SSY098 - Image Analysis

Lecture 10 - Camera & 3D Geometry

Torsten Sattler
(slides adapted from Olof Enqvist)

Last Lecture

Jan. 20	Introduction, Linear classifiers and filtering	
Jan. 23	Filtering, gradients, scale	Lab 1
Jan. 27	Local features	
Jan. 30	Learning a classifier	Lab 2
Feb. 3	Convolutional neural networks	
Feb. 6	More convolutional neural networks	
Feb. 10	Robust model fitting and RANSAC	Lab 3
Feb. 13	Image registration	
Feb. 17	**Camera Geometry**	Lab 4
Feb. 20	More camera geometry	
Feb. 24	Generative neural networks	
Feb. 27	Generative neural networks	
Mar. 2	Visual Localization & Feature Learning	
Mar. 9	No lecture	

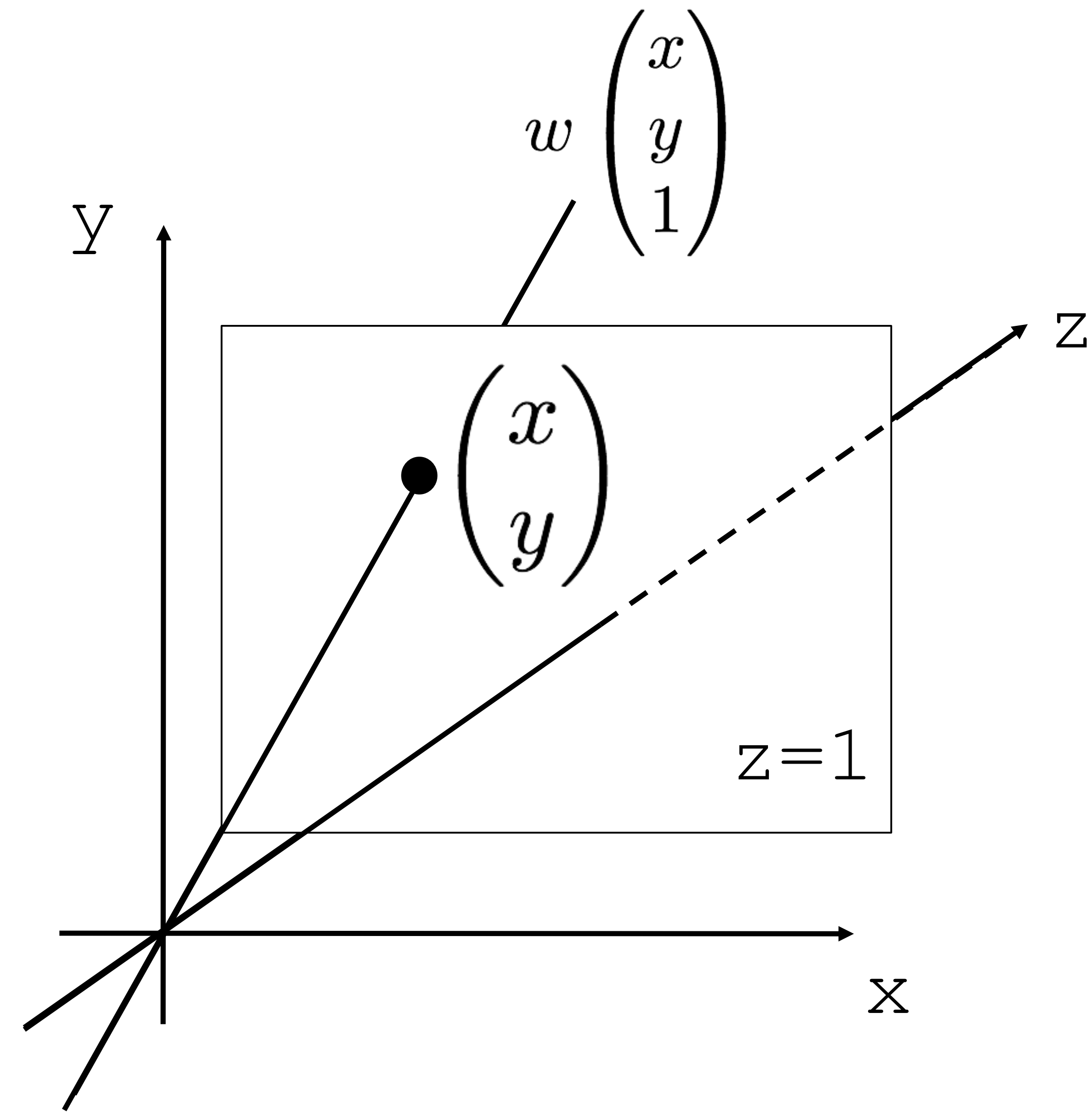
Last Lecture

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

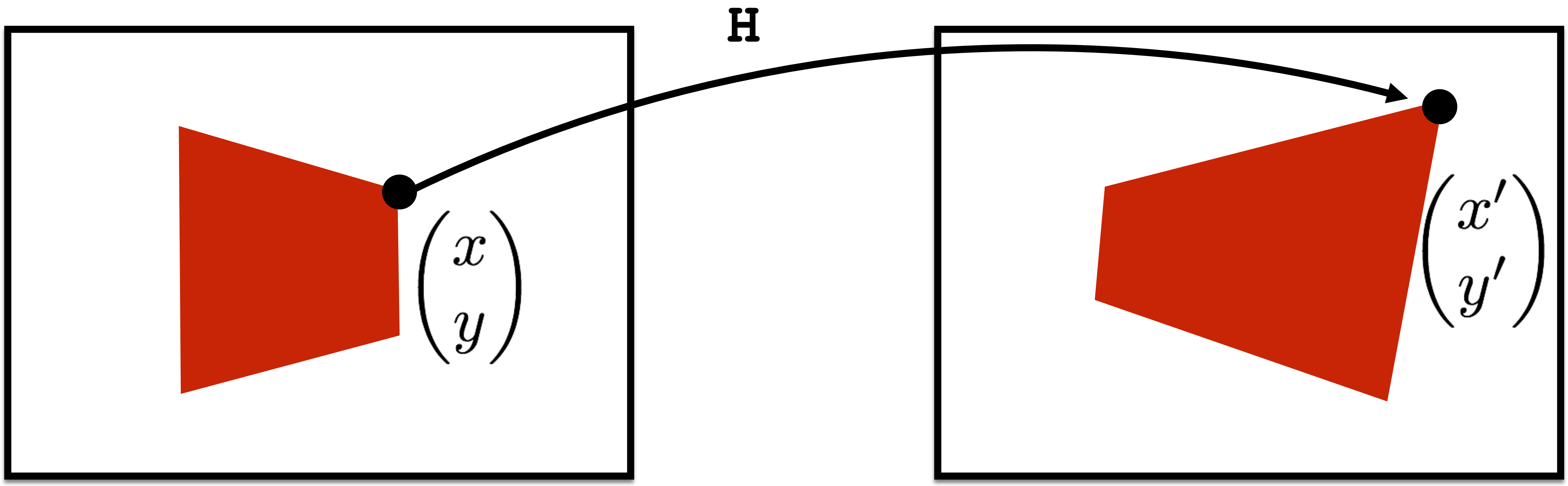
$$\begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

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

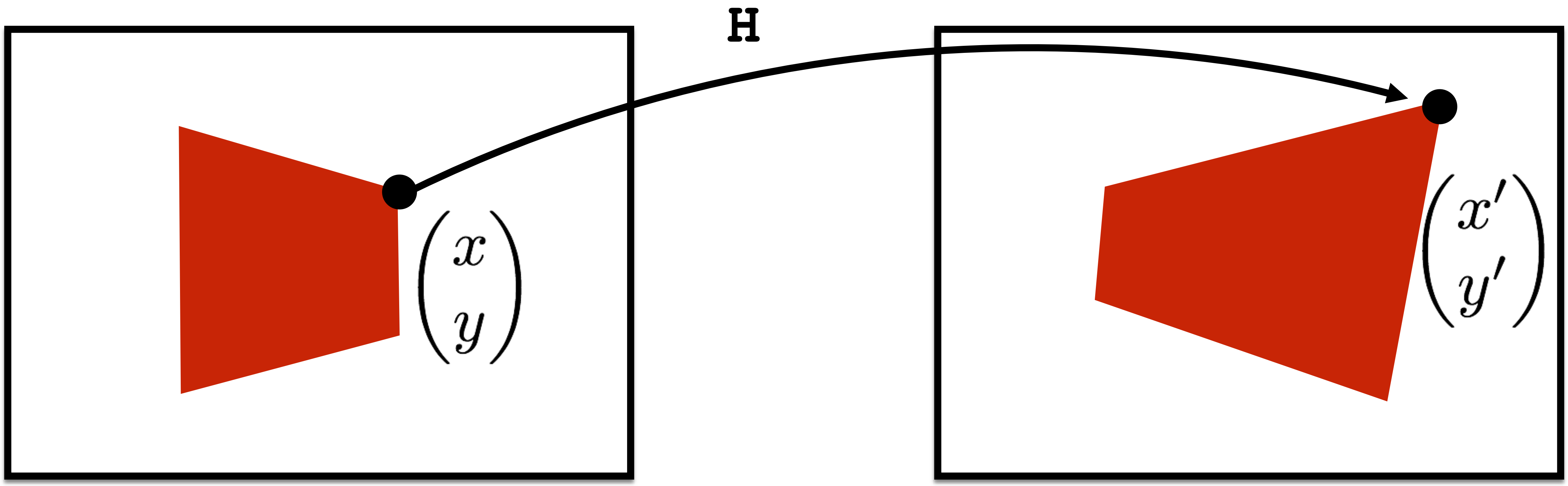
- $\mathbf{H}$ needs to be invertible
- $\mathbf{H}$ has 8 Degrees-of-Freedom (DoF)

Projective mapping (homography)

Using Homogenous Coordinates

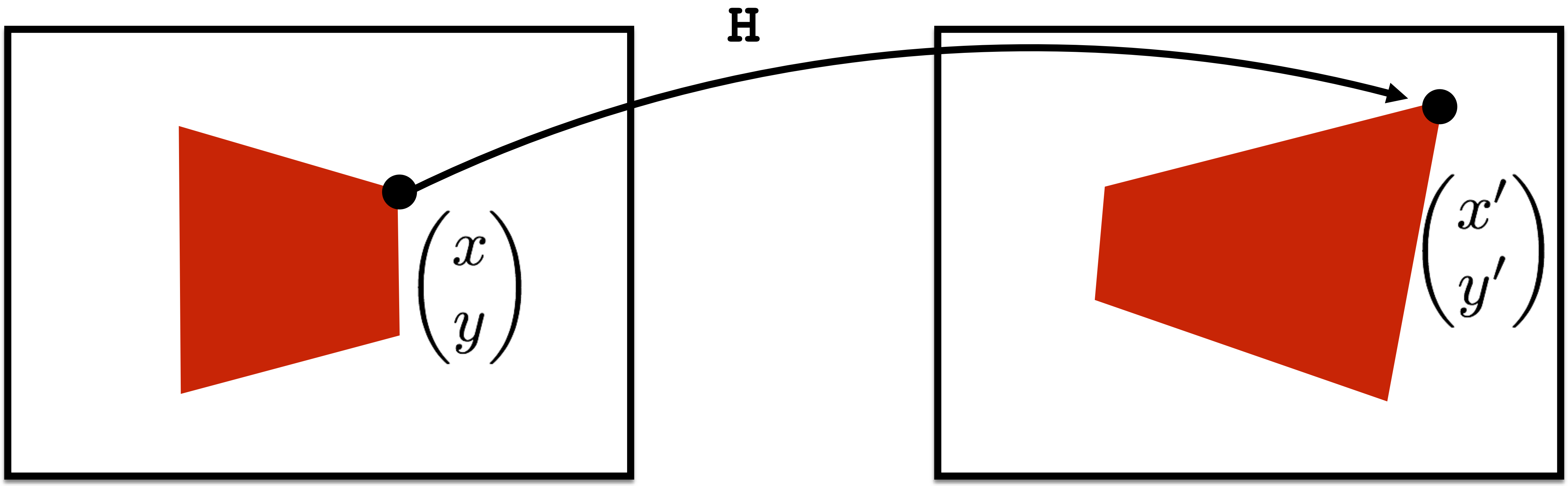


Using Homogenous Coordinates



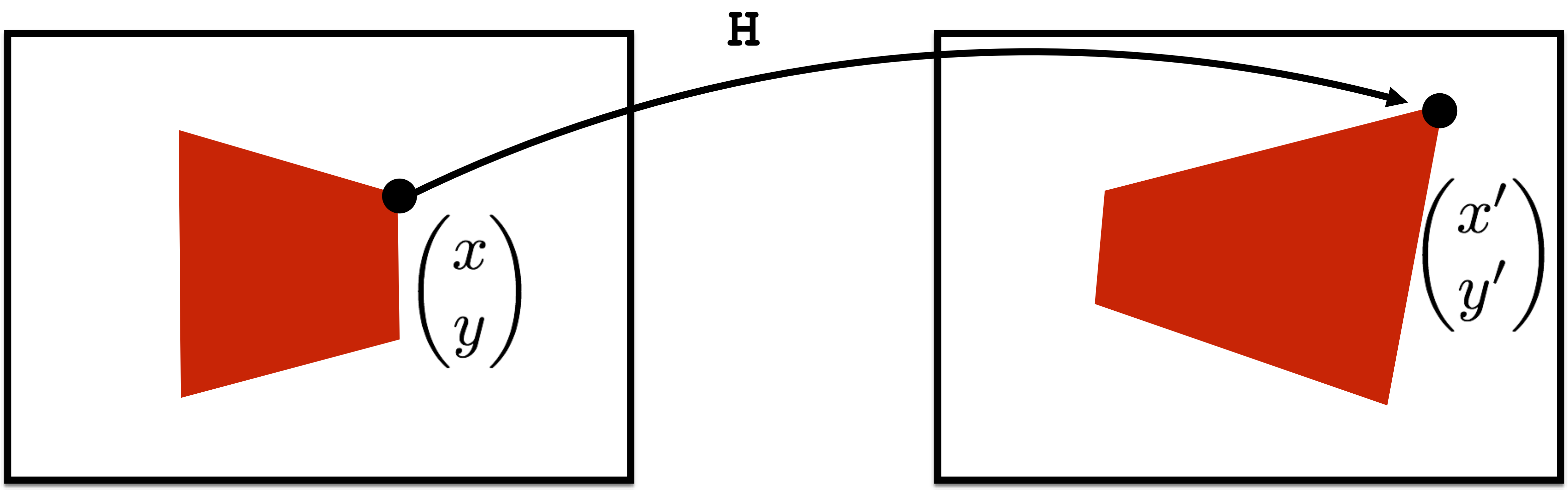
- "Homogenize": $\begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$

