Multi-view Geometry (Two View Geometry)

Chul Min Yeum

Assistant Professor

Civil and Environmental Engineering

University of Waterloo, Canada
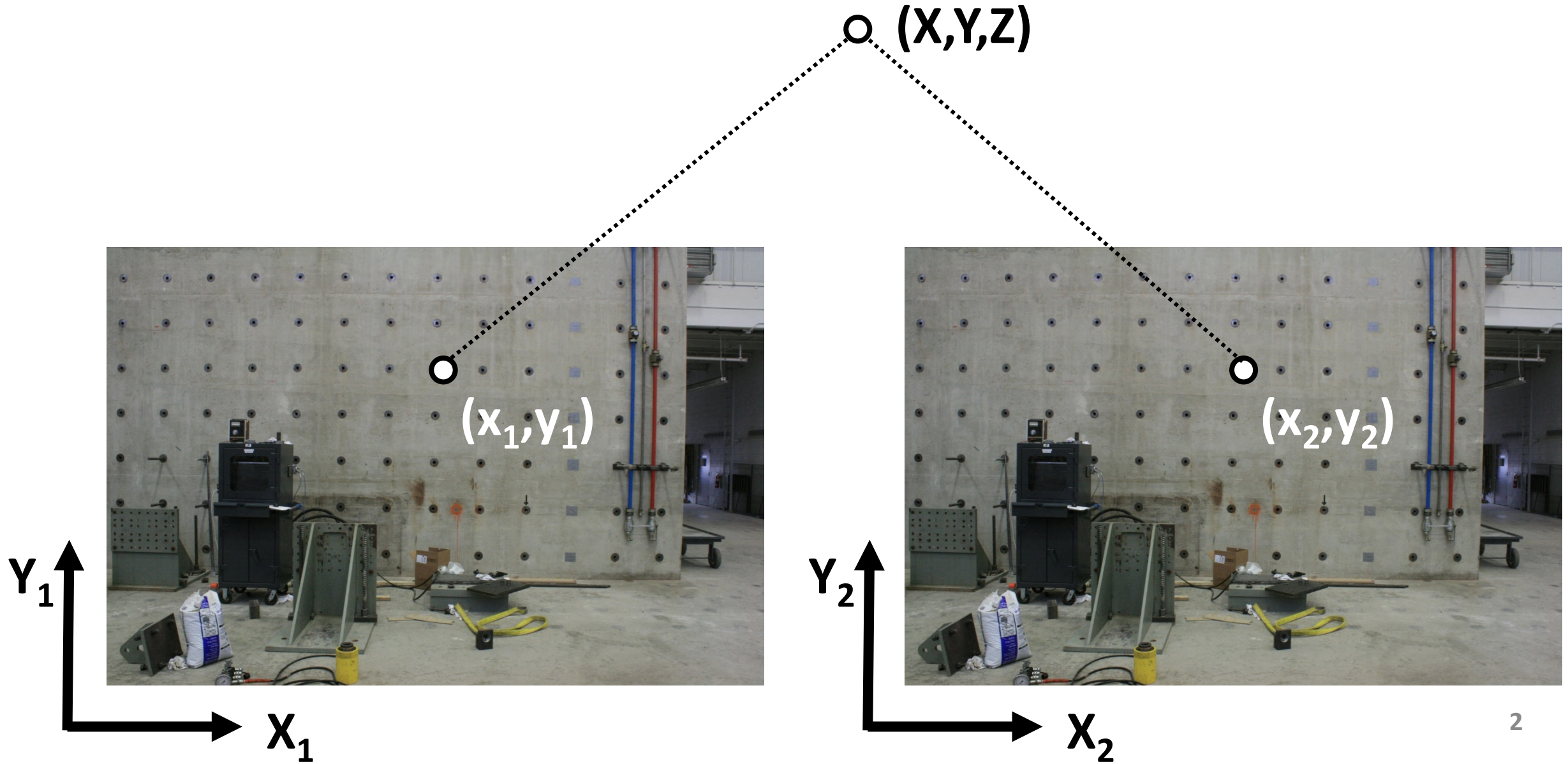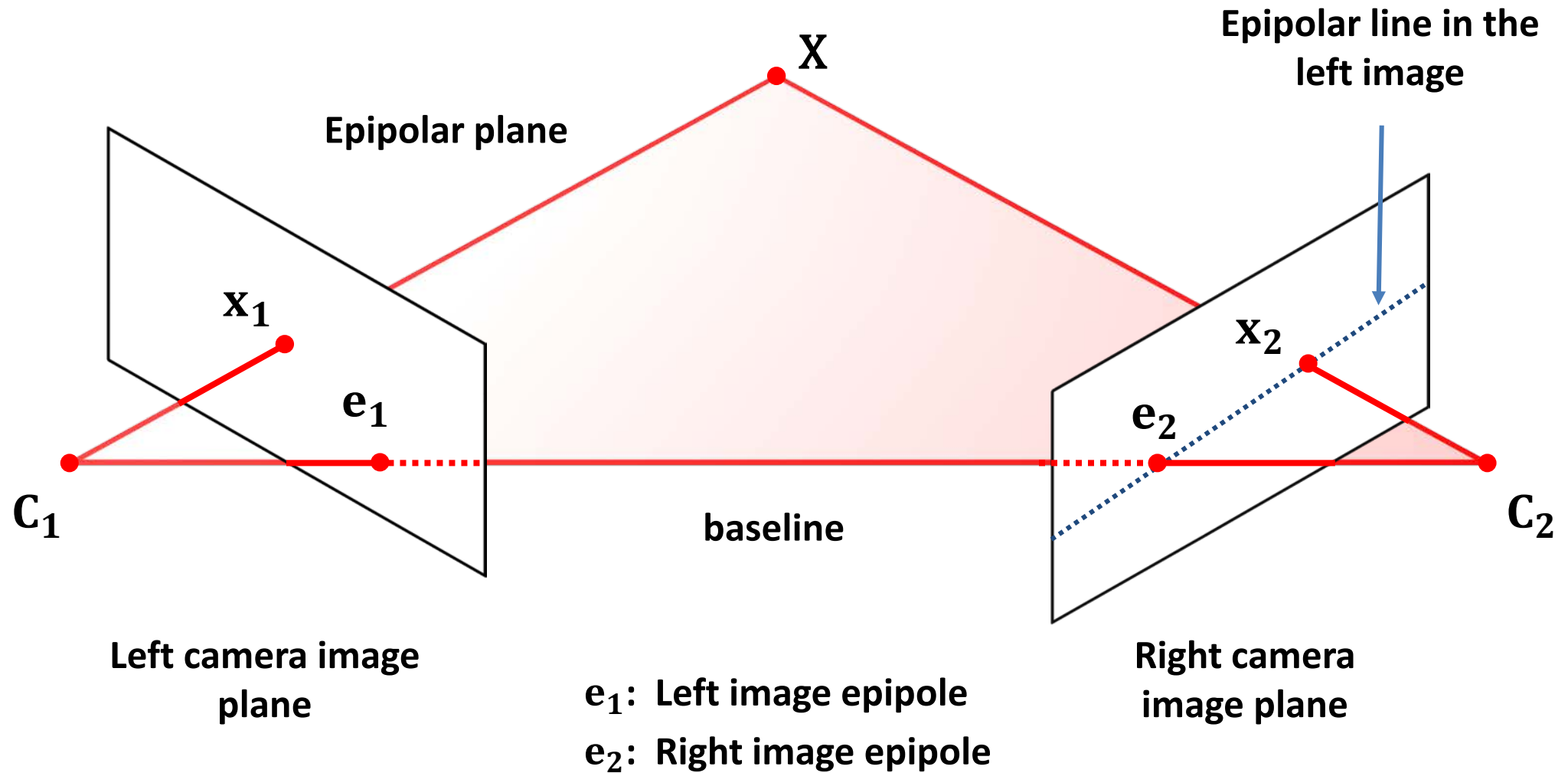
CIVE 497 – CIVE 700: Smart Structure Technology

