SSY097 - Image Analysis

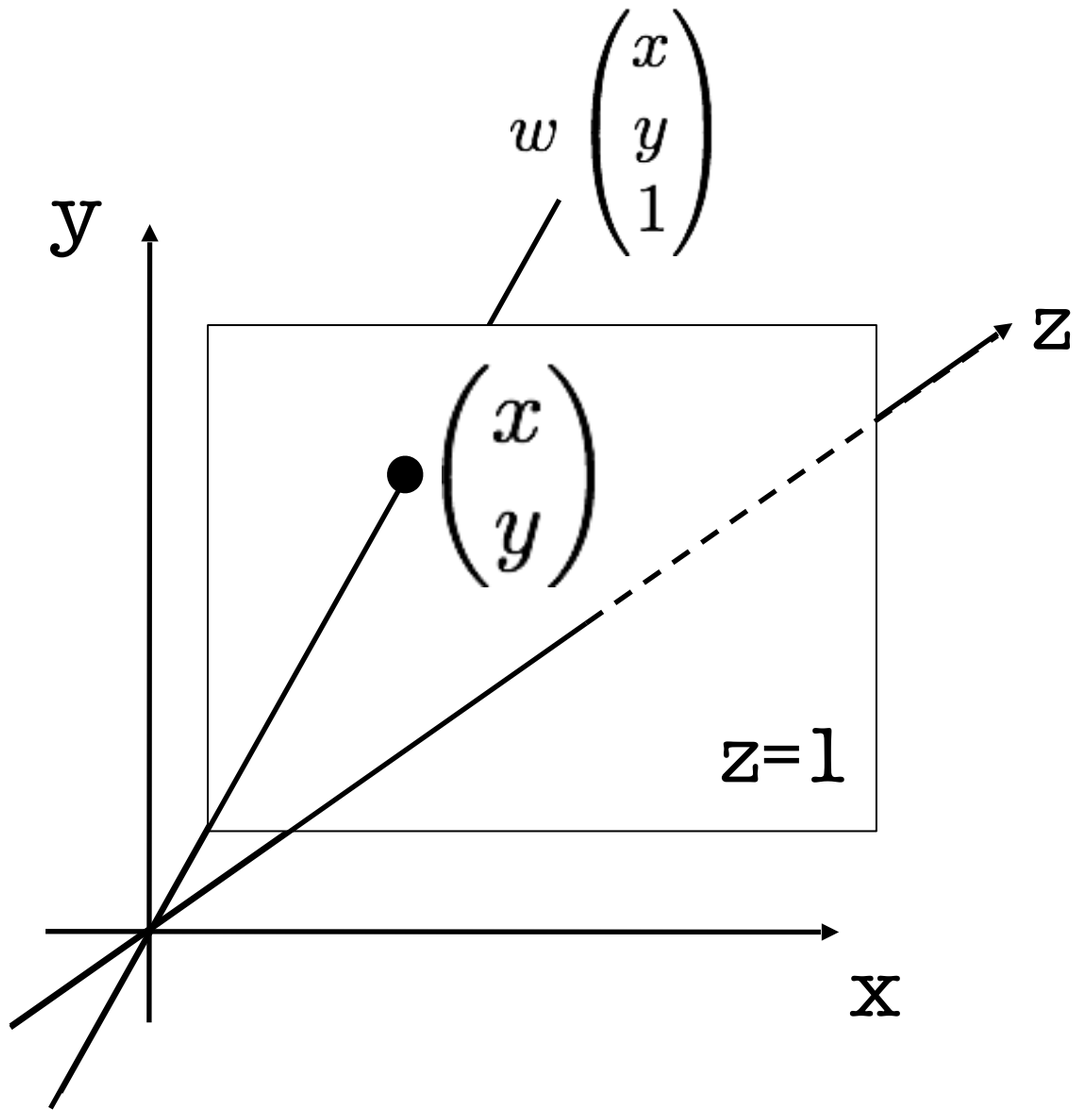
Lecture 10 - Camera & 3D Geometry

Torsten Sattler
(slides adapted from Olof Enqvist)

Last Lecture

Homogeneous coordinates

$$\begin{pmatrix} x \\ y \end{pmatrix} \rightarrow w \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad , \quad w \neq 0$$



De-homogenization:

$$w \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} x/w \\ y/w \end{pmatrix}$$

Homogeneous coordinates

Last Lecture

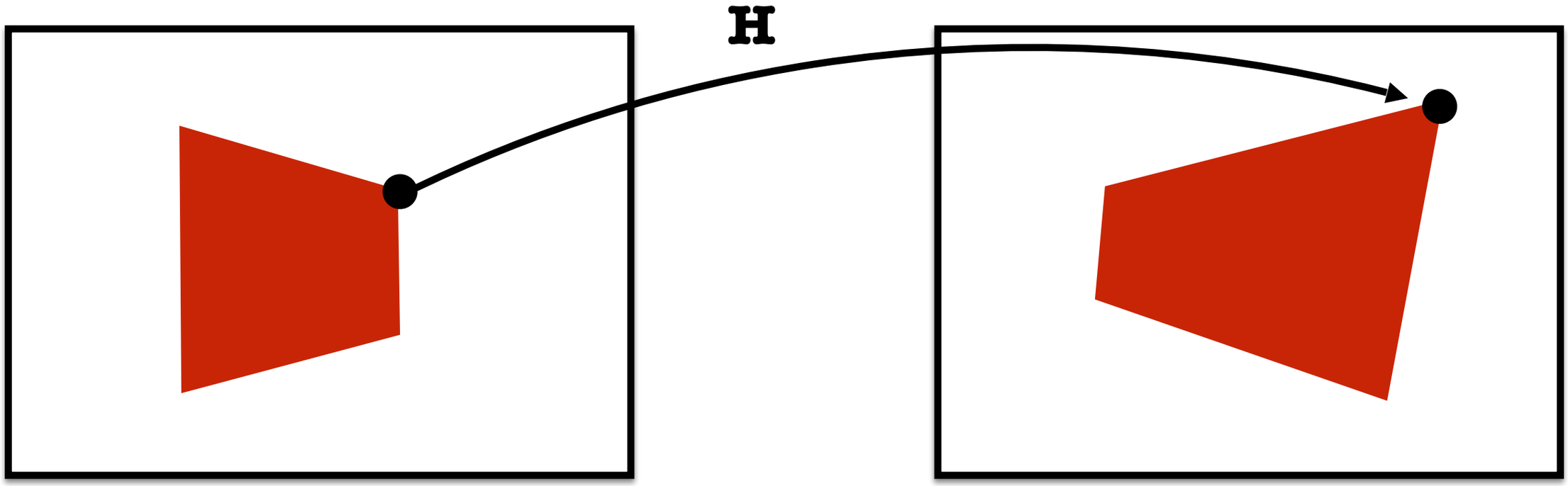
$$\begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

or $\hat{\mathbf{x}} = \mathbf{H}\mathbf{x}$

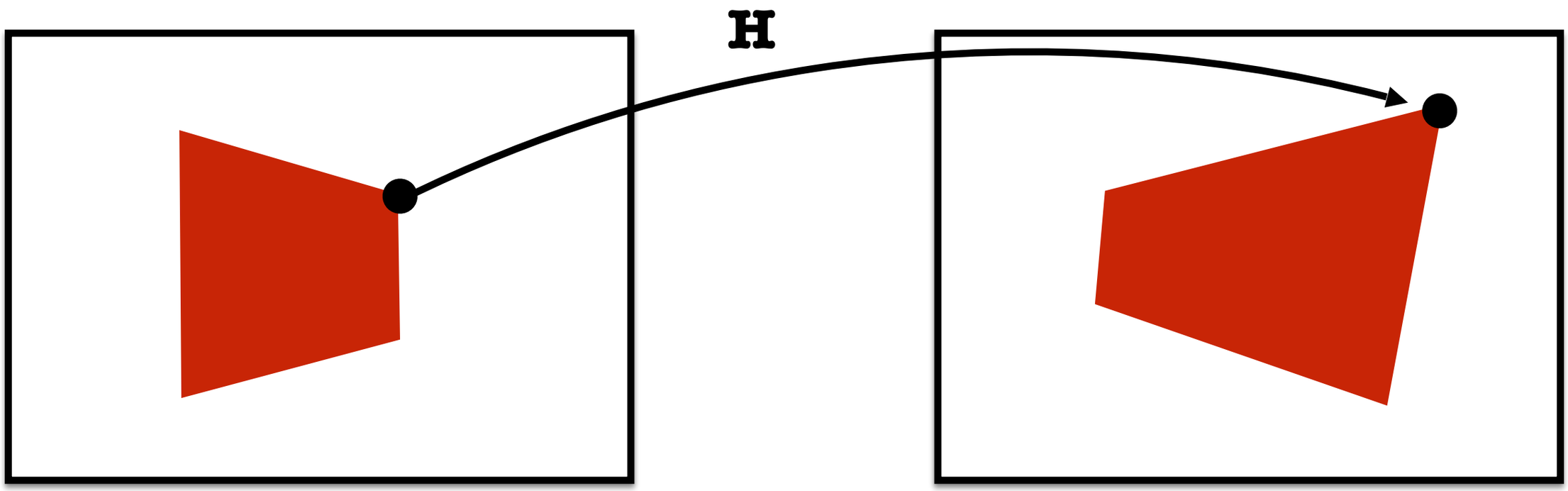
- $\mathbf{H}$ needs to be invertible
- $\mathbf{H}$ has 8 Degrees-of-Freedom (DoF)

Projective mapping (homography)

Last Lecture

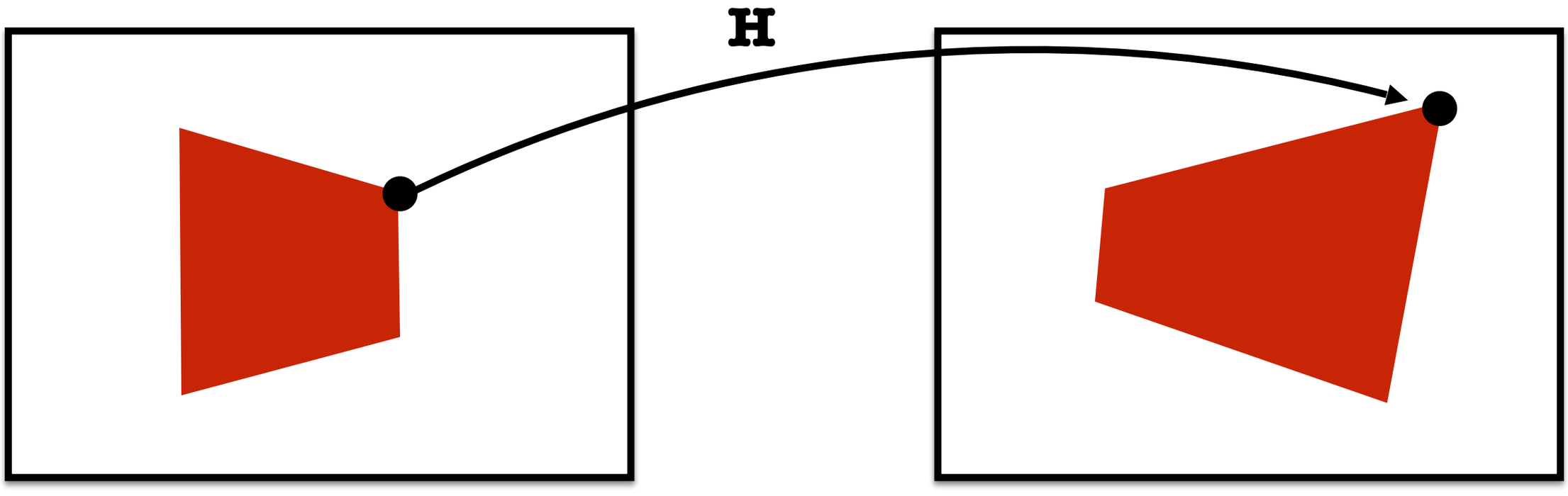


Last Lecture



- "Homogenize": $\begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$
- Apply **H**:

Last Lecture



- "Homogenize": $\begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$
- Apply **H**: $\begin{pmatrix} x'' \\ y'' \\ z'' \end{pmatrix} = \mathtt{H} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$
- De-homogenize: $\begin{pmatrix} x'' \\ y'' \\ z'' \end{pmatrix} \mapsto \begin{pmatrix} x''/z'' \\ y''/z'' \end{pmatrix} = \begin{pmatrix} x' \\ y' \end{pmatrix}$

Last Lecture

Objective

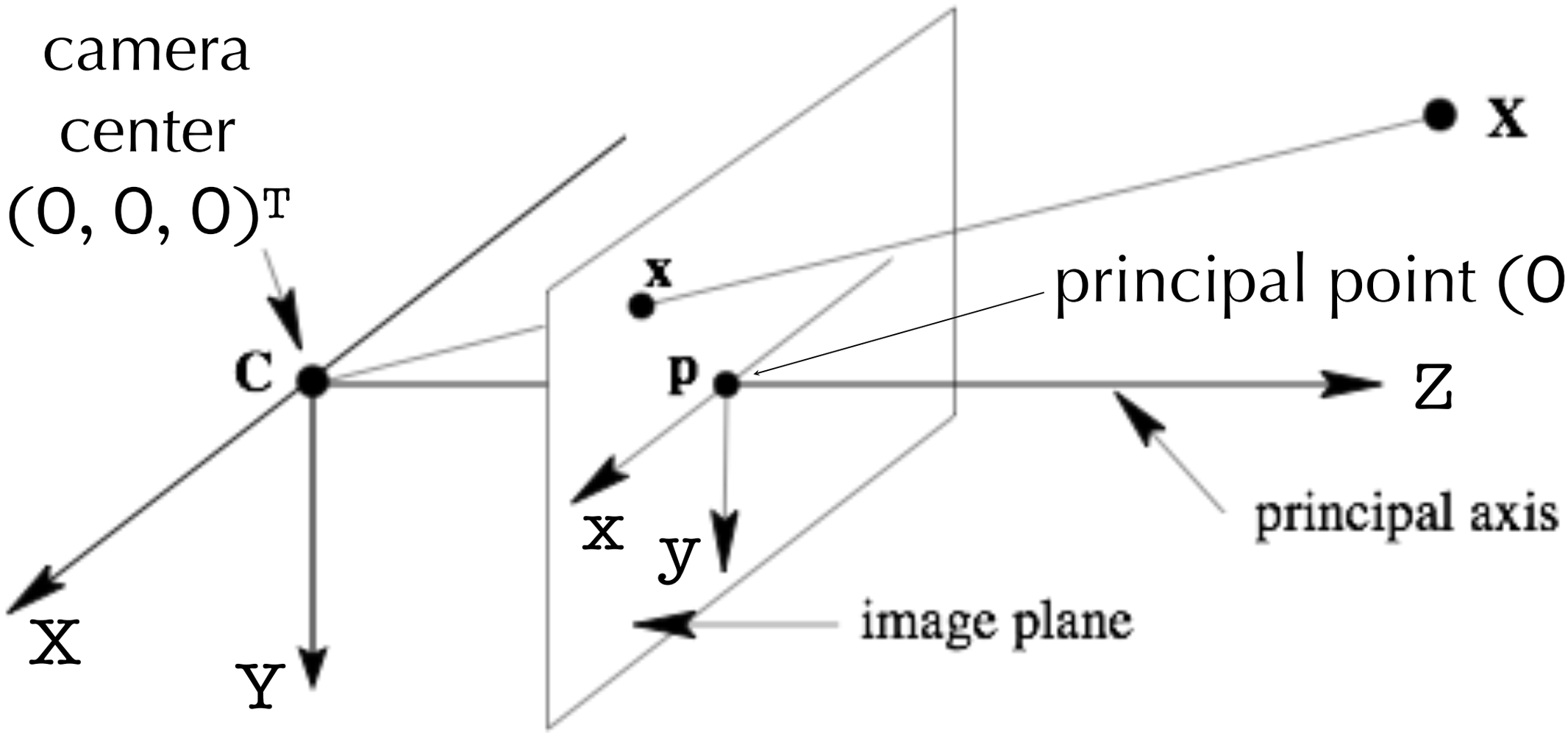
Given n≥4 2D to 2D point correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{x}_i'\}$, determine the 2D homography matrix $\mathbf{H}$ such that $\mathbf{x}_i' = \mathbf{H}\mathbf{x}_i$

Algorithm

- Normalize points: $\tilde{\mathbf{x}}_i = \mathbf{T}_{norm}\mathbf{x}_i, \tilde{\mathbf{x}}_i' = \mathbf{T}'_{norm}\mathbf{x}_i'$
- Apply DLT algorithm to $\tilde{\mathbf{x}}_i \leftrightarrow \tilde{\mathbf{x}}_i'$
- Denormalize solution: $\mathbf{H} = \mathbf{T}'^{-1}_{norm}\tilde{\mathbf{H}}\mathbf{T}_{norm}$

Computing Homographies

Last Lecture



Pinhole camera model

Last Lecture

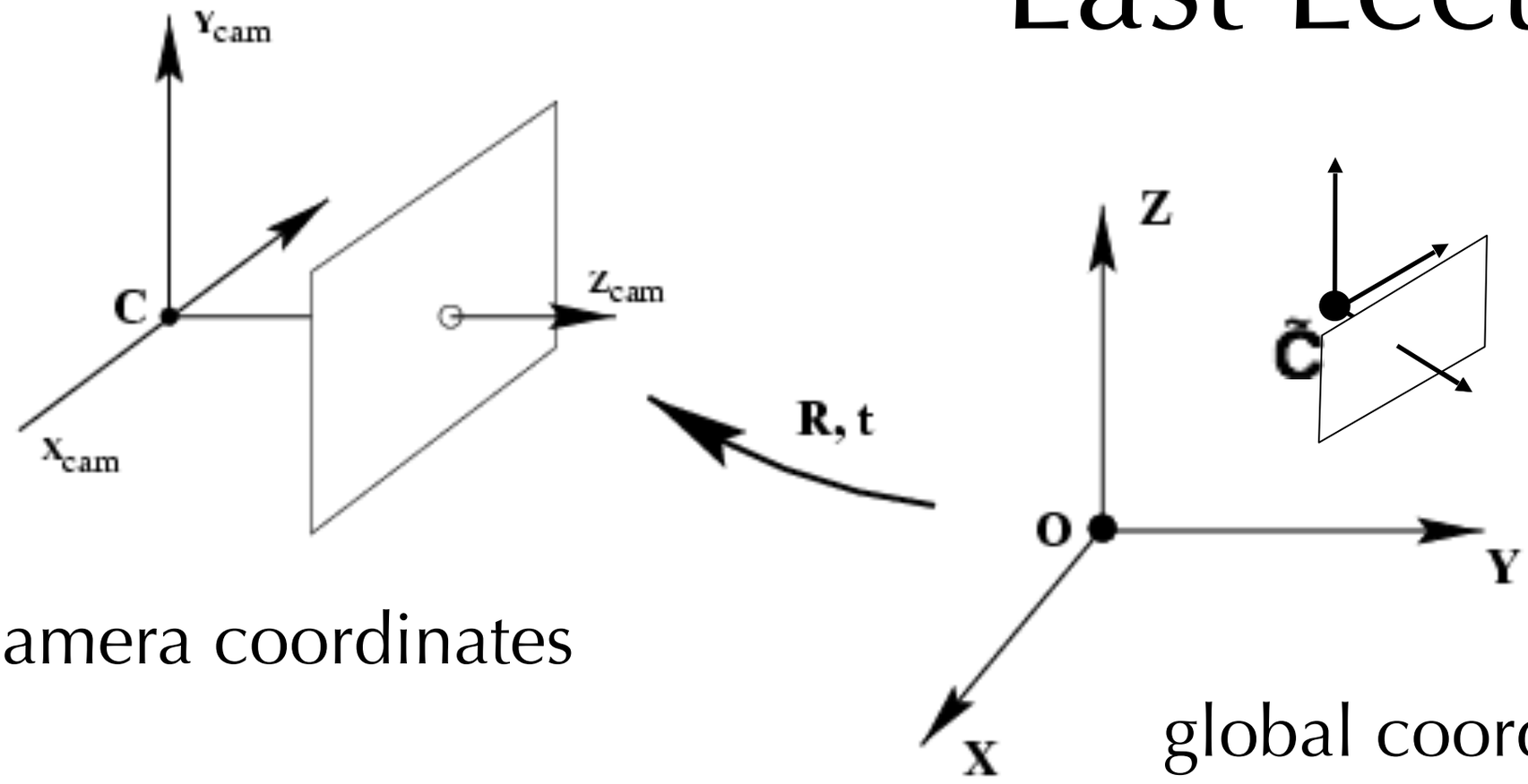
General intrinsic camera calibration matrix:

$$\mathrm{K} = \begin{pmatrix} f & s & p_x \\ 0 & \alpha f & p_y \\ 0 & 0 & 1 \end{pmatrix}$$

Projection

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \mathrm{K}\mathbf{X} \mapsto \begin{pmatrix} x'/z' \\ y'/z' \end{pmatrix}$$

Last Lecture



camera coordinates

global coordinates

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathtt{K}\left(\mathbf{R}\mathbf{X}_{\text{global}} + \mathbf{t}\right) = \mathtt{K}\left[\mathbf{R}|\mathbf{t}\right] \begin{pmatrix} \mathbf{X} \\ 1 \end{pmatrix} = \mathtt{P} \begin{pmatrix} \mathbf{X} \\ 1 \end{pmatrix}$$

Extrinsic and intrinsic camera parameters

projection matrix

More Camera Geometry

Rolling Shutter Effect

video

Rolling Shutter Effect

video

Rolling Shutter Effect

Global shutter

Rolling shutter

youtu.be/7TGKFdrY9aw

Slide credit: Cenek Albl

Rolling Shutter Effect

Global shutter

Rolling shutter

youtu.be/7TGKFdrY9aw

Slide credit: Cenek Albl

Rolling Shutter Effect

Global shutter

Rolling shutter

youtu.be/7TGKFdrY9aw

Slide credit: Cenek Albl

Rolling Shutter Effect

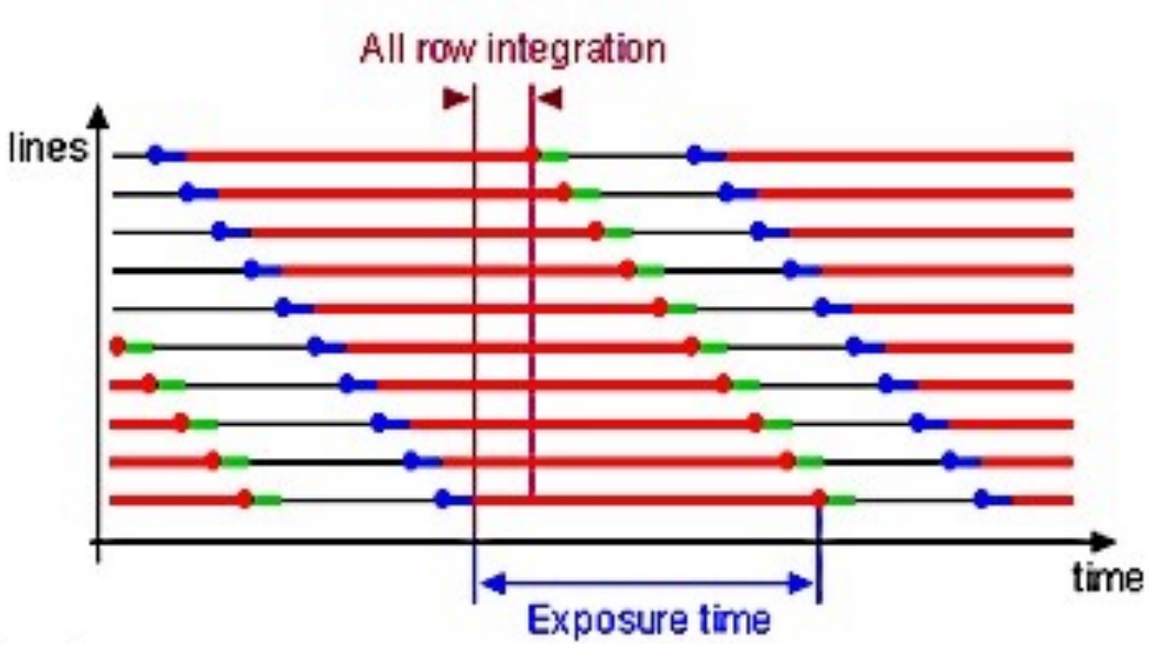


Image recorded line by line

Slide credit: Cenek Albl

Rolling Shutter Effect

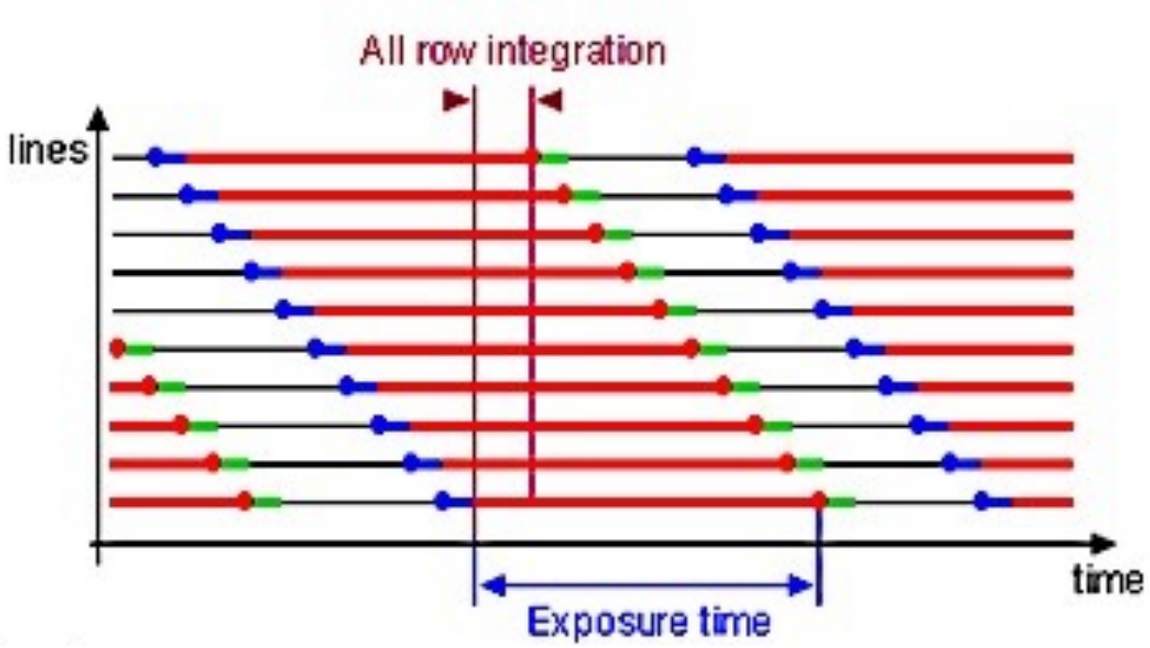


Image recorded line by line

Rolling shutter effect

Slide credit: Cenek Albl

Rolling Shutter Effect

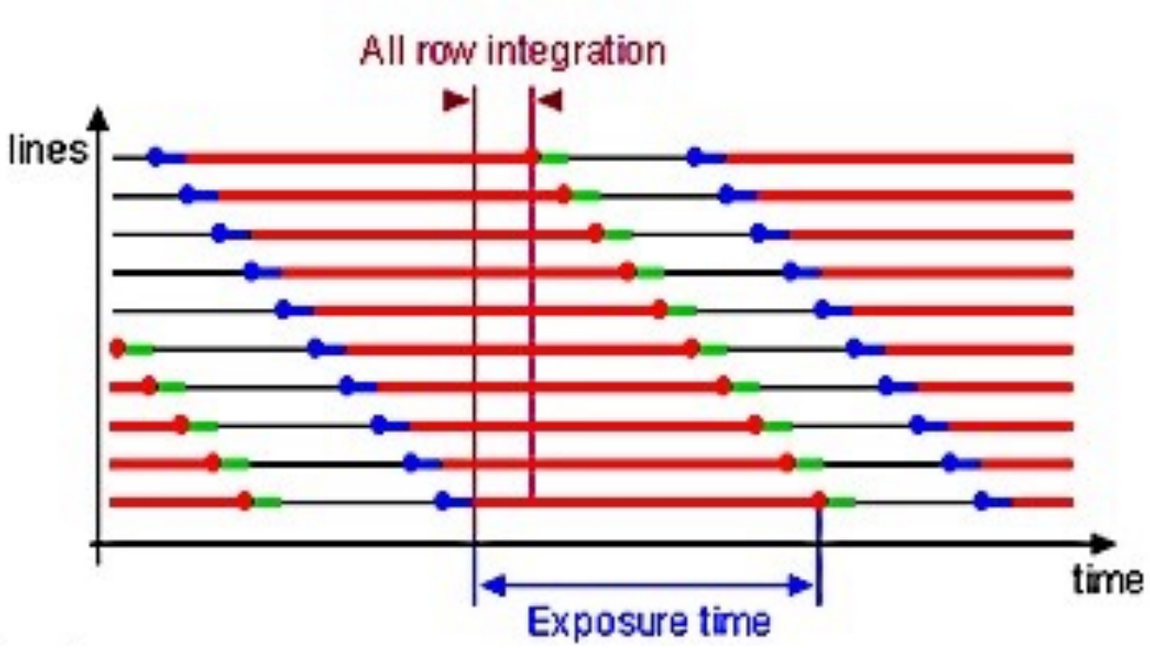


Image recorded line by line

Rolling shutter effect

- Rolling shutter cameras cheaper
- Faster frame rates

Slide credit: Cenek Albl

Rolling Shutter Effect

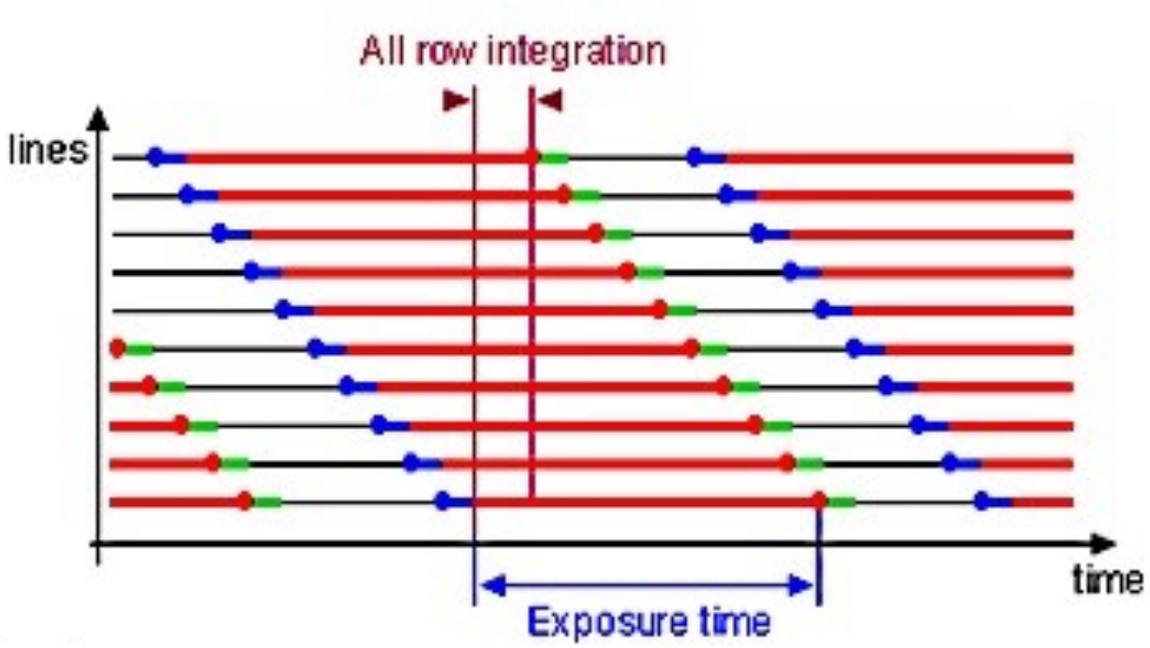


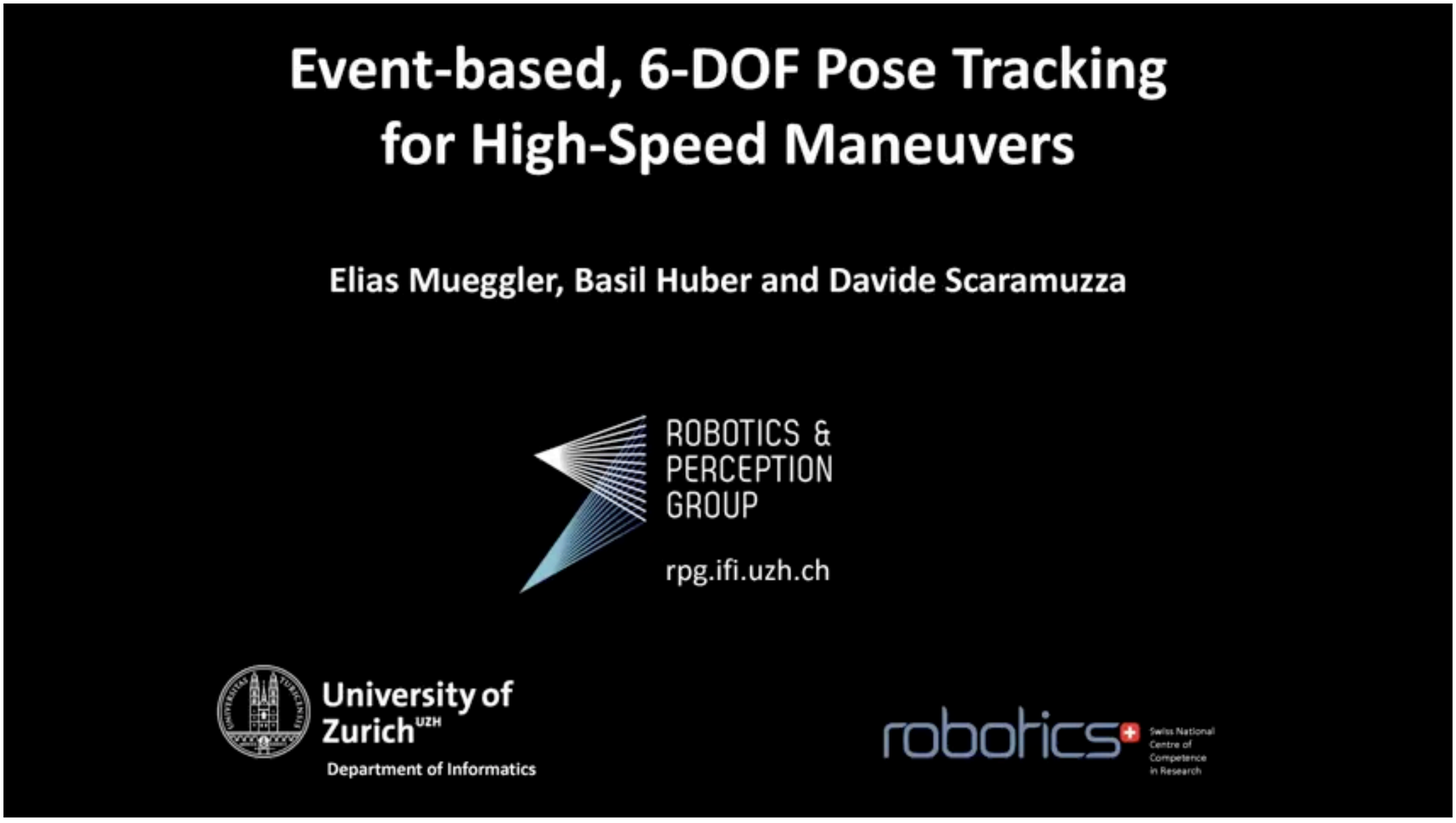
Image recorded line by line

Rolling shutter effect

- Rolling shutter cameras cheaper
- Faster frame rates
- Better adaption to illumination changes