Using Homogenous Coordinates



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

Using Homogenous Coordinates



$$\begin{pmatrix} x \\ y \end{pmatrix} \qquad \begin{pmatrix} x' \\ y' \end{pmatrix}$$

- "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

<u>Objective</u>

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

<u>Algorithm</u>

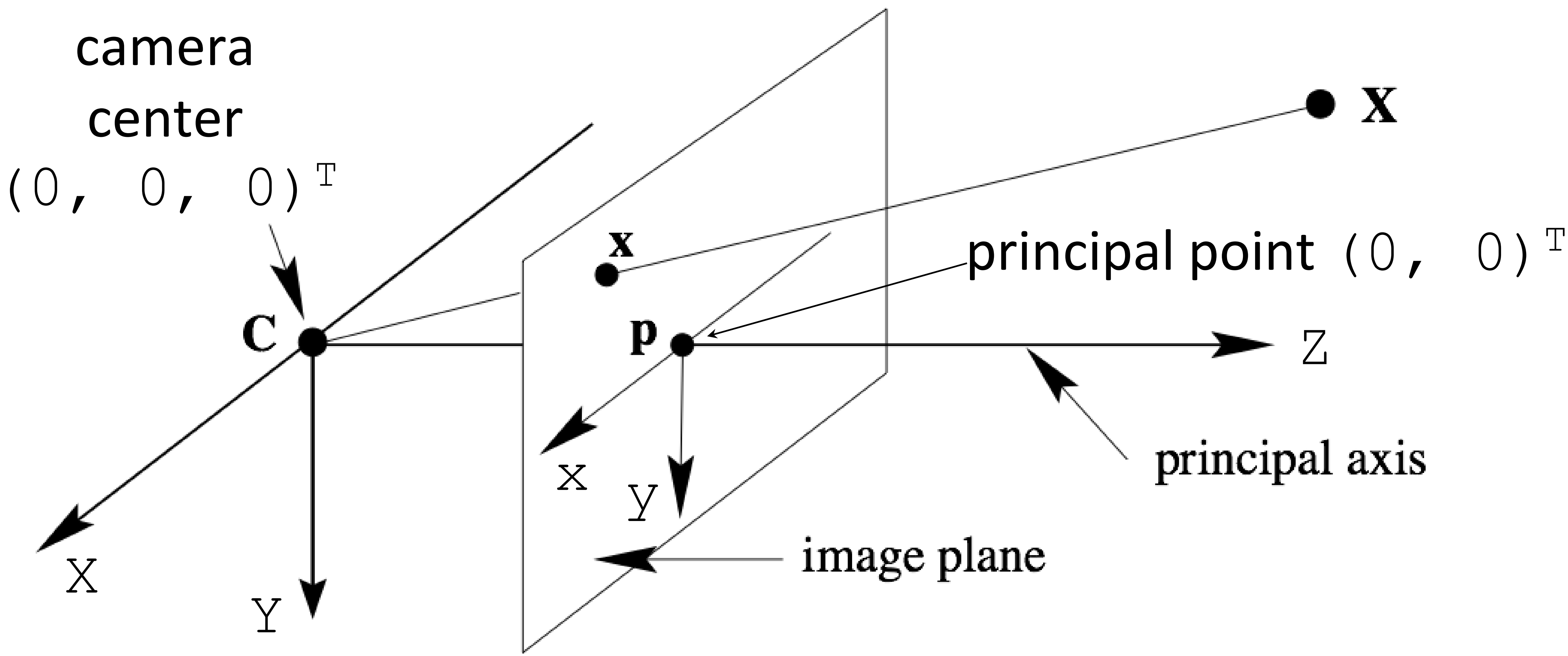
- Normalize points: $\tilde{\mathbf{x}}_i = \mathbf{T}_{\mathrm{norm}}\mathbf{x}_i, \tilde{\mathbf{x}}_i' = \mathbf{T}'_{\mathrm{norm}}\mathbf{x}_i'$
- Apply DLT algorithm to $\tilde{\mathbf{x}}_i \leftrightarrow \tilde{\mathbf{x}}_i'$
- Denormalize solution: $\mathbf{H} = \mathbf{T}'^{-1}_{\mathrm{norm}}\tilde{\mathbf{H}}\mathbf{T}_{\mathrm{norm}}$

Normalization (independently per image):

- Translate points such that centroid is at origin
- Isotropic scaling such that mean distance to origin is $\sqrt{2}$

Hartley and Zisserman. *Multiple View Geometry in Computer Vision*, 2nd edition, Cambridge University Press, 2004.

Last Lecture



Pinhole camera model

Last Lecture

General **intrinsic camera calibration matrix**:

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

Projection

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

Last Lecture

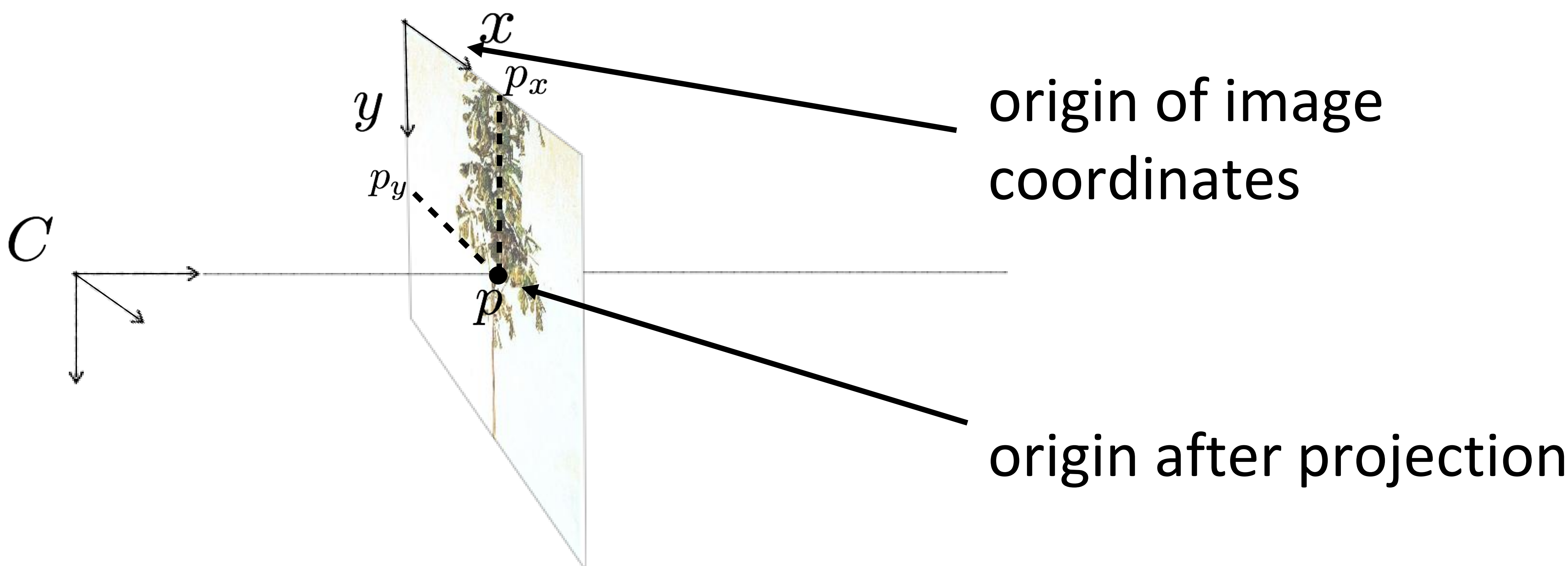
General **intrinsic camera calibration matrix**:

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

Projection

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

Last Lecture

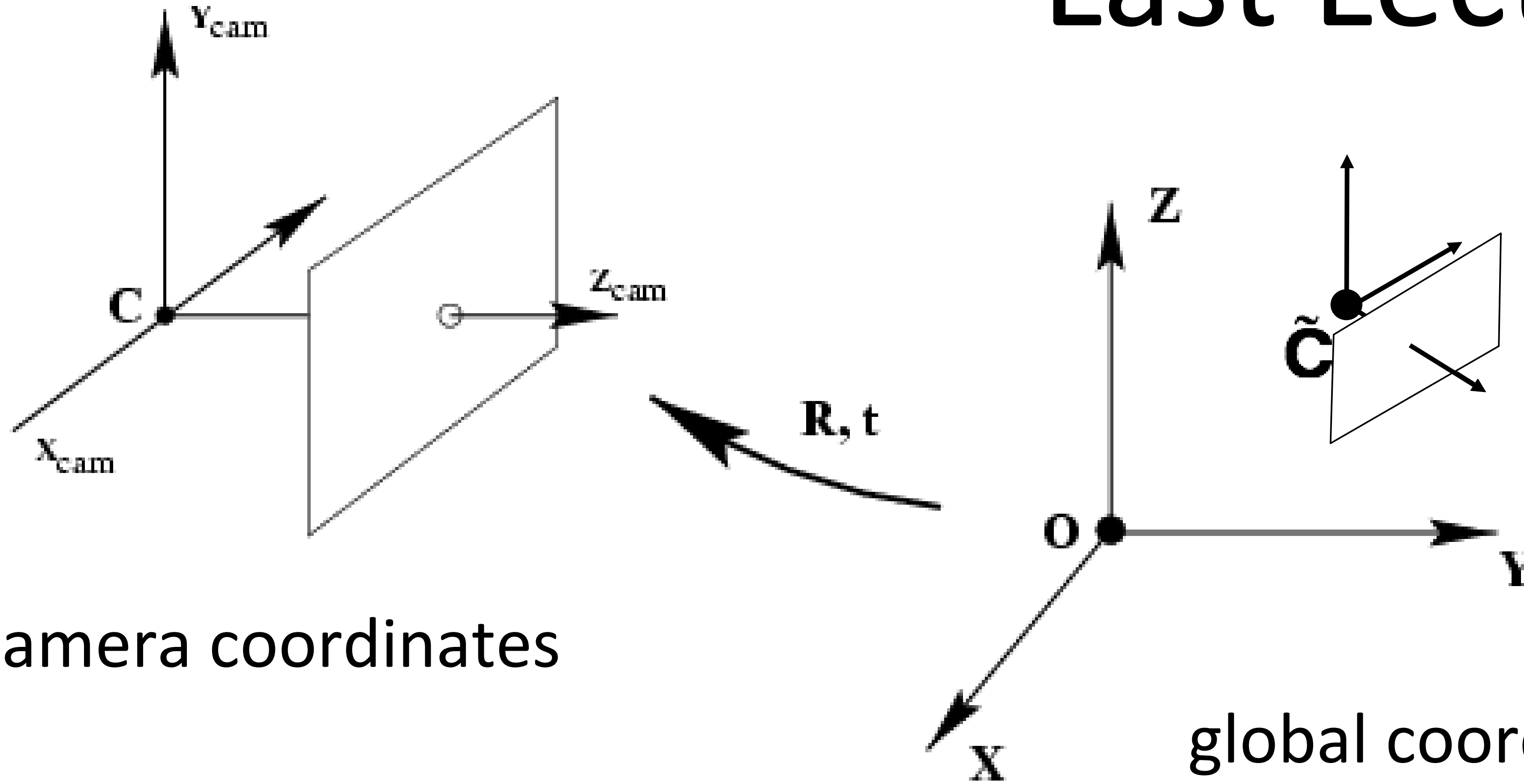


Mapping to pixel coordinates: $\begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} x + p_y \\ y + p_y \end{pmatrix}$

Projection as matrix multiplication:

