Introduction to Computer Vision

Kaveh Fathian
Assistant Professor
Computer Science Department
Colorado School of Mines

Lecture 12

Fundamental Equations of Computer Vision

1. Image Filtering $$h[m,n] = \sum_{k,l} f[k,l]\, I[m+k, n+l]$$

2. Optical Flow $$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t)$$

3. Camera Geometry $$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R}\ \mathbf{t}\end{bmatrix}\mathbf{X} \qquad x^T F x' = 0$$

4. Machine Learning $$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2. \qquad y = \varphi(\sum_{i=1}^{n} w_i x_i + b) = \varphi(\mathbf{w}^T \mathbf{x} + b)$$

The camera as a coordinate transformation

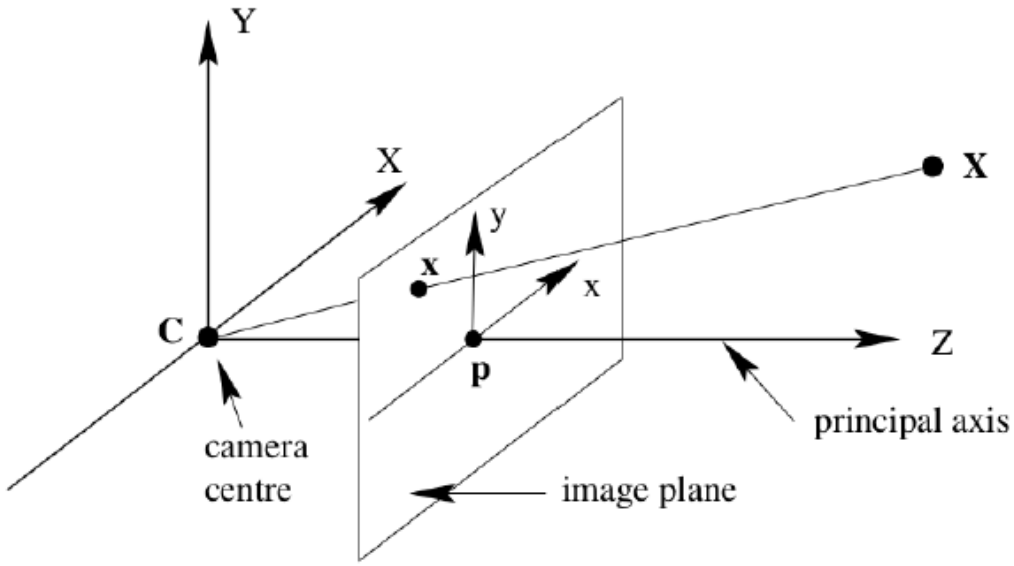
homogeneous coordinates

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

2D image point

camera matrix

3D world point



General pinhole camera matrix

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$$

$$\mathbf{P} = \begin{bmatrix} \alpha_x & s & p_x \\ 0 & \alpha_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{ccc|c} r_1 & r_2 & r_3 & t_1 \\ r_4 & r_5 & r_6 & t_2 \\ r_7 & r_8 & r_9 & t_3 \end{array}\right]$$

intrinsic parameters

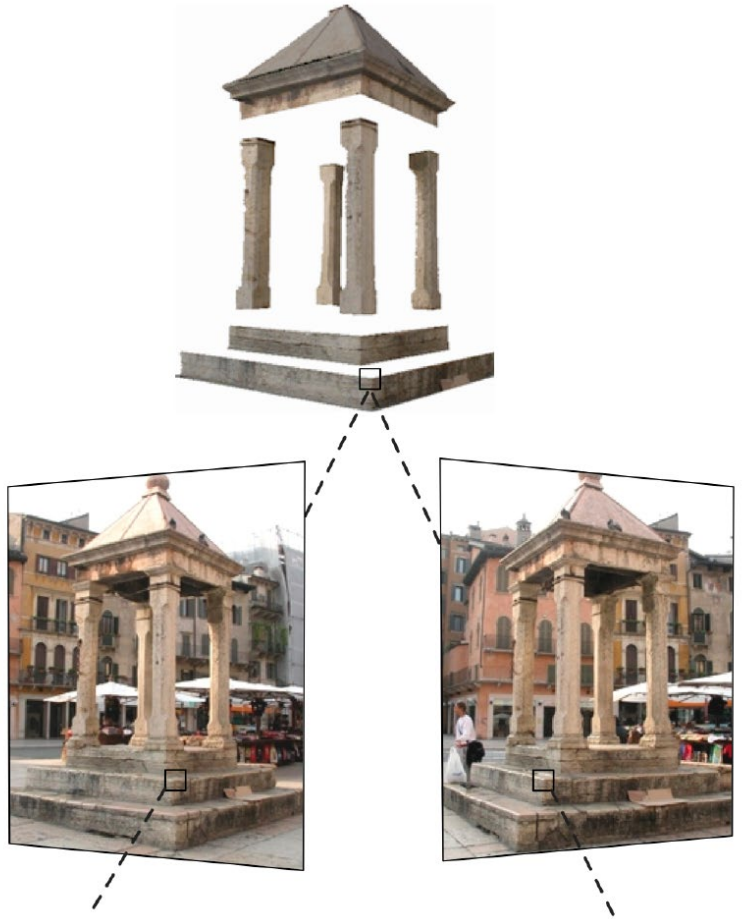
extrinsic parameters

$$\mathbf{R} = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} \qquad \mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

3D rotation

3D translation

Two-view geometry



Two-View Geometry

Learning outcomes:

- Epipolar Geometry
- Essential and Fundamental Matrices
- Eight-Point Algorithm
- Homography
- RANSAC
- Triangulation
- Stereo Vision

Reconstruction Problem

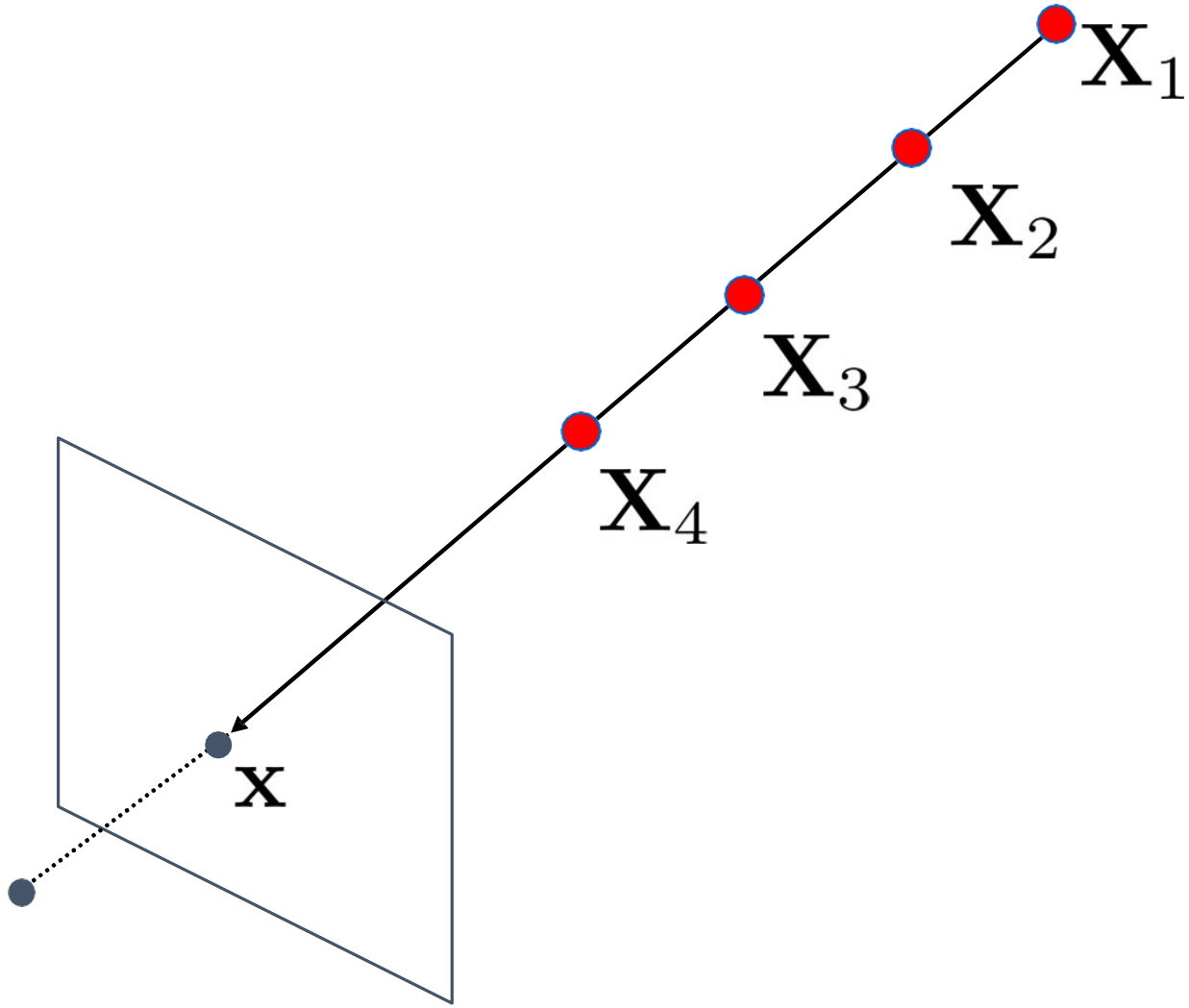
$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

Known Known Unknown

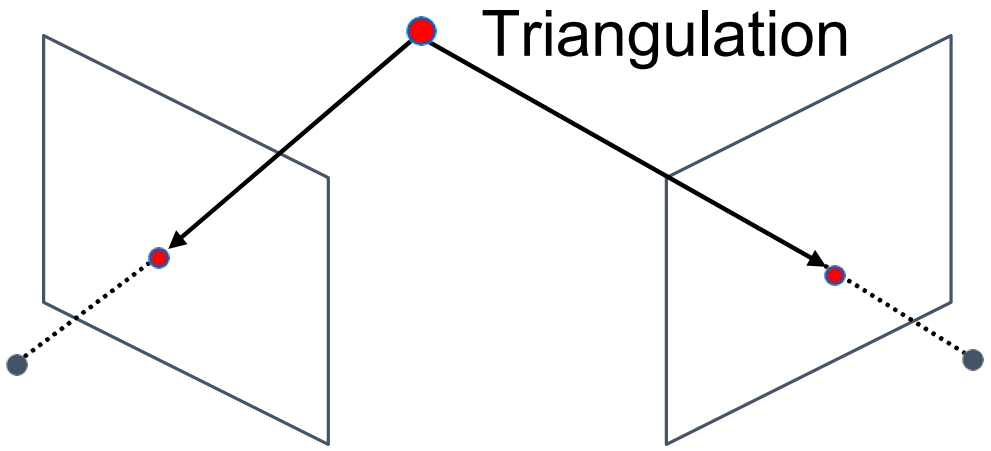
$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

Known Unknown Unknown

How to find these?



Stereo Vision



Multiple View Reconstruction

Two cameras, simultaneous views

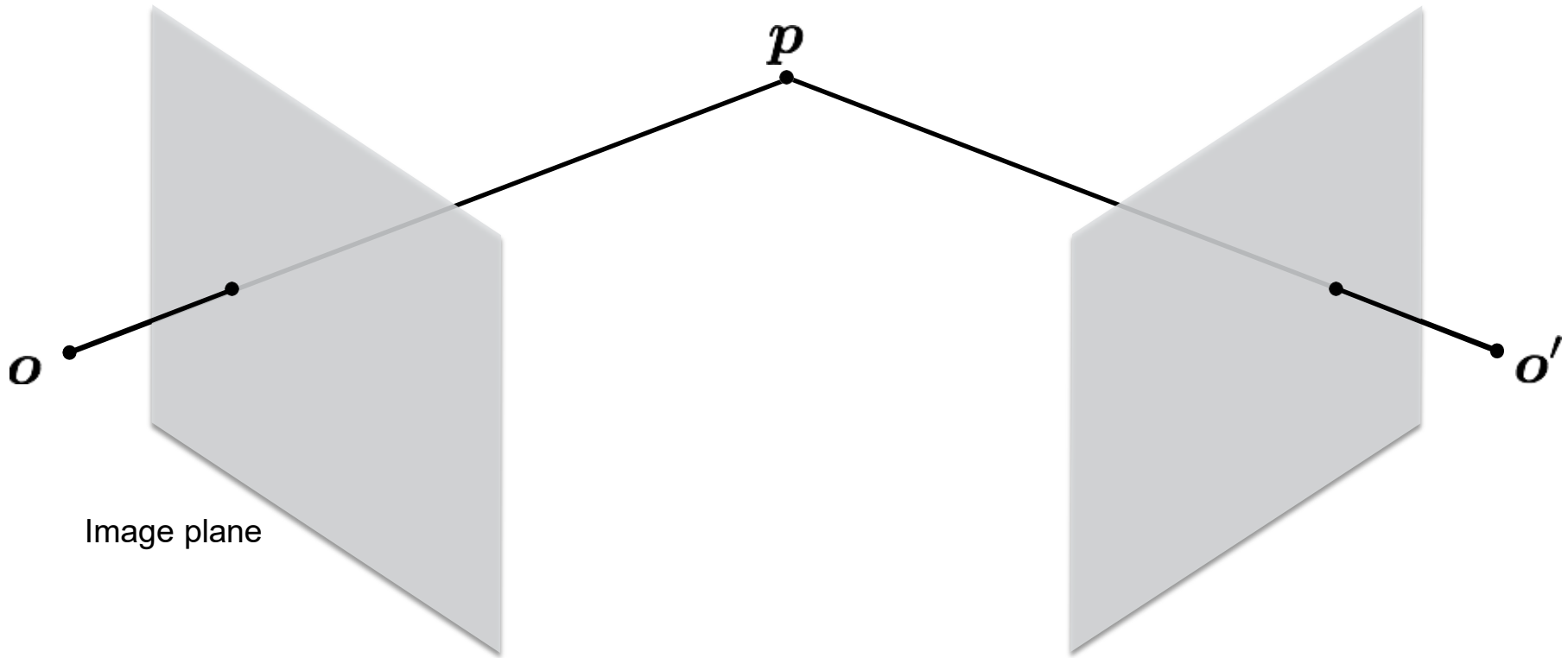
Single moving camera and static scene

Stereo vision

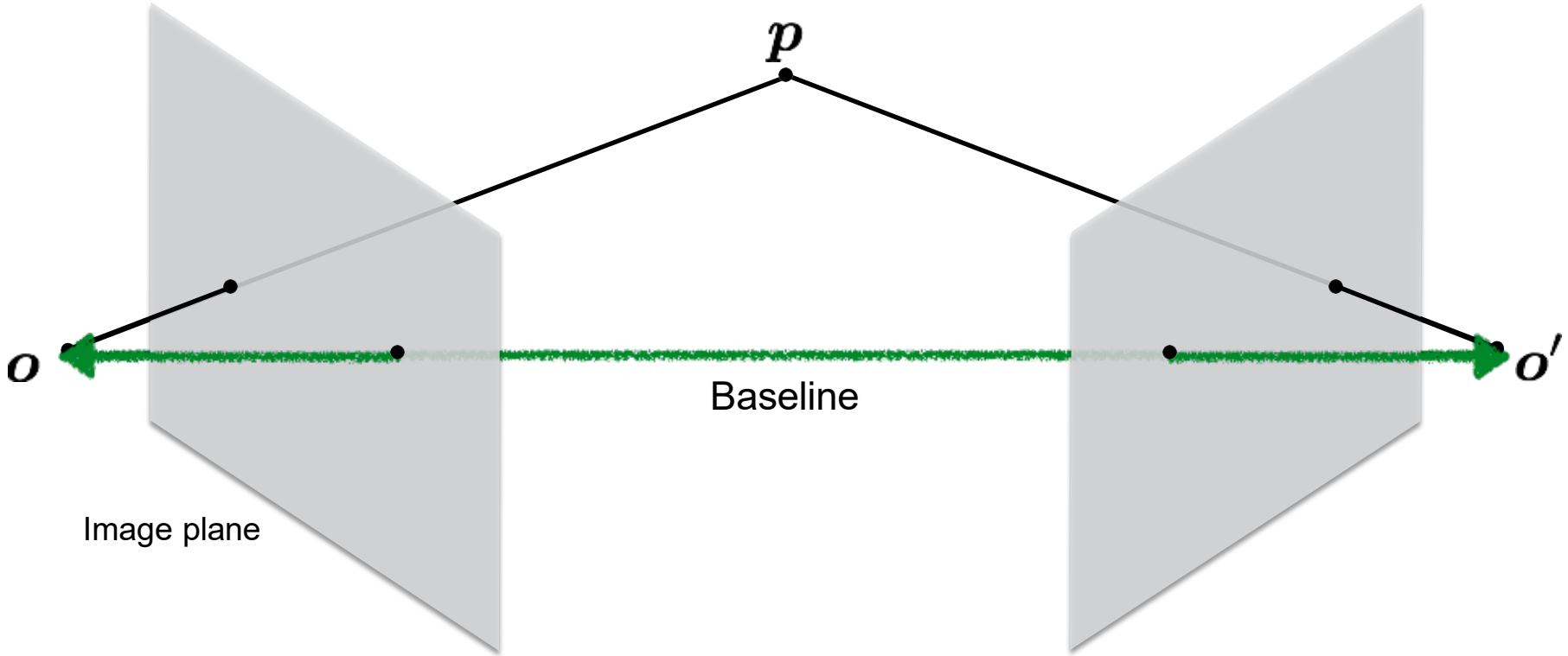
Structure from motion /
SLAM Optical flow