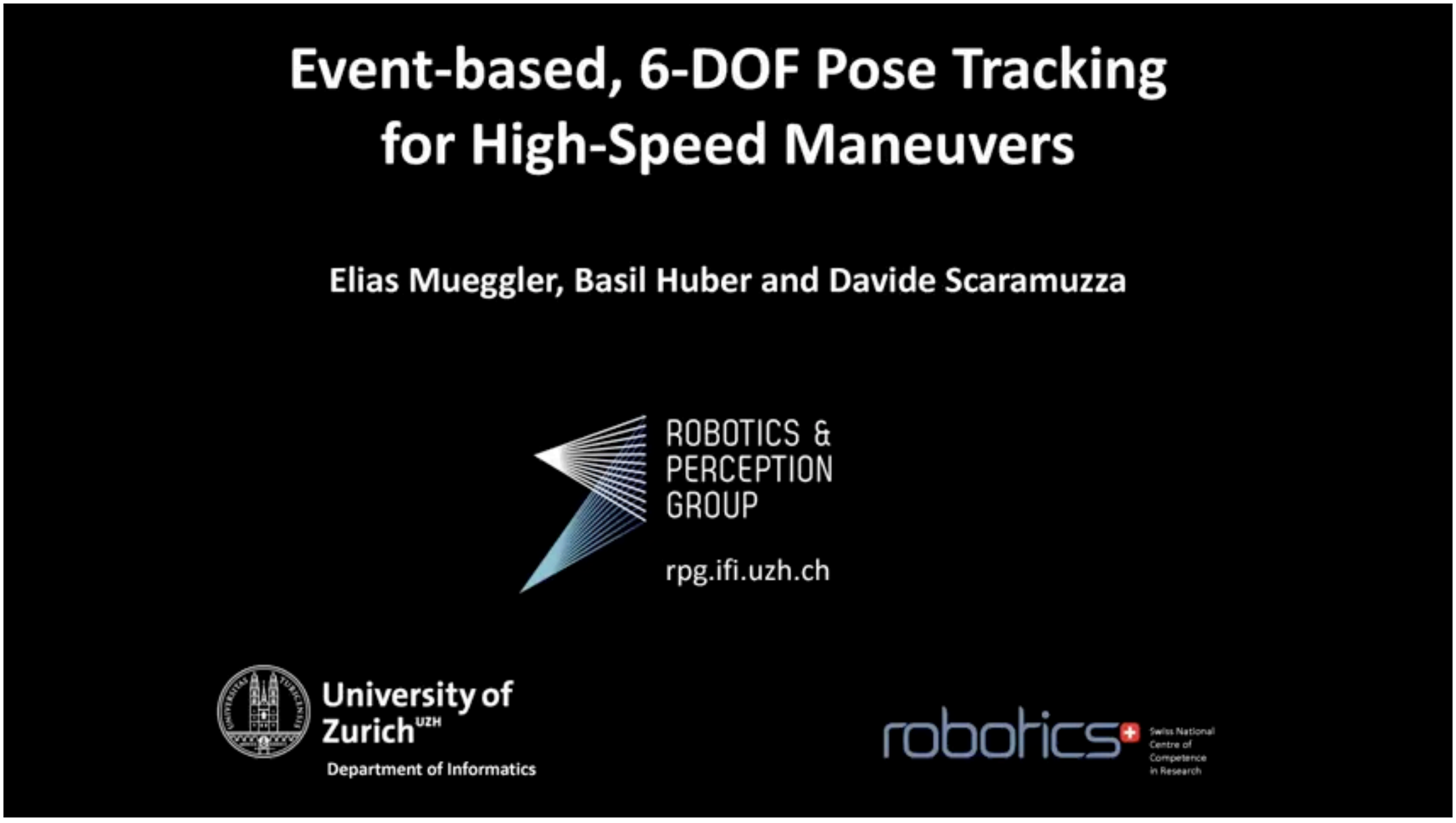
Slide credit: Cenek Albl

Event Cameras



video

Event Cameras



video

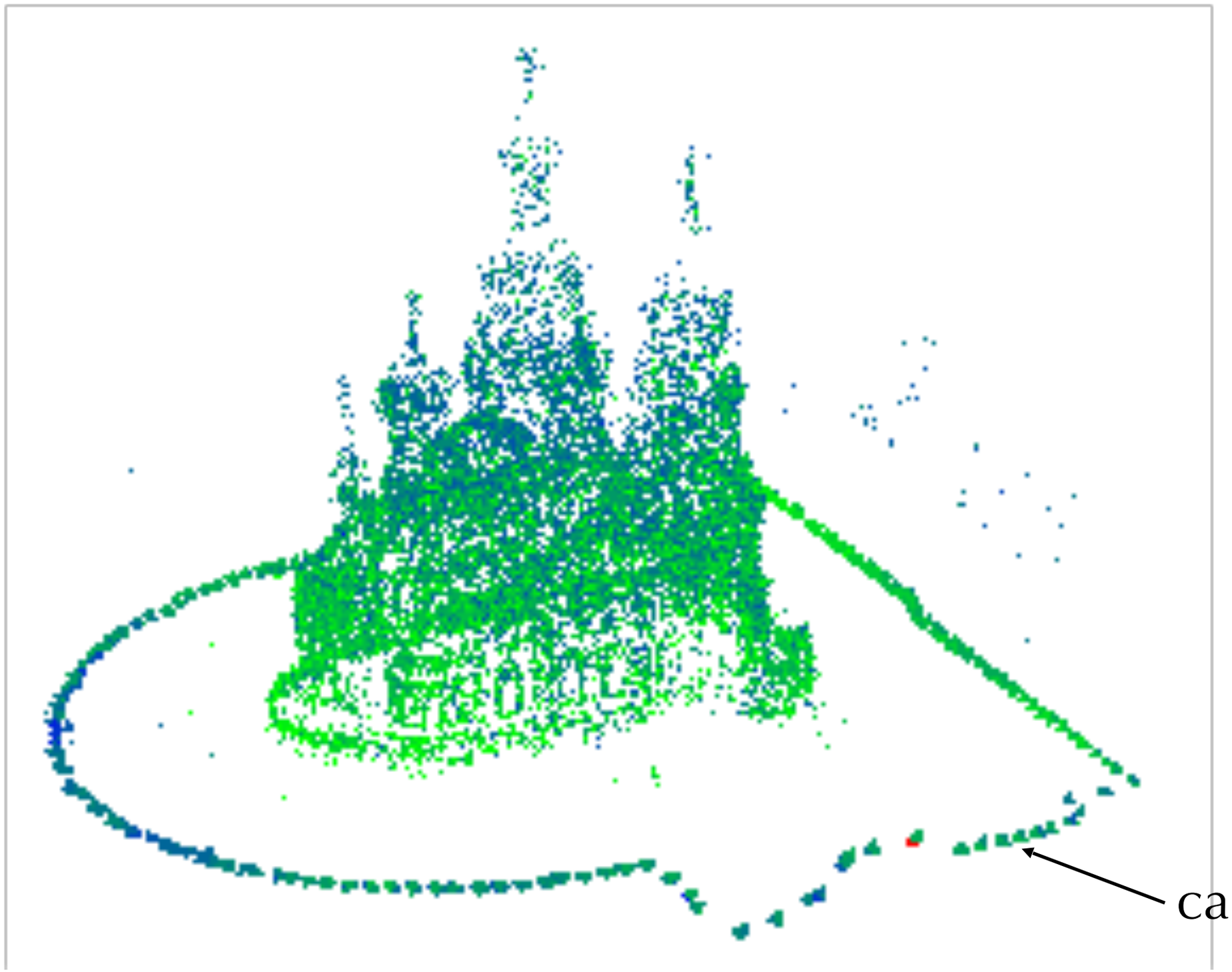
Today

3D Reconstruction

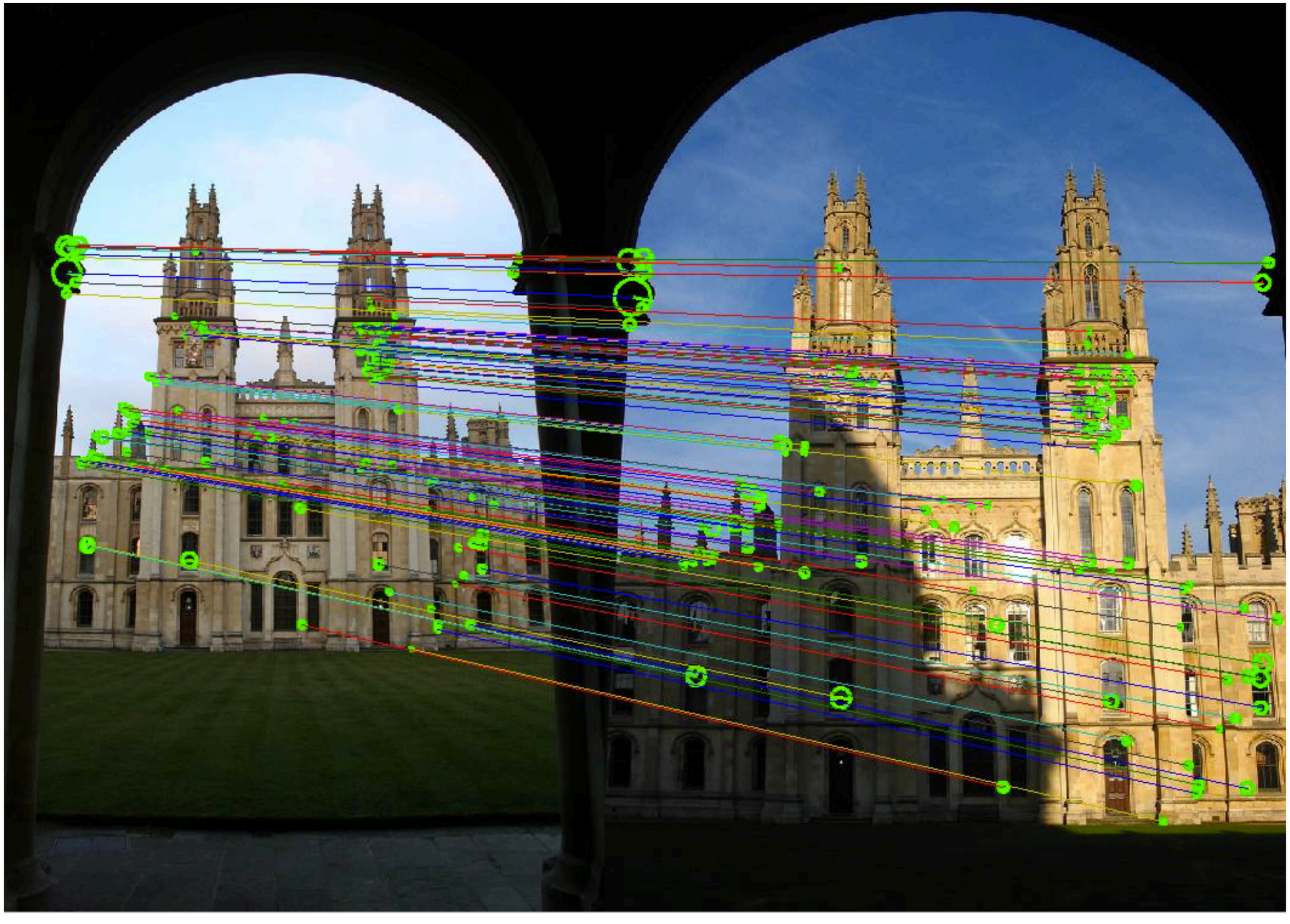
Structure-from-Motion

Structure-from-Motion

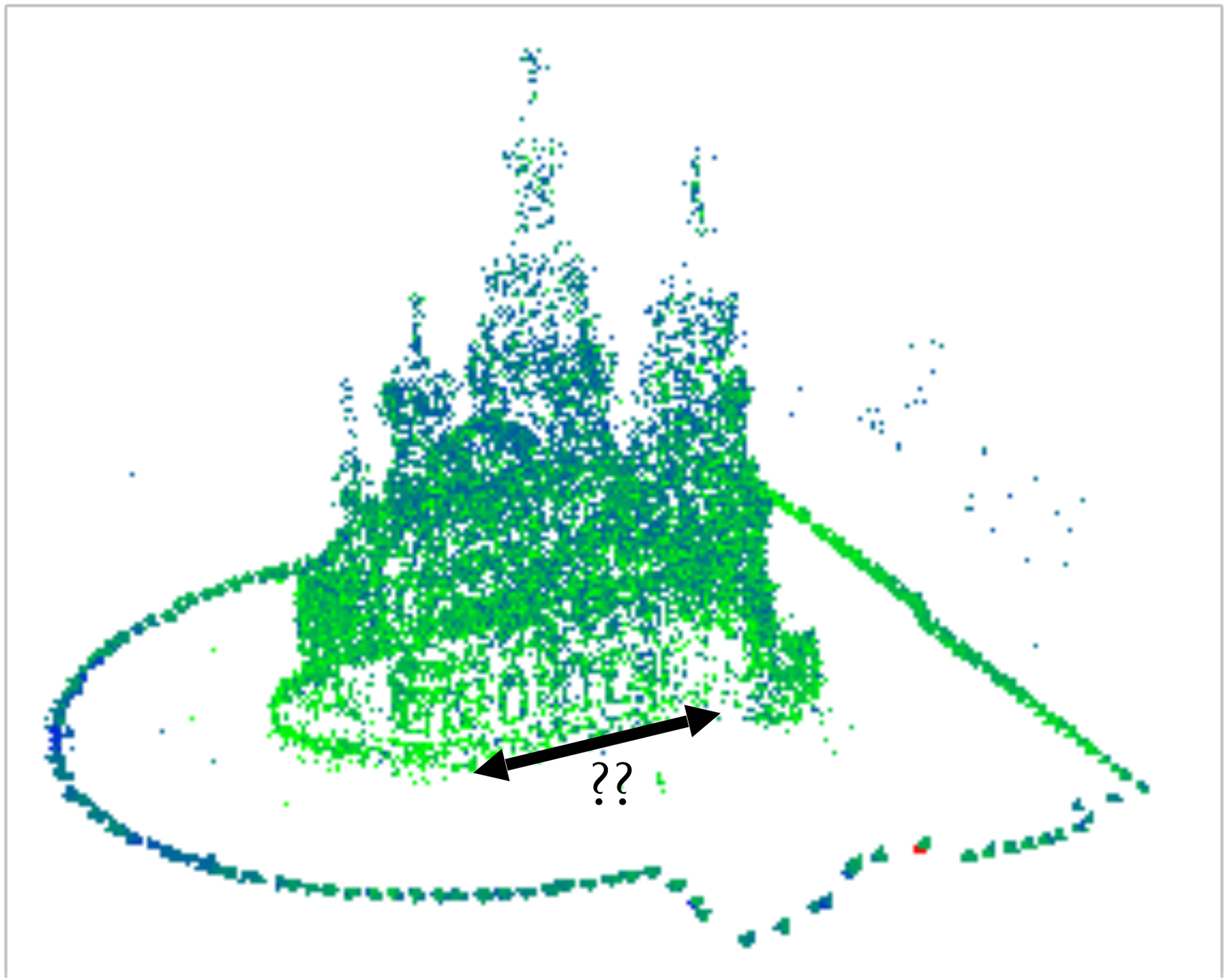
Structure-from-Motion



The Measurements



Scale of a 3D Model



Recovering Scale?

photo credit: Zuzana Kukelova

Recovering Scale?

photo credit: [Miguel Mendez]

3D Models Are Defined Up To Scale

3D point $\mathbf{X}$ seen from camera with pose $\mathbf{R}$, $\mathbf{t}$, intrinsics $\mathbf{K}$

$$\mathbf{K}\left(\mathbf{R}\mathbf{X}+\mathbf{t}\right) = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \end{pmatrix}$$

3D Models Are Defined Up To Scale

3D point $\mathbf{X}$ seen from camera with pose $\mathbf{R}$, $\mathbf{t}$, intrinsics $\mathbf{K}$

$$\mathbf{K}\left(\mathbf{R}\mathbf{X}+\mathbf{t}\right) \quad = \quad \begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \end{pmatrix}$$

Scale 3D scene by arbitrary factor $s \neq 0$

$$\mathbf{K}\left(\mathbf{R}(s\mathbf{X})+s\mathbf{t}\right) \quad = \quad s\mathbf{K}\left(\mathbf{R}\mathbf{X}+\mathbf{t}\right)$$

3D Models Are Defined Up To Scale

3D point $\mathbf{X}$ seen from camera with pose $\mathbf{R}$, $\mathbf{t}$, intrinsics $\mathbf{K}$

$$\mathbf{K}\left(\mathbf{R}\mathbf{X}+\mathbf{t}\right) = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \end{pmatrix}$$

Scale 3D scene by arbitrary factor $s \neq 0$

$$\begin{aligned} \mathbf{K}\left(\mathbf{R}(s\mathbf{X})+s\mathbf{t}\right) &= s\mathbf{K}\left(\mathbf{R}\mathbf{X}+\mathbf{t}\right) \\ &= s\begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} \frac{sx}{sz} \\ \frac{sy}{sz} \end{pmatrix} \end{aligned}$$

3D Models Are Defined Up To Scale

3D point $\mathbf{X}$ seen from camera with pose $\mathbf{R}$, $\mathbf{t}$, intrinsics $\mathbf{K}$

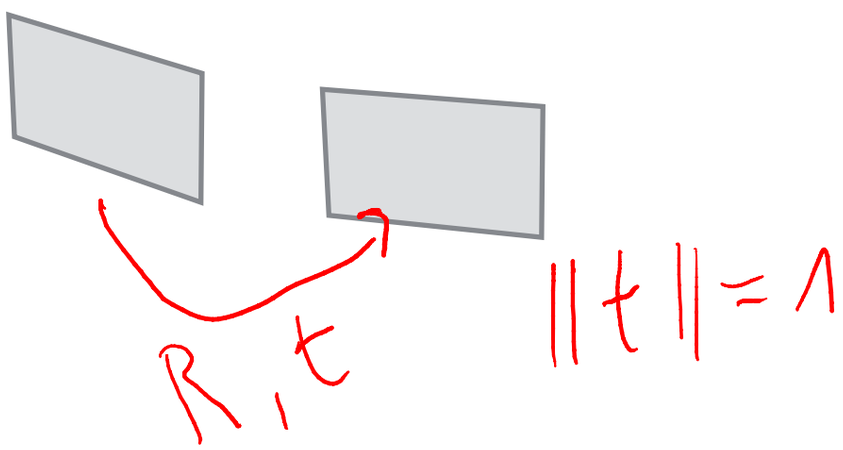
$$\mathbf{K}\left(\mathbf{R}\mathbf{X}+\mathbf{t}\right) = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \end{pmatrix}$$

Scale 3D scene by arbitrary factor $s \neq 0$

$$\begin{aligned} \mathbf{K}\left(\mathbf{R}(s\mathbf{X})+s\mathbf{t}\right) &= s\mathbf{K}\left(\mathbf{R}\mathbf{X}+\mathbf{t}\right) \\ &= s\begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} \frac{sx}{sz} \\ \frac{sy}{sz} \end{pmatrix} = \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \end{pmatrix} \end{aligned}$$

