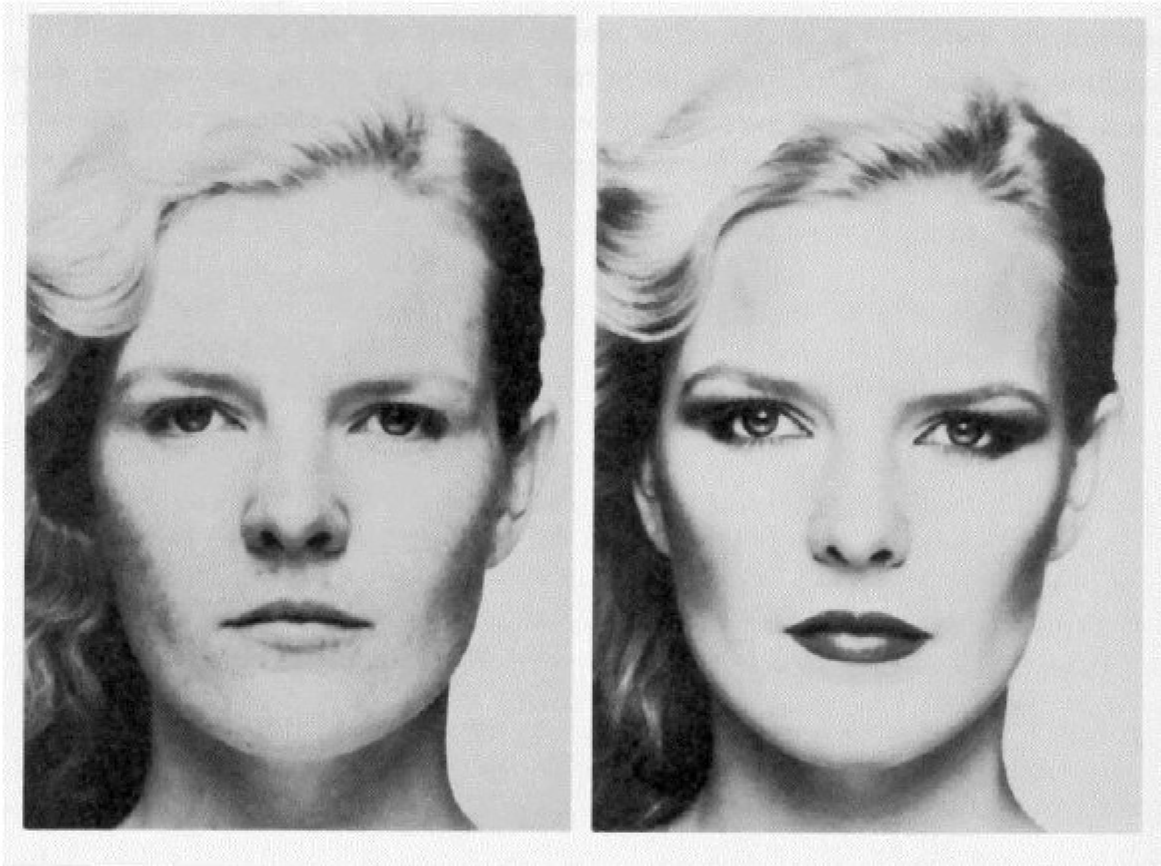
钟 凡
zhongfan@sdu.edu.cn

三维感知

- 人眼具备获取场景三维信息的能力

Visual Cues

- **Shading**

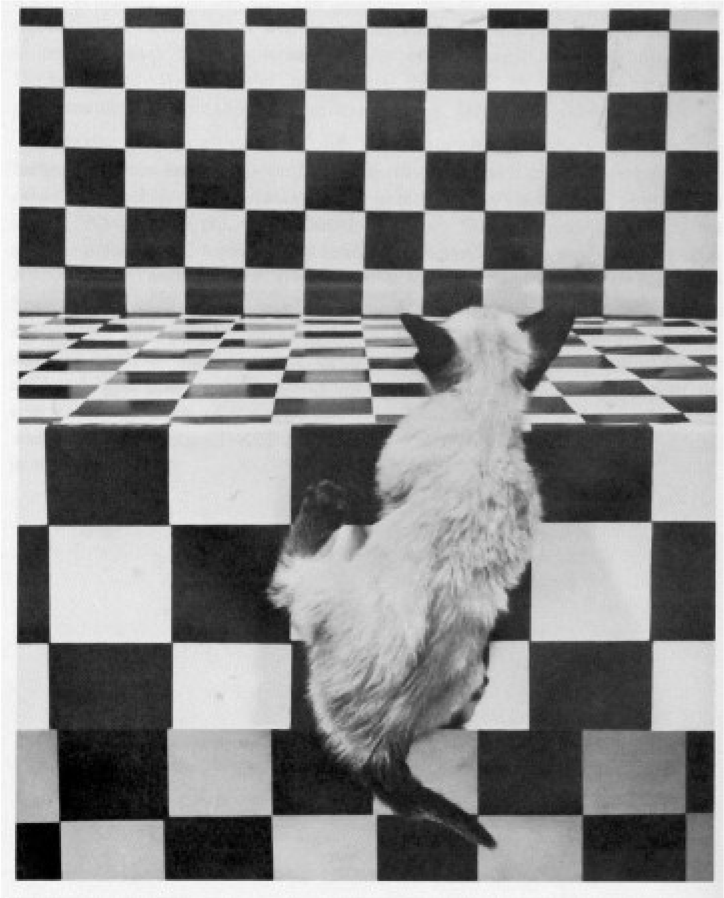


Merle Norman Cosmetics, Los Angeles

Visual Cues

- Shading

- Texture



The Visual Cliff, by William Vandivert, 1960

Visual Cues

- Shading
- Texture
- **Focus**

From *The Art of Photography*, Canon

RWTH AACHEN UNIVERSITY

Visual Cues

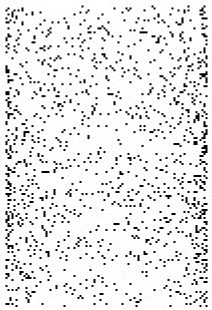
- Shading
- Texture
- Focus
- **Perspective**

Slide credit: Steve Seitz

Visual Cues

- Shading
- Texture
- Focus
- Perspective
- **Motion**

Figures from L. Zhang

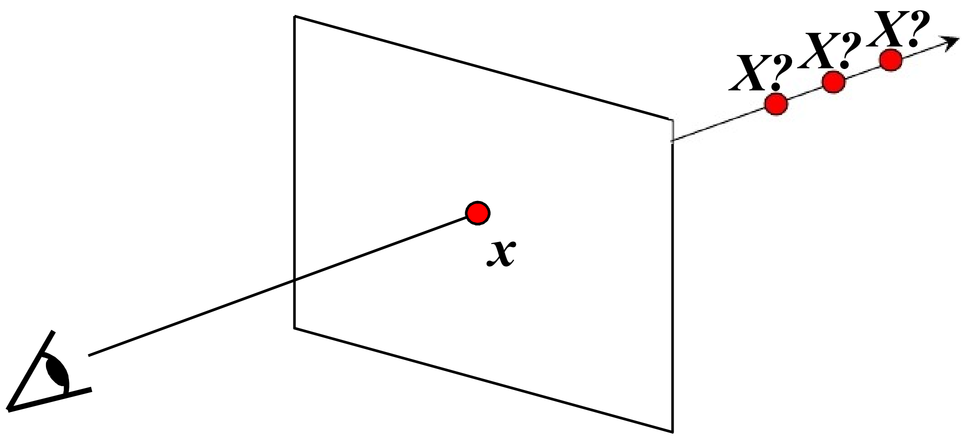


Slide credit: Steve Seitz, Kristen Grauman

http://www.brainconnection.com/teasers/?main=illusion/motion-shape

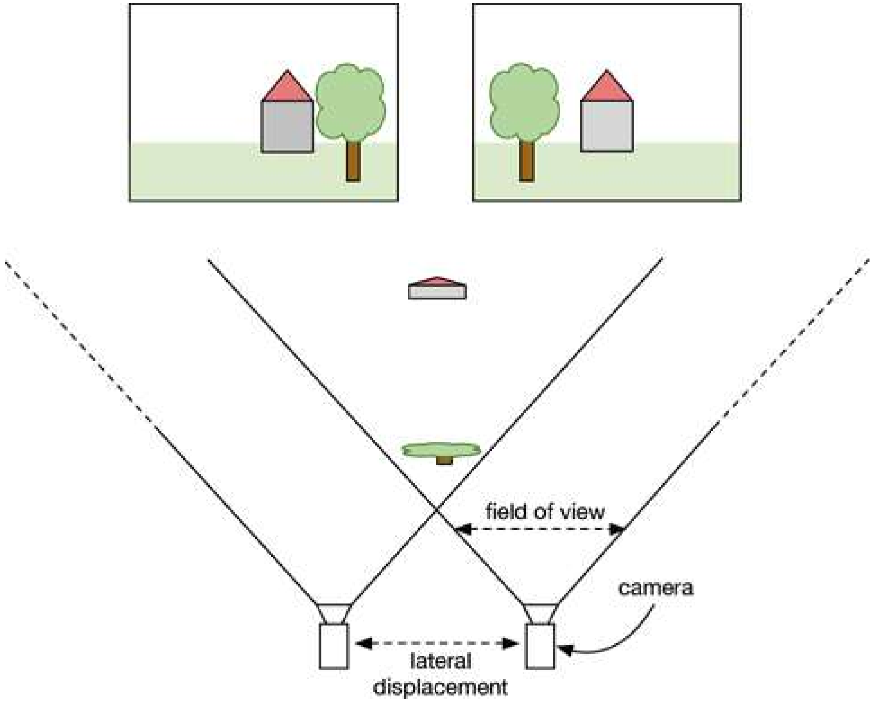
三维感知

- 单张图像?



立体视觉

- 基于双目/多目的三维感知



立体视觉

- 基于双目计算深度（RGB->RGBD）

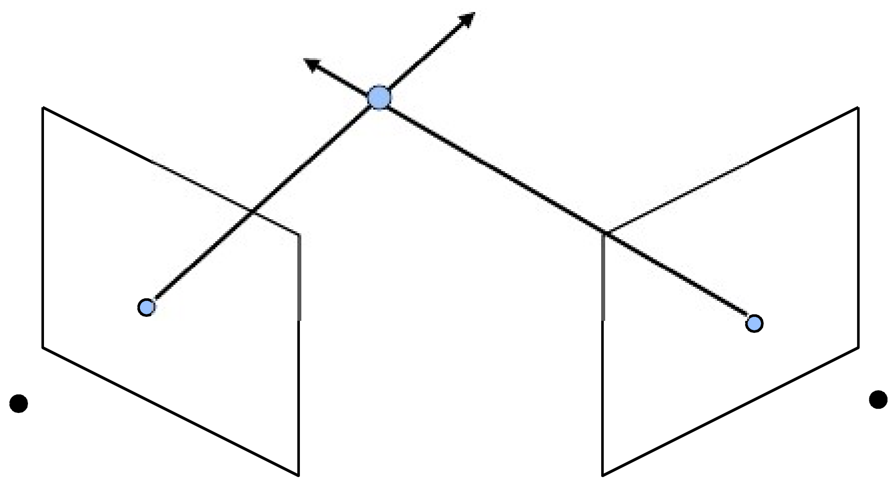
Image 1

Image 2

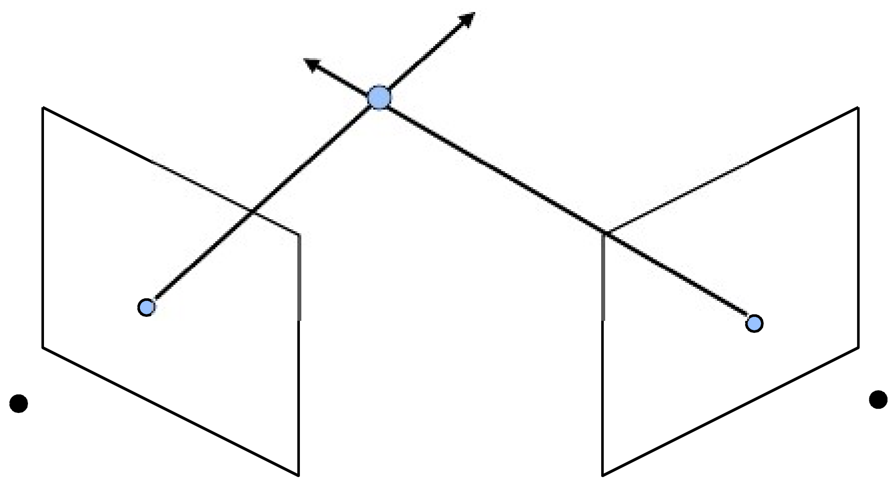
立体视觉

- 基于多目重建三维模型

三角化（Triangulation)