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

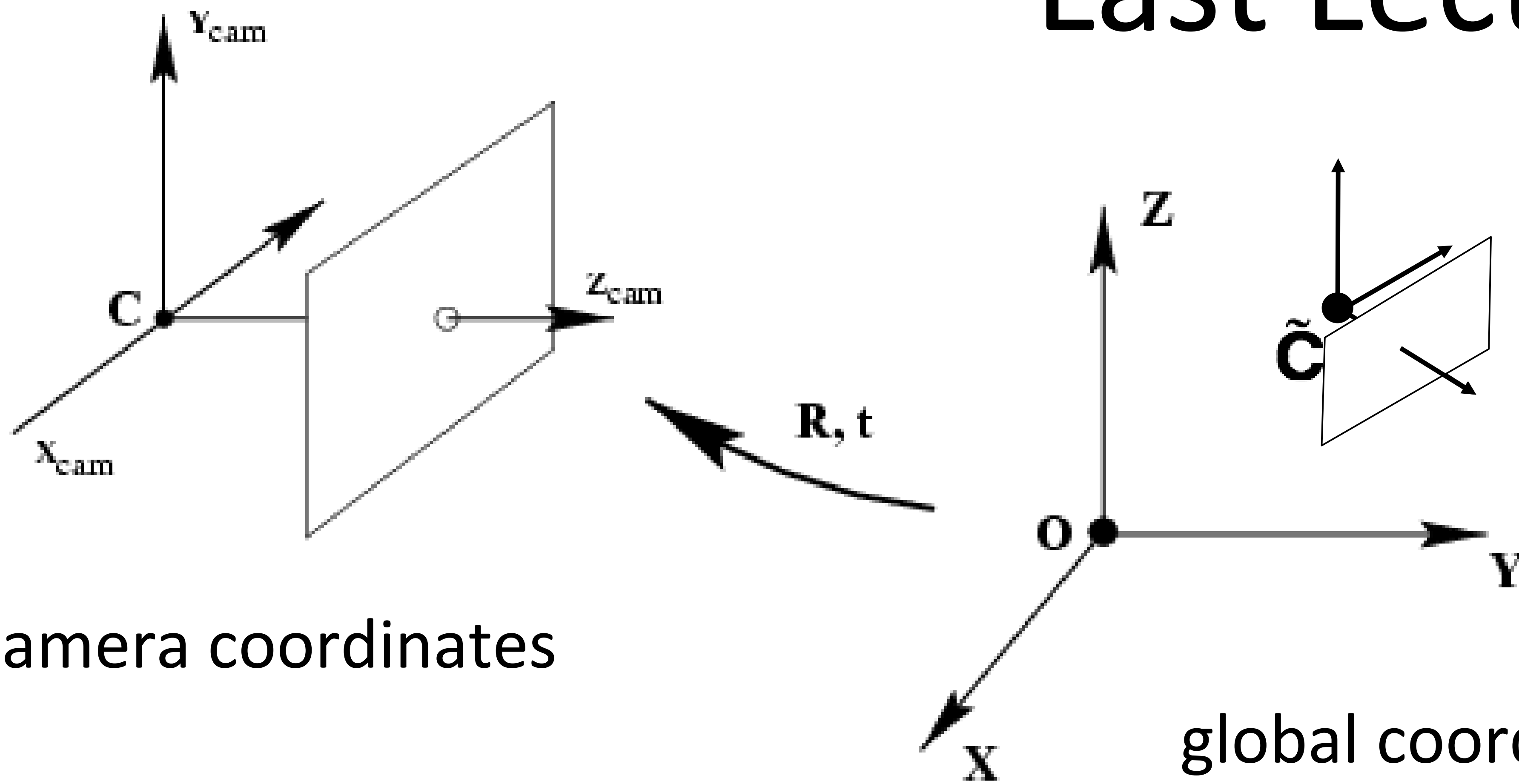
Last Lecture



camera coordinates

global coordinates

Last Lecture

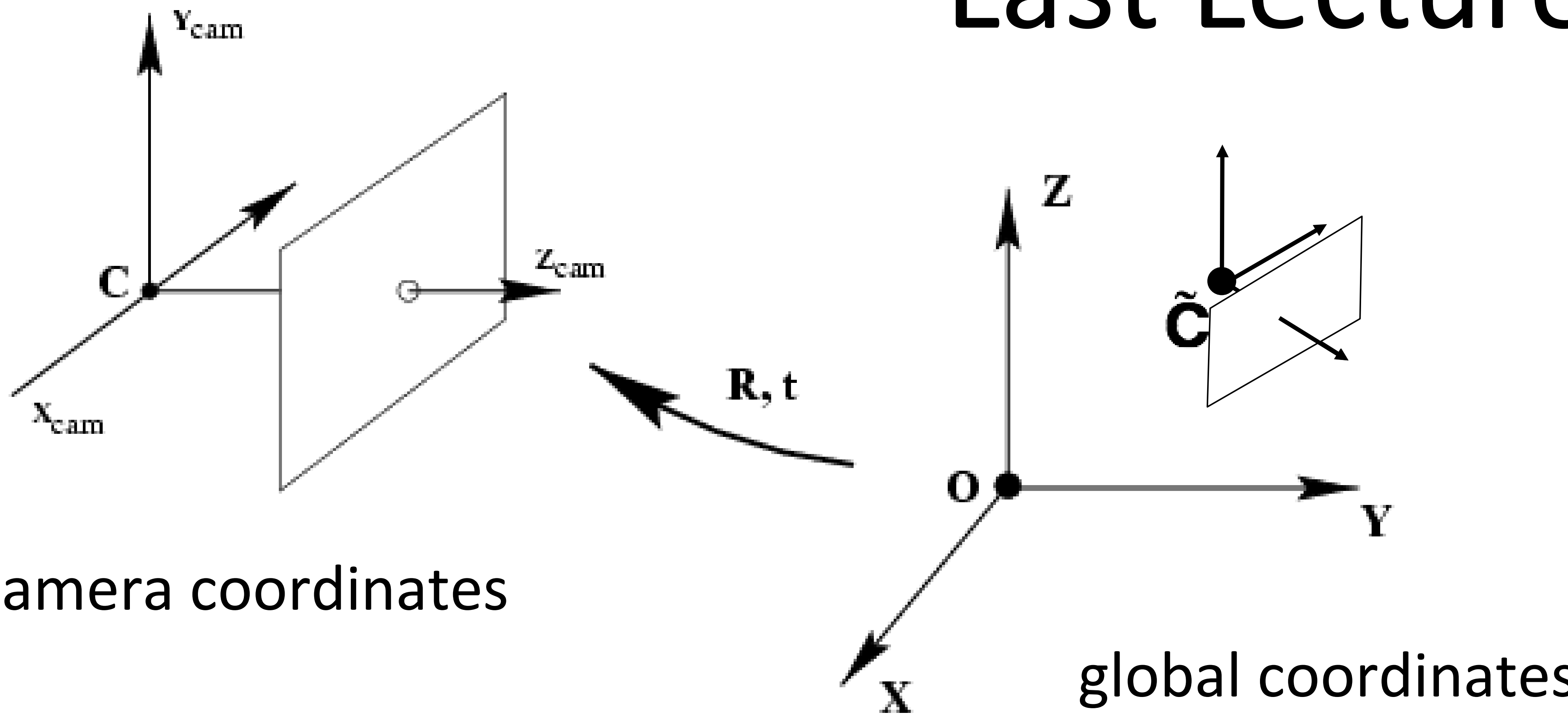


camera coordinates

global coordinates

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

Last Lecture



camera coordinates

global coordinates

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

Extrinsic and intrinsic camera parameters

Last Lecture

Radial Distortion

Last Lecture

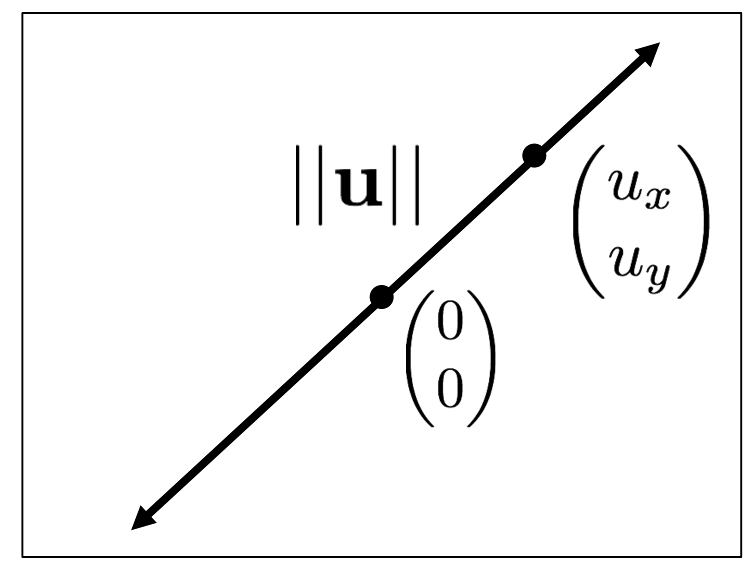
Radial Distortion

Project 3D point into camera coordinates

$$\mathbf{u} = \begin{pmatrix} u_x \\ u_y \end{pmatrix} = \begin{pmatrix} X/Z \\ Y/Z \end{pmatrix}$$

Compute radial distortion factor

$$r(\mathbf{u}) = 1 + \kappa_1 ||\mathbf{u}||^2 + \kappa_2 ||\mathbf{u}||^4$$



Compute pixel position

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f \cdot r(\mathbf{u}) \cdot u_x + p_x \\ f \cdot r(\mathbf{u}) \cdot u_y + p_y \end{pmatrix}$$