Epipolar Geometry

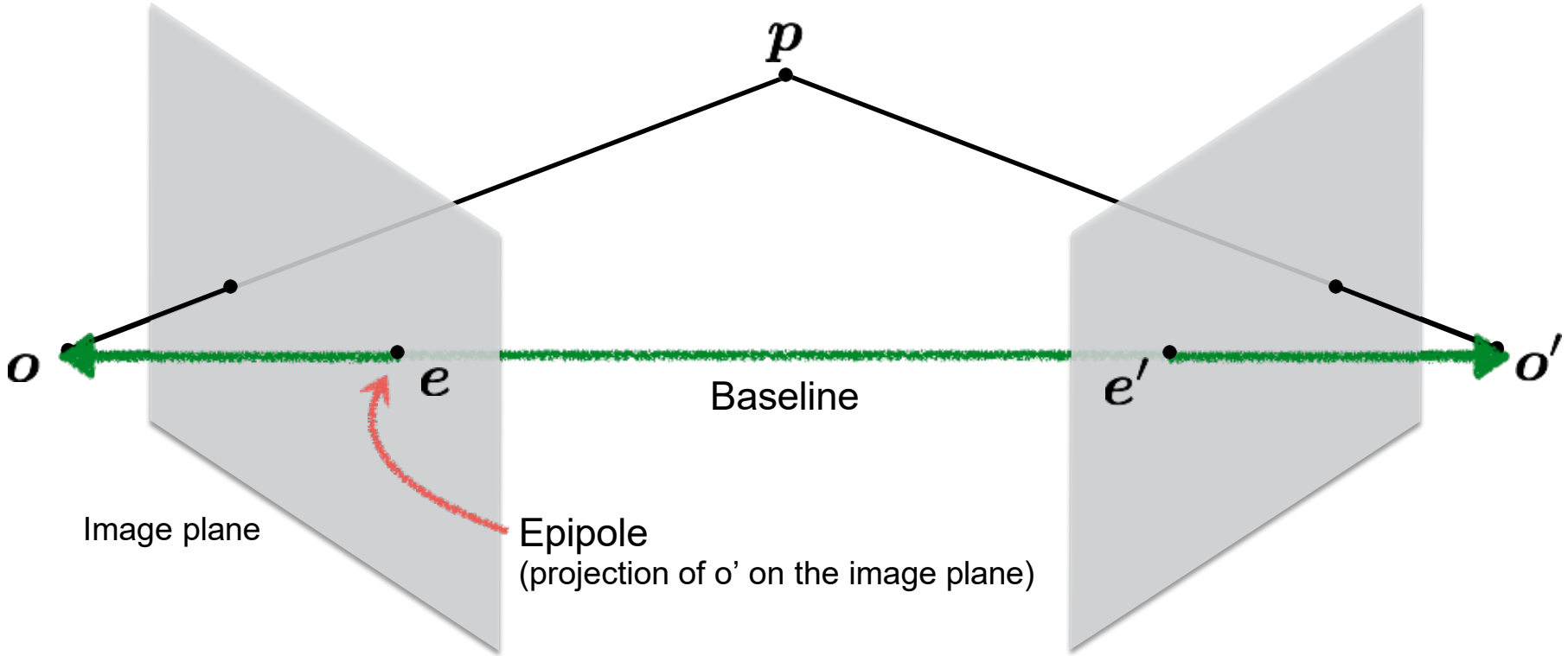
Notation



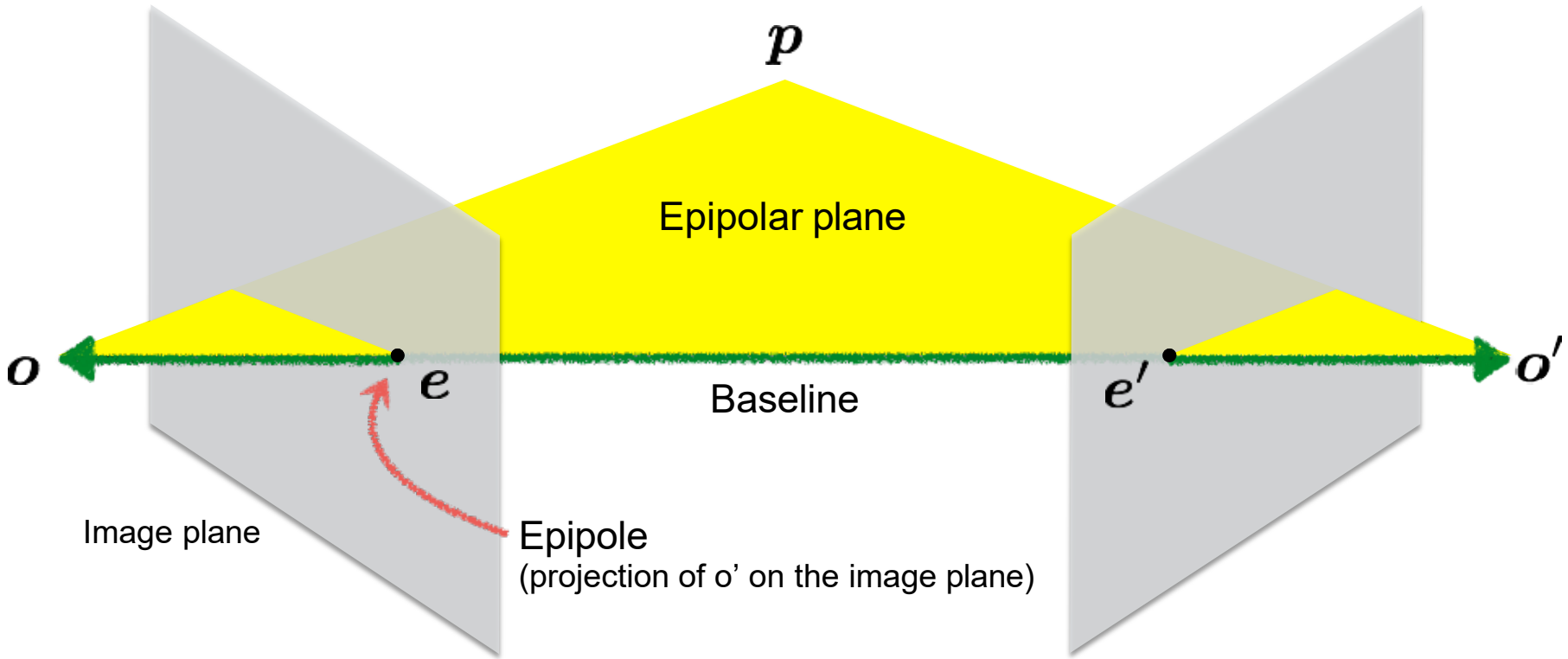
Notation



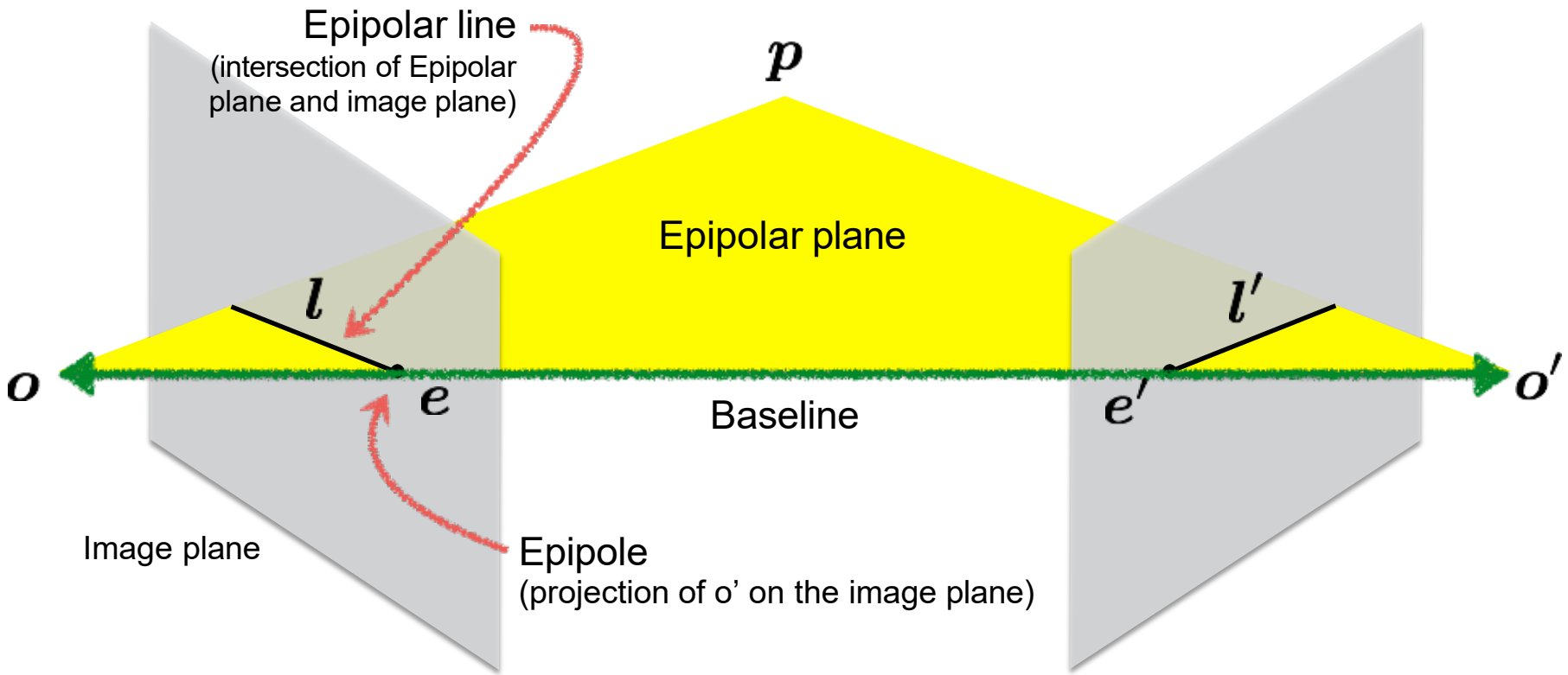
Notation


p
o
e
e′
o′
Baseline
Image plane
Epipole
(projection of o’ on the image plane)

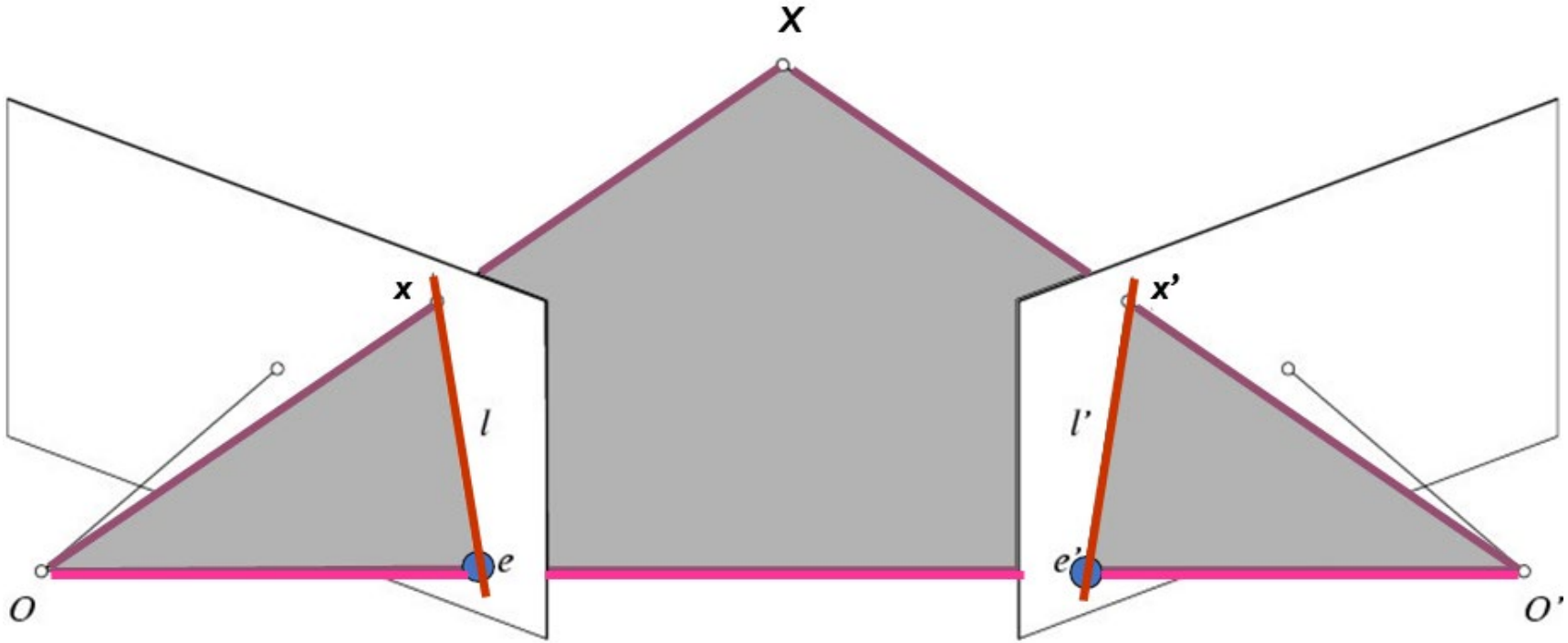
Notation


p
Epipolar plane
o
e
Baseline
e′
o′
Image plane
Epipole
(projection of o’ on the image plane)

Notation


Epipolar line
(intersection of Epipolar plane and image plane)
p
Epipolar plane
l
l′
o
e
e′
o′
Baseline
Image plane
Epipole
(projection of o' on the image plane)

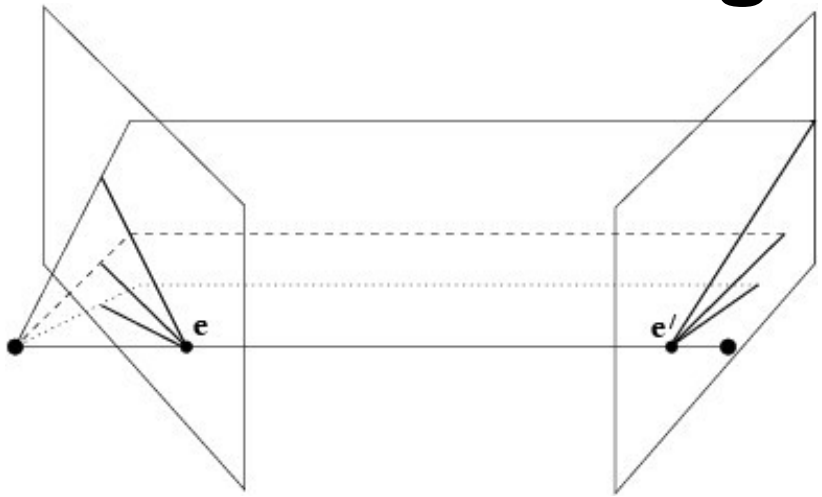
Notation

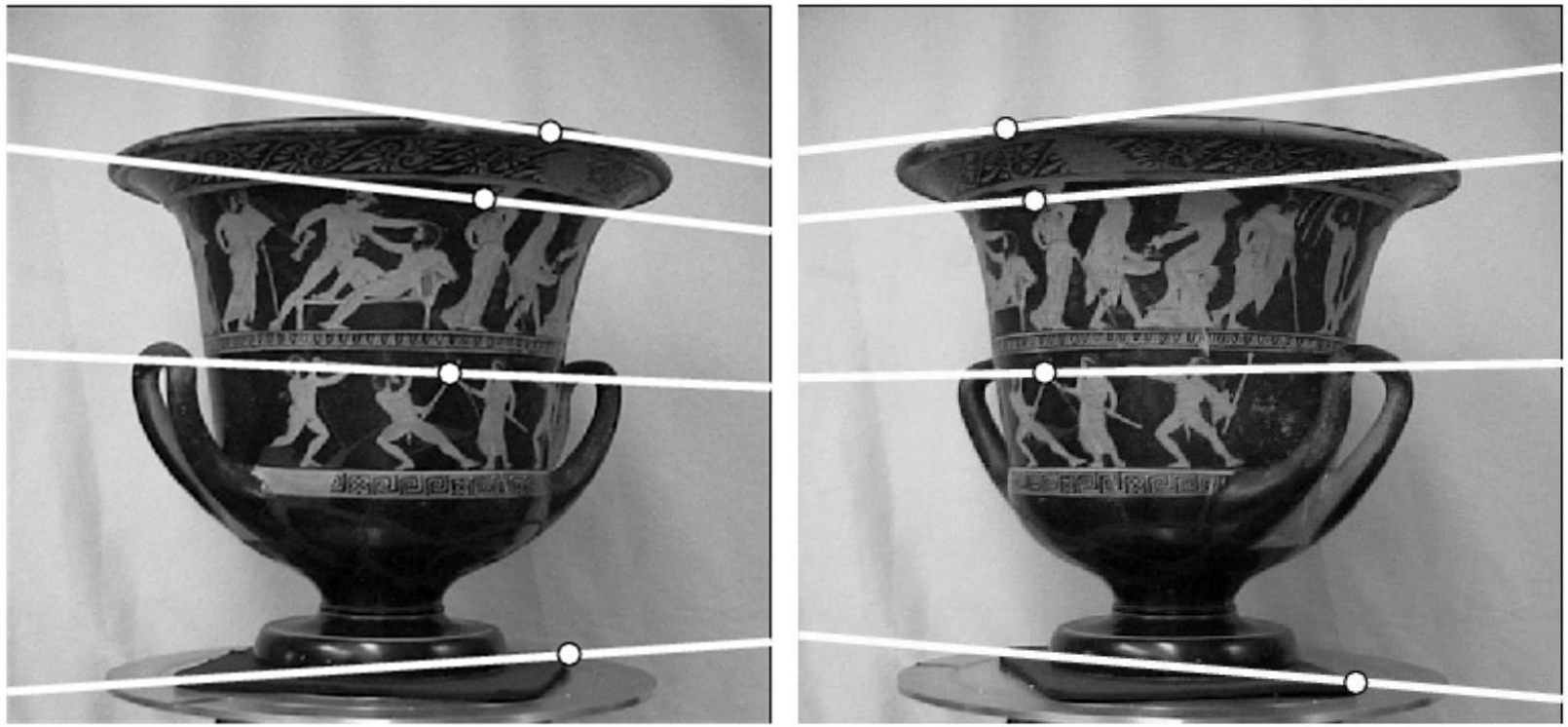


- **Baseline**: Line connecting the two camera centers
- **Epipoles**:
  - Intersections of baseline with image planes, or
  - Projections of the other camera center
- **Epipolar Plane**: Plane containing baseline
- **Epipolar Lines**: Intersections of epipolar plane with image planes (always come in corresponding pairs)

Example: Inward Facing Cameras

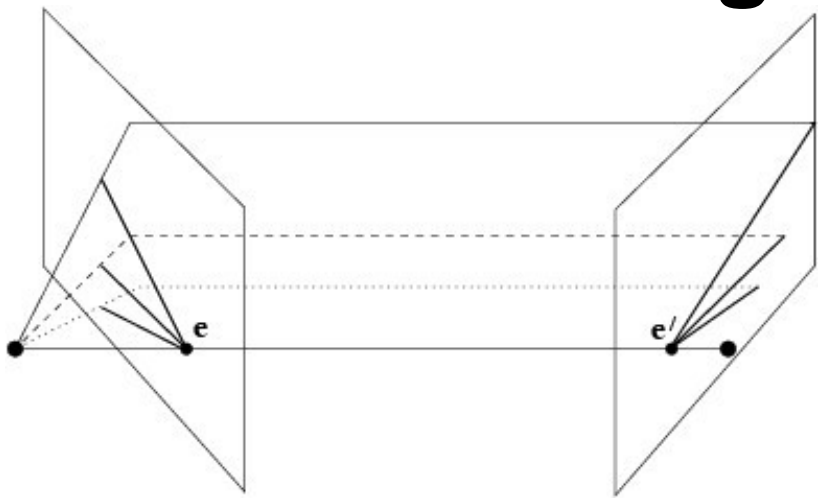
Where is the epipole in this image?

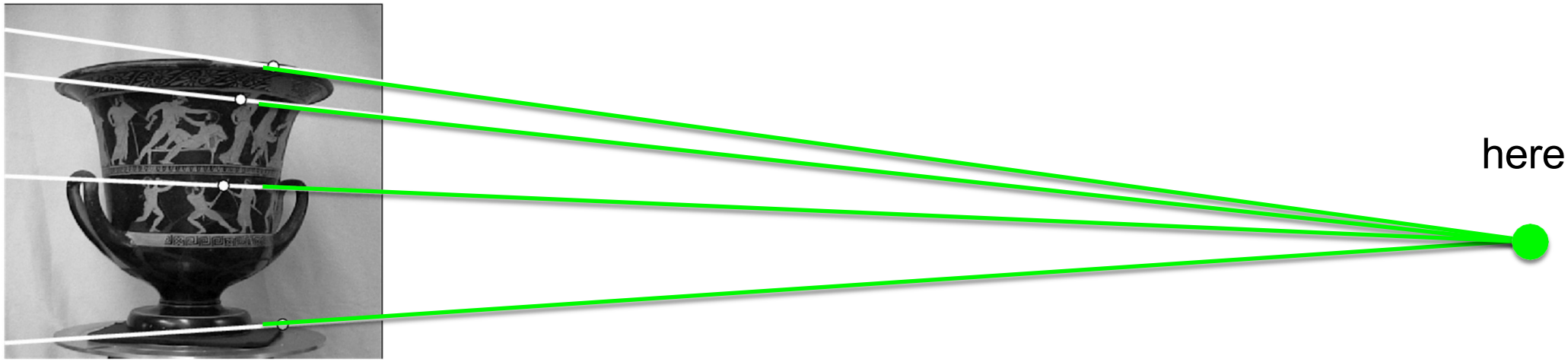




Example: Inward Facing Cameras

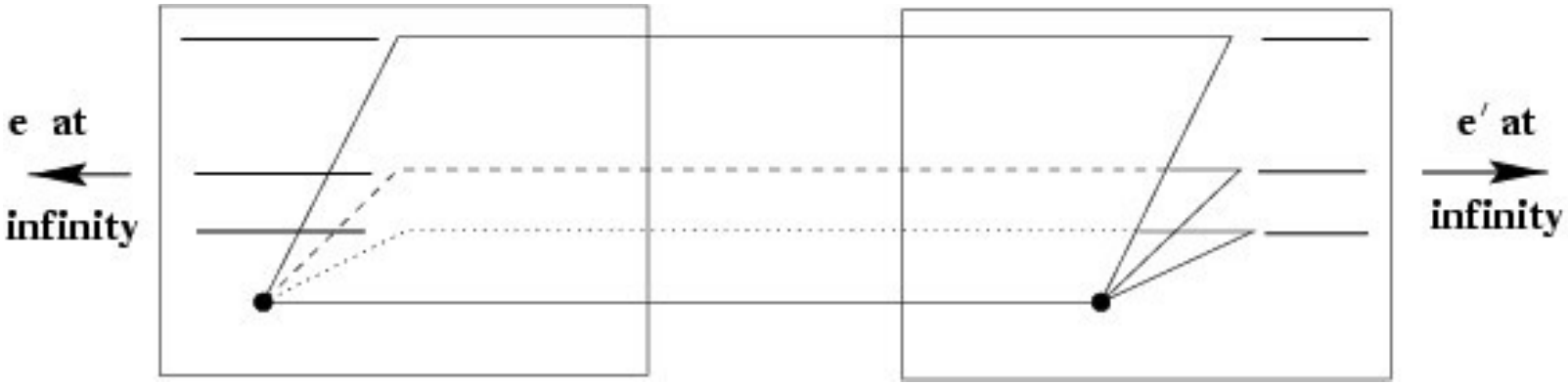
Where is the epipole in this image?



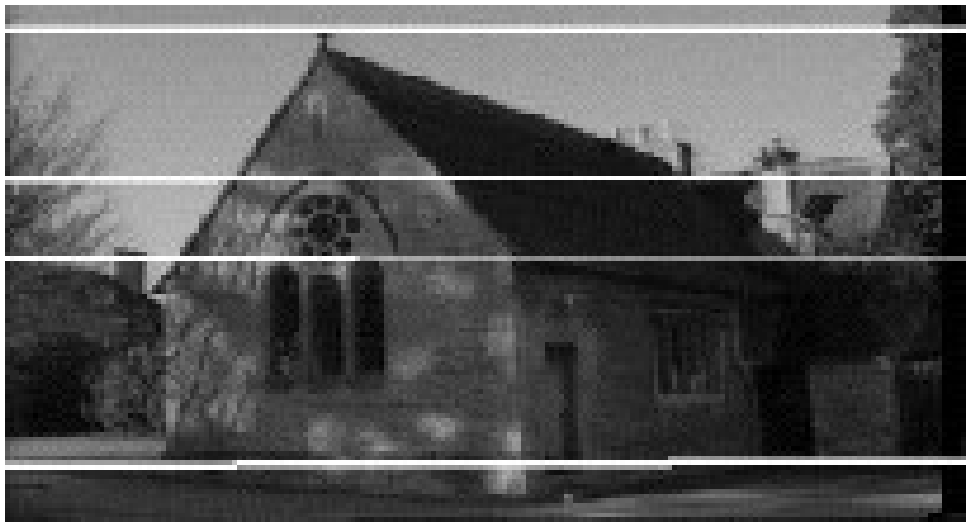


here!