Last updated: 2019-03-08

**UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING**

Epipolar line in the left image

X

Epipolar plane

x_1

e_1

x_2

e_2

C_1

baseline

C_2

Left camera image plane

Right camera image plane

e_1: Left image epipole

e_2: Right image epipole

Cameras $\mathbf{P_1}$ and $\mathbf{P_2}$ such that

$$x_1 = \mathbf{P_1}X \qquad x_2 = \mathbf{P_2}X$$

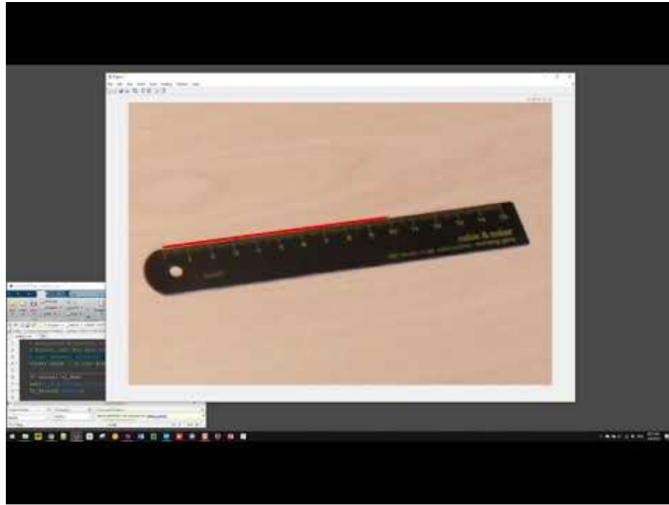Baseline between the cameras is non-zero

- Given an image point in the first view, where is the corresponding point in the second view?
- What is the relative position of the cameras?
- What is the 3D geometry of the scene?

Problem 4: Improved 3D Planar Measurement Tool (30 points)

You are going to build an improved 3D planar measurement tool. This application is designed to measure a 3D distance on a plane **without picking the corners of the booklet in advance**. Users will take a photo in a planar surface where a calibrate paper (here, booklet) is placed and simply click two points on the image for measurement. This reduces a step for picking the corners of your calibration paper.

Here is a demo video:

$$\begin{pmatrix} x_1' \\ x_2' \\ x_3' \end{pmatrix} \cong \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

$$\mathbf{x'} = \mathbf{Hx}$$

(a) Build your own measurement tool and evaluate your measurement using the images provided (see the folder of `prob4_img`). You may need to estimate homography based on SIFT feature matching. The exact size of the booklet is **24 cm x 31.5 cm** and use `cover.jpg` to solve this problem.

(b) Prepare your own calibration paper and take photos similar to the ones in (a). Then, evaluate your tool.

(c) Compare measurements using this new tool and the one developed in Task 3.
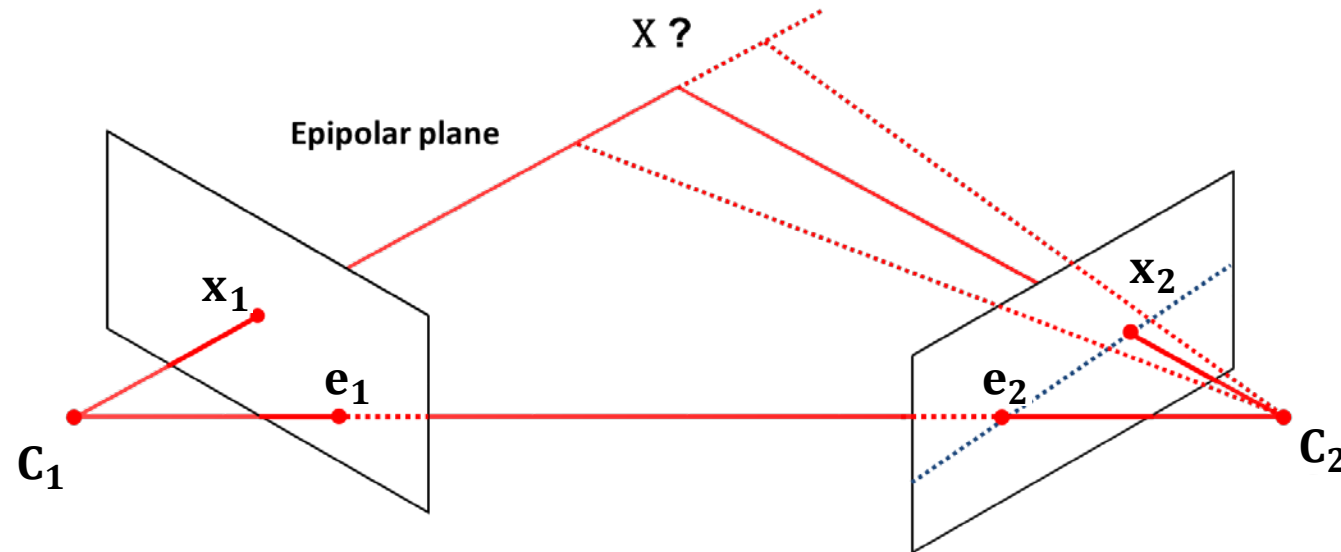
Note that you should not do hard cording. This means the corner points of the booklet on your **test image** is completely unknown.