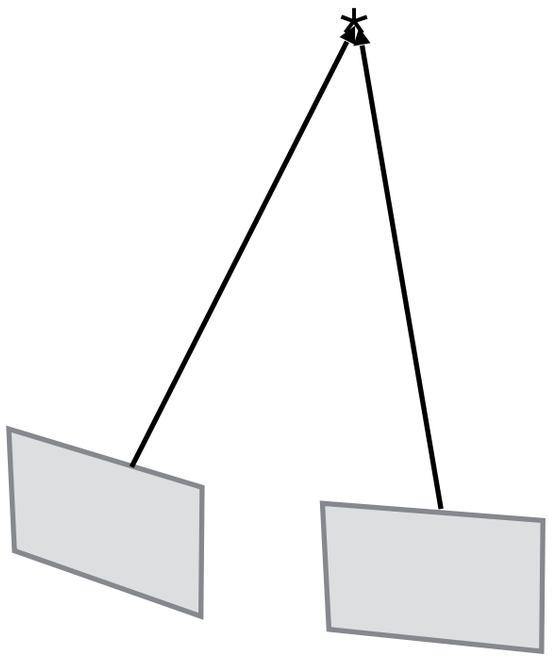
Sequential Structure-from-Motion

Initialize motion from two views



Relative pose for two images

Sequential Structure-from-Motion

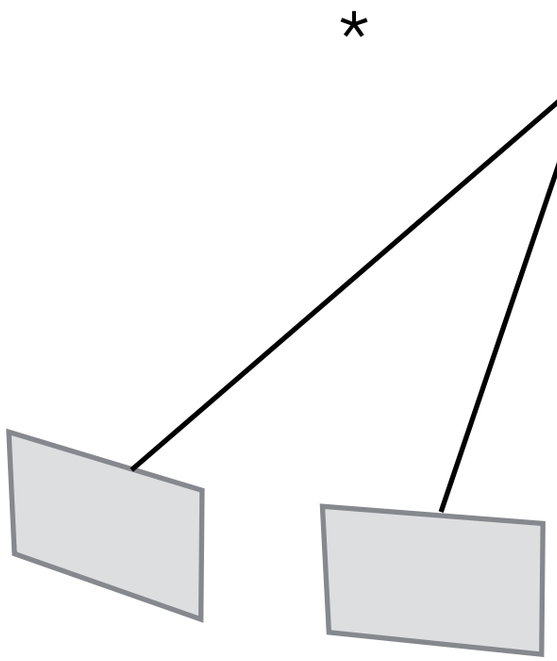


Initialize motion from two views

Initialize structure from two views

Triangulate 3D points

Sequential Structure-from-Motion

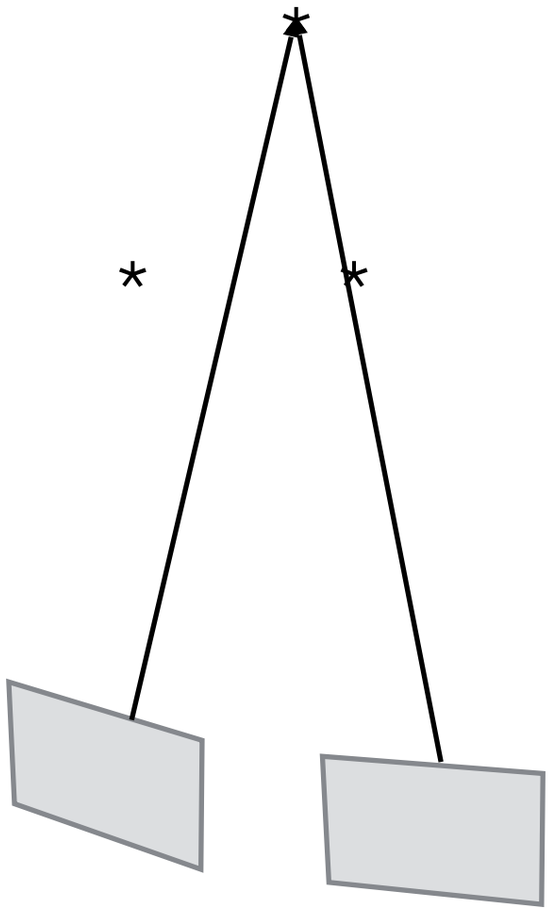


Initialize motion from two views

Initialize structure from two views

Triangulate 3D points

Sequential Structure-from-Motion

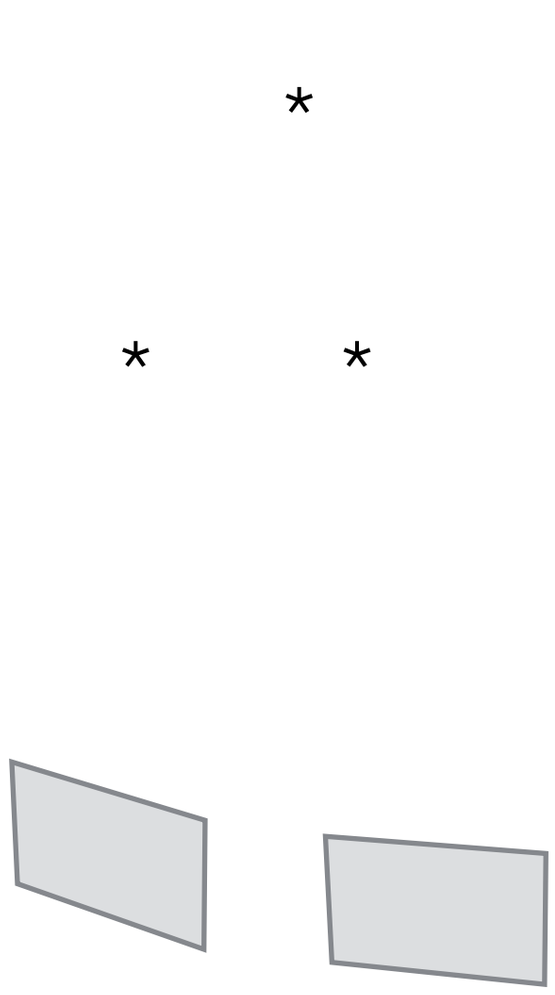


Initialize motion from two views

Initialize structure from two views

Triangulate 3D points

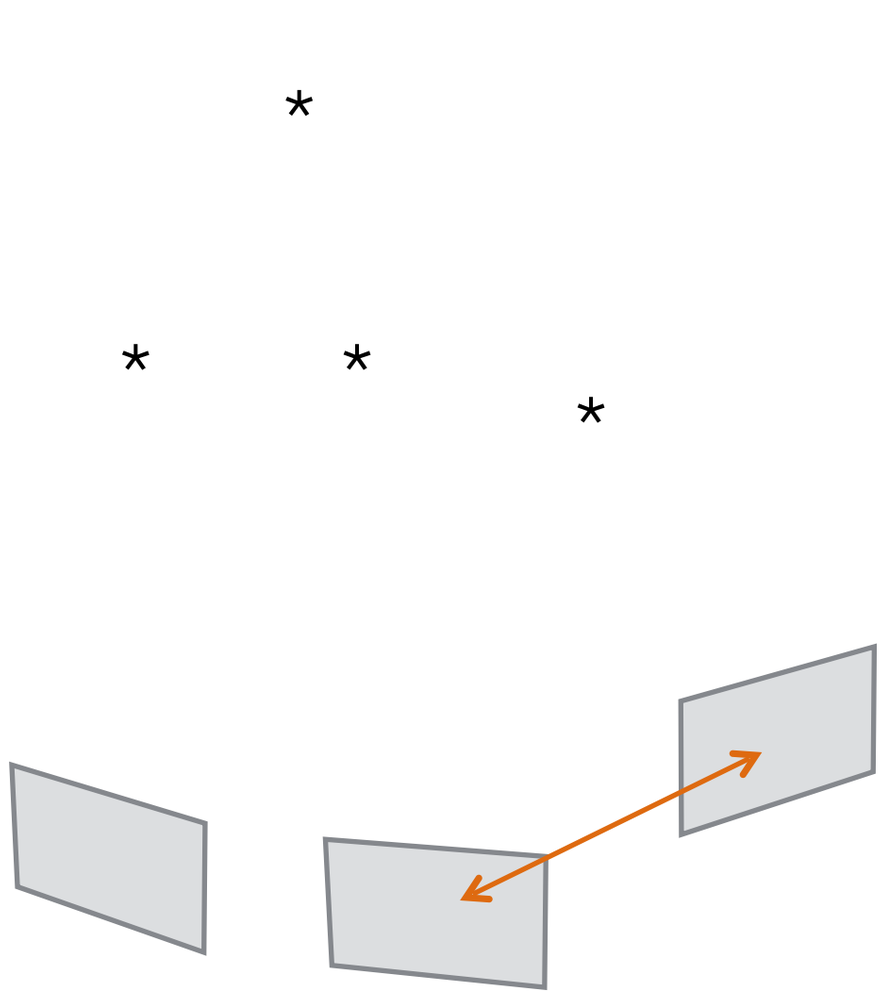
Sequential Structure-from-Motion



Initialize motion from two views

Initialize structure from two views

Sequential Structure-from-Motion



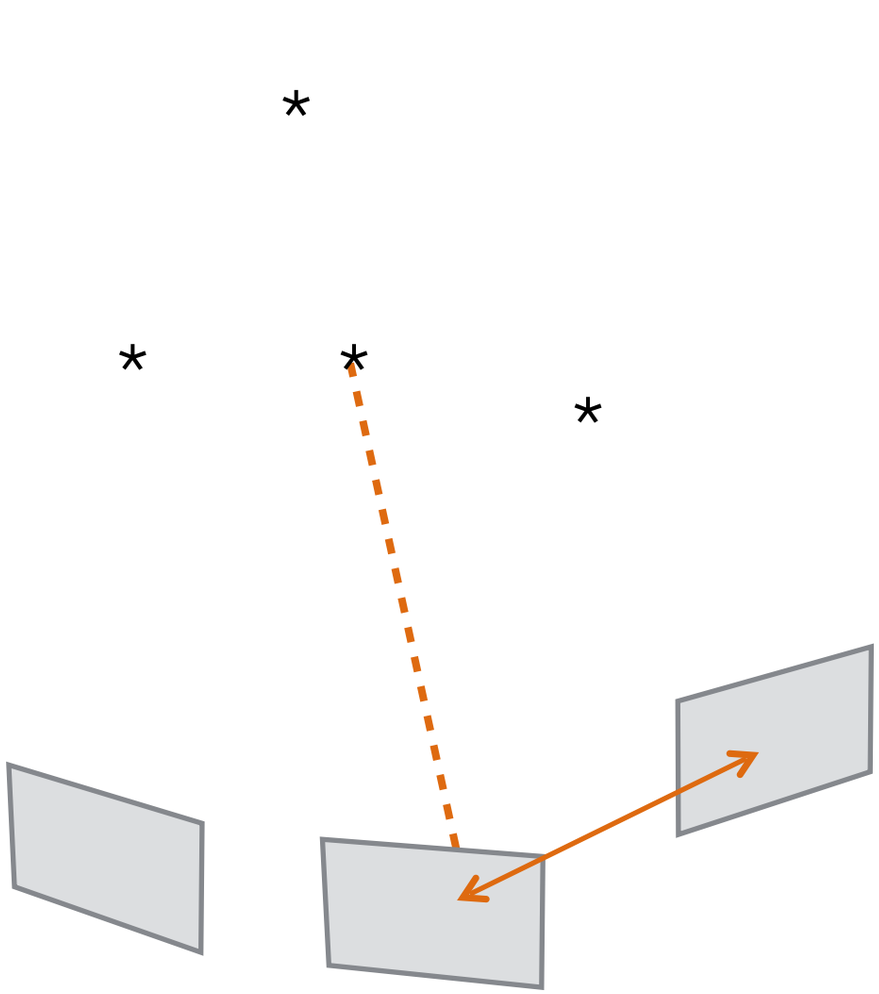
Initialize motion from two views

Initialize structure from two views

Extend motion

Match features

Sequential Structure-from-Motion



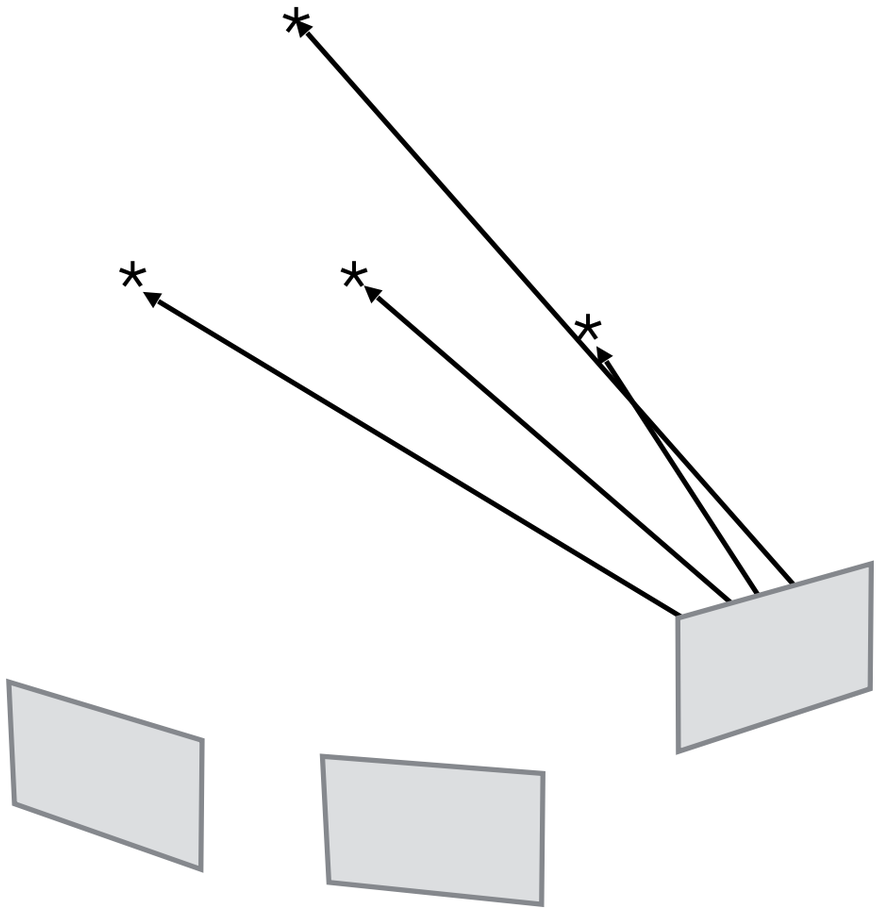
Initialize motion from two views

Initialize structure from two views

Extend motion

Transfer matches to 3D

Sequential Structure-from-Motion



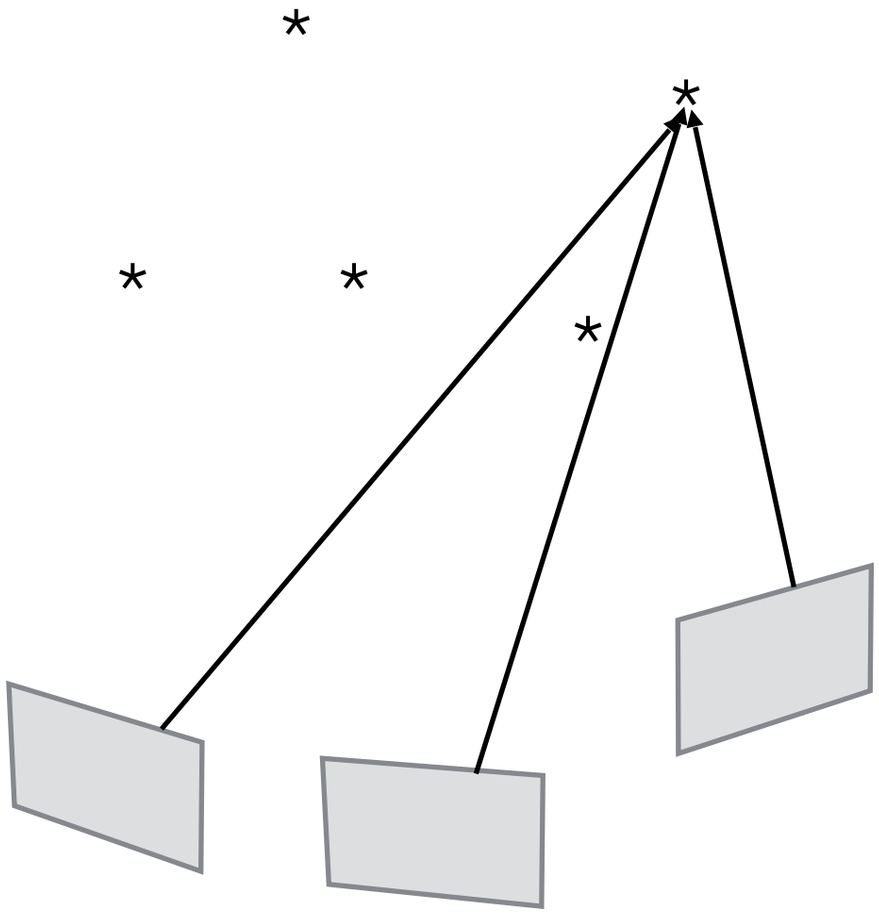
Initialize motion from two views

Initialize structure from two views

Extend motion

Camera pose for third camera

Sequential Structure-from-Motion



Initialize motion from two views

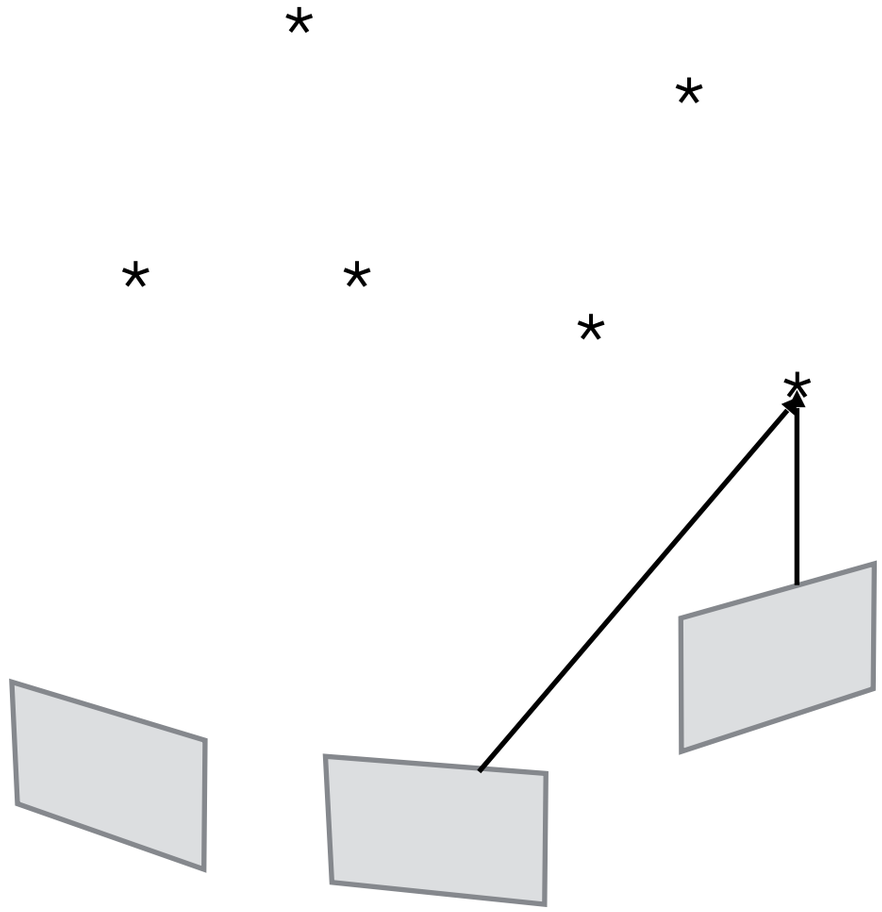
Initialize structure from two views

Extend motion

Extend structure

Triangulate points

Sequential Structure-from-Motion



Initialize motion from two views

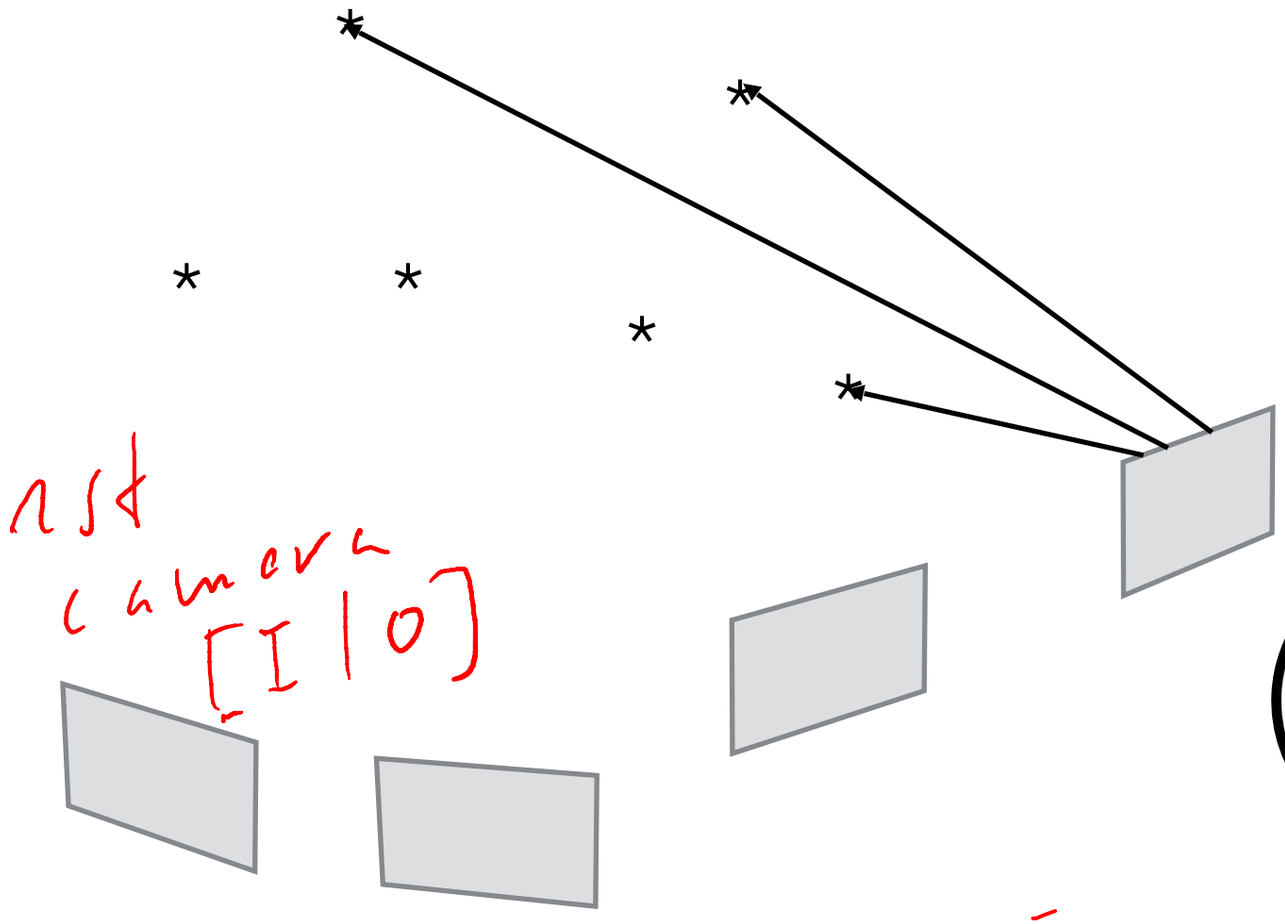
Initialize structure from two views

Extend motion

Extend structure

Triangulate points

Sequential Structure-from-Motion



Initialize motion from two views

Initialize structure from two views

Extend motion

Extend structure

Camera pose for fourth image

Today

- Relative Pose Estimation
- Triangulation
- Absolute Pose Estimation

Today

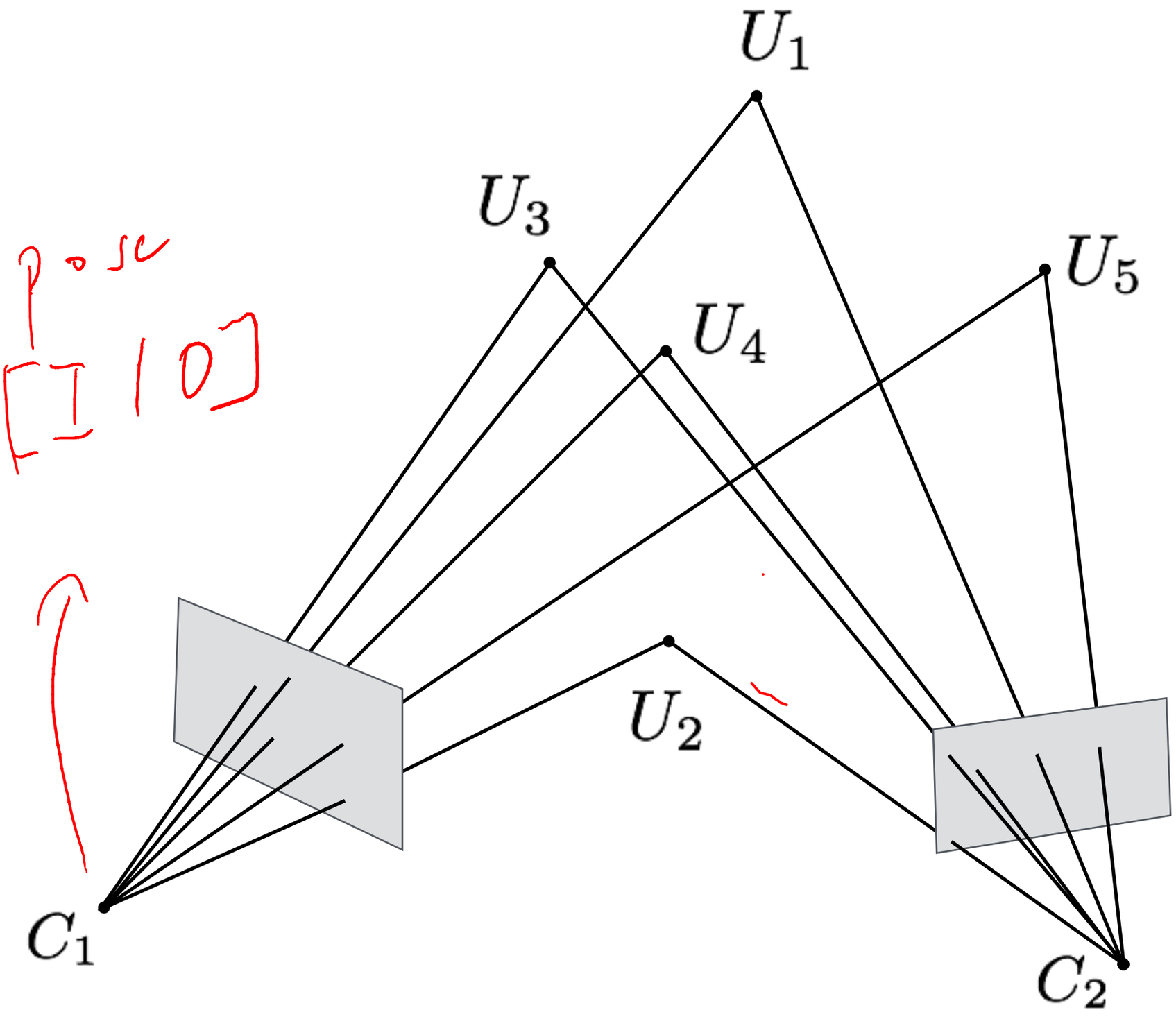
- **Relative Pose Estimation**
- Triangulation
- Absolute Pose Estimation

Cross Product as Matrix Multiplication

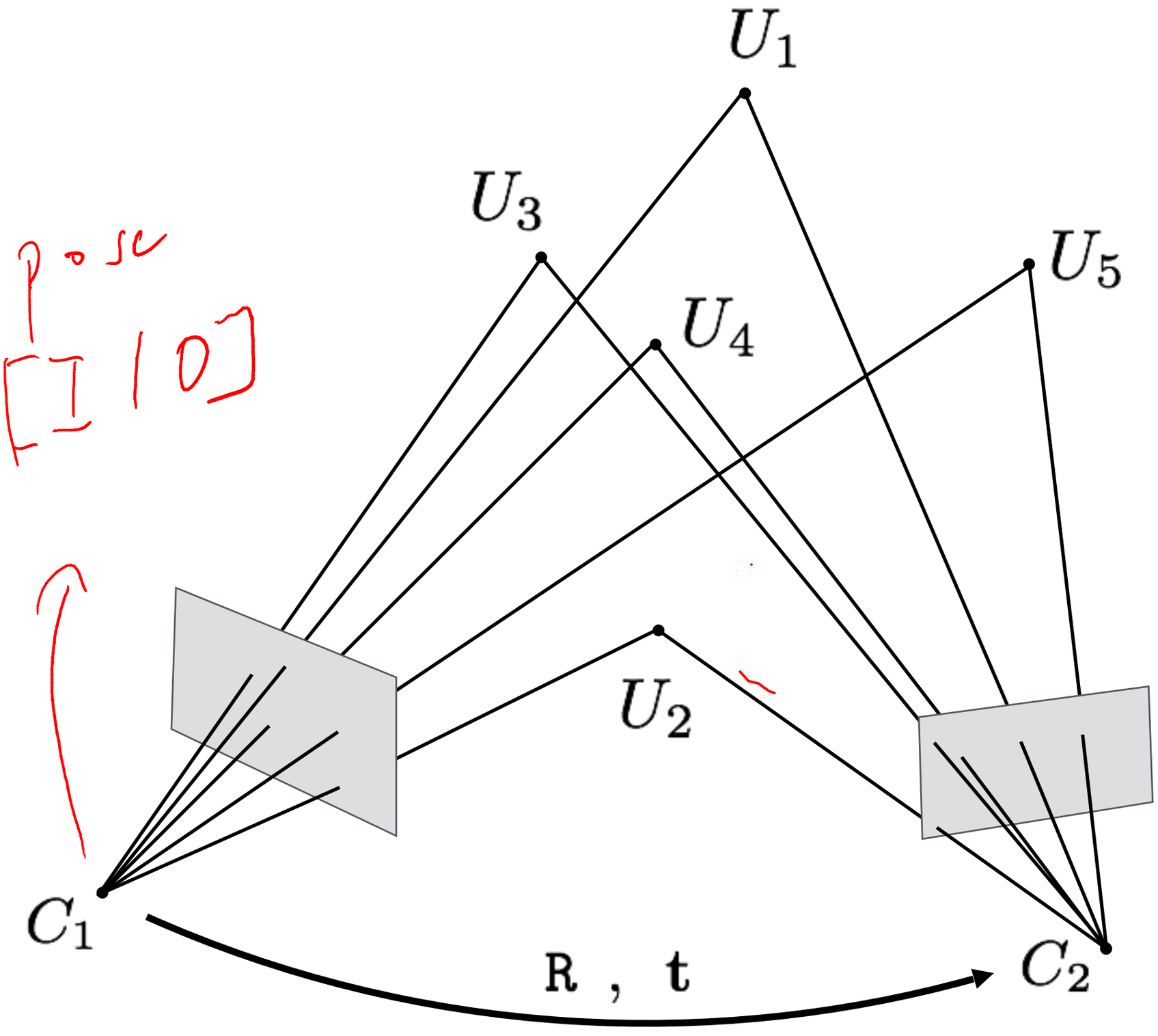
$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \times \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = [\mathbf{a}]_\times \mathbf{b}$$

skew symmetric matrix

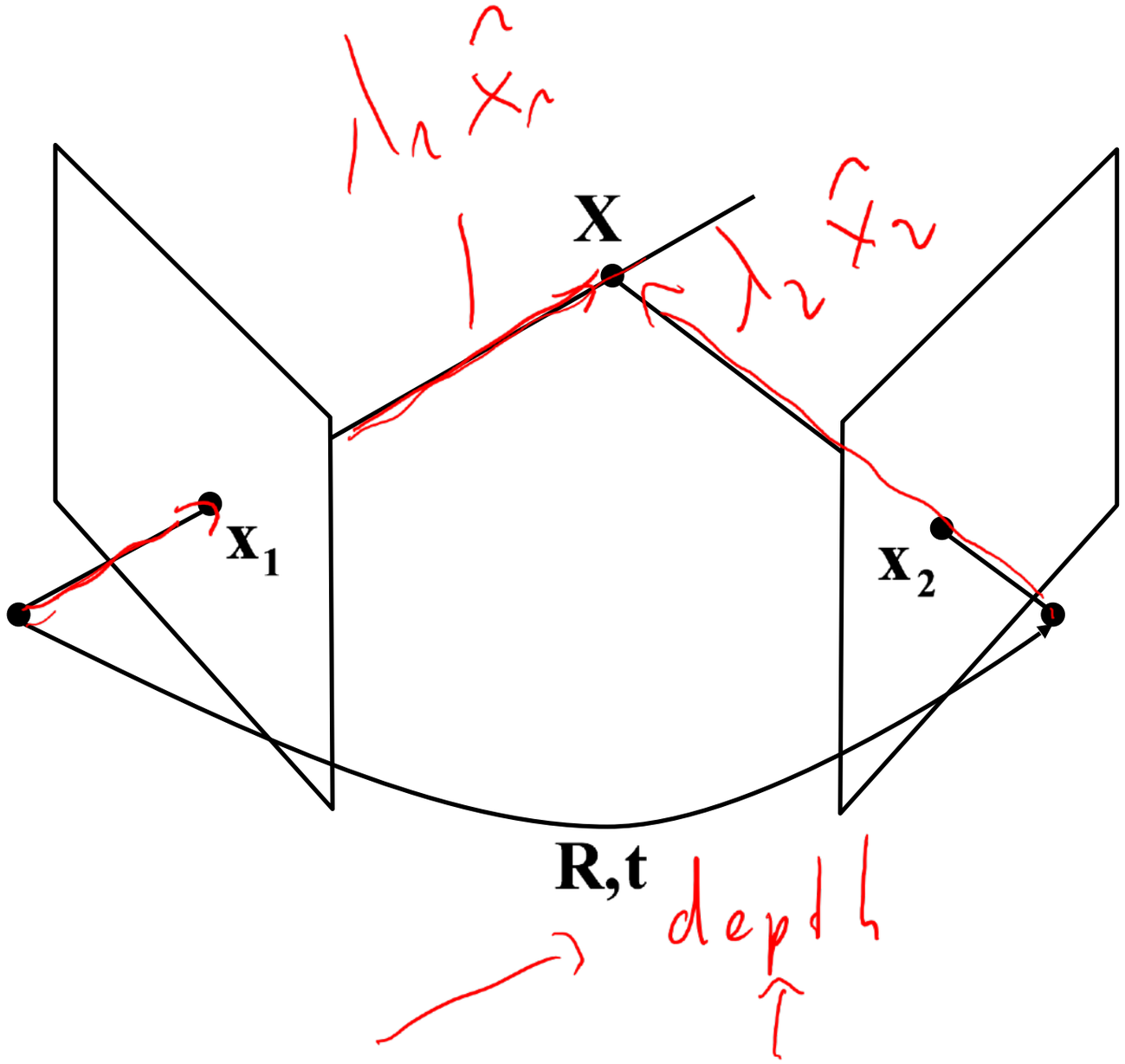
Relative Pose Estimation



Relative Pose Estimation

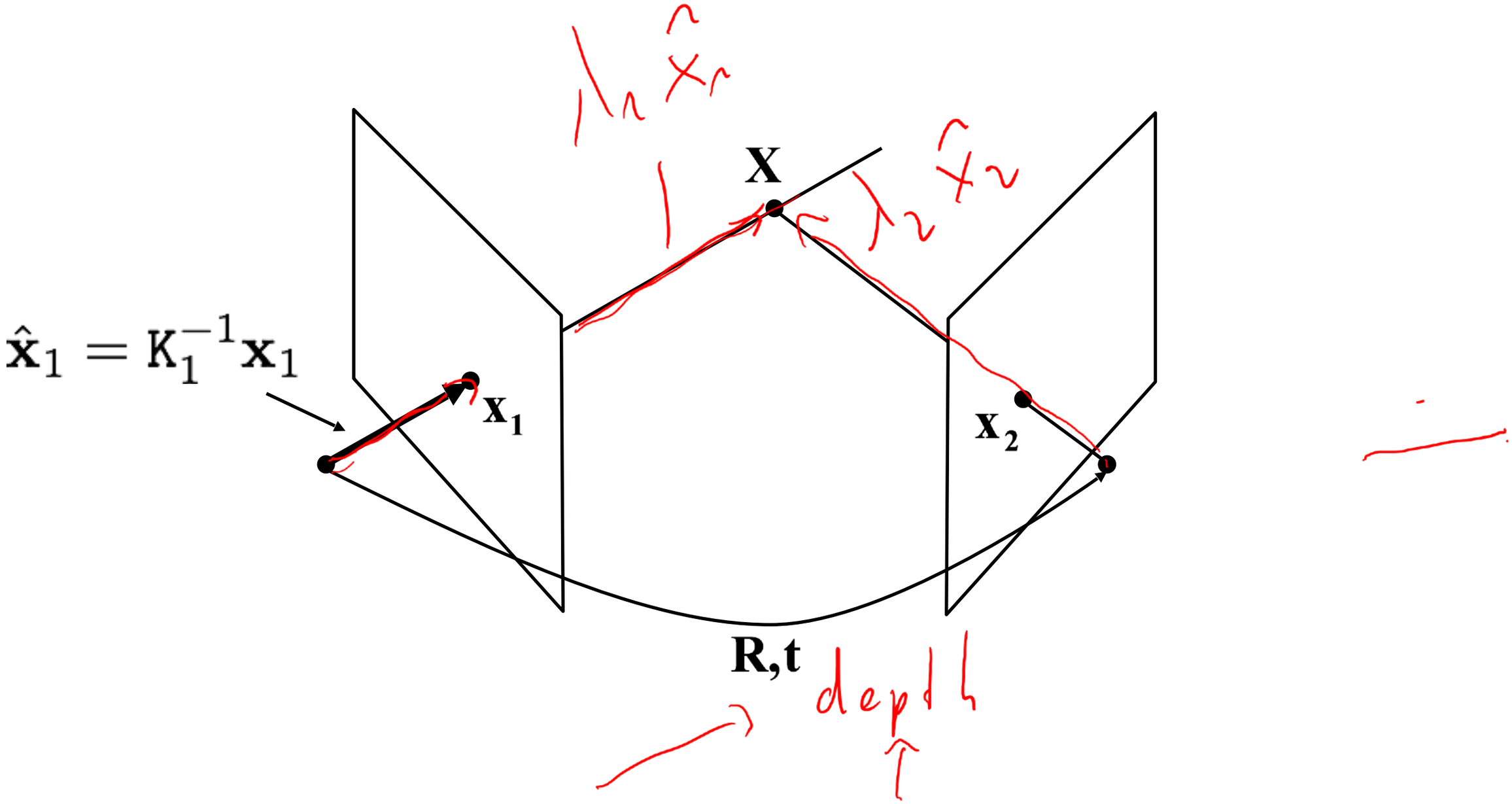


Epipolar Geometry

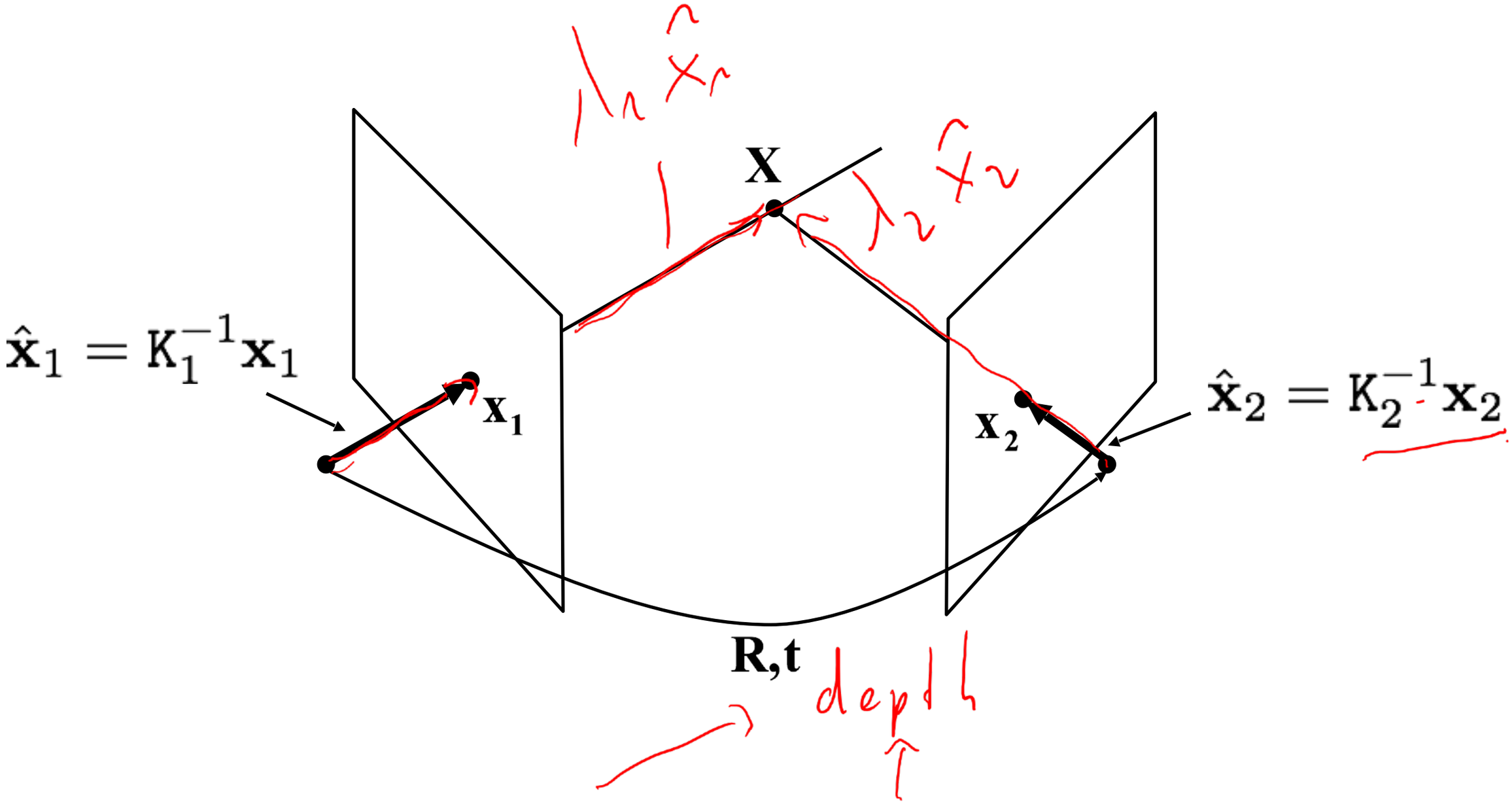


Epipolar Geometry

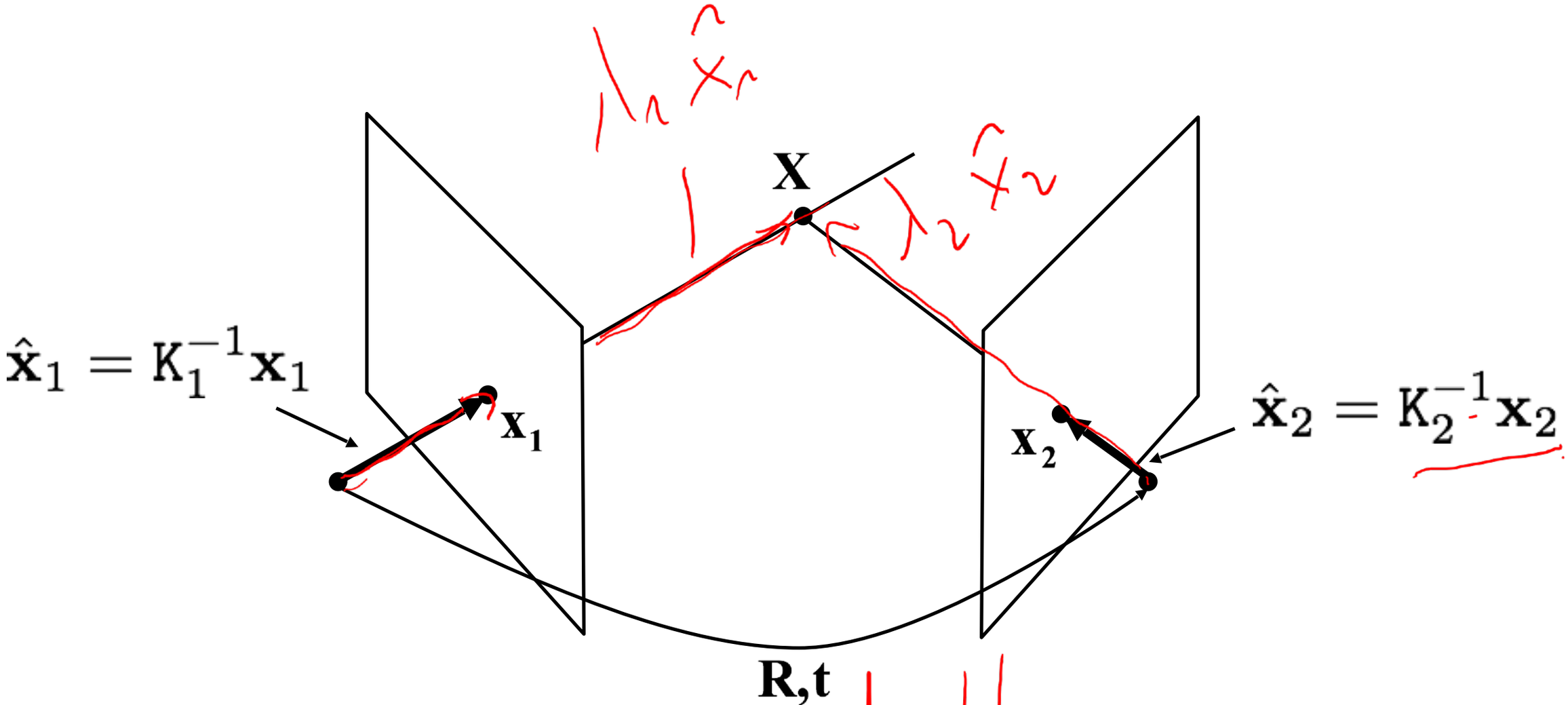
$$\hat{\mathbf{x}}_1 = \mathrm{K}_1^{-1}\mathbf{x}_1$$



Epipolar Geometry

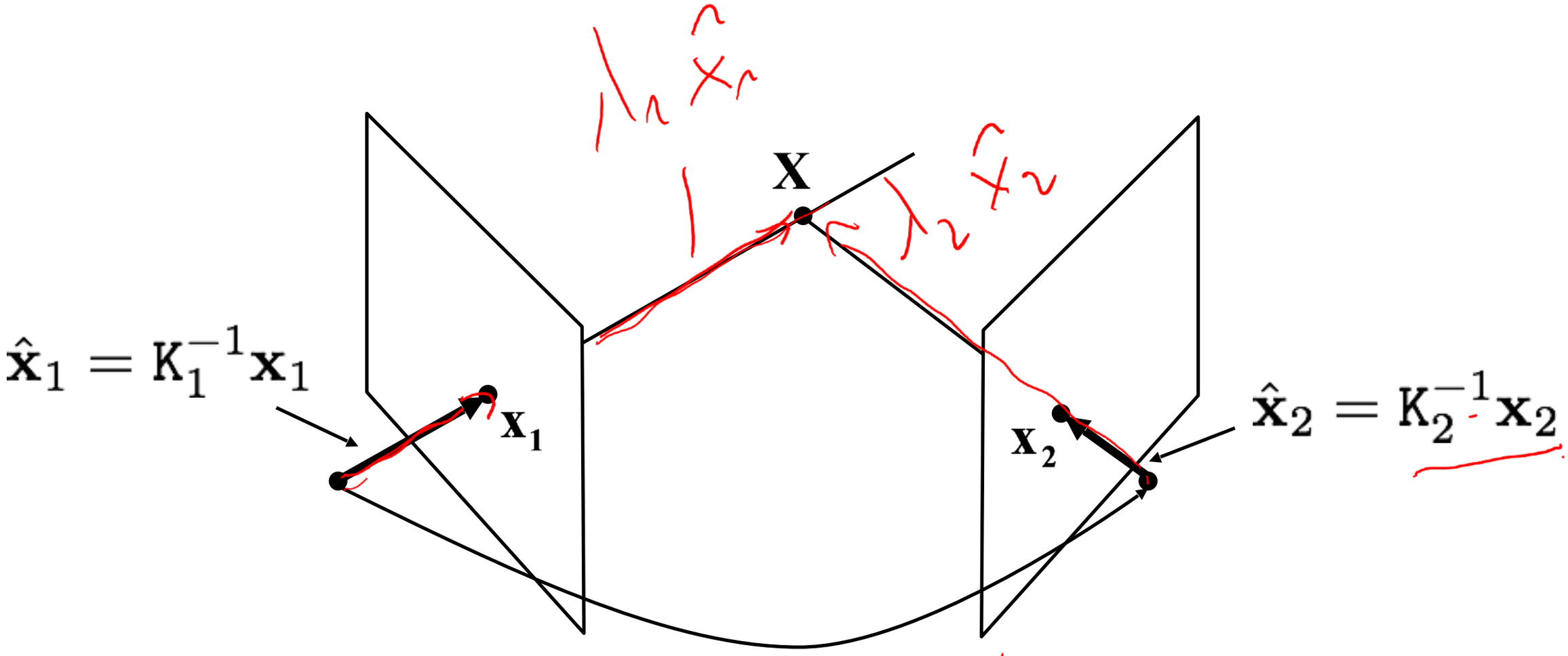


Epipolar Geometry



$$\mathbf{X} = \lambda_2 \hat{\mathbf{x}}_2 = \mathrm{R}\lambda_1 \hat{\mathbf{x}}_1 + \mathbf{t}$$