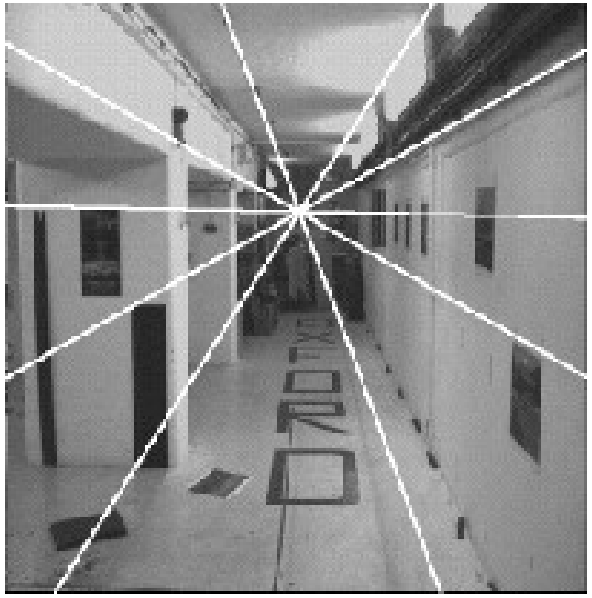
Example: Motion Parallel to Image Plane

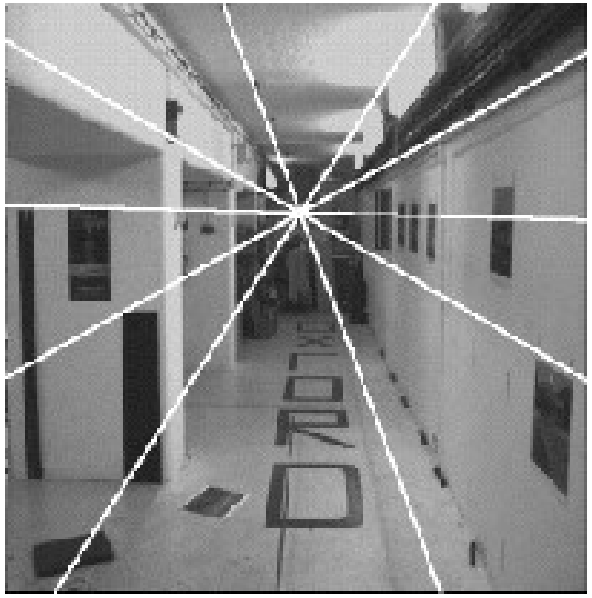


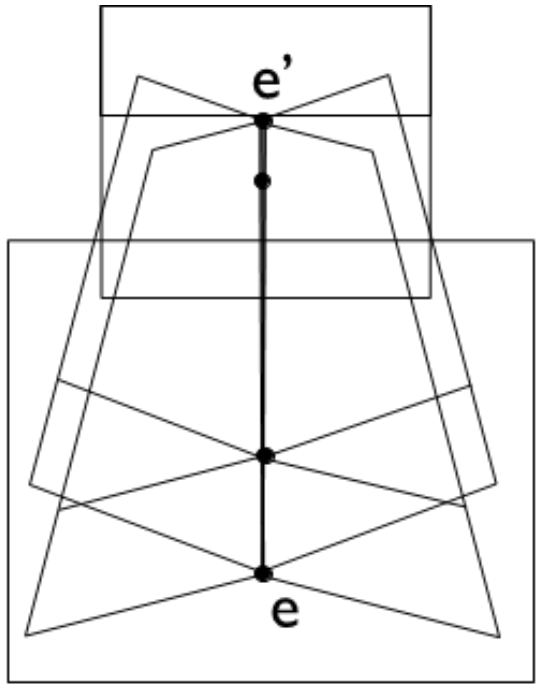
Where is the epipole?



Example: Forward Motion

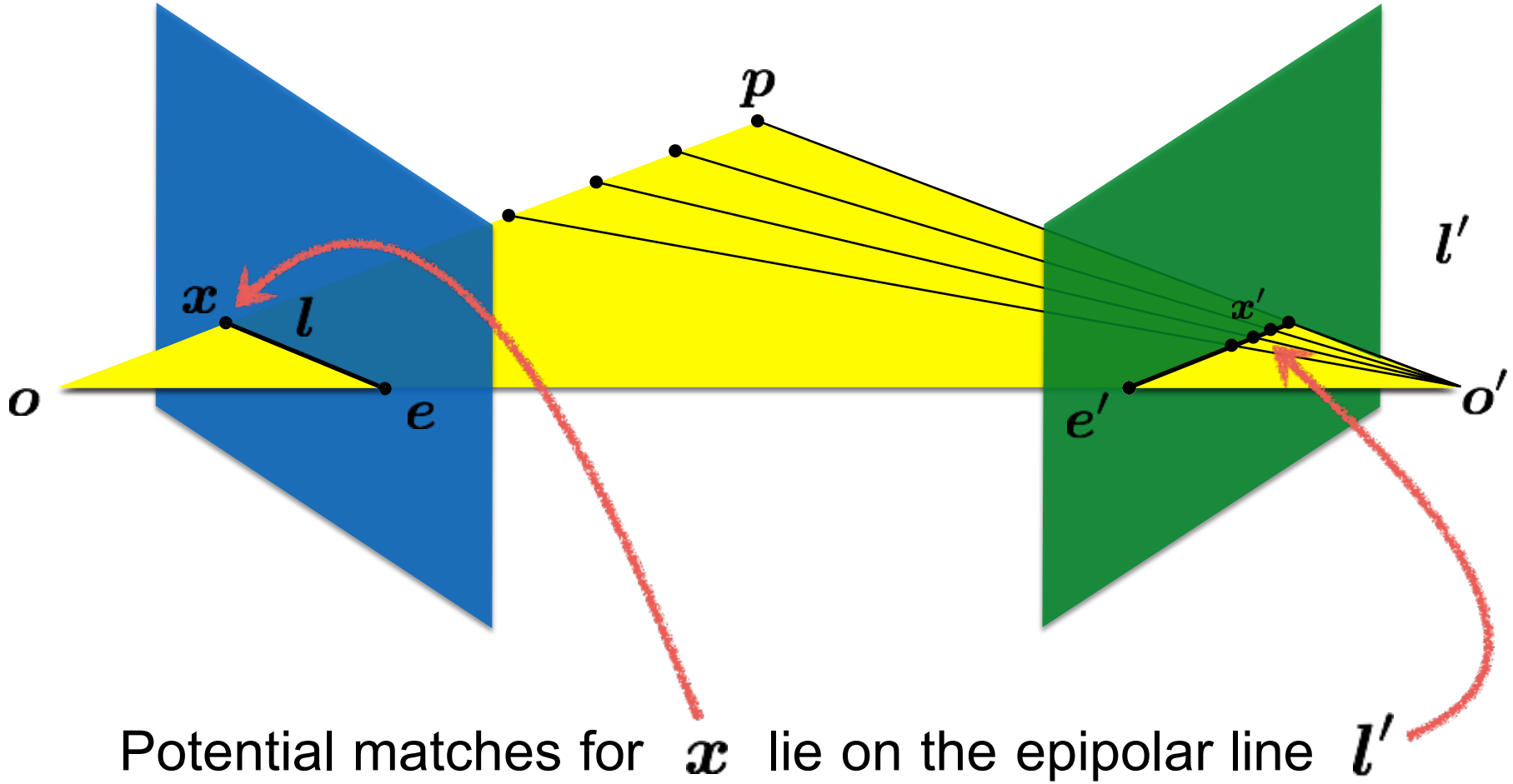






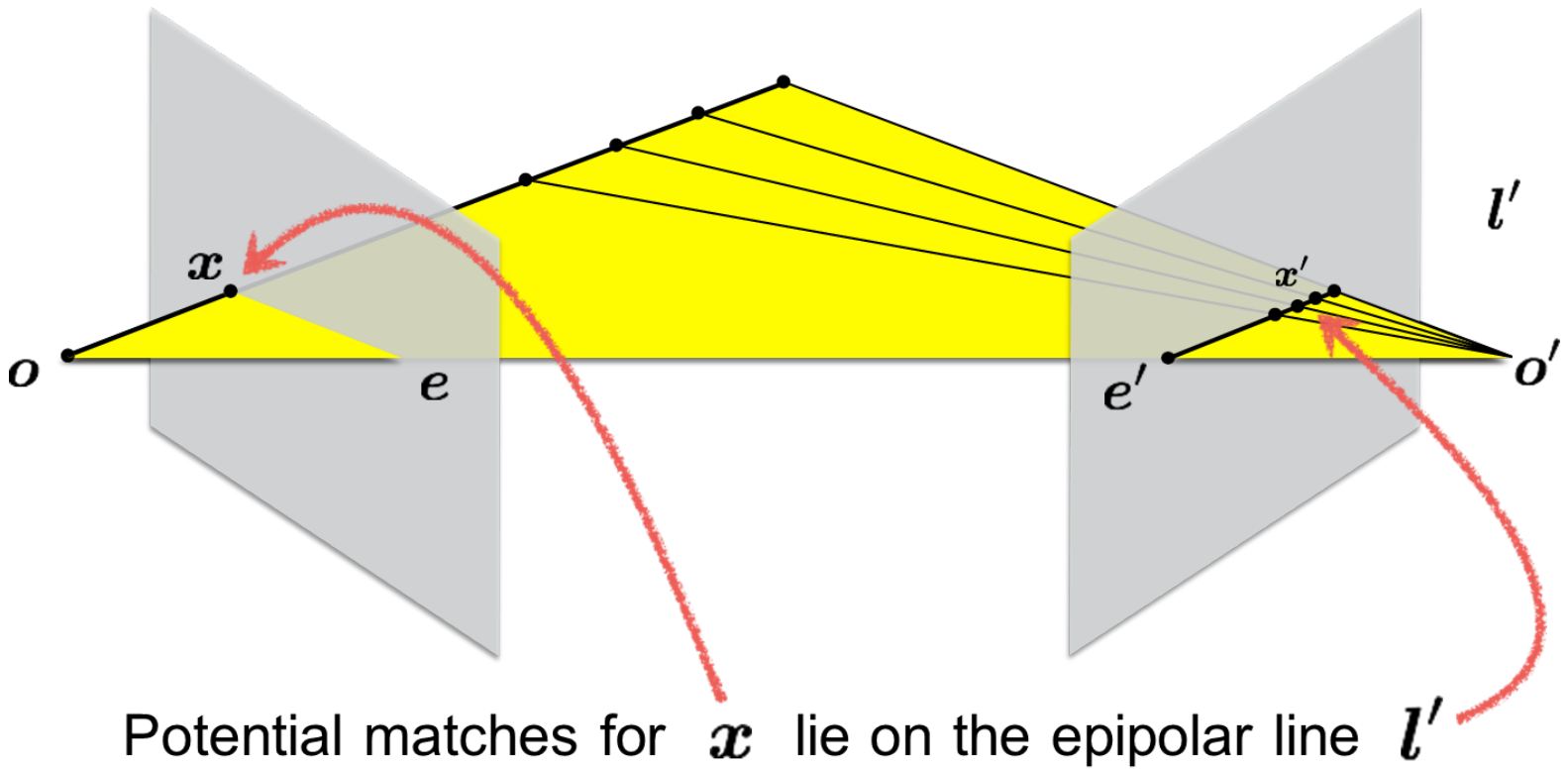
Epipole has same coordinates in both images. Points move along lines radiating from e:“Focus of expansion”

Epipolar Constraint



Potential matches for x lie on the epipolar line l'

What is this useful for?

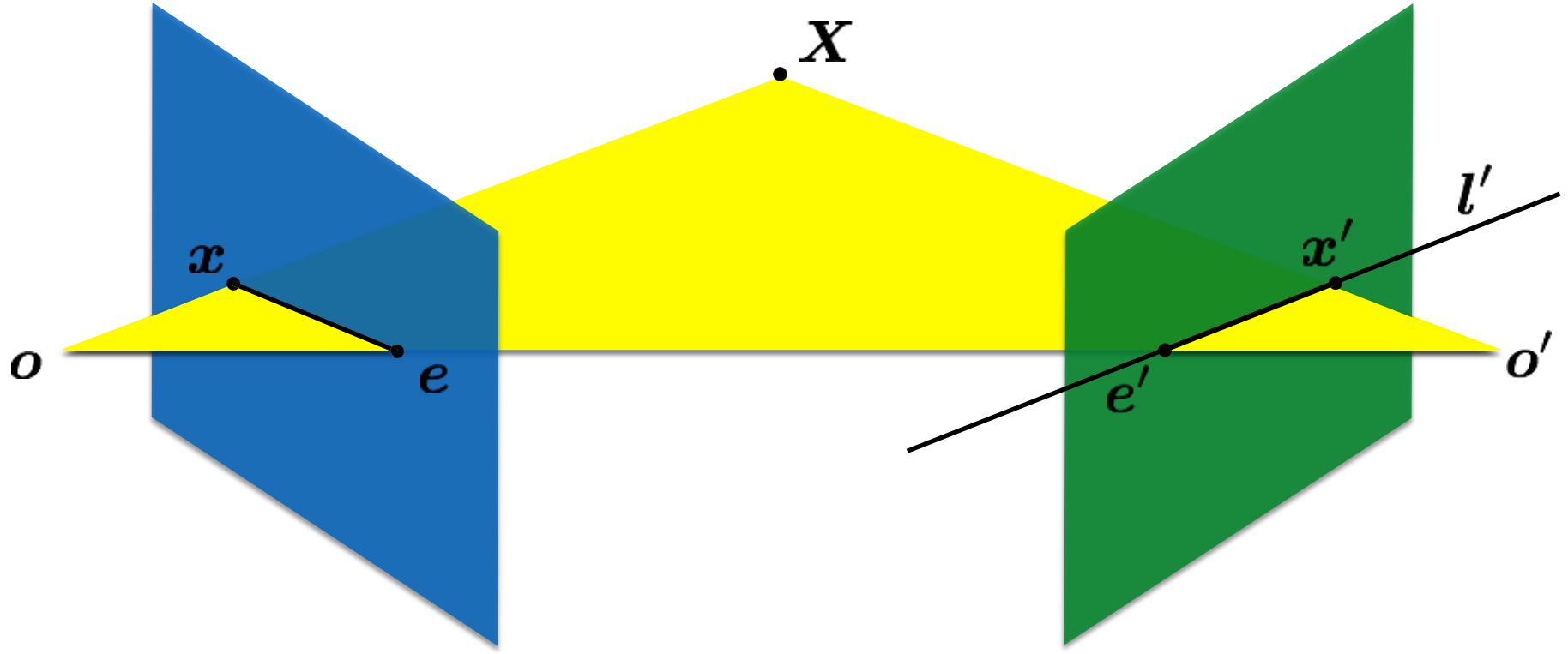


- If I know x, and have calibrated cameras (known intrinsics K, K' and extrinsic matrix), I can restrict x' to be along l'
- If we have enough x, x' correspondences,

Essential Matrix

Given a point in one image, multiplying by the **essential matrix** will tell us the **epipolar line** in the second view.

$$\mathbf{E}\boldsymbol{x} = \boldsymbol{l}'$$



Motivation

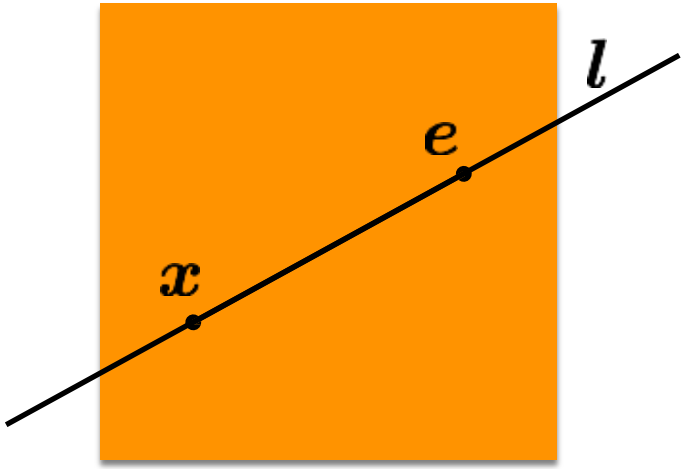
The Essential Matrix is a 3 x 3 matrix that encodes **epipolar geometry**

Given a point in one image, multiplying by the **essential matrix** will tell us the **epipolar line** in the second image.

We can estimate relative position & orientation between the cameras (and the 3D position of corresponding image points) using **essential matrix**

Epipolar Line

$$ax + by + c = 0$$ in vector form $$\boldsymbol{l} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

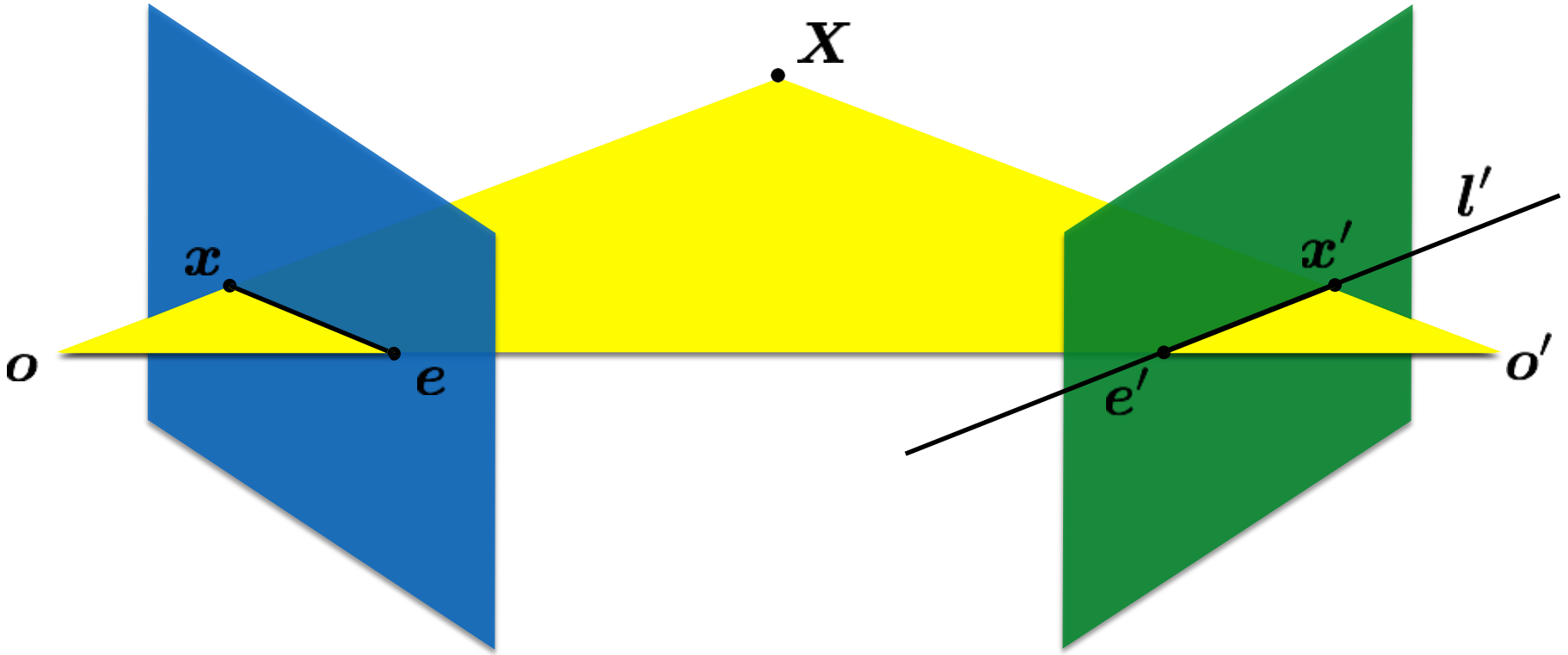


If the point $\boldsymbol{x}$ is on the epipolar line $\boldsymbol{l}$ then

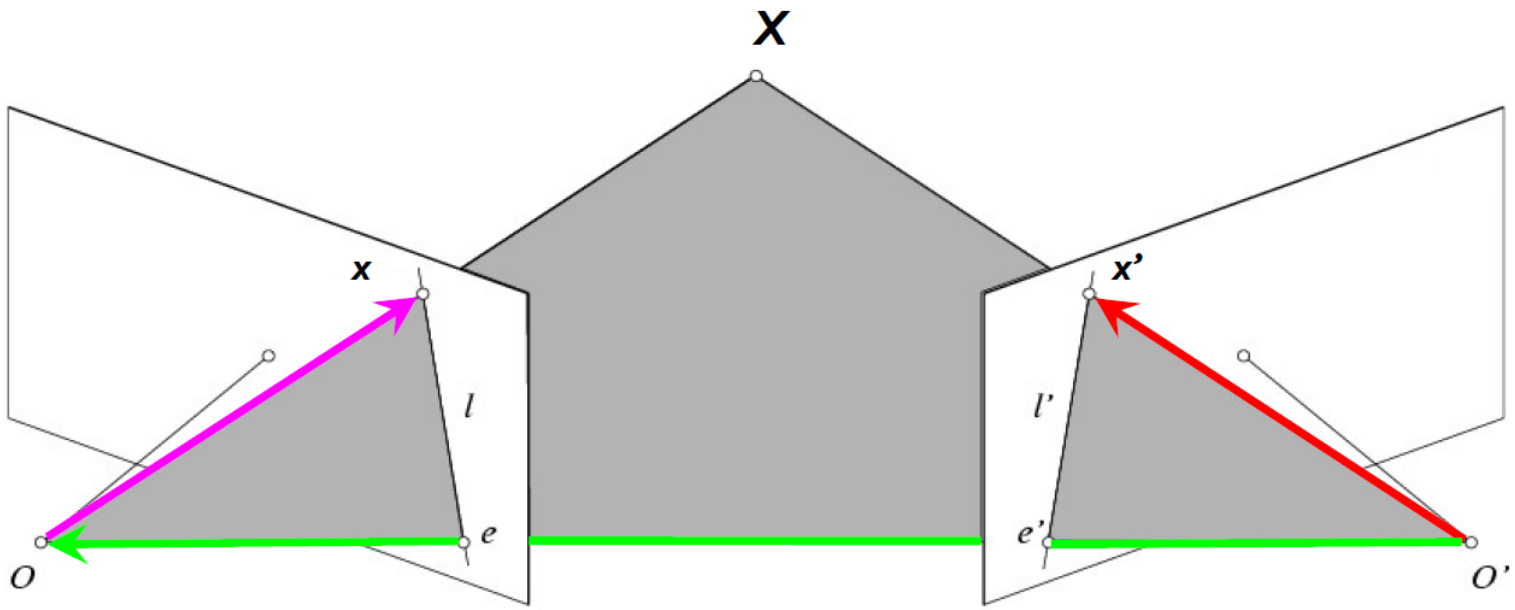
$$\boldsymbol{x}^\top \boldsymbol{l} = 0$$