Given the image of a point in one view, what can we say about its position in another?
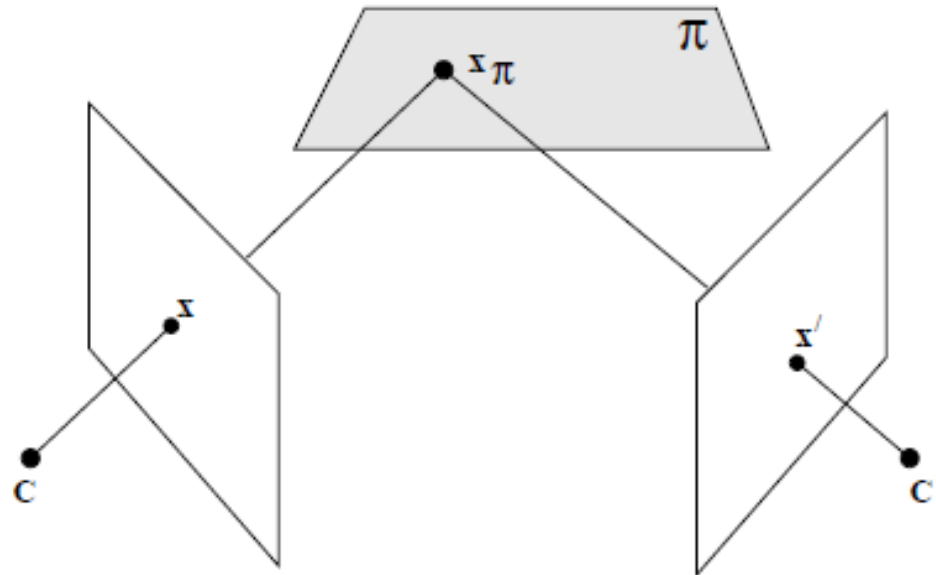

A point in one image "generates" a line in the other image. This line is known as an epipolar line, and the geometry which gives rise to it is known as epipolar geometry.

Projective transformation between images induced by a plane

$$\mathbf{x} = H_{1\pi}\mathbf{x}_\pi \qquad \mathbf{x}' = H_{2\pi}\mathbf{x}_\pi$$

$$\mathbf{x}' = H_{2\pi}\mathbf{x}_\pi$$

$$= H_{2\pi}H_{1\pi}^{-1}\mathbf{x} = H\mathbf{x}$$

H can be computed from the correspondence of four points on the plane

The vector cross product also can be expressed as the product of a skew-symmetric matrix and a vector:[11]

$$\mathbf{a} \times \mathbf{b} = [\mathbf{a}]_\times \mathbf{b} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$\mathbf{a} \times \mathbf{b} = [\mathbf{b}]_\times^T \mathbf{a} = \begin{bmatrix} 0 & b_3 & -b_2 \\ -b_3 & 0 & b_1 \\ b_2 & -b_1 & 0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix},$$

where superscript ^{T} refers to the transpose operation, and $[\mathbf{a}]_\times$ is defined by:

$$[\mathbf{a}]_\times \stackrel{\text{def}}{=} \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}.$$
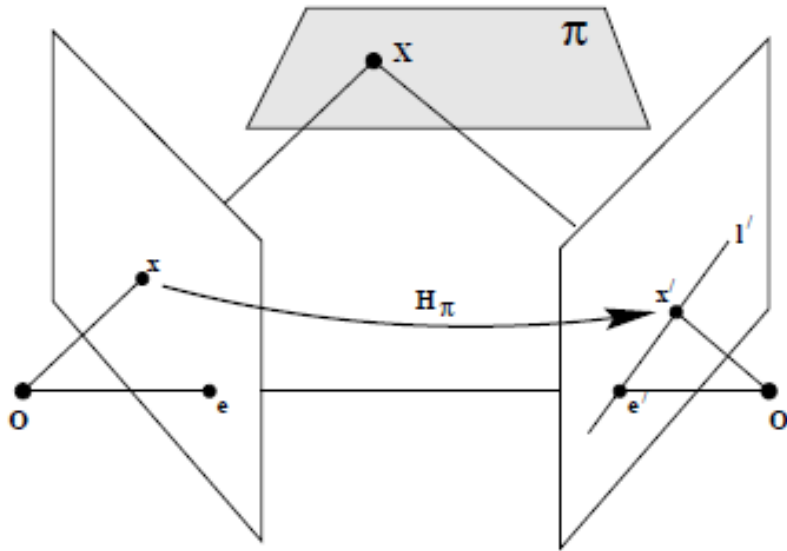
[a]$_x$ is a 3x3 skew-symmetric matrix of rank 2. a is the null-vector of [a]$_x$.