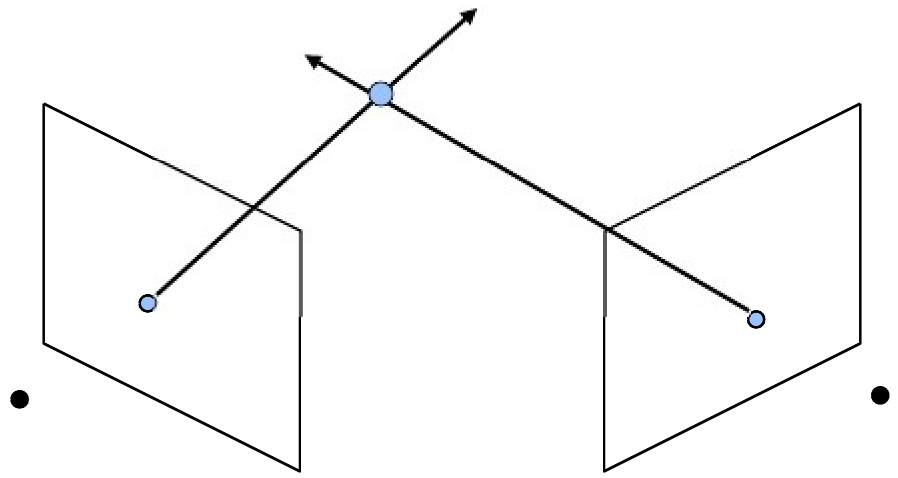
三角化（Triangulation)



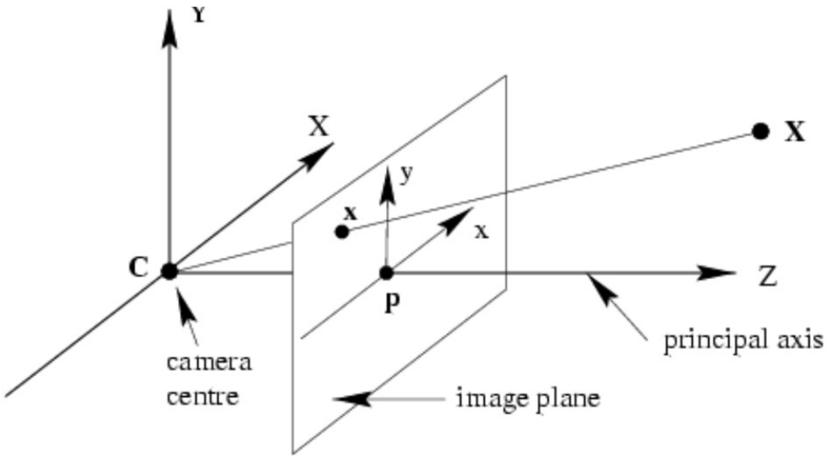
需要知道哪些信息?

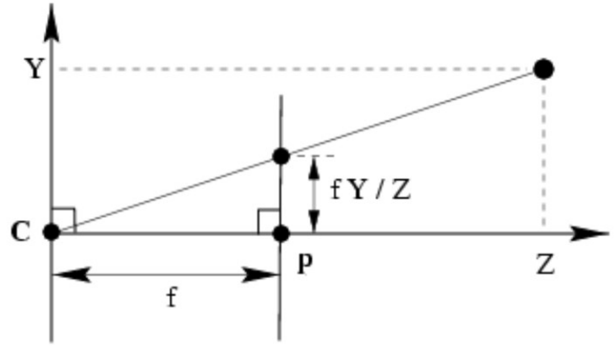
三角化（Triangulation)

- 外参：相机的位置、朝向
- 内参：焦距等参数
- 像素对应关系



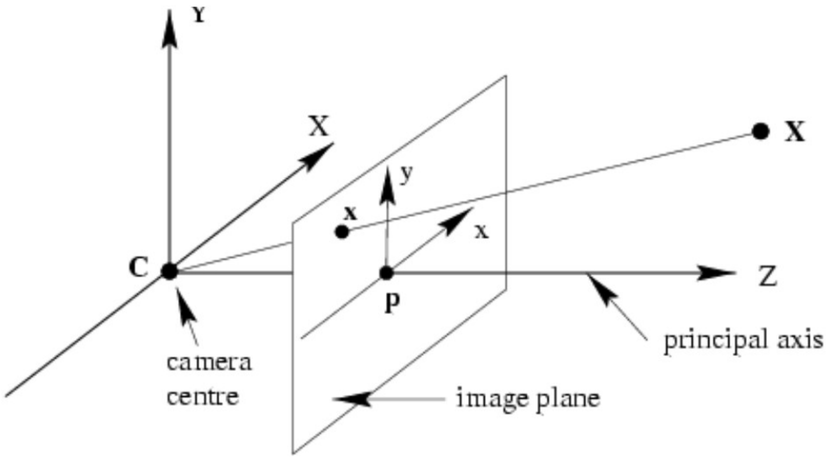
针孔相机模型

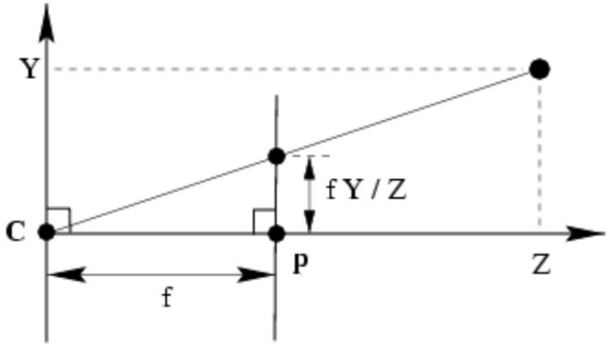




$$(x, y) = (\frac{fX}{Z}, \frac{fY}{Z})$$

针孔相机模型

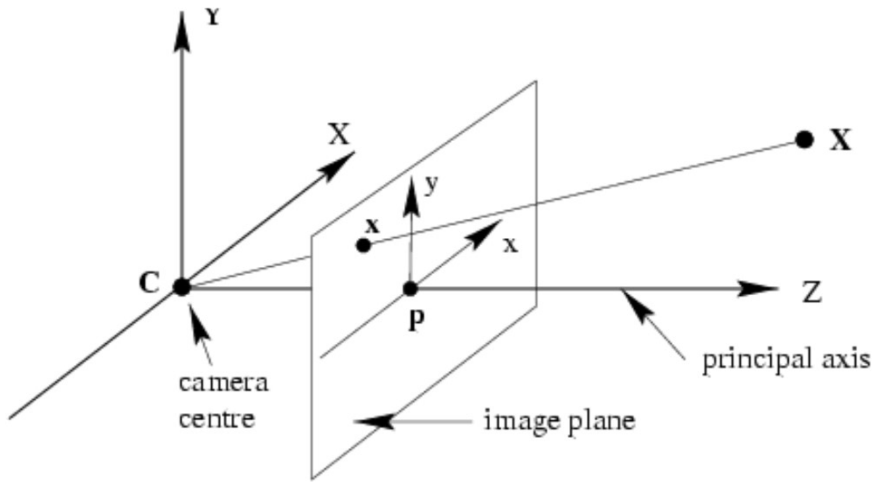




$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} f & & \\ & f & \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & 0 \\ & 1 & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

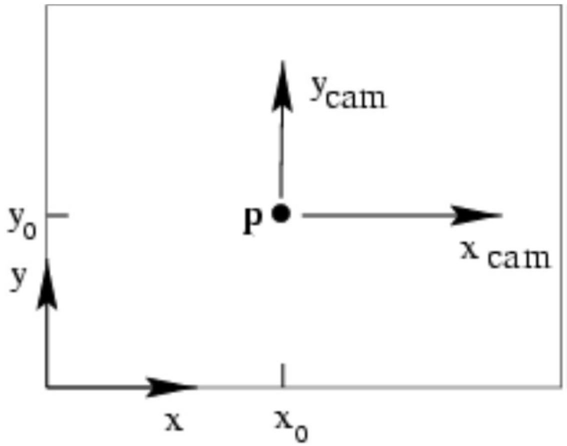
相机坐标系

- 主轴(Principla axis):
  - 从中心出发，与图像平面垂直
- 主点(Principla point):
  - 主轴与图像平面的交点，理想情况在图像中心 p=(0, 0)



主点偏移

- 主点不在图像中心

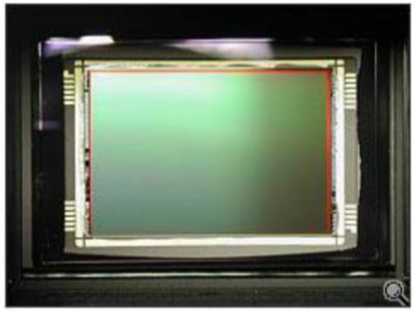


$$p = (p_x, p_y)$$

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & 0 \\ & 1 & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

像素长宽比

- **CCD单元长宽比不为1**

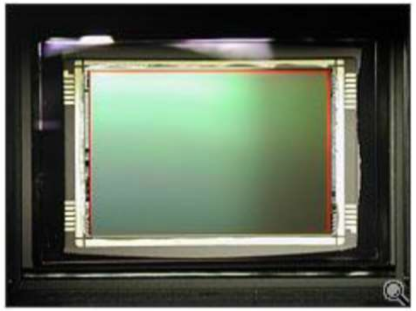


Pixel size: $\frac{1}{m_x} \times \frac{1}{m_y}$

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} m_x & & \\ & m_y & \\ & & 1 \end{bmatrix} \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & 0 \\ & 1 & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

像素长宽比

- **CCD单元长宽比不为1**

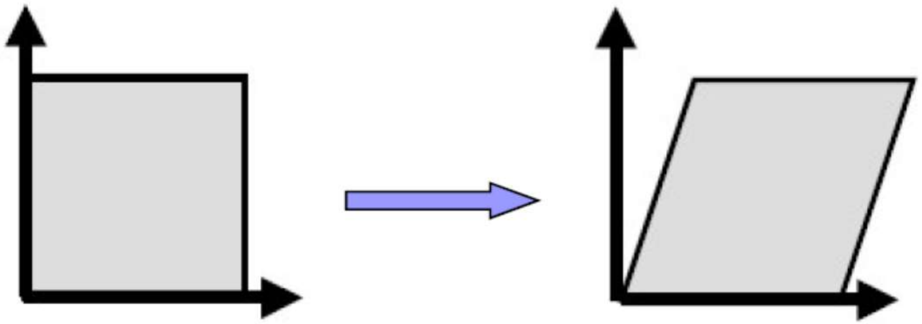


Pixel size: $\frac{1}{m_x} \times \frac{1}{m_y}$

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} f_x & & p_x \\ & f_y & p_y \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & 0 \\ & 1 & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