Rolling Shutter Effect

Global shutter

Rolling shutter

youtu.be/7TGKFdrY9aw

Slide credit: Cenek Albl

Rolling Shutter Effect

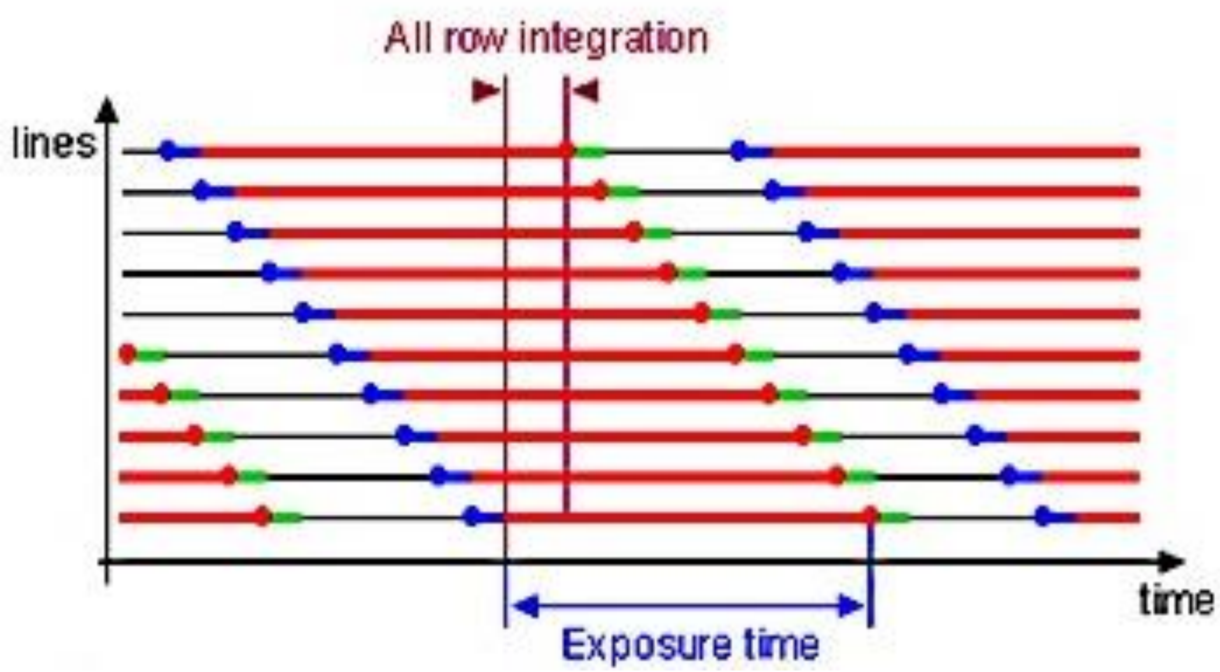


Image recorded line by line

Slide credit: Cenek Albl

Rolling Shutter Effect

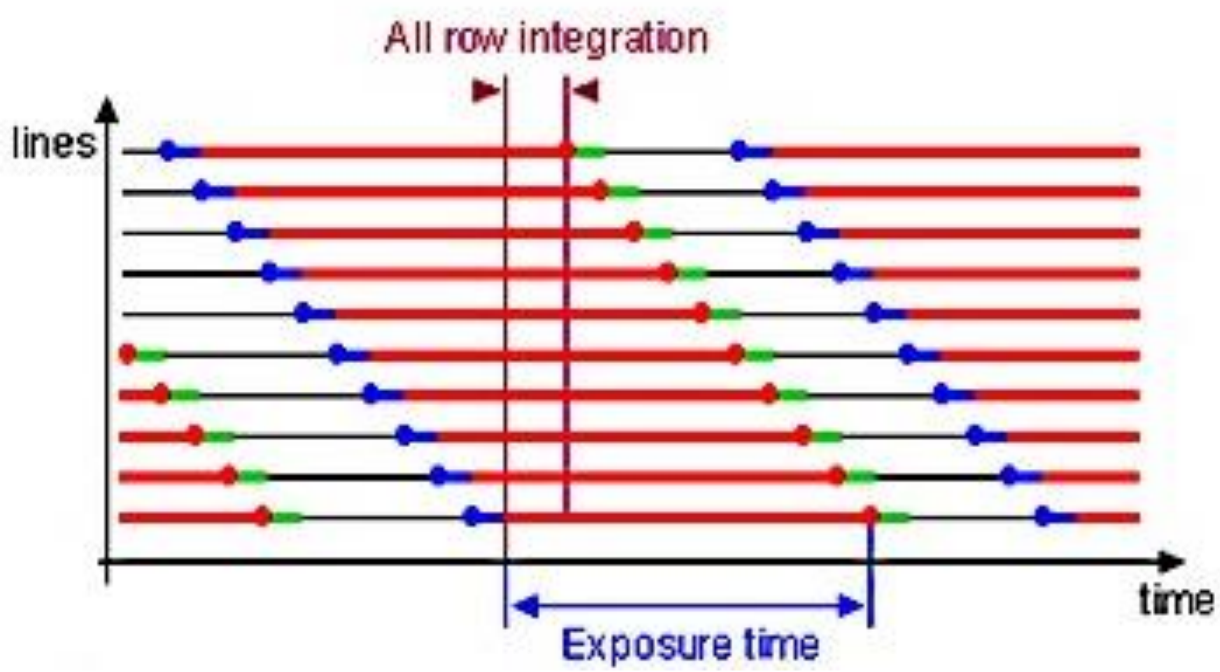


Image recorded line by line

Rolling shutter effect

Slide credit: Cenek Albl

Rolling Shutter Effect

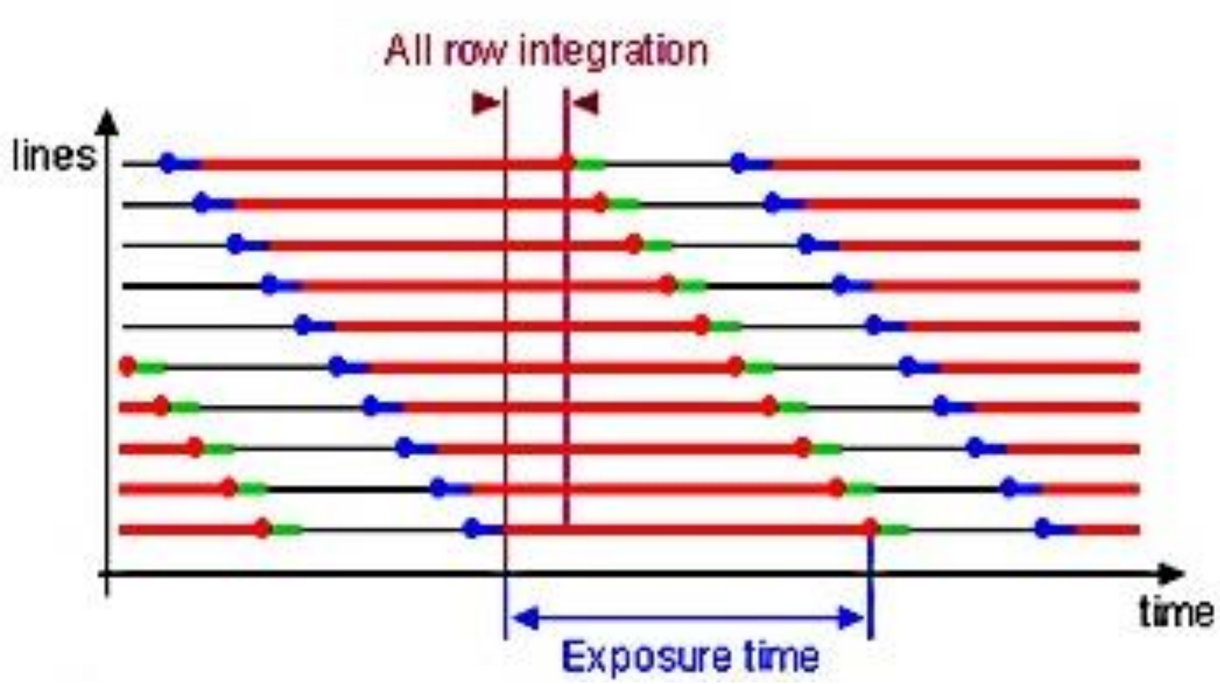


Image recorded line by line

Rolling shutter effect

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

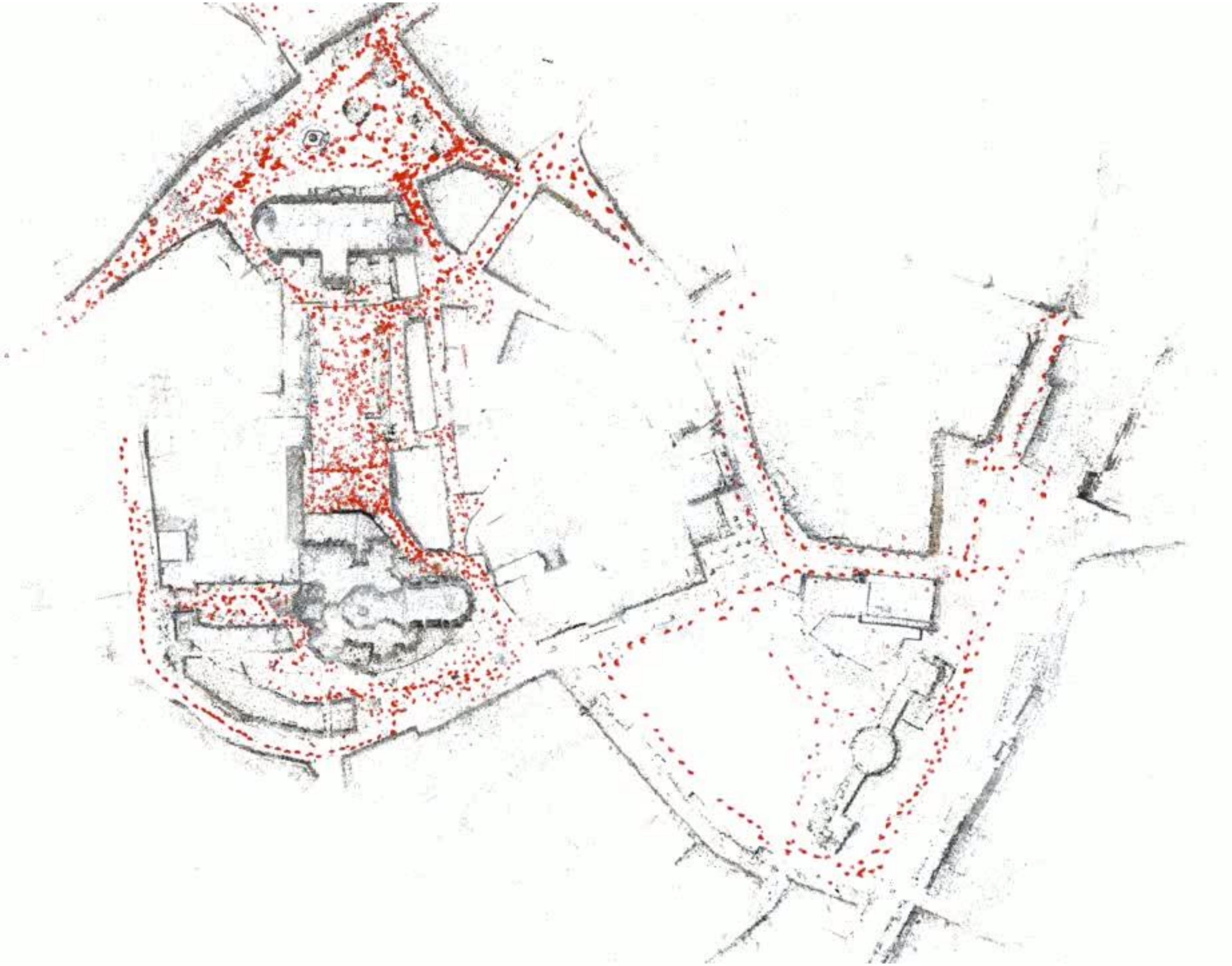
Slide credit: Cenek Albl

More Camera Geometry

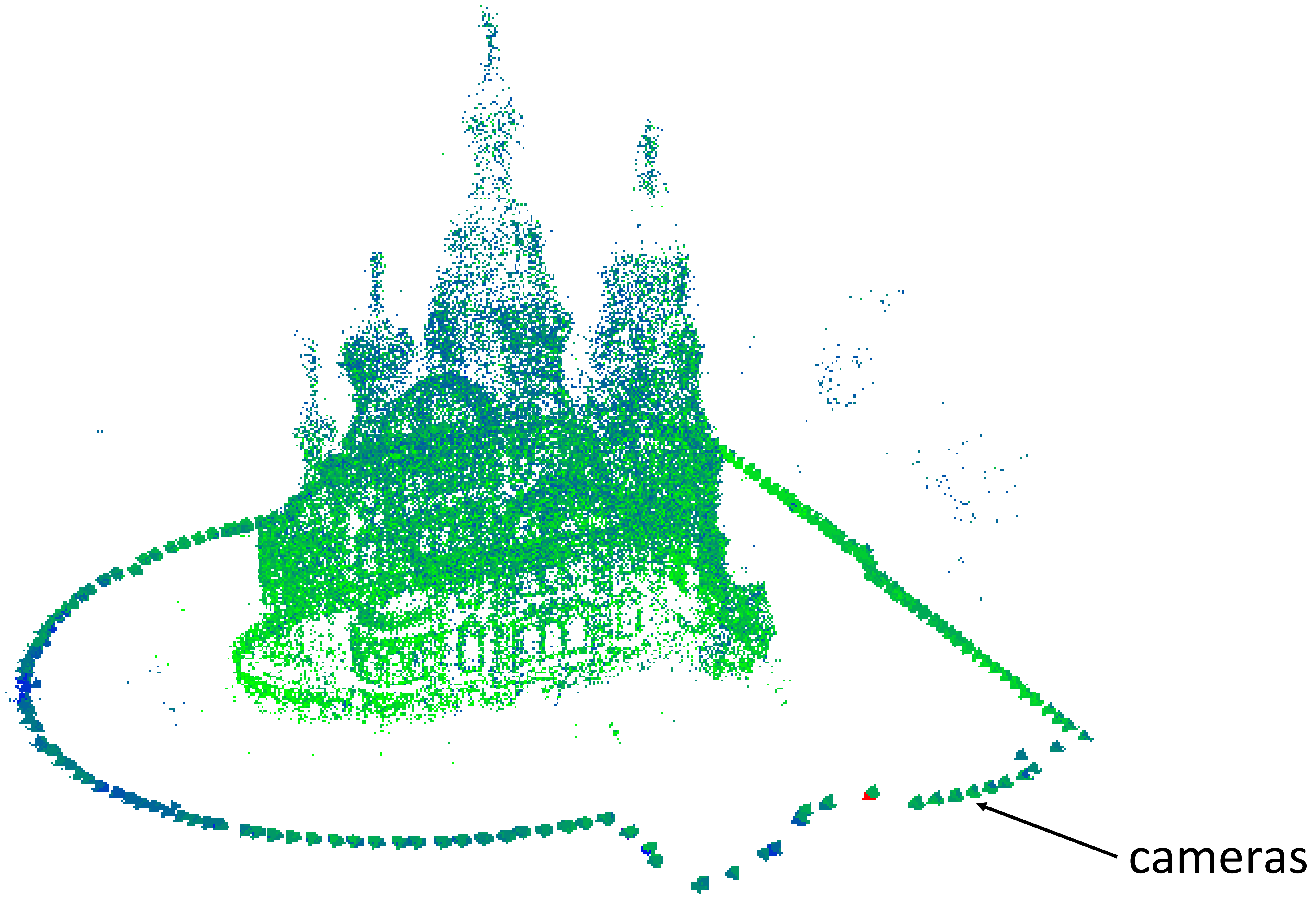
Today

3D Reconstruction

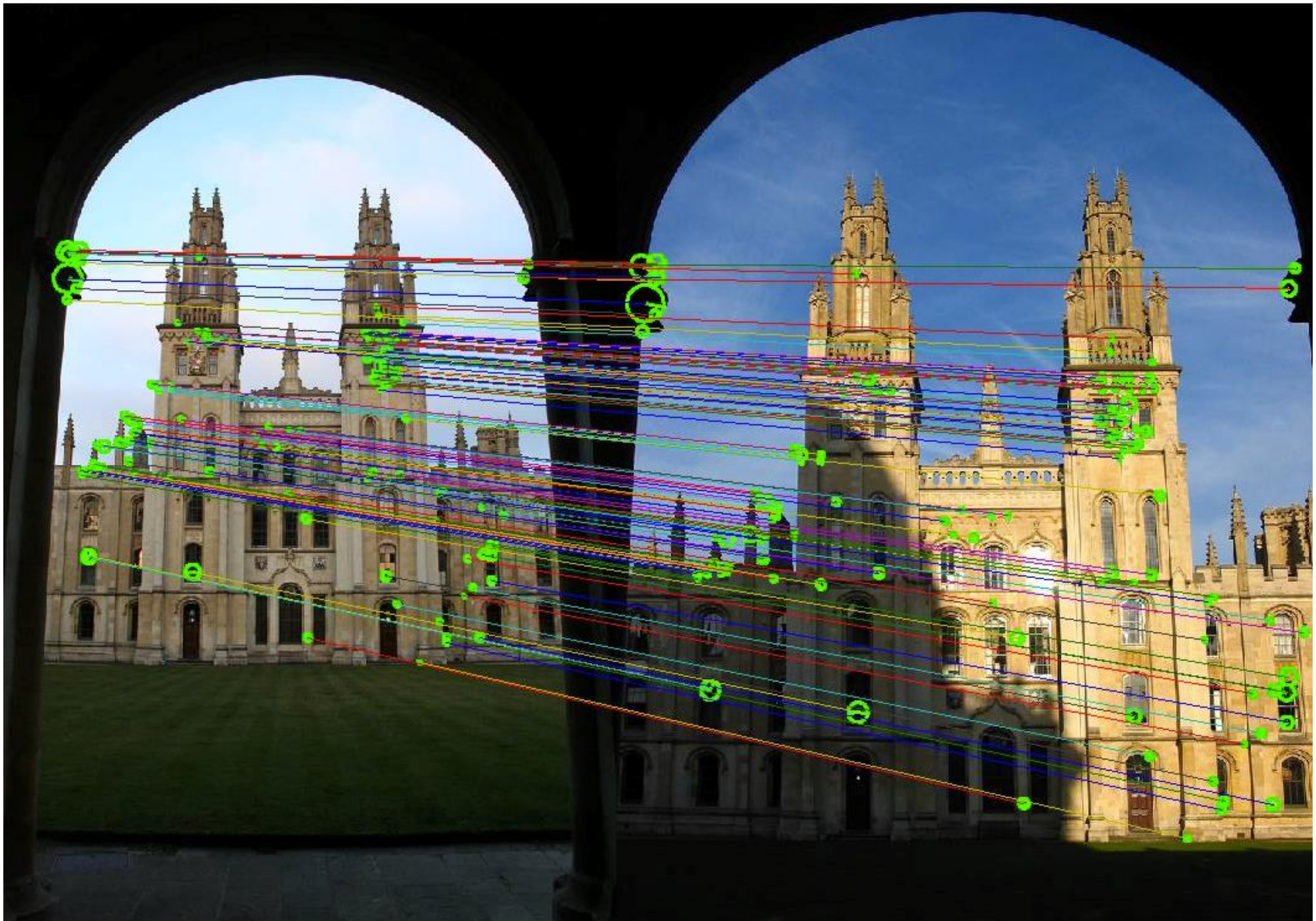
Structure-from-Motion



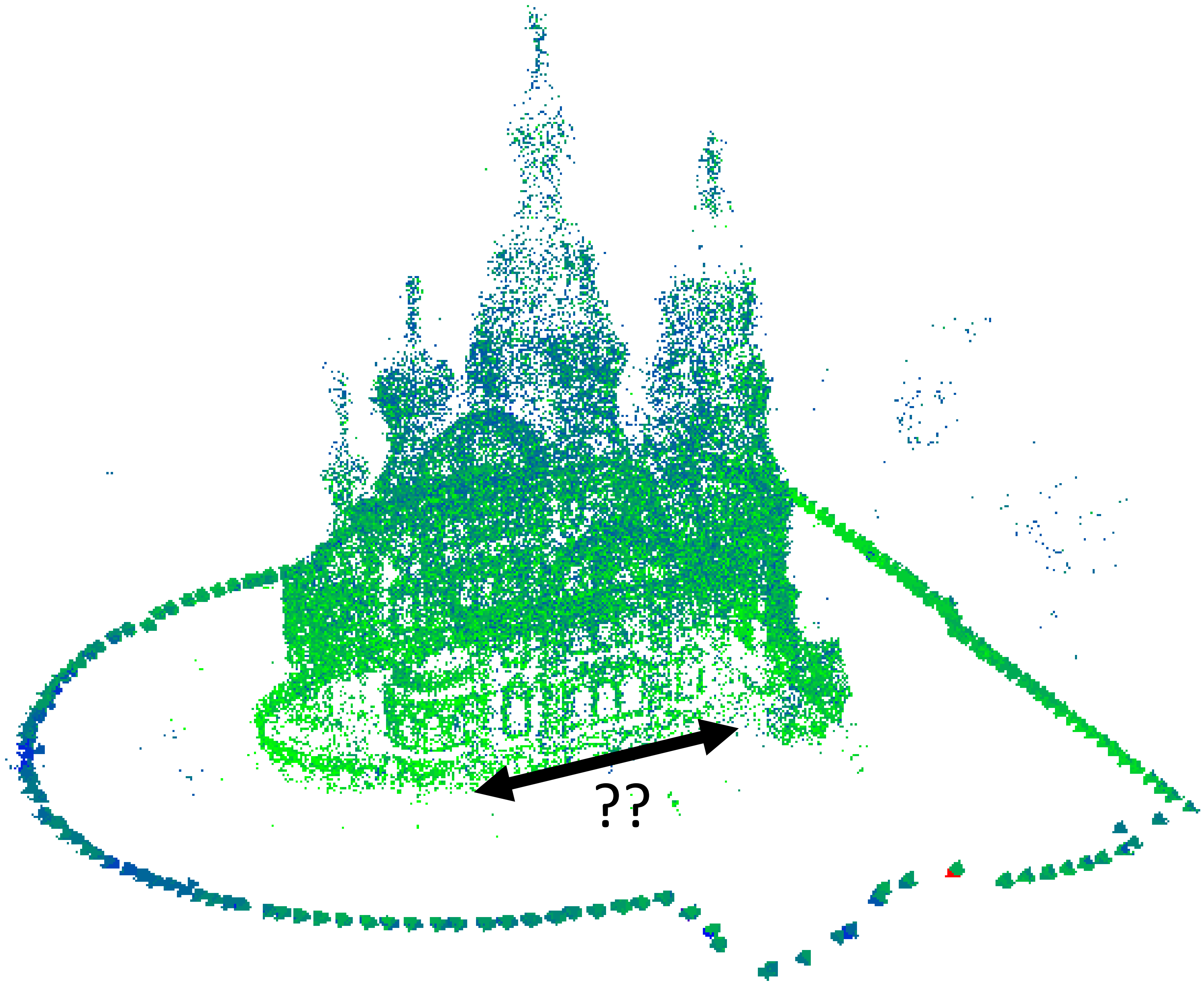
Structure-from-Motion



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 $\mathtt{R}$, $\mathbf{t}$, intrinsics $\mathtt{K}$