https://en.wikipedia.org/wiki/Cross_product

Given two camera looking at the same scenes, there exists a 3 x 3 matrix **F**, of rank 2 that captures the most fundamental relationship between the pixel $\mathbf{x_1}$ and $\mathbf{x_2}$ in the two cameras for the same scene point **X**

$$\mathbf{x}_2^T F \mathbf{x}_1 = 0$$

We call **F** the <u>**fundamental matrix**</u>.

Step 1: Point transfer via a plane $\mathbf{x}' = \mathrm{H}_\pi \mathbf{x}$

Step 2 : Construct the epipolar line $\mathbf{l}' = \mathbf{e}' \times \mathbf{x}' = [\mathbf{e}']_\times \mathbf{x}'$

$$\mathbf{l}' = [\mathbf{e}']_\times \mathrm{H}_\pi \mathbf{x} = \mathrm{F}\mathbf{x}$$

$$\mathrm{F} = [\mathbf{e}']_\times \mathrm{H}_\pi$$

rank(AB)≤min(rank(A),rank(B)).

This shows that F is a 3×3 rank 2 matrix.

1. F can be of great help in solving the stereo correspondence problem. This is the problem of finding the $\mathbf{x_2}$ in the second image that corresponds to a given $\mathbf{x_1}$ in the first image. If we know **F**, **we can confine our search to the line $\mathbf{l_2} = \mathbf{Fx_1}$ in the second image**, the corresponding pixel in the first image is on the epipolar line $\mathbf{l_1} = \mathbf{F^T x_2}$.

2. The determinant of F is always zero: det(**F**)=0. This follows from the fact that for all n x n matrices, the determinant obeys the multiplicative properties: det(AB) = det(A)·det(B). In $\mathrm{F}=[e_2]_\times \mathrm{H}_\pi$, the matrix is skew symmetric and therefore det($[e']_X$)=0.

3. The second-image epipole $\mathbf{e_2}$ is the left null-vector of **F** and the first-image epipole e is its right null vector: $\mathbf{e_2}^{\mathrm{T}}\mathbf{F} = 0$ and $\mathbf{F}\mathbf{e_1} = 0$. To prove $\mathbf{e_2}$ is the left null-vector, we note that $\mathbf{x_2}$ for a given $\mathbf{x_1}$ is on the right image line $\mathbf{l_2} = \mathbf{F}\mathbf{x_1}$. Since e' is also on this line, we have $\mathbf{e_2^T l_2} = \mathbf{e_2^T F x_1} = \mathbf{0}$. Since $\mathbf{e_2^T F x_1} = \mathbf{0}$ must be true for every pixel x in the first image, it must be the case that $\mathbf{e_2^T F} = \mathbf{0}$.

4. If **F** is the fundamental matrix for a given ordered pair of cameras, the fundamental matrix becomes $\mathbf{F^T}$ if you reverse the order of the cameras.

5. **F** is a rank 2 homogeneous matrix.

For a given pixel x, there exists a world point P^+x on the corresponding ray where $P^+ = P^T(PP^T)^{-1}$ (Pseudoinverse of P). This claim in the based on the observation that the location of the image of this world point is the same as x:

$$P \cdot (P^+x) = P \cdot \left(P^T(PP^T)^{-1}x\right) = PP^T(PP^T)^{-1}x = x$$

Since P is of rank 3, the 3 x 3 matrix PP^T is of full rank. The inverse is $(PP^T)^{-1}$ therefore guaranteed to exist.

