Outline

- Epipolar geometry
- Eight-point algorithm
- Recovering R and T from E

Some slides were based on notes by Luke Fletcher

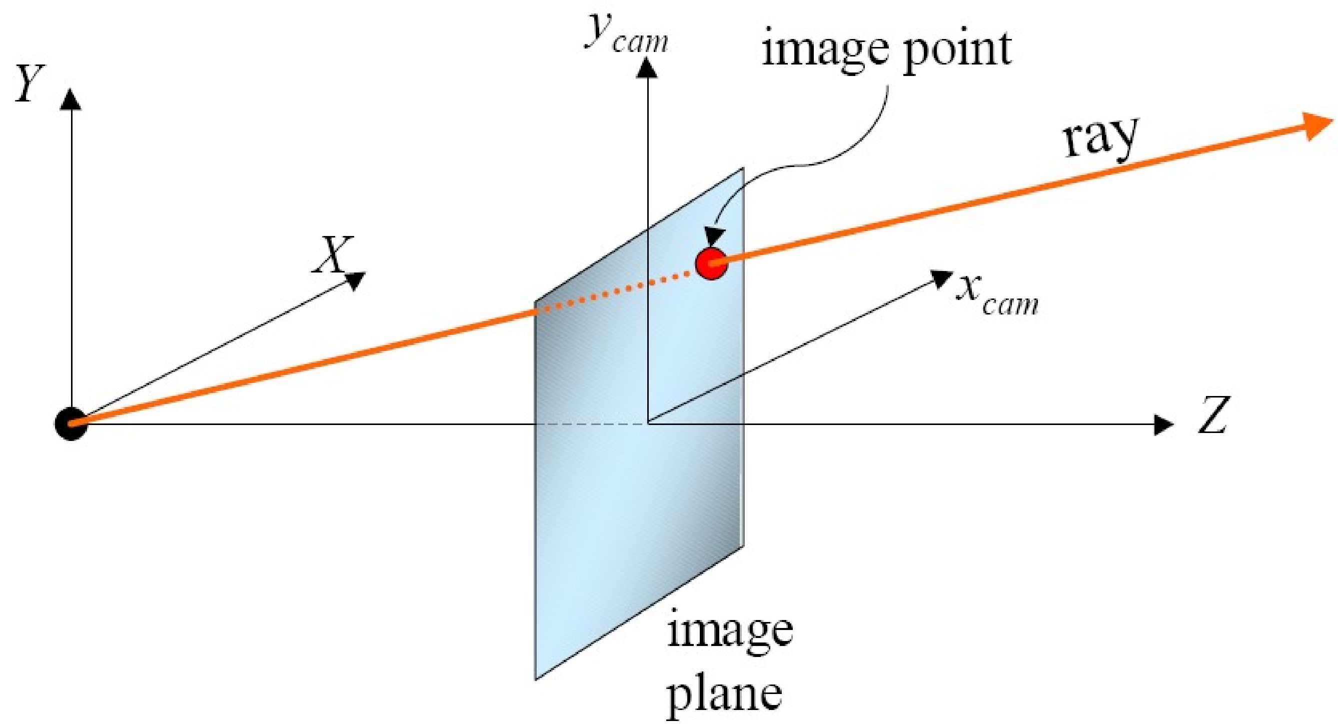
From 3D Points to Pixels

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}^{\mathrm{T}} & -\mathbf{R}^{\mathrm{T}}\mathbf{t} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

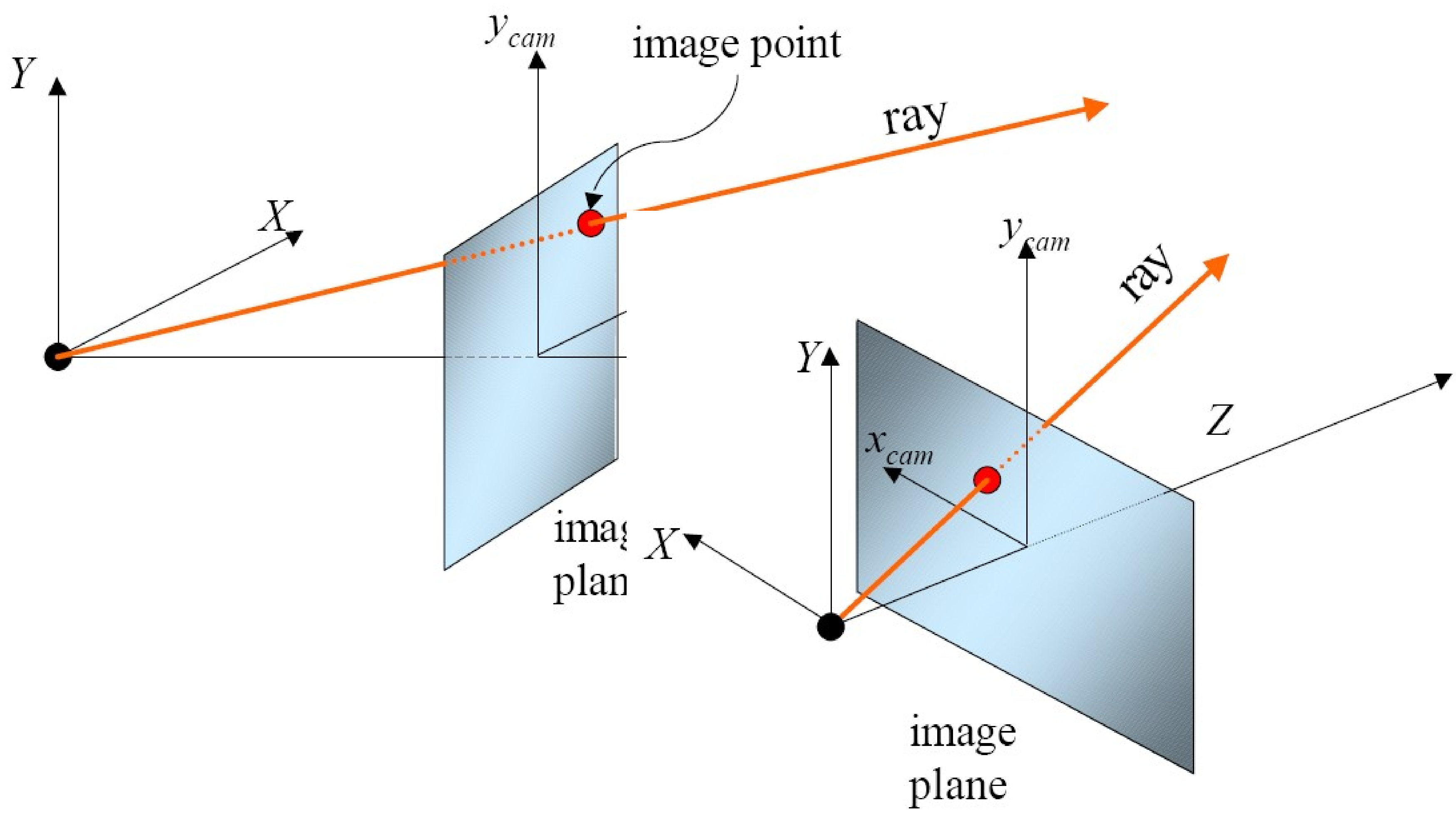
$$\Leftrightarrow \mathbf{x} = \mathbf{K}\left[\mathbf{R}^{\mathrm{T}} \mid -\mathbf{R}^{\mathrm{T}}\mathbf{t}\right]\mathbf{X}$$