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

3D Models Are Defined Up To Scale

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

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

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

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

3D Models Are Defined Up To Scale

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

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

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

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

3D Models Are Defined Up To Scale

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

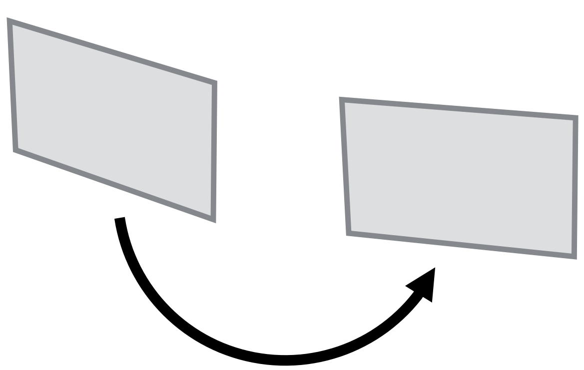
$$\mathtt{K}\left(\mathtt{R}\mathbf{X}+\mathbf{t}\right) \quad = \quad \begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \end{pmatrix}$$

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

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

Sequential Structure-from-Motion

Initialize motion from two views



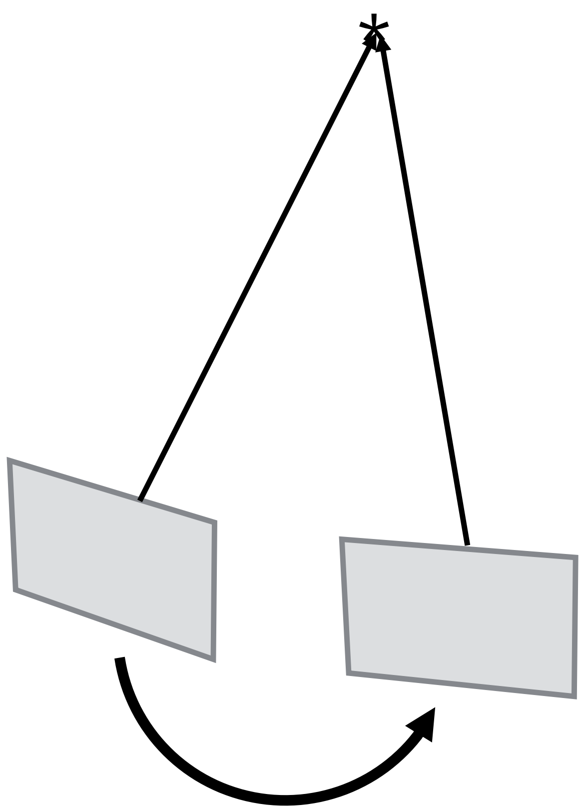
$\mathtt{R}, \mathbf{t}$ with $||\mathbf{t}|| = 1$

Relative pose for two images

Sequential Structure-from-Motion

Initialize motion from two views

Initialize structure from two views



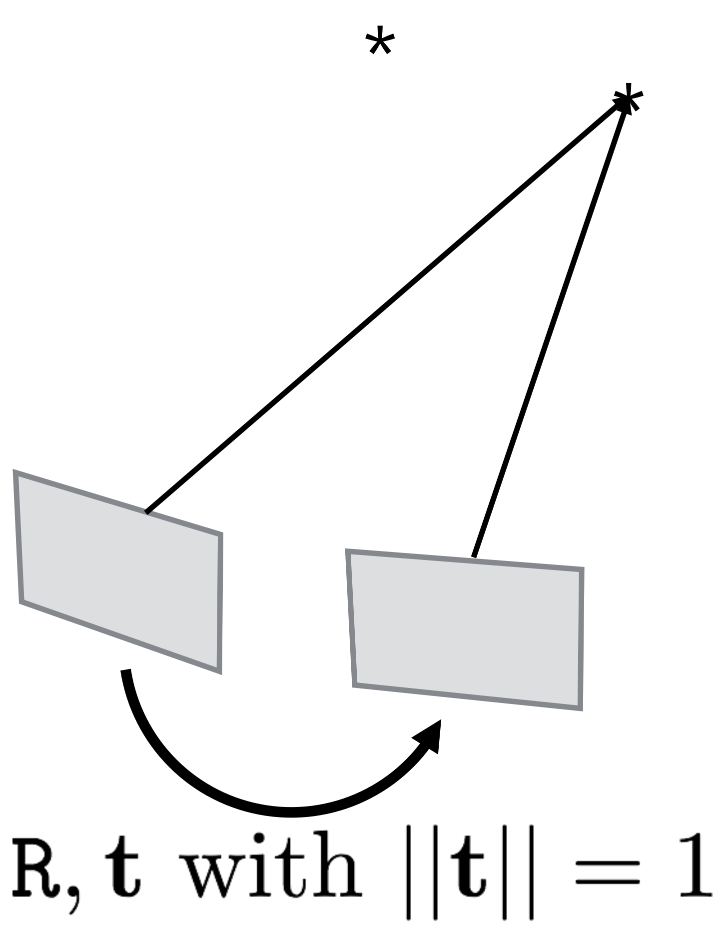
$\mathrm{R}, \mathbf{t} \text{ with } ||\mathbf{t}|| = 1$