By construction, $e_2 = P_2 C_1$ and $e_1 = P_1 C_2$
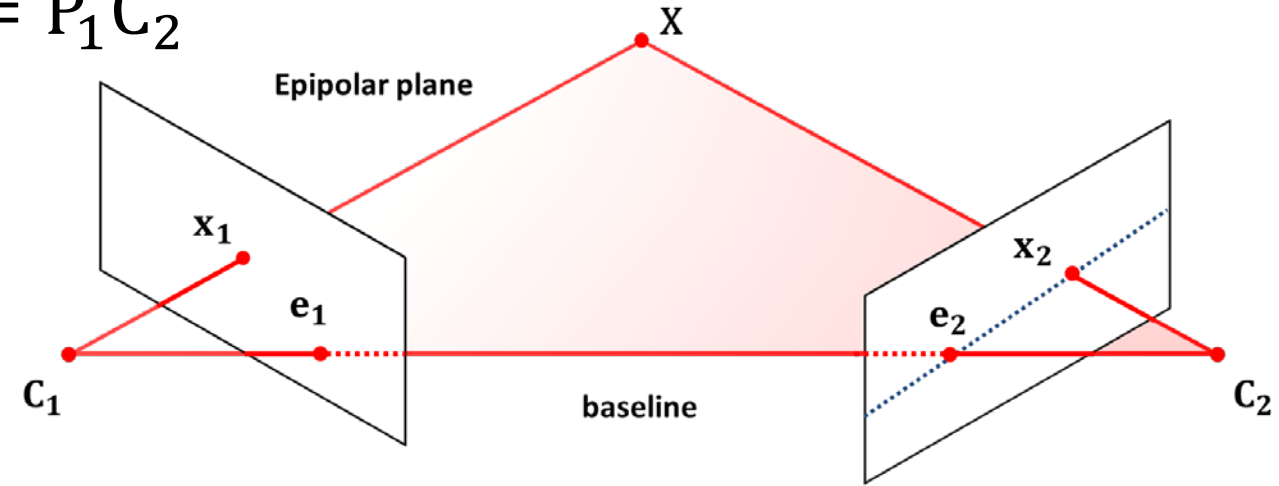
Two points on l_2: the epipole e_2 and the pixel at $P_2 \cdot (P_1^+ x_2)$. Therefore, $l_2 =$

$e_2 \times P_2 \cdot (P_1^+ x_2) = [e_2]_\times P_2 P_1^+ x_2$. Therefore, $l_2 = [e_2]_\times P_2 P_1^+ x_2 = \mathbf{F} x_2$ where

$\mathbf{F} = [e_2]_\times P_2 P_1^+.$

$$\mathbf{F} = [e_2]_\times P_2 P_1^+$$

Camera centre. The camera centre is the 1-dimensional right null-space C of P, i.e. $PC = 0$.

⋄ **Finite camera** (M is not singular) $C = \begin{pmatrix} -M^{-1}p_4 \\ 1 \end{pmatrix}$

⋄ **Camera at infinity** (M is singular) $C = \begin{pmatrix} d \\ 0 \end{pmatrix}$ where d is the null 3-vector of M,

i.e. $Md = 0$.

Column points. For $i = 1, \ldots, 3$, the column vectors p_i are vanishing points in the image corresponding to the X, Y and Z axes respectively. Column p_4 is the image of the coordinate origin.

Principal plane. The principal plane of the camera is P^3, the last row of P.

Axis planes. The planes P^1 and P^2 (the first and second rows of P) represent planes in space through the camera centre, corresponding to points that map to the image lines $x = 0$ and $y = 0$ respectively.

Principal point. The image point $x_0 = Mm^3$ is the principal point of the camera, where $m^{3\top}$ is the third row of M.

Principal ray. The principal ray (axis) of the camera is the ray passing through the camera centre C with direction vector $m^{3\top}$. The principal axis vector $v = \det(M)m^3$ is directed towards the front of the camera.

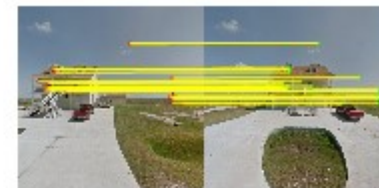(a) Read input GPS from a geo-tagged image

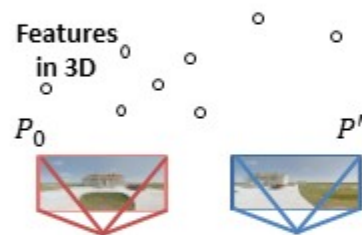(b) Download 360° panoramas (PN) near input GPS