$$\Leftrightarrow \mathbf{x} = \mathbf{P}\,\mathbf{X}$$

The Epipolar Geometry

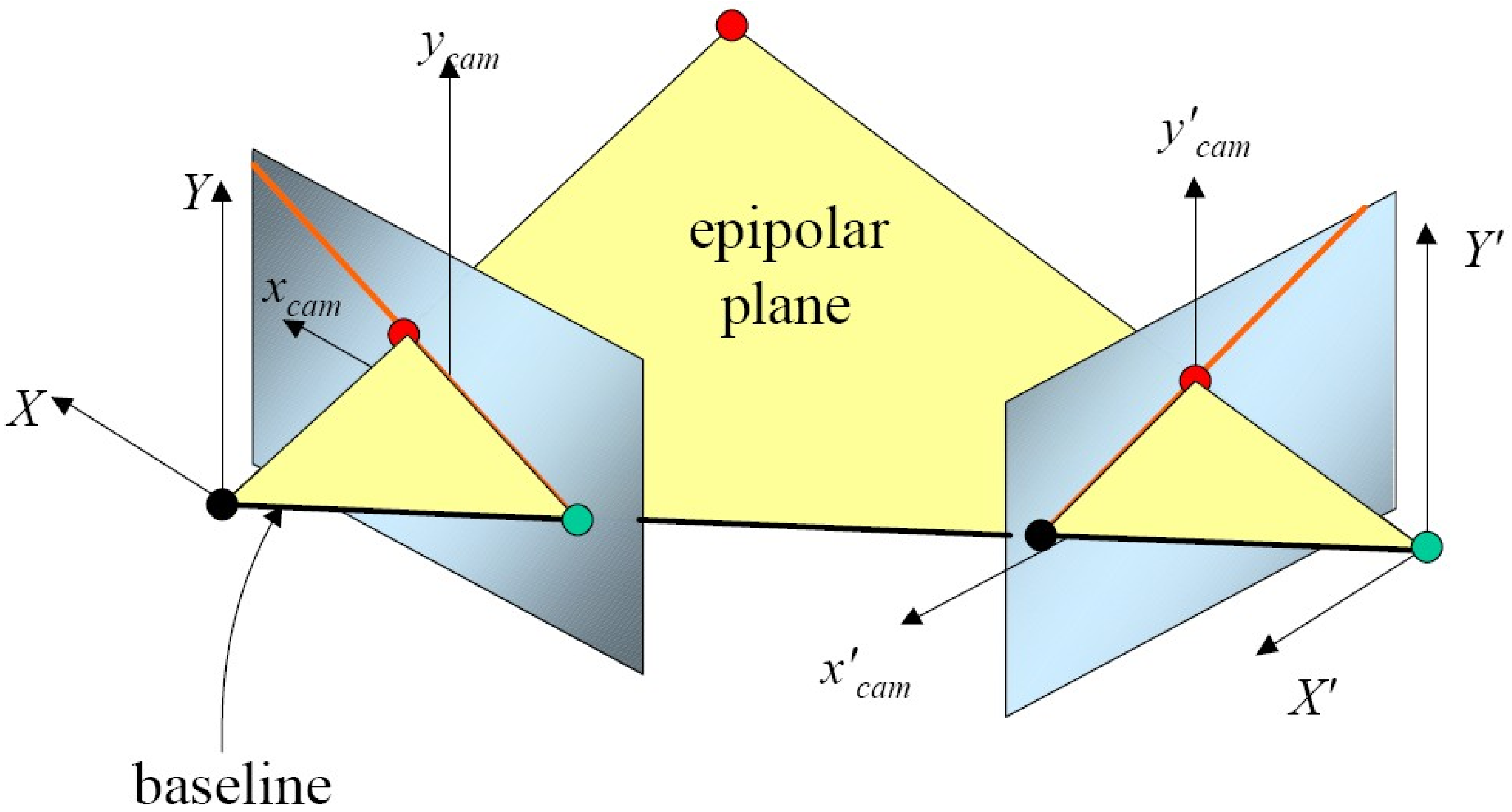


The Epipolar Geometry

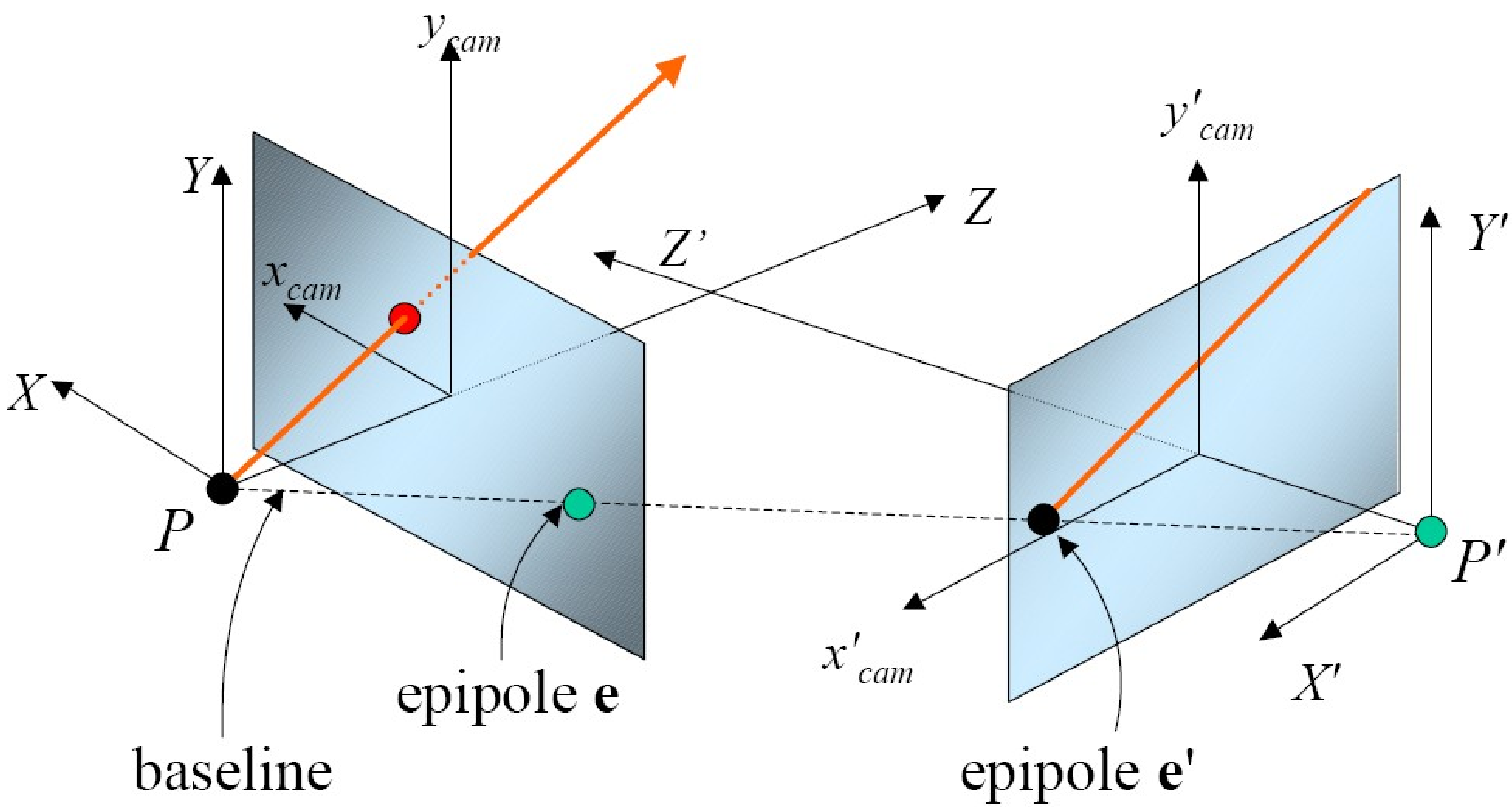


The Epipolar Geometry

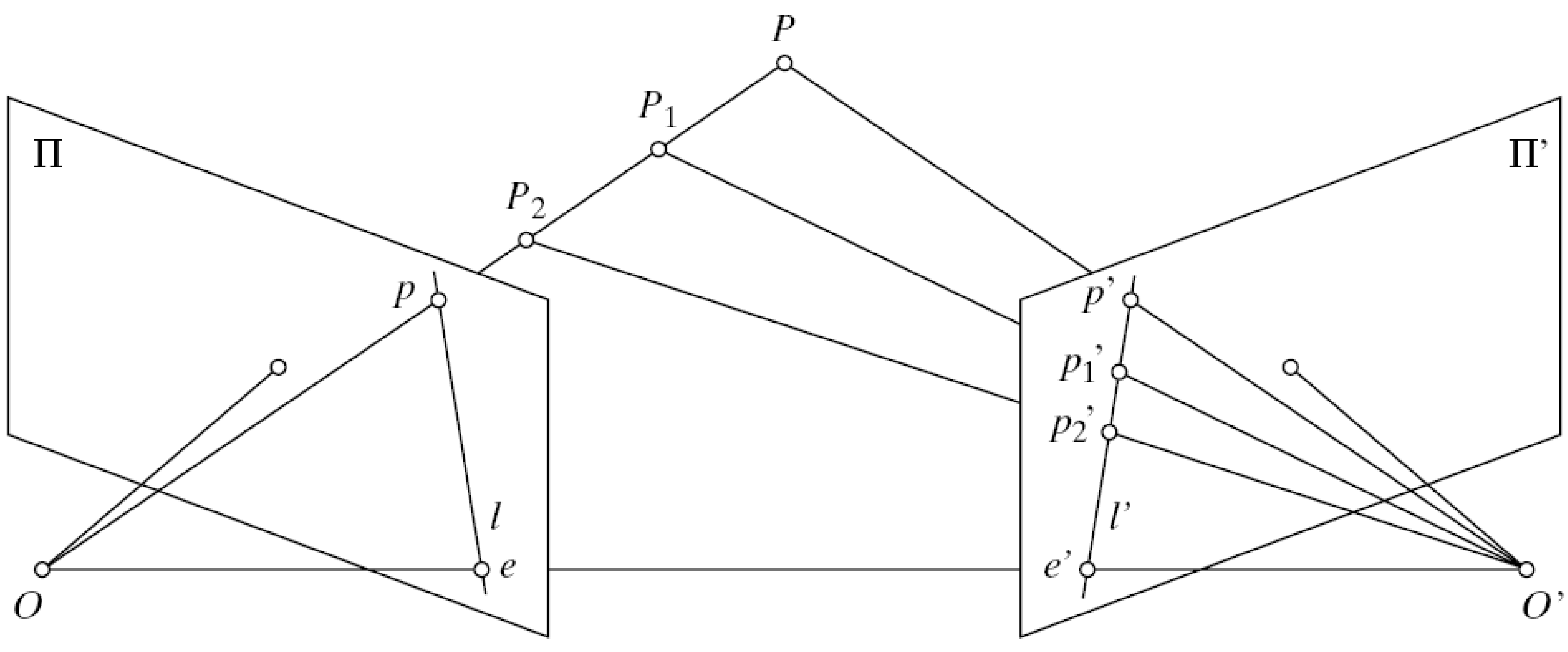
- Assume that we have a set of correspondences