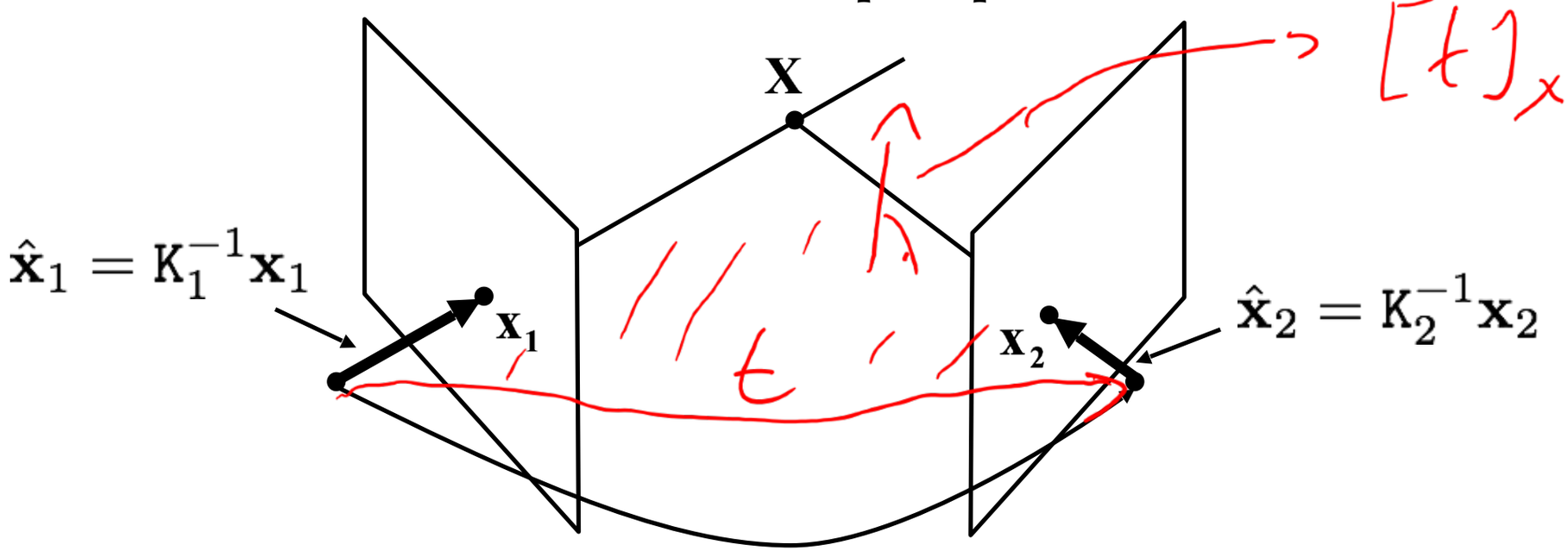
Epipolar Geometry



$$\mathbf{X} = \lambda_2 \hat{\mathbf{x}}_2 = \mathrm{R}\lambda_1 \hat{\mathbf{x}}_1 + \mathbf{t}$$

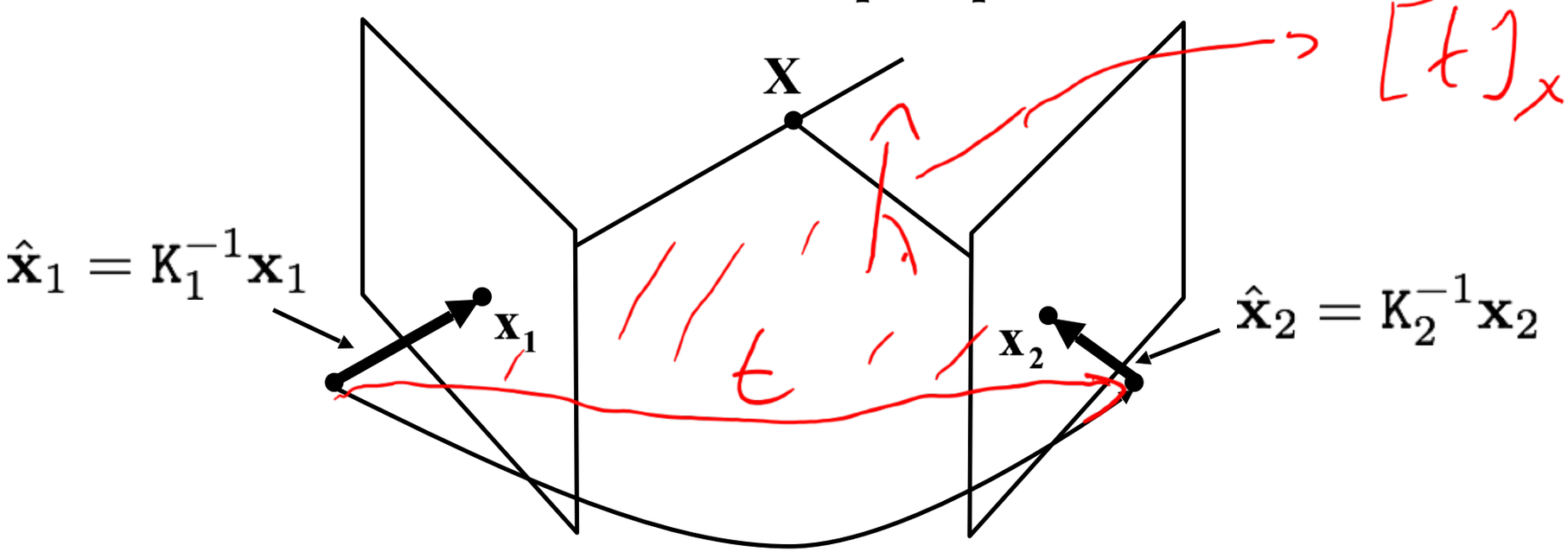
unknown

Epipolar Geometry



$$\lambda_2\hat{\mathbf{x}}_2 = \mathrm{R}\lambda_1\hat{\mathbf{x}}_1 + \mathbf{t}$$

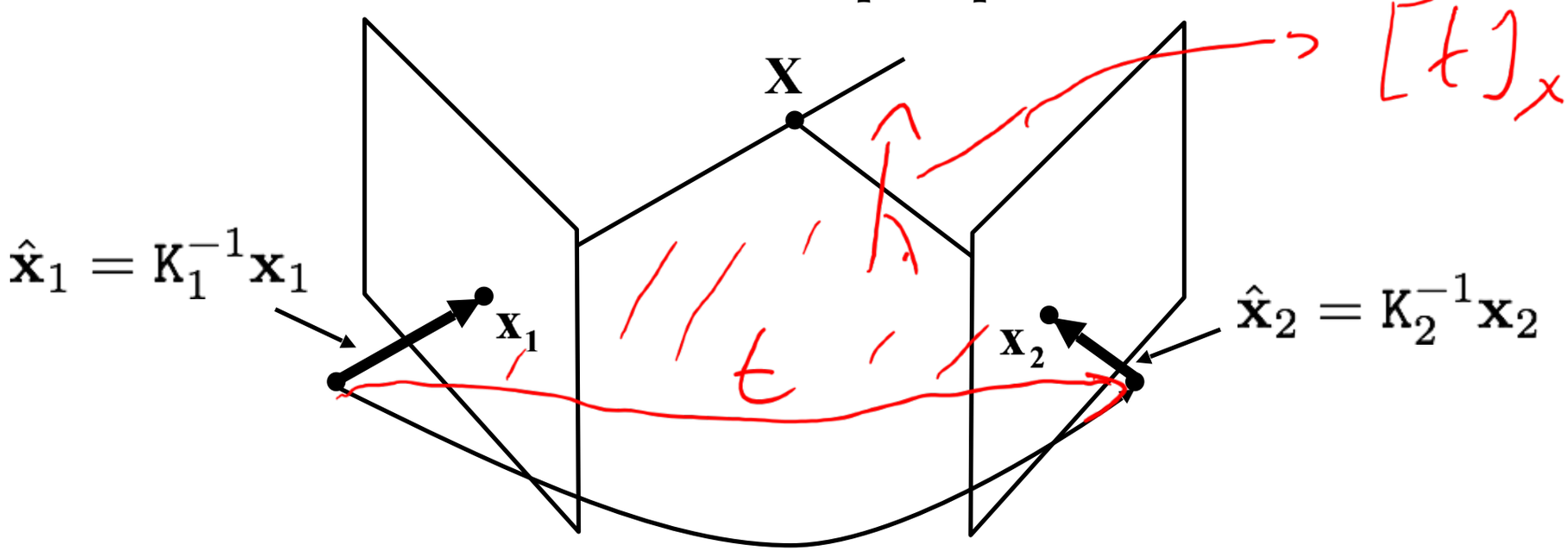
Epipolar Geometry



$$\lambda_2\hat{\mathbf{x}}_2 = \mathrm{R}\lambda_1\hat{\mathbf{x}}_1 + \mathbf{t}$$

$$\Leftrightarrow \quad [\mathbf{t}]_\times\lambda_2\hat{\mathbf{x}}_2 = [\mathbf{t}]_\times\mathrm{R}\lambda_1\hat{\mathbf{x}}_1 + [\mathbf{t}]_\times\mathbf{t}$$

Epipolar Geometry

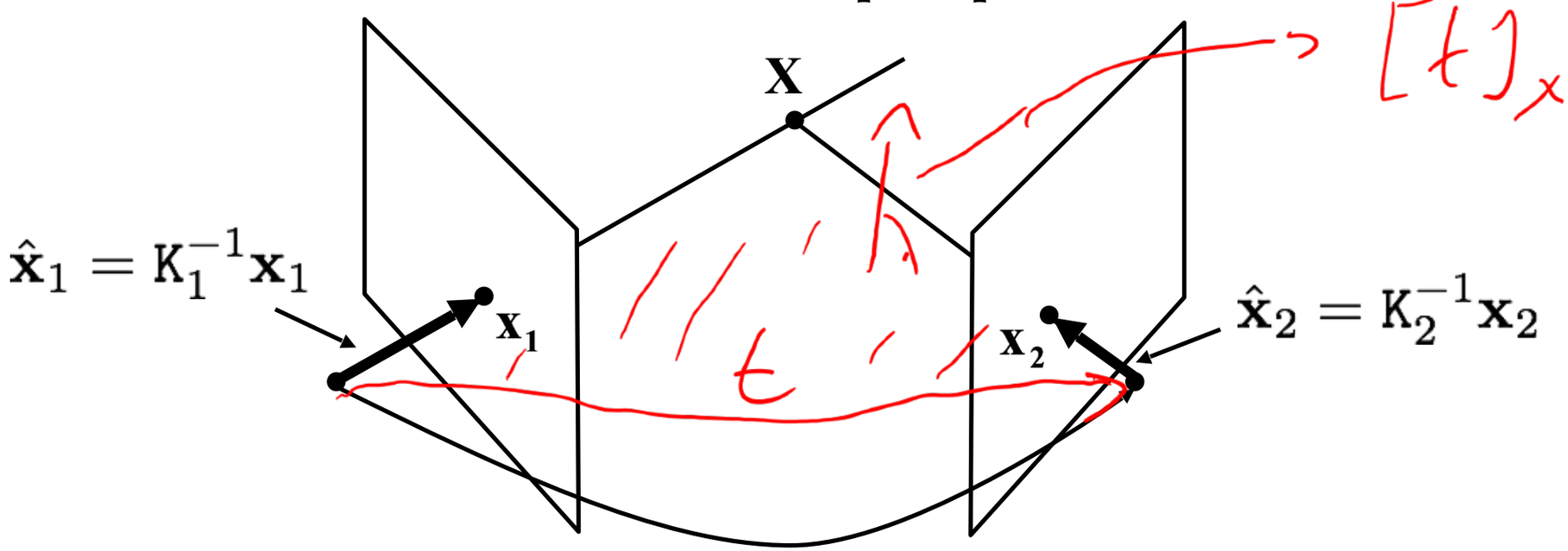


$$\lambda_2\hat{\mathbf{x}}_2 = \mathrm{R}\lambda_1\hat{\mathbf{x}}_1 + \mathbf{t}$$

$$\Leftrightarrow \quad [\mathbf{t}]_\times\lambda_2\hat{\mathbf{x}}_2 = [\mathbf{t}]_\times\mathrm{R}\lambda_1\hat{\mathbf{x}}_1 + [\mathbf{t}]_\times\mathbf{t}$$

$$\Leftrightarrow \quad \hat{\mathbf{x}}_2^T[\mathbf{t}]_\times\lambda_2\hat{\mathbf{x}}_2 = \hat{\mathbf{x}}_2^T[\mathbf{t}]_\times\mathrm{R}\lambda_1\hat{\mathbf{x}}_1$$

Epipolar Geometry



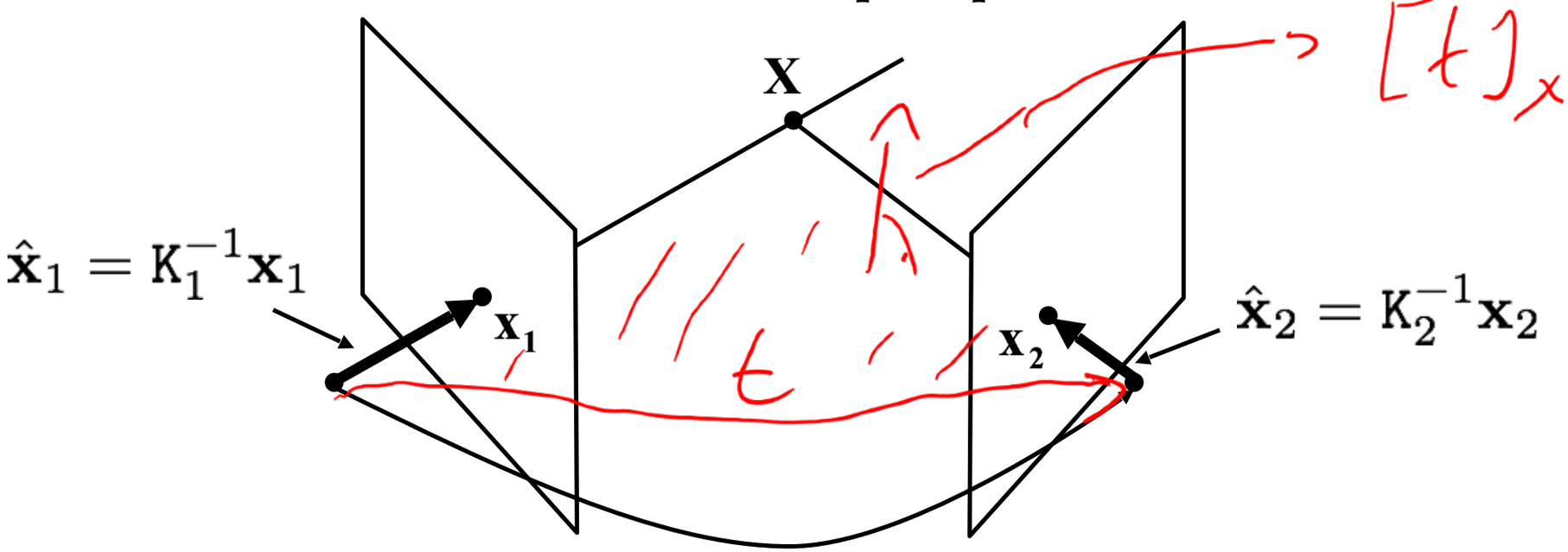
$$\lambda_2\hat{\mathbf{x}}_2 = \mathrm{R}\lambda_1\hat{\mathbf{x}}_1 + \mathbf{t}$$

$$\Leftrightarrow \quad [\mathbf{t}]_\times\lambda_2\hat{\mathbf{x}}_2 = [\mathbf{t}]_\times\mathrm{R}\lambda_1\hat{\mathbf{x}}_1 + [\mathbf{t}]_\times\mathbf{t}$$

$$\Leftrightarrow \quad \hat{\mathbf{x}}_2^T[\mathbf{t}]_\times\lambda_2\hat{\mathbf{x}}_2 = \hat{\mathbf{x}}_2^T[\mathbf{t}]_\times\mathrm{R}\lambda_1\hat{\mathbf{x}}_1$$

$$\Leftrightarrow \quad \mathbf{0} = \hat{\mathbf{x}}_2^T[\mathbf{t}]_\times\mathrm{R}\lambda_1\hat{\mathbf{x}}_1$$

Epipolar Geometry



$$\lambda_2\hat{\mathbf{x}}_2 = \mathrm{R}\lambda_1\hat{\mathbf{x}}_1 + \mathbf{t}$$

$$\Leftrightarrow \quad [\mathbf{t}]_\times\lambda_2\hat{\mathbf{x}}_2 = [\mathbf{t}]_\times\mathrm{R}\lambda_1\hat{\mathbf{x}}_1 + [\mathbf{t}]_\times\mathbf{t}$$

$$\Leftrightarrow \quad \hat{\mathbf{x}}_2^T[\mathbf{t}]_\times\lambda_2\hat{\mathbf{x}}_2 = \hat{\mathbf{x}}_2^T[\mathbf{t}]_\times\mathrm{R}\lambda_1\hat{\mathbf{x}}_1$$

$$\Leftrightarrow \quad \mathbf{0} = \hat{\mathbf{x}}_2^T[\mathbf{t}]_\times\mathrm{R}\lambda_1\hat{\mathbf{x}}_1$$

$$\Leftrightarrow \quad \mathbf{0} = \hat{\mathbf{x}}_2^T[\mathbf{t}]_\times\mathrm{R}\hat{\mathbf{x}}_1$$

The Essential Matrix

$$\mathbf{0} = \hat{\mathbf{x}}_2^T [\mathbf{t}]_\times \mathbf{R} \hat{\mathbf{x}}_1$$

↳ essential matrix

The Essential Matrix

$$\mathbf{0} = \hat{\mathbf{x}}_2^T [\mathbf{t}]_\times \mathbf{R} \hat{\mathbf{x}}_1$$

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

↳ essential matrix

The Essential Matrix

$$\mathbf{0} = \hat{\mathbf{x}}_2^T [\mathbf{t}]_\times \mathbf{R} \hat{\mathbf{x}}_1$$

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

→ essential matrix

- $\mathbf{E}$ is 3x3 matrix, has 5 DoF (degrees-of-freedom)
- $\mathbf{E}$ has two equal singular values, third singular value is 0

The Essential Matrix

$$\mathbf{0} = \hat{\mathbf{x}}_2^T [\mathbf{t}]_\times \mathbf{R} \hat{\mathbf{x}}_1$$

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

→ essential matrix

- $\mathbf{E}$ is 3x3 matrix, has 5 DoF (degrees-of-freedom)

- $\mathbf{E}$ has two equal singular values, third singular value is 0
- $\mathbf{E}$ has rank 2

The Fundamental Matrix

$$\hat{x}_i = K_i^{-1} x_i$$

$$(K_2^{-1} x_2)^T$$

$$= x_2^T K_2^{-T}$$

fundamental matrix

The Fundamental Matrix

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

$\hat{x}_i = K_i^{-1} x_i$

$(K_2^{-1} x_2)^T$

$= x_2^T K_2^{-T}$

fundamental matrix

The Fundamental Matrix

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

$$\hat{x}_i = K_i^{-1} x_i$$

$$\mathbf{0} = \mathbf{x}_2^T \mathbf{K}_2^{-T} \mathbf{E} \mathbf{K}_1^{-1} \mathbf{x}_1$$

$$(K_2^{-1} x_2)^T = x_2^T K_2^{-T}$$

fundamental matrix

The Fundamental Matrix

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

$$\mathbf{0} = \mathbf{x}_2^T \mathbf{K}_2^{-T} \mathbf{E} \mathbf{K}_1^{-1} \mathbf{x}_1$$

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

$\hat{x}_i = K_i^{-1} x_i$

$(K_2^{-1} x_2)^T$

$= x_2^T K_2^{-T}$

fundamental matrix

The Fundamental Matrix

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

$$\mathbf{0} = \mathbf{x}_2^T \mathbf{K}_2^{-T} \mathbf{E} \mathbf{K}_1^{-1} \mathbf{x}_1$$

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

- $\mathbf{F}$ has 7 DoF
- $\mathbf{F}$ has rank 2

The Fundamental Matrix

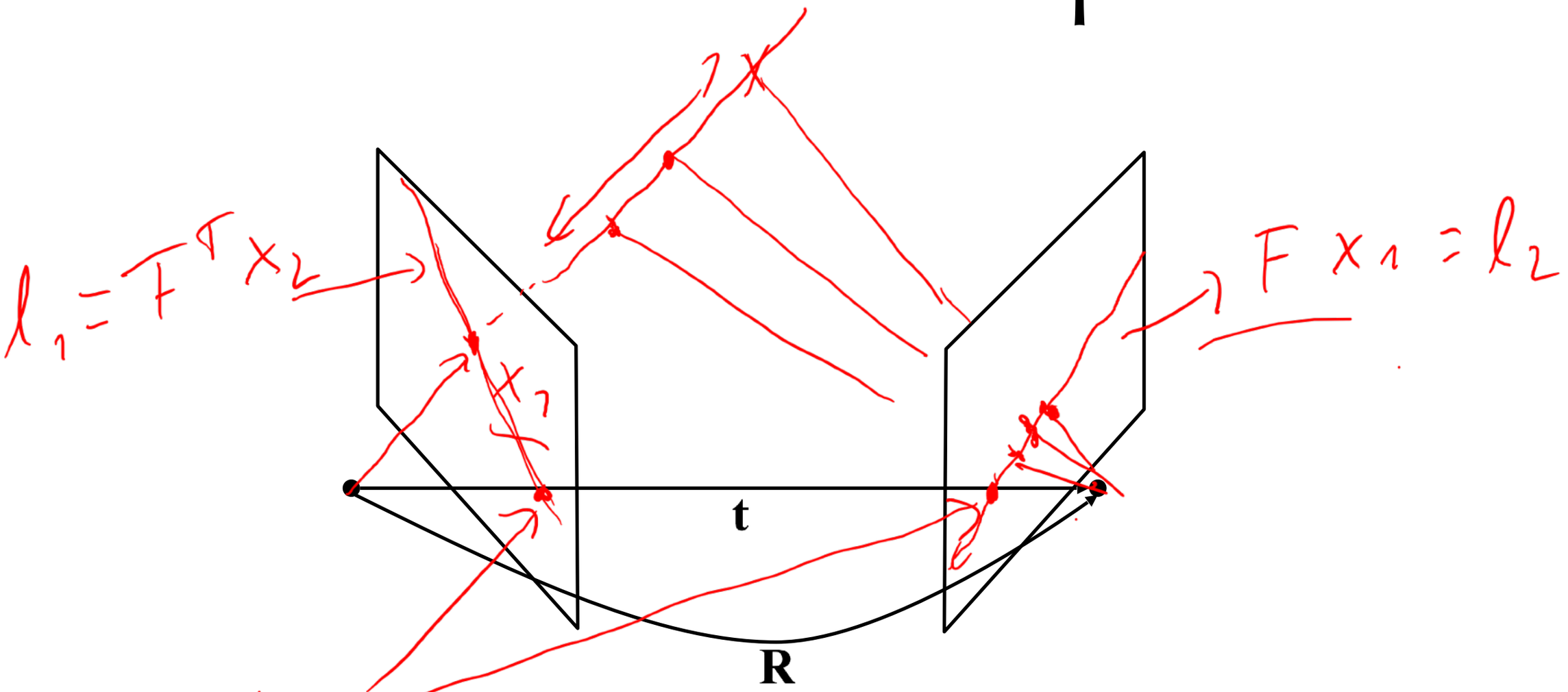
$$\mathbf{0} = \hat{\mathbf{x}}_2^T \mathbf{E} \hat{\mathbf{x}}_1$$

$$\mathbf{0} = \mathbf{x}_2^T \mathbf{K}_2^{-T} \mathbf{E} \mathbf{K}_1^{-1} \mathbf{x}_1$$

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

- $\mathbf{F}$ has 7 DoF
- $\mathbf{F}$ has rank 2
- Computing $\mathbf{F}$ does not require intrinsic calibration

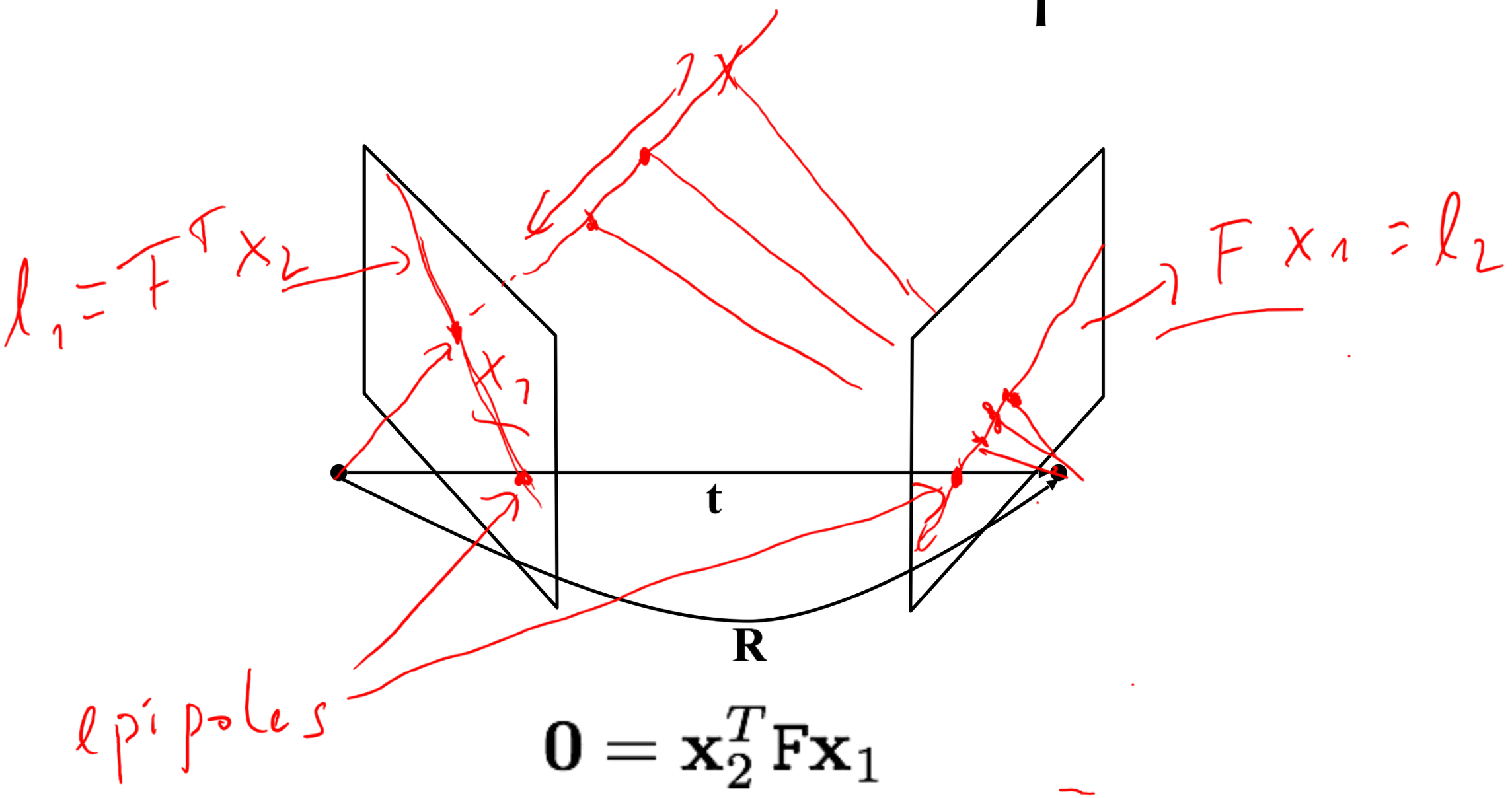
Geometric Interpretation



epipoles

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

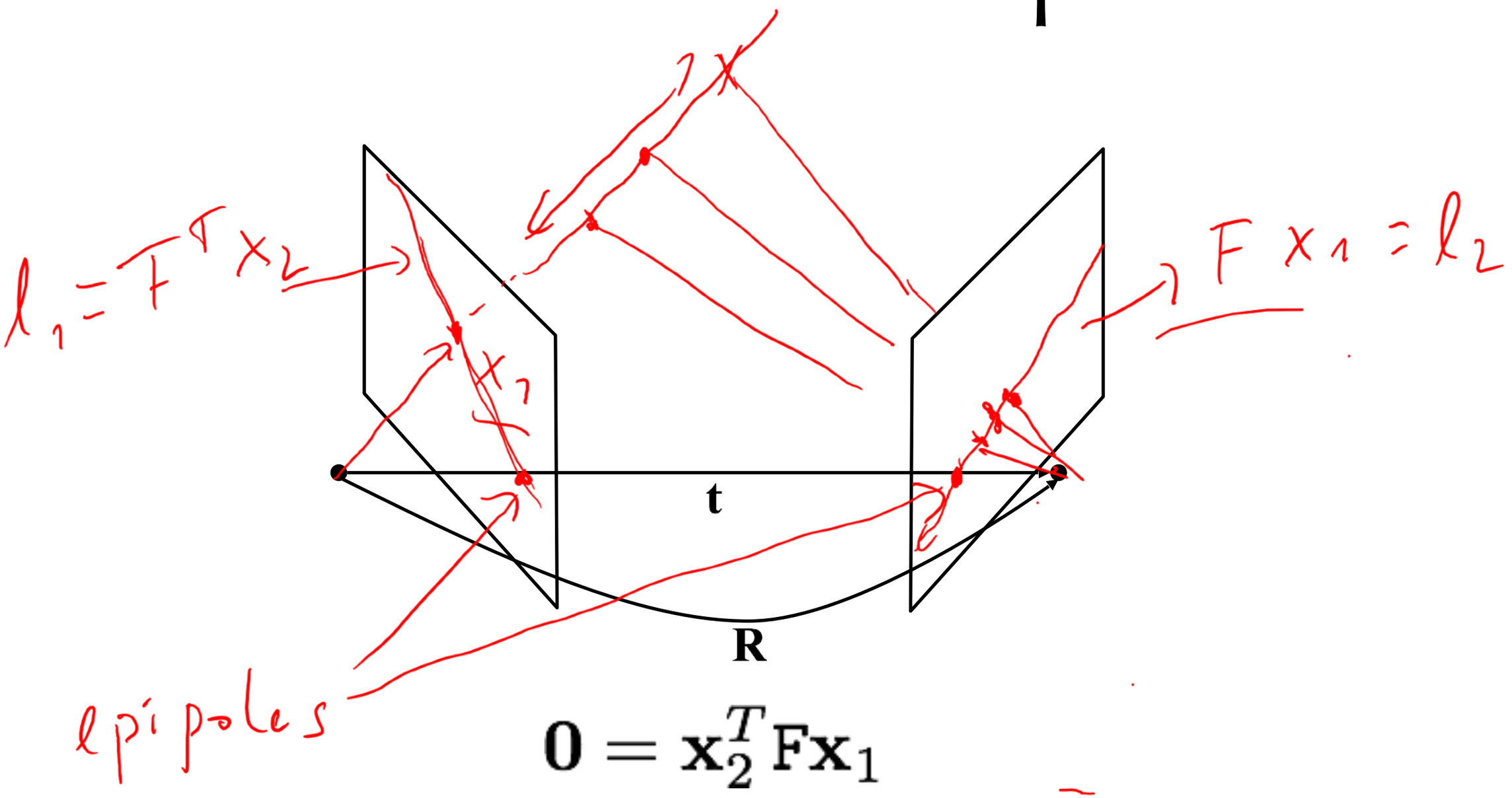
Geometric Interpretation



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

- $\mathbf{F}$ maps points in first image to lines in second image
- $\mathbf{F}^T$ maps points in second image to lines in first image

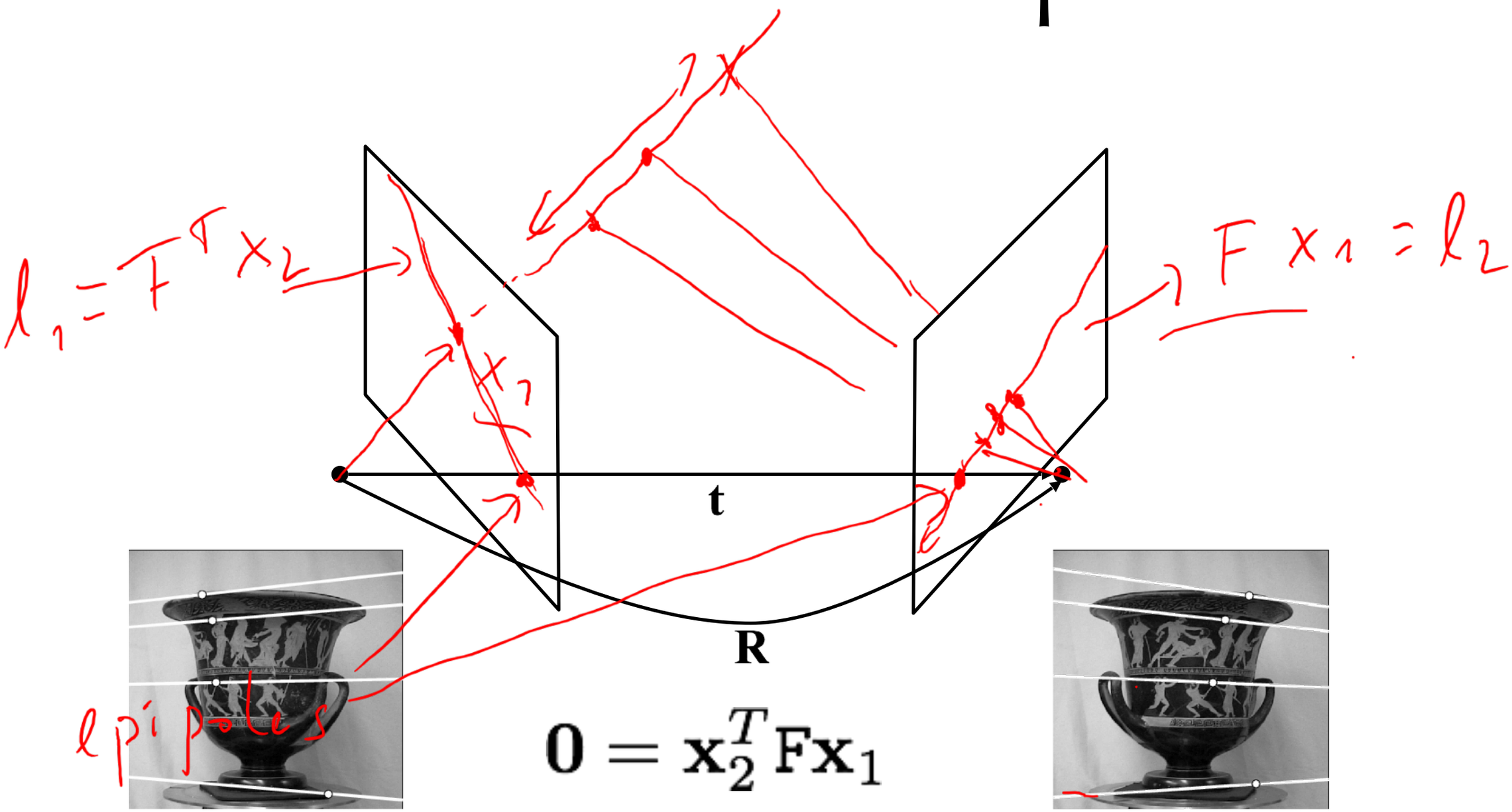
Geometric Interpretation



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

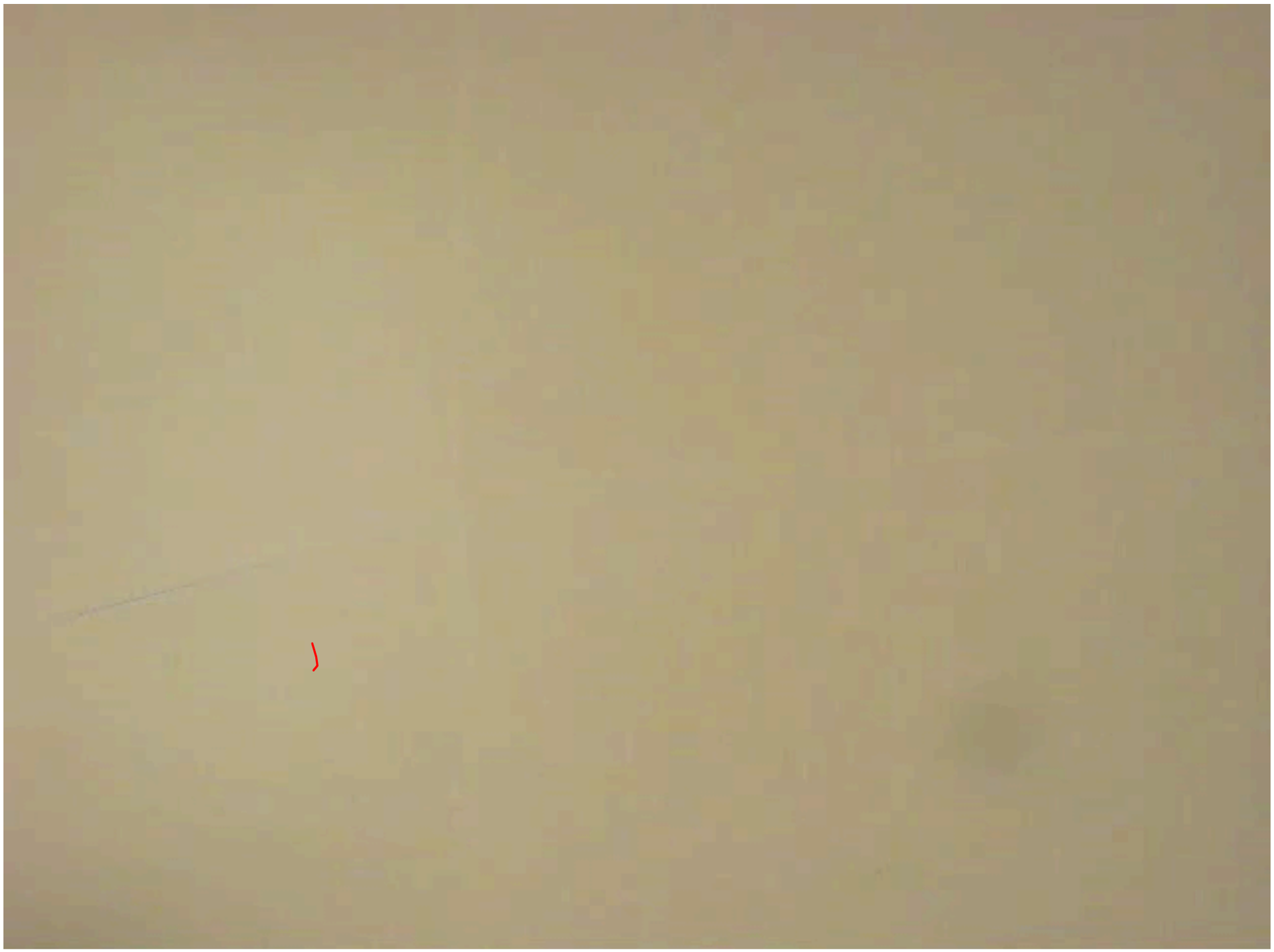
- $\mathbf{F}$ maps points in first image to lines in second image
- $\mathbf{F}^T$ maps points in second image to lines in first image
- Lines are called **epipolar lines**