Triangulate 3D points

Sequential Structure-from-Motion

Initialize motion from two views

Initialize structure from two views

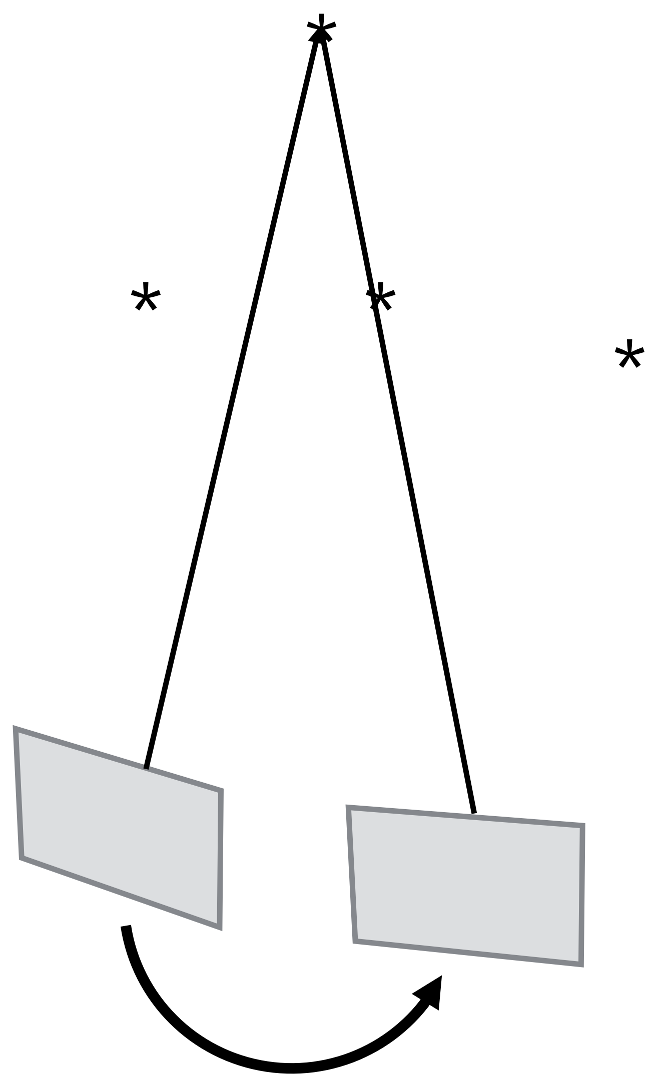


Triangulate 3D points

Sequential Structure-from-Motion

Initialize motion from two views

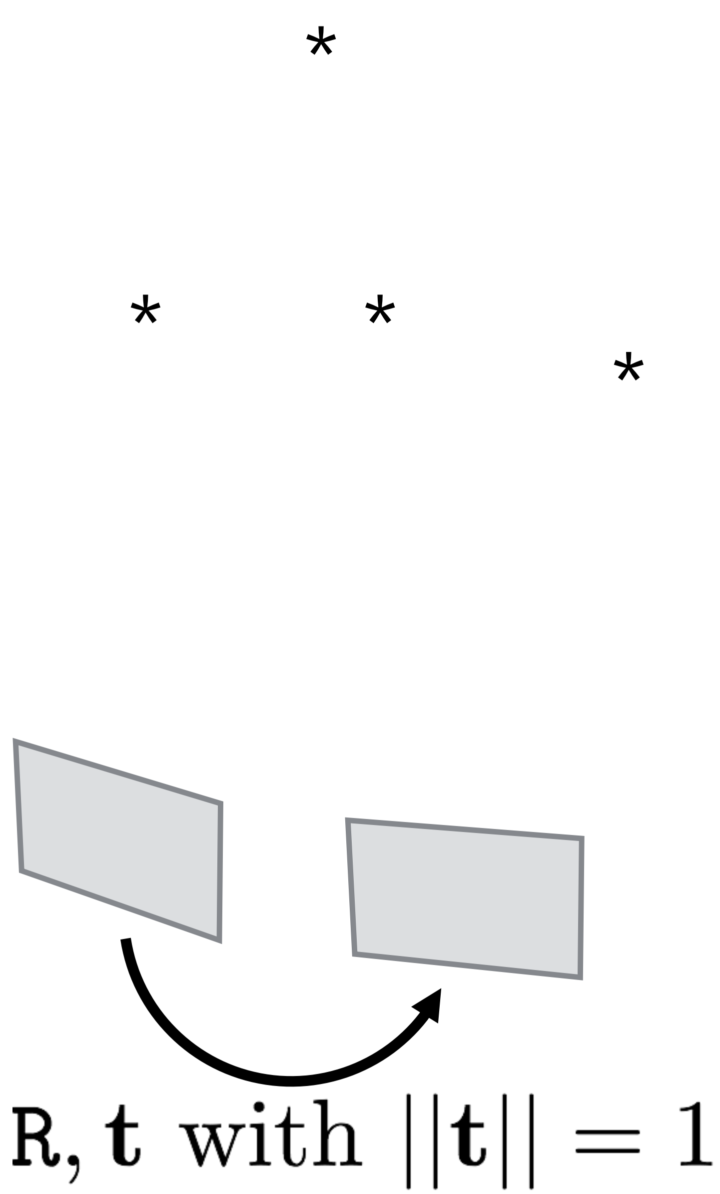
Initialize structure from two views



$$\mathrm{R}, \mathbf{t} \text{ with } ||\mathbf{t}|| = 1$$

Triangulate 3D points

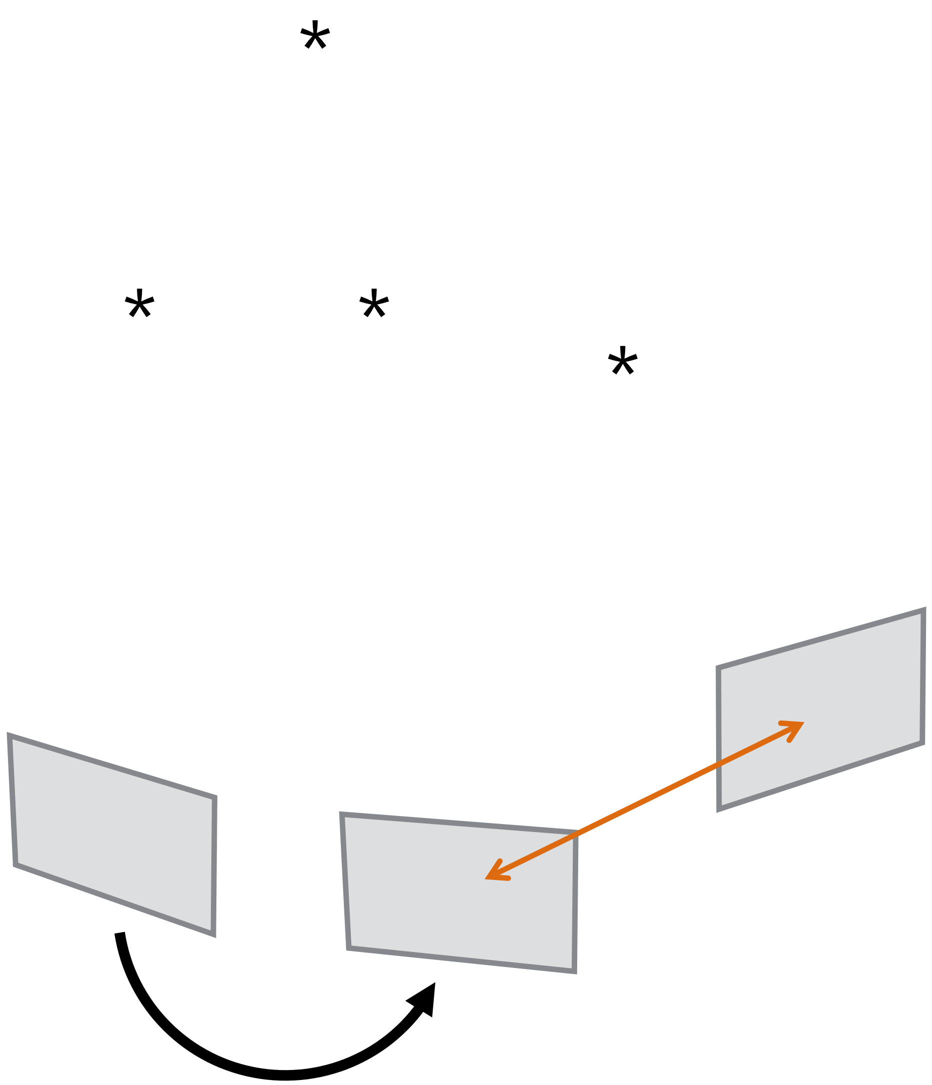
Sequential Structure-from-Motion



Initialize motion from two views

Initialize structure from two views

Sequential Structure-from-Motion



$$\mathbf{R}, \mathbf{t} \text{ with } ||\mathbf{t}|| = 1$$

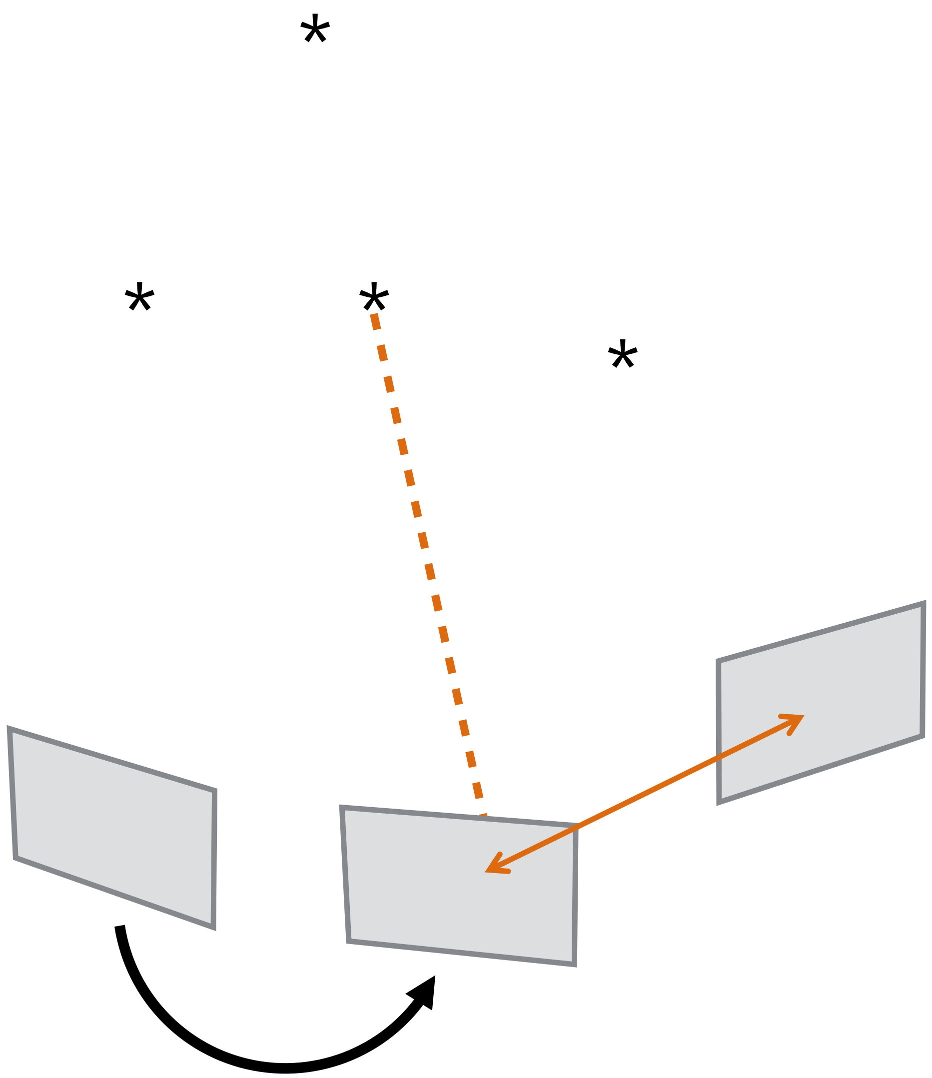
Match features

Initialize motion from two views

Initialize structure from two views

Extend motion

Sequential Structure-from-Motion



$\mathrm{R}, \mathbf{t}$ with $||\mathbf{t}|| = 1$

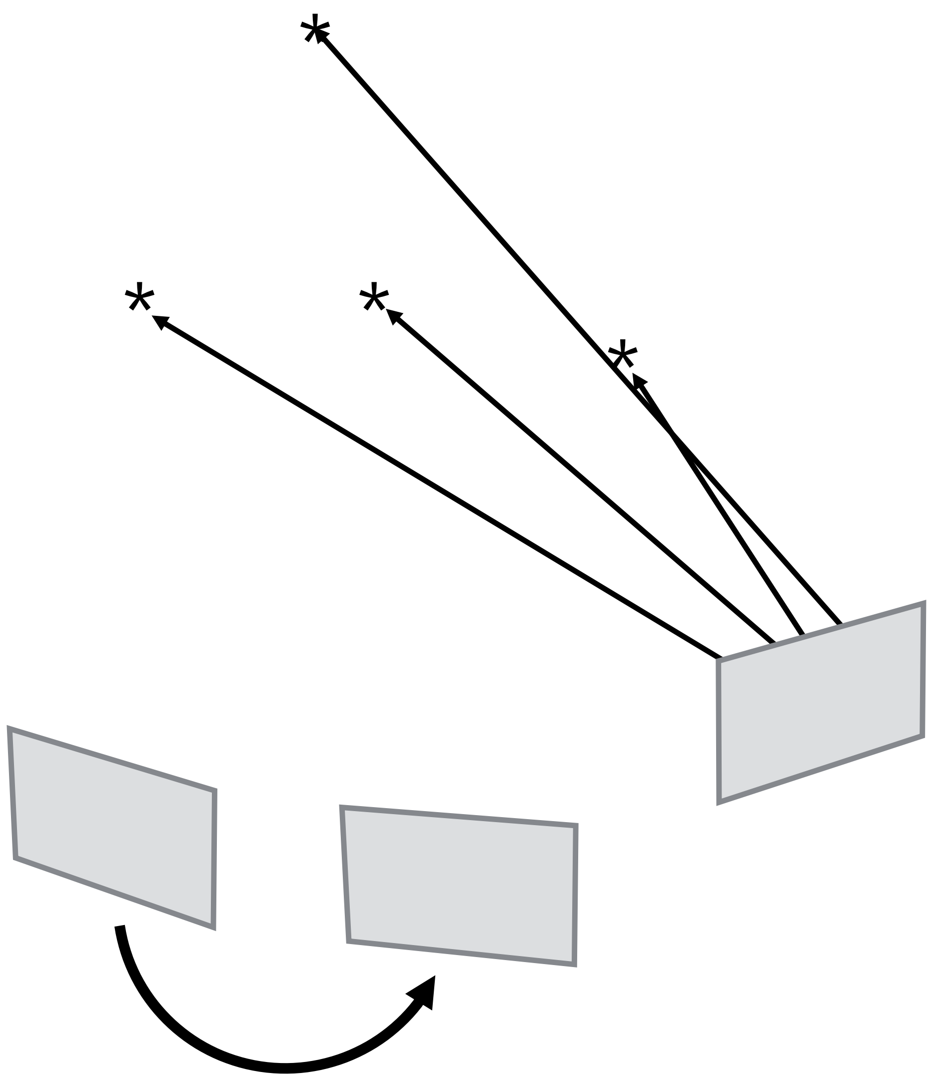
Transfer matches to 3D

Initialize motion from two views

Initialize structure from two views

Extend motion

Sequential Structure-from-Motion



$\mathrm{R}, \mathbf{t}$ with $||\mathbf{t}|| = 1$

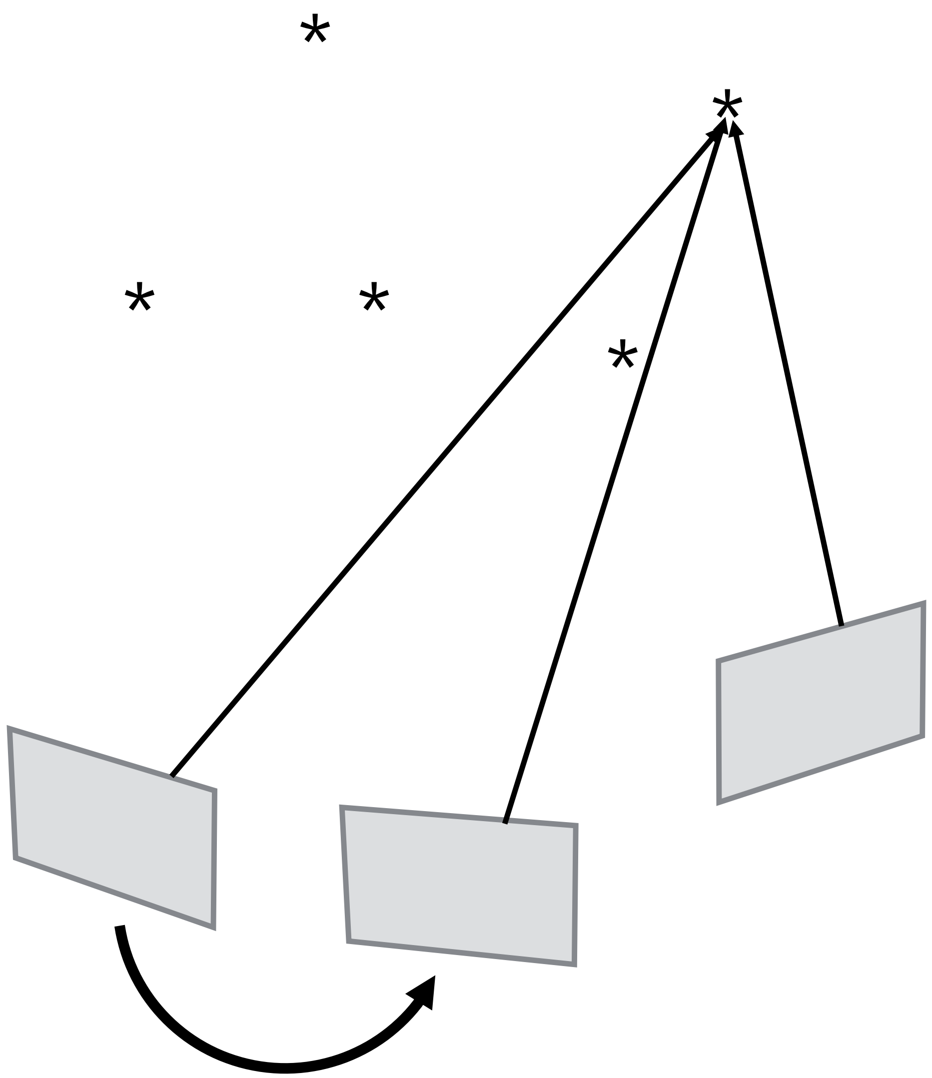
Camera pose for third camera

Initialize motion from two views

Initialize structure from two views

Extend motion

Sequential Structure-from-Motion



$\mathrm{R}, \mathbf{t} \text{ with } ||\mathbf{t}|| = 1$

Initialize motion from two views

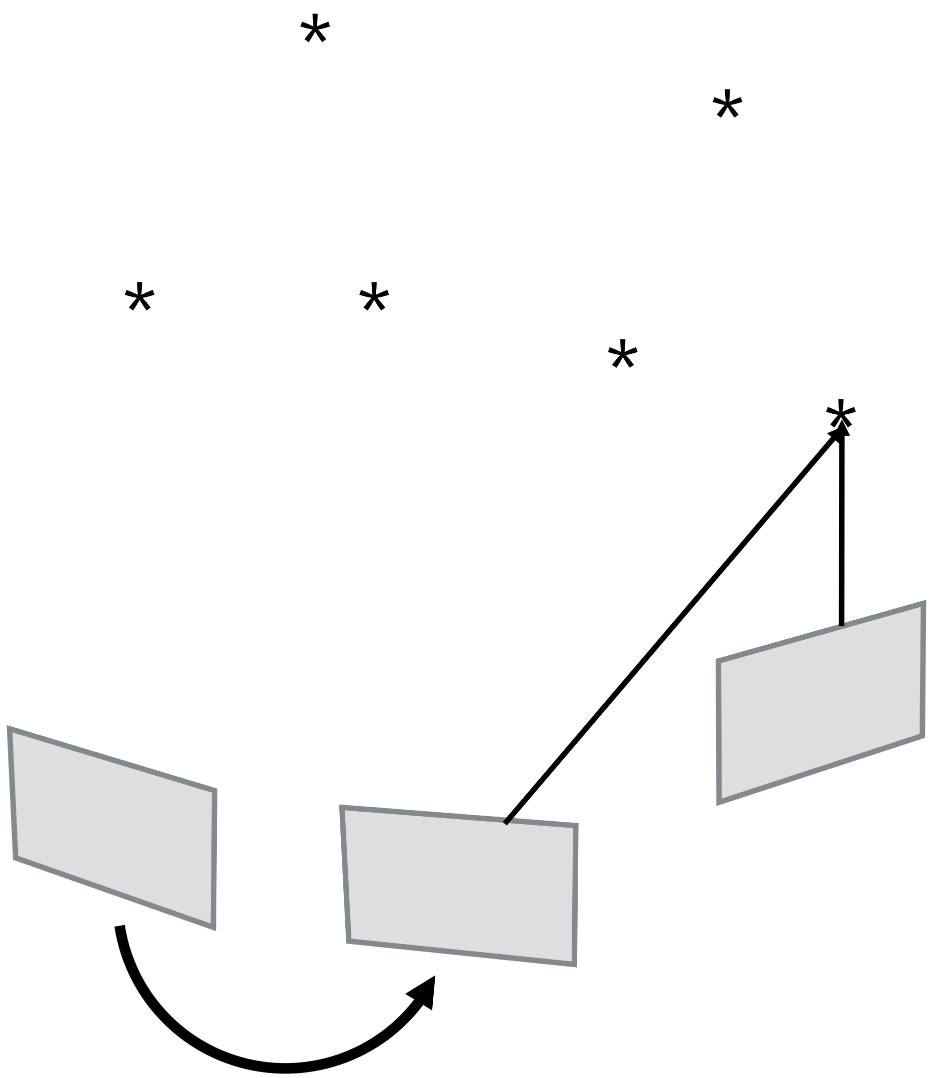
Initialize structure from two views

Extend motion

Extend structure

Triangulate points

Sequential Structure-from-Motion



$\mathbf{R}, \mathbf{t}$ with $||\mathbf{t}|| = 1$

Triangulate points

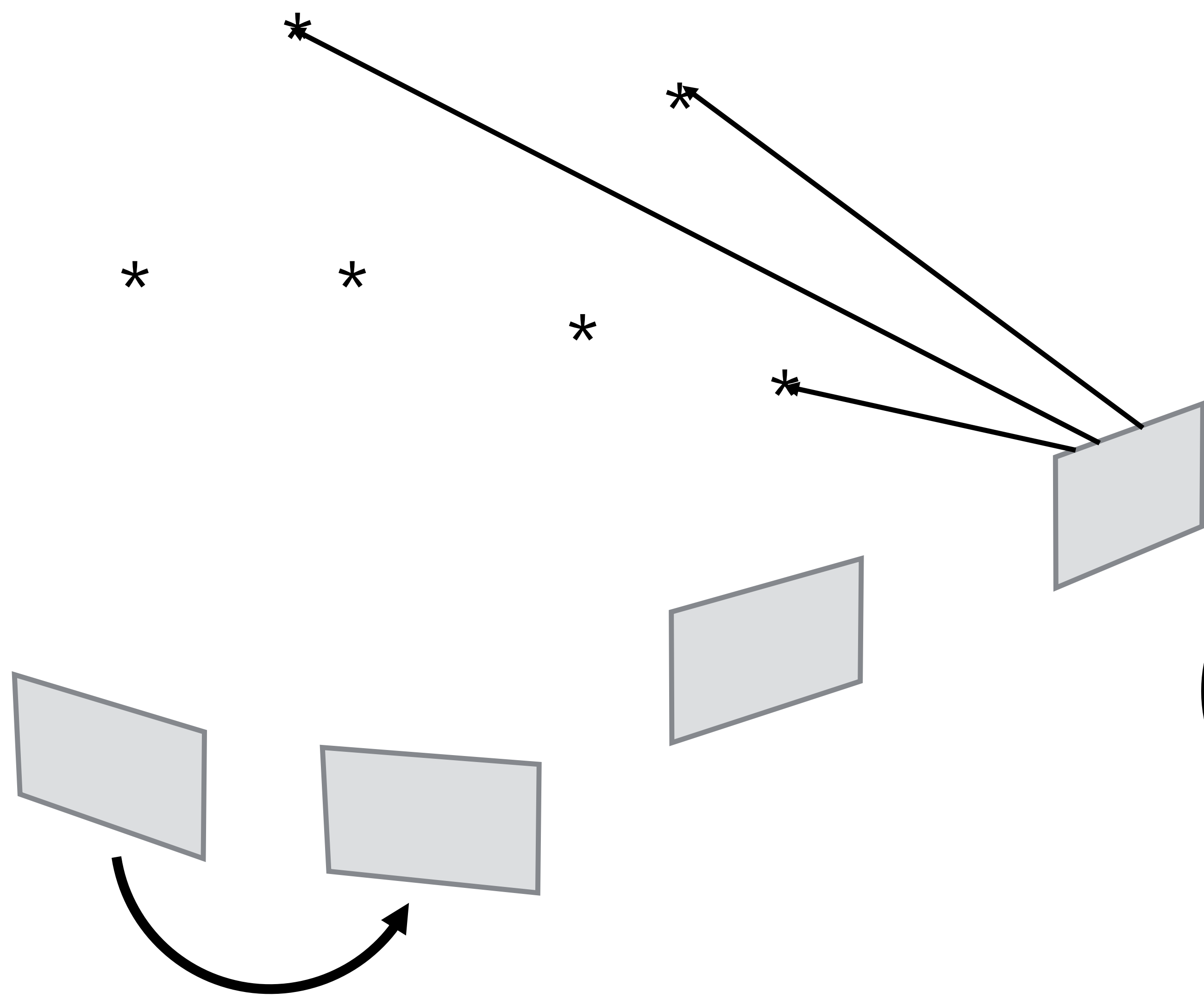
Initialize motion from two views

Initialize structure from two views

Extend motion

Extend structure

Sequential Structure-from-Motion



$$\mathrm{R}, \mathbf{t} \text{ with } ||\mathbf{t}|| = 1$$

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

Initialize motion from two views

Initialize structure from two views

Extend motion

Extend structure

Today

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

Initialize motion from two views

Initialize structure from two views

Extend motion

Extend structure

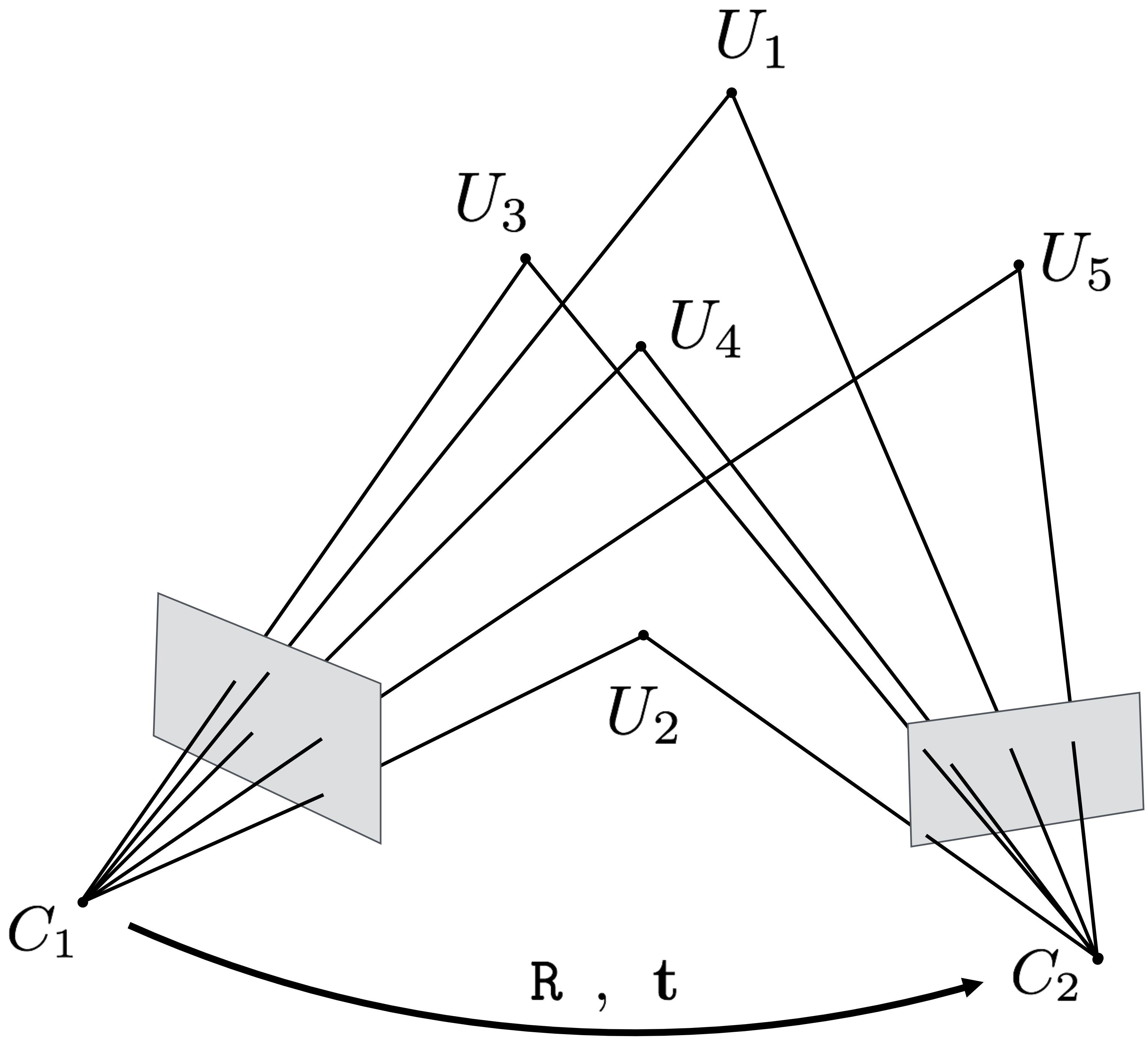
Cross Product as Matrix Multiplication

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

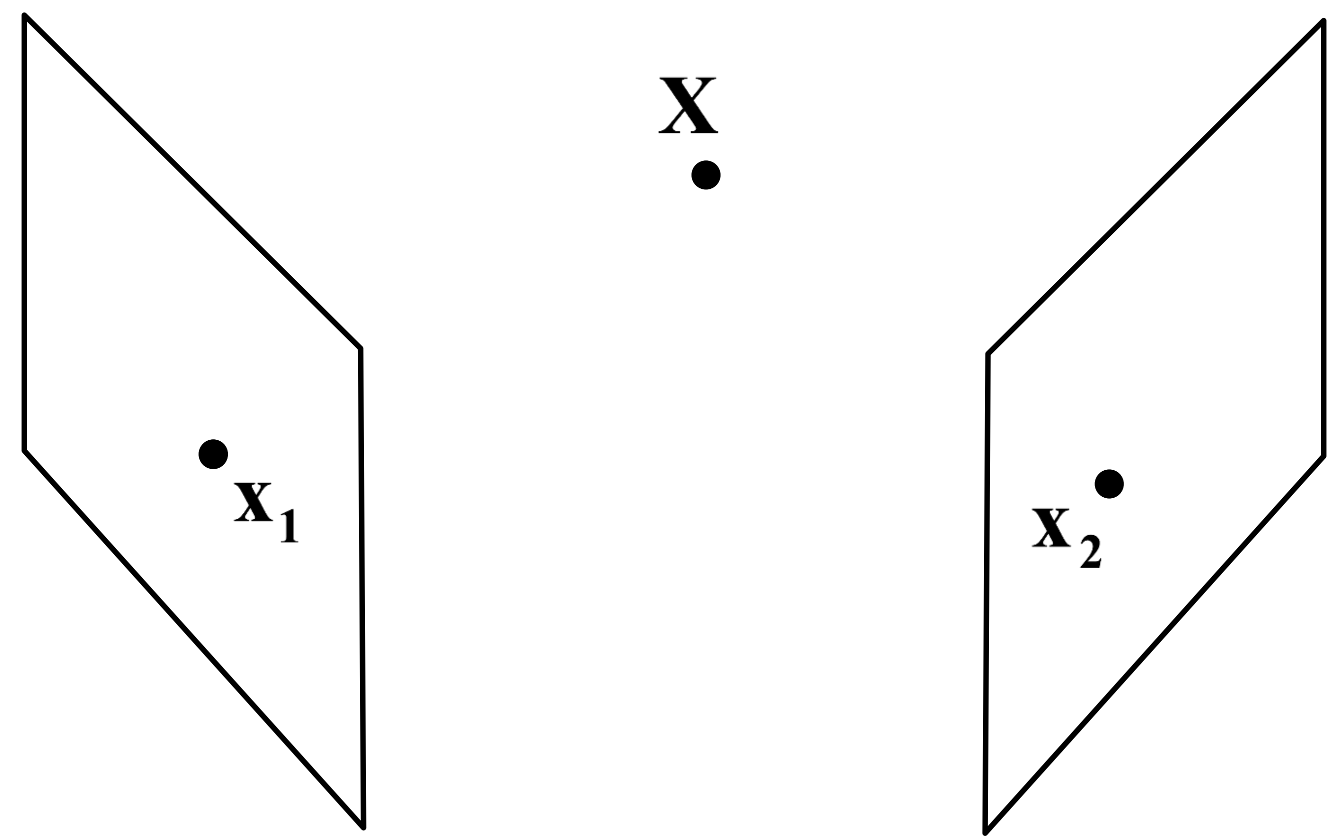
Cross Product as Matrix Multiplication

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

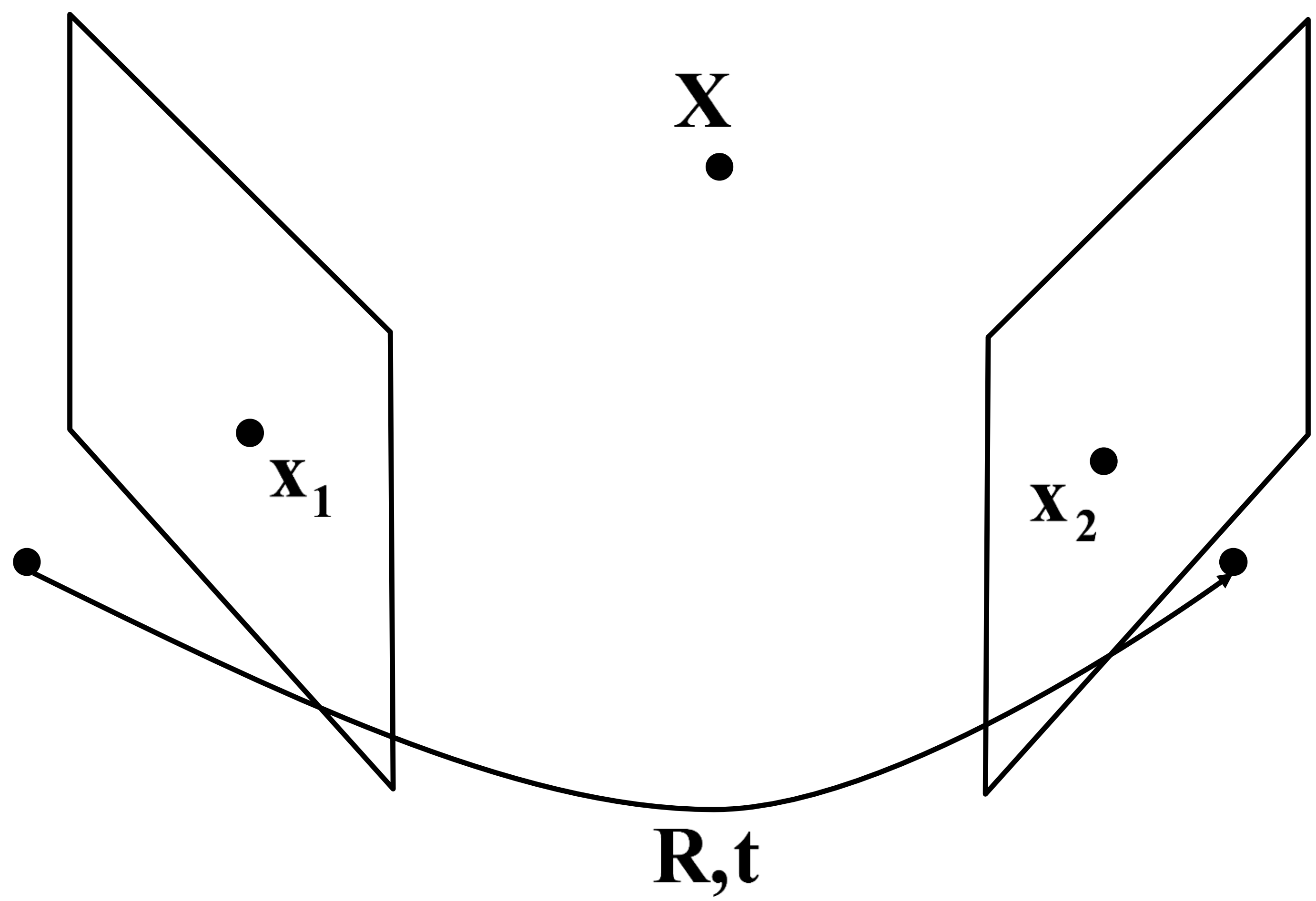
Relative Pose Estimation



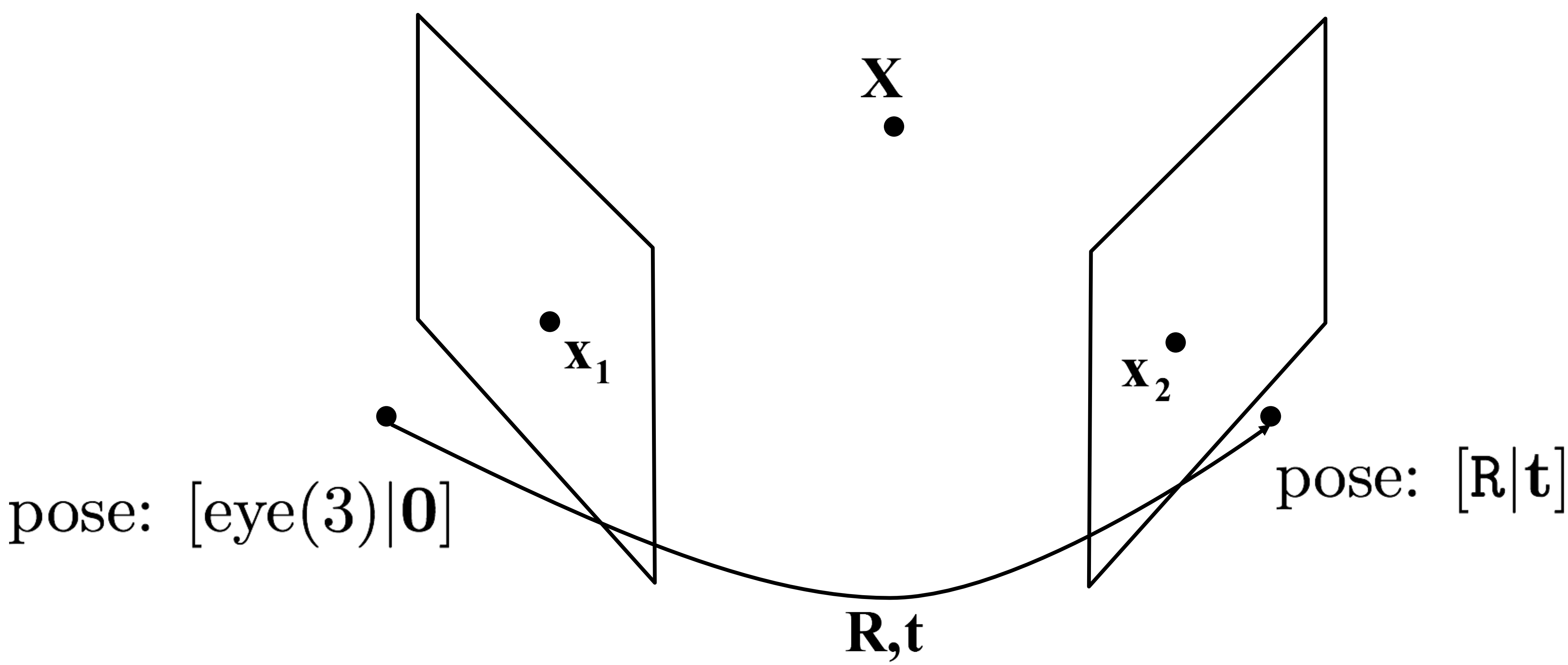
Epipolar Geometry



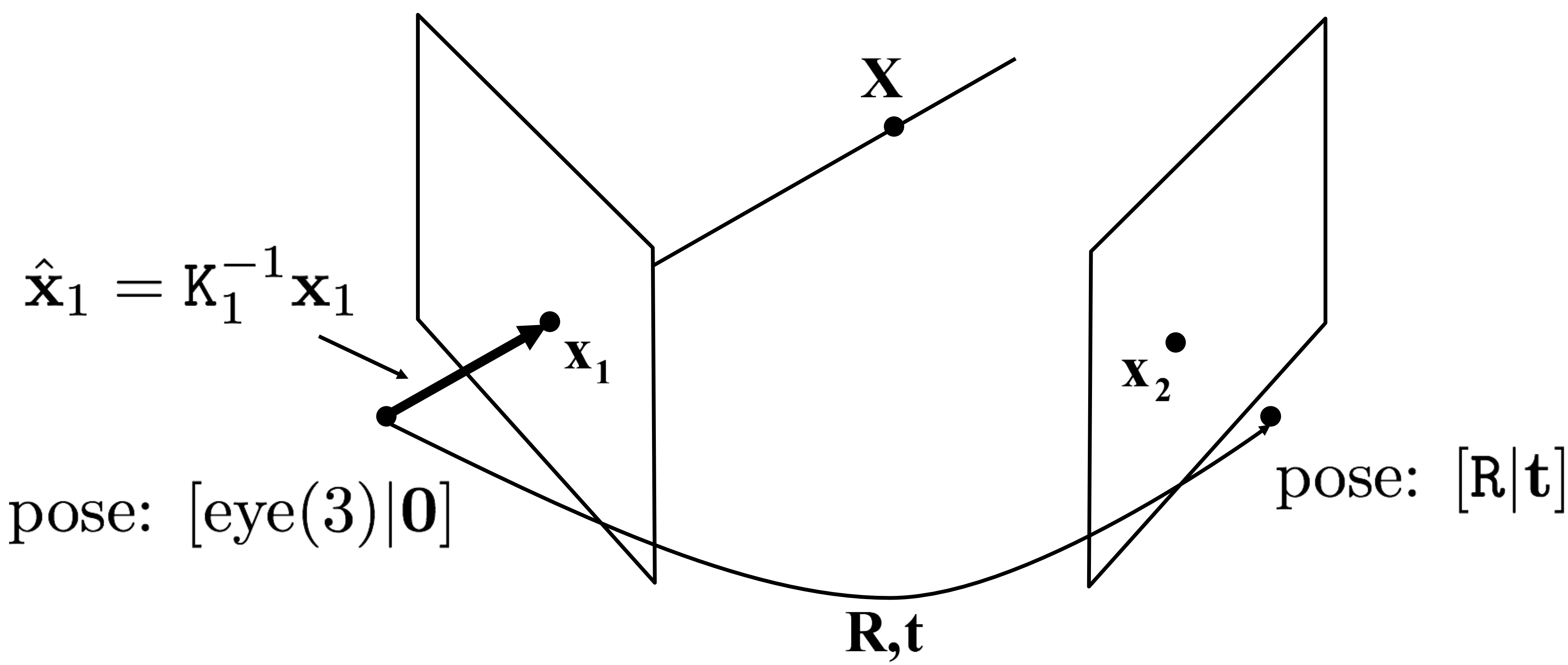
Epipolar Geometry



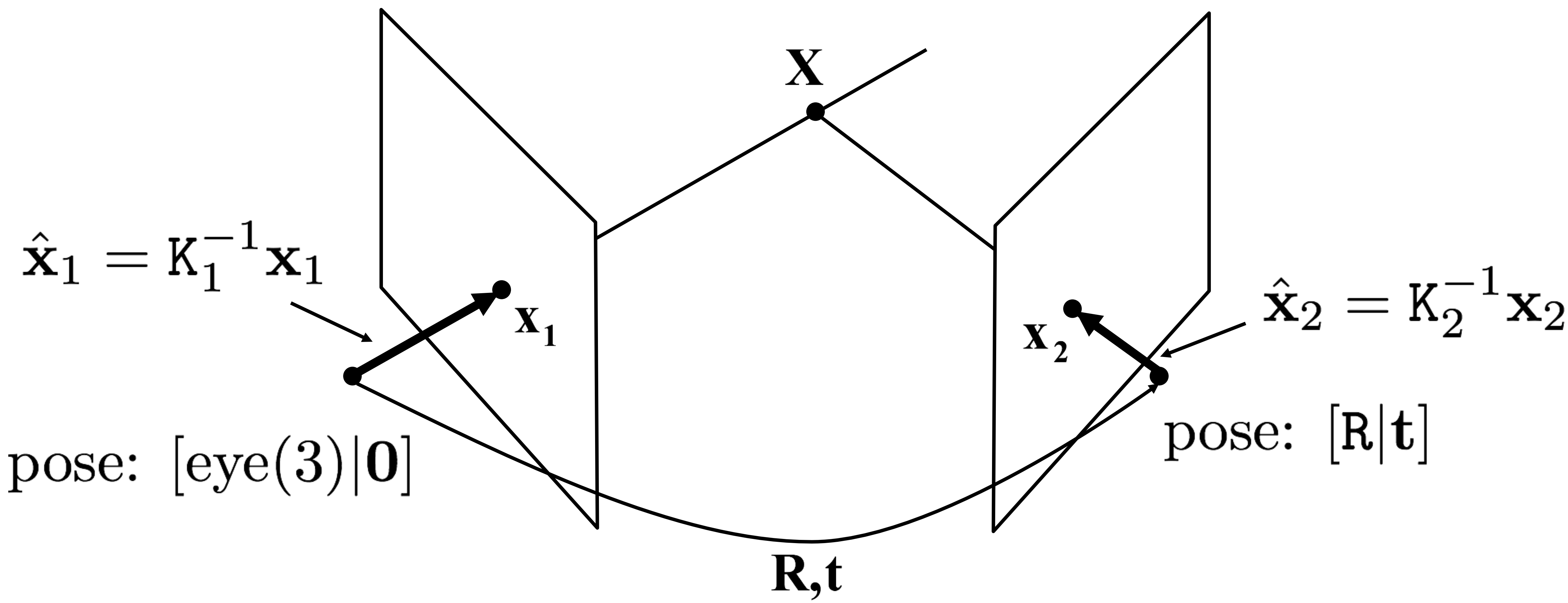
Epipolar Geometry



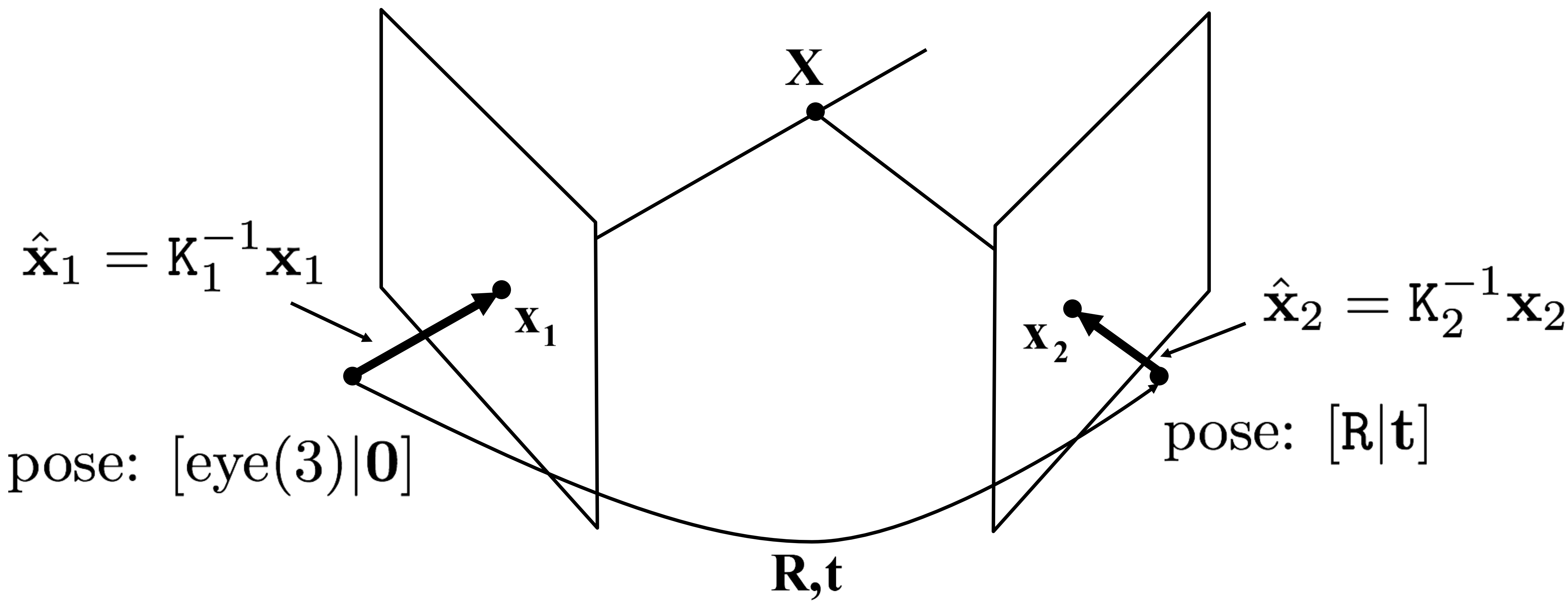
Epipolar Geometry



Epipolar Geometry

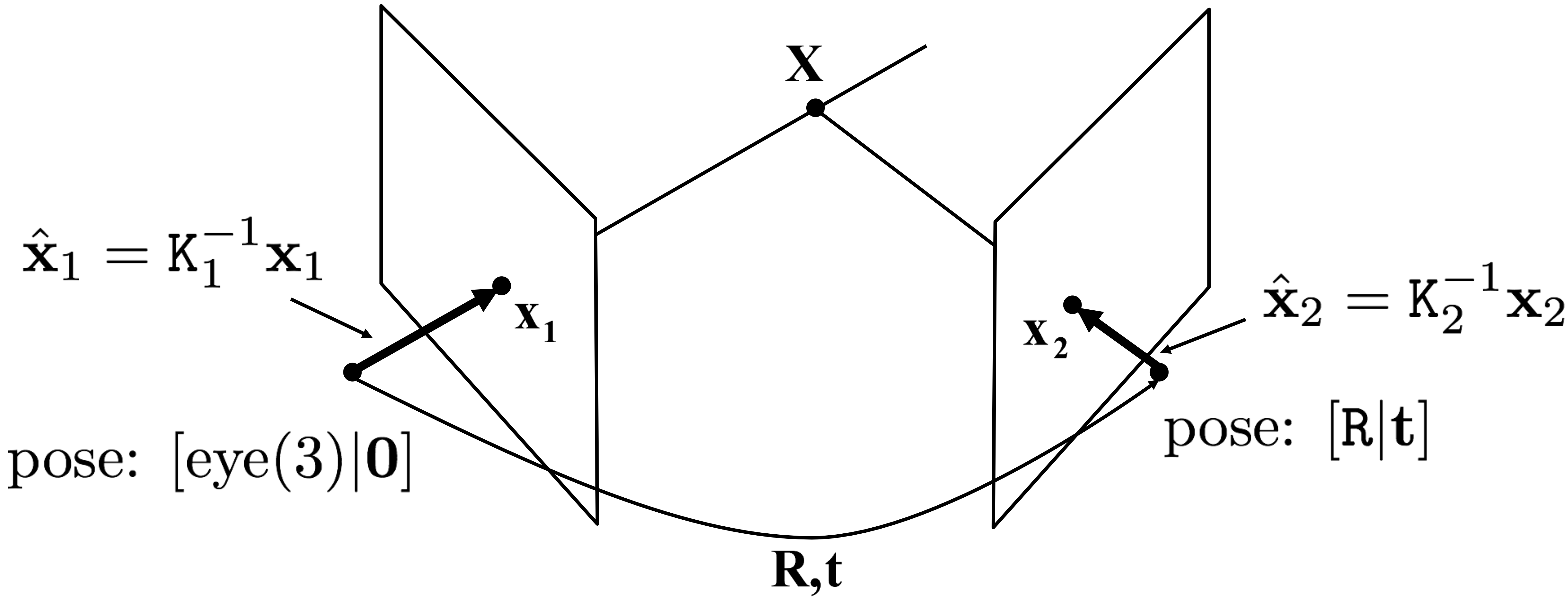


Epipolar Geometry



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