So if $\boldsymbol{x}'^{\top}\boldsymbol{l}'=0$ and $\mathbf{E}\boldsymbol{x}=\boldsymbol{l}'$ then

$$\boxed{\boldsymbol{x}'^{\top}\mathbf{E}\boldsymbol{x}=0}$$



Essential Matrix



$$\hat{x} \cdot [t \times (R\hat{x}')] = 0 \quad \Rightarrow \quad \hat{x}^T E \hat{x}' = 0 \quad \text{with} \quad E = [t]_\times R$$

Essential Matrix
(Longuet-Higgins, 1981)

Note: $[t]_\times$ is matrix representation of cross product

- E is a 3x3 matrix which relates corresponding pairs of (homogeneous) image points across pairs of images (for calibrated cameras)
- Is used to estimates relative position/orientation

Properties of Essential Matrix

Longuet-Higgins equation $\boldsymbol{x}'^{\top}\mathbf{E}\boldsymbol{x}=0$

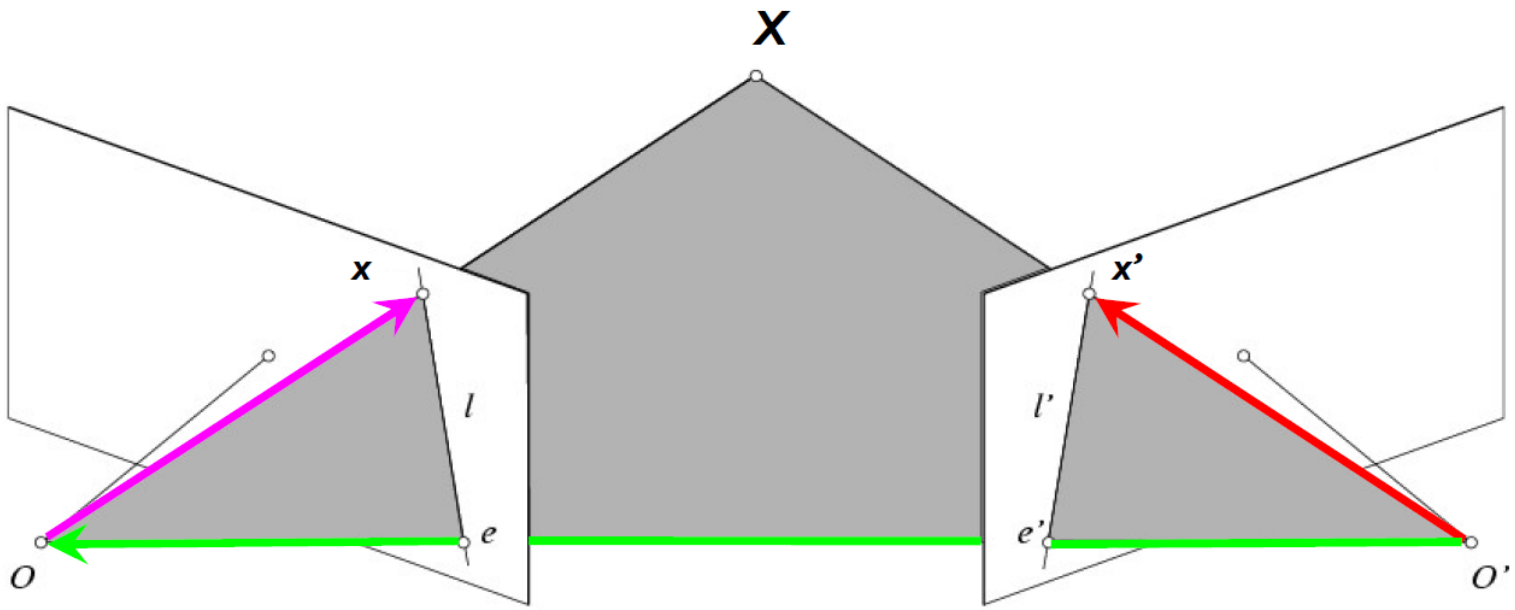
Epipolar lines $\boldsymbol{x}^{\top}\boldsymbol{l}=0$ $\quad \boldsymbol{x}'^{\top}\boldsymbol{l}'=0$

$\boldsymbol{l}'=\mathbf{E}\boldsymbol{x}$ $\quad \boldsymbol{l}=\mathbf{E}^{T}\boldsymbol{x}'$

Epipoles $\boldsymbol{e}'^{\top}\mathbf{E}=\mathbf{0}$ $\quad \mathbf{E}\boldsymbol{e}=\mathbf{0}$

2D points are expressed in camera coordinate system

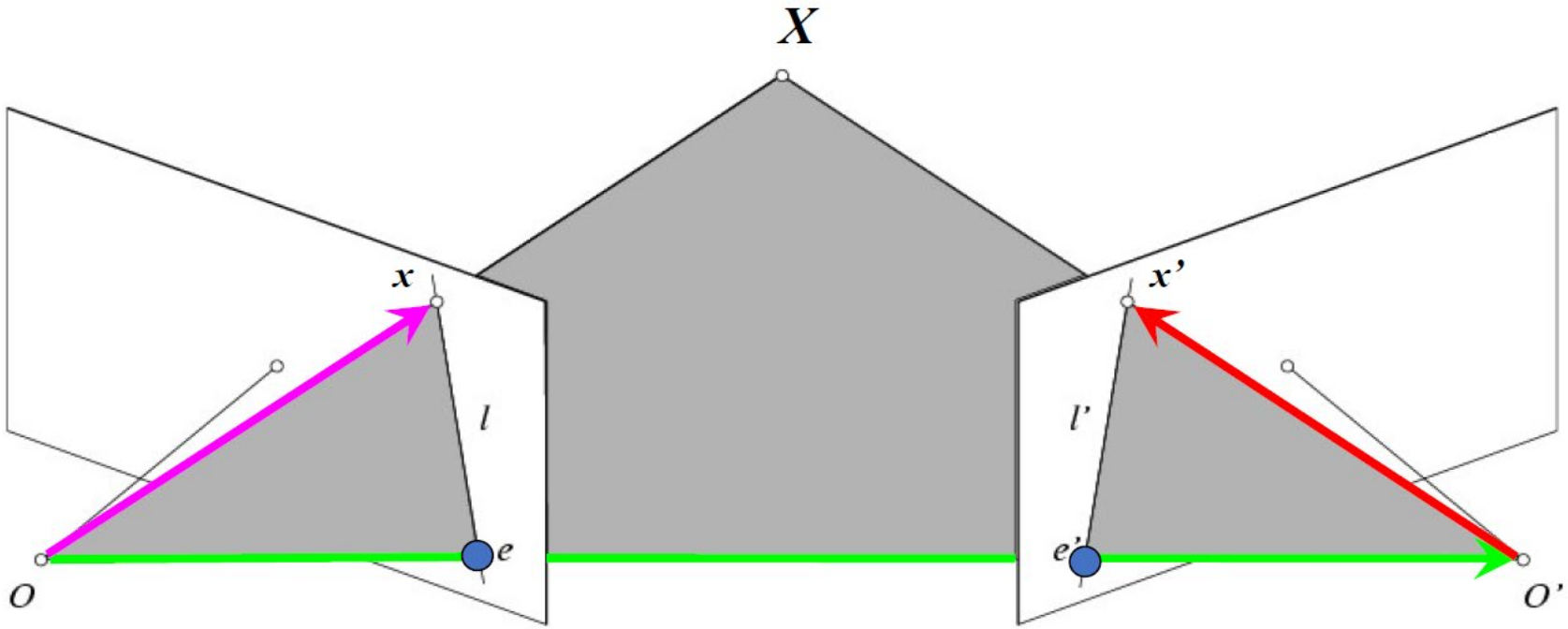
Additional Slide: Properties of the Essential matrix



$$\hat{x} \cdot [t \times (R\hat{x}')] = 0 \quad \Rightarrow \quad \hat{x}^T E \hat{x}' = 0 \quad \text{with} \quad E = [t]_\times R$$

- $E\ x'$ is the epipolar line associated with x' ($l = E\ x'$)
- $E^T x$ is the epipolar line associated with x ($l' = E^T x$)
- $E\ e' = 0$ and $E^T e = 0$
- E is singular (rank two)
- E has five degrees of freedom: (3 for R, 2 for t because it's up to a scale)

Epipolar Constraint: Uncalibrated Setting



If we don't know intrinsics K and K', then we can write the epipolar constraint in terms of unknown normalized coordinates:

$$\hat{x}^T E \hat{x}' = 0 \qquad x = K\hat{x}, \quad x' = K'\hat{x}'$$

The Fundamental Matrix

Without knowing K and K', we can define a similar relation using unknown normalized coordinates

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

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

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

$$\Rightarrow \quad x^T F x' = 0 \quad \text{with} \quad F = K^{-T} E K'^{-1}$$

Fundamental Matrix
(Faugeras and Luong, 1992)

Fundamental Matrix

Without knowing K and K', we can define a similar relation using unknown normalized coordinates

$$\hat{x}'^{\top} \mathbf{E} \hat{x} = 0$$

The essential matrix operates on image points expressed in **2D coordinates** expressed in the camera coordinate system

$$\hat{x}' = \mathbf{K}'^{-1} x' \qquad \hat{x} = \mathbf{K}^{-1} x$$

$\hat{x}'$: camera point; x': image point

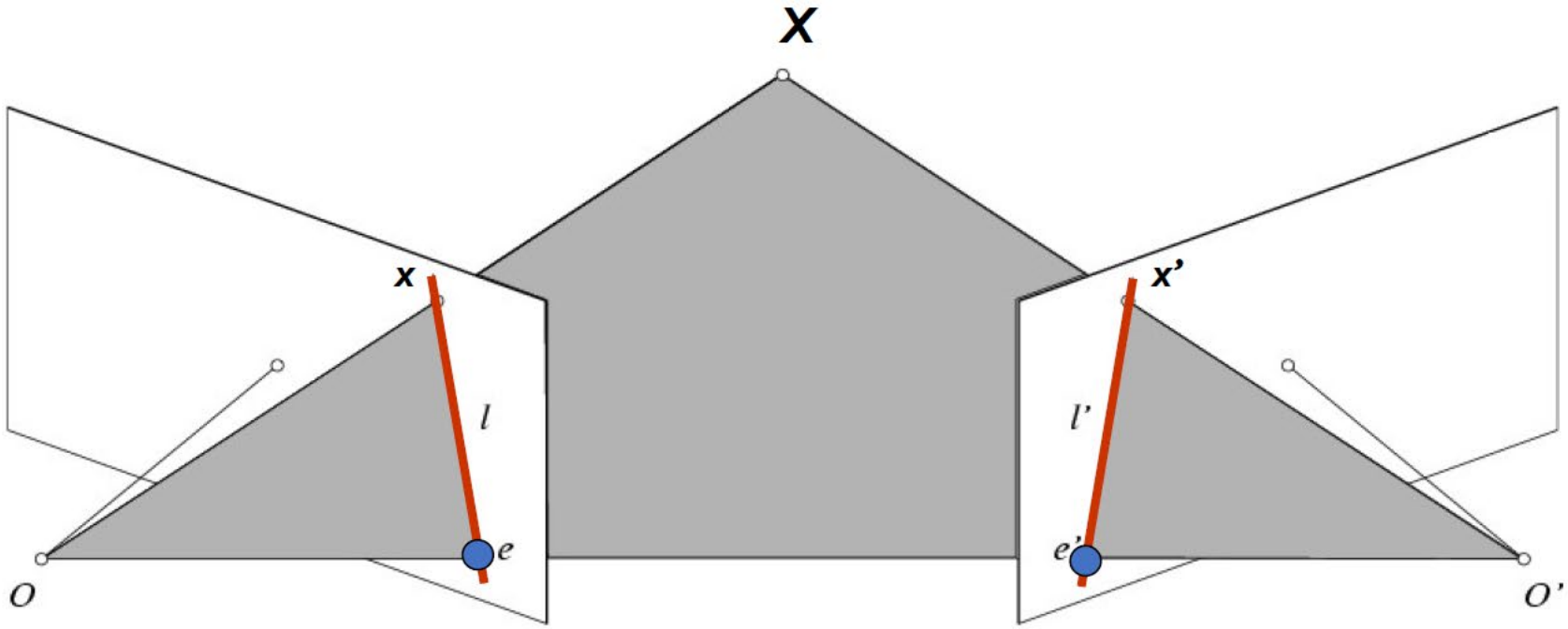
Writing out the epipolar constraint in terms of image coordinates

$$\mathbf{K}'^{-\top} \mathbf{E} \mathbf{K}^{-1} x = 0$$

$$x'^{\top} (\mathbf{K}'^{-\top} \mathbf{E} \mathbf{K}^{-1}) x = 0$$

$$x'^{\top} \mathbf{F} x = 0$$

Properties of the Fundamental matrix



$$x^T F x' = 0 \quad \text{with} \quad F = K^{-T} E K'^{-1}$$

- $F\ x' = 0$ is the epipolar line l associated with x'
- $F^T x = 0$ is the epipolar line l' associated with x
- F is singular (rank two): $\det(F) = 0$
- $F\ e' = 0$ and $F^T e = 0$ (nullspaces of $F = e'$; nullspace of $F^T = e'$)
- F has seven degrees of freedom: 9 entries but defined up to scale, $\det(F) = 0$

Additional Slide: F in more detail

- F is a 3x3 matrix
- F is Rank 2: one column is a linear combination of the other two

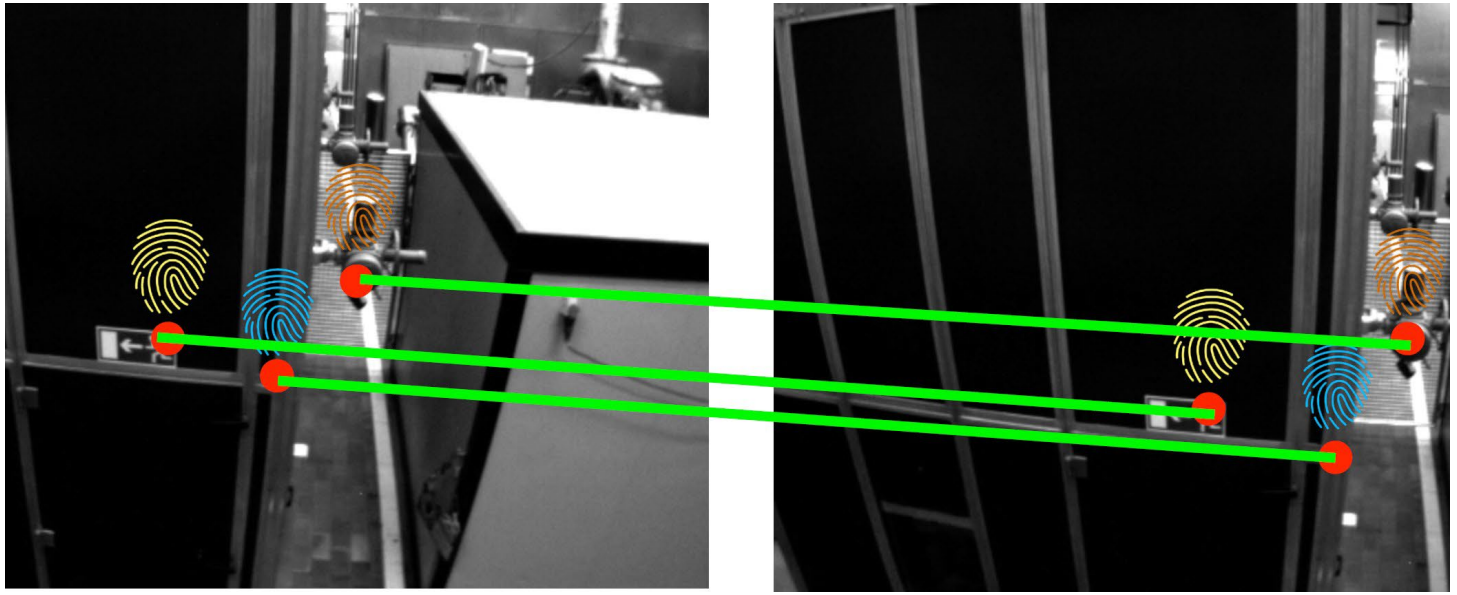
$$\begin{bmatrix} a & b & \alpha a + \beta b \\ c & d & \alpha c + \beta d \\ e & f & \alpha e + \beta f \end{bmatrix}$$

- Determined up to scale
- 7 degrees of freedom
- Given x projected from X into image 1, F constrains the projection of x' into image 2 to an epipolar line

Fundamental Matrix

$$\boxed{\boldsymbol{x}_m'^{\top}\mathbf{F}\boldsymbol{x}_m = 0}$$

$$\begin{bmatrix} x'_m & y'_m & 1 \end{bmatrix} \begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix} \begin{bmatrix} x_m \\ y_m \\ 1 \end{bmatrix} = 0$$



Estimating the Fundamental Matrix

- **8-point algorithm**
  - Least squares solution using SVD on equations from 8 pairs of correspondences
  - Enforce $\det(F) = 0$ constraint using SVD on F

$$\begin{bmatrix} x_1x'_1 & x_1y'_1 & x_1 & y_1x'_1 & y_1y'_1 & y_1 & x'_1 & y'_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_Mx'_M & x_My'_M & x_M & y_Mx'_M & y_My'_M & y_M & x'_M & y'_M & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \end{bmatrix} = \mathbf{0}$$

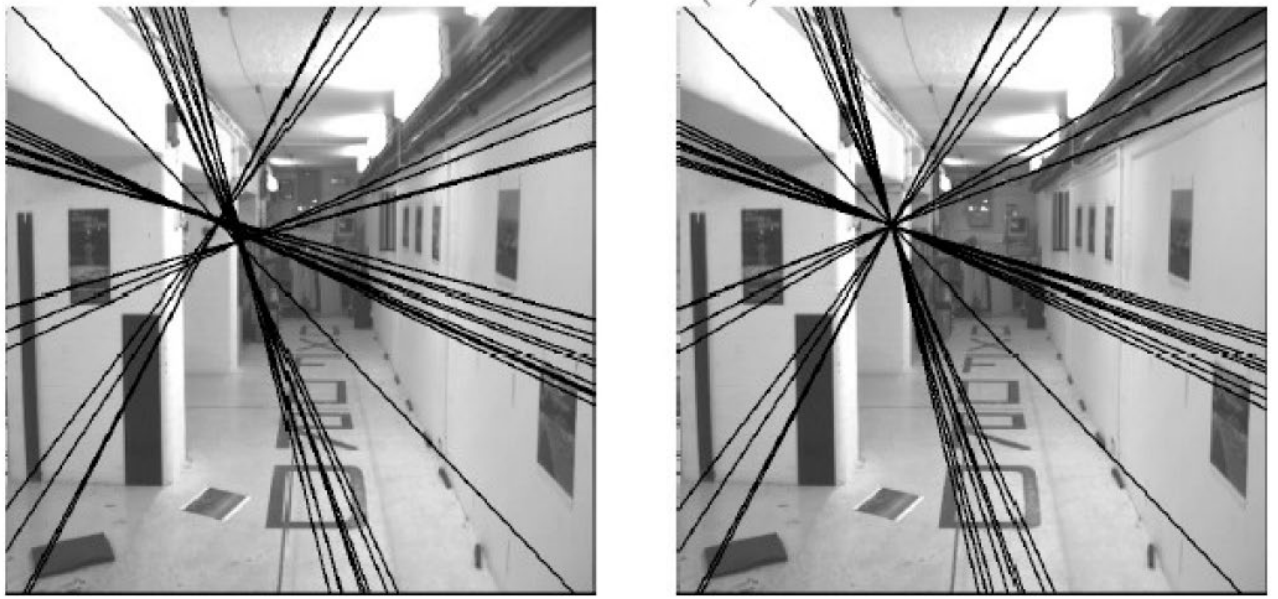
8-Point Algorithm

- Solve a system of homogeneous linear equations
  - Write down the system of equations
  - Solve f from $Af = 0$ using SVD

```
Python Numpy:
U, S, Vh = np.linalg.svd(A)
F = Vh[-1,:]
F = np.reshape(F, (3,3))
```

8-Point Algorithm

Fundamental matrix has rank 2 : $\det(\mathtt{F}) = 0$.