像素不是矩形

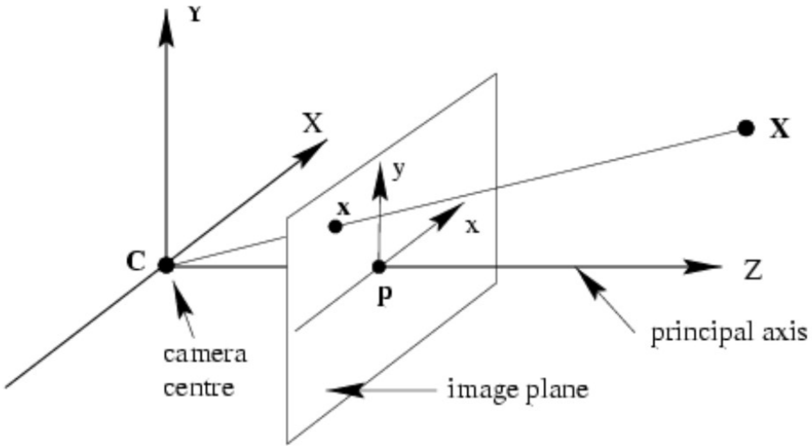
- **CCD行与列不垂直**



$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} f_x & s & p_x \\ & f_y & p_y \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & 0 \\ & 1 & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

相机内参

- 从相机坐标系的三维点 X= (X,Y,Z)到像素坐标的变换

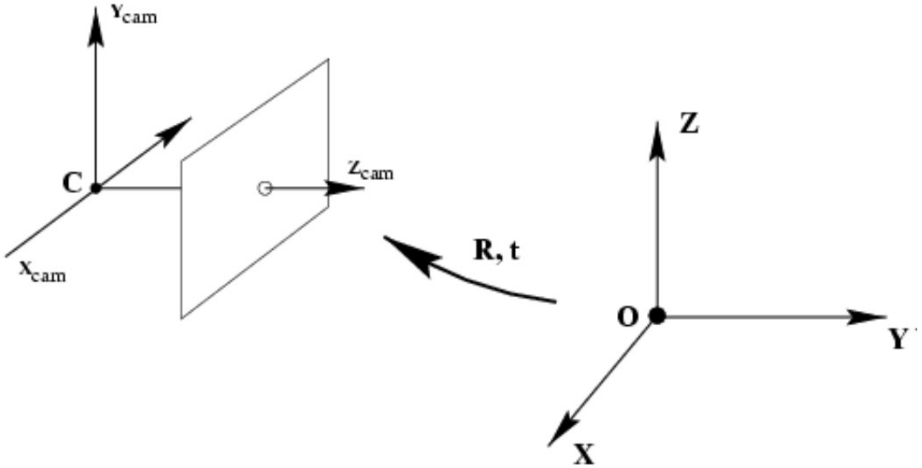


$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} f_x & s & p_x \\ & f_y & p_y \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & 0 \\ & 1 & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

x=K [I | 0] X

内参矩阵K

如果X点不在相机坐标系……



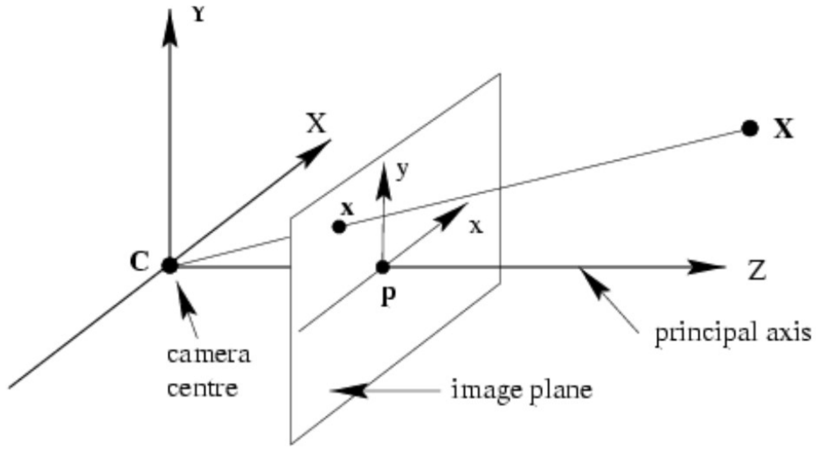
$$\tilde{X}_{cam} = R\left(\tilde{X} - \tilde{C}\right)$$

$$X_{cam} = \begin{bmatrix} R & -R\tilde{C} \\ 0 & 1 \end{bmatrix} \begin{pmatrix} \tilde{X} \\ 1 \end{pmatrix} = \begin{bmatrix} R & -R\tilde{C} \\ 0 & 1 \end{bmatrix} X$$

$$x = K[I \mid 0]X_{cam} = K\left[R \mid -R\tilde{C}\right]X \qquad P = K[R \mid t], \qquad t = -R\tilde{C}$$

相机外参

- 从世界坐标系到相机坐标系的变换



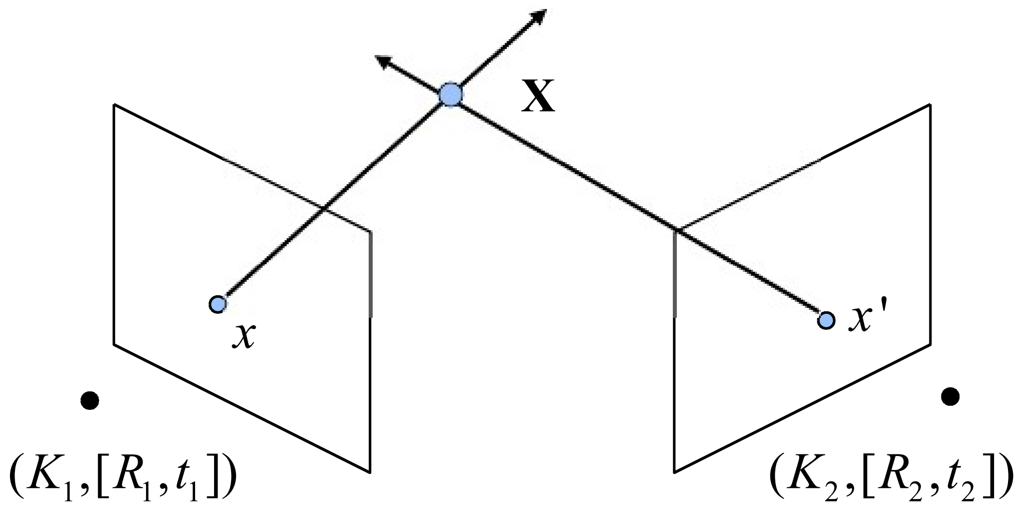
$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} f_x & s & p_x \\ & f_y & p_y \\ & & 1 \end{bmatrix} \begin{bmatrix} \cdots & & & t_x \\ & \cdots & & t_y \\ & & \cdots & t_z \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

x=K [R | t] X

内参矩阵K　　外参矩阵[R|t]

三维重建

- 相机内参
  - 内参矩阵K
  - 只与相机内部结构有关
- 相机外参
  - R, t
- 立体匹配
  - 像素对应关系

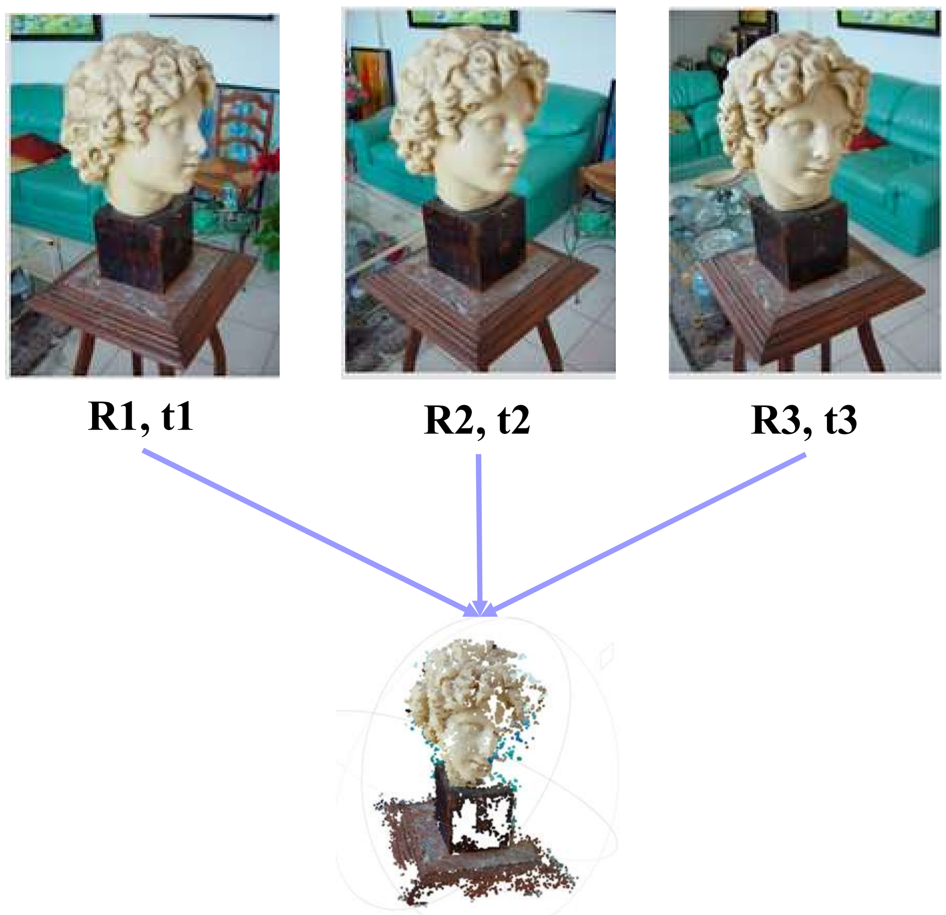


点对(x,x’), 三维点**X**

三维重建

- 相机内参
  - 内参矩阵K
  - 只与相机内部结构有关
- 相机外参
  - R, t
- 立体匹配
  - 像素对应关系

输入：不同视角的图像



输出：三维点云

运动推断结构（Structure from Motion，SFM）

- 运动：相机的运动
- 结构：场景的三维点云
- SFM：从相机运动获取场景的三维点云

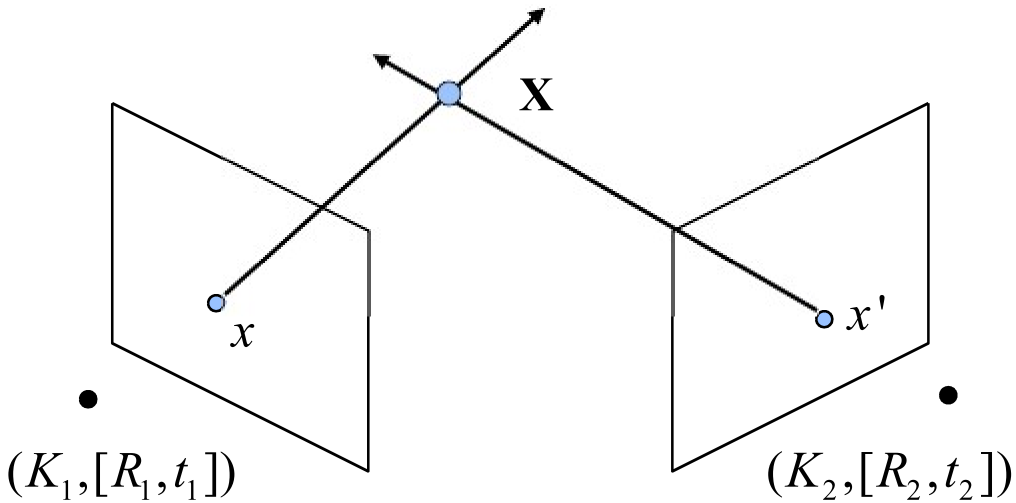
运动推断结构（Structure from Motion，SFM）

- 如果运动已知
  - 相机参数(K1, [R1, t1]) (K2, [R2, t2])……已知

???