(c) Generate two rectilinear images (RT) from the nearest PNs to the input GPS
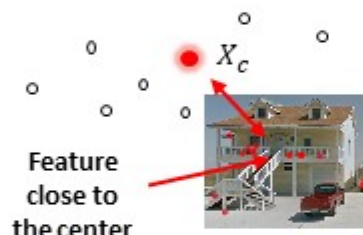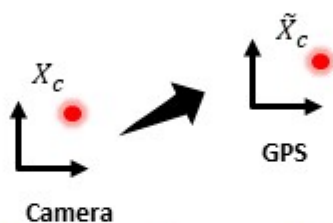
(d) Detect a building from each of these RTs
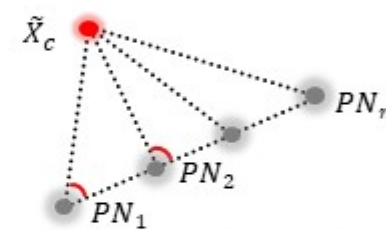
(e) Extract and match features

(f) Estimate projection matrices

(g) Define a approximate location of the building

(h) Transform the coordinate system

(i) Compute a direction for each PN

(j) Extract the optimal RTs by considering their direction

(k) Detect a building from the optimal RTs

(l) Localize building images

- Stage 1: Download the panoramas (PN) near a target building
- Stage 2: Compute the building location in GPS
- Stage 3: Generate the optimal rectilinear images (RT) using their proper direction
- Stage 4: Detect the target building from each of rectilinear images

- If we don't know K_1, K_2, R, or t, can we estimate F for two images?

- Yes, given enough correspondences

The fundamental matrix F is defined by

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

for any pair of matches x and x' in two images.

Let x=$(u,v,1)^T$ and x'=$(u',v',1)^T$,

each match gives a linear equation

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

$$uu' f_{11} + vu' f_{12} + u' f_{13} + uv' f_{21} + vv' f_{22} + v' f_{23} + uf_{31} + vf_{32} + f_{33} = 0$$

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

In reality, instead of solving $\mathbf{A}\mathbf{f} = 0$, we seek $\mathbf{f}$ to minimize $\|\mathbf{A}\mathbf{f}\|$, least eigenvector of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$.

- **F** should have rank 2

- To enforce that **F** is of rank 2, **F** is replaced by **F'** that minimizes $\|\mathbf{F} - \mathbf{F}'\|$ subject to the rank constraint.

- This is achieved by SVD. Let $\mathbf{F} = \mathbf{U}\mathbf{\Sigma}\mathbf{V^T}$, where

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} \qquad \Sigma' = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

then $\mathbf{F} = \mathbf{U}\mathbf{\Sigma}'\mathbf{V^T}$ is the solution.

8 Point Algorithm (MATLAB)

```matlab
% Build the constraint matrix
A = [x2(1,:)'.*x1(1,:)'   x2(1,:)'.*x1(2,:)'  x2(1,:)' ...
     x2(2,:)'.*x1(1,:)'   x2(2,:)'.*x1(2,:)'  x2(2,:)' ...
     x1(1,:)'             x1(2,:)'            ones(npts,1) ];


[U,D,V] = svd(A);


% Extract fundamental matrix from the column of V
% corresponding to the smallest singular value.
F = reshape(V(:,9),3,3)';


% Enforce rank2 constraint
[U,D,V] = svd(F);
F = U*diag([D(1,1) D(2,2) 0])*V';
```

$$\begin{bmatrix} u_1u_1' & v_1u_1' & u_1' & u_1v_1' & v_1v_1' & v_1' & u_1 & v_1 & 1 \\ u_2u_2' & v_2u_2' & u_2' & u_2v_2' & v_2v_2' & v_2' & u_2 & v_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_nu_n' & v_nu_n' & u_n' & u_nv_n' & v_nv_n' & v_n' & u_n & v_n & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