Left : Uncorrected $\mathtt{F}$ – epipolar lines are not coincident.

Right : Epipolar lines from corrected $\mathtt{F}$.

8-Point Algorithm

- Solve a system of homogeneous linear equations
  - Write down the system of equations
  - Solve f from $Af = 0$ using SVD

- Resolve $\det(F) = 0$ constraint using SVD

• Resolve $\det(F) = 0$ constraint using SVD

```
Python Numpy:
U, S, Vh = np.linalg.svd(F)
S[-1] = 0
F = U @ np.diagflat(S) @ Vh
```

@ operator = matrix multiplication

Decompose E to get R, t

SVD: $\mathbf{E} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$

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

We get FOUR solutions:

$\mathbf{R}_1 = \mathbf{U}\mathbf{W}\mathbf{V}^\top \quad \mathbf{R}_2 = \mathbf{U}\mathbf{W}^\top\mathbf{V}^\top$

two possible rotations

$\mathbf{T}_1 = U_3 \quad \mathbf{T}_2 = -U_3$

two possible translations

We get FOUR solutions

$$\mathbf{R}_1 = \mathbf{U}\mathbf{W}\mathbf{V}^\top \quad \mathbf{T}_1 = U_3$$

$$\mathbf{R}_1 = \mathbf{U}\mathbf{W}\mathbf{V}^\top \quad \mathbf{T}_2 = -U_3$$

$$\mathbf{R}_2 = \mathbf{U}\mathbf{W}^\top\mathbf{V}^\top \quad \mathbf{T}_2 = -U_3$$

$$\mathbf{R}_2 = \mathbf{U}\mathbf{W}^\top\mathbf{V}^\top \quad \mathbf{T}_1 = U_3$$

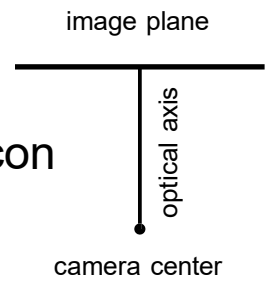
Which one do we choose? Chirality constraints:

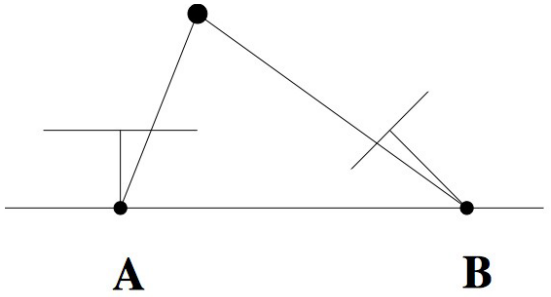
- Compute determinant of R, valid solution must be equal to 1
  (note: det(R) = -1 means rotation and reflection)
- Compute 3D point using triangulation, valid solution has positive depth value
  (Note: negative depth means point is behind the camera)

Correct Solution

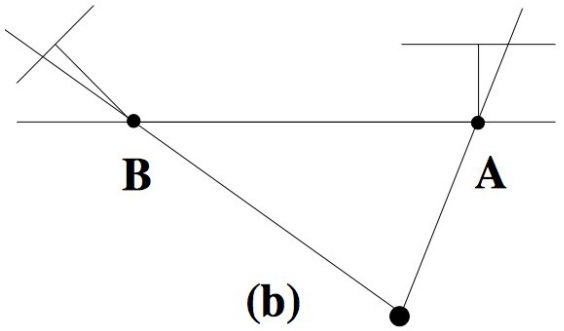
The configuration where Points are in front of cameras

Camera Icon

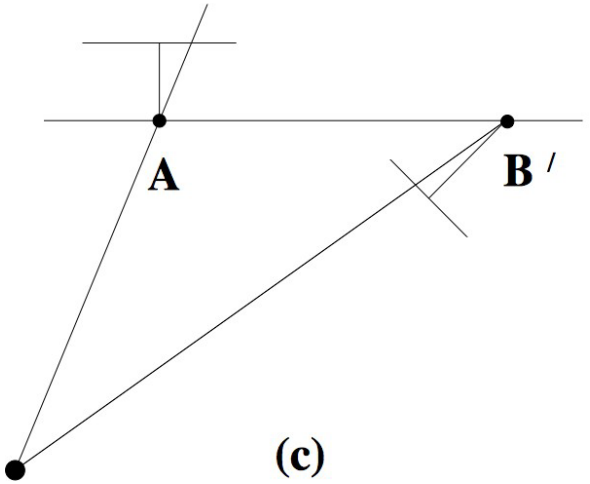




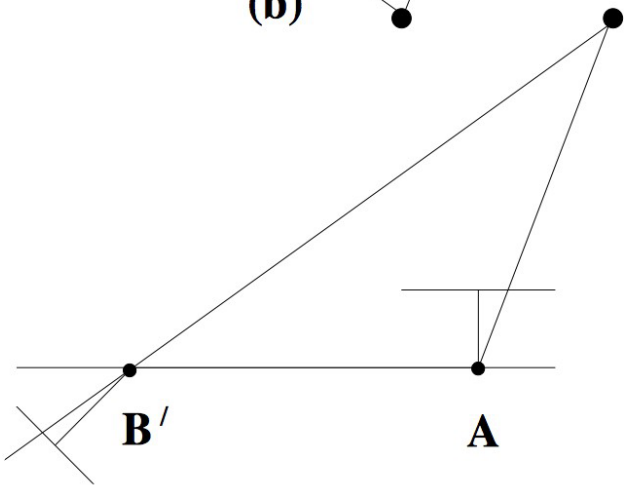
(a)



(b)



(c)

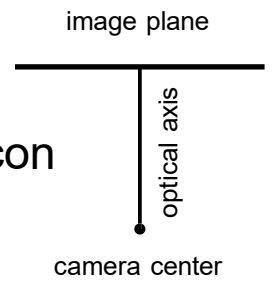


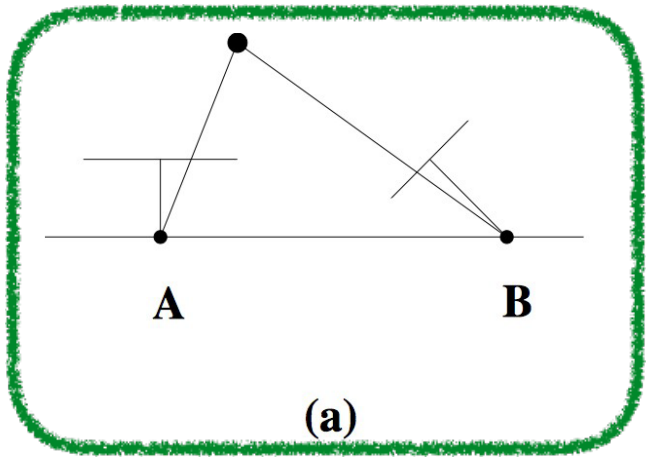
(d)

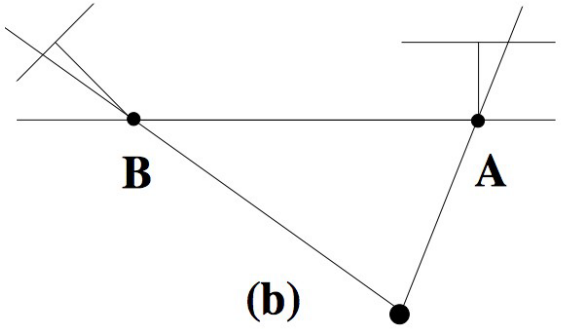
Correct Solution

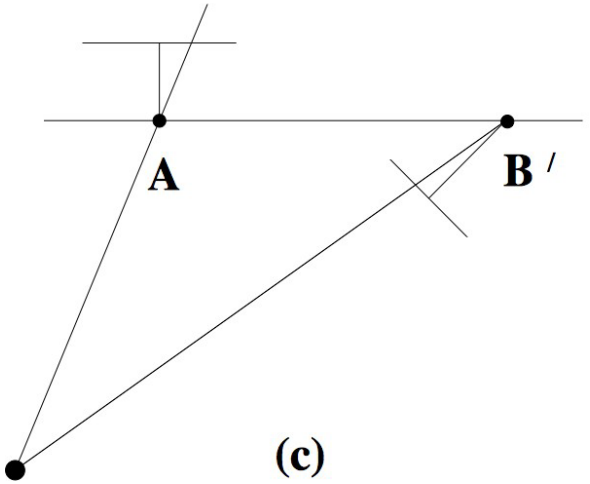
The configuration where Points are in front of cameras

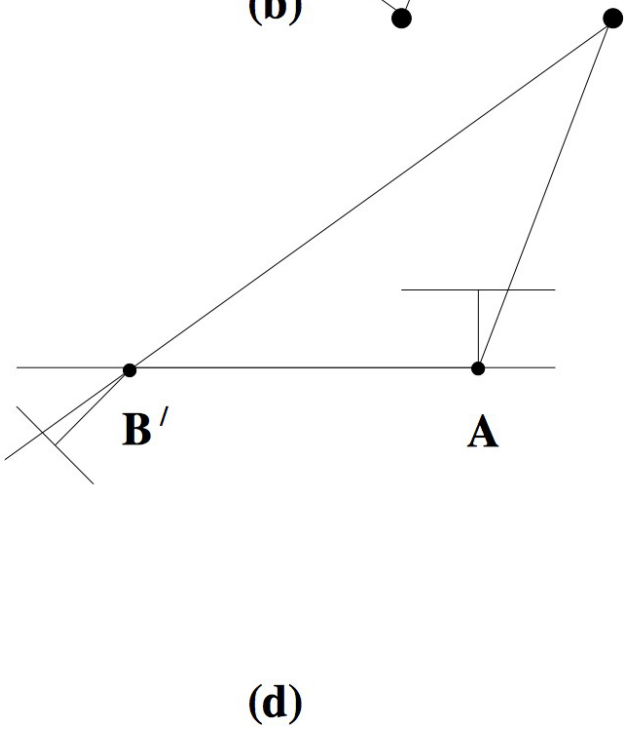
Camera Icon





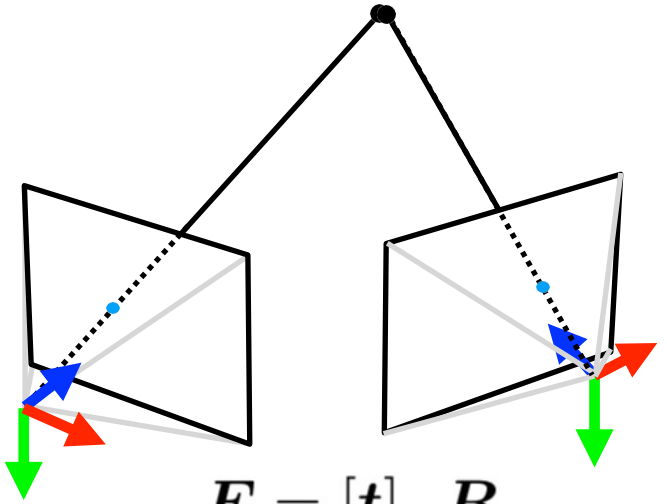






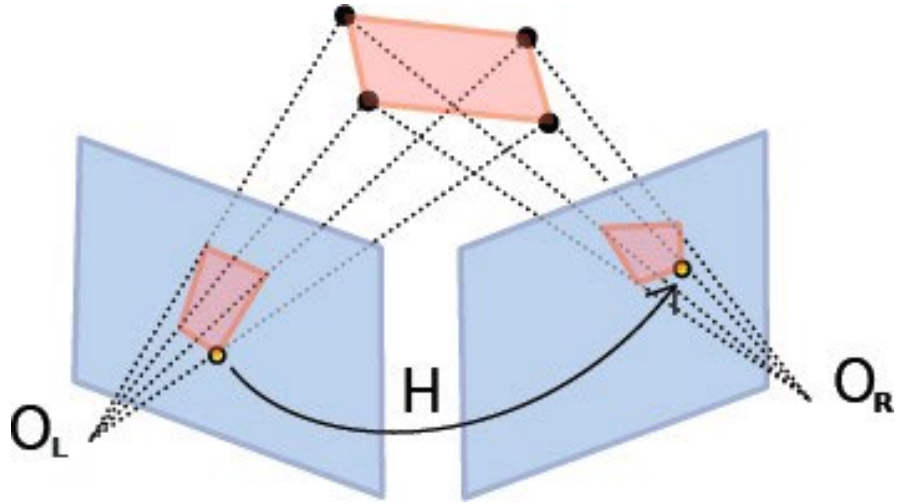
8-point Method: Limitations

Number of correspondences:
do we really need 8 points?



Scene structures:
There are certain configurations of
3D points that make the algorithm fail

Parallax: what if $t = 0$?



Homography

We can use homographies when…

- The scene is planar

- The scene is very far or has small (relative) depth variation → scene is approximately planar

- The scene is captured under camera rotation only (no translation)

Panorama: stitching images from different viewpoints

Use image homographies:

Homography Matrix

Given matched image points (P, P'), the homography matrix H relates

$$P' = H \cdot P$$

Note: Homography holds only for co-planar points or rotation-only motion

Estimating the Homography Matrix

From expanding the homography equation for each point correspondence:

$$P' = H \cdot P \quad \text{or} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \alpha \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

We obtain the matrix-vector form:

$$\mathbf{A}_i \boldsymbol{h} = \mathbf{0}$$

$$\mathbf{A}_i = \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix}$$

$$\boldsymbol{h} = \begin{bmatrix} h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 & h_8 & h_9 \end{bmatrix}^\top$$

Estimating the Homography Matrix

Stack together constraints from multiple point correspondences

$$\mathbf{A}\boldsymbol{h} = \mathbf{0}$$

$$\begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix}$$

$$\begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix}$$

$$\vdots$$

$$\begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix}$$

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Estimating the Homography Matrix

- **Direct Linear Transform (DLT)** algorithm
  - Create matrix $\boldsymbol{A}$ from point correspondences
  - Compute SVD: $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathbf{T}}$

```
Python Numpy:
U, S, Vh = np.linalg.svd(A)
```

  - Store singular vector of the smallest singular value $\boldsymbol{h} = \boldsymbol{v_i}$
  - Reshape to get $\boldsymbol{H}$

```
h = Vh[-1,:]
H = np.reshape(h, (3,3))
```

Decomposing Homography Matrix

The decomposition of $H = R - \frac{t\, n^T}{d}$ into R, t, n:

- SVD: $H = U\, \Sigma\, V^T$
- $R = U\, V^T$
- Let $M = H\ - R$
- SVD: $M = U_M\, \Sigma_M\, V_M^T$
- Let u, v be the left and right singular vectors, and σ be the corresponding singular value in SVD of M
- $t = \sigma d u,\ \ n = v$

Decomposing Homography Matrix

Example:

$$H = \begin{bmatrix} 1.1 & -0.2 & 0.3 \\ 0.1 & 0.9 & -0.1 \\ 0.05 & 0.1 & 1.0 \end{bmatrix}$$

$$H = U\Sigma V^T \Rightarrow U = \begin{bmatrix} -0.89 & 0.45 & -0.06 \\ -0.35 & -0.76 & -0.55 \\ -0.30 & -0.47 & 0.83 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} 1.23 & 0 & 0 \\ 0 & 0.91 & 0 \\ 0 & 0 & 0.78 \end{bmatrix}, \quad V^T = \begin{bmatrix} -0.92 & 0.38 & -0.06 \\ -0.38 & -0.91 & -0.15 \\ -0.06 & -0.15 & 0.99 \end{bmatrix}$$

$$R = UV^T \Rightarrow R = \begin{bmatrix} 0.99 & -0.10 & 0.09 \\ 0.10 & 0.99 & -0.03 \\ -0.09 & 0.04 & 0.99 \end{bmatrix}$$

$$M = H - R = \begin{bmatrix} 0.11 & -0.10 & 0.21 \\ 0.00 & -0.09 & -0.07 \\ 0.14 & 0.06 & 0.01 \end{bmatrix}$$

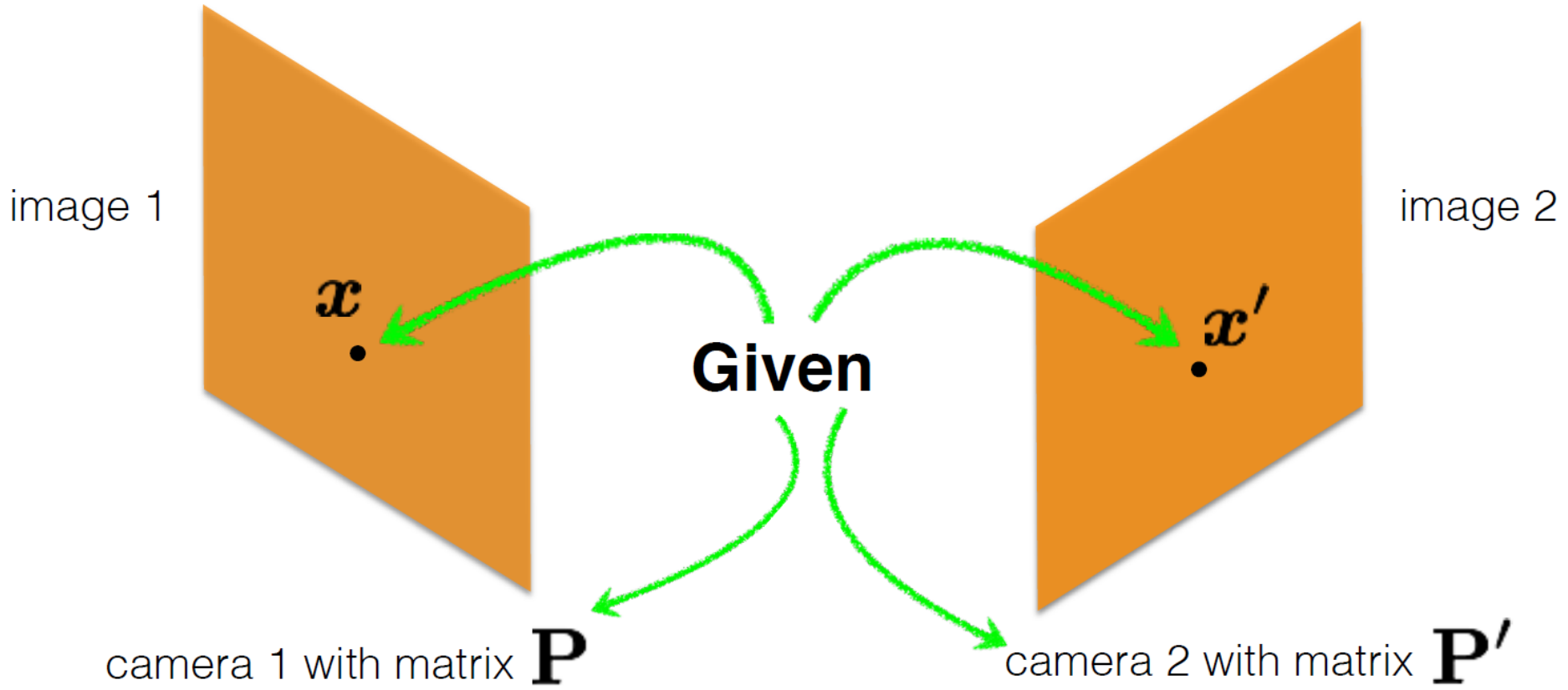
$$M = U_M \Sigma_M V_M^T \Rightarrow U_M = \begin{bmatrix} -0.82 & -0.57 & -0.07 \\ 0.30 & -0.52 & 0.80 \\ -0.49 & 0.64 & 0.59 \end{bmatrix}, \quad \Sigma_M = \begin{bmatrix} 0.29 & 0 & 0 \\ 0 & 0.12 & 0 \\ 0 & 0 & 0.01 \end{bmatrix}, \quad V_M^T = \begin{bmatrix} -0.47 & 0.87 & -0.16 \\ -0.88 & -0.46 & 0.12 \\ 0.05 & 0.19 & 0.98 \end{bmatrix}$$

$$\mathbf{t} = \sigma_1 \mathbf{u}_1 = 0.29 \begin{bmatrix} -0.82 \\ 0.30 \\ -0.49 \end{bmatrix} = \begin{bmatrix} -0.24 \\ 0.09 \\ -0.14 \end{bmatrix}$$