运动推断结构（Structure from Motion，SFM）

- 如果运动已知
  - 相机参数(K1, [R1, t1]) (K2, [R2, t2])……已知



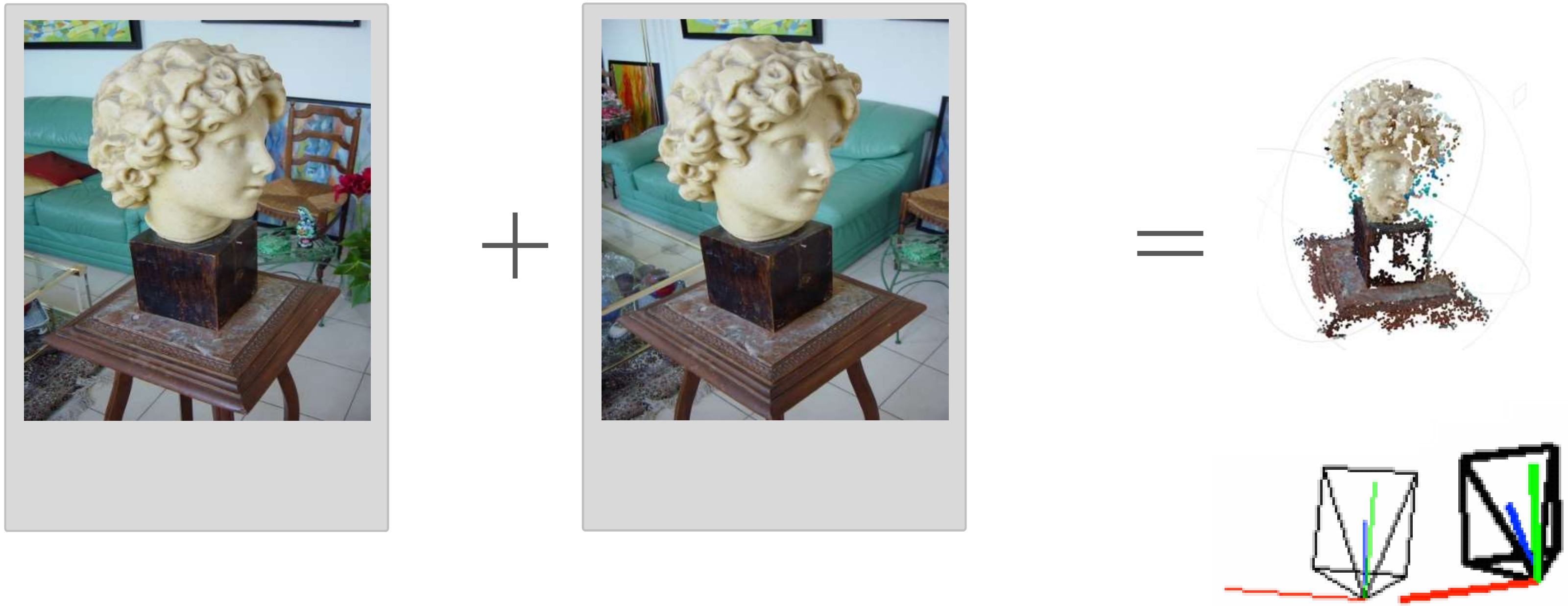
只需要图像匹配+三角化

运动推断结构（Structure from Motion，SFM）

- 运动：相机的运动
- 结构：场景的三维点云
- SFM：从相机运动获取场景的三维点云

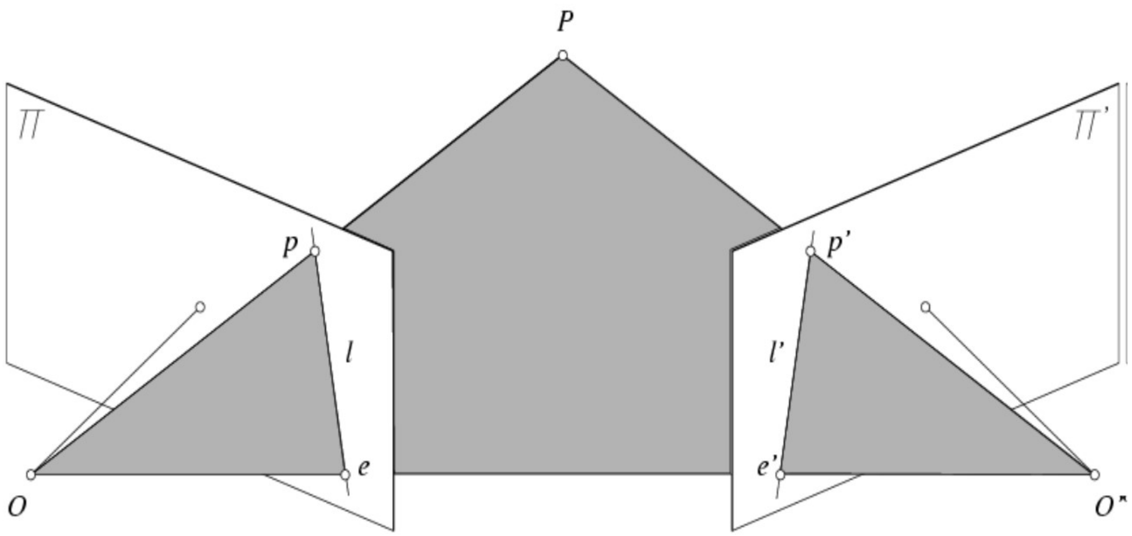
相机运动和三维点云都未知！！

二视图SFM



极线几何（Epipolar Geometry)

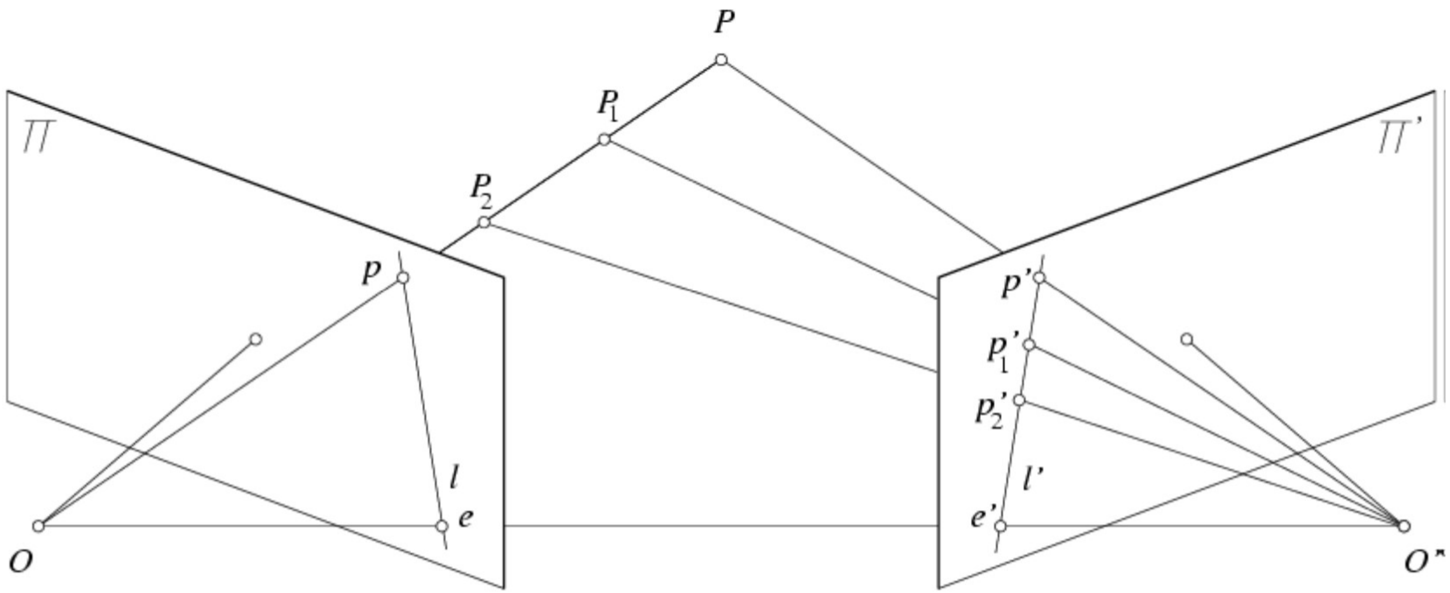
- 相机中心O，O’
- 基线OO’
  - Baseline
  - 相机中心的连线
- 极平面POO’
  - Epipolar Plane
  - P与基线构成的平面
- 极线l, l’
  - Epipolar Line
  - 极平面与图像的交
- 极点e, e’
  - Epipolar Point
  - 基线与图像的交



所有极平面交于基线

所有极线交于极点

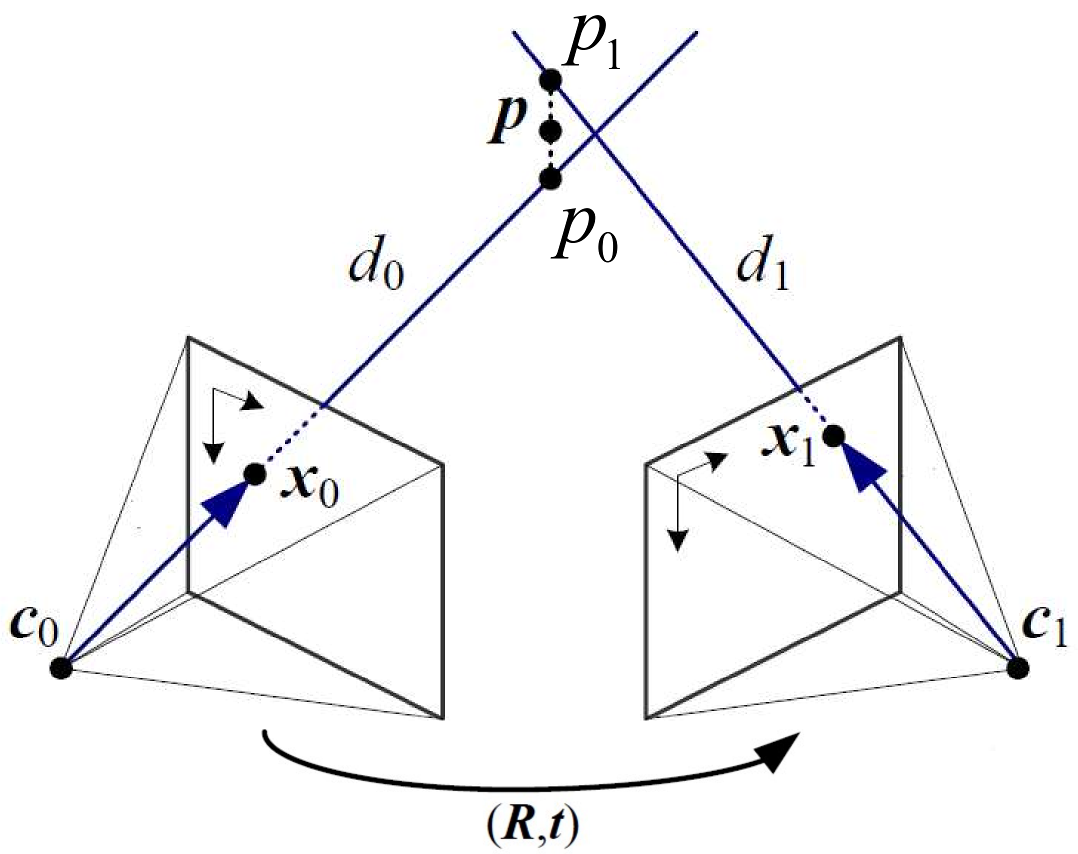
极线约束（Epipolar Constraint)



左视图点p在右视图的对应点p’一定位于l’上

右视图点p’在左视图的对应点p一定位于l上

极线约束（Epipolar Constraint)

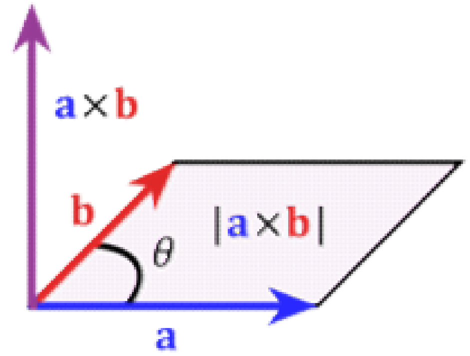


$$\hat{x}_j = K^{-1}x_j, \|\hat{x}_j\| = 1$$

p0和p1分别是p在Cam0和Cam1的相机坐标系下的坐标

$$d_1\hat{\boldsymbol{x}}_1 = \boldsymbol{p}_1 = \boldsymbol{R}\boldsymbol{p}_0 + \boldsymbol{t} = \boldsymbol{R}(d_0\hat{\boldsymbol{x}}_0) + \boldsymbol{t}$$

向量叉积



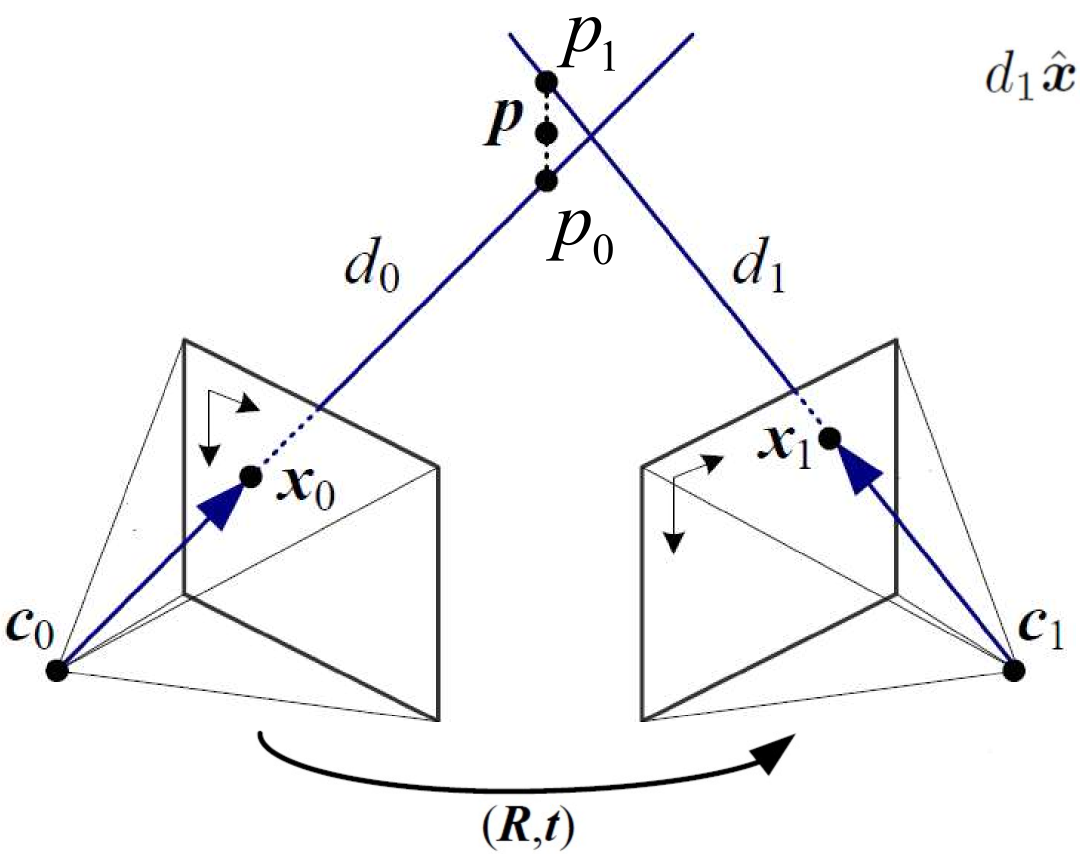
$$\mathbf{a}\times\mathbf{b} = \begin{vmatrix} a_2 & a_3 \\ b_2 & b_3 \end{vmatrix}\mathbf{i} - \begin{vmatrix} a_1 & a_3 \\ b_1 & b_3 \end{vmatrix}\mathbf{j} + \begin{vmatrix} a_1 & a_2 \\ b_1 & b_2 \end{vmatrix}\mathbf{k}$$

$$= (a_2 b_3 - a_3 b_2)\mathbf{i} - (a_1 b_3 - a_3 b_1)\mathbf{j} + (a_1 b_2 - a_2 b_1)\mathbf{k}.$$

$$a \times b = [a]_\times b \qquad [a]_\times = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix}$$

叉积矩阵

极线约束（Epipolar Constraint)



$$d_1\hat{\boldsymbol{x}}_1 = \boldsymbol{p}_1 = \boldsymbol{R}\boldsymbol{p}_0 + \boldsymbol{t} = \boldsymbol{R}(d_0\hat{\boldsymbol{x}}_0) + \boldsymbol{t}$$

两边与t叉积

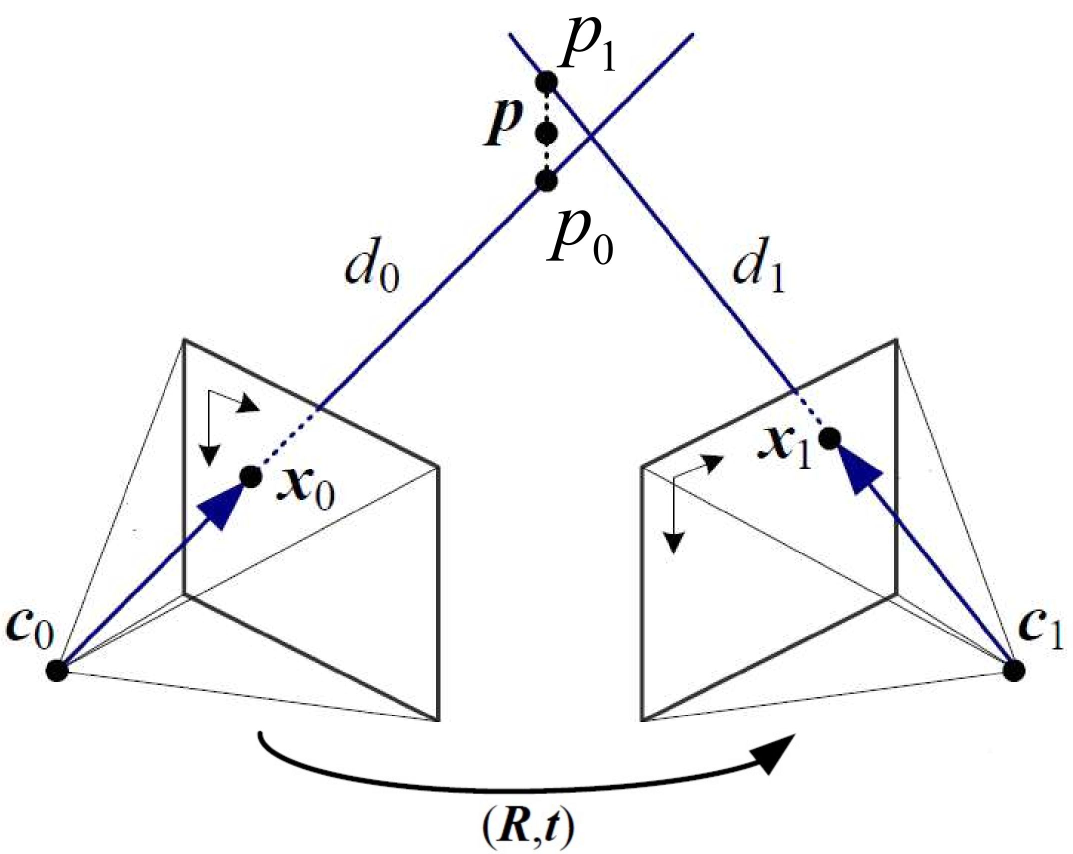
$$d_1[\boldsymbol{t}]_\times\hat{\boldsymbol{x}}_1 = d_0[\boldsymbol{t}]_\times\boldsymbol{R}\hat{\boldsymbol{x}}_0$$

两边与 $\hat{x}_1$ 点积

$$d_0\hat{\boldsymbol{x}}_1^T([\boldsymbol{t}]_\times\boldsymbol{R})\hat{\boldsymbol{x}}_0 = d_1\hat{\boldsymbol{x}}_1^T[\boldsymbol{t}]_\times\hat{\boldsymbol{x}}_1 = 0.$$

$$\hat{\boldsymbol{x}}_1^T\boldsymbol{E}\,\hat{\boldsymbol{x}}_0 = 0 \qquad \boldsymbol{E} = [\boldsymbol{t}]_\times\boldsymbol{R}$$

本质矩阵（Essential Matrix)



$$\hat{\boldsymbol{x}}_1^T \boldsymbol{E} \, \hat{\boldsymbol{x}}_0 = 0$$

$$\boldsymbol{E} = [\boldsymbol{t}]_\times \boldsymbol{R}$$

$$\boldsymbol{E} = [\boldsymbol{t}]_{\times} \boldsymbol{R}$$

本质矩阵E由R, t决定

已知E，能否求出R, t？

从 E 到 R, t

- 任意本质矩阵都可以通过SVD分解为如下形式:

$$E = U\mathrm{diag}(1,1,0)V^T$$

本质矩阵的秩为2，且两个非零奇异值相等

从 E 到 R, t

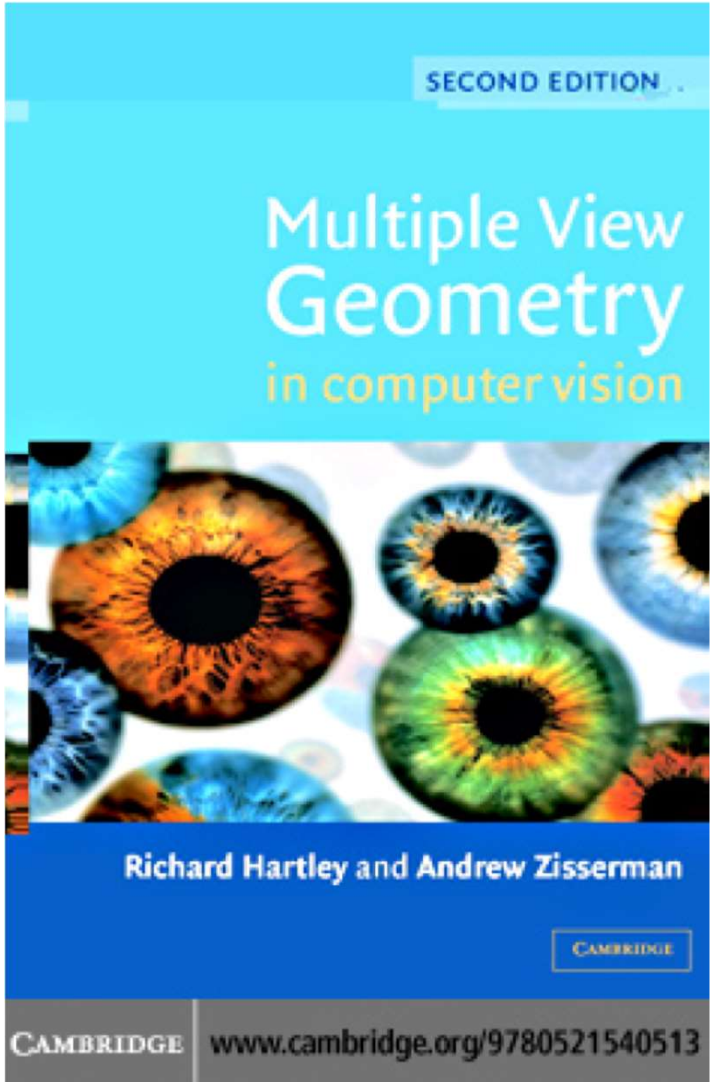
- 对 $E=[t]_\times R=U\mathrm{diag}(1,1,0)V^T$ ，相应的[R, t]存在4种可能：

$$[R,t]=[UWV^T,+u_3] \text{ or } [UWV^T,-u_3]$$
$$\text{or } [UW^TV^T,+u_3] \text{ or } [UW^TV^T,-u_3]$$

u_3 是U的第3个（最小奇异值）奇异向量

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

从 E 到 R, t



上述结论的证明参考 9.6节

从 E 到 R, t

- 4种可能的解

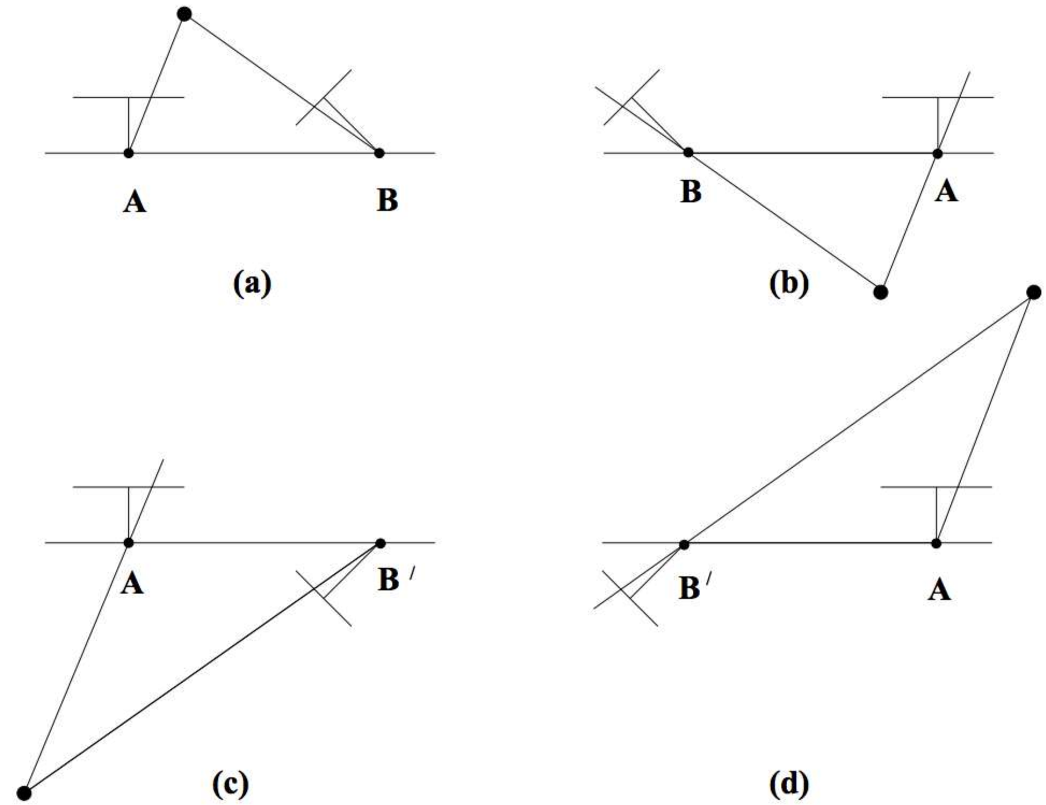


Fig. 9.12. **The four possible solutions for calibrated reconstruction from E.** *Between the left and right sides there is a baseline reversal. Between the top and bottom rows camera B rotates* 180° *about the baseline. Note, only in (a) is the reconstructed point in front of both cameras.*

只有(a)计算出的三维点在两个相机前方!

从 E 到 R, t

- 已知R, t，可通过三角化计算三维点坐标
- 对四种可能解，分别计算三维点，选取三维点在相机前方最多的解作为结果

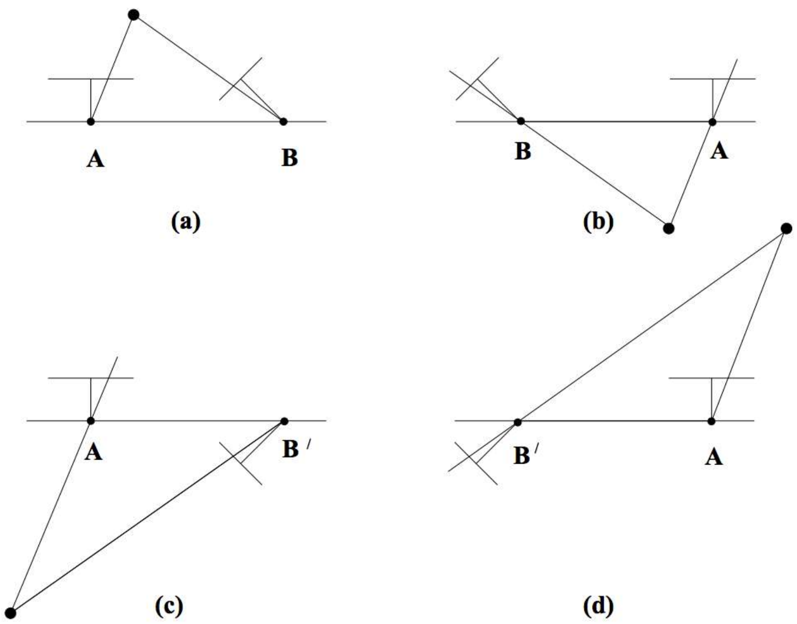
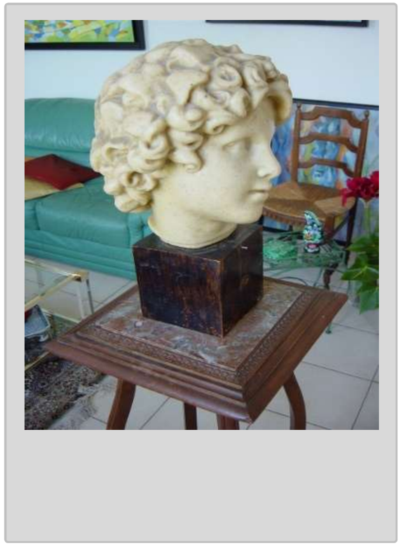
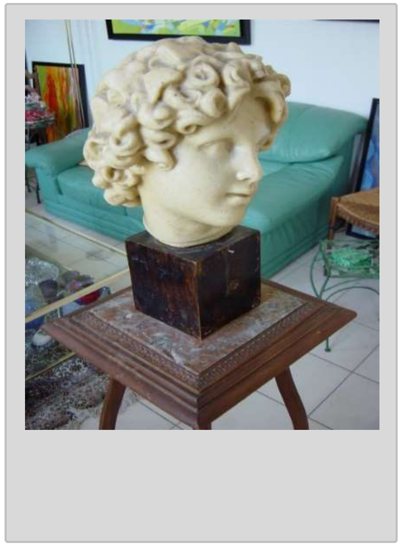


Fig. 9.12. **The four possible solutions for calibrated reconstruction from E.** *Between the left and right sides there is a baseline reversal. Between the top and bottom rows camera B rotates 180° about the baseline. Note, only in (a) is the reconstructed point in front of both cameras.*

如何计算本质矩阵E?

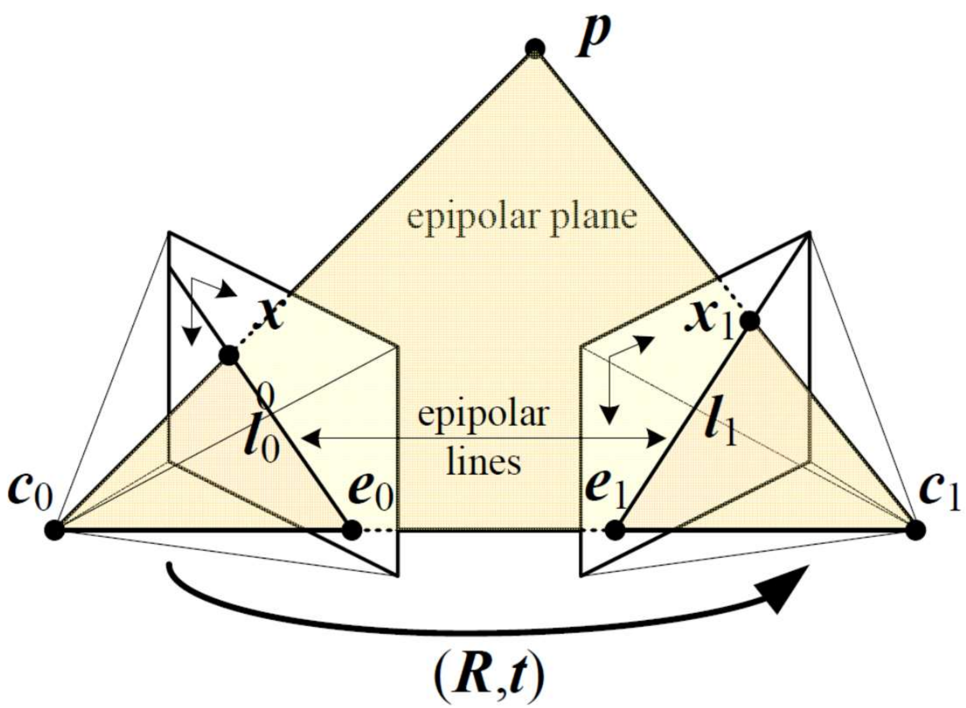


+



$\rightarrow$ **E**

基础矩阵（Fundamental Matrix)



$$\hat{\boldsymbol{x}}_1^T \boldsymbol{E}\, \hat{\boldsymbol{x}}_0 = 0$$

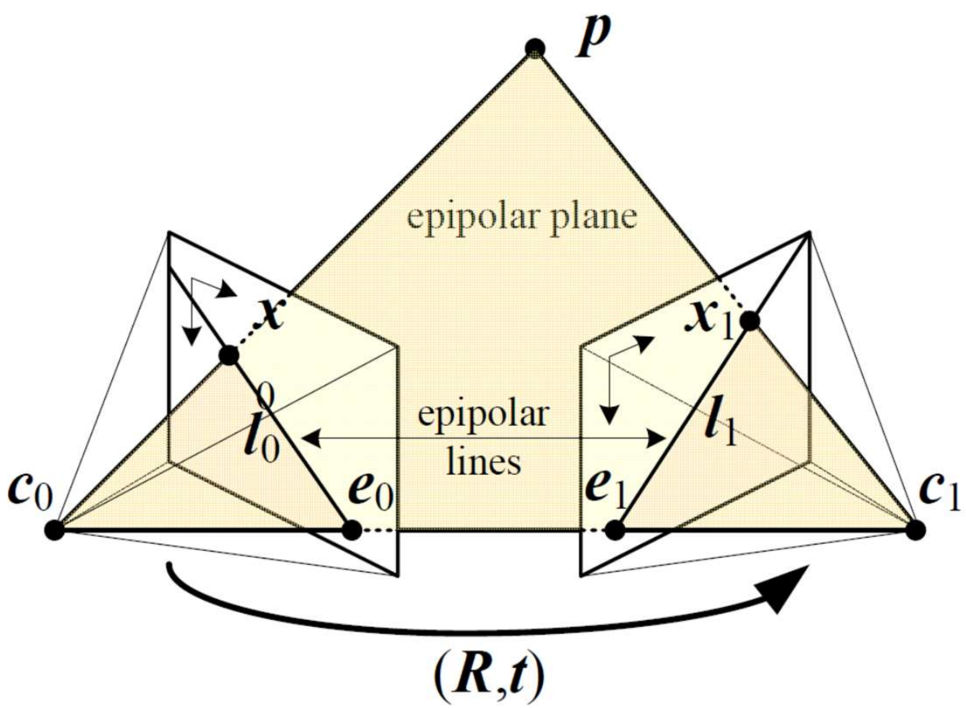
$$\hat{x}_j = K_j^{-1} x_j, \| \hat{x}_j \| = 1$$

$$x_1^T K_1^{-T} E K_0^{-1} x_0 = 0$$

$$x_1^T F x_0 = 0$$

基础矩阵： $F = K_1^{-T} E K_0^{-1}$

基础矩阵（Fundamental Matrix)



$$x_1^T F x_0 = 0$$

x0, x1为对应点像素坐标

基础矩阵（Fundamental Matrix)

$$x'^T Fx = 0 \quad \Leftrightarrow \quad (x', y', 1)\begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{pmatrix}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = 0$$

$$\Downarrow$$

$$x'xf_{11} + x'yf_{12} + x'f_{13} + y'xf_{21} + y'yf_{22} + y'f_{23} + xf_{31} + yf_{32} + f_{33} = 0$$

$\Downarrow$ 如果有n个点对 $(x_i, x'_i), i = 1, \ldots, n$

$$\mathbf{Af} = \begin{bmatrix} x'_1x_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ x'_nx_n & x'_ny_n & x'_n & y'_nx_n & y'_ny_n & y'_n & x_n & y_n & 1 \end{bmatrix}\mathbf{f} = \mathbf{0}$$

§ findFundamentalMat() [1/2]

```
Mat cv::findFundamentalMat ( InputArray   points1,
                             InputArray   points2,
                             int          method = FM_RANSAC,
                                          ransacReprojThreshold =
                             double       3.,
                             double       confidence = 0.99,
                             OutputArray  mask = noArray()
                           )
```

Python:

```
retval, mask = cv.findFundamentalMat( points1, points2[, method[, ransacReprojThreshold[, confidence[, mask]]]] )
```

Calculates a fundamental matrix from the corresponding points in two images.

Parameters

points1	Array of N points from the first image. The point coordinates should be floating-point (single or double precision).
points2	Array of the second image points of the same size and format as points1 .
method	Method for computing a fundamental matrix. • **CV_FM_7POINT** for a 7-point algorithm. $N = 7$ • **CV_FM_8POINT** for an 8-point algorithm. $N \geq 8$ • **CV_FM_RANSAC** for the RANSAC algorithm. $N \geq 8$ • **CV_FM_LMEDS** for the LMedS algorithm. $N \geq 8$

基础矩阵 => 本质矩阵

$$F = K_1^{-T} E K_0^{-1}$$

$$\Downarrow$$

$$E = K_1^{T} F K_0$$

从对应点直接估计本质矩阵

n个点对 $(x_i, x'_i), i = 1, \ldots, n$

$$\hat{x} = K^{-1}x$$

n个点对 $(\hat{x}_i, \hat{x}'_i), i = 1, \ldots, n$

$$\hat{x}'^T E \hat{x} = 0$$

§ findEssentialMat() [1/2]

```
Mat cv::findEssentialMat ( InputArray    points1,
                           InputArray    points2,
                           InputArray    cameraMatrix,
                           int           method = RANSAC,
                           double        prob = 0.999,
                           double        threshold = 1.0,
                           OutputArray   mask = noArray()
                         )
```

Python:

```
retval, mask = cv.findEssentialMat( points1, points2, cameraMatrix[, method[, prob[, threshold[, mask]]]] )
retval, mask = cv.findEssentialMat( points1, points2[, focal[, pp[, method[, prob[, threshold[, mask]]]]]]     )
```

Calculates an essential matrix from the corresponding points in two images.

Parameters

points1 Array of N (N >= 5) 2D points from the first image. The point coordinates should be floating-point (single or double precision).

points2 Array of the second image points of the same size and format as points1 .

cameraMatrix Camera matrix $K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$. Note that this function assumes that points1 and points2 are feature points from cameras with the same camera matrix.

method Method for computing an essential matrix.

- **RANSAC** for the RANSAC algorithm.
- **LMEDS** for the LMedS algorithm.

$\boldsymbol{E} = [\boldsymbol{t}]_{\times}\boldsymbol{R}$ 且两非零奇异值相等

所以自由度为5

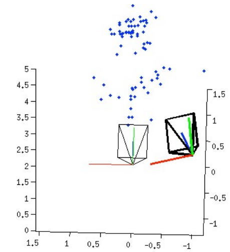
二视图SFM

- 内参矩阵K已知

keypoints → match

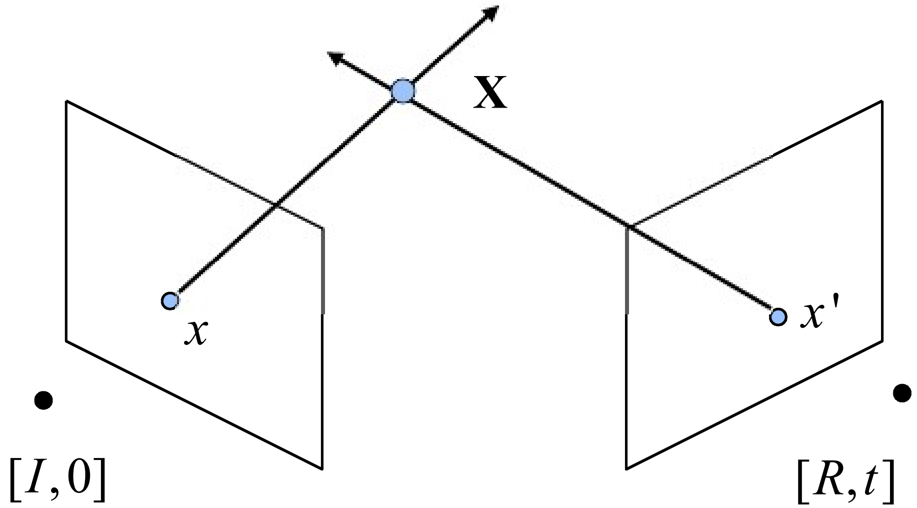
keypoints → match

match → fundamental matrix → essential matrix → [R|t] → triangulation →



二视图SFM

- 只能求出相机的相对位置和姿态
- 位移t只能得到方向



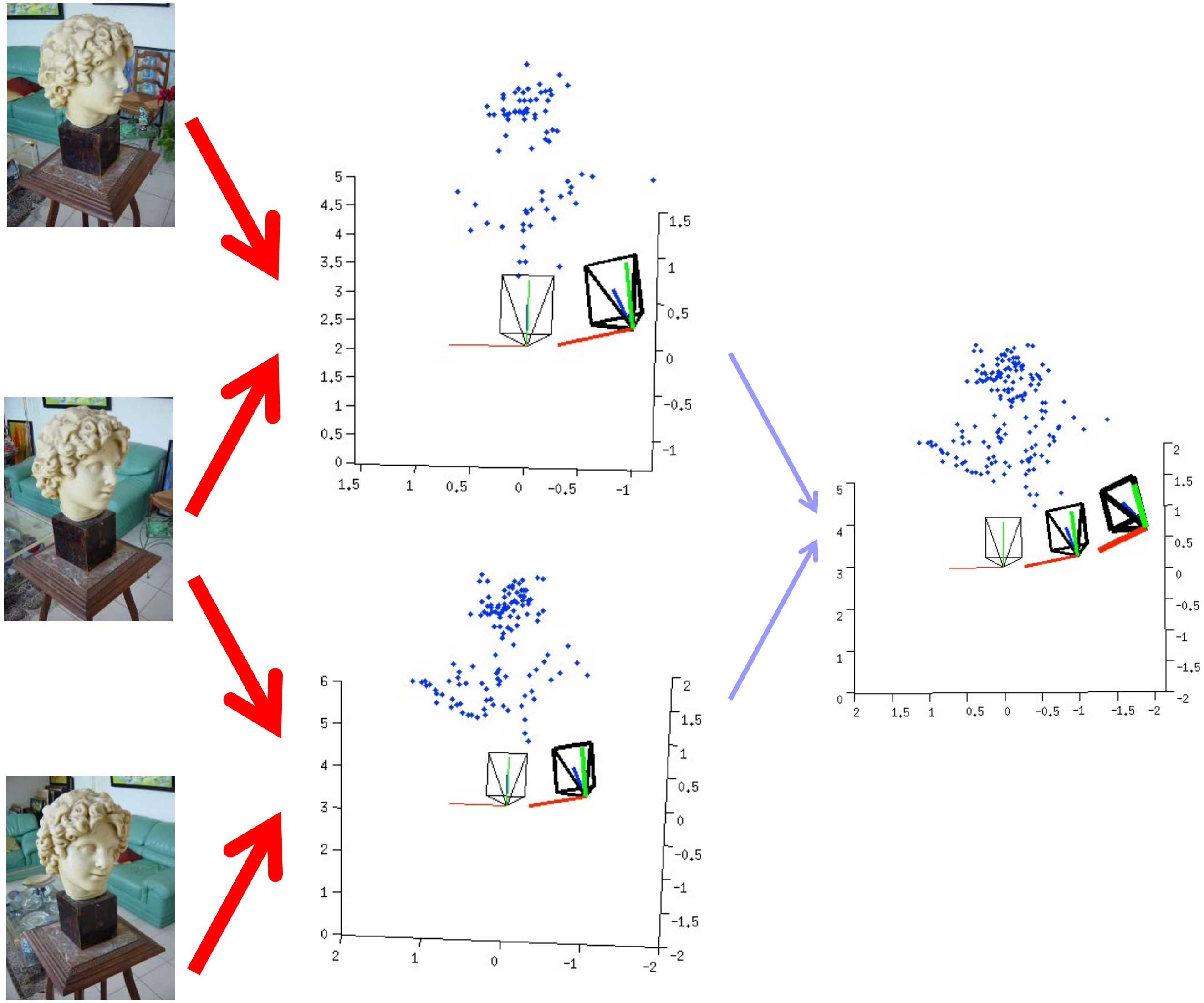
以第1个相机坐标系为参考坐标系

$$[R,t]=[UWV^T,+u_3]\ \text{or} \ldots$$

$$\| t \| = \| u_3 \| = 1$$

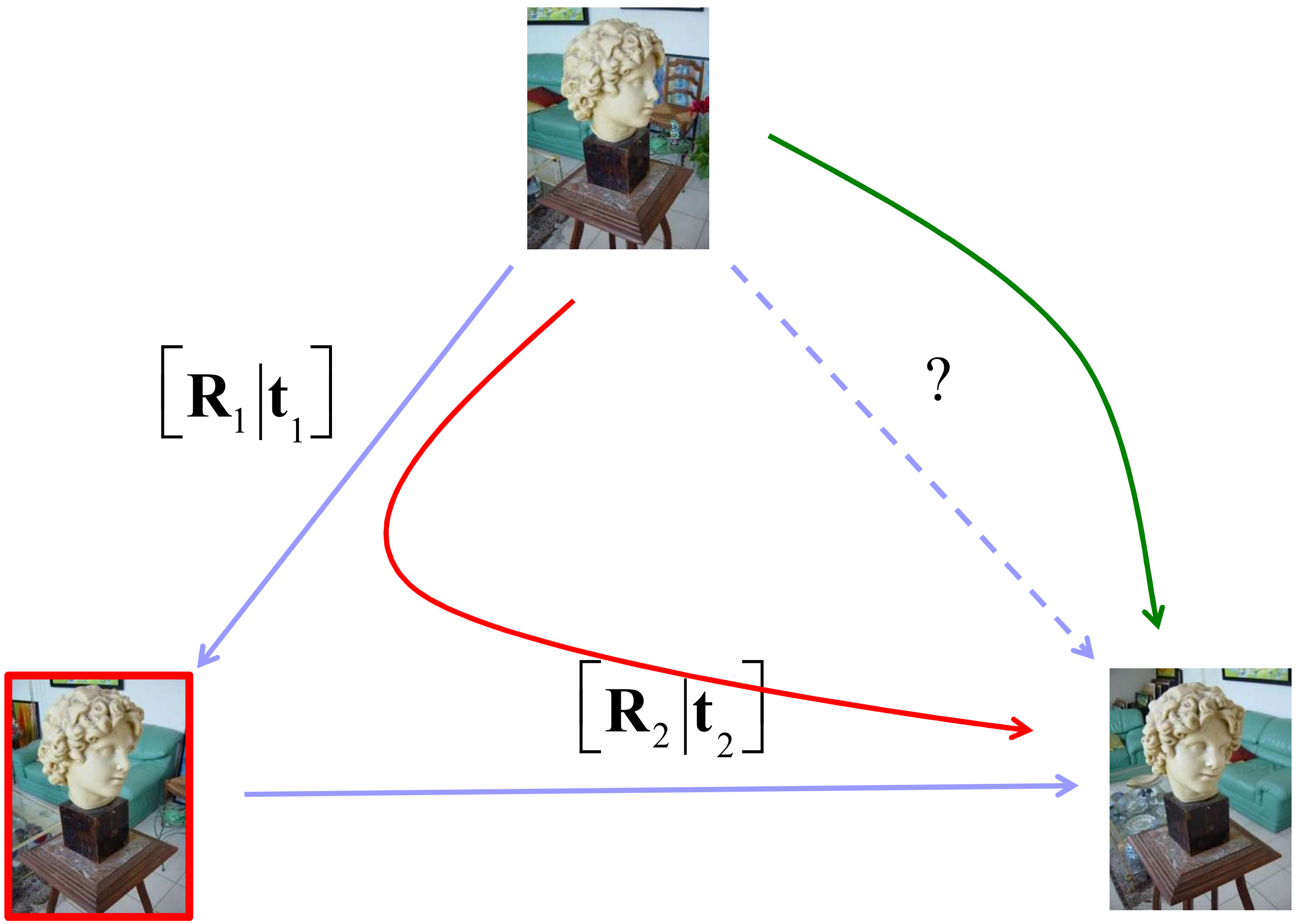
t 的长度未知

多个视图

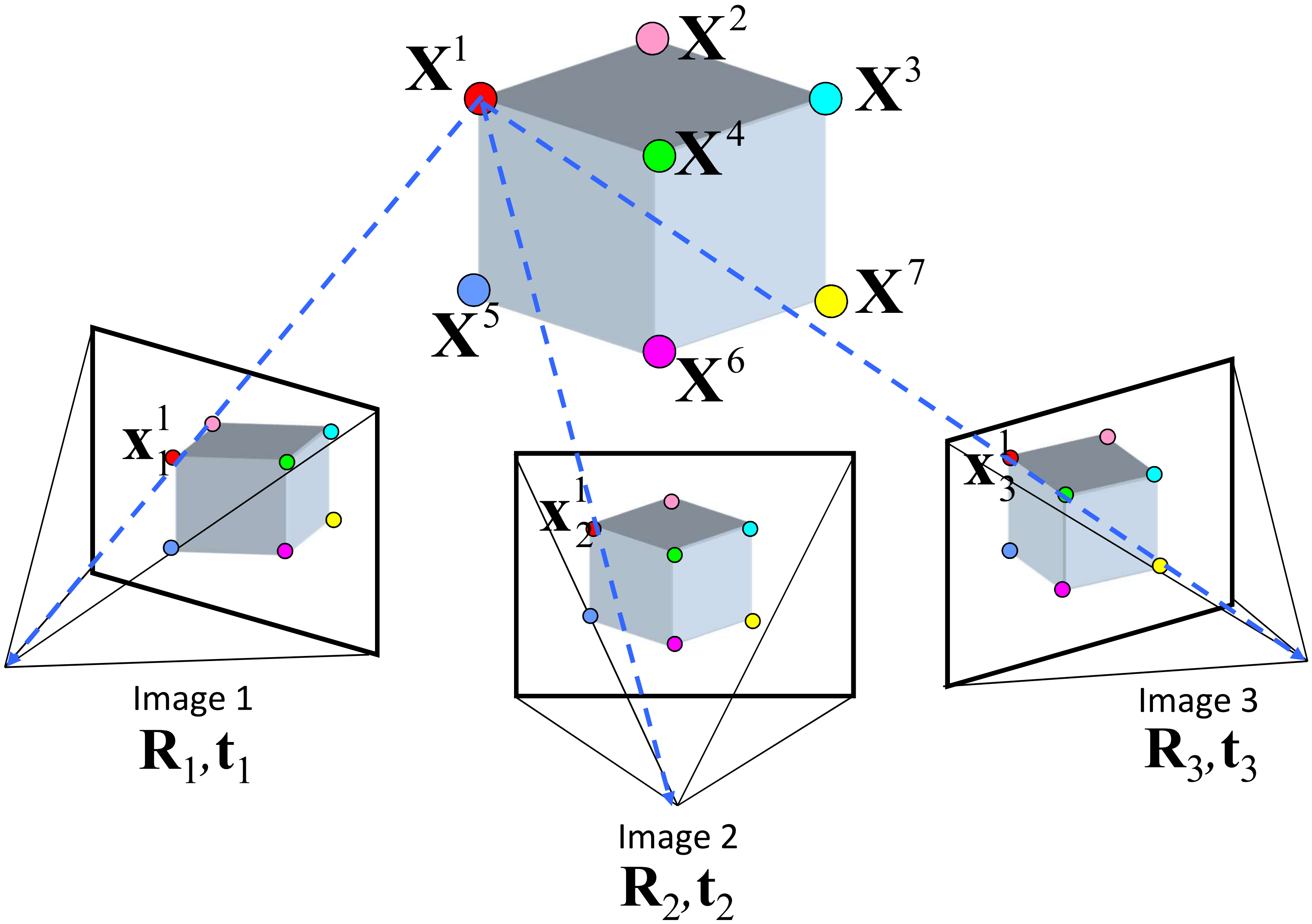


多个视图

- 二视图+融合的方法可能存在冲突

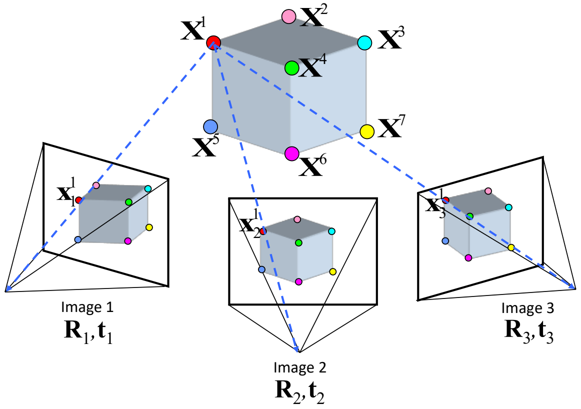


同一三维点可能在多个视图同时被观察到



同一三维点可能在多个视图同时被观察到

	Point 1	Point 2	Point 3
Image 1	$\mathbf{x}_1^1 = \mathbf{K}\left[\mathbf{R}_1 \middle\| \mathbf{t}_1\right]\mathbf{X}^1$	$\mathbf{x}_1^2 = \mathbf{K}\left[\mathbf{R}_1 \middle\| \mathbf{t}_1\right]\mathbf{X}^2$	
Image 2	$\mathbf{x}_2^1 = \mathbf{K}\left[\mathbf{R}_2 \middle\| \mathbf{t}_2\right]\mathbf{X}^1$	$\mathbf{x}_2^2 = \mathbf{K}\left[\mathbf{R}_2 \middle\| \mathbf{t}_2\right]\mathbf{X}^2$	$\mathbf{x}_2^3 = \mathbf{K}\left[\mathbf{R}_2 \middle\| \mathbf{t}_2\right]\mathbf{X}^3$
Image 3	$\mathbf{x}_3^1 = \mathbf{K}\left[\mathbf{R}_3 \middle\| \mathbf{t}_3\right]\mathbf{X}^1$		$\mathbf{x}_3^3 = \mathbf{K}\left[\mathbf{R}_3 \middle\| \mathbf{t}_3\right]\mathbf{X}^3$



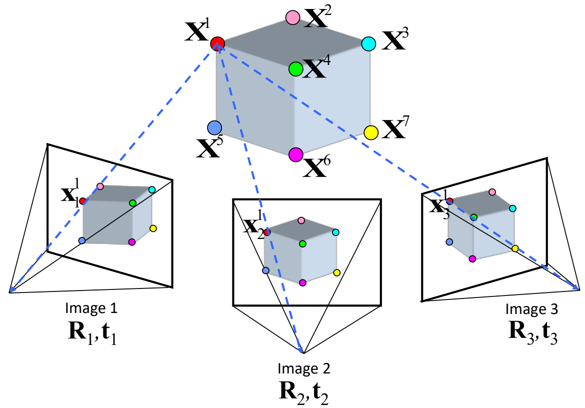
同一三维点可能在多个视图同时被观察到

	Point 1	Point 2	Point 3
Image 1	$\mathbf{x}_1^1=\mathbf{K}\left[\mathbf{R}_1\middle\|\mathbf{t}_1\right]\mathbf{X}^1$	$\mathbf{x}_1^2=\mathbf{K}\left[\mathbf{R}_1\middle\|\mathbf{t}_1\right]\mathbf{X}^2$	
Image 2	$\mathbf{x}_2^1=\mathbf{K}\left[\mathbf{R}_2\middle\|\mathbf{t}_2\right]\mathbf{X}^1$	$\mathbf{x}_2^2=\mathbf{K}\left[\mathbf{R}_2\middle\|\mathbf{t}_2\right]\mathbf{X}^2$	$\mathbf{x}_2^3=\mathbf{K}\left[\mathbf{R}_2\middle\|\mathbf{t}_2\right]\mathbf{X}^3$
Image 3	$\mathbf{x}_3^1=\mathbf{K}\left[\mathbf{R}_3\middle\|\mathbf{t}_3\right]\mathbf{X}^1$		$\mathbf{x}_3^3=\mathbf{K}\left[\mathbf{R}_3\middle\|\mathbf{t}_3\right]\mathbf{X}^3$

$\left[\mathbf{R}_1\middle|\mathbf{t}_1\right],\left[\mathbf{R}_2\middle|\mathbf{t}_2\right],\left[\mathbf{R}_3\middle|\mathbf{t}_3\right]$ 和 $\mathbf{X}^1,\mathbf{X}^2,\mathbf{X}^3,\cdots$

需要使得在图像上的投影 $x_1^1,x_2^1,x_3^1,x_1^2,\ldots$

与图像上对应特征点的位置 $\tilde{x}_1^1,\tilde{x}_2^1,\tilde{x}_3^1,\tilde{x}_1^2,\ldots$ 尽量一致



集束调整（Bundle Adjustment)

$$\min \sum_i \sum_j \left(\tilde{\mathbf{x}}_i^j - \mathbf{K}\left[\mathbf{R}_i \middle| \mathbf{t}_i\right]\mathbf{X}^j \right)^2$$

- 非线性最小二乘
- 使用二视图结果初始化

$\left[\mathbf{R}_1 \middle| \mathbf{t}_1\right], \left[\mathbf{R}_2 \middle| \mathbf{t}_2\right], \left[\mathbf{R}_3 \middle| \mathbf{t}_3\right]$ 和 $\mathbf{X}^1, \mathbf{X}^2, \mathbf{X}^3, \cdots$

需要使得在图像上的投影 $x_1^1, x_2^1, x_3^1, x_1^2, \ldots$

与图像上对应特征点的位置 $\tilde{x}_1^1, \tilde{x}_2^1, \tilde{x}_3^1, \tilde{x}_1^2, \ldots$ 尽量一致