Geometric Interpretation



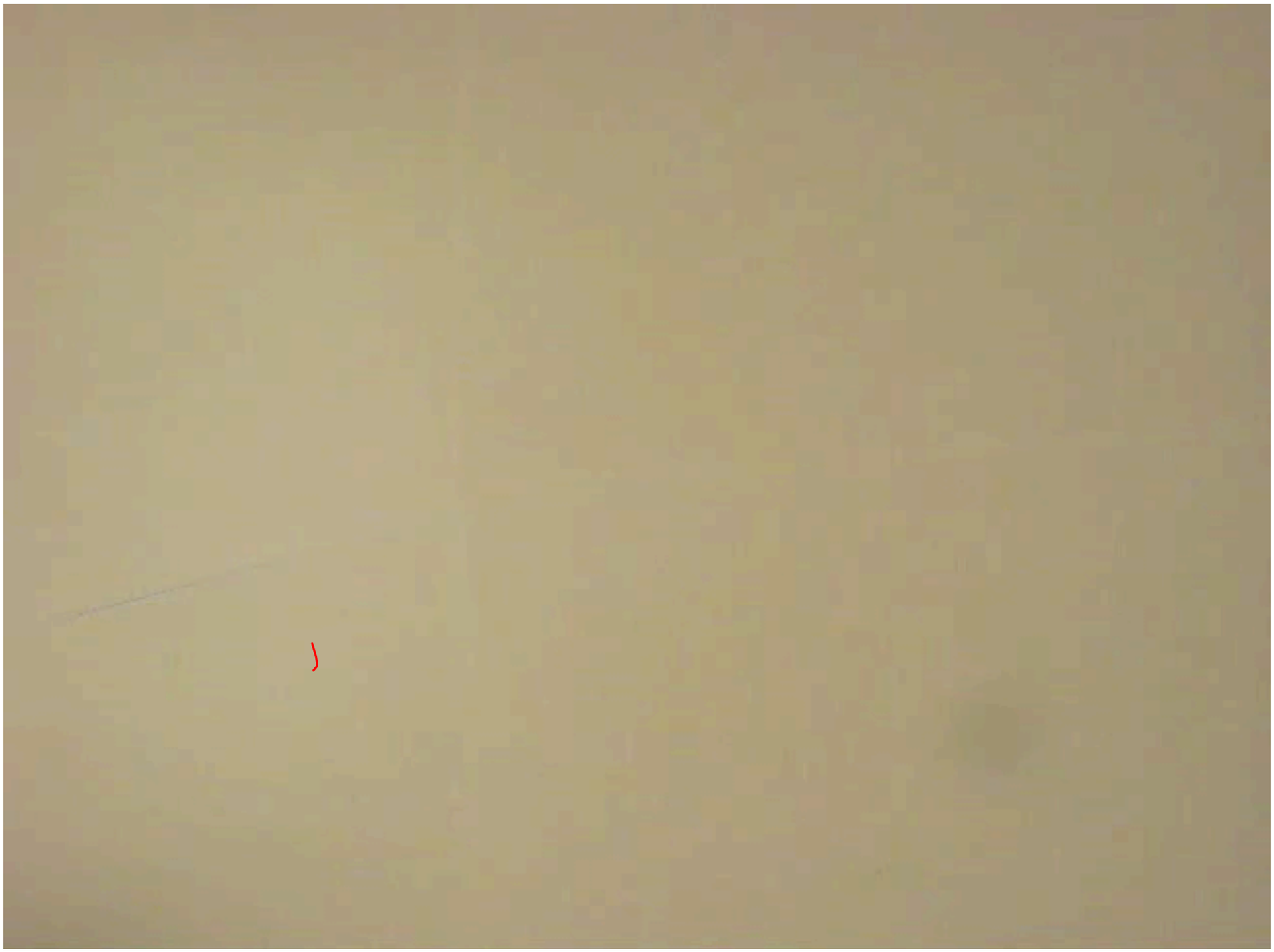
- $\mathbf{F}$ maps points in first image to lines in second image
- $\mathbf{F}^T$ maps points in second image to lines in first image
- Lines are called **epipolar lines**

Want to Know More?



http://danielwedge.com/fmatrix/

Want to Know More?



http://danielwedge.com/fmatrix/

Computing $\mathbf{E}$ and $\mathbf{F}$

Computing $\mathbf{E}$ and $\mathbf{F}$

- Estimate 2D-2D matches between images
- Compute $\mathbf{E}$ / $\mathbf{F}$ using RANSAC:

Computing $\mathbf{E}$ and $\mathbf{F}$

- Estimate 2D-2D matches between images
- Compute $\mathbf{E}$ / $\mathbf{F}$ using RANSAC:
  - Linear solver (8 points): $\mathbf{E}$ and $\mathbf{F}$

Computing $\mathbf{E}$ and $\mathbf{F}$

- Estimate 2D-2D matches between images
- Compute $\mathbf{E}$ / $\mathbf{F}$ using RANSAC:
  - Linear solver (8 points): $\mathbf{E}$ and $\mathbf{F}$
  - Minimal solver (7 points): $\mathbf{E}$ and $\mathbf{F}$

Computing $\mathbf{E}$ and $\mathbf{F}$

- Estimate 2D-2D matches between images
- Compute $\mathbf{E}$ / $\mathbf{F}$ using RANSAC:
  - Linear solver (8 points): $\mathbf{E}$ and $\mathbf{F}$
  - Minimal solver (7 points): $\mathbf{E}$ and $\mathbf{F}$
  - Calibrated solver (5 points): Only $\mathbf{E}$

Computing $\mathbf{E}$ and $\mathbf{F}$

- Estimate 2D-2D matches between images
- Compute $\mathbf{E}$ / $\mathbf{F}$ using RANSAC:
  - Linear solver (8 points): $\mathbf{E}$ and $\mathbf{F}$
  - Minimal solver (7 points): $\mathbf{E}$ and $\mathbf{F}$
  - Calibrated solver (5 points): Only $\mathbf{E}$
- Measure error using Sampson Error (see exercise)

Computing $\mathbf{E}$ and $\mathbf{F}$

- Estimate 2D-2D matches between images
- Compute $\mathbf{E}$ / $\mathbf{F}$ using RANSAC:
  - Linear solver (8 points): $\mathbf{E}$ and $\mathbf{F}$
  - Minimal solver (7 points): $\mathbf{E}$ and $\mathbf{F}$
  - Calibrated solver (5 points): Only $\mathbf{E}$
  - Measure error using Sampson Error (see exercise)
- Refine $\mathbf{E}$ / $\mathbf{F}$ based on all inliers

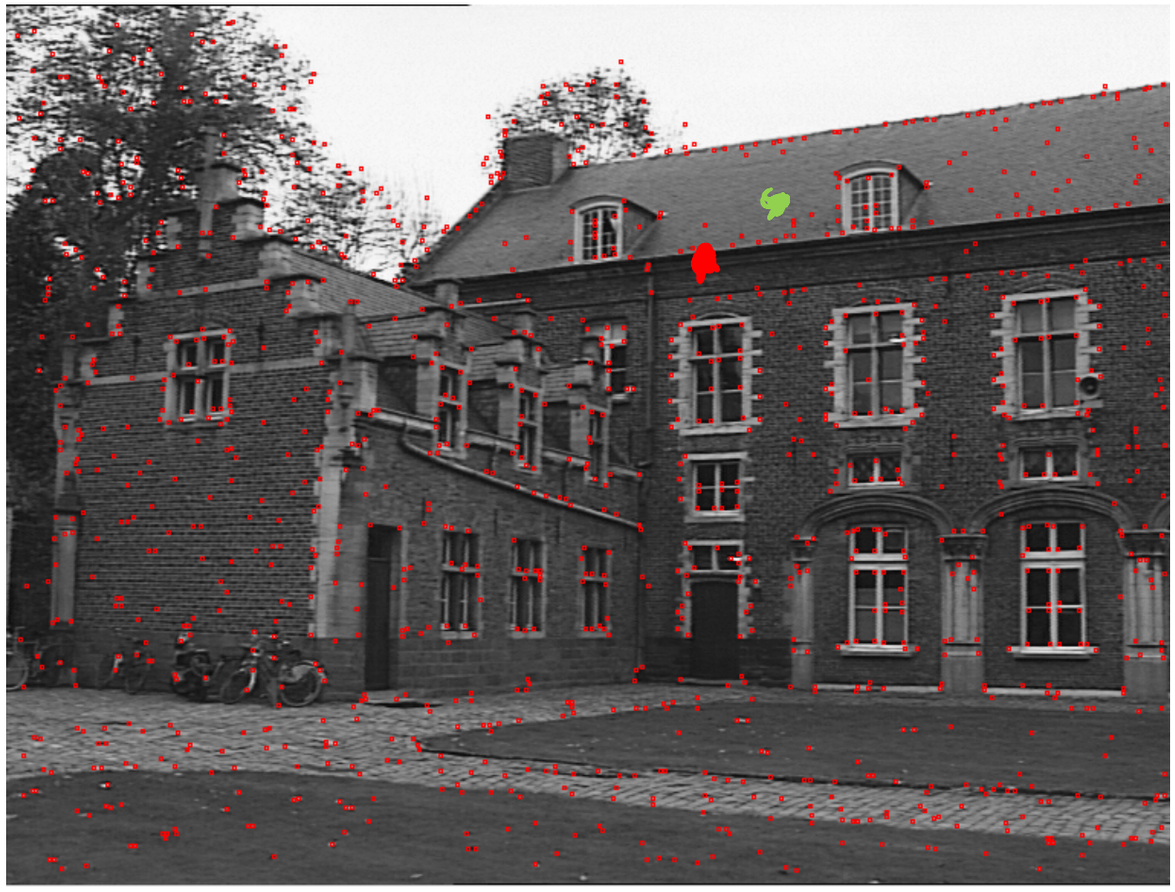
Computing $\mathbf{E}$ and $\mathbf{F}$

- Estimate 2D-2D matches between images
- Compute $\mathbf{E}$ / $\mathbf{F}$ using RANSAC:
  - Linear solver (8 points): $\mathbf{E}$ and $\mathbf{F}$
  - Minimal solver (7 points): $\mathbf{E}$ and $\mathbf{F}$
  - Calibrated solver (5 points): Only $\mathbf{E}$
  - Measure error using Sampson Error (see exercise)
- Refine $\mathbf{E}$ / $\mathbf{F}$ based on all inliers
- Search for additional matches

Computing $\mathbf{E}$ and $\mathbf{F}$

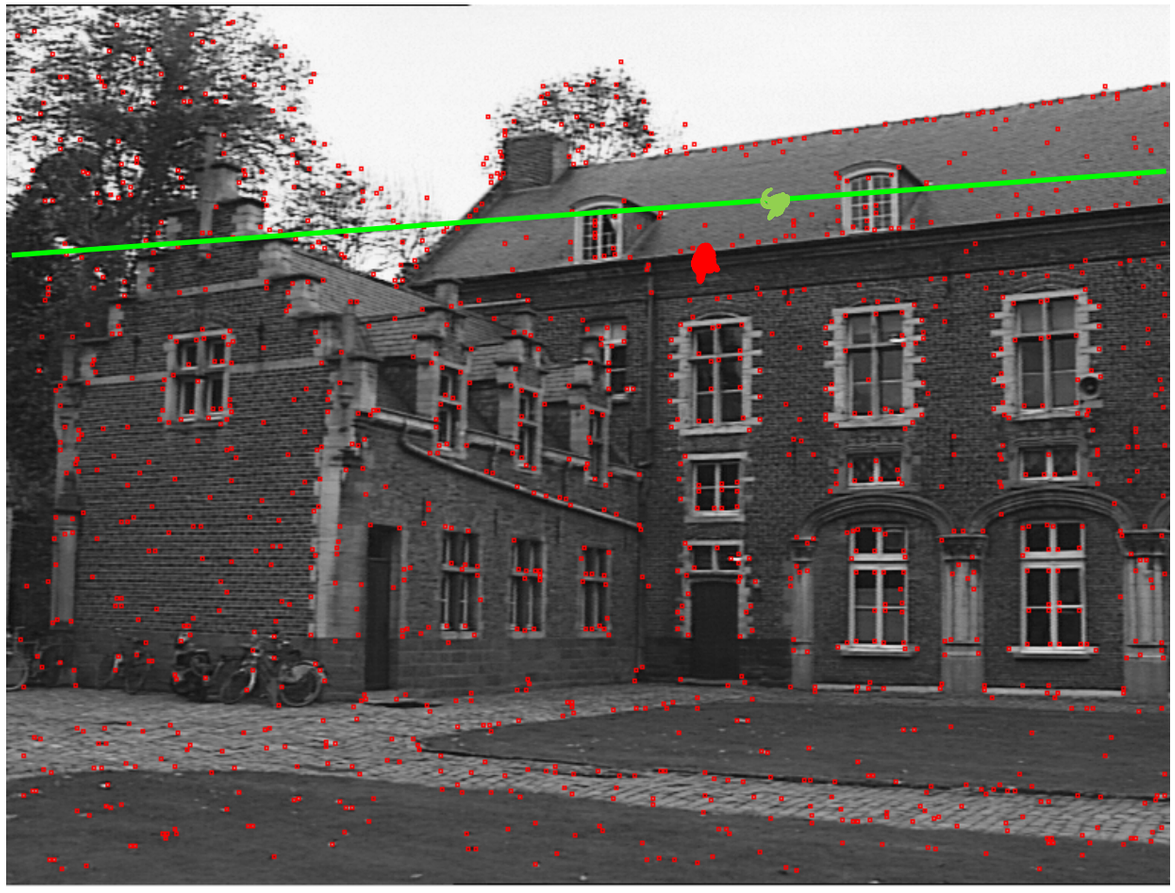
- Estimate 2D-2D matches between images
- Compute $\mathbf{E}$ / $\mathbf{F}$ using RANSAC:
  - Linear solver (8 points): $\mathbf{E}$ and $\mathbf{F}$
  - Minimal solver (7 points): $\mathbf{E}$ and $\mathbf{F}$
  - Calibrated solver (5 points): Only $\mathbf{E}$
  - Measure error using Sampson Error (see exercise)
- Refine $\mathbf{E}$ / $\mathbf{F}$ based on all inliers
- Search for additional matches
- Refine $\mathbf{E}$ / $\mathbf{F}$ using inliers and additional matches

Finding More Matches



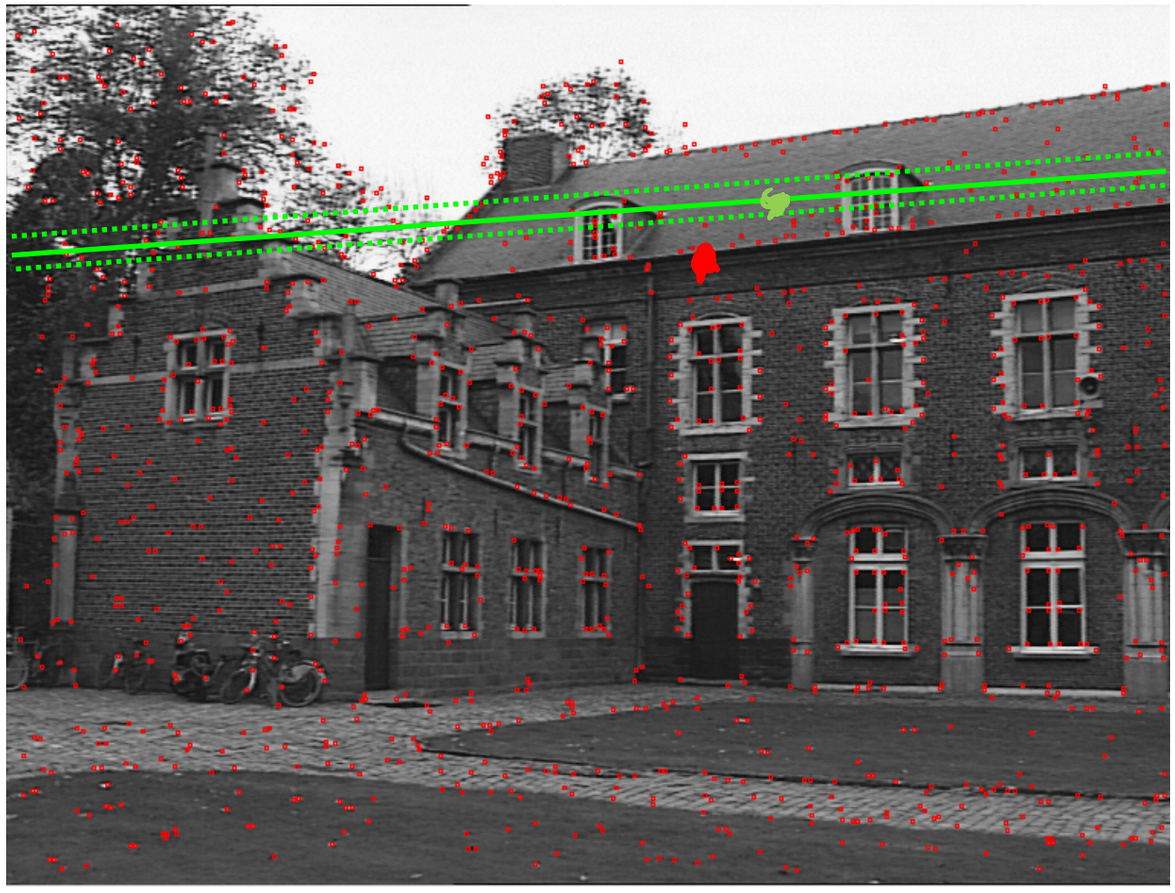
slide credit: Marc Pollefeys

Finding More Matches



slide credit: Marc Pollefeys

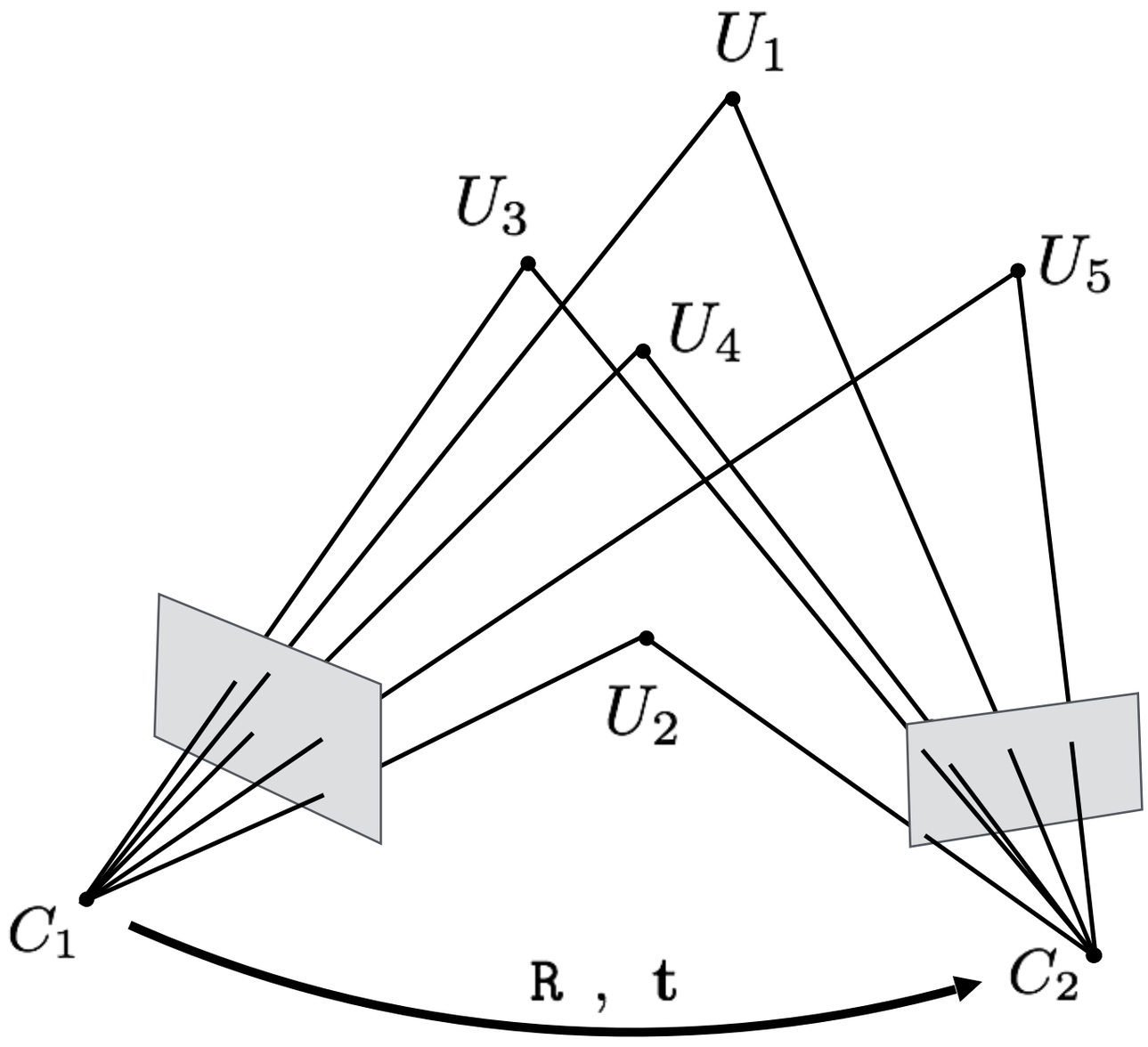
Finding More Matches



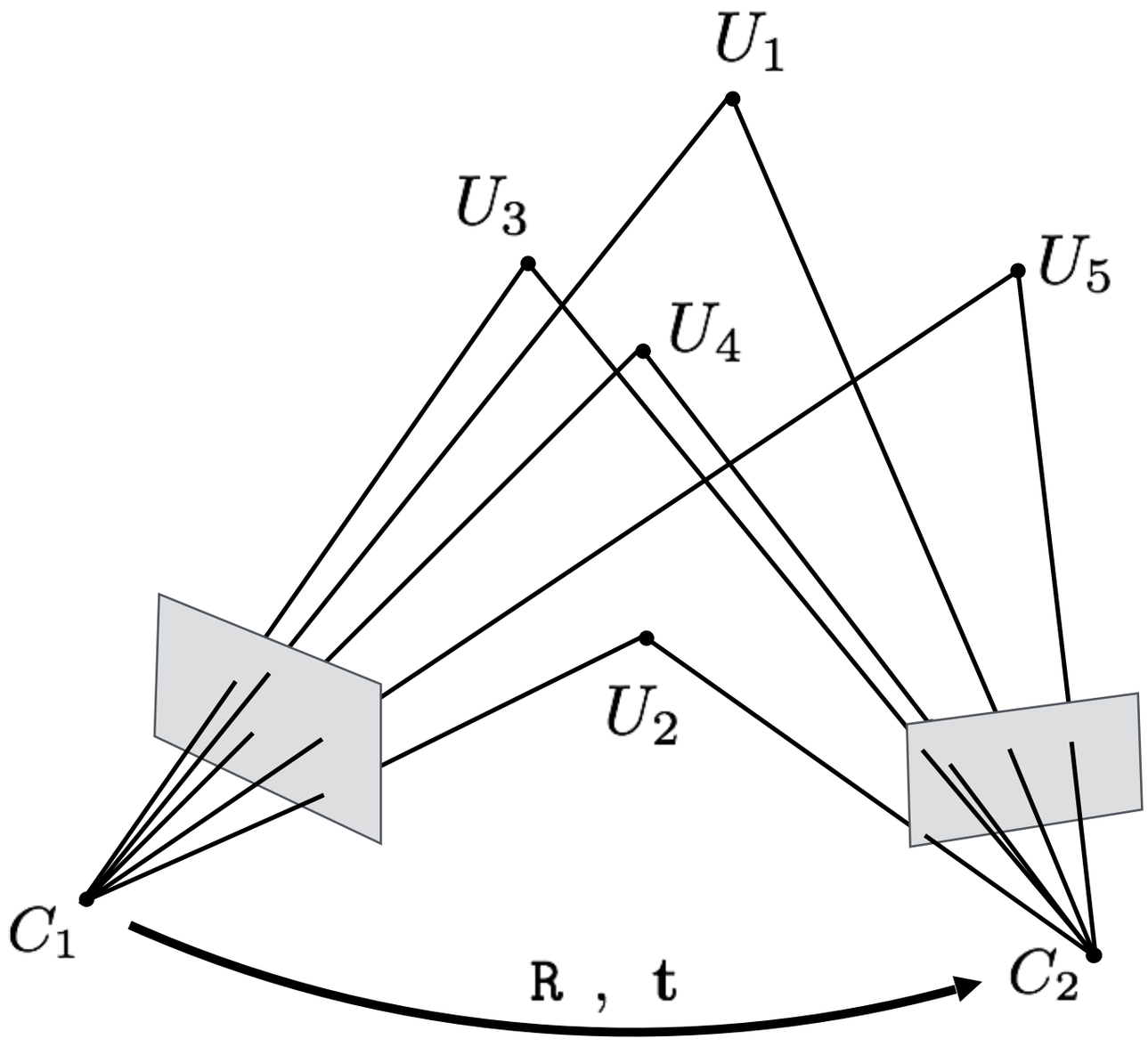
- Find matches close to epipolar line
- Same criterion used to filter outliers

slide credit: Marc Pollefeys

Relative Pose Estimation

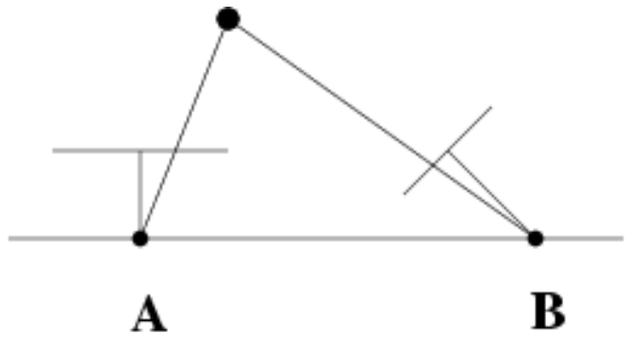


Relative Pose Estimation

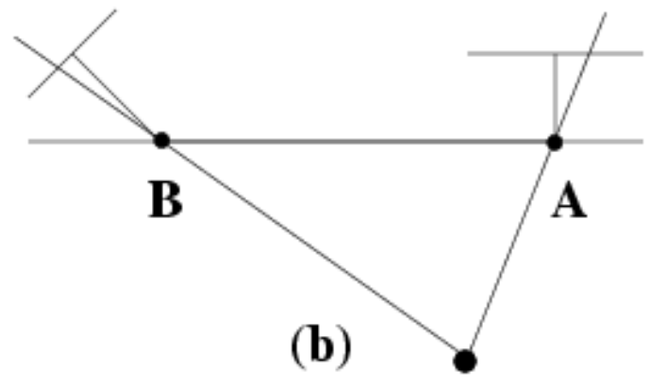


- Compute $\mathbf{E}$ / $\mathbf{F}$
- Decompose $\mathbf{E}$ / $\mathbf{F}$ to obtain rotation and translation

Relative Pose Estimation

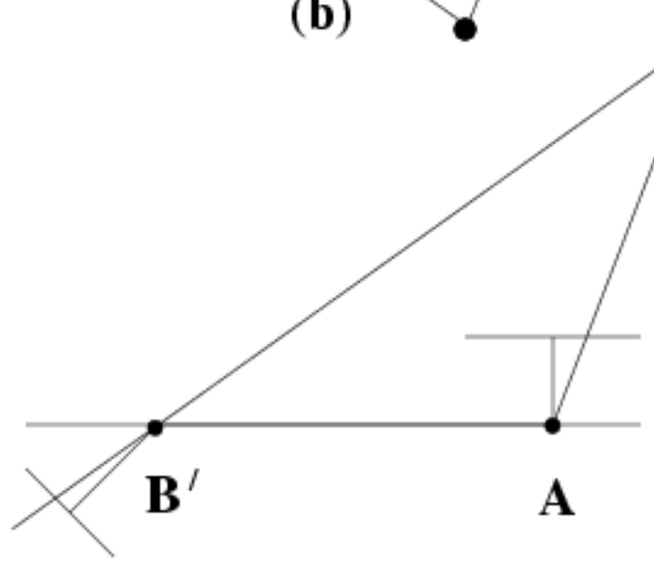


(a)



(b)

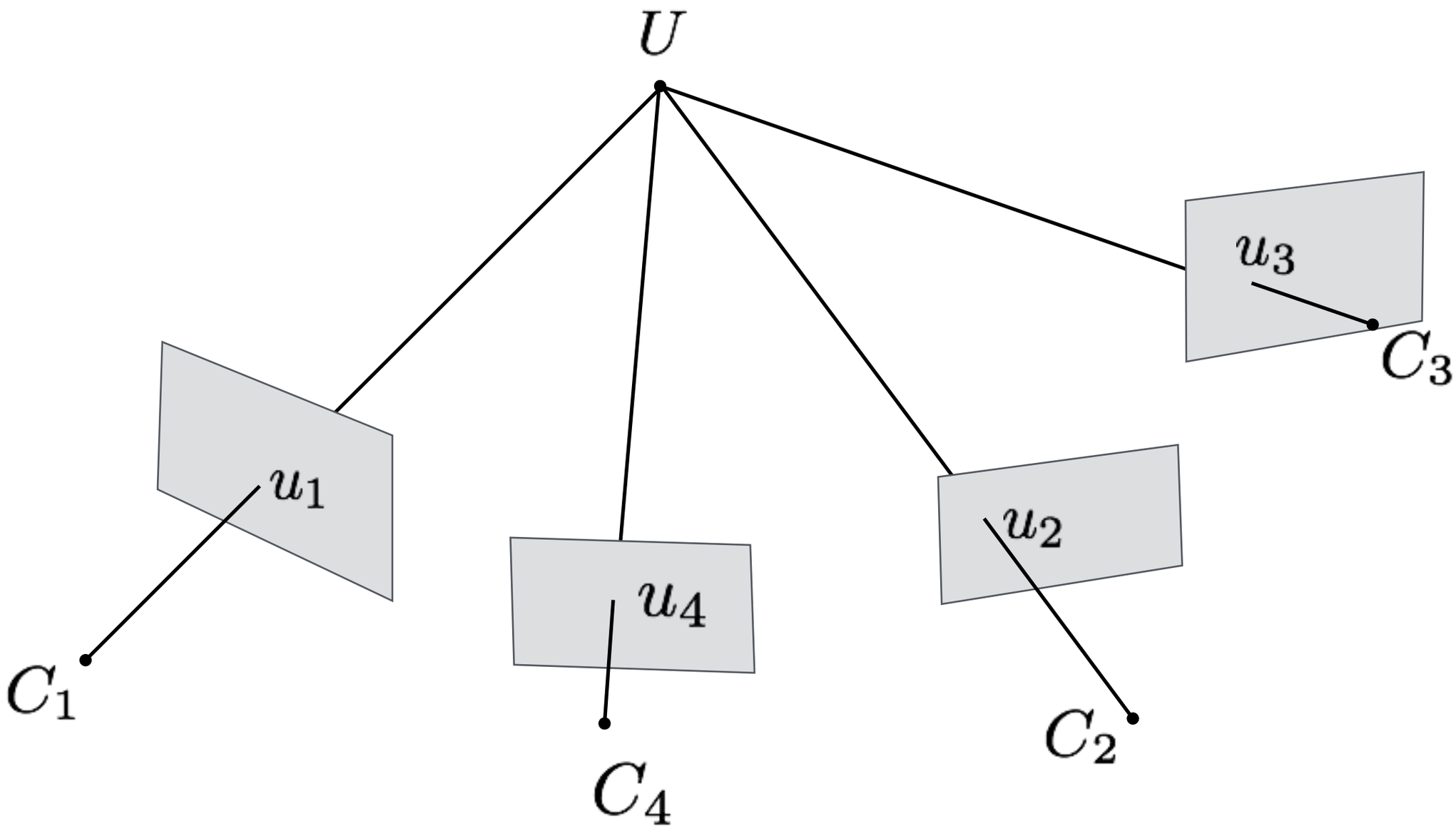
(c)



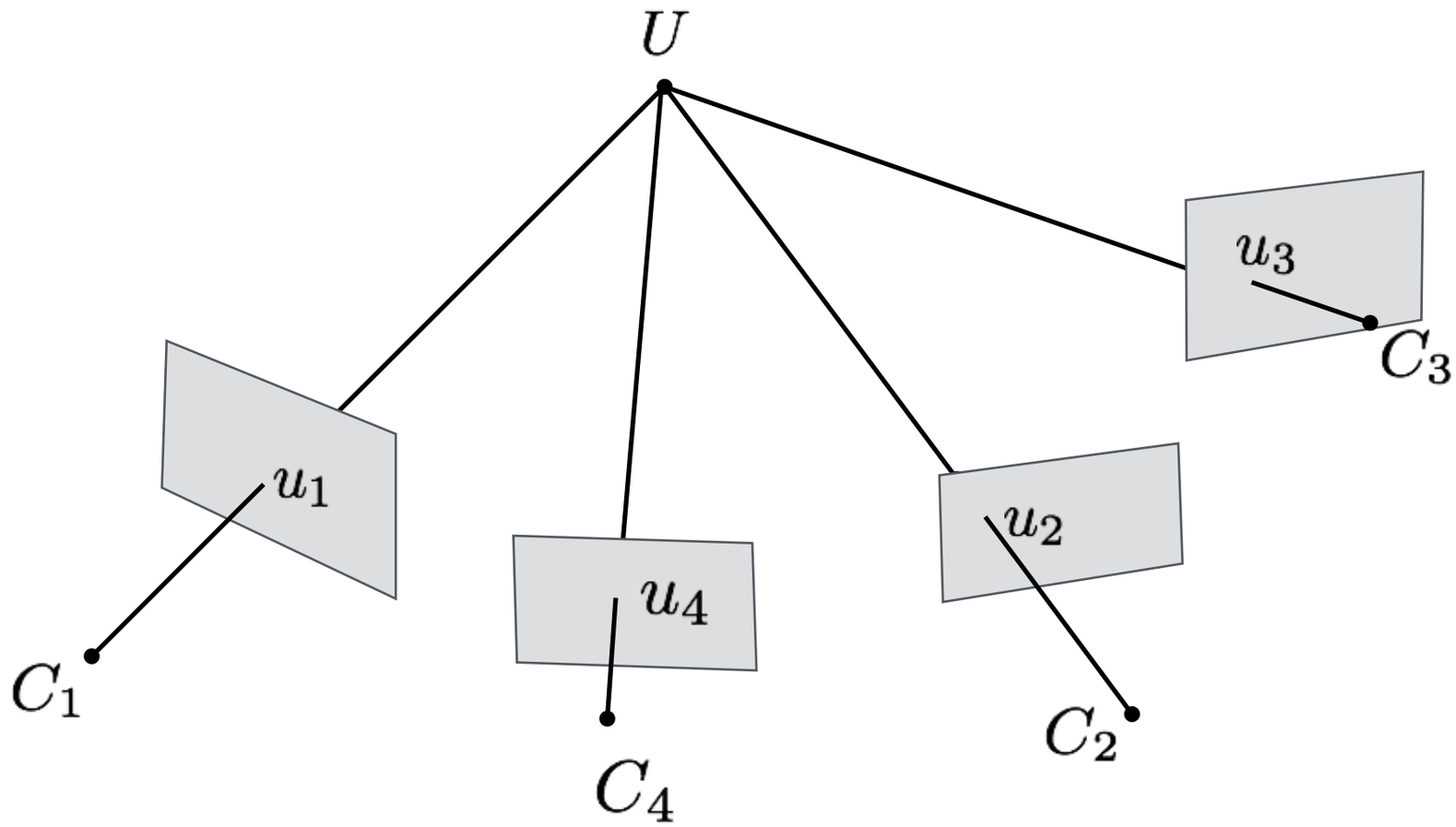
(d)

figure from Hartley and Zisserman, 2004

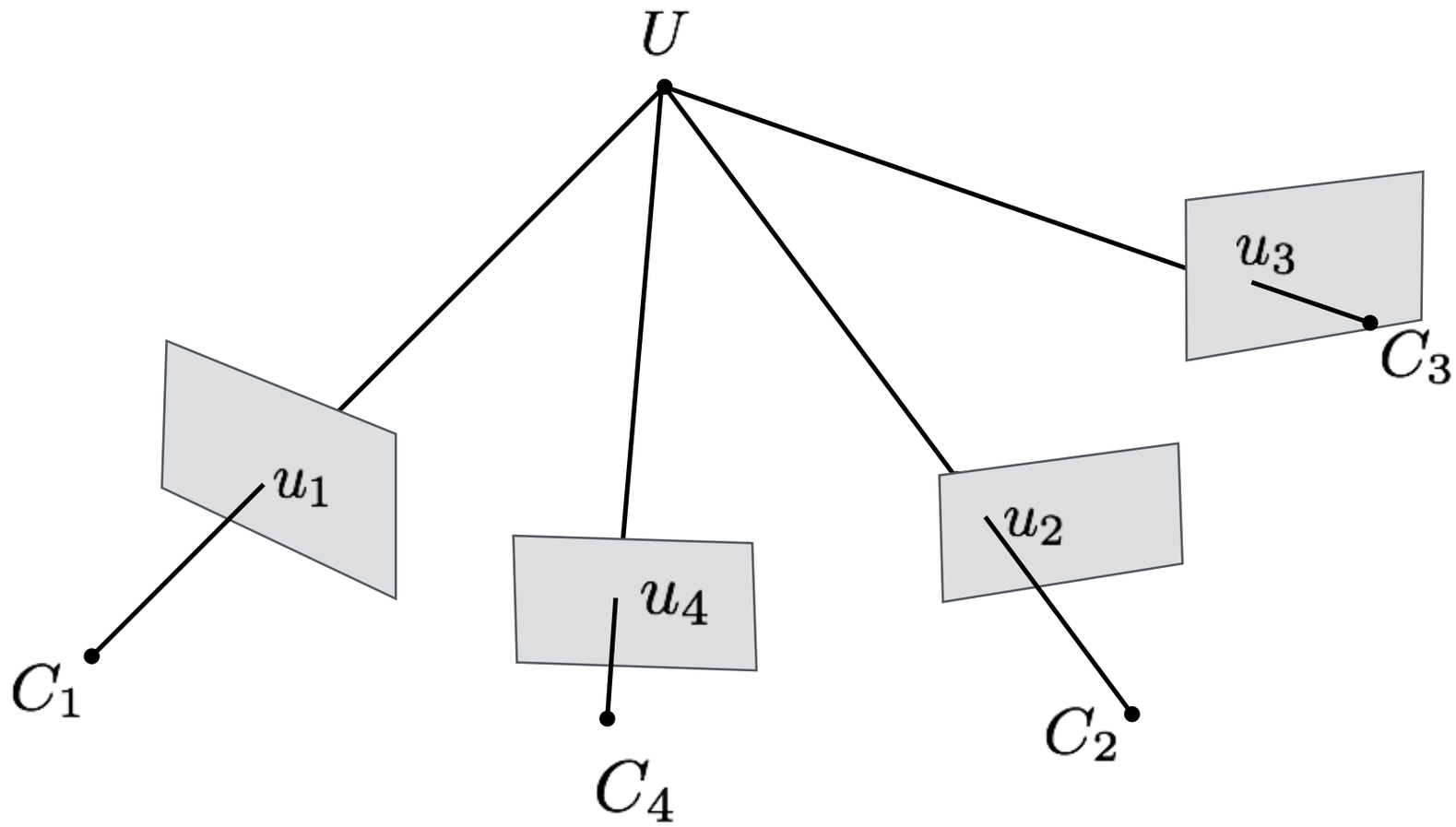
Triangulation



Triangulation using RANSAC

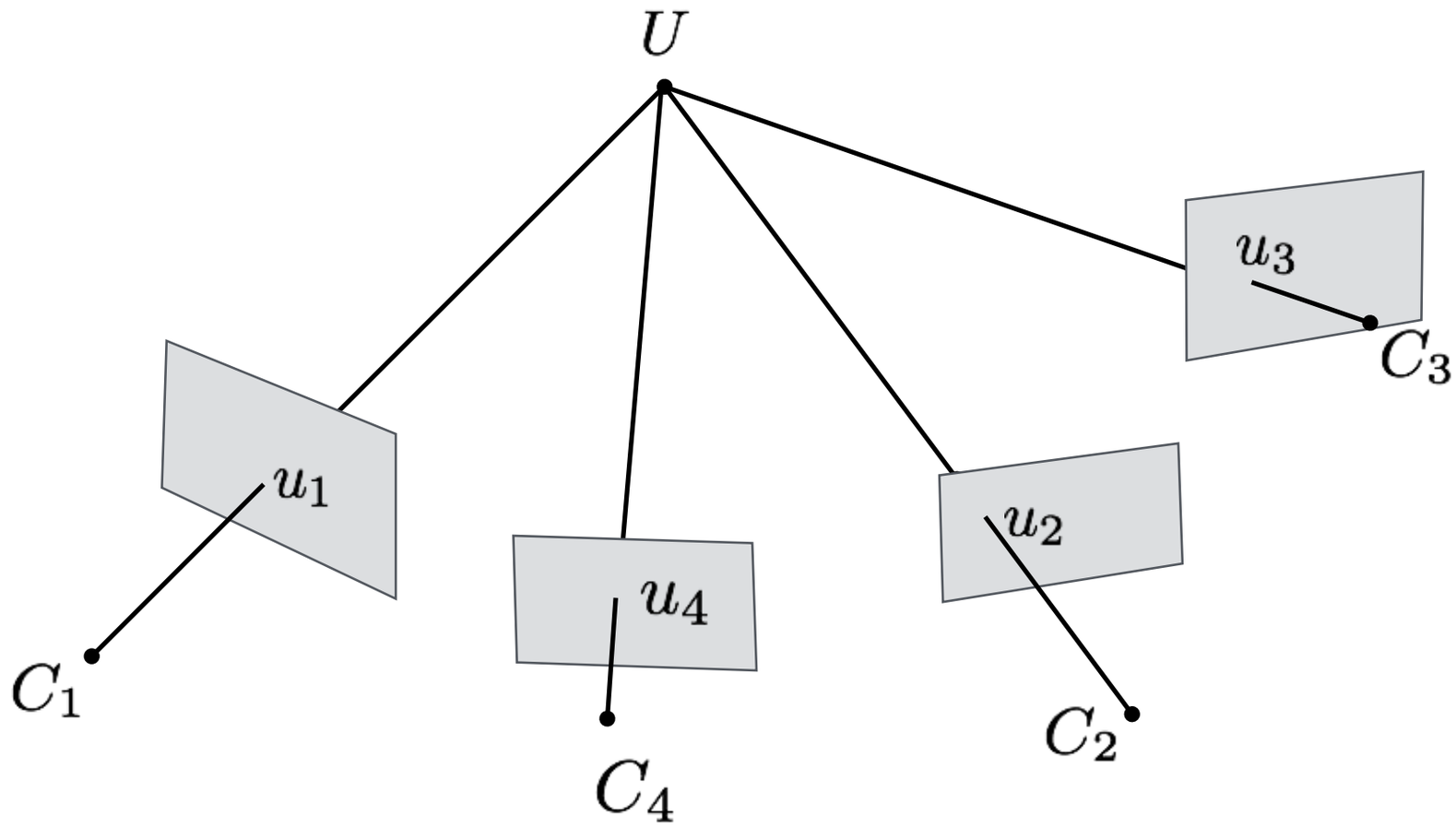


Triangulation using RANSAC



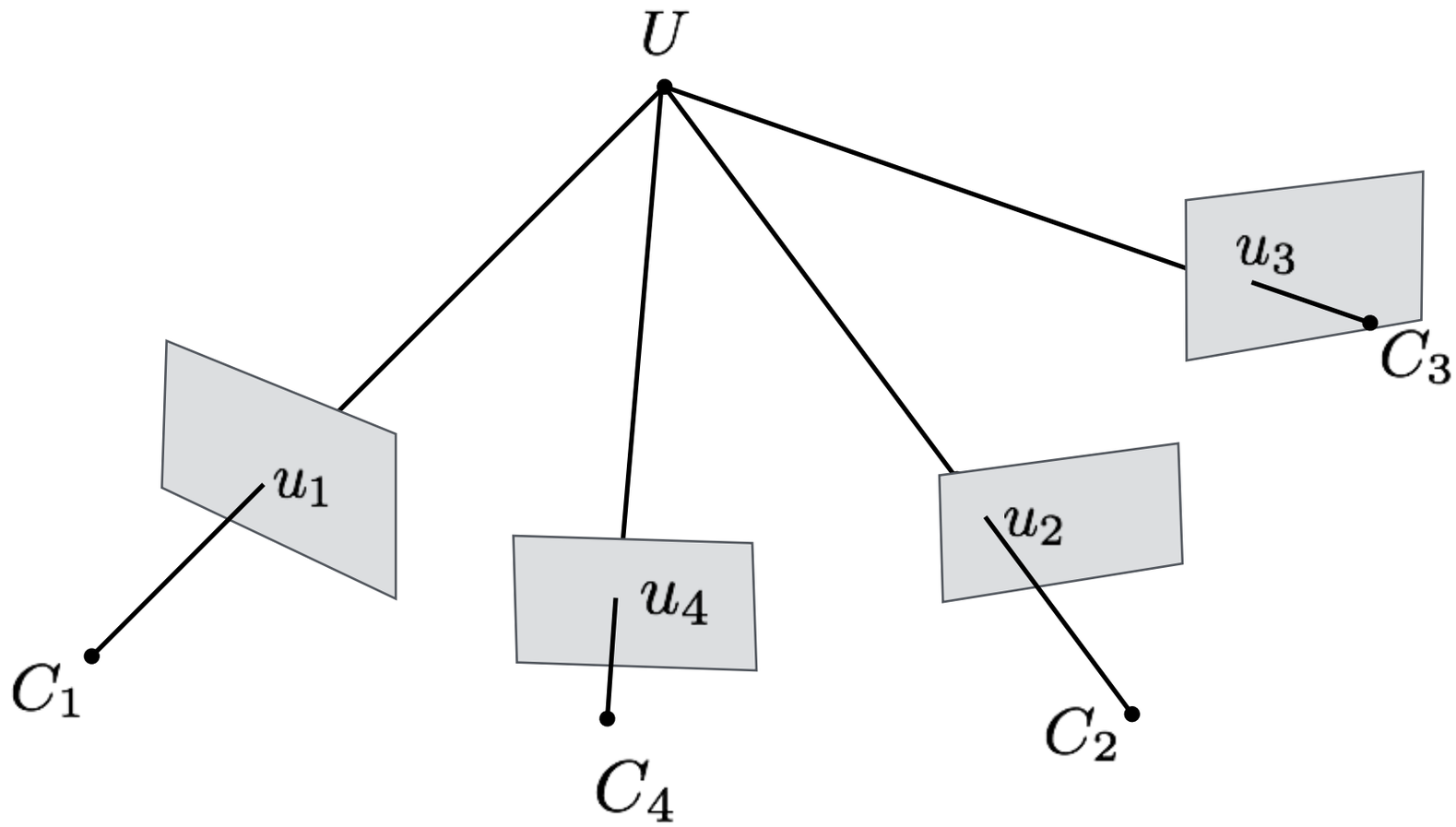
- Given: Projection matrices, track of 2D features
- Inside RANSAC loop:

Triangulation using RANSAC



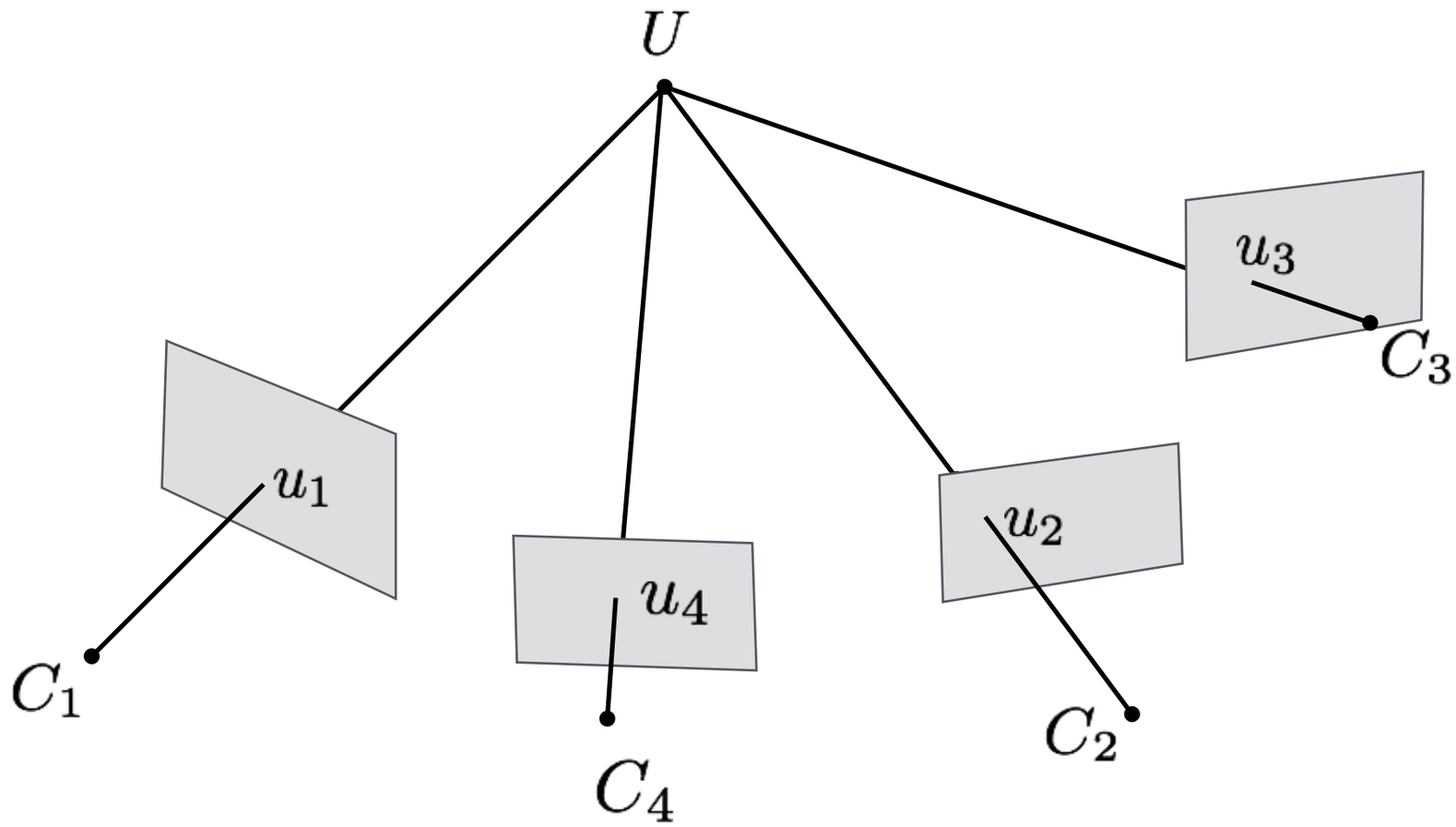
- Given: Projection matrices, track of 2D features
- Inside RANSAC loop:
  - Triangulate point using **minimal solver**

Triangulation using RANSAC



- Given: Projection matrices, track of 2D features
- Inside RANSAC loop:
  - Triangulate point using **minimal solver**
  - Determine inliers based on **reprojection error**

Triangulation using RANSAC



- Given: Projection matrices, track of 2D features
- Inside RANSAC loop:
  - Triangulate point using **minimal solver**
  - Determine inliers based on **reprojection error**
- Refine point position by minimizing sum of squared errors

Minimal Solver for Triangulation

Perspective projection in homogeneous coordinates

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

$\mathbf{X} \in \mathbb{R}^4$

$\mathbf{P} \in \mathbb{R}^{3\times 4}$

unknown

Minimal Solver for Triangulation

Perspective projection in homogeneous coordinates

$$\lambda \mathbf{x} = \mathtt{P}\mathbf{X} \quad \Leftrightarrow \quad \begin{pmatrix} \lambda x \\ \lambda y \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{pmatrix} \mathbf{X}$$

$X \in \mathbb{R}^4$

$P \in \mathbb{R}^{3\times 4}$

unknown

Minimal Solver for Triangulation

Perspective projection in homogeneous coordinates

$$\lambda \mathbf{x} = \mathtt{P}\mathbf{X} \quad \Leftrightarrow \quad \begin{pmatrix} \lambda x \\ \lambda y \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{pmatrix} \mathbf{X}$$

$X \in \mathbb{R}^4$

$P \in \mathbb{R}^{3\times 4}$

unknown

Re-arrange, insert last row into first two rows:

$$\begin{aligned} \mathbf{P}_3\mathbf{X}x &= \mathbf{P}_1\mathbf{X} \\ \mathbf{P}_3\mathbf{X}y &= \mathbf{P}_2\mathbf{X} \end{aligned}$$

Minimal Solver for Triangulation

Perspective projection in homogeneous coordinates

$$\lambda \mathbf{x} = \mathtt{P}\mathbf{X} \quad \Leftrightarrow \quad \begin{pmatrix} \lambda x \\ \lambda y \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{pmatrix} \mathbf{X}$$

$X \in \mathbb{R}^4$

$P \in \mathbb{R}^{3\times 4}$

unknown

Re-arrange, insert last row into first two rows:

$$\begin{aligned} \mathbf{P}_3\mathbf{X}x &= \mathbf{P}_1\mathbf{X} \\ \mathbf{P}_3\mathbf{X}y &= \mathbf{P}_2\mathbf{X} \end{aligned}$$

Results in two linear equations:

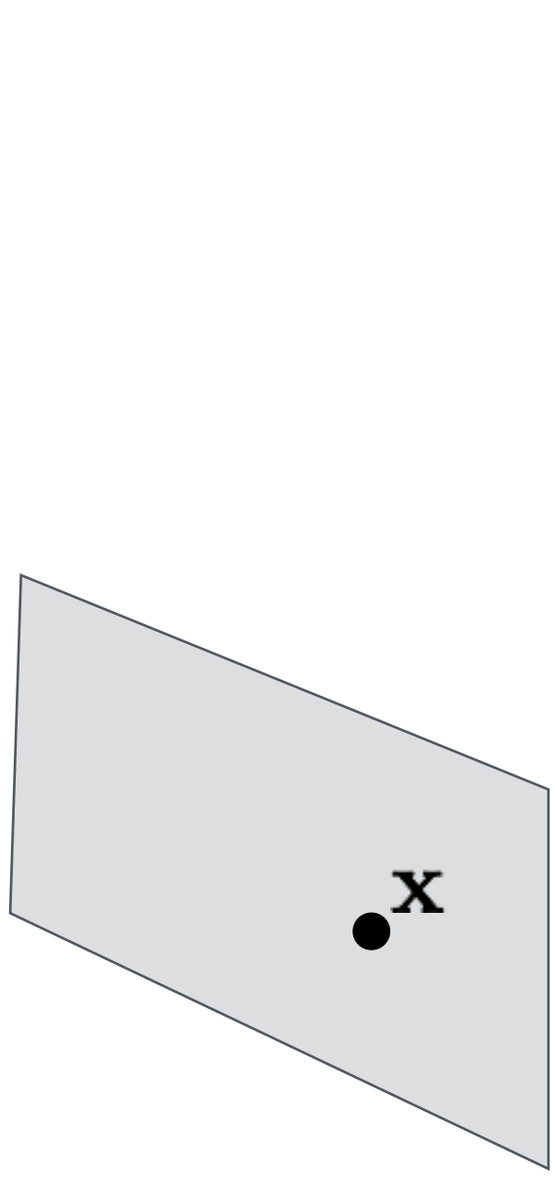
$$\begin{pmatrix} \mathbf{P}_3 x - \mathbf{P}_1 \\ \mathbf{P}_3 y - \mathbf{P}_2 \end{pmatrix} \mathbf{X} = \mathbf{0}$$

Minimal Solver for Triangulation

Need two images to solve for the 4 unknowns

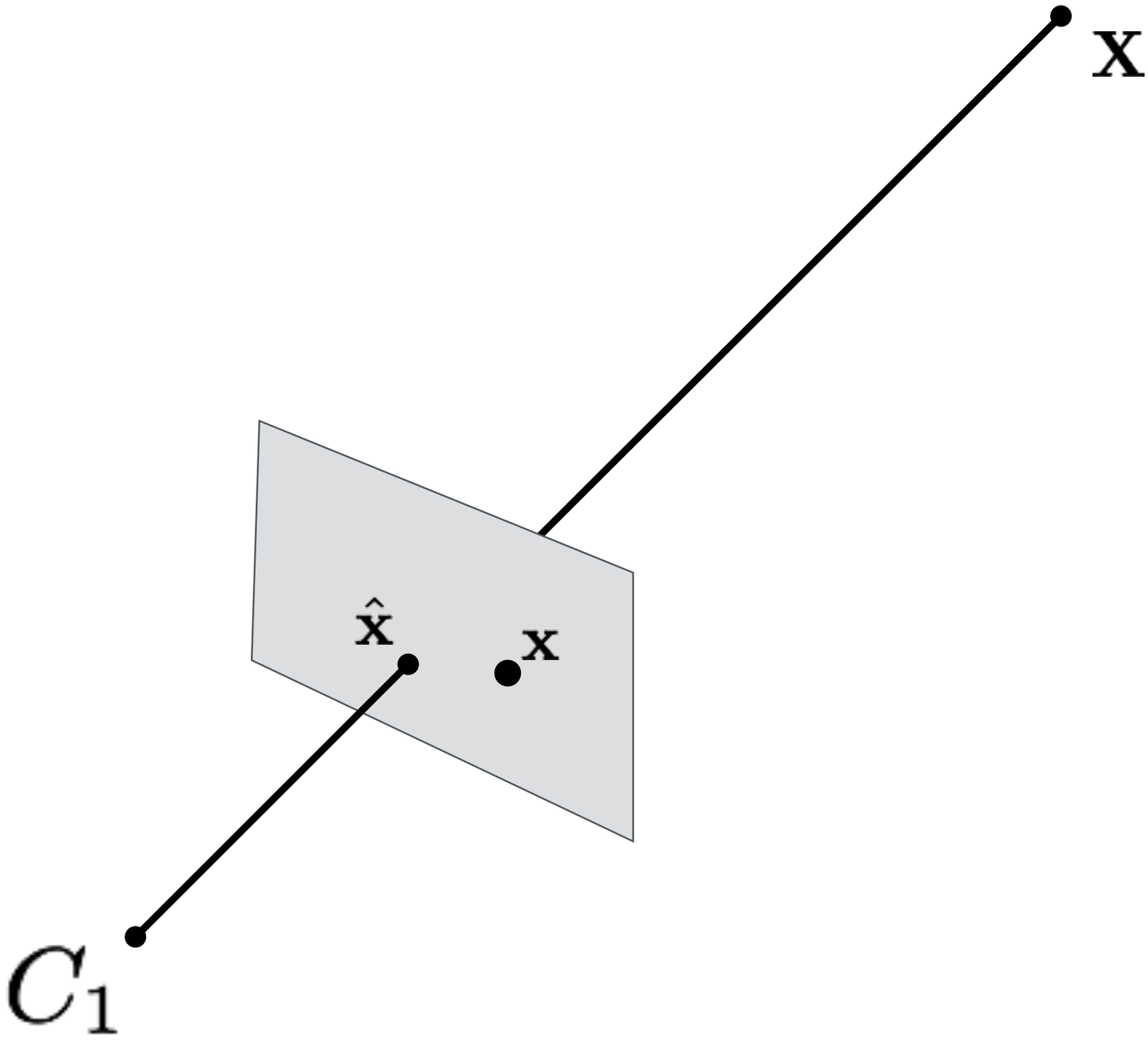
$$\begin{pmatrix} \mathbf{P}_3 x - \mathbf{P}_1 \\ \mathbf{P}_3 y - \mathbf{P}_2 \\ \mathbf{P}'_3 x - \mathbf{P}'_1 \\ \mathbf{P}'_3 y - \mathbf{P}'_2 \end{pmatrix} \mathbf{X} = \mathbf{0}$$

The Reprojection Error

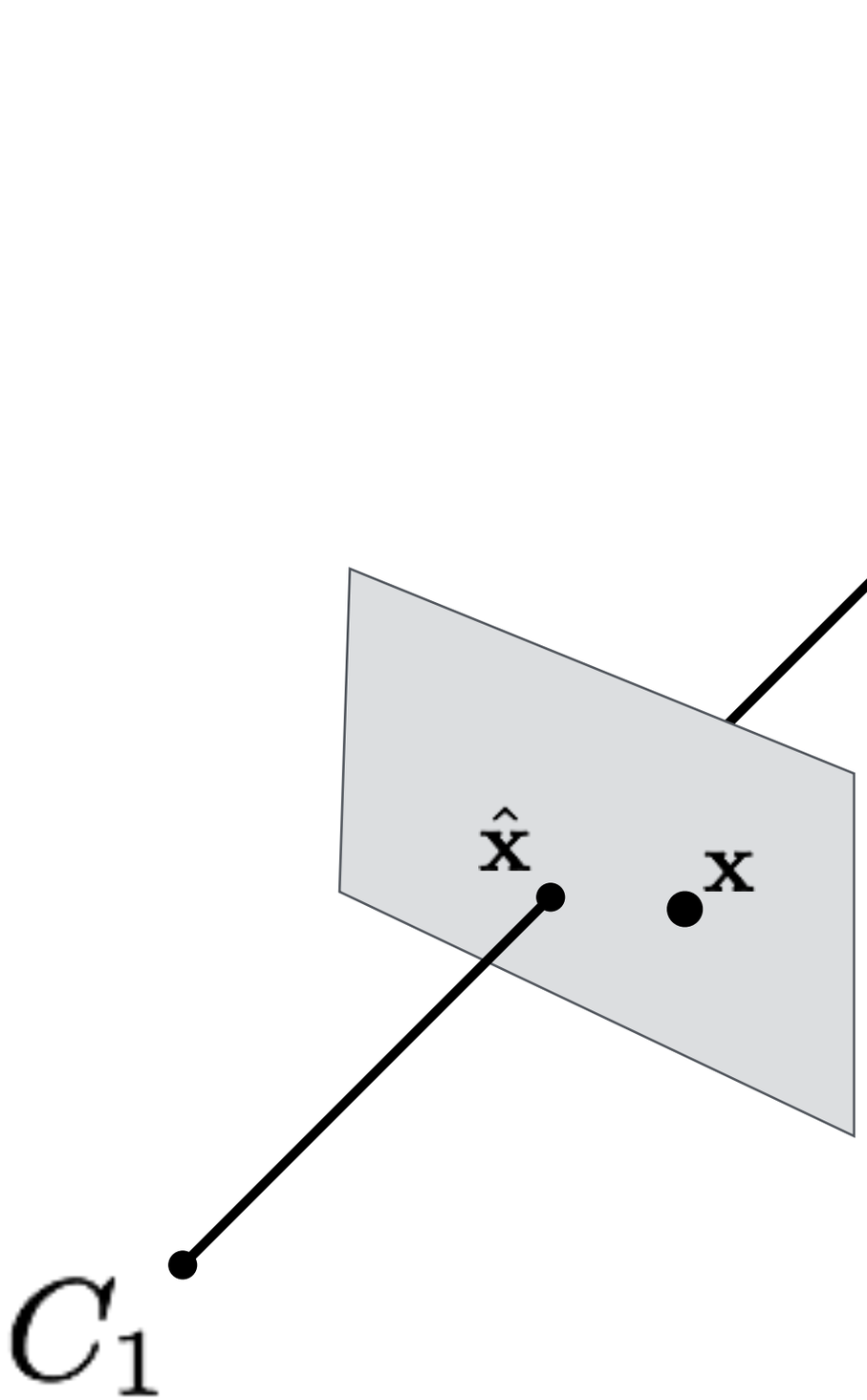


C_1

The Reprojection Error

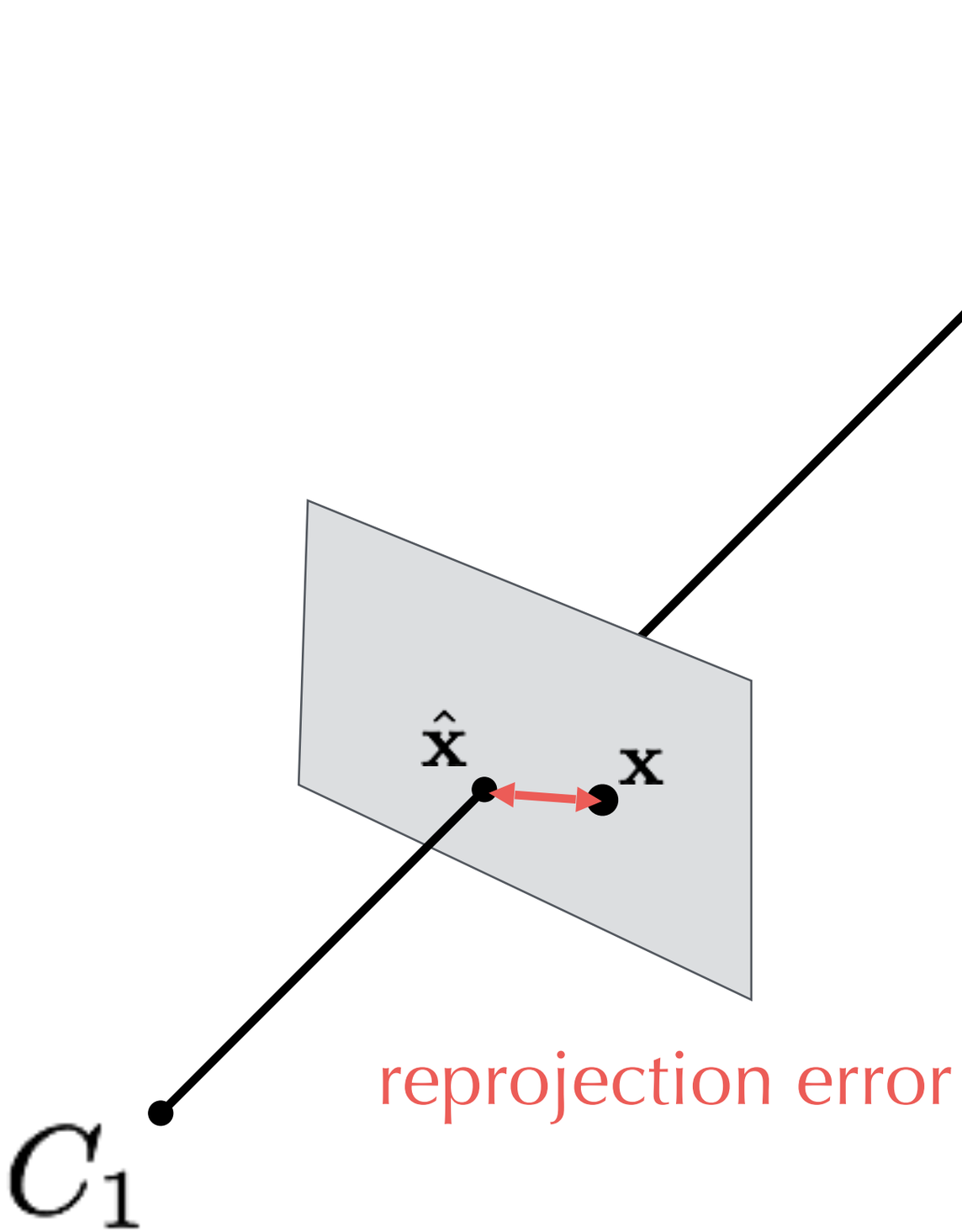


The Reprojection Error



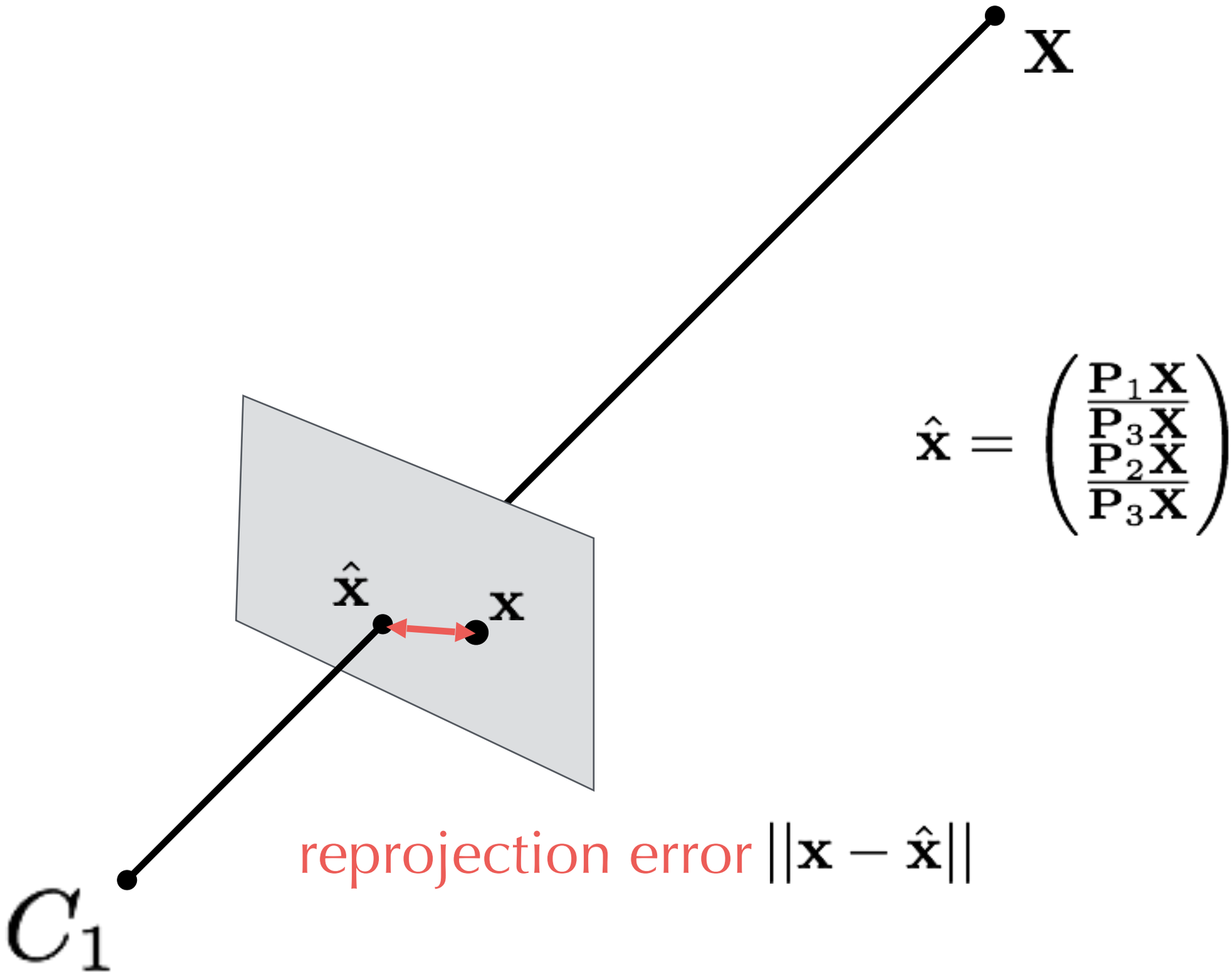
$$\hat{\mathbf{x}} = \begin{pmatrix} \frac{\mathbf{P}_1\mathbf{X}}{\mathbf{P}_3\mathbf{X}} \\ \frac{\mathbf{P}_2\mathbf{X}}{\mathbf{P}_3\mathbf{X}} \end{pmatrix}$$

The Reprojection Error



$$\hat{\mathbf{x}} = \begin{pmatrix} \frac{\mathbf{P}_1\mathbf{X}}{\mathbf{P}_3\mathbf{X}} \\ \frac{\mathbf{P}_2\mathbf{X}}{\mathbf{P}_3\mathbf{X}} \end{pmatrix}$$

The Reprojection Error



Least Squares Solution

Maximum likelihood estimate:

$$\min_{\mathbf{X}} \sum_i ||\mathbf{x}_i - \hat{\mathbf{x}}_i||^2$$

Least Squares Solution

Maximum likelihood estimate:

$$\min_{\mathbf{X}} \sum_i ||\mathbf{x}_i - \hat{\mathbf{x}}_i||^2 = \min_{\mathbf{X}} \sum_i ||\mathbf{x}_i - \begin{pmatrix} \frac{\mathbf{P}_1^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \\ \frac{\mathbf{P}_2^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \end{pmatrix} ||^2$$

Least Squares Solution

Maximum likelihood estimate:

$$\min_{\mathbf{X}} \sum_i ||\mathbf{x}_i - \hat{\mathbf{x}}_i||^2 = \min_{\mathbf{X}} \sum_i ||\mathbf{x}_i - \begin{pmatrix} \frac{\mathbf{P}_1^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \\ \frac{\mathbf{P}_2^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \end{pmatrix} ||^2$$

$$\min_{\mathbf{X}} f(\mathbf{X}) = \min_{\mathbf{X}} \sum_i \Delta_i^T \Delta_i \qquad \Delta_i = \mathbf{x}_i - \begin{pmatrix} \frac{\mathbf{P}_1^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \\ \frac{\mathbf{P}_2^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \end{pmatrix}$$

Least Squares Solution

Maximum likelihood estimate:

$$\min_{\mathbf{X}} \sum_i ||\mathbf{x}_i - \hat{\mathbf{x}}_i||^2 = \min_{\mathbf{X}} \sum_i ||\mathbf{x}_i - \begin{pmatrix} \frac{\mathbf{P}_1^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \\ \frac{\mathbf{P}_2^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \end{pmatrix} ||^2$$

$$\min_{\mathbf{X}} f(\mathbf{X}) = \min_{\mathbf{X}} \sum_i \Delta_i^T \Delta_i \qquad \Delta_i = \mathbf{x}_i - \begin{pmatrix} \frac{\mathbf{P}_1^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \\ \frac{\mathbf{P}_2^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \end{pmatrix}$$

Cost function non-linear …

Least Squares Solution

Maximum likelihood estimate:

$$\min_{\mathbf{X}} \sum_i ||\mathbf{x}_i - \hat{\mathbf{x}}_i||^2 = \min_{\mathbf{X}} \sum_i ||\mathbf{x}_i - \begin{pmatrix} \frac{\mathbf{P}_1^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \\ \frac{\mathbf{P}_2^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \end{pmatrix} ||^2$$

$$\min_{\mathbf{X}} f(\mathbf{X}) = \min_{\mathbf{X}} \sum_i \Delta_i^T \Delta_i \qquad \Delta_i = \mathbf{x}_i - \begin{pmatrix} \frac{\mathbf{P}_1^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \\ \frac{\mathbf{P}_2^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \end{pmatrix}$$

Cost function non-linear …

… but we have initial guess from RANSAC

Gradient Descent

$$\min_{\mathbf{X}} f(\mathbf{X}) = \min_{\mathbf{X}} \sum_i \Delta_i^T \Delta_i \ , \ \Delta_i = \mathbf{x}_i - \begin{pmatrix} \frac{\mathbf{P}_1^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \\ \frac{\mathbf{P}_2^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \end{pmatrix} ,$$

slide credit: Him See Lee

Gradient Descent

$$\min_{\mathbf{X}} f(\mathbf{X}) = \min_{\mathbf{X}} \sum_i \Delta_i^T \Delta_i \ , \ \Delta_i = \mathbf{x}_i - \begin{pmatrix} \frac{\mathbf{P}_1^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \\ \frac{\mathbf{P}_2^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \end{pmatrix} \ , \ \Delta = \begin{pmatrix} \Delta_1 \\ \vdots \\ \Delta_n \end{pmatrix}$$

slide credit: Him See Lee

Gradient Descent

$$\min_{\mathbf{X}} f(\mathbf{X}) = \min_{\mathbf{X}} \sum_i \Delta_i^T \Delta_i \ , \ \Delta_i = \mathbf{x}_i - \begin{pmatrix} \frac{\mathbf{P}_1^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \\ \frac{\mathbf{P}_2^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \end{pmatrix} \ , \ \Delta = \begin{pmatrix} \Delta_1 \\ \vdots \\ \Delta_n \end{pmatrix}$$

Initialization: $\mathbf{X}_k = \mathbf{X}_0$

slide credit: Him See Lee

Gradient Descent

$$\min_{\mathbf{X}} f(\mathbf{X}) = \min_{\mathbf{X}} \sum_i \Delta_i^T \Delta_i \ , \ \Delta_i = \mathbf{x}_i - \begin{pmatrix} \frac{\mathbf{P}_1^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \\ \frac{\mathbf{P}_2^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \end{pmatrix} \ , \ \Delta = \begin{pmatrix} \Delta_1 \\ \vdots \\ \Delta_n \end{pmatrix}$$

Initialization: $\mathbf{X}_k = \mathbf{X}_0$

Compute gradient: $\nabla f(\mathbf{X}_k) = \mathrm{J}^T \Delta$

slide credit: Him See Lee

Gradient Descent

$$\min_{\mathbf{X}} f(\mathbf{X}) = \min_{\mathbf{X}} \sum_i \Delta_i^T \Delta_i \ , \ \Delta_i = \mathbf{x}_i - \begin{pmatrix} \frac{\mathbf{P}_1^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \\ \frac{\mathbf{P}_2^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \end{pmatrix} \ , \ \Delta = \begin{pmatrix} \Delta_1 \\ \vdots \\ \Delta_n \end{pmatrix}$$

Initialization: $\mathbf{X}_k = \mathbf{X}_0$

Compute gradient: $\nabla f(\mathbf{X}_k) = \mathtt{J}^T \Delta$

$\mathtt{J} = \dfrac{\partial f(\mathbf{X})}{\partial \mathbf{X}}$: Jacobian

slide credit: Him See Lee

Gradient Descent

$$\min_{\mathbf{X}} f(\mathbf{X}) = \min_{\mathbf{X}} \sum_i \Delta_i^T \Delta_i \ , \ \Delta_i = \mathbf{x}_i - \begin{pmatrix} \frac{\mathbf{P}_1^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \\ \frac{\mathbf{P}_2^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \end{pmatrix} \ , \ \Delta = \begin{pmatrix} \Delta_1 \\ \vdots \\ \Delta_n \end{pmatrix}$$

Initialization: $\mathbf{X}_k = \mathbf{X}_0$

Compute gradient: $\nabla f(\mathbf{X}_k) = \mathtt{J}^T \Delta$

Update: $\mathbf{X}_{k+1} = \mathbf{X}_k - \eta \nabla f(\mathbf{X}_k)$

$\mathtt{J} = \dfrac{\partial f(\mathbf{X})}{\partial \mathbf{X}}$: Jacobian

slide credit: Him See Lee

Gradient Descent

$$\min_{\mathbf{X}} f(\mathbf{X}) = \min_{\mathbf{X}} \sum_i \Delta_i^T \Delta_i \ , \ \Delta_i = \mathbf{x}_i - \begin{pmatrix} \frac{\mathbf{P}_1^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \\ \frac{\mathbf{P}_2^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \end{pmatrix} , \Delta = \begin{pmatrix} \Delta_1 \\ \vdots \\ \Delta_n \end{pmatrix}$$

Initialization: $\mathbf{X}_k = \mathbf{X}_0$

Compute gradient: $\nabla f(\mathbf{X}_k) = \mathtt{J}^T \Delta$

Update: $\mathbf{X}_{k+1} = \mathbf{X}_k - \eta \nabla f(\mathbf{X}_k)$

$\mathtt{J} = \dfrac{\partial f(\mathbf{X})}{\partial \mathbf{X}}$: Jacobian $\quad$ η : Step size

slide credit: Him See Lee

Gradient Descent

$$\min_{\mathbf{X}} f(\mathbf{X}) = \min_{\mathbf{X}} \sum_i \Delta_i^T \Delta_i \ , \ \Delta_i = \mathbf{x}_i - \begin{pmatrix} \frac{\mathbf{P}_1^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \\ \frac{\mathbf{P}_2^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \end{pmatrix} \ , \ \Delta = \begin{pmatrix} \Delta_1 \\ \vdots \\ \Delta_n \end{pmatrix}$$

Initialization: $\mathbf{X}_k = \mathbf{X}_0$

Iterate until convergence

- Compute gradient: $\nabla f(\mathbf{X}_k) = \mathtt{J}^T \Delta$
- Update: $\mathbf{X}_{k+1} = \mathbf{X}_k - \eta \nabla f(\mathbf{X}_k)$

$\mathtt{J} = \dfrac{\partial f(\mathbf{X})}{\partial \mathbf{X}}$: Jacobian $\quad$ η : Step size

slide credit: Him See Lee

Gradient Descent

$$\min_{\mathbf{X}} f(\mathbf{X}) = \min_{\mathbf{X}} \sum_i \Delta_i^T \Delta_i \ , \ \Delta_i = \mathbf{x}_i - \begin{pmatrix} \frac{\mathbf{P}_1^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \\ \frac{\mathbf{P}_2^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \end{pmatrix} \ , \ \Delta = \begin{pmatrix} \Delta_1 \\ \vdots \\ \Delta_n \end{pmatrix}$$

Initialization: $\mathbf{X}_k = \mathbf{X}_0$

Iterate until convergence

- Compute gradient: $\nabla f(\mathbf{X}_k) = \mathtt{J}^T \Delta$
- Update: $\mathbf{X}_{k+1} = \mathbf{X}_k - \eta \nabla f(\mathbf{X}_k)$

$\mathtt{J} = \dfrac{\partial f(\mathbf{X})}{\partial \mathbf{X}}$: Jacobian $\quad$ η : Step size

Slow convergence near minimum point!

slide credit: Him See Lee

Newton's Method

2nd order approximation (quadratic Taylor expansion):

$$f(\mathbf{X}+\delta)|_{\mathbf{X}=\mathbf{X}_k} = f(\mathbf{X}) + \nabla f(\mathbf{X})^T \delta + \frac{1}{2}\delta^T \mathrm{H} \delta \Big|_{\mathbf{X}=\mathbf{X}_k}$$

slide credit: Him See Lee

Newton's Method

2nd order approximation (quadratic Taylor expansion):

$$f(\mathbf{X}+\delta)|_{\mathbf{X}=\mathbf{X}_k} = f(\mathbf{X}) + \nabla f(\mathbf{X})^T \delta + \frac{1}{2}\delta^T \mathtt{H} \delta \Big|_{\mathbf{X}=\mathbf{X}_k}$$

Hessian matrix: $\mathtt{H} = \dfrac{\partial^2 f(\mathbf{X}+\delta)}{\partial^2 \delta}\Big|_{\mathbf{X}=\mathbf{X}_k}$

slide credit: Him See Lee

Newton's Method

2nd order approximation (quadratic Taylor expansion):

$$f(\mathbf{X}+\delta)|_{\mathbf{X}=\mathbf{X}_k} = f(\mathbf{X}) + \nabla f(\mathbf{X})^T \delta + \frac{1}{2}\delta^T \mathtt{H} \delta \Big|_{\mathbf{X}=\mathbf{X}_k}$$

Hessian matrix: $\mathtt{H} = \frac{\partial^2 f(\mathbf{X}+\delta)}{\partial^2 \delta}\Big|_{\mathbf{X}=\mathbf{X}_k}$

Find δ that minimizes $f(\mathbf{X}+\delta)|_{\mathbf{X}=\mathbf{X}_k}$!

slide credit: Him See Lee

Newton's Method

Differentiate and set to 0 gives:

$$\delta = -\mathtt{H}^{-1}\nabla f(\mathbf{X}_k)$$

Update: $\mathbf{X}_{k+1} = \mathbf{X}_k + \delta$

slide credit: Him See Lee

Newton's Method

Differentiate and set to 0 gives:

$$\delta = -\mathtt{H}^{-1}\nabla f(\mathbf{X}_k)$$

Update: $\mathbf{X}_{k+1} = \mathbf{X}_k + \delta$

Computation of H is not trivial (2nd order derivatives) and optimization might get stuck at saddle point!

slide credit: Him See Lee

Gauss-Newton

Approximate Hessian matrix by dropping 2nd order terms:

$$\mathtt{H} \approx \mathtt{J}^T \mathtt{J}$$

slide credit: Him See Lee

Gauss-Newton

Approximate Hessian matrix by dropping 2nd order terms:

$$\mathtt{H} \approx \mathtt{J}^T \mathtt{J}$$

Solve normal equation:

$$\mathtt{J}^T \mathtt{J} \delta = -\mathtt{J}^t \Delta$$

slide credit: Him See Lee

Gauss-Newton

Approximate Hessian matrix by dropping 2nd order terms:

$$\mathtt{H} \approx \mathtt{J}^T \mathtt{J}$$

Solve normal equation:

$$\mathtt{J}^T \mathtt{J} \delta = -\mathtt{J}^t \Delta$$

Might get stuck and slow convergence at saddle point!

slide credit: Him See Lee

Levenberg-Marquardt

Regularized Gauss-Newton with damping factor

$$\left(\mathtt{J}^T\mathtt{J} + \lambda\mathtt{I}\right)\delta = -\mathtt{J}^t\Delta$$

slide credit: Him See Lee

Levenberg-Marquardt

Regularized Gauss-Newton with damping factor

$$\left(\mathtt{J}^T\mathtt{J} + \lambda\mathtt{I}\right)\delta = -\mathtt{J}^t\Delta$$

$\lambda \to 0$: Gauss-Newton (when convergence is rapid)
$\lambda \to \infty$: Gradient descent (when convergence is slow)

slide credit: Him See Lee

Levenberg-Marquardt

Regularized Gauss-Newton with damping factor

$$\left(\mathtt{J}^T\mathtt{J} + \lambda\mathtt{I}\right)\delta = -\mathtt{J}^t\Delta$$

$\lambda \to 0$: Gauss-Newton (when convergence is rapid)
$\lambda \to \infty$: Gradient descent (when convergence is slow)

Adapt $\mathfrak{\lambda}$ during optimization:

- Decrease $\mathfrak{\lambda}$ when function value decreases

slide credit: Him See Lee

Levenberg-Marquardt

Regularized Gauss-Newton with damping factor

$$\left(\mathtt{J}^T\mathtt{J} + \lambda\mathtt{I}\right)\delta = -\mathtt{J}^t\Delta$$

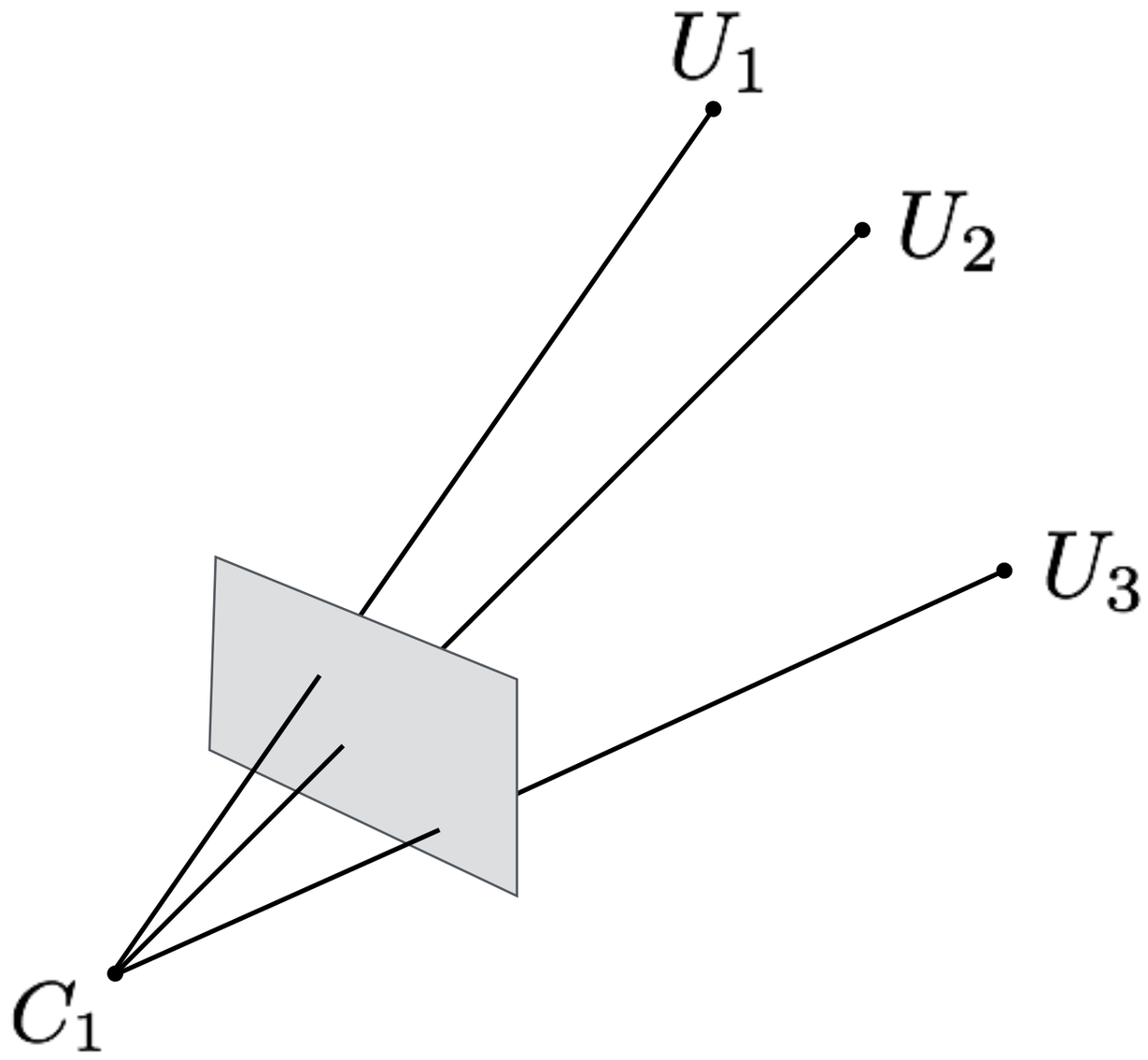
$\lambda \to 0$: Gauss-Newton (when convergence is rapid)
$\lambda \to \infty$: Gradient descent (when convergence is slow)

Adapt λ during optimization:

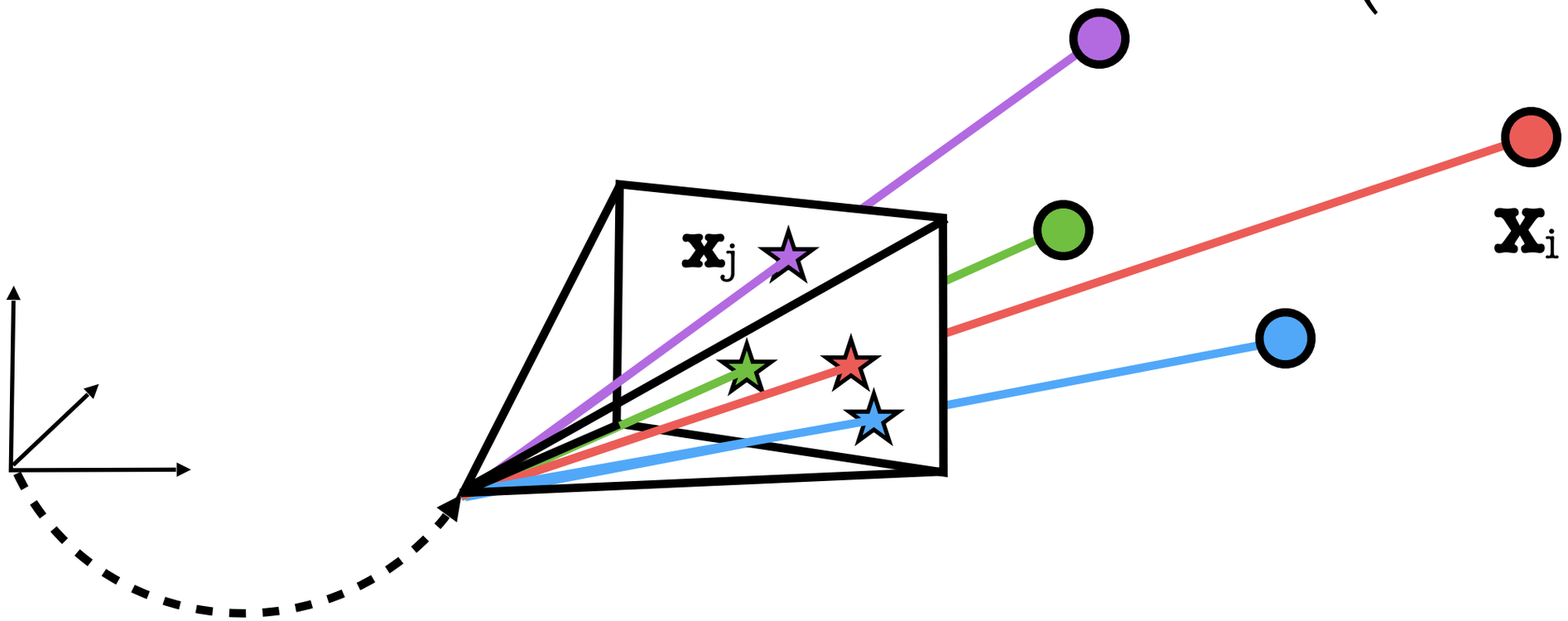
- Decrease λ when function value decreases
- Increase λ otherwise

slide credit: Him See Lee

Absolute Pose Estimation



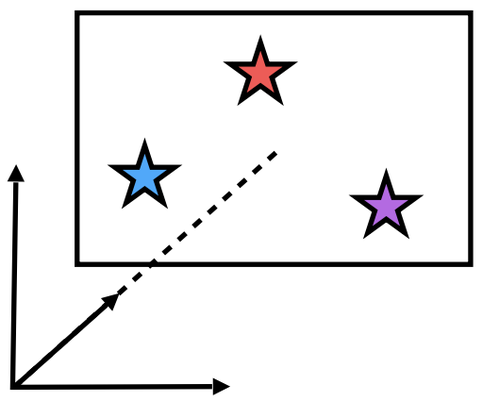
n-Point Pose Problem (PnP)



$[\mathbb{R}|\mathbb{t}]$

- Given: *n* 2D-3D correspondences ($\mathbf{x}_i$, $\mathbf{X}_i$)
- Compute pose $[\mathbb{R}|\mathbb{t}]$ s.t. $\mathbb{K}[\mathbb{R}|\mathbb{t}]\mathbf{X}_i = \alpha_i\mathbf{x}_i$, $\alpha_i>0$
- Optionally: Also estimate internal calibration matrix $\mathbb{K}$
  - In form of individual parameters
  - In form of projection matrix $\mathbb{P} = \mathbb{K}[\mathbb{R}|\mathbb{t}]$

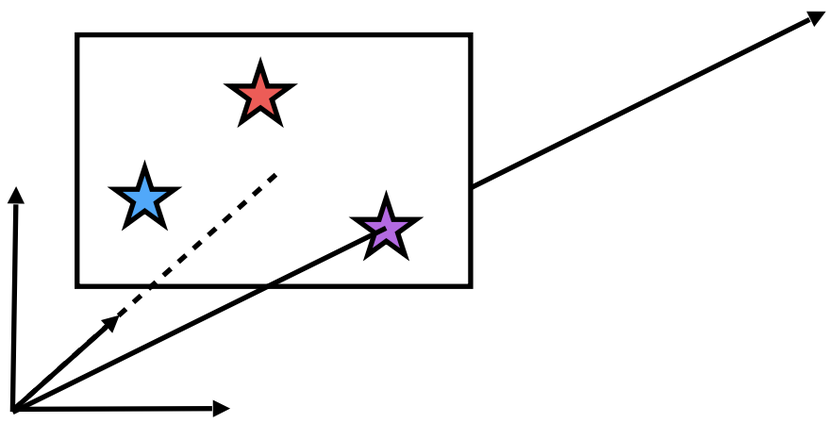
3-Point Pose Problem (P3P)



camera coordinates

- **Case**: Intrinsic calibration known [Haralick et al., ICVJ'94]

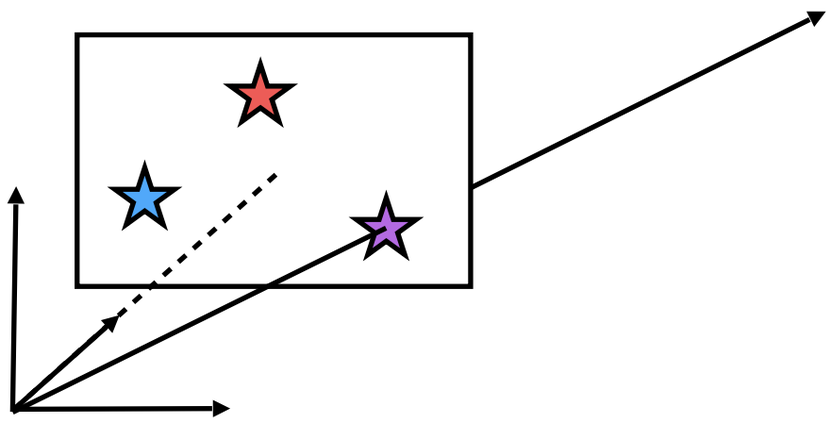
3-Point Pose Problem (P3P)



camera coordinates

- **Case**: Intrinsic calibration known [Haralick et al., ICVJ'94]

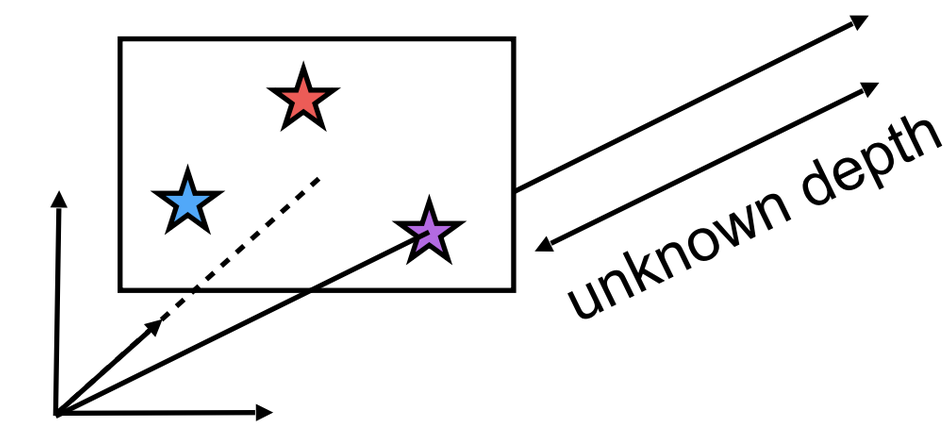
3-Point Pose Problem (P3P)



camera coordinates

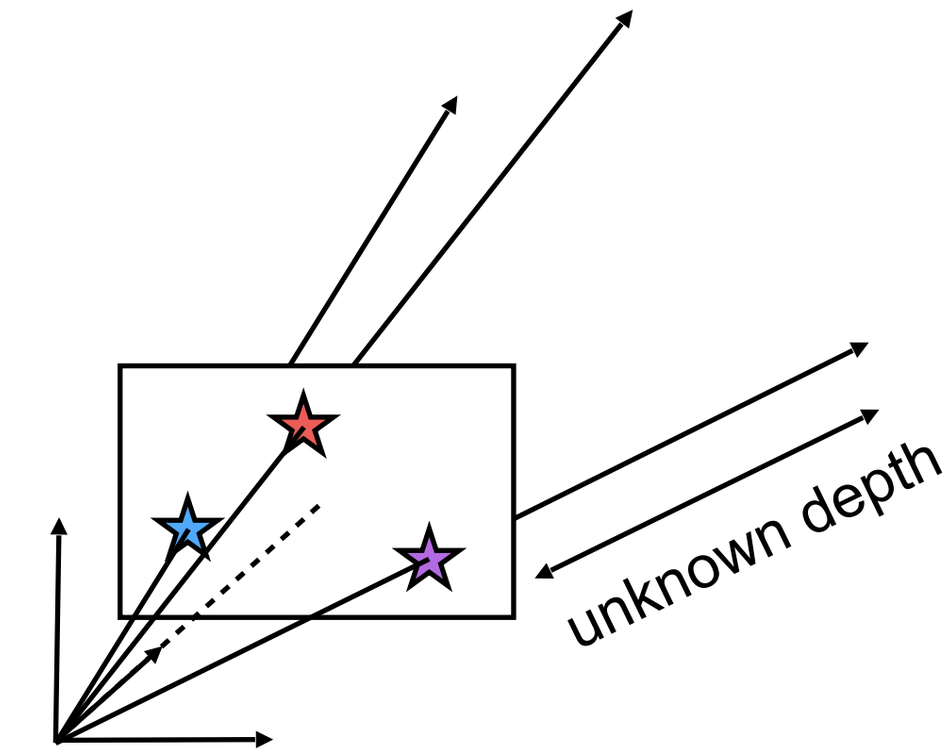
- **Case**: Intrinsic calibration known [Haralick et al., ICVJ'94]

3-Point Pose Problem (P3P)



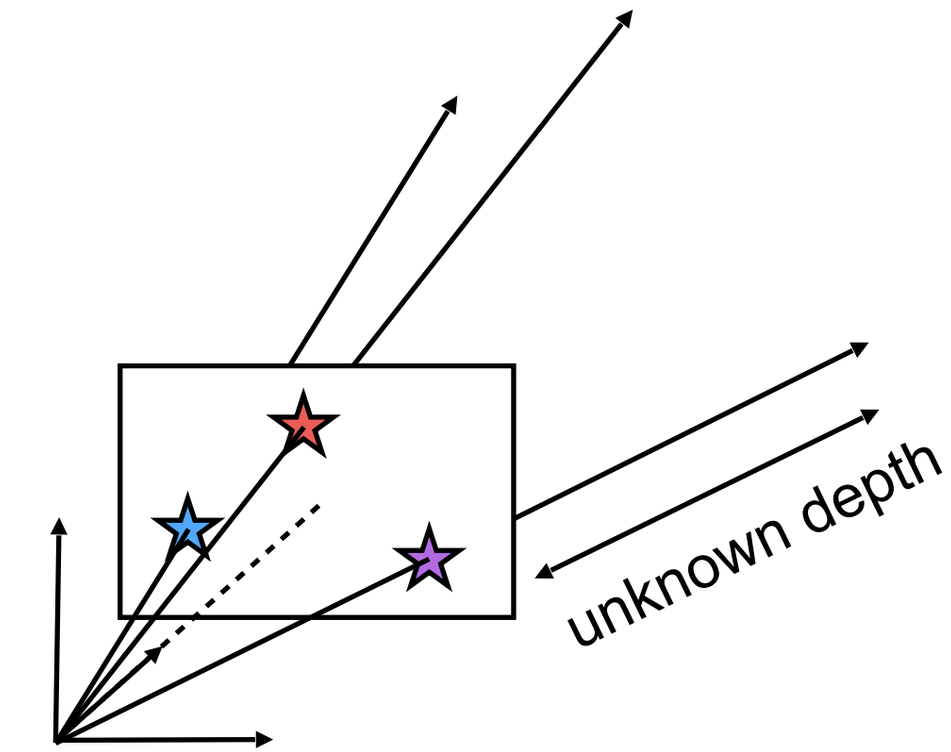
- **Case**: Intrinsic calibration known [Haralick et al., ICVJ'94]