~10000   ~10000   ~100   ~10000   ~10000   ~100   ~100   ~100   1

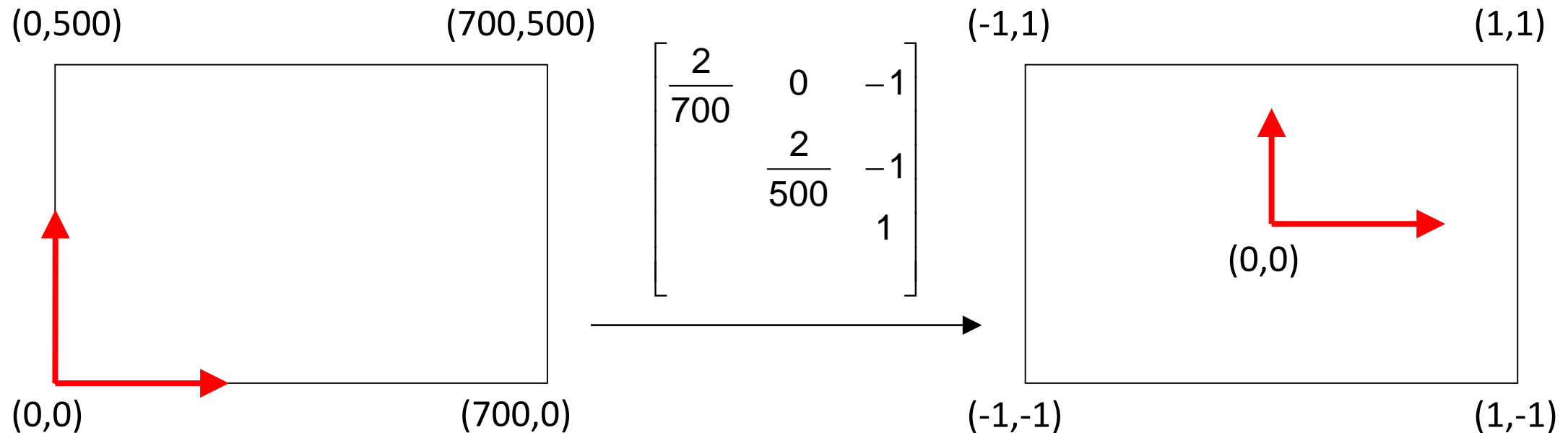⚠ Orders of magnitude difference between column of data matrix
→ least-squares yields poor results

normalized least squares yields good results

Transform image to become [-1,1] x [-1,1]

(0,500)   (700,500)   (-1,1)   (1,1)

$$\begin{bmatrix} \dfrac{2}{700} & 0 & -1 \\[2mm] & \dfrac{2}{500} & -1 \\[2mm] & & 1 \end{bmatrix}$$

(0,0)

(0,0)   (700,0)   (-1,-1)   (1,-1)

1. Transform input by $\hat{\mathbf{x}}_i = \mathbf{T}\mathbf{x}_i$ and $\hat{\mathbf{x}}_i' = \mathbf{T}\mathbf{x}_i'$

2. Call 8-point on to obtain $\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_i'$ to obtain $\hat{\mathbf{F}}$

3. $\mathbf{F} = \mathbf{T'}^\mathbf{T}\hat{\mathbf{F}}\mathbf{T}$

$$\mathbf{x'}^\mathrm{T}\mathbf{F}\mathbf{x} = 0$$

$$\boxed{\hat{\mathbf{x}}'^\mathrm{T}\mathbf{T'}^{-\mathrm{T}}}\mathbf{F}\boxed{\mathbf{T}^{-1}\hat{\mathbf{x}}} = 0$$

$$\hat{\mathbf{F}}$$

Normalized 8-point Algorithm (MATLAB)

```matlab
[x1, T1] = normalise2dpts(x1);
[x2, T2] = normalise2dpts(x2);

% Build the constraint matrix
A = [x2(1,:)'.*x1(1,:)'    x2(1,:)'.*x1(2,:)'   x2(1,:)' ...
     x2(2,:)'.*x1(1,:)'    x2(2,:)'.*x1(2,:)'   x2(2,:)' ...
     x1(1,:)'              x1(2,:)'             ones(npts,1) ];


[U,D,V] = svd(A);


% Extract fundamental matrix from the column of V
% corresponding to the smallest singular value.
F = reshape(V(:,9),3,3)';


% Enforce rank2 constraint
[U,D,V] = svd(F);
F = U*diag([D(1,1) D(2,2) 0])*V';


% Denormalise
F = T2'*F*T1
```

See tutorials:

fundamental_matrix_8points_V1.m

fundamental_matrix_norm_8points_V1.m

Slide Credits and References

- Lecture notes: Robert Collins
- Lecture notes: Avinash Kak
- Lecture notes: Noah Snavely
- Lecture notes: Richard Hartely and Andrew Zisserman
- Hartley, Richard, and Andrew Zisserman. Multiple view geometry in computer vision. Cambridge university press, 2003.