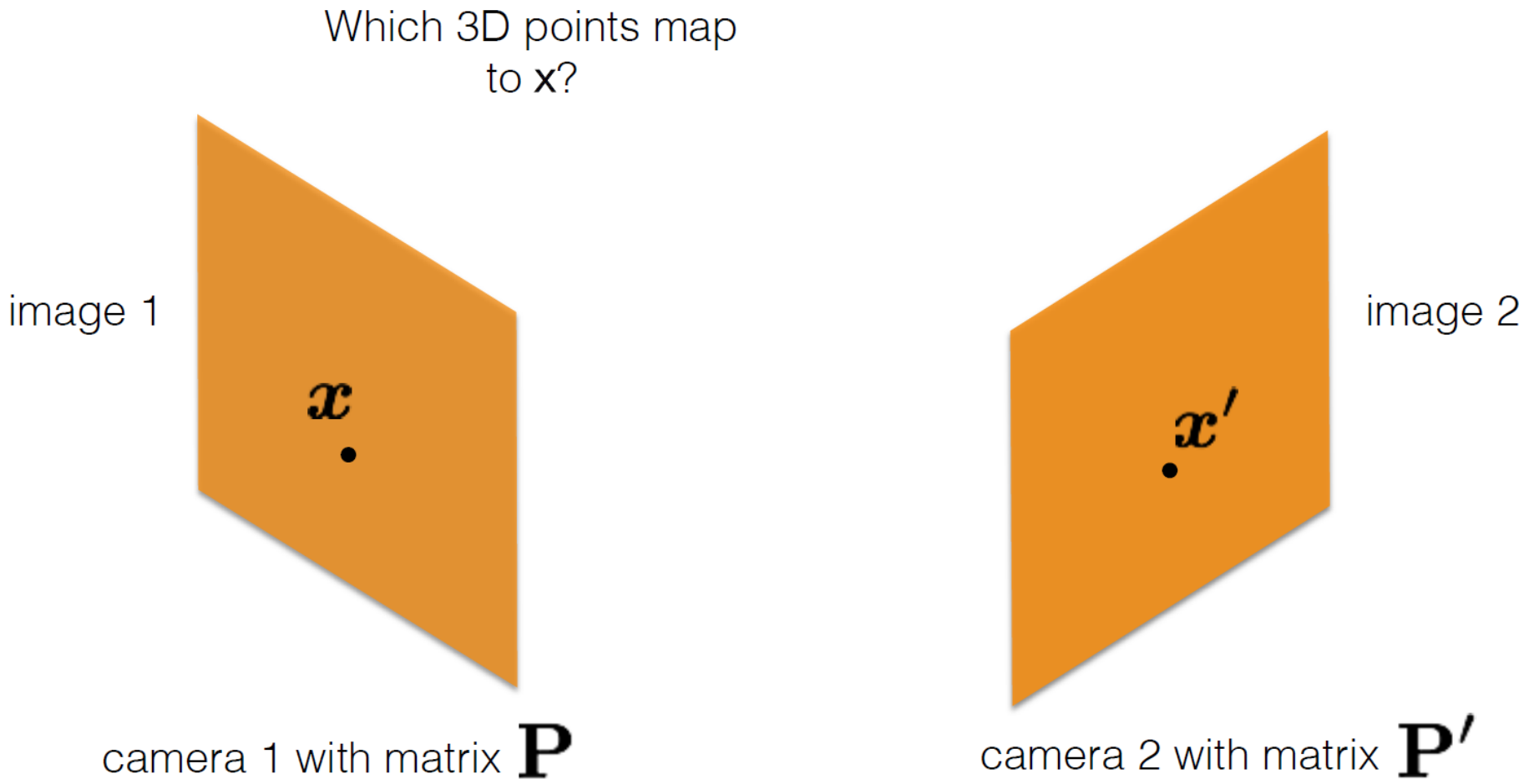
$$\mathbf{n} = \mathbf{v}_1 = \begin{bmatrix} -0.47 \\ -0.88 \\ 0.05 \end{bmatrix}$$

3D Reconstruction

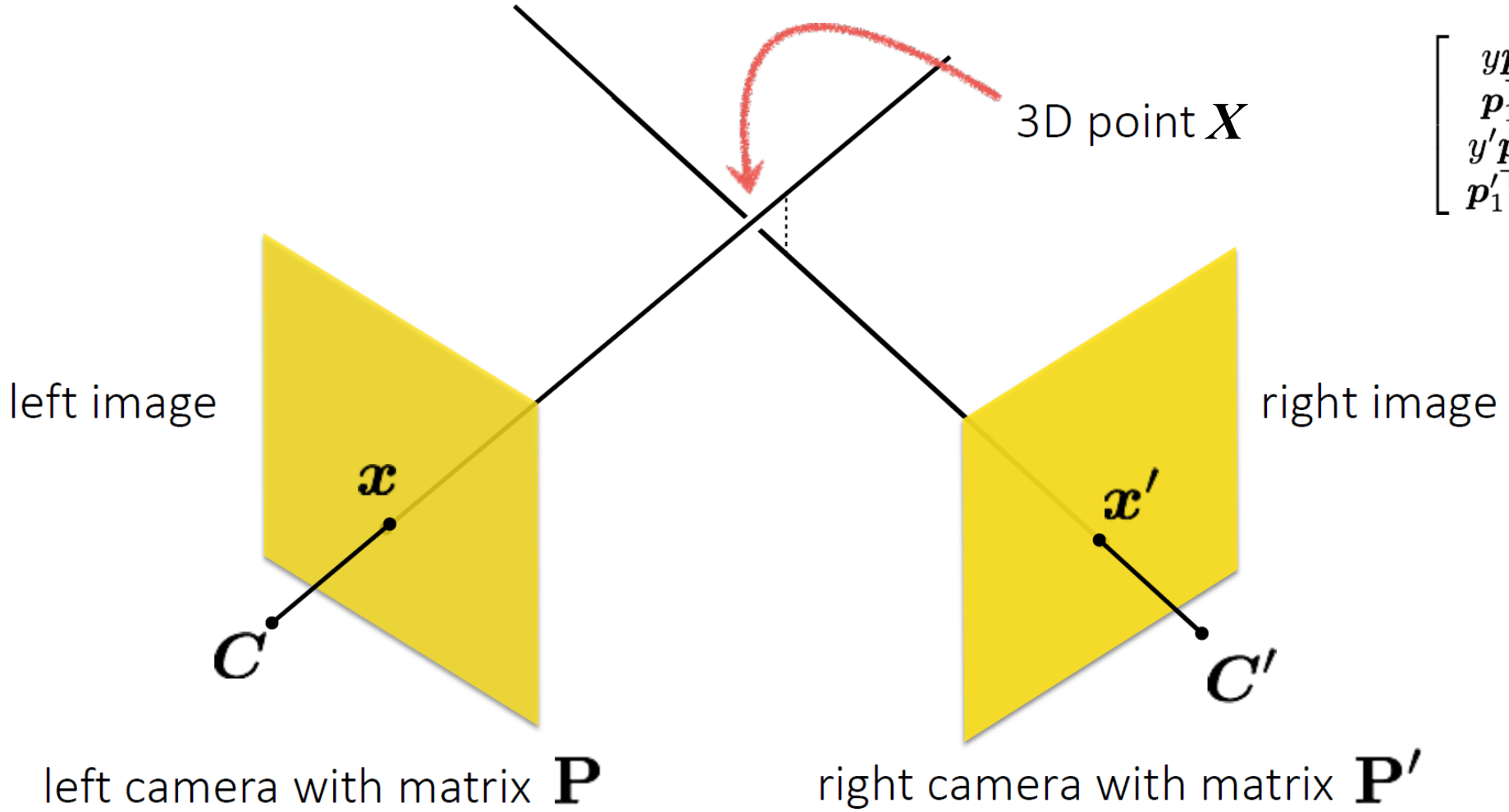
Triangulation



Triangulation



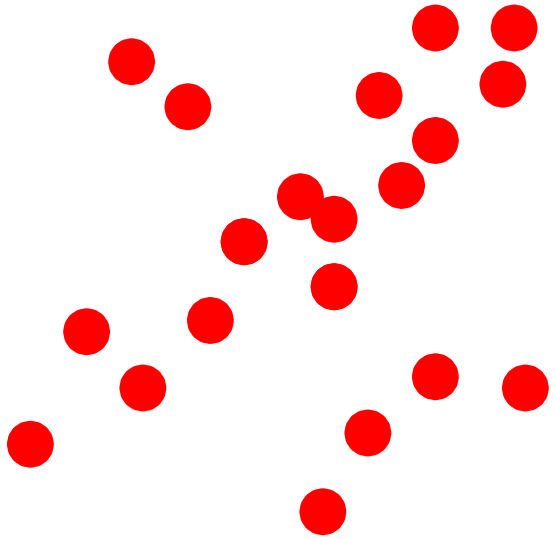
Triangulation



$$\begin{bmatrix} y\boldsymbol{p}_3^\top - \boldsymbol{p}_2^\top \\ \boldsymbol{p}_1^\top - x\boldsymbol{p}_3^\top \\ y'\boldsymbol{p}_3'^\top - \boldsymbol{p}_2'^\top \\ \boldsymbol{p}_1'^\top - x'\boldsymbol{p}_3'^\top \end{bmatrix} \boldsymbol{X} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Random Sample Consensus (RANSAC)

Fitting lines (with outliers)

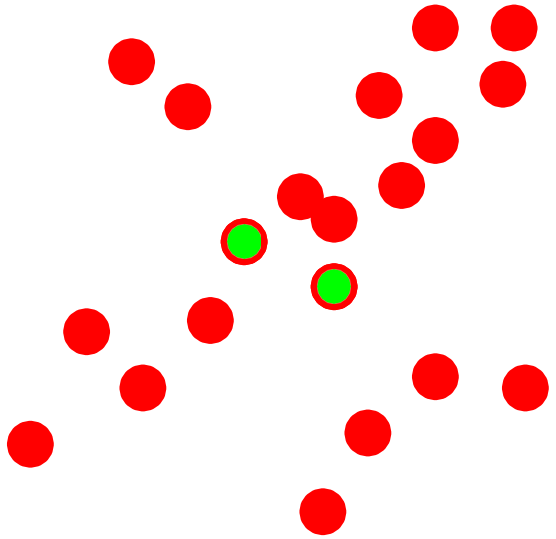


Algorithm:

1. Sample (randomly) the number of points required to fit the model
2. Solve for model parameters using samples
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Fitting lines (with outliers)

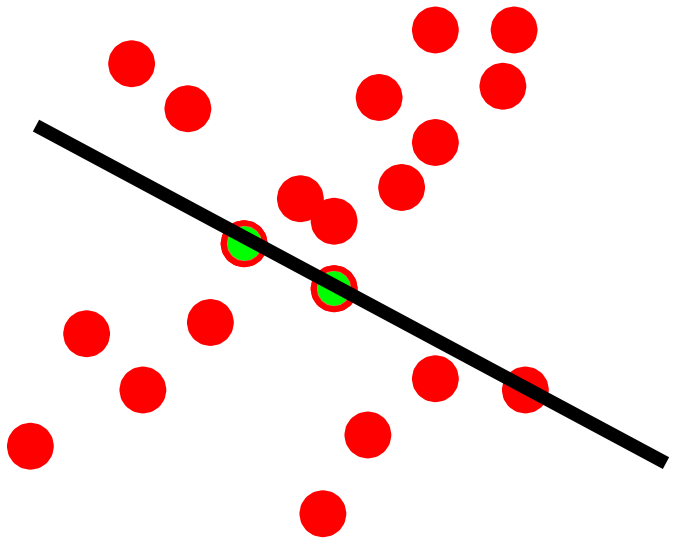


Algorithm:

1. **Sample (randomly) the number of points required to fit the model**
2. Solve for model parameters using samples
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Fitting lines (with outliers)

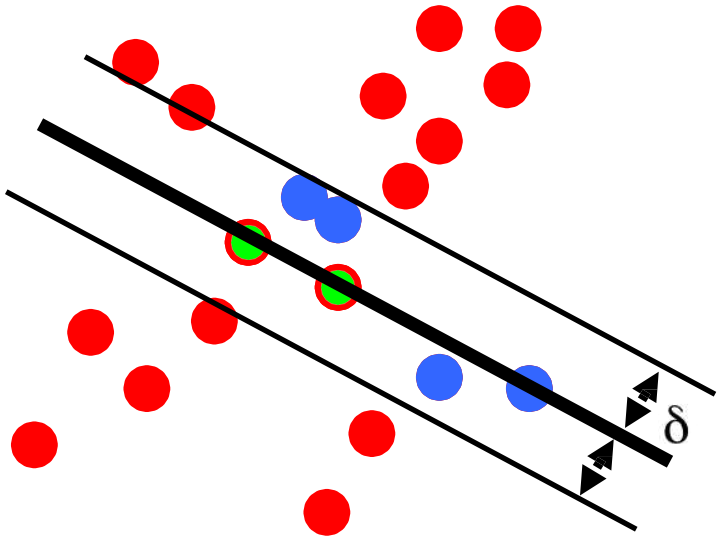


Algorithm:

1. Sample (randomly) the number of points required to fit the model
2. **Solve for model parameters using samples**
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Fitting lines (with outliers)



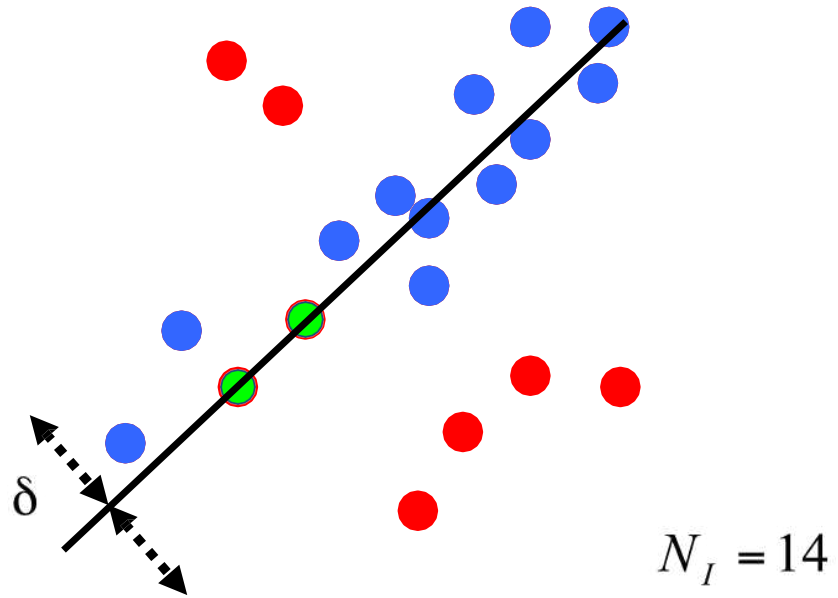
$N_I = 6$

Algorithm:

1. Sample (randomly) the number of points required to fit the model
2. Solve for model parameters using samples
3. **Score by the fraction of inliers within a preset threshold of the model**

Repeat 1-3 until the best model is found with high confidence

Fitting lines (with outliers)



Algorithm:

1. Sample (randomly) the number of points required to fit the model
2. Solve for model parameters using samples
3. Score by the fraction of inliers within a preset threshold of the model

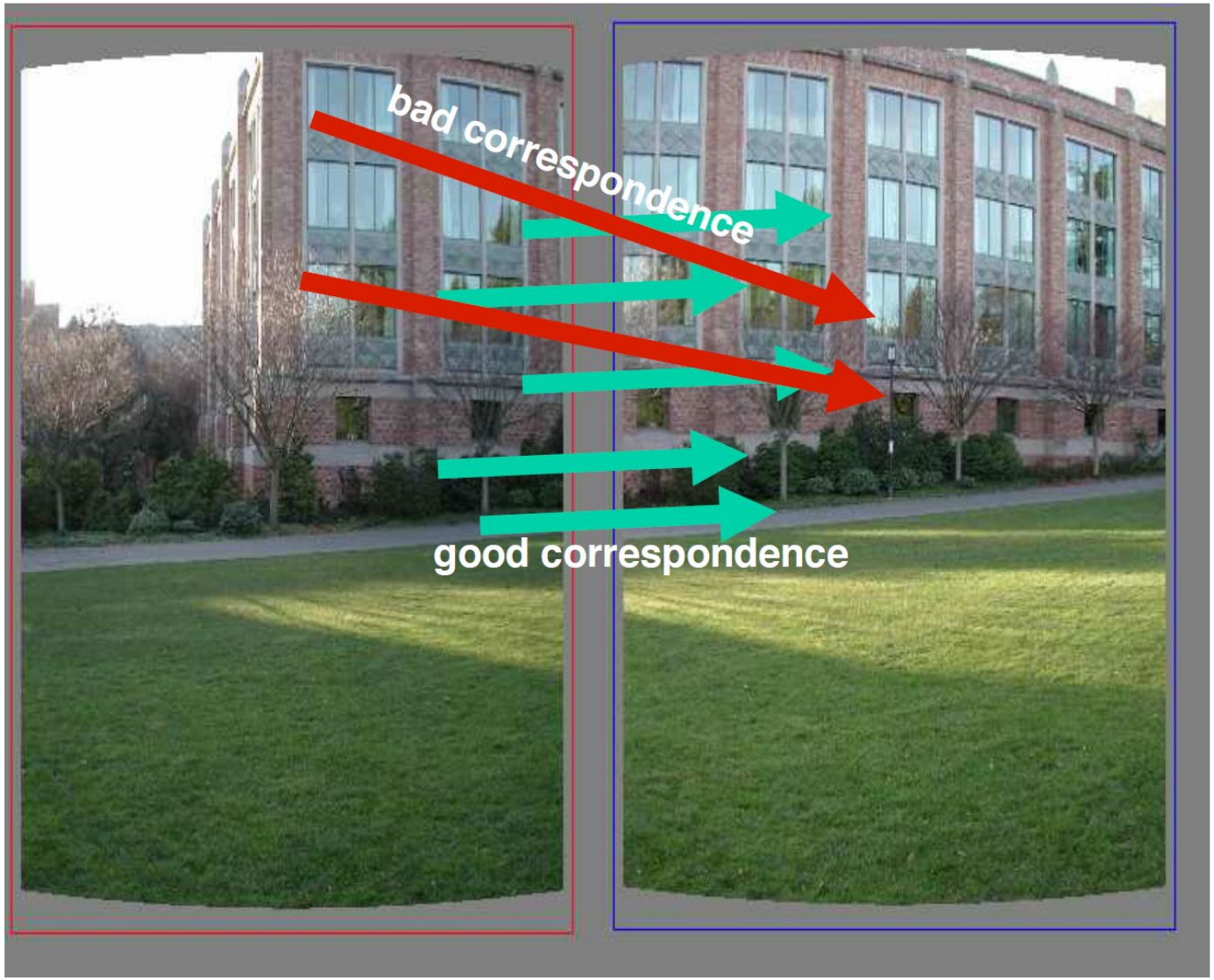
Repeat 1-3 until the best model is found with high confidence

Example

Given two images, find matching features (e.g., SIFT) and a translation transform