3-Point Pose Problem (P3P)

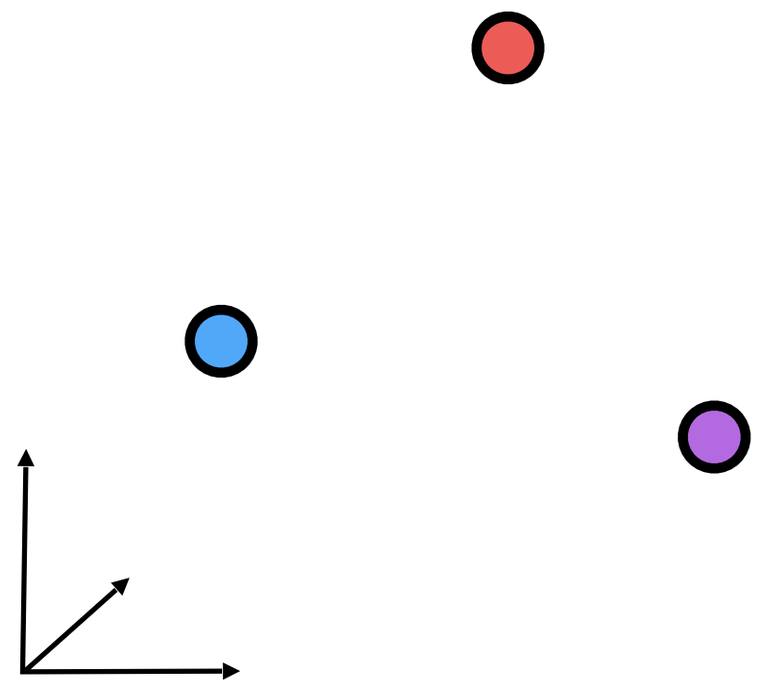


- **Case**: Intrinsic calibration known [Haralick et al., ICVJ'94]

3-Point Pose Problem (P3P)



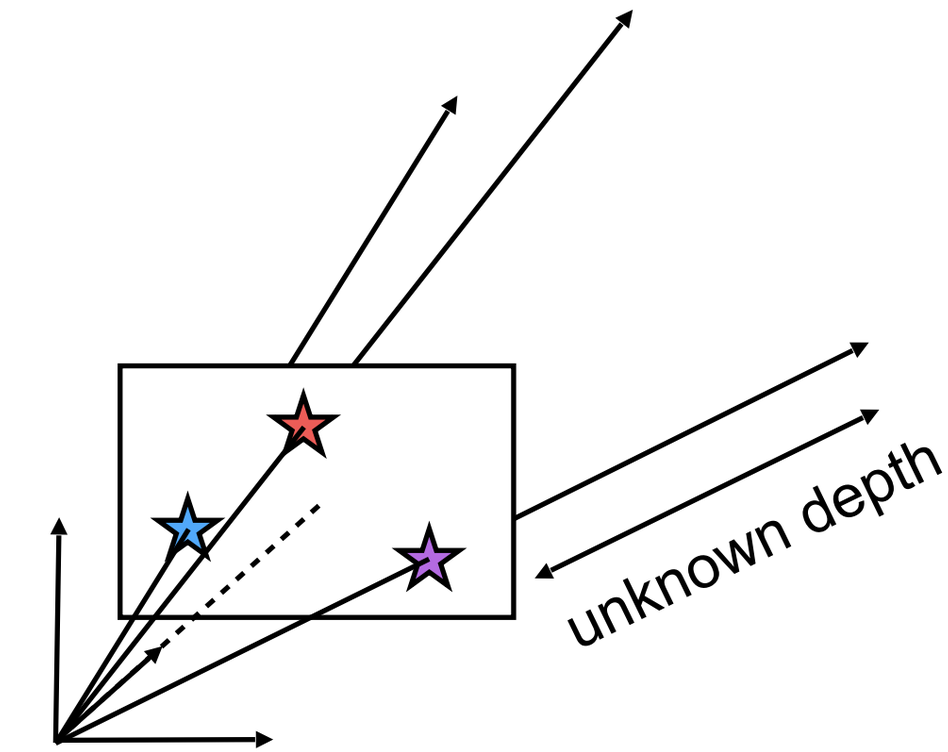
camera coordinates

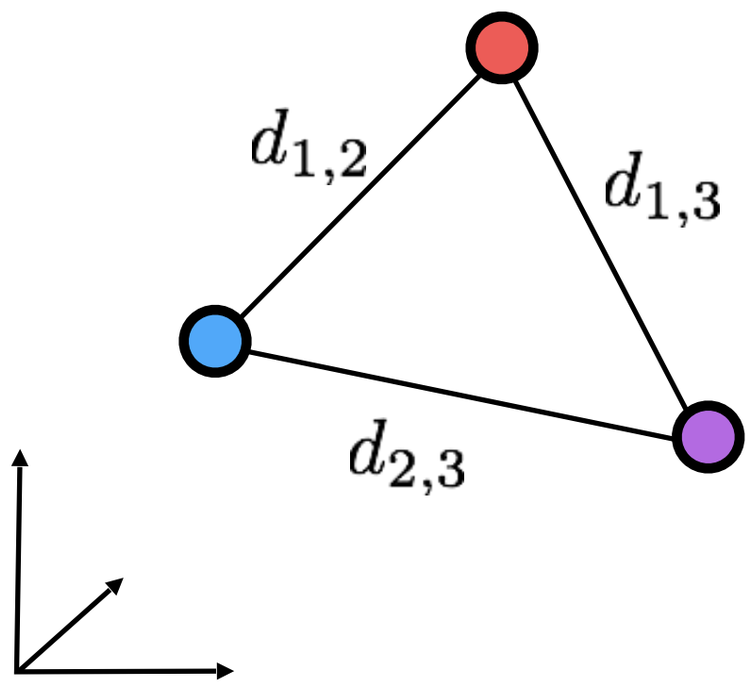


world coordinates

- **Case**: Intrinsic calibration known [Haralick et al., ICVJ'94]

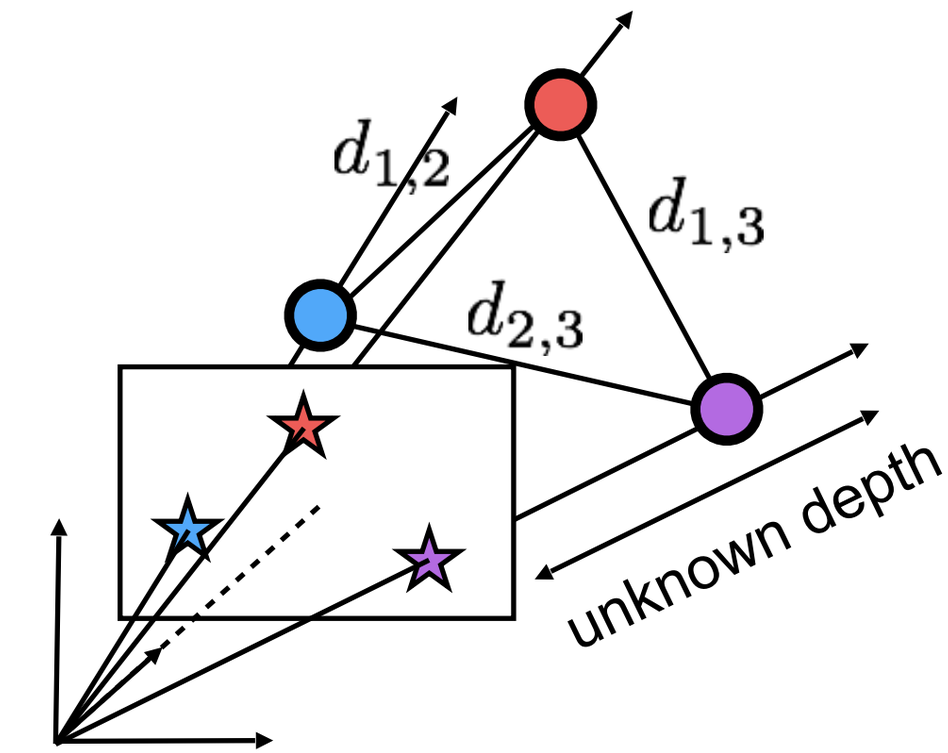
3-Point Pose Problem (P3P)



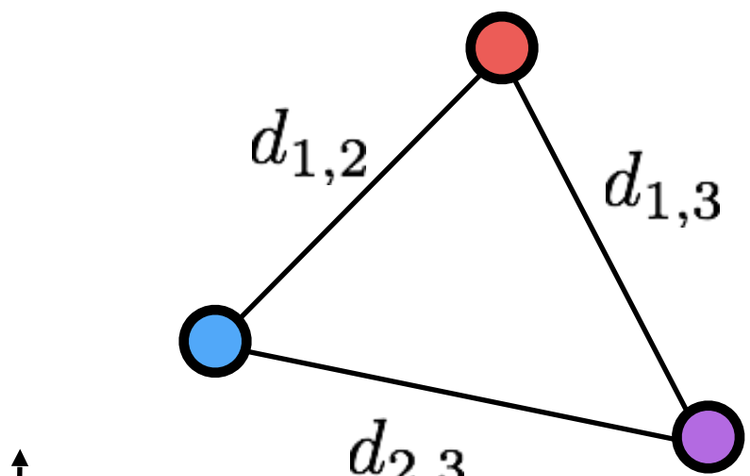


- **Case**: Intrinsic calibration known [Haralick et al., ICVJ'94]

3-Point Pose Problem (P3P)



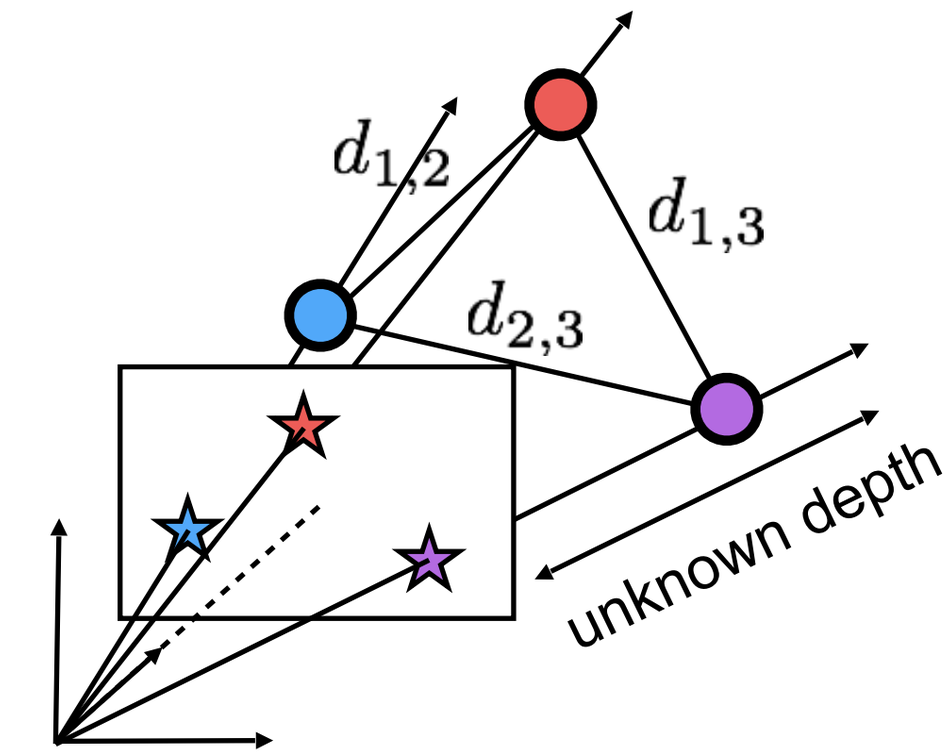
camera coordinates



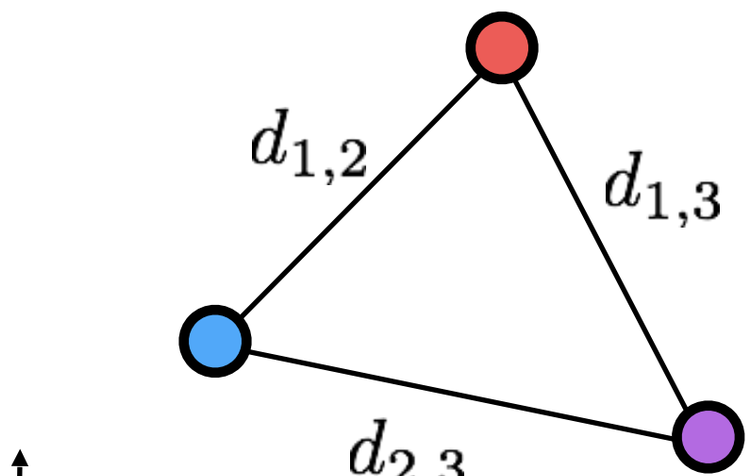
world coordinates

- **Case**: Intrinsic calibration known [Haralick et al., ICVJ'94]

3-Point Pose Problem (P3P)



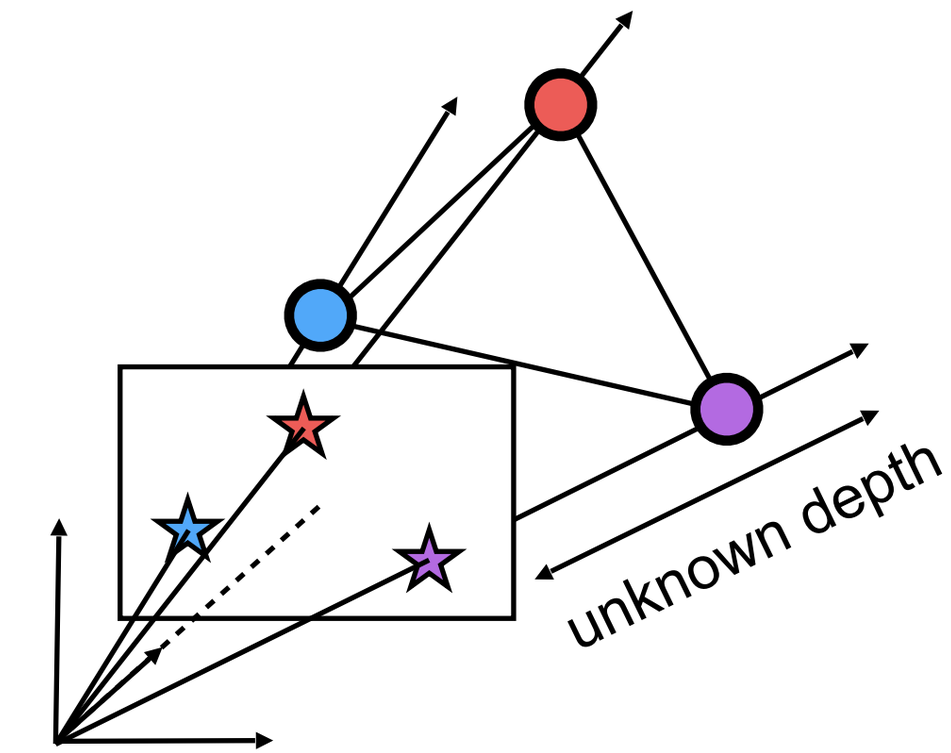
camera coordinates



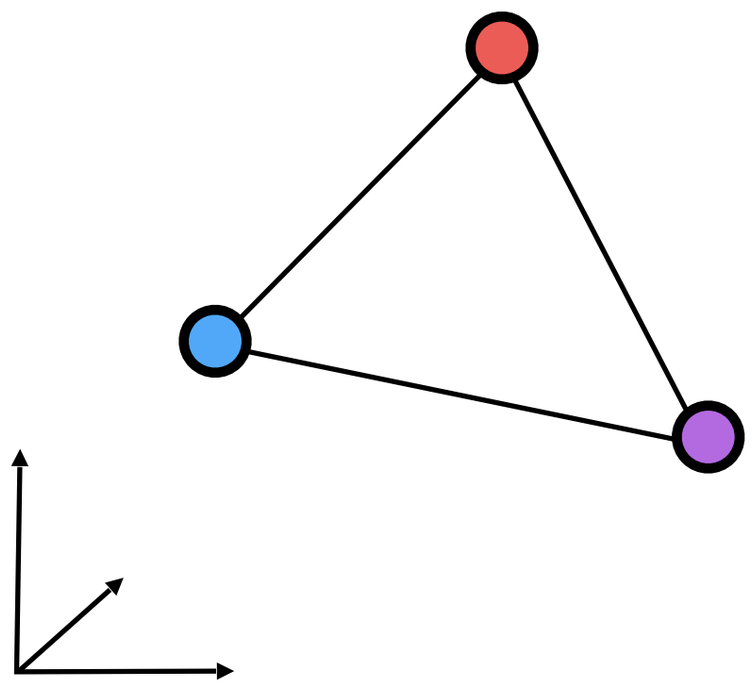
world coordinates

- **Case**: Intrinsic calibration known [Haralick et al., ICVJ'94]
- Recover depths: Solve 4th degree univariate polynomial [Fischler, Bolles, CACM'91]

3-Point Pose Problem (P3P)



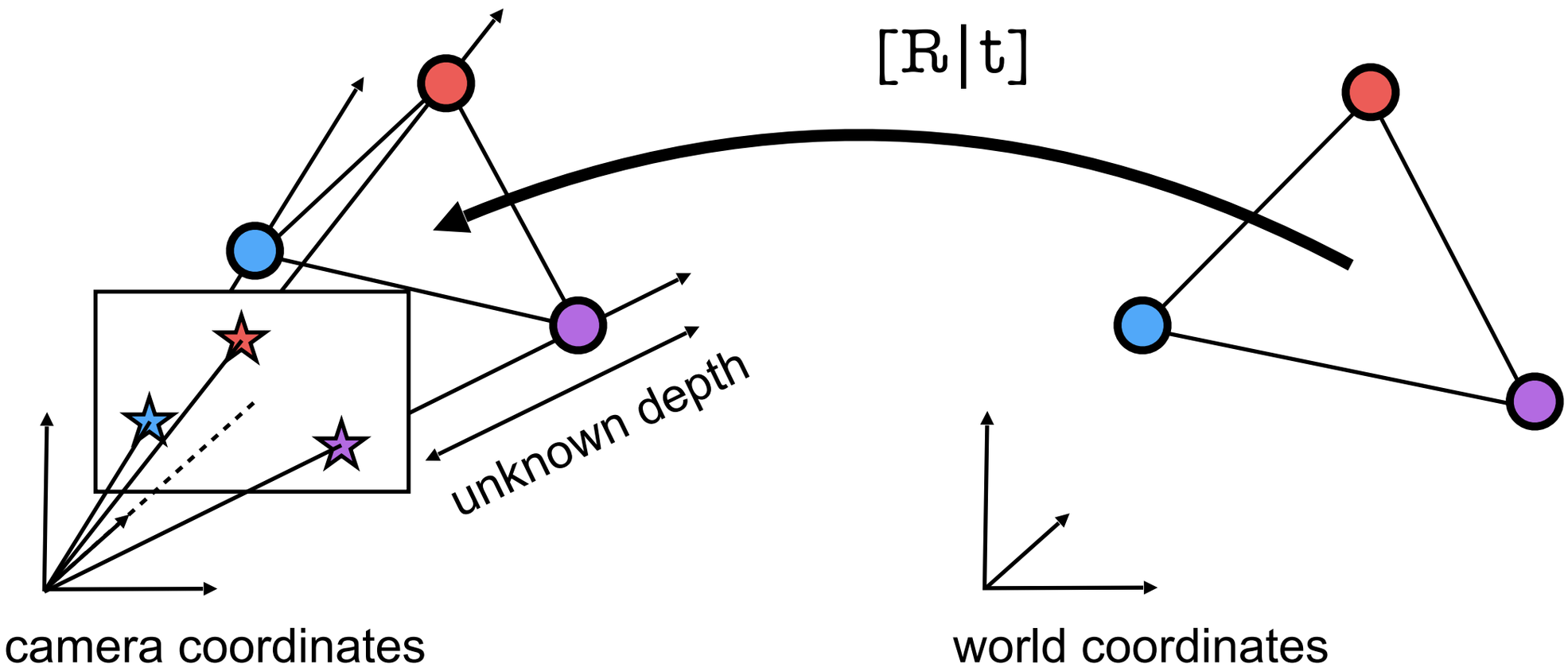
camera coordinates



world coordinates

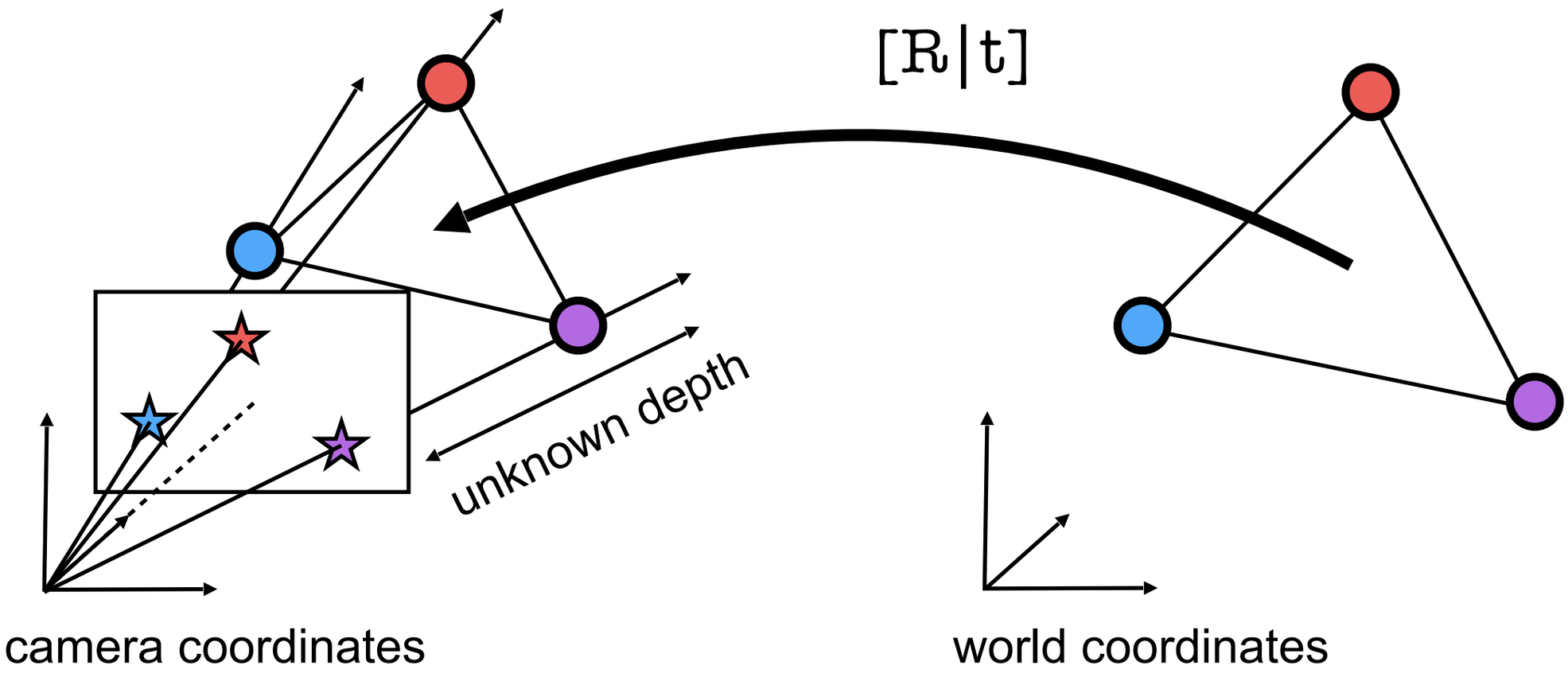
- **Case**: Intrinsic calibration known [Haralick et al., ICVJ'94]
- Recover depths: Solve 4th degree univariate polynomial [Fischler, Bolles, CACM'91]

3-Point Pose Problem (P3P)



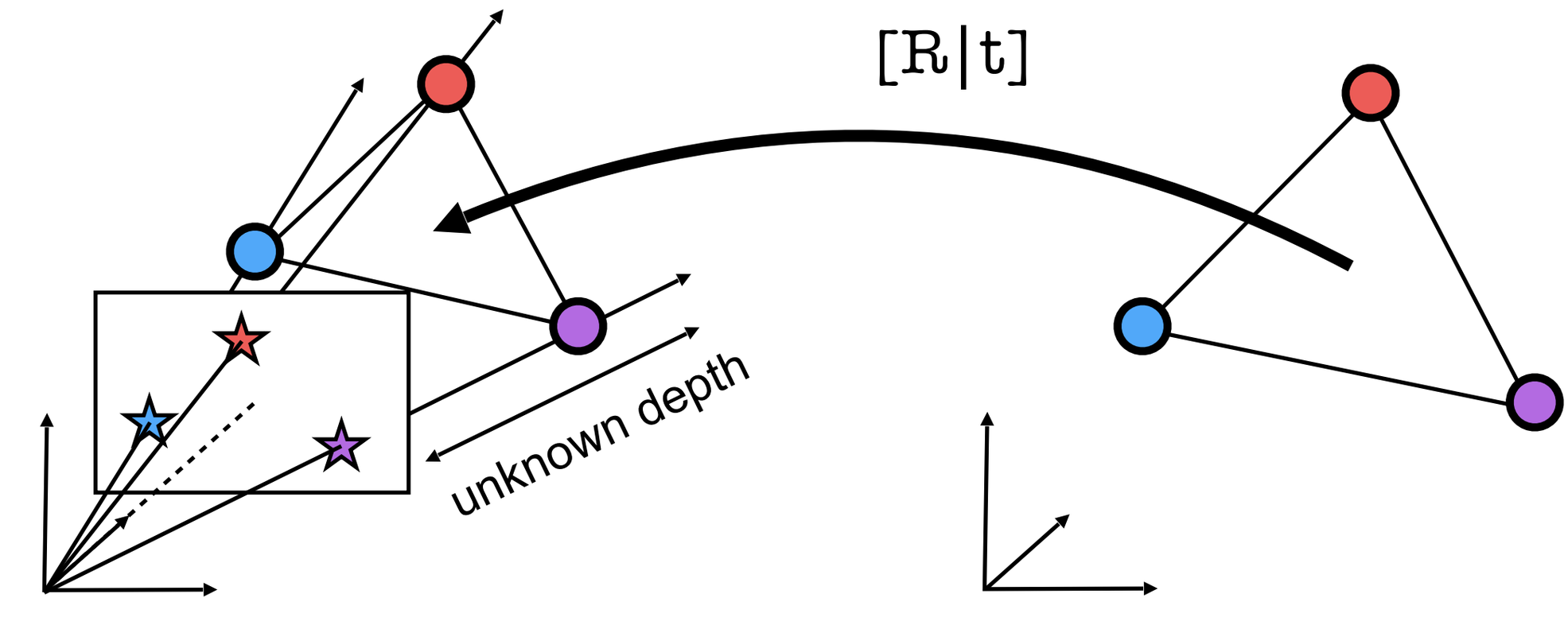
- **Case**: Intrinsic calibration known [Haralick et al., ICVJ'94]
- Recover depths: Solve 4th degree univariate polynomial [Fischler, Bolles, CACM'91]

3-Point Pose Problem (P3P)



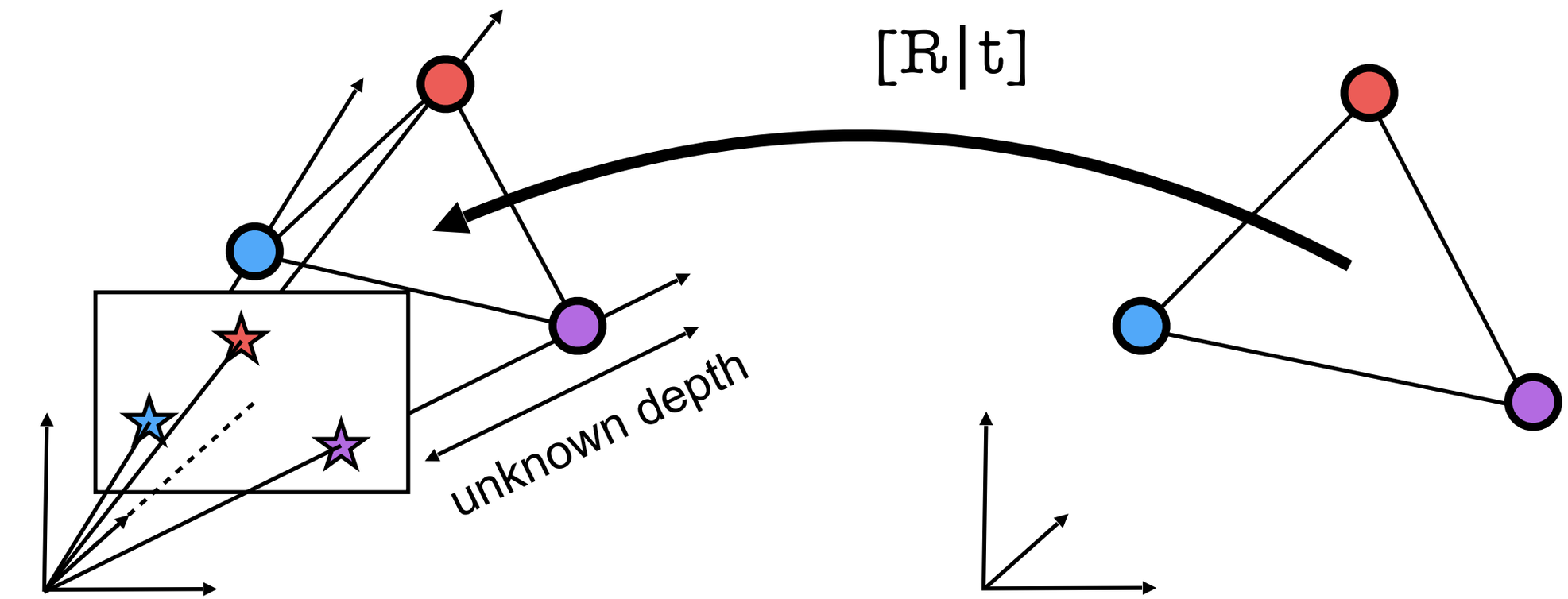
- **Case**: Intrinsic calibration known [Haralick et al., ICVJ'94]
- Recover depths: Solve 4th degree univariate polynomial [Fischler, Bolles, CACM'91]
- Recover pose by aligning local and global point positions

3-Point Pose Problem (P3P)



- **Case**: Intrinsic calibration known [Haralick et al., ICVJ'94]
- Recover depths: Solve 4th degree univariate polynomial [Fischler, Bolles, CACM'91]
- Recover pose by aligning local and global point positions
- Very efficient: ~2μs total [Kneip et al., CVPR'11] [code]

3-Point Pose Problem (P3P)



- **Case**: Intrinsic calibration known [Haralick et al., ICVJ'94]
- Recover depths: Solve 4th degree univariate polynomial [Fischler, Bolles, CACM'91]
- Recover pose by aligning local and global point positions
- Very efficient: ~2µs total [Kneip et al., CVPR'11] [code]
- Up to four solutions: Disambiguate using 4th point

Unknown Intrinsics

General projection equation:

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathtt{P}\mathbf{X}$$

Unknown Intrinsics

General projection equation:

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathtt{P}\mathbf{X}$$

6-point DLT algorithm, similar to homography DLT:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \times \mathtt{P}\mathbf{X} = \mathbf{0}$$

Unknown Intrinsics

Two linear independent equations per 2D-3D match:

$$\begin{bmatrix} \mathbf{0}^\top & -w_i \mathbf{X}_i{}^\top & y_i \mathbf{X}_i{}^\top \\ w_i \mathbf{X}_i{}^\top & \mathbf{0}^\top & -x_i \mathbf{X}_i{}^\top \end{bmatrix} \begin{pmatrix} \mathbf{P}^1 \\ \mathbf{P}^2 \\ \mathbf{P}^3 \end{pmatrix} = \mathbf{0}$$

Unknown Intrinsics

Two linear independent equations per 2D-3D match:

$$\begin{bmatrix} \mathbf{0}^\top & -w_i \mathbf{X}_i{}^\top & y_i \mathbf{X}_i{}^\top \\ w_i \mathbf{X}_i{}^\top & \mathbf{0}^\top & -x_i \mathbf{X}_i{}^\top \end{bmatrix} \begin{pmatrix} \mathbf{P}^1 \\ \mathbf{P}^2 \\ \mathbf{P}^3 \end{pmatrix} = \mathbf{0}$$

12 unknowns (11 DoF): 6 points for minimal solution

Unknown Intrinsics

Two linear independent equations per 2D-3D match:

$$\begin{bmatrix} \mathbf{0}^\top & -w_i \mathbf{X}_i{}^\top & y_i \mathbf{X}_i{}^\top \\ w_i \mathbf{X}_i{}^\top & \mathbf{0}^\top & -x_i \mathbf{X}_i{}^\top \end{bmatrix} \begin{pmatrix} \mathbf{P}^1 \\ \mathbf{P}^2 \\ \mathbf{P}^3 \end{pmatrix} = \mathbf{0}$$

12 unknowns (11 DoF): 6 points for minimal solution

Linear least squares solution for >6 points

Unknown Intrinsics

Two linear independent equations per 2D-3D match:

$$\begin{bmatrix} \mathbf{0}^\top & -w_i \mathbf{X}_i{}^\top & y_i \mathbf{X}_i{}^\top \\ w_i \mathbf{X}_i{}^\top & \mathbf{0}^\top & -x_i \mathbf{X}_i{}^\top \end{bmatrix} \begin{pmatrix} \mathbf{P}^1 \\ \mathbf{P}^2 \\ \mathbf{P}^3 \end{pmatrix} = \mathbf{0}$$

12 unknowns (11 DoF): 6 points for minimal solution

Linear least squares solution for >6 points

Don’t forget normalization (normalized DLT)

Unknown Intrinsics

Two linear independent equations per 2D-3D match:

$$\begin{bmatrix} \mathbf{0}^\top & -w_i \mathbf{X}_i{}^\top & y_i \mathbf{X}_i{}^\top \\ w_i \mathbf{X}_i{}^\top & \mathbf{0}^\top & -x_i \mathbf{X}_i{}^\top \end{bmatrix} \begin{pmatrix} \mathbf{P}^1 \\ \mathbf{P}^2 \\ \mathbf{P}^3 \end{pmatrix} = \mathbf{0}$$

12 unknowns (11 DoF): 6 points for minimal solution

Linear least squares solution for >6 points

Don’t forget normalization (normalized DLT)

Degenerate if all points in single plane

Lessons Learned

- Main lessons from this lecture
  - Incremental Structure-from-Motion
    - Relative pose estimation via essential / fundamental matrix
    - Triangulation via RANSAC
  - Absolute pose estimation

Lessons Learned

- Main lessons from this lecture
  - Incremental Structure-from-Motion
    - Relative pose estimation via essential / fundamental matrix
    - Triangulation via RANSAC
  - Absolute pose estimation
- Next lecture: Generative Neural Networks