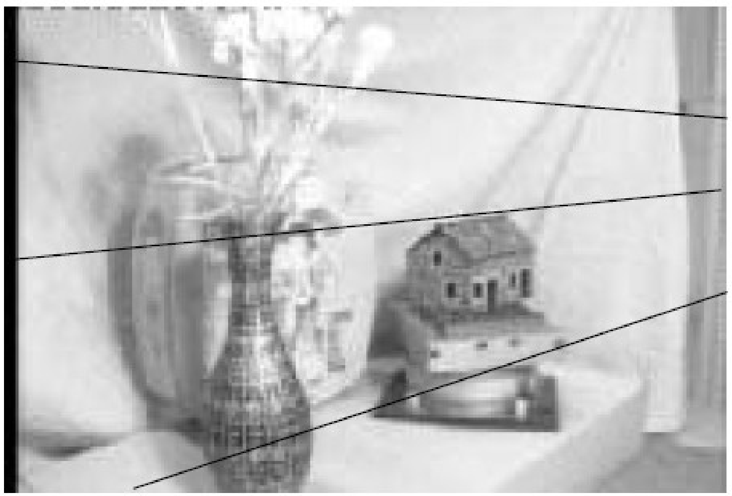
The Epipolar Geometry

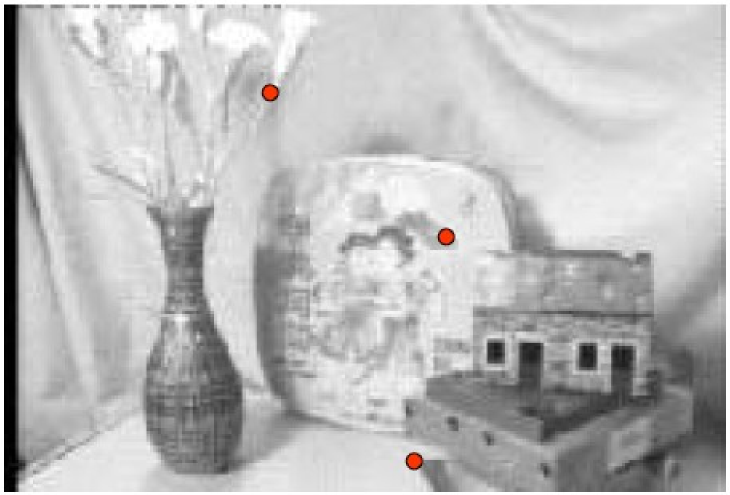


Epipolar Lines

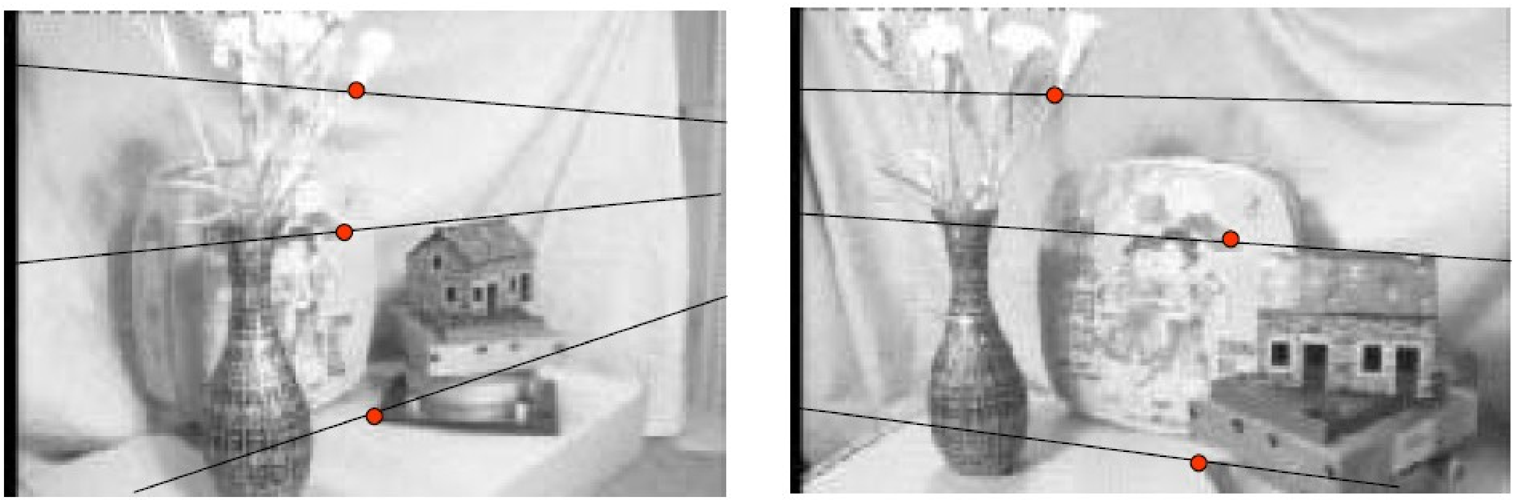


The Epipolar Geometry



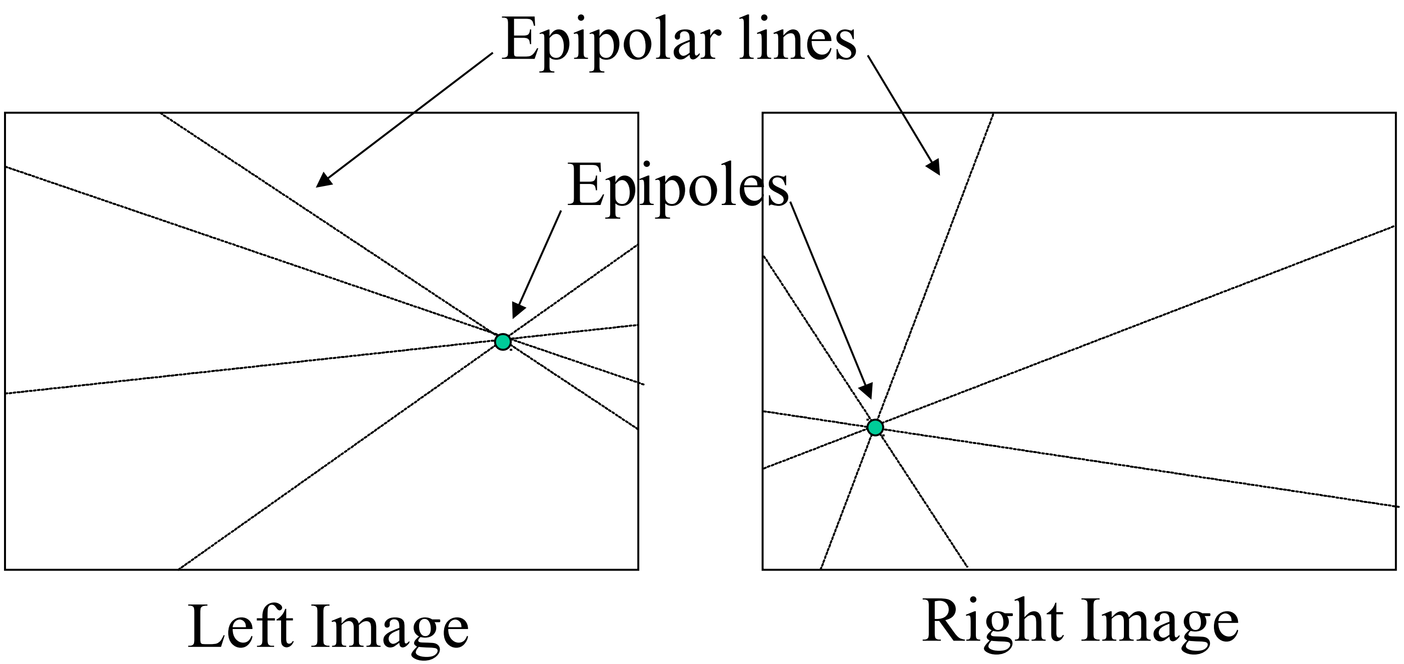


The Epipolar Geometry



Epipolar Geometry continued

– All of the epipolar lines in each image pass through the epipole in that image

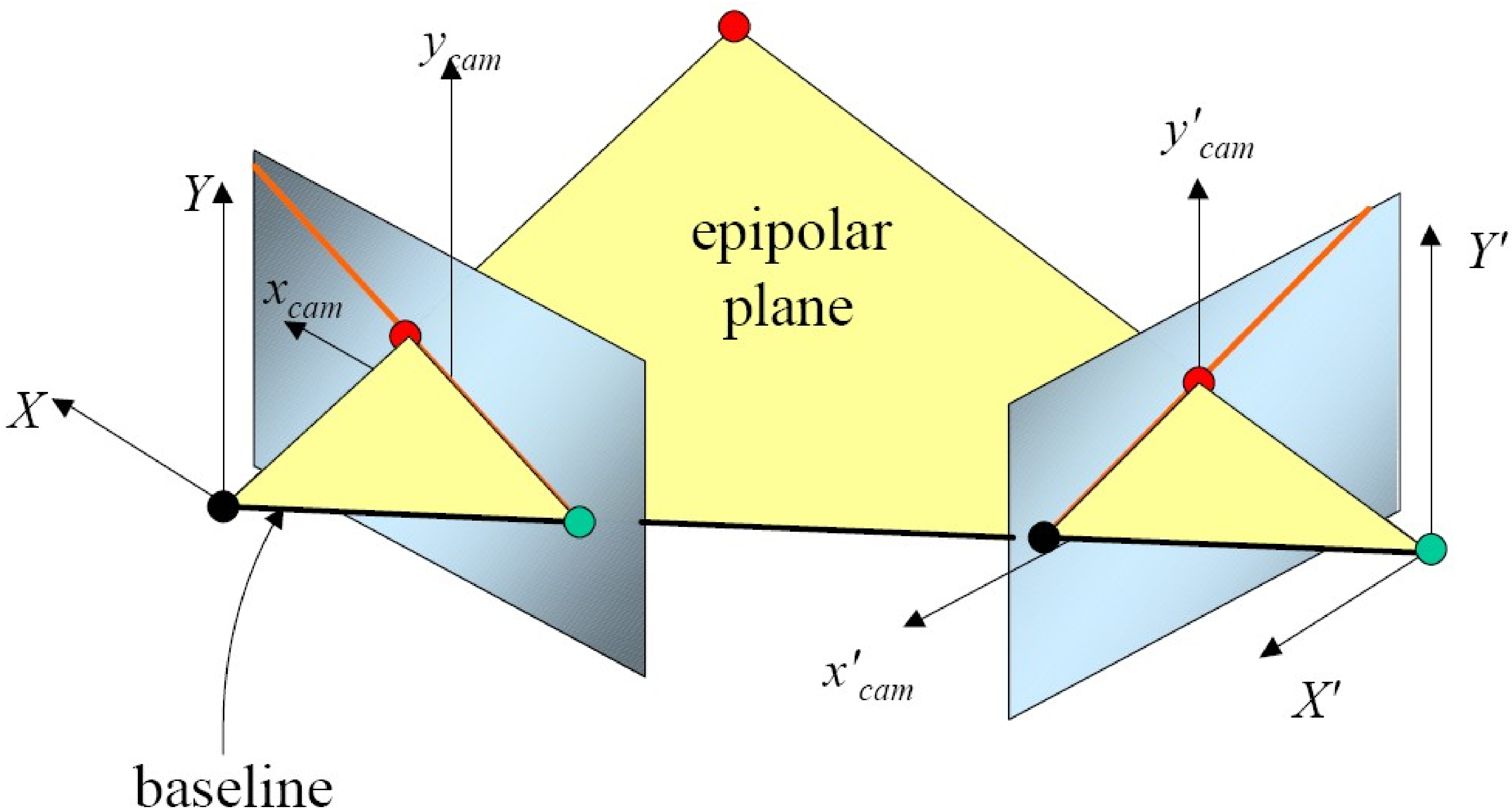


Consequences

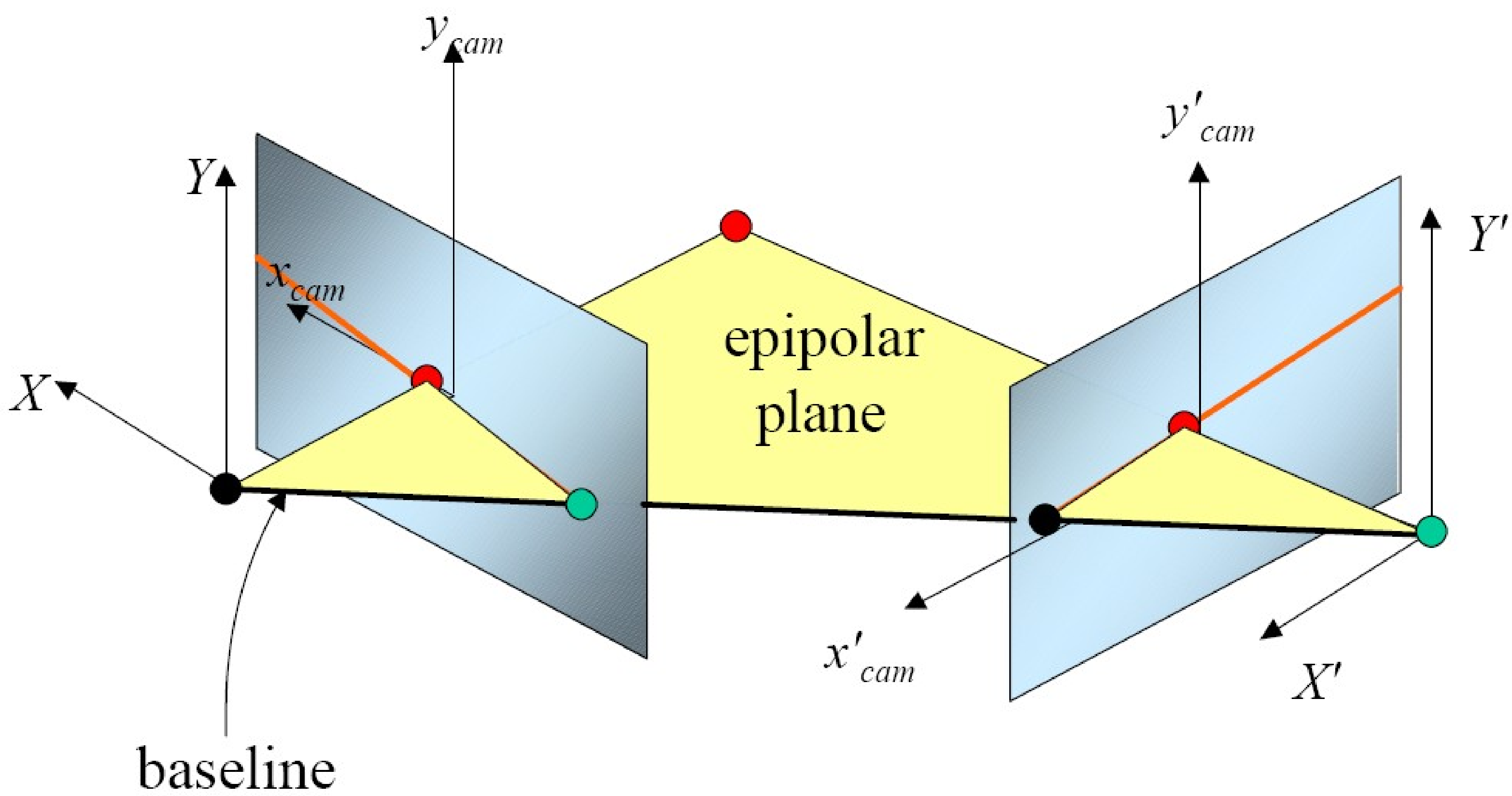
- The epipolar constraint
  - For every point observed in the left image we know that its correspondence must lie along the corresponding epipolar line in the right image
  - For every epipolar line in the left image there is a corresponding epipolar line in the right image
- This observation can substantially simplify the search for correspondences

Epipolar Geometry

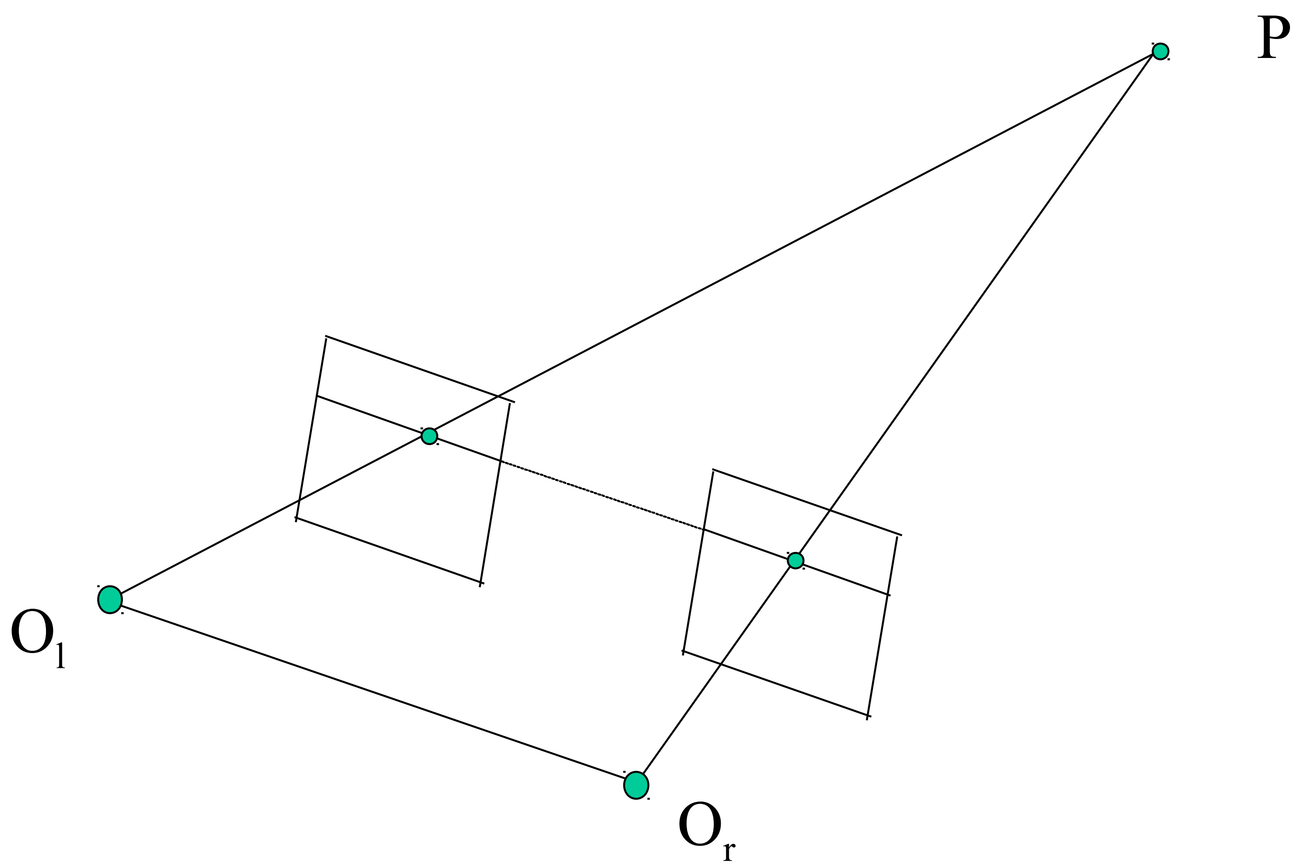
- Depending on the location of 3D points, the epipolar plane rotates about the baseline
  - The family is called epipolar pencil



Epipolar Geometry



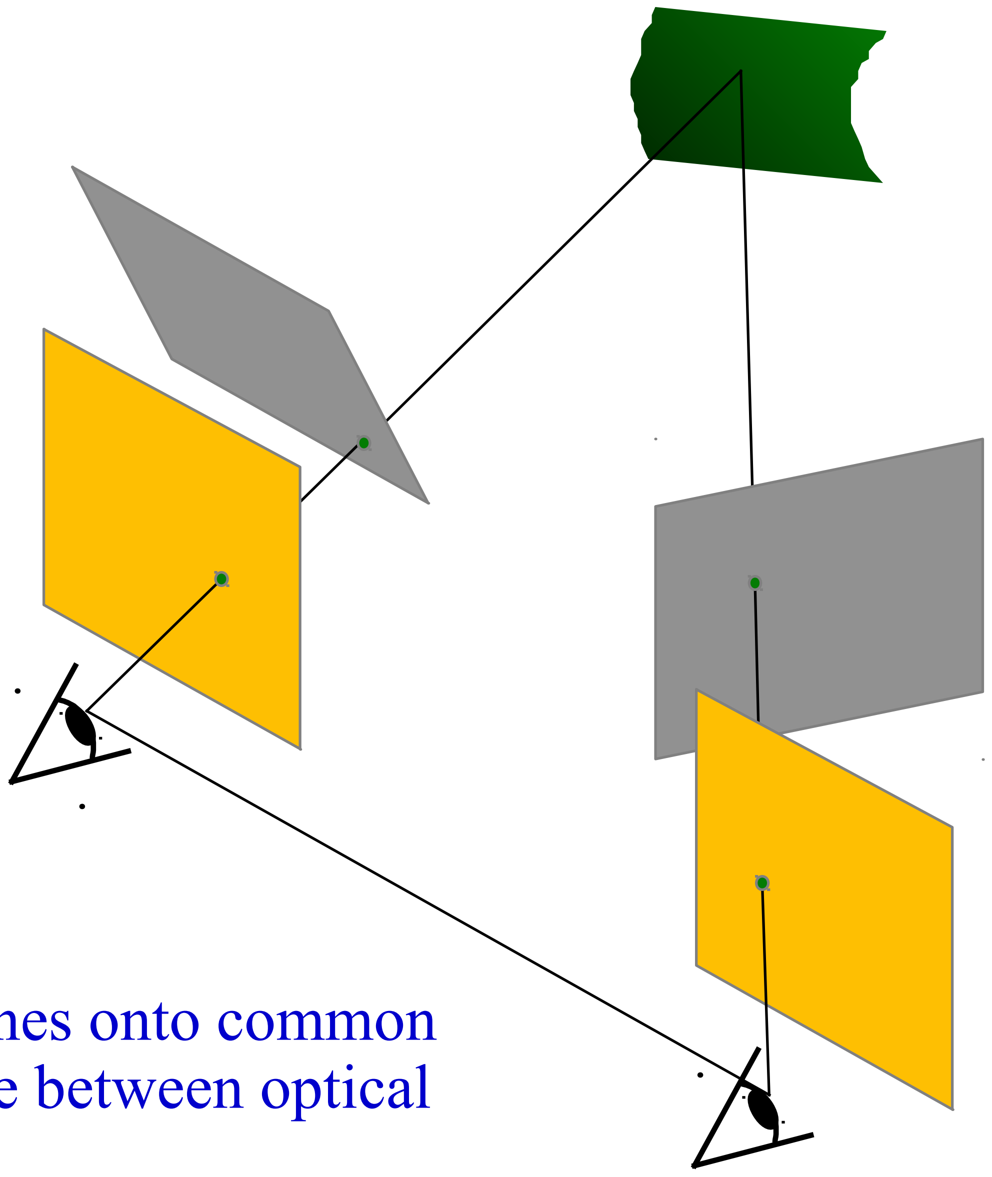
Special Case



Special Case

- In the special case of the stereo setup shown in the previous slide where the image planes are aligned with each other, the epipolar lines correspond to rows in the image
- That is the epipoles in both images are at infinity along the x axis.
- Note that it is often possible to rectify a stereo pair so that it appears to have this special structure.

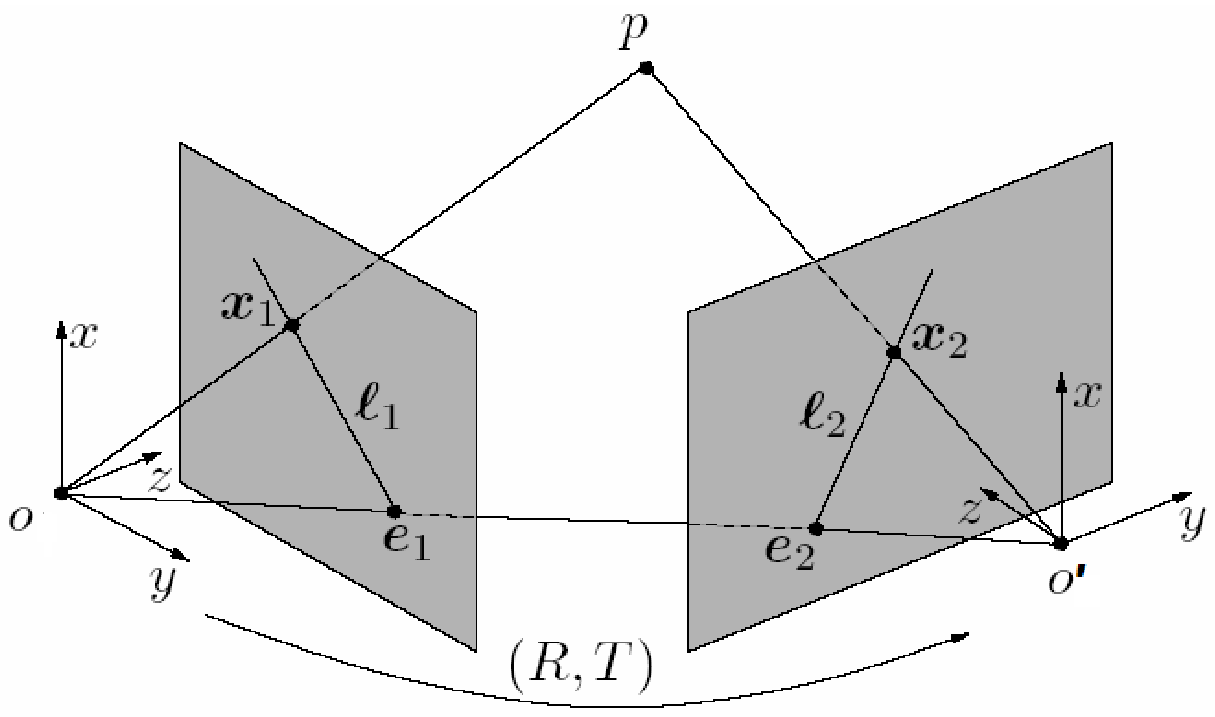
We can always achieve this geometry with image rectification



(Seitz)

- Image reprojection
  - reproject image planes onto common plane parallel to line between optical centers
- Notice, only focal point of camera really matters

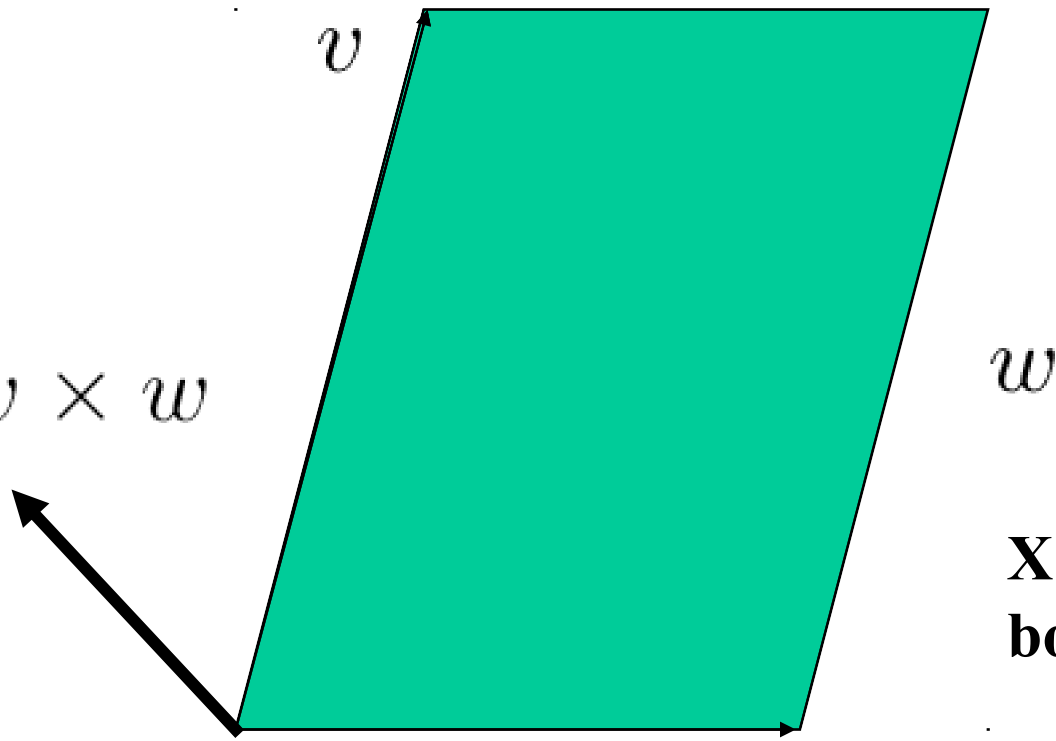
Epipolar Geometry



Rx_1: direction of vector OP,
T : direction of vector O'O
x2 : direction of vector O'P

These three vectors form a plane
Note: R, T specify the left camera's pose and position in the right camera's

Cross Product



$$x = v \times w$$

X is normal to both v and w

Skew-symmetric matrix

$$T = [t_1, t_2, t_3]$$

$$\hat{T} = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix}$$

Then, for any vector $x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$, we have

$$T \times x = \hat{T}x$$

Epipolar Geometry

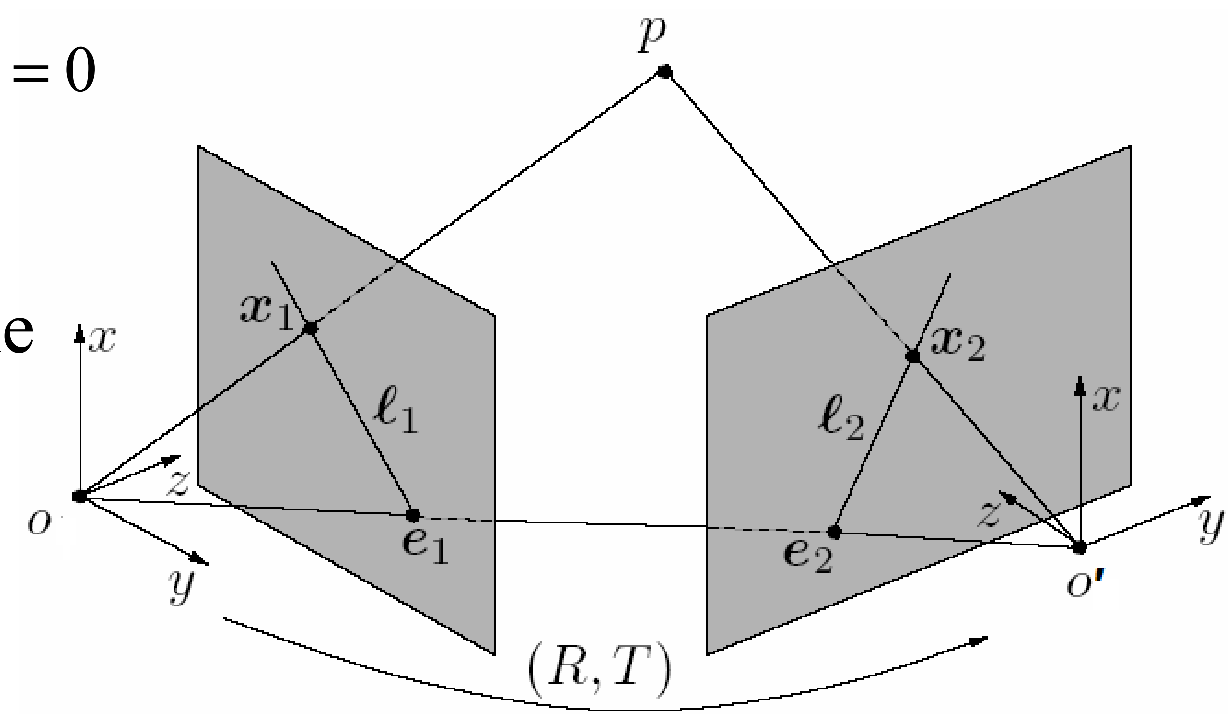
$T \times Rx_1$ gives the normal of the plane

Since x_2 is within the plane also, we have

$x_2^T T \times Rx_1 = 0$ and we can write it as

$$x_2^T \hat{T} R x_1 = 0$$

$E = \hat{T}R$ is called the essential matrix in the calibrated case



Epipolar Geometry

Rigid transformation between two cameras

$$\boldsymbol{X}_2 = R\boldsymbol{X}_1 + T.$$

$$\lambda_2 \boldsymbol{x}_2 = R\lambda_1 \boldsymbol{x}_1 + T.$$

Denote T^ as cross product T x:

$$\lambda_2 \widehat{T} \boldsymbol{x}_2 = \widehat{T} R \lambda_1 \boldsymbol{x}_1.$$

Epipolar Geometry

$$\lambda_2 \widehat{T} \boldsymbol{x}_2 = \widehat{T} R \lambda_1 \boldsymbol{x}_1. \qquad \widehat{T} \boldsymbol{x}_2 = T \times \boldsymbol{x}_2$$

$\langle \boldsymbol{x}_2, \widehat{T} \boldsymbol{x}_2 \rangle = {\boldsymbol{x}_2}^T \widehat{T} \boldsymbol{x}_2$ is zero

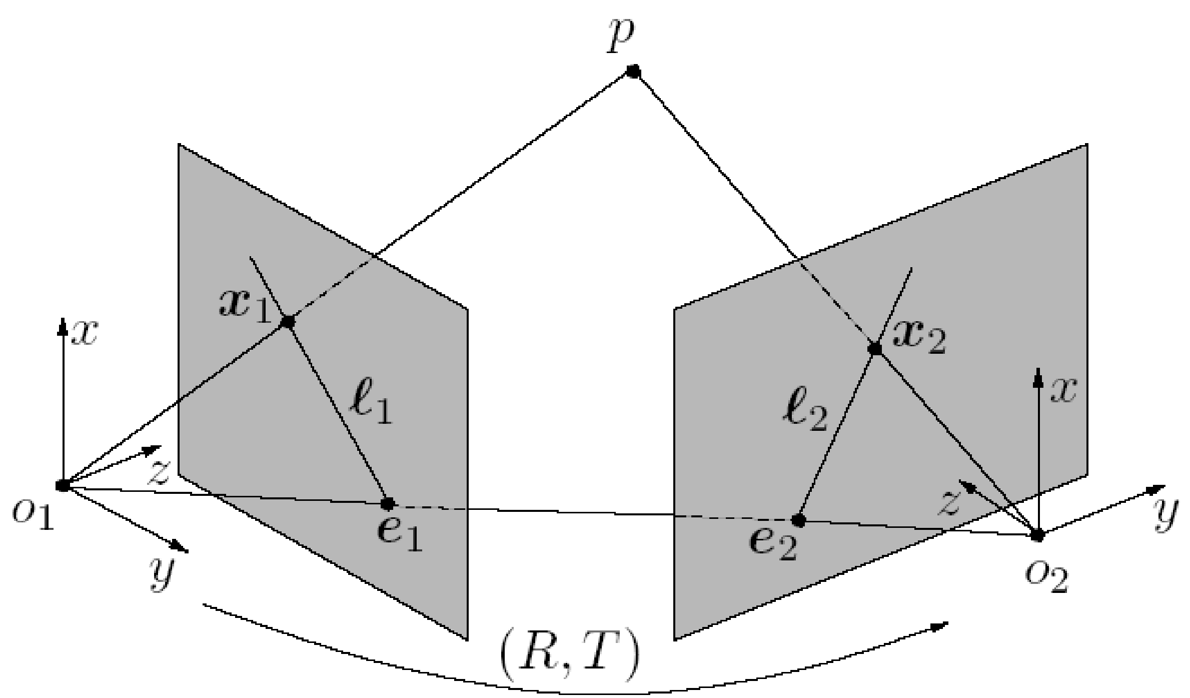
Therefore:

$$\langle \boldsymbol{x}_2, T \times R \boldsymbol{x}_1 \rangle = 0, \quad \text{or} \quad \boxed{\boldsymbol{x}_2^T \widehat{T} R \boldsymbol{x}_1 = 0.}$$

The two epipoles $\mathbf{e}_1, \mathbf{e}_2 \in \mathbb{R}^3$, *with respect to the first and second camera frames, respectively, are the left and right null spaces of* E, *respectively:*

$$\boldsymbol{e}_2^T E = 0, \quad E\boldsymbol{e}_1 = 0.$$

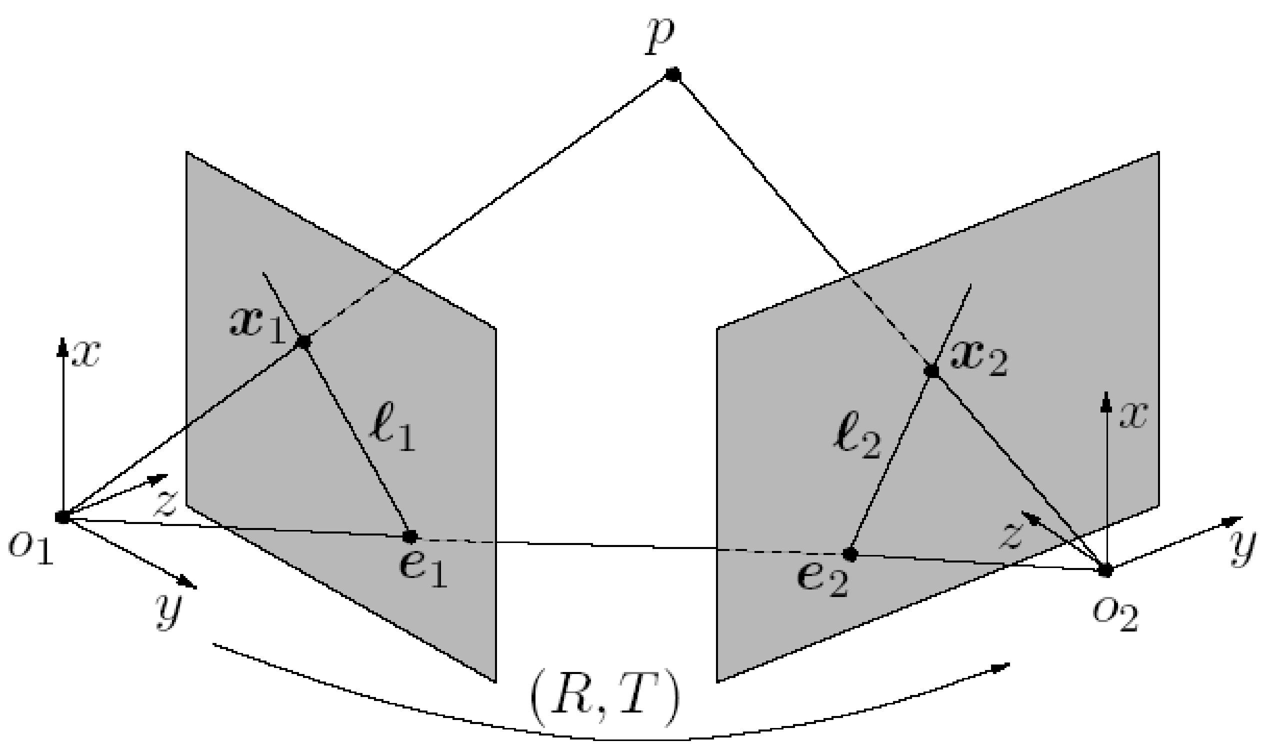
That is, $\mathbf{e}_2 \sim T$ *and* $\mathbf{e}_1 \sim R^T T$. *We recall that* $\sim$ *indicates equality up to a scalar factor.*



The epipolar lines $\boldsymbol{\ell}_1, \boldsymbol{\ell}_2 \in \mathbb{R}^3$ *associated with the two image points* $\boldsymbol{x}_1, \boldsymbol{x}_2$ *can be expressed as*

$$\boldsymbol{\ell}_2 \sim E\boldsymbol{x}_1, \quad \boldsymbol{\ell}_1 \sim E^T\boldsymbol{x}_2 \quad \in \mathbb{R}^3,$$

where $\boldsymbol{\ell}_1, \boldsymbol{\ell}_2$ *are in fact the normal vectors to the epipolar plane expressed with respect to the two camera frames, respectively.*



A Line in a Plane

- An epipolar line on an image plane can be described by the general equation for a line

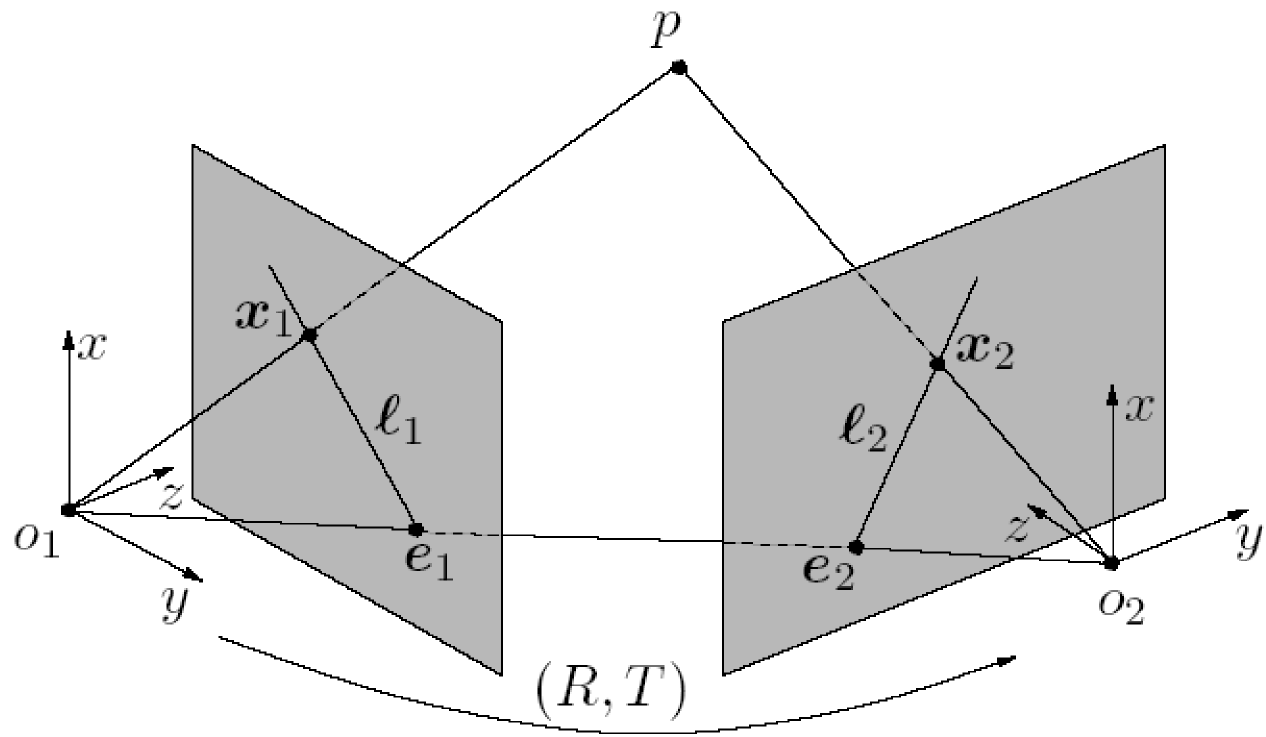
$$ax + by + c = 0$$

  - The coefficients of an epipolar line are given by

$$Ex_1 \text{ and } E^T x_2$$

In each image, both the image point and the epipole lie on the epipolar line

$$\boldsymbol{\ell}_i^T \mathbf{e}_i = 0, \quad \boldsymbol{\ell}_i^T \boldsymbol{x}_i = 0, \quad i = 1, 2.$$



Normalized 8 Point Algorithm

- Essential matrix **F** can be determined from 8 or more point correspondences.
- Procedure:
  1. Normalise image points.
  2. Determine $\mathbf{F}_{norm}$ for normalised points using least squares.
  3. Enforce singularity: replace $\mathbf{F}_{norm}$ by $\mathbf{F'}_{norm}$ such that $\det(\mathbf{F'}_{norm})=0$.
  4. Denormalise: determine **F** from $\mathbf{F'}_{norm}$.

Normalized 8 Point Algorithm

Define transformations $\mathbf{T}_{norm}$ and $\mathbf{T'}_{norm}$ each consisting of a translation and a scaling, that transform each set of image points so that their

- centroid is at the origin and
- the RMS distance from the origin is √2.

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \mathbf{T}_{norm} \tilde{\mathbf{x}}_{(i)cam} \qquad \begin{bmatrix} u'_i \\ v'_i \\ 1 \end{bmatrix} = \mathbf{T'}_{norm} \tilde{\mathbf{x}}'_{(i)cam}$$

here i denotes the i^{th} image point.

Normalized 8 Point Algorithm

- We need to translate our points so their centroid is at the origin.
- Then we want to scale them so their RMS is √ 2

- Construct $\mathbf{T}_{norm}$ from a translation and a scaling component:

$$\mathbf{T}_{norm} = \mathbf{T}_{scale}\,\mathbf{T}_{trans}$$

Normalized 8 Point Algorithm

- The translation component is

$$\mathbf{T}_{trans} = \begin{bmatrix} 1 & 0 & -\bar{x} \\ 0 & 1 & -\bar{y} \\ 0 & 0 & 1 \end{bmatrix}$$

- The scaling component is

$$\mathbf{T}_{scale} = \begin{bmatrix} \sqrt{2}/RMS & 0 & 0 \\ 0 & \sqrt{2}/RMS & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where $RMS = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left((x_i - \bar{x})^2 + (y_i - \bar{y})^2\right)}$

Normalized 8 Point Algorithm

Determining **F** from normalised coordinates.

$$\mathbf{x}'^{T}_{norm}\,\mathbf{F}\mathbf{x}_{norm} = 0$$

$$\begin{bmatrix} u' & v' & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 0$$

$$\begin{bmatrix} f_{11}u'+f_{21}v'+f_{31} & f_{12}u'+f_{22}v'+f_{32} & f_{13}u'+f_{23}v'+f_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 0$$

Normalized 8 Point Algorithm

$$\begin{bmatrix} f_{11}u'+f_{21}v'+f_{31} & f_{12}u'+f_{22}v'+f_{32} & f_{13}u'+f_{23}v'+f_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 0$$

$$f_{11}u'u + f_{21}v'u + f_{31}u + f_{12}u'v + f_{22}v'v + f_{32}v + f_{13}u' + f_{23}v' + f_{33} = 0$$

$$f_{11}u'u + f_{12}u'v + f_{13}u' + f_{21}v'u + f_{22}v'v + f_{23}v' + f_{31}u + f_{32}v + f_{33} = 0$$

Normalized 8 Point Algorithm

$$f_{11}u'u + f_{12}u'v + f_{13}u' + f_{21}v'u + f_{22}v'v + f_{23}v' + f_{31}u + f_{32}v + f_{33} = 0$$

$$\begin{bmatrix} u'u & u'v & u' & v'u & v'v & v' & u & v & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

Normalized 8 Point Algorithm

- For n point correspondences this becomes

$$\begin{bmatrix} u'_1 u_1 & u'_1 v_1 & u'_1 & v'_1 u_1 & v'_1 v_1 & v'_1 & u_1 & v_1 & 1 \\ u'_2 u_2 & u'_2 v_2 & u'_2 & v'_2 u_2 & v'_2 v_2 & v'_2 & u_2 & v_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u'_n u_n & u'_n v_n & u'_n & v'_n u_n & v'_n v_n & v'_n & u_n & v_n & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

Normalized 8 Point Algorithm

Least Squares Solution

- Form a vector $\mathbf{f}_{norm}$ containing the elements of $\mathbf{F}_{norm}$

$$\mathbf{f}_{norm} = \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix}$$

- Where the elements are indexed as follows:

$$\mathbf{F}_{norm} = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}$$

Normalized 8 Point Algorithm

- Find least squares solution of

$$\mathbf{A}\mathbf{f}_{norm} = \mathbf{0}, \qquad \text{for} \quad \mathbf{f}_{norm} \neq \mathbf{0}$$

where **A** is constructed from the normalised image points

$$\mathbf{A} = \begin{bmatrix} u'_1 u_1 & u'_1 v_1 & u'_1 & v'_1 u_1 & v'_1 v_1 & v'_1 & u_1 & v_1 & 1 \\ u'_2 u_2 & u'_2 v_2 & u'_2 & v'_2 u_2 & v'_2 v_2 & v'_2 & u_2 & v_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u'_n u_n & u'_n v_n & u'_n & v'_n u_n & v'_n v_n & v'_n & u_n & v_n & 1 \end{bmatrix}$$

- $\mathbf{f}_{norm}$ can only be determined up to a scale.

Normalized 8 Point Algorithm

Enforcing Singularity

- Let $\mathbf{F}_{norm} = \mathbf{UDV}^{\mathrm{T}}$ be the SVD of $\mathbf{F}_{norm}$

$$\mathbf{F}_{norm} = \mathbf{U}\begin{bmatrix} D_1 & 0 & 0 \\ 0 & D_2 & 0 \\ 0 & 0 & D_3 \end{bmatrix}\mathbf{V}^{\mathrm{T}}$$

where $D_1 > D_2 > D_3$.

Normalized 8 Point Algorithm

- Define $\mathbf{F}'_{norm}$

$$\mathbf{F}'_{norm} = \mathbf{U} \begin{bmatrix} D_1 & 0 & 0 \\ 0 & D_2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{V}^{\mathrm{T}}$$

- $\mathbf{F}'_{norm}$ is rank 2 as required.

Normalized 8 Point Algorithm

Denormalisation

- Define the Fundamental Matrix **F**:

$$\mathbf{F} = (\mathbf{T}'_{norm})^{\mathrm{T}}\,\mathbf{F}'_{norm}\,\mathbf{T}_{norm}$$

Normalized 8 Point Algorithm

- Find least squares solution of

$$\mathbf{A}\mathbf{f}_{norm} = \mathbf{0}, \qquad \text{for} \quad \mathbf{f}_{norm} \neq \mathbf{0}$$

where **A** is constructed from the normalised image points

$$\mathbf{A} = \begin{bmatrix} u'_1 u_1 & u'_1 v_1 & u'_1 & v'_1 u_1 & v'_1 v_1 & v'_1 & u_1 & v_1 & 1 \\ u'_2 u_2 & u'_2 v_2 & u'_2 & v'_2 u_2 & v'_2 v_2 & v'_2 & u_2 & v_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u'_n u_n & u'_n v_n & u'_n & v'_n u_n & v'_n v_n & v'_n & u_n & v_n & 1 \end{bmatrix}$$

- $\mathbf{f}_{norm}$ can only be determined up to a scale.

Normalized 8 Point Algorithm

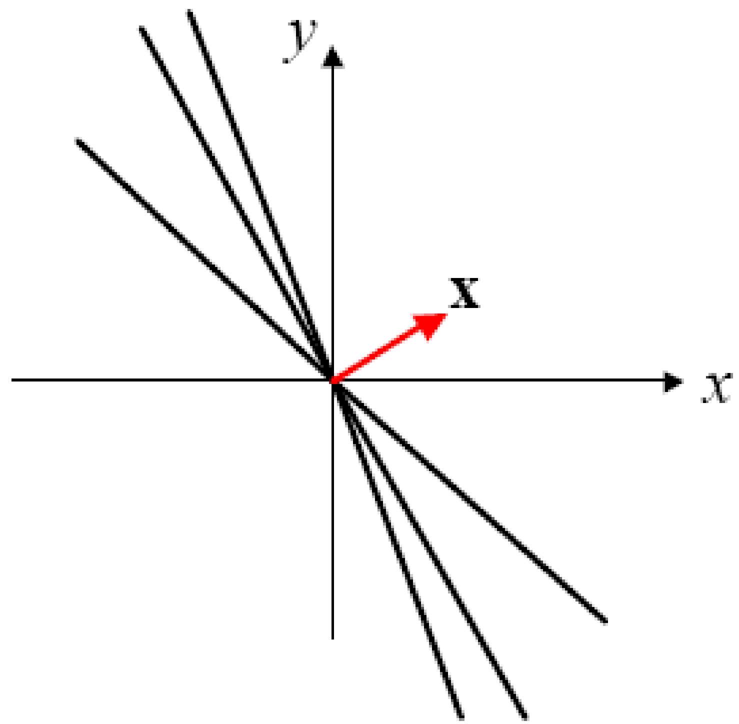
- The solution is to choose f_{norm} to be the eigenvector associated with the smallest eigenvalue of $A^T A$
- It can also be obtained from the singular value decomposition (SVD) of A

$$A = USV^T$$

  - Choose f_{norm} to be the last column of V (corresponding to the smallest singular value)

Why the Eigenvector of the Smallest Eigenvalue of $A^T A$?

- Note that we can only determine f_{norm} up to a scale
  - In the 2D case, this means that we find a direction that is most perpendicular to all the n lines



Why the Eigenvector of the Smallest Eigenvalue of $A^T A$?

- Since the scaling is arbitrary, we fix the scale by
$$\|x\| = 1$$
- We define $\varepsilon = Ax$
  - Since we want Ax to be zero (but we can not in general), we can do the best we can by choosing x to minimize $||Ax||^2$

Why the Eigenvector of the Smallest Eigenvalue of A^TA?

- The problem is a constrained optimization one

$$x^* = \arg\min_x (Ax)^T (Ax) = \arg\min_x x^T A^T Ax,$$

$$x^T x = 1 \ \ (\text{which is same as } \|\mathrm{x}\| = 1)$$

- We use the method of Lagrange multipliers and define V

$$V = \mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x} + \lambda(1 - \mathbf{x}^T\mathbf{x})$$

  – Here λ is a Lagrange multiplier

Why the Eigenvector of the Smallest Eigenvalue of $A^T A$?

- To find the minima of V, we take the derivative with respect to x and λ, and set them to zero

$$dV/d\mathbf{x} = 2\mathbf{A}^T\mathbf{A}\mathbf{x} - 2\lambda\mathbf{x} = 0$$

$$\Rightarrow \mathbf{A}^T\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$$

$$dV / d\lambda = (1 - x^T x) = 0$$

  - The solution must be an eigenvector (of unit length) of $A^T A$

Why the Eigenvector of the Smallest Eigenvalue of A^TA?

- Which one?
  - Note that we want to minimize $\mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x}$
  - Suppose x is the unit length eigenvector associated with eigenvalue λ_i

$$x^T A^T A x = \lambda_i$$

  - To minimize $\mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x}$, we want to choose the eigenvector of A^TA associated with the smallest eigenvalue

Why Singular Value Decomposition?

- Let us look at the singular value decomposition of A
  - For any n x m matrix A, there exist unitary matrices U (n x n) and V (m x m) such that

$$\mathbf{U} = [\mathbf{u}_1 | \mathbf{u}_2 | \ldots | \mathbf{u}_n] \longleftarrow n \times n \text{ matrix} \qquad U^{-1} = U^T$$

$$\mathbf{V} = [\mathbf{v}_1 | \mathbf{v}_2 | \ldots | \mathbf{v}_m] \longleftarrow m \times m \text{ matrix} \qquad V^{-1} = V^T$$

$$\mathbf{A} = \mathbf{USV}^{\mathrm{T}}, \text{ where } \mathbf{S} = \begin{bmatrix} \mathbf{S}_1 & 0 \\ 0 & 0 \end{bmatrix}, \quad \mathbf{S}_1 = \begin{bmatrix} s_1 & 0 & \ldots & 0 \\ 0 & s_2 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & s_p \end{bmatrix}$$

and $s_1 \geq s_2 \geq \ldots \geq s_\mathrm{p} \geq 0$, $p = \min\{n,m\}$

Why Singular Value Decomposition?

- We have the following

$$A = USV^T \quad AV = US \quad \mathrm{Av}_{\mathrm{i}} = s_i u_i$$

$$A^T = VSU^T \quad A^T U = VS \quad A^T u_i = s_i v_i$$

$$A^T A v_i = A^T s_i u_i = s_i A^T u_i = s_i^2 v_i$$

$$AA^T u_i = A s_i v_i = s_i A v_i = s_i^2 u_i$$

  - s_i^2 is an eigenvalue of $\mathbf{AA}^\mathrm{T}$ or $\mathbf{A}^\mathrm{T}\mathbf{A}$,
  - $\mathbf{u}_i$ is an eigenvector of $\mathbf{AA}^\mathrm{T}$ and
  - $\mathbf{v}_i$ is an eigenvector of $\mathbf{A}^\mathrm{T}\mathbf{A}$.

Why Singular Value Decomposition?

- $\mathbf{V} = [\mathbf{v}_1|\mathbf{v}_2|\ldots|\mathbf{v}_m]$
- is a matrix of eigenvectors of $\mathbf{A}^T\mathbf{A}$ with associated eigenvalues s_i^2. The eigenvector corresponding to the smallest eigenvalue of $\mathbf{A}^T\mathbf{A}$ is $\mathbf{v}_m$.
- Hence the non-zero $\mathbf{x}$ that minimises

$$\mathbf{A}\mathbf{x} = \mathbf{0}$$

is $\mathbf{x} = \mathbf{v}_m$.

Normalized 8 Point Algorithm

Enforcing Singularity

- Let $\mathbf{F}_{norm} = \mathbf{UDV}^{\mathrm{T}}$ be the SVD of $\mathbf{F}_{norm}$

$$\mathbf{F}_{norm} = \mathbf{U}\begin{bmatrix} D_1 & 0 & 0 \\ 0 & D_2 & 0 \\ 0 & 0 & D_3 \end{bmatrix}\mathbf{V}^{\mathrm{T}}$$

where $D_1 > D_2 > D_3$.

Normalized 8 Point Algorithm

- Define $\mathbf{F'}_{norm}$

$$\mathbf{F'}_{norm} = \mathbf{U} \begin{bmatrix} D_1 & 0 & 0 \\ 0 & D_2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{V}^{\mathrm{T}}$$

- $\mathbf{F'}_{norm}$ is rank 2 as required.

Why?

Normalized 8 Point Algorithm

Denormalisation

- Define the Fundamental Matrix **F**:

$$\mathbf{F} = (\mathbf{T'}_{norm})^{\mathrm{T}}\, \mathbf{F'}_{norm}\, \mathbf{T}_{norm}$$

Estimating Essential Matrix

- Note that in the calibrated case, we assume that the intrinsic camera parameters are known, we can compute the essential matrix from F by

$$\mathbf{E} = \mathbf{K'}^{T}\mathbf{F}\mathbf{K}$$

  - Where K and K' are the camera parameters for the left and right cameras

Estimating R and T from E

- We need to know the relative positions of two cameras
  - We do singular value decomposition of E

$$\mathbf{E} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}}$$

  - Then rotation and translation are given by

$$\mathbf{R} = \mathbf{U}\mathbf{W}\mathbf{V}^{\mathrm{T}} \text{ or } \mathbf{U}\mathbf{W}^{\mathrm{T}}\mathbf{V}^{\mathrm{T}}$$

$$\mathbf{t} = \mathbf{u}_3 \text{ or } -\mathbf{u}_3$$

where $\mathbf{u}_3$ is the last column of $\mathbf{U}$, and $\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Why?

- Note that E has a special form $E = \hat{T}R$

$$E = USV^T = \hat{\mathrm{T}}\mathrm{R} \quad \Rightarrow \quad ER^T = USV^T R^T = US(RV)^T = \hat{T}$$

  - Which means U, S, and RV are SVD of $\hat{T}$
  - We thus have (according to the properties of SVD)

$$\hat{T}(Rv_1) = s_1 u_1 \quad \text{and} \quad \hat{T}(Rv_2) = s_2 u_2$$

$$u_1 = \pm R v_2, \quad u_2 = \mp R v_1 \quad \text{and} \quad u_3 = R v_3$$

$$\begin{bmatrix} u_1 & u_2 & u_3 \end{bmatrix} = R\begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}\begin{bmatrix} 0 & \mp 1 & 0 \\ \pm 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} u_1 & u_2 & u_3 \end{bmatrix}\begin{bmatrix} 0 & \mp 1 & 0 \\ \pm 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1}\begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}^T$$

$$= U\begin{bmatrix} 0 & \pm 1 & 0 \\ \mp 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}V^T$$

How about T?

- The direction of the translation vector is given by u_3
  - But we can not recover the true magnitude of the translation vector because we can only recover E up to a scale

A Numerical Example

$$R=\begin{bmatrix}\cos(\pi/3) & 0 & \sin(\pi/3)\\ 0 & 1 & 0\\ -\sin(\pi/3) & 0 & \cos(\pi/3)\end{bmatrix}=\begin{bmatrix}1/2 & 0 & \sqrt{3}/2\\ 0 & 1 & 0\\ -\sqrt{3}/2 & 0 & 1/2\end{bmatrix}\qquad T=\begin{bmatrix}3\\2\\1\end{bmatrix}$$

```
theta=pi/3;
R=[cos(theta) 0 -sin(theta)
   0          1    0
   sin(theta) 0 cos(theta)];
T=[3 2 1]';
E=skew(T)*R;
[U,S,V]=svd(E);
 if det(U) < 0,
    U(:,3)=-U(:,3);
 end
```

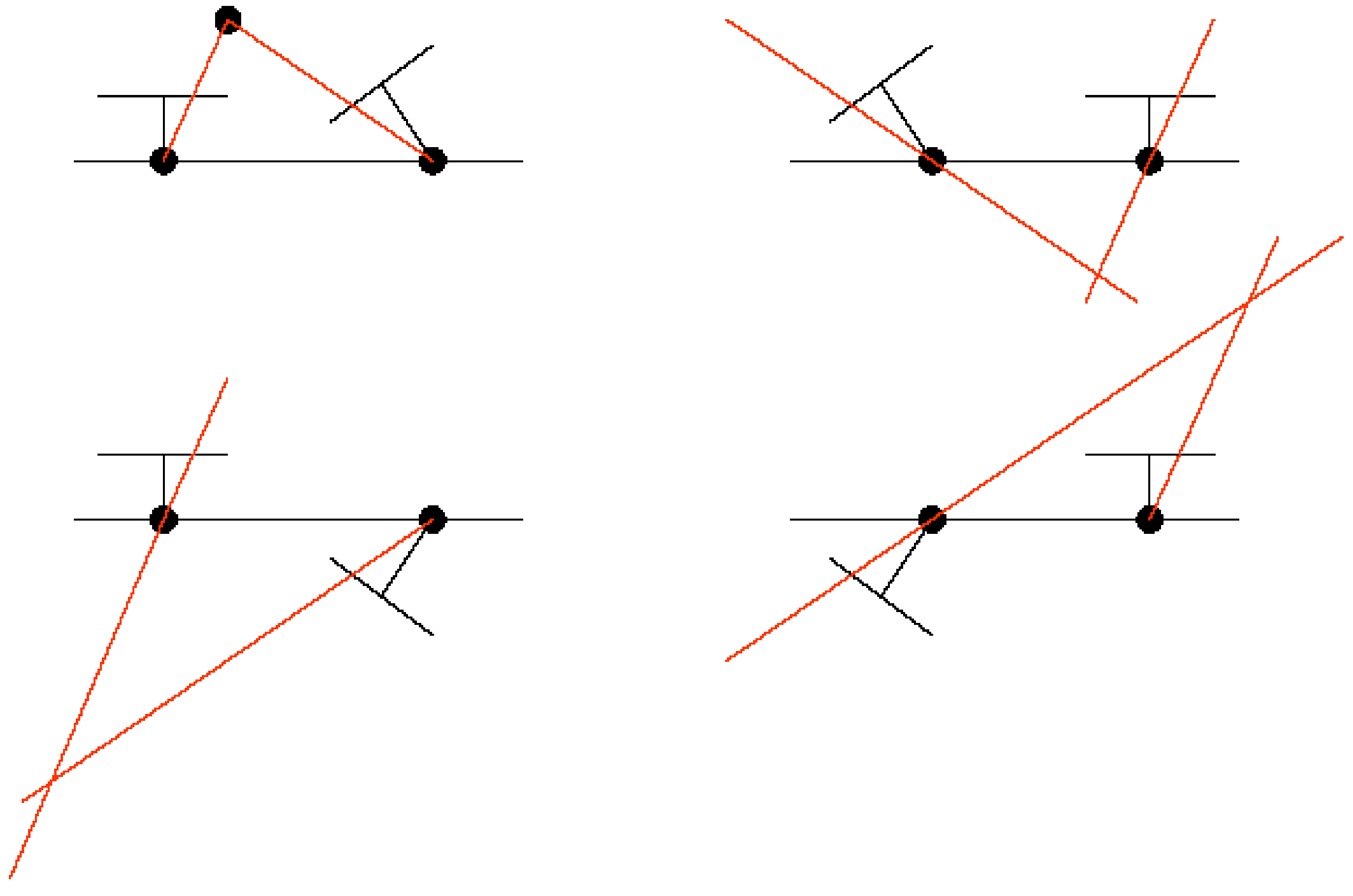
```
if det(V) < 0,
   V(:,3)=-V(:,3);
end
W=[0 -1 0; 1 0 0; 0 0 1];
R1=U*W*V';
R11=U*W'*V';
T1=U(:,3)*norm(T);
```

Estimating R and T from E

- There are four possible solutions $\mathbf{P}_{cam} = [\mathbf{R} \mid \mathbf{t}]$

1. $\mathbf{P}_{cam} = [\mathbf{UWV}^{\mathrm{T}} \mid \mathbf{u}_3]$
2. $\mathbf{P}_{cam} = [\mathbf{UWV}^{\mathrm{T}} \mid -\mathbf{u}_3]$
3. $\mathbf{P}_{cam} = [\mathbf{UW}^{\mathrm{T}}\mathbf{V}^{\mathrm{T}} \mid \mathbf{u}_3]$
4. $\mathbf{P}_{cam} = [\mathbf{UW}^{\mathrm{T}}\mathbf{V}^{\mathrm{T}} \mid -\mathbf{u}_3]$

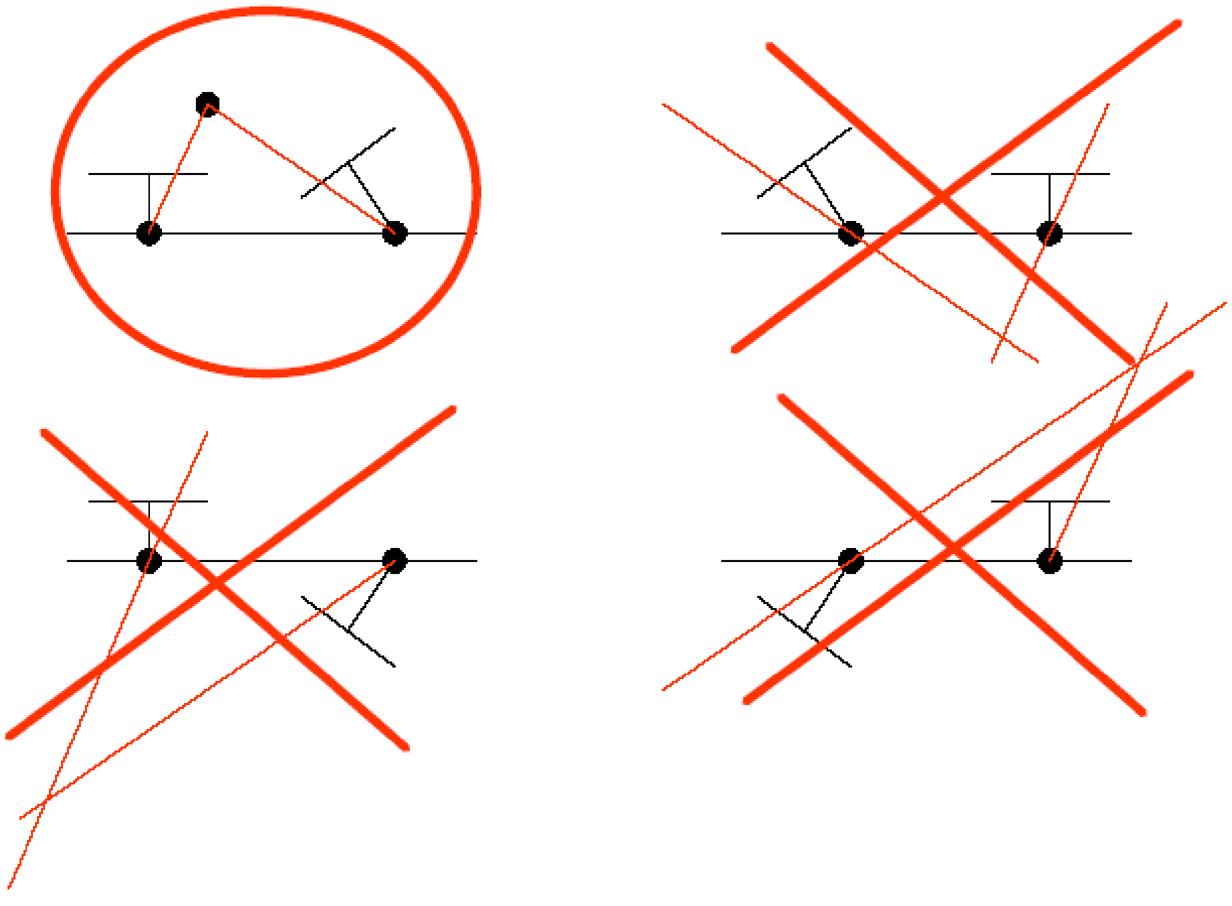
Estimating R and T from E



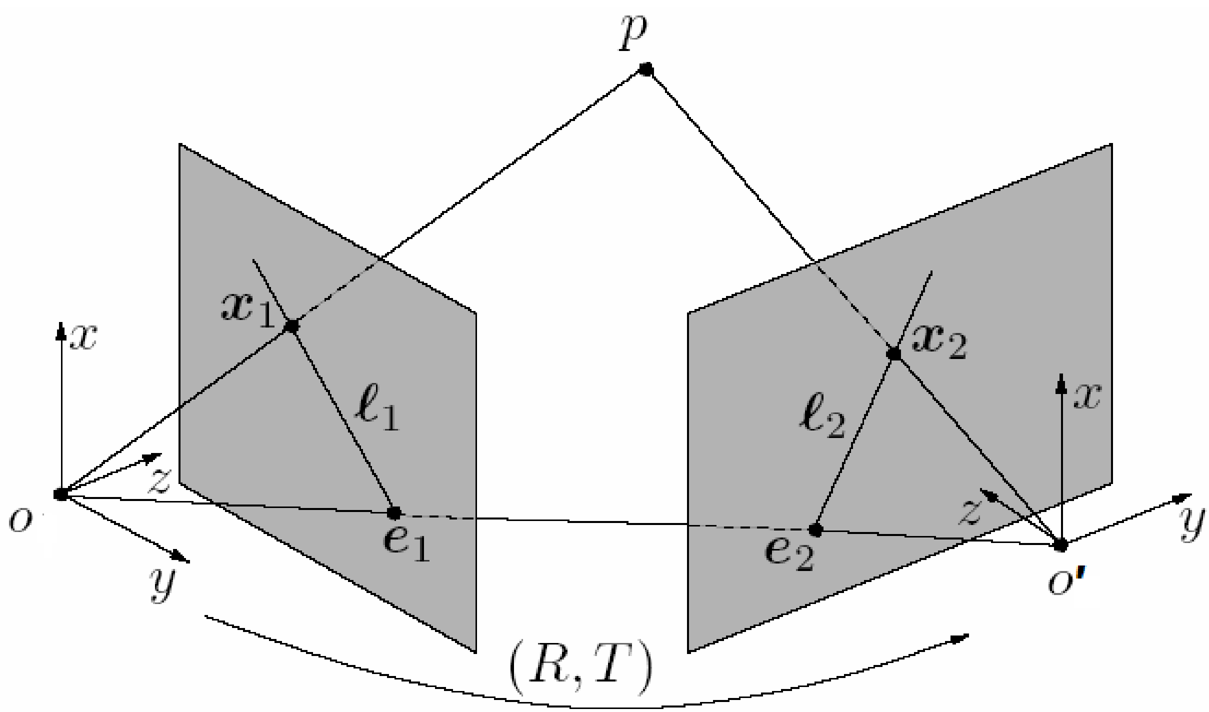
How to Choose the Correct One

- Positive depth constraints
  - For the correct pair, all the data points should be in front of the both cameras
  - For each pair, we can compute the 3D points of each correspondence, and check if the Z (depth) is positive in both camera coordinates

Estimating R and T from E



3D Reconstruction from a Pair of Correspondence



Rx_1 : direction of vector OP,

T : direction of vector O'O

x2 : direction of vector O'P

These three vectors form a plane

3D Reconstruction from a Pair of Correspondence

- Let P=[X Y Z W]^T be the 3D point in the left camera's coordinate system
  - Let $[x_2, y_2, 1]^T$ be the corresponding point in the normalized image plane of the right camera
  - Let $[x_1, y_1, 1]^T$ be the corresponding point in the normalized image plane of the left camera
  - For the left camera, we have

$$\begin{cases} x_1 = X / Z \\ y_1 = Y / Z \end{cases} \Rightarrow \begin{cases} -1X + 0Y + x_1 Z = 0 \\ 0X - 1Y + y_1 Z = 0 \end{cases}$$

3D Reconstruction from a Pair of Correspondence

- For the right camera, the 3D point in its coordinate system is

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ W' \end{bmatrix} = \begin{bmatrix} R & T \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} \Rightarrow \begin{cases} X' = R_{11}X + R_{12}Y + R_{13}Z + T_1W \\ Y' = R_{21}X + R_{22}Y + R_{23}Z + T_2W \\ Z' = R_{31}X + R_{32}Y + R_{33}Z + T_3W \end{cases}$$

  - Similar to the left camera, we have

$$\begin{cases} x_2 = X'/Z' \\ y_2 = Y'/Z' \end{cases} \Rightarrow \begin{cases} -1X' + 0Y' + x_2Z' = 0 \\ 0X' - 1Y' + y_2Z' = 0 \end{cases}$$

  - Thus

$$\begin{cases} (-R_{11} + x_2R_{31})X + (-R_{12} + x_2R_{32})Y + (-R_{13} + x_2R_{33})Z + (-T_1 + x_2T_3)W = 0 \\ (-R_{21} + y_2R_{31})X + (-R_{22} + y_2R_{32})Y + (-R_{23} + y_2R_{33})Z + (-T_2 + y_2T_3)W = 0 \end{cases}$$

3D Reconstruction from a Pair of Correspondence

- Thus, we have a linear problem to solve

$$A = \begin{bmatrix} -1 & 0 & x_1 & 0 \\ 0 & -1 & y_1 & 0 \\ -R_{11} + x_2 R_{31} & -R_{12} + x_2 R_{32} & -R_{13} + x_2 R_{33} & -T_1 + x_2 T_3 \\ -R_{21} + y_2 R_{31} & -R_{22} + y_2 R_{32} & -R_{23} + y_2 R_{33} & -T_2 + y_2 T_3 \end{bmatrix}, \quad \mathrm{A} \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = 0$$

  - Due to measurement noise, we may not be able to find an exact solution
  - We compute the SVD of the A matrix and take the last column of V to be the best solution

3D Reconstruction from a Pair of Correspondence

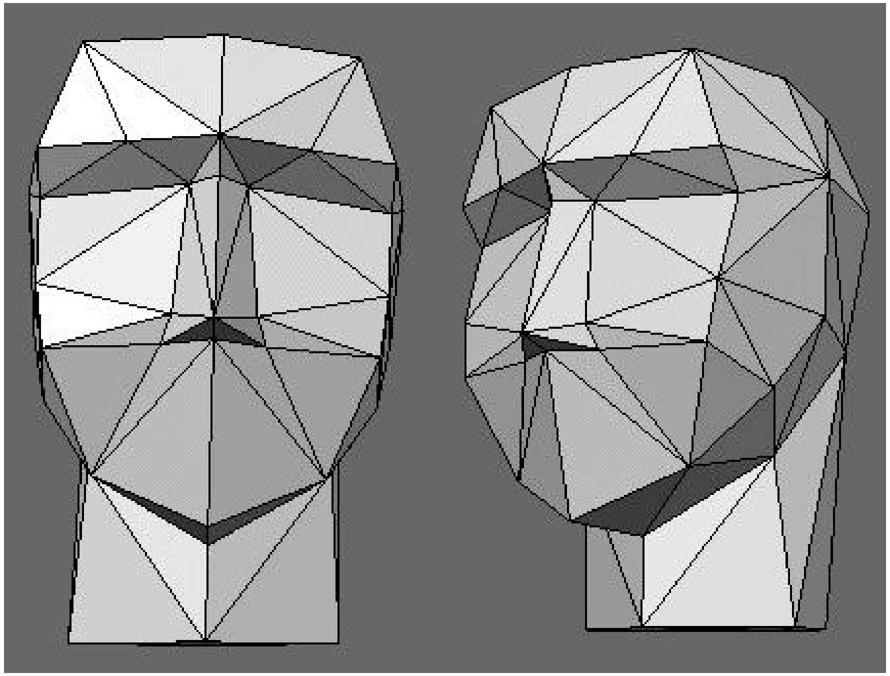
- What is the depth of the point in the left camera?
- What is the depth of the point in the right camera?
- To pick the correct solution among the four possible ones, pick one test point and calculate its depths in both cameras

Putting Everything Together: A Complete Example

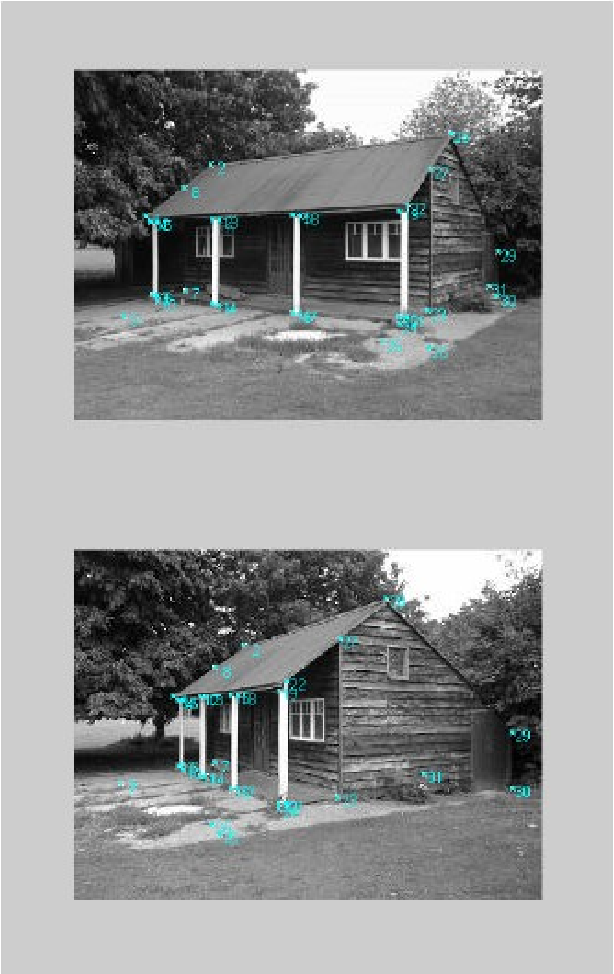
- Here I use a synthetic example to illustrate how all the steps work
- The object – a cube-like wire frame
- The cameras
- The projective matrices for both cameras
- 3D Reconstruction
  - Corresponding points
  - Fundamental matrices
  - Essential matrices, R, and T

3D Models

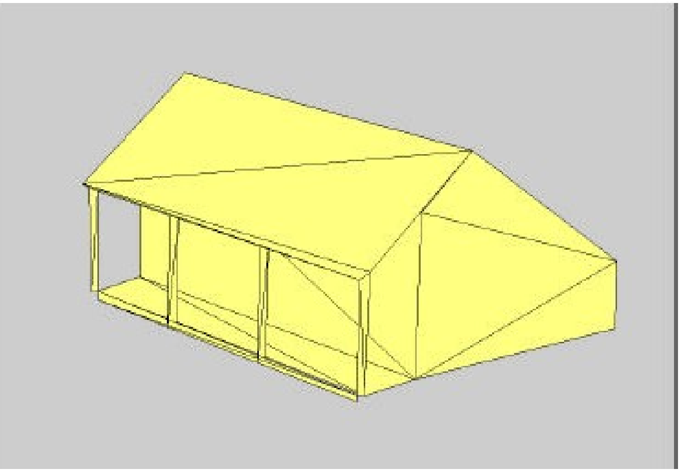
- To generate a 3D model from 3D points (called point clouds), we first need to form triangles to approximate the underlying surface



3D Models – cont.

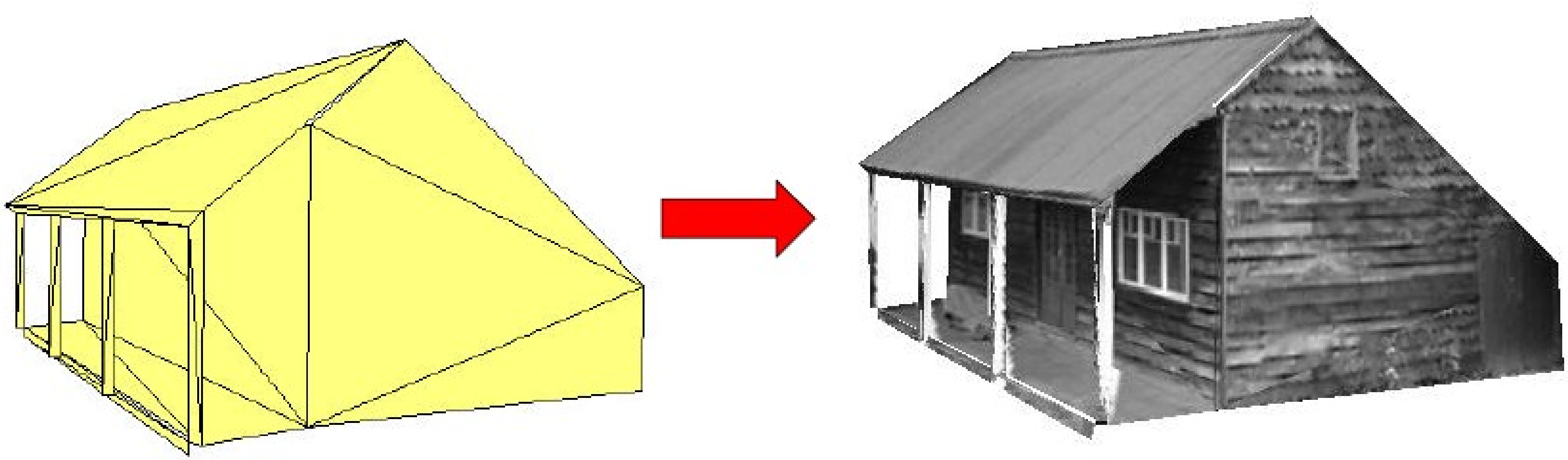


- Identifying which points form triangular planar regions enables us to build a polygon model of the scene.



Texture Mapping

- To make the model more realistic, we can map textures from the original images on the planar surfaces



Texture Mapping

- In a VRML file, this can be done easily
  - For each triangle,
    - Specify an image using “Texture ImageTexture”
    - Specify the texture coordinates using “texCooord TextureCoordinate”
    - Specify the correspondence between the points for the vertices and the texture coordinates using texCoordIndex

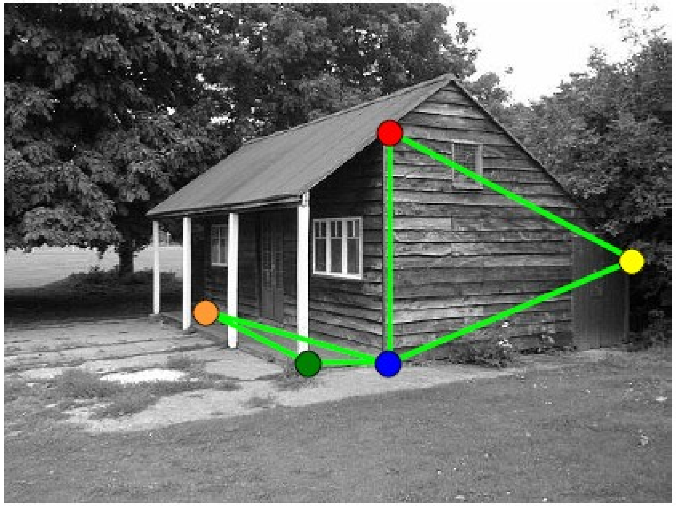
A VRML Example

```
Shape {
  appearance Appearance {
    texture ImageTexture {
      url "IS3045large.jpg"
    }
  }
  geometry IndexedFaceSet {
    solid FALSE
    coord USE MYPOINTS
    coordIndex [
        0, 2, 3, -1 ]
    texCoord  TextureCoordinate {
      point [
          0.00    0.00,
          1.00    0.00,
          1.00    1.00]
    }
    texCoordIndex [ 0, 1, 2, -1 ]
  }
}
```

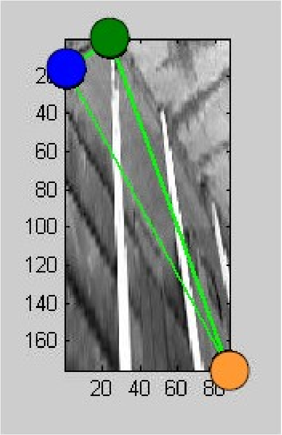
```
#VRML V2.0 utf8

Background {
    skyColor    [0.9, 0.95, 1]
}

DEF MYPOINTS Coordinate {
  point [
          0.24    0.26  10.81,
          0.28   -0.46  10.38,
         -0.45    0.11  10.51,
         -0.35   -0.46  10.18,
          0.35    0.46  10.16,
          0.45   -0.11   9.83,
         -0.28    0.46   9.96,
         -0.18   -0.11   9.64 ]
}
```

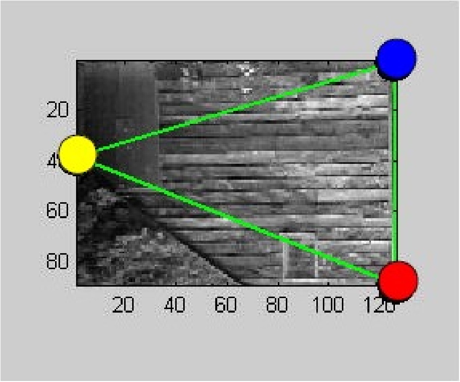
Texture Mapping – cont.

- However, the texture mapping in VRML assumes that the texture image is taken from a front-parallel perspective



original image

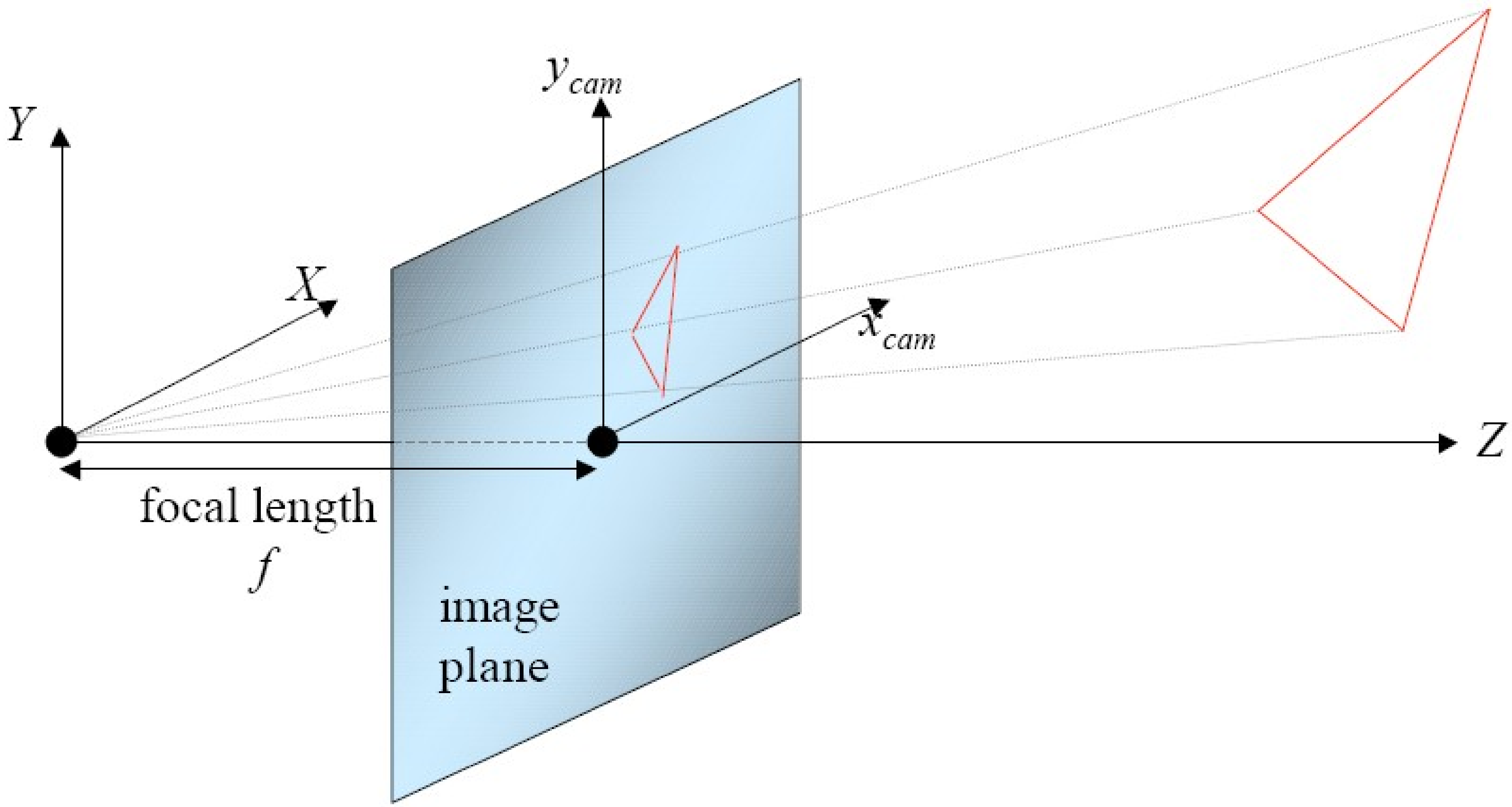




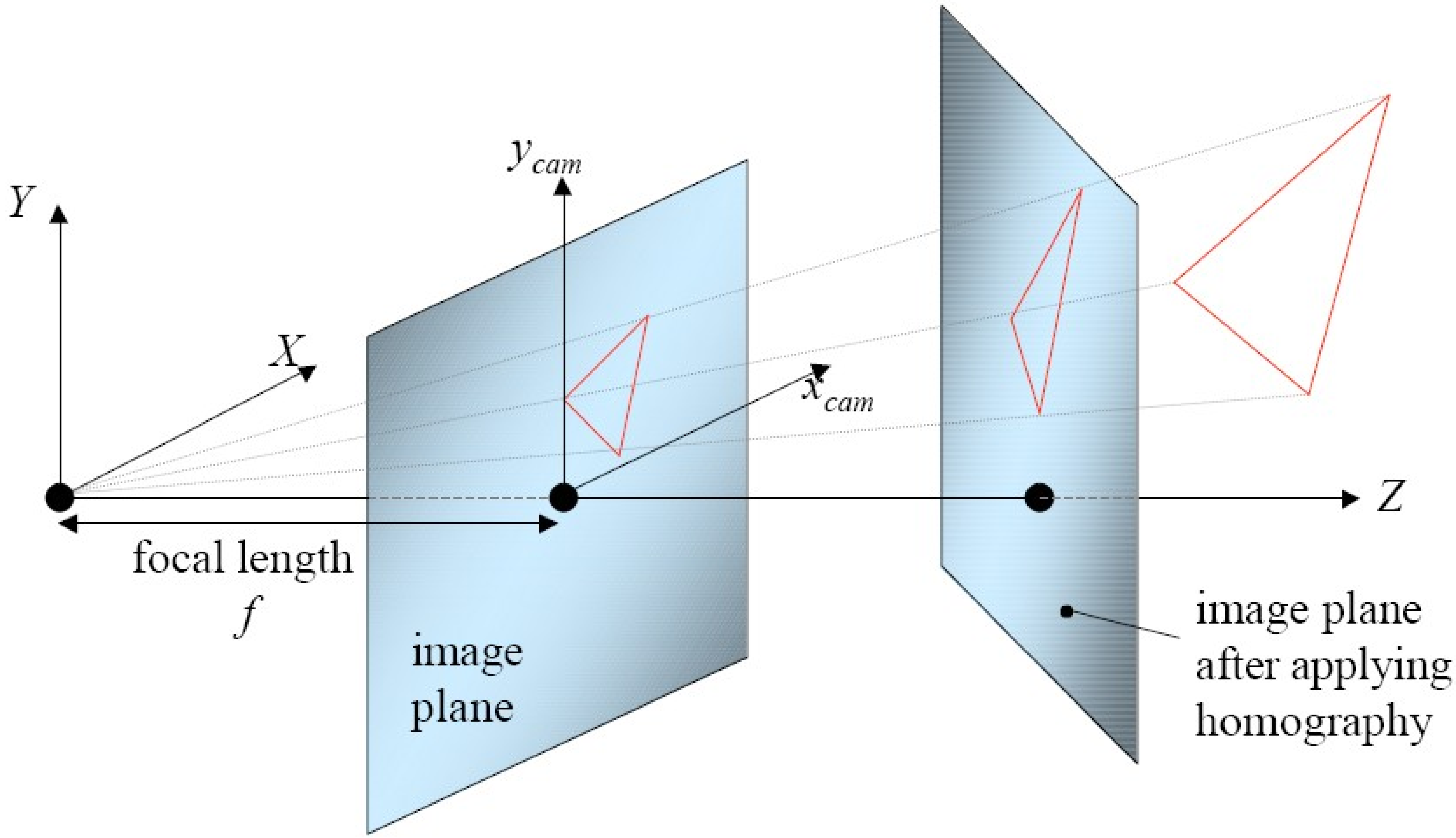
Two examples of rectified planar regions.

Texture Mapping – cont.

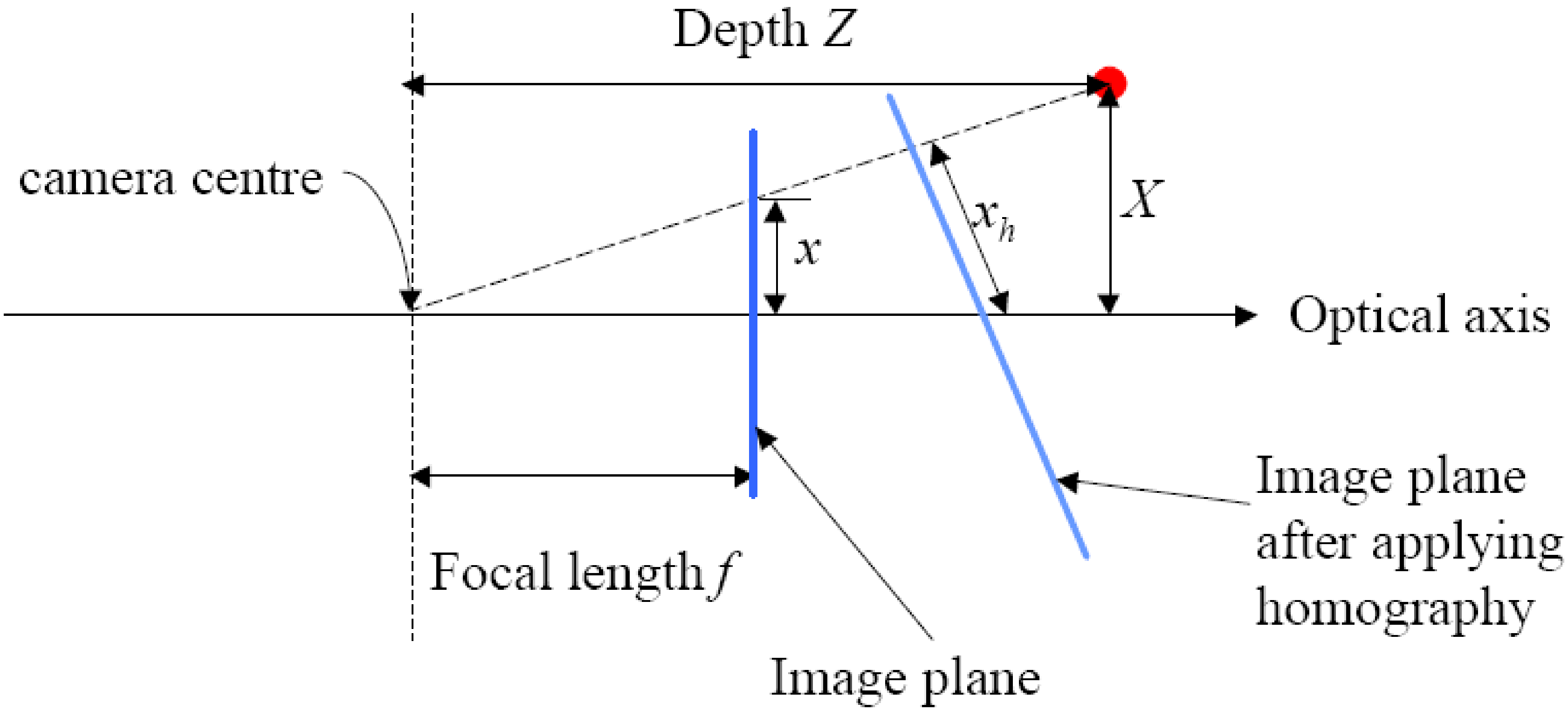
- How can we achieve that?



Texture Mapping – cont.



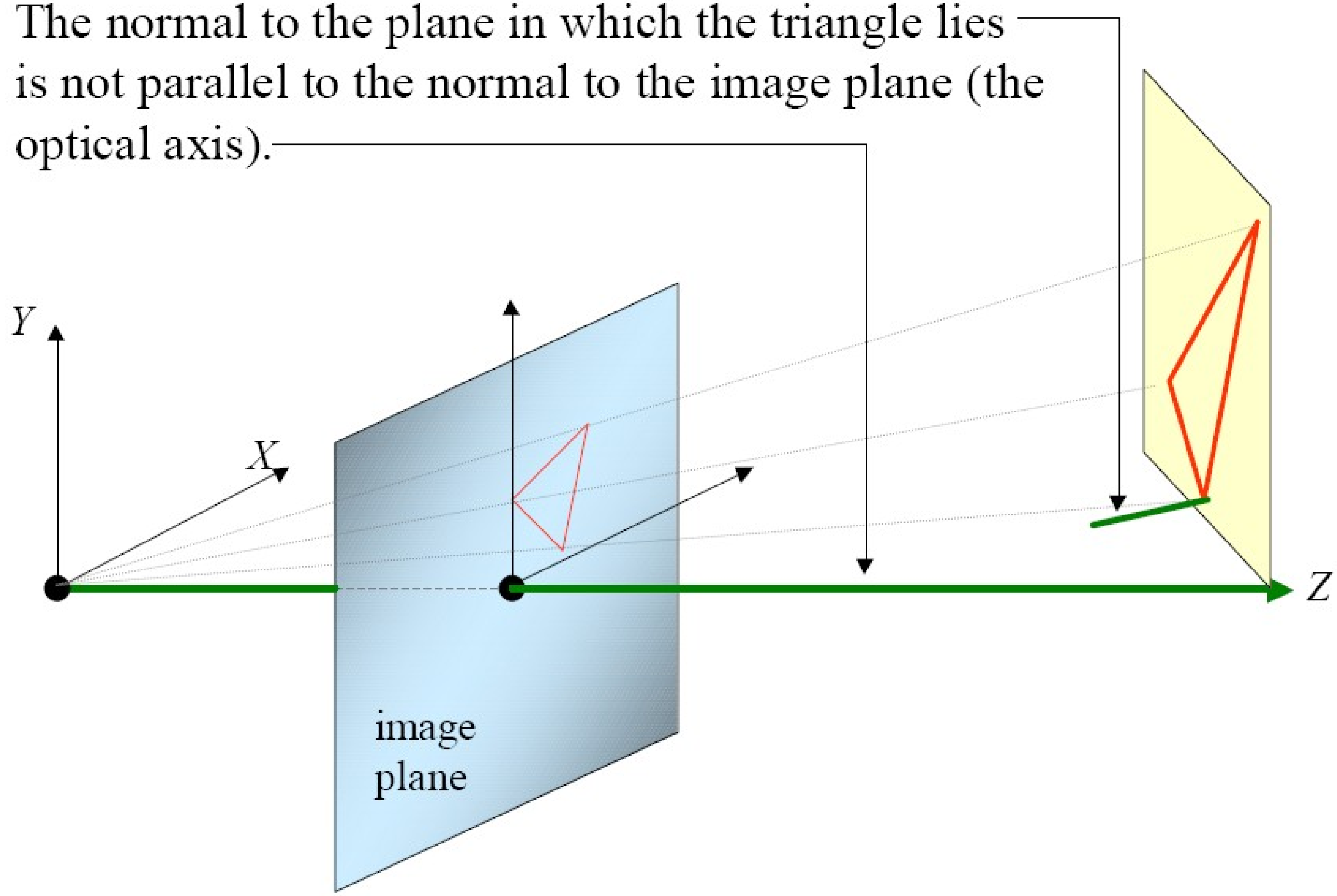
Texture Mapping – cont.



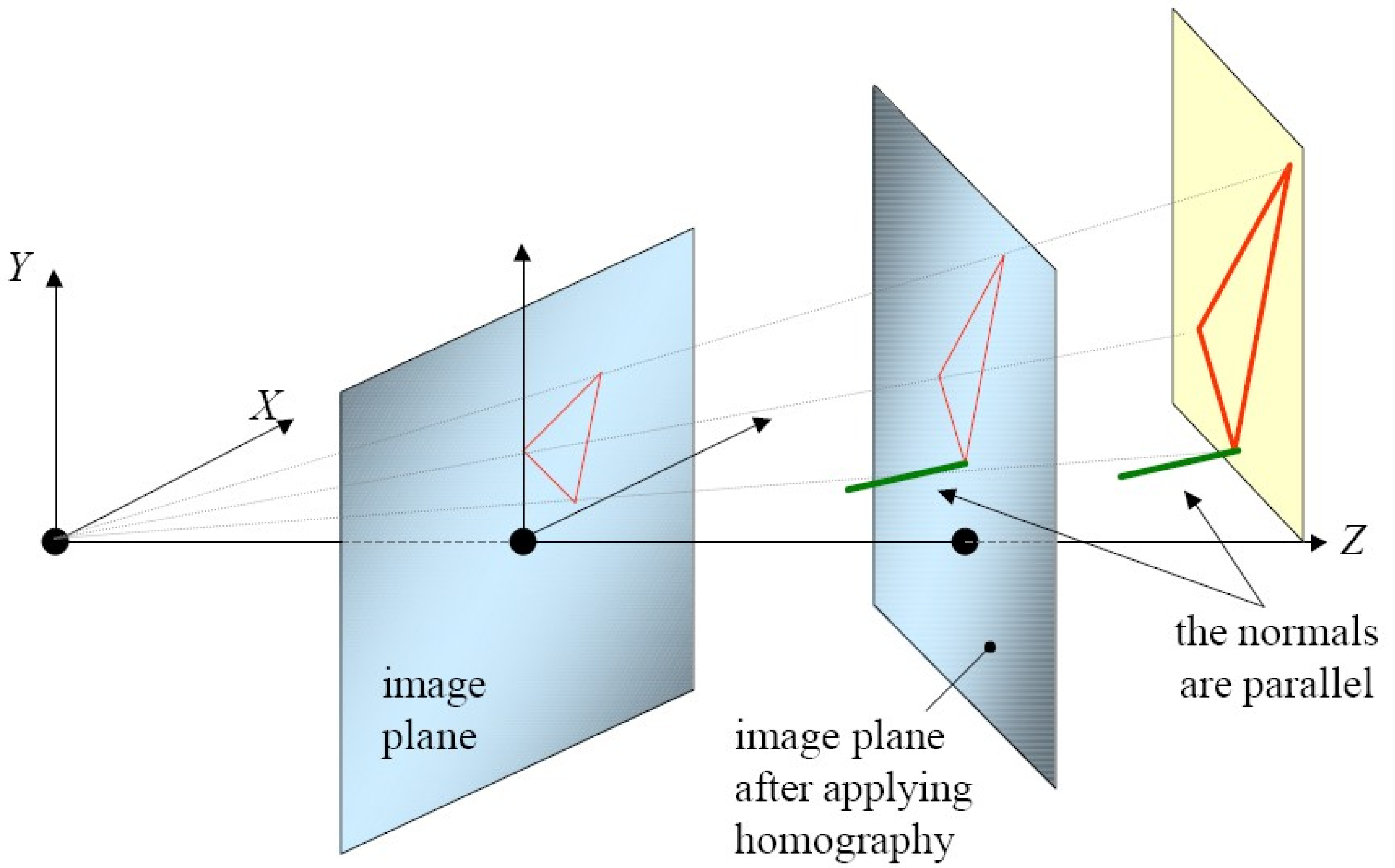
- That means, the normal of the new image plane has to be parallel of the normal of the planar surface given by the triangle

Texture Mapping – cont.

- The normal to the plane in which the triangle lies is not parallel to the normal to the image plane (the optical axis).

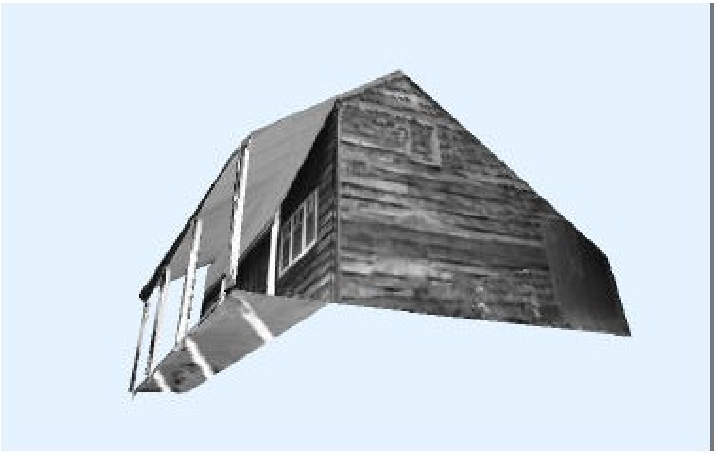


Texture Mapping – cont.



Texture Mapping – cont.

- There is also a scaling issue
  - The texture image for each triangle has to be consistent in size with other triangles
  - How to resolve this problem?



Summary

- Now we know how to estimate 3D points
  - Given the intrinsic camera parameters and correspondences (at least eight) in a stereo pair,
  - We can recover the points in 3D and also the relative camera position (up to a scale) and pose
    - By using eight point algorithms

- Next time: correspondence and advanced topic