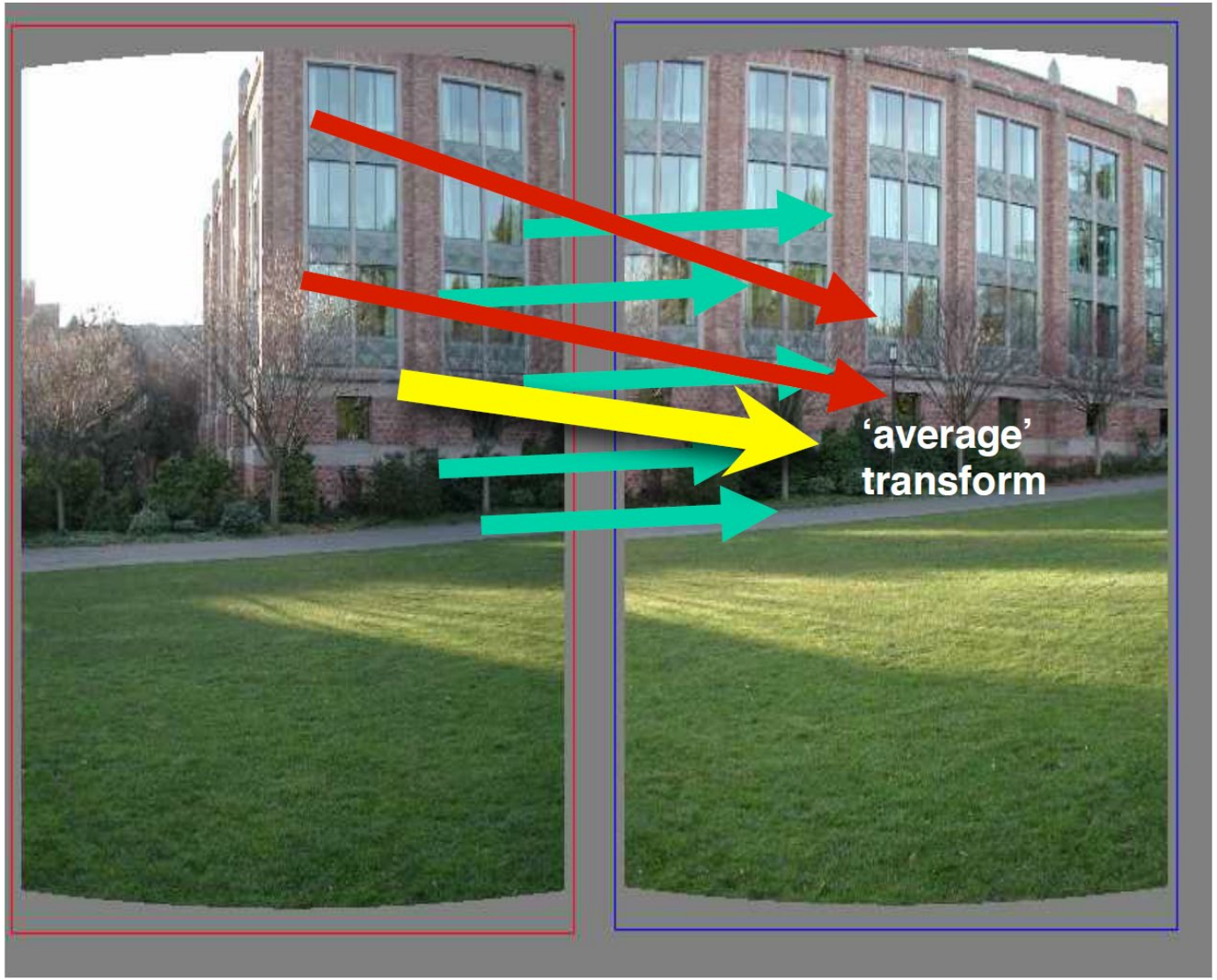
Example

Matched points will usually contain bad correspondences



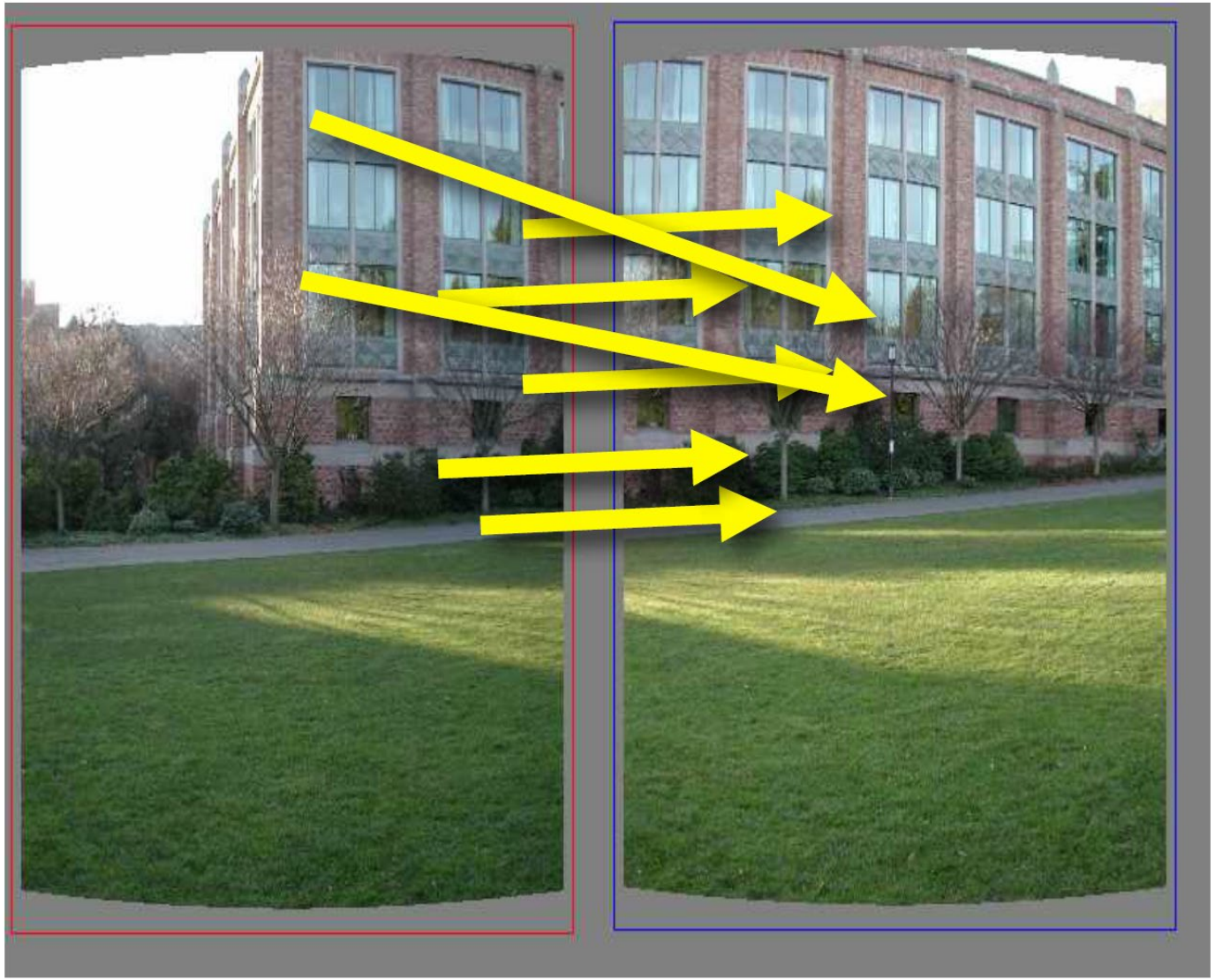
Example

Least squares solutions will find the "average" transform



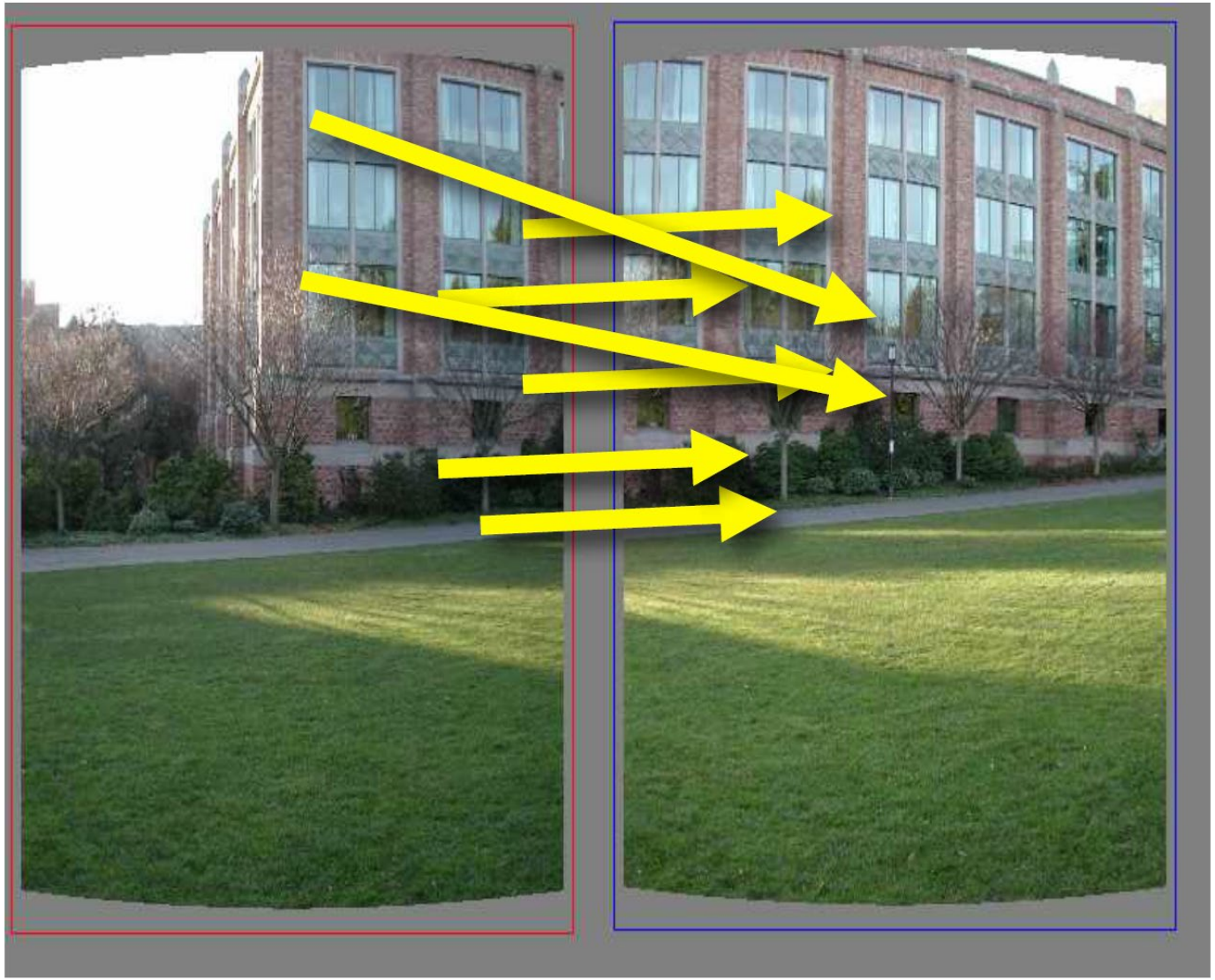
Example

Use RANSAC!



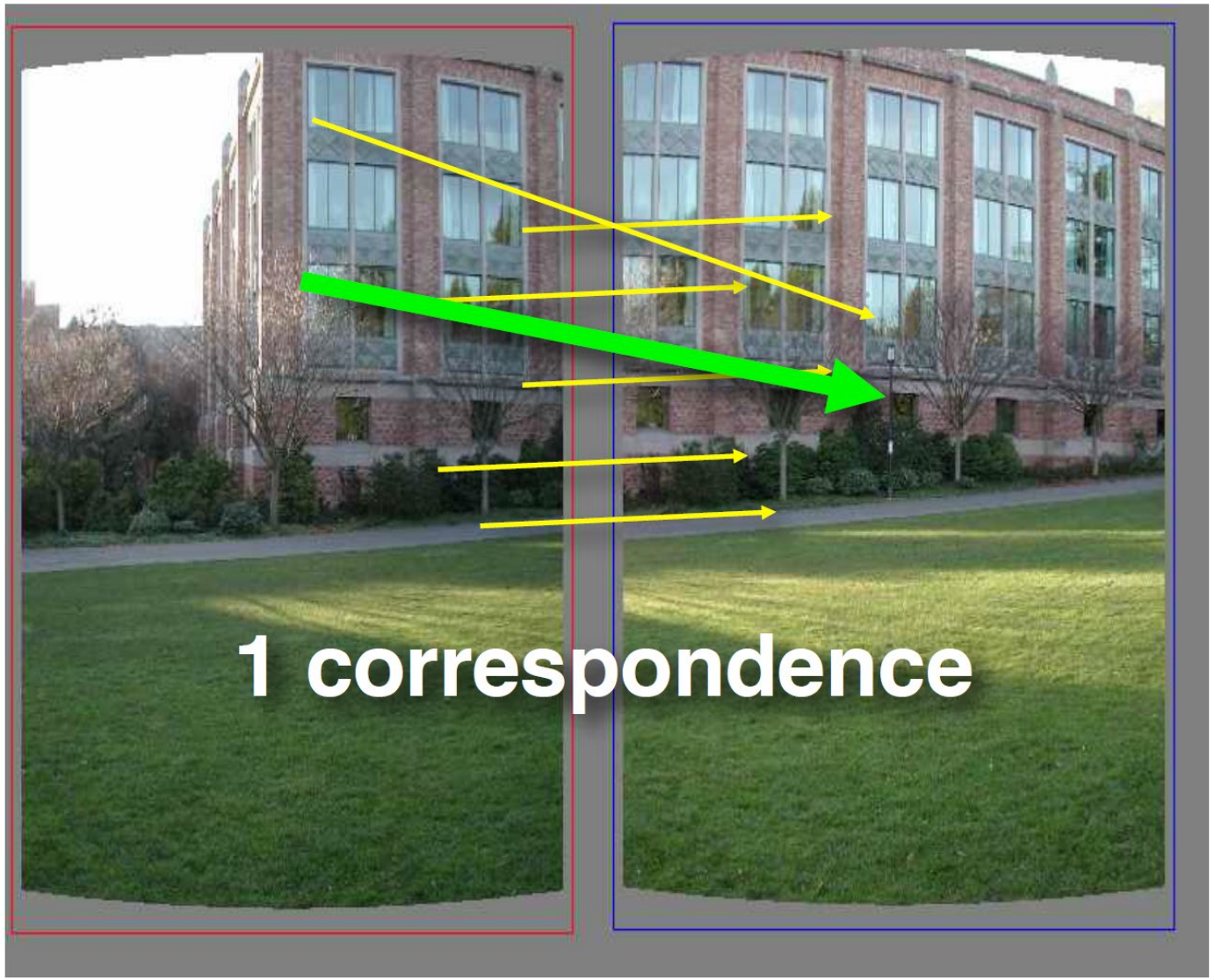
Example

Need only one correspondence, to find translation model



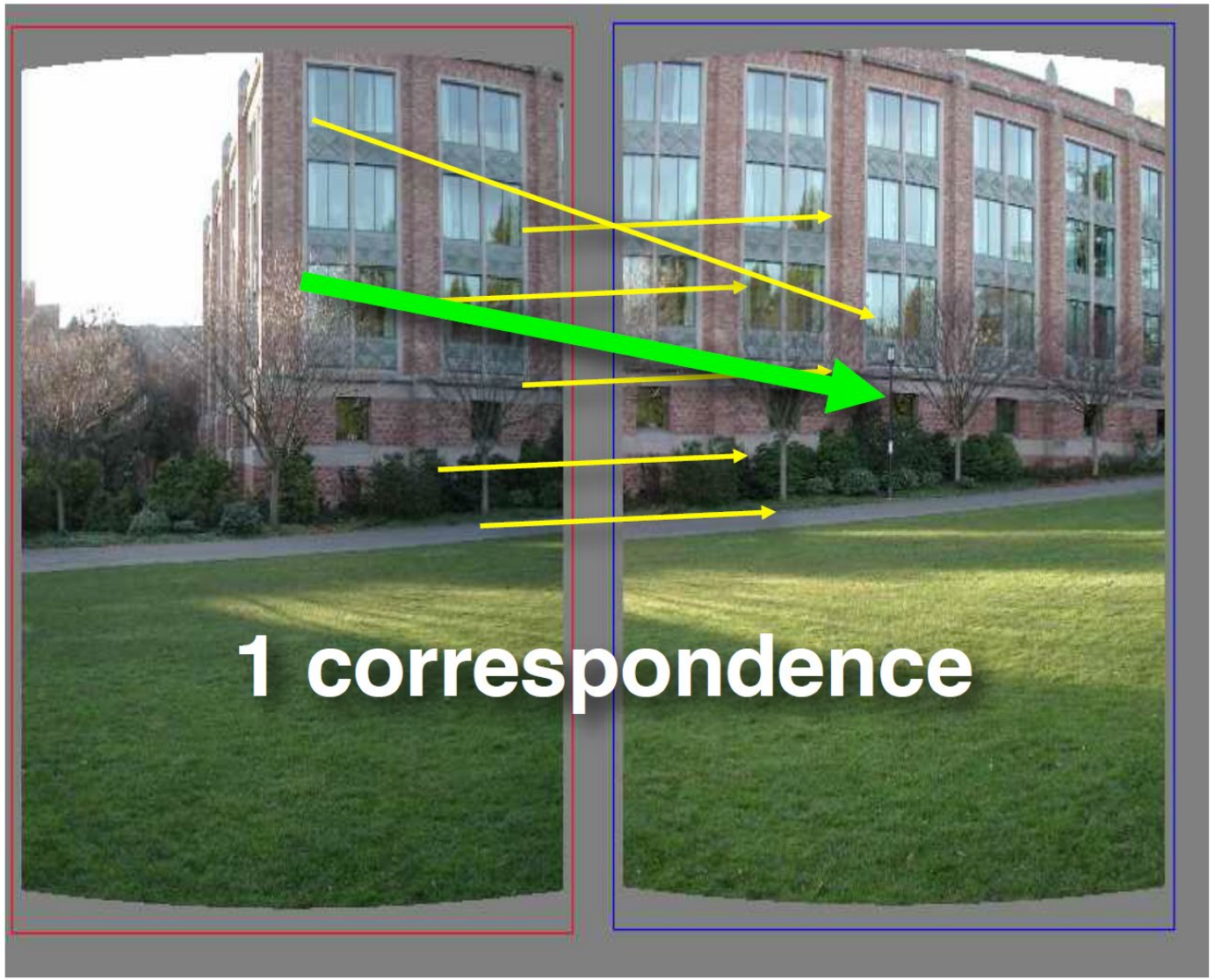
Example

Pick one correspondence, count inliers



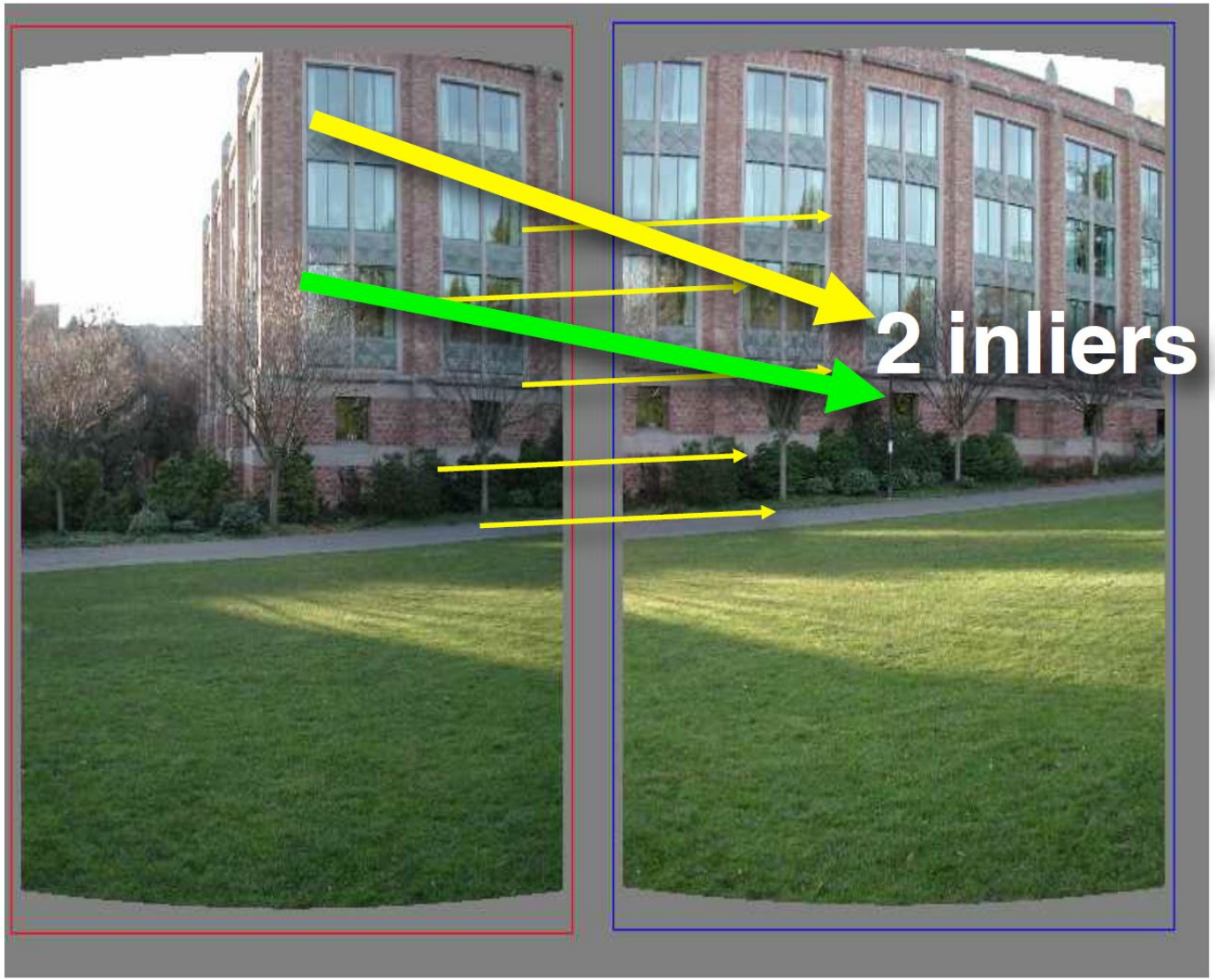
Example

Pick one correspondence, count inliers



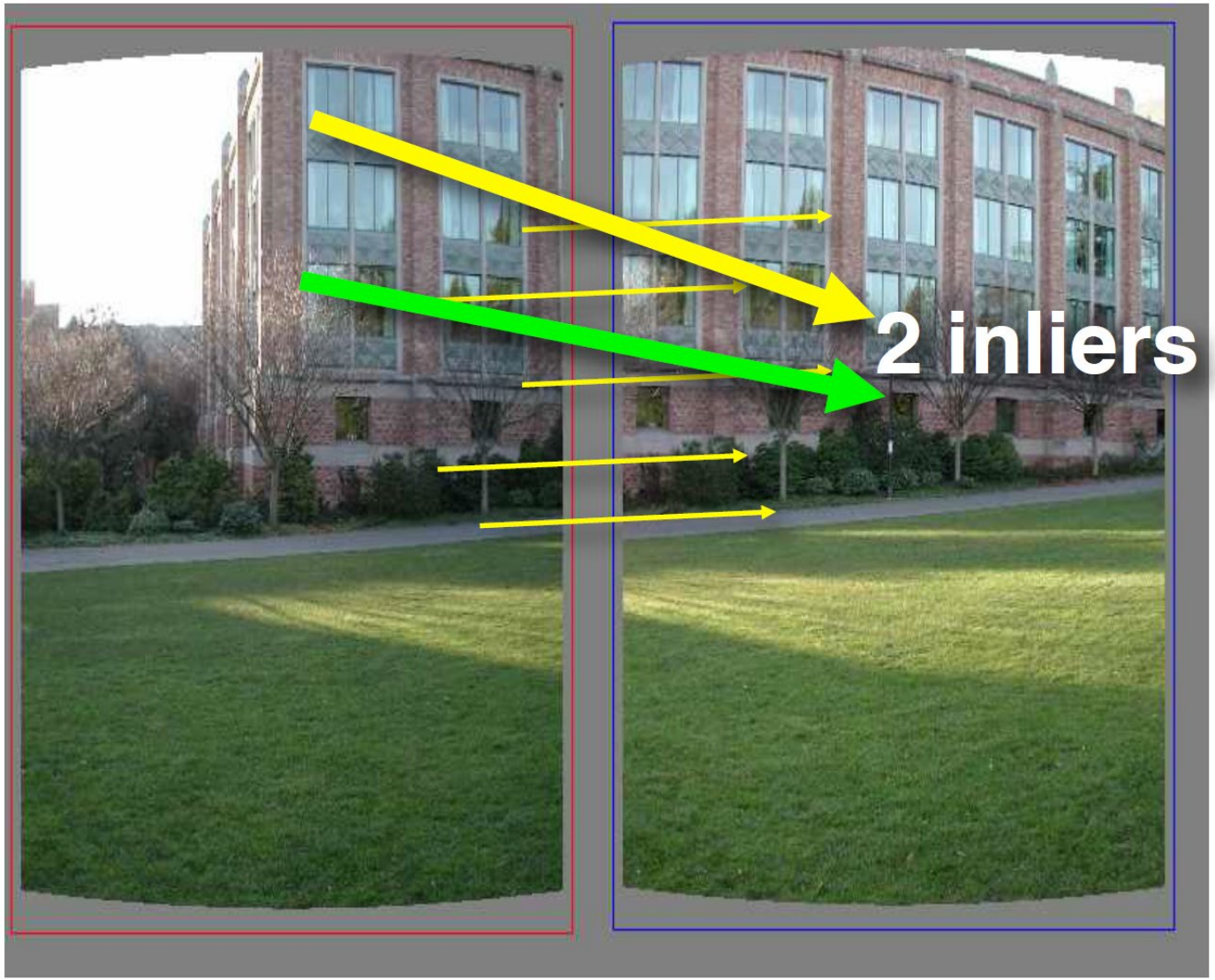
Example

Pick one correspondence, count inliers



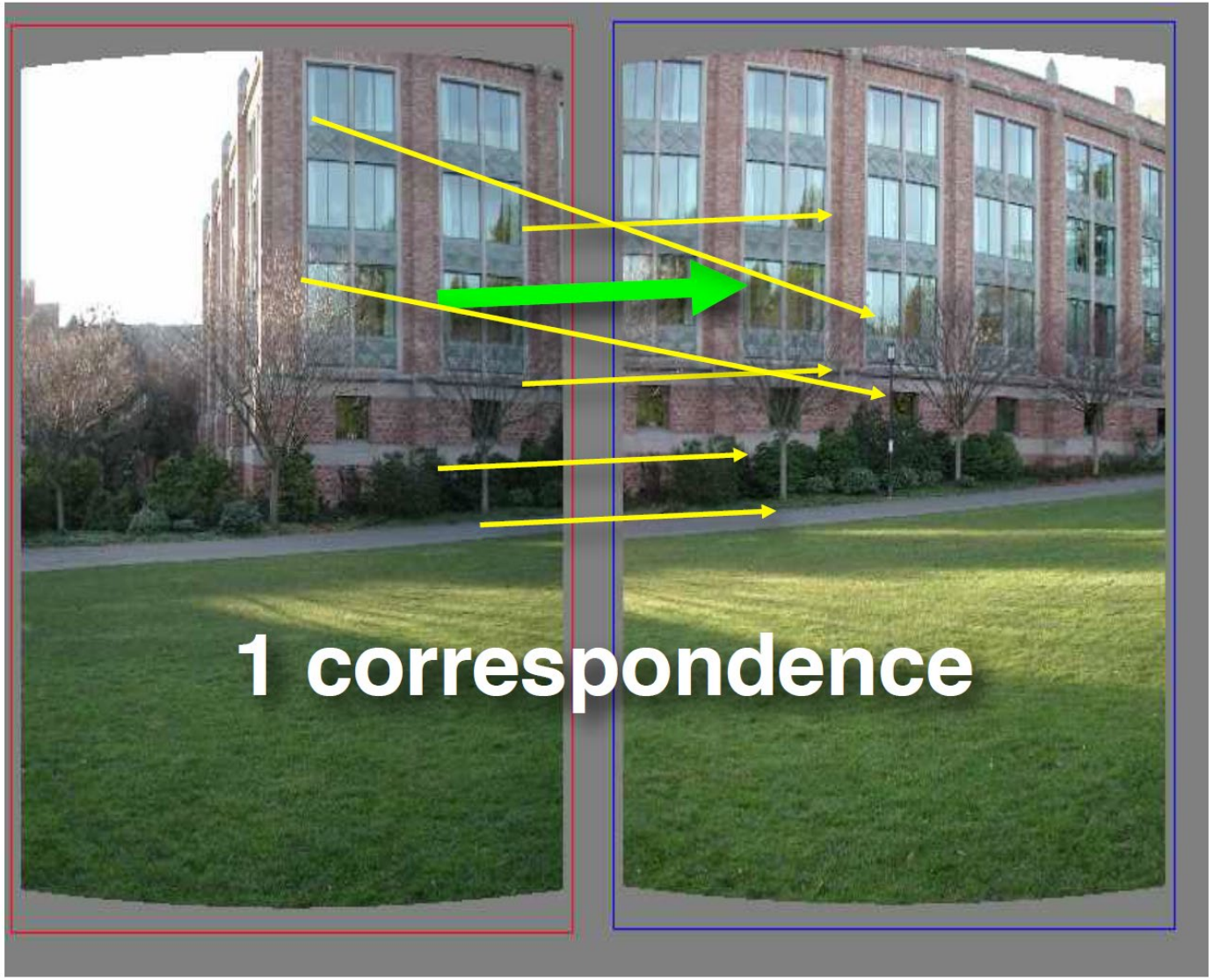
Example

Pick one correspondence, count inliers



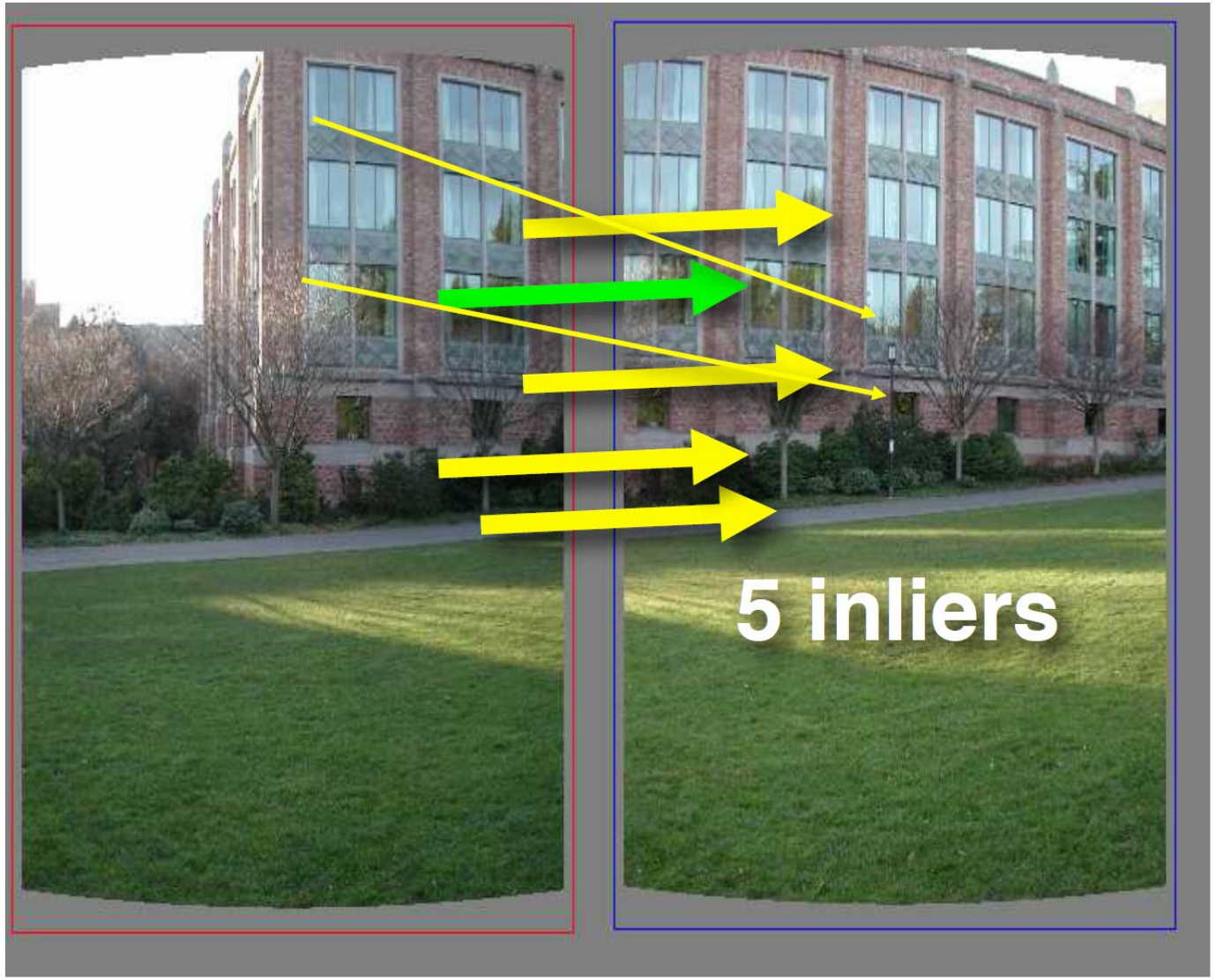
Example

Pick one correspondence, count inliers



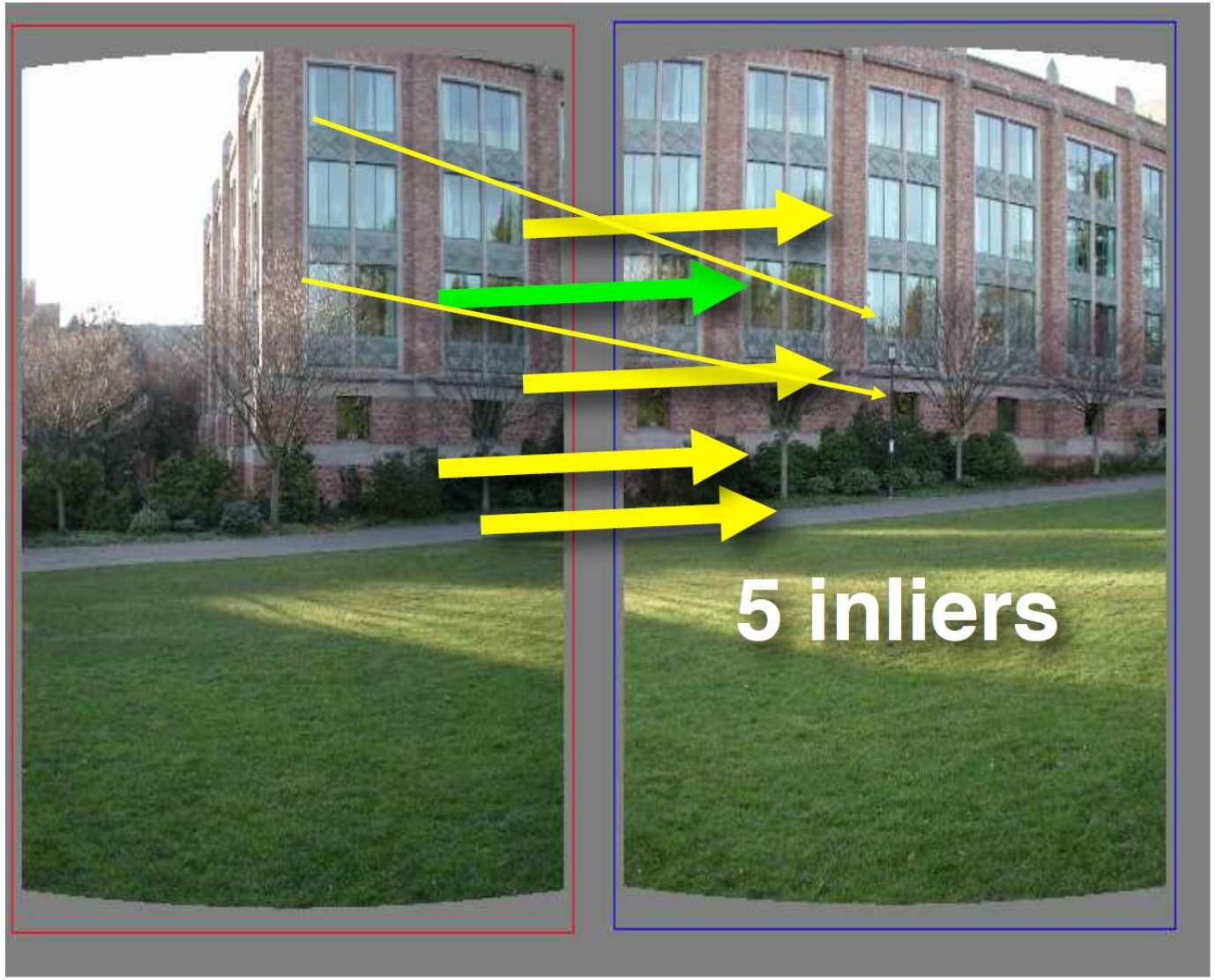
Example

Pick one correspondence, count inliers



Example

Pick the model with the **highest** number of **inliers**!



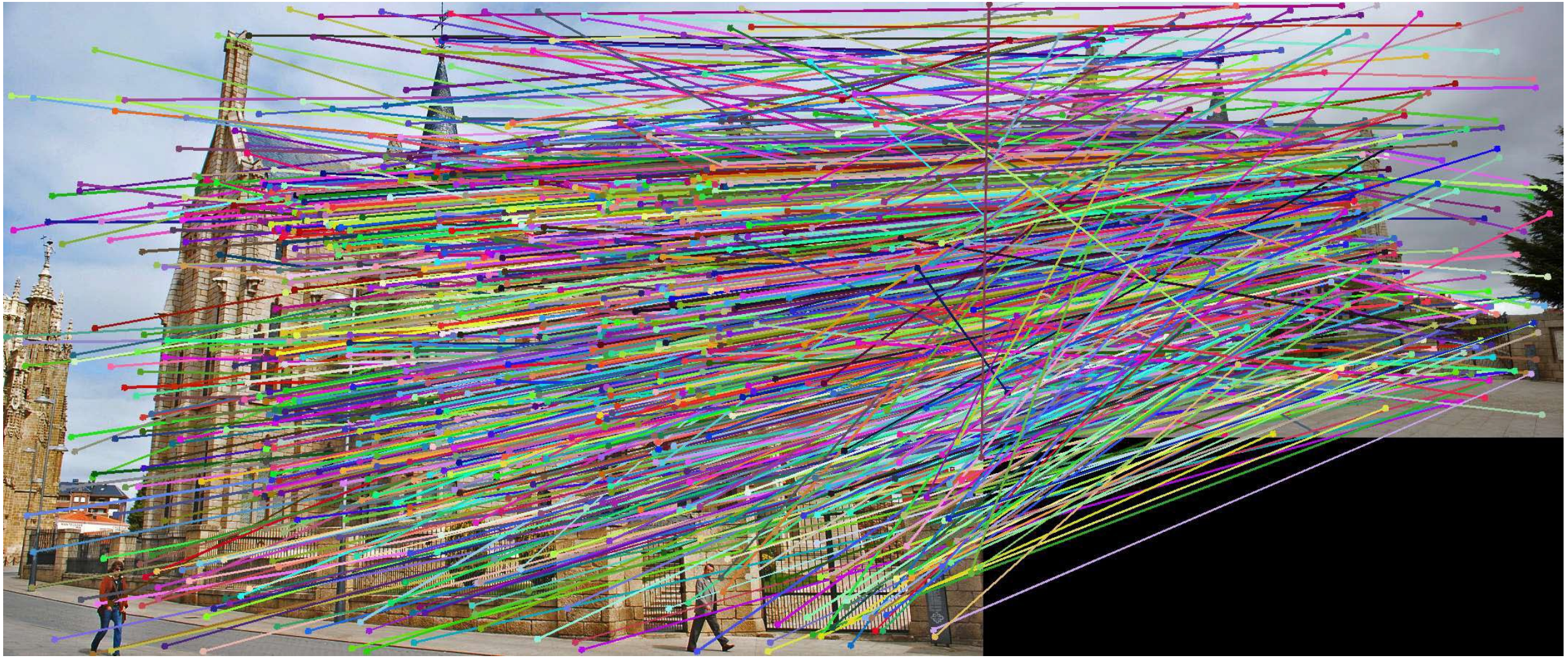
Estimating homography using RANSAC

- RANSAC loop
  1. Get 4 point correspondences (randomly)
  2. Compute H using DLT
  3. Count inliers ($Hp = p'$)
  4. Keep H if largest number of inliers
- Recompute H using all inliers

Estimating essential matrix using RANSAC

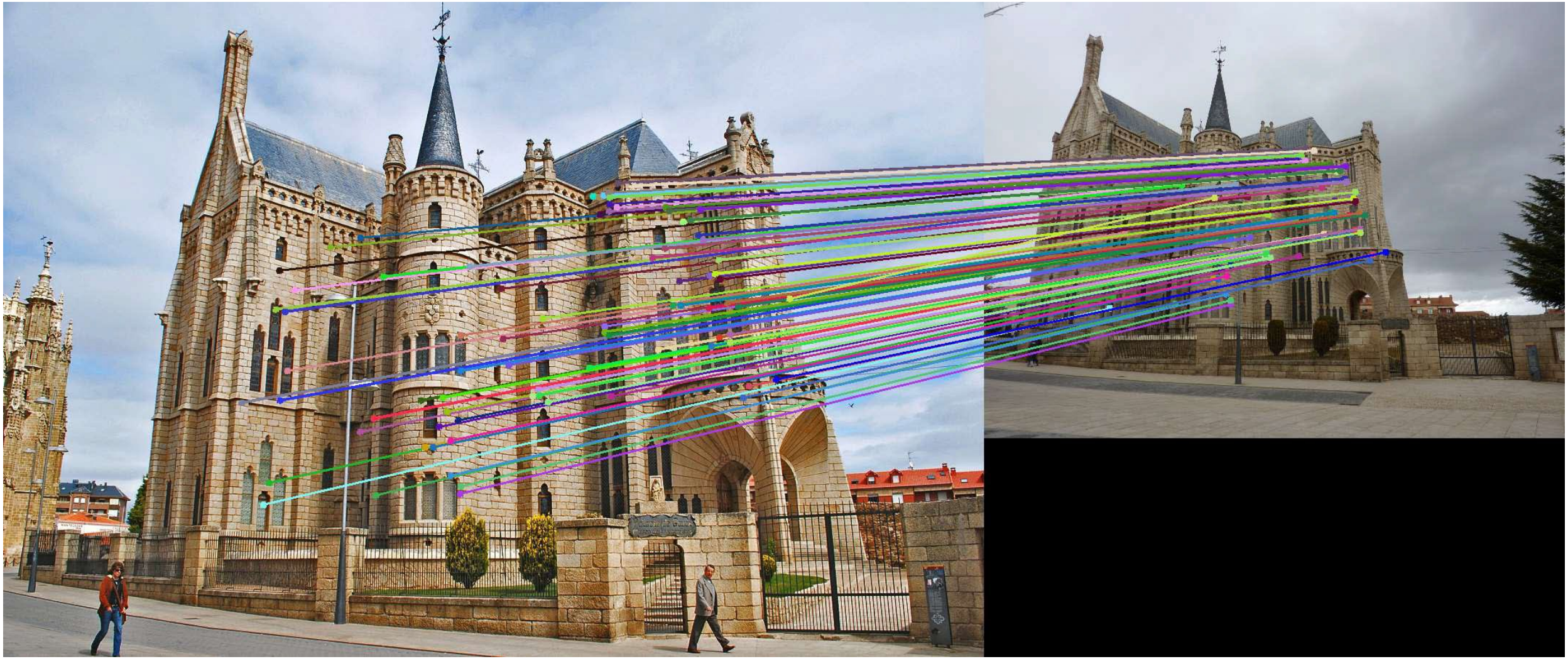
- RANSAC loop
  1. Get 8 point correspondences (randomly)
  2. Compute E using 8-point algorithm
  3. Count inliers ($p^T E p' = 0$)
  4. Keep E if largest number of inliers
- Recompute E using all inliers

SIFT



VLFeat’s 800 most confident matches among 10,000+ local features

RANSAC



RANSAC conclusions

Good

- Robust to outliers
- Applicable to large number of models (or objective functions)
- Parameters are easy to choose

Bad

- Computational time grows quickly (exponentially) with outlier percentage and number of parameters in the model
- Not good for getting multiple fits

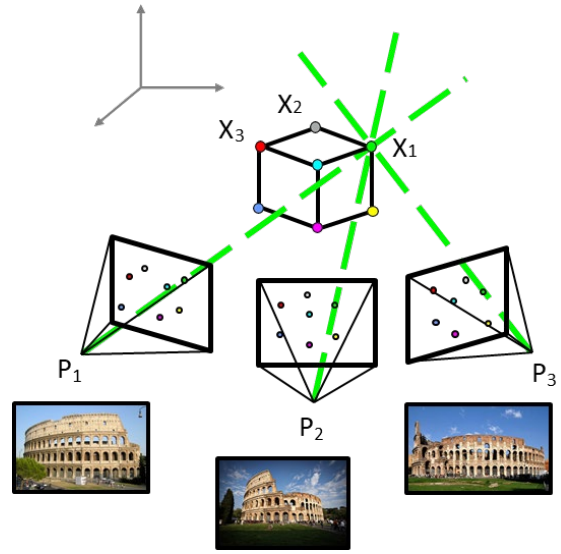
Common applications

- Estimating fundamental matrix (relating two views)
- Computing a homography (e.g., image stitching)

Structure from Motion (SfM)

Structure from Motion (SfM)

- Joint estimation of …
  - Structure $\mathbf{X}_i$
  - Cameras $\mathbf{P}_i$
- From motion, i.e.
  - images at different viewpoints



[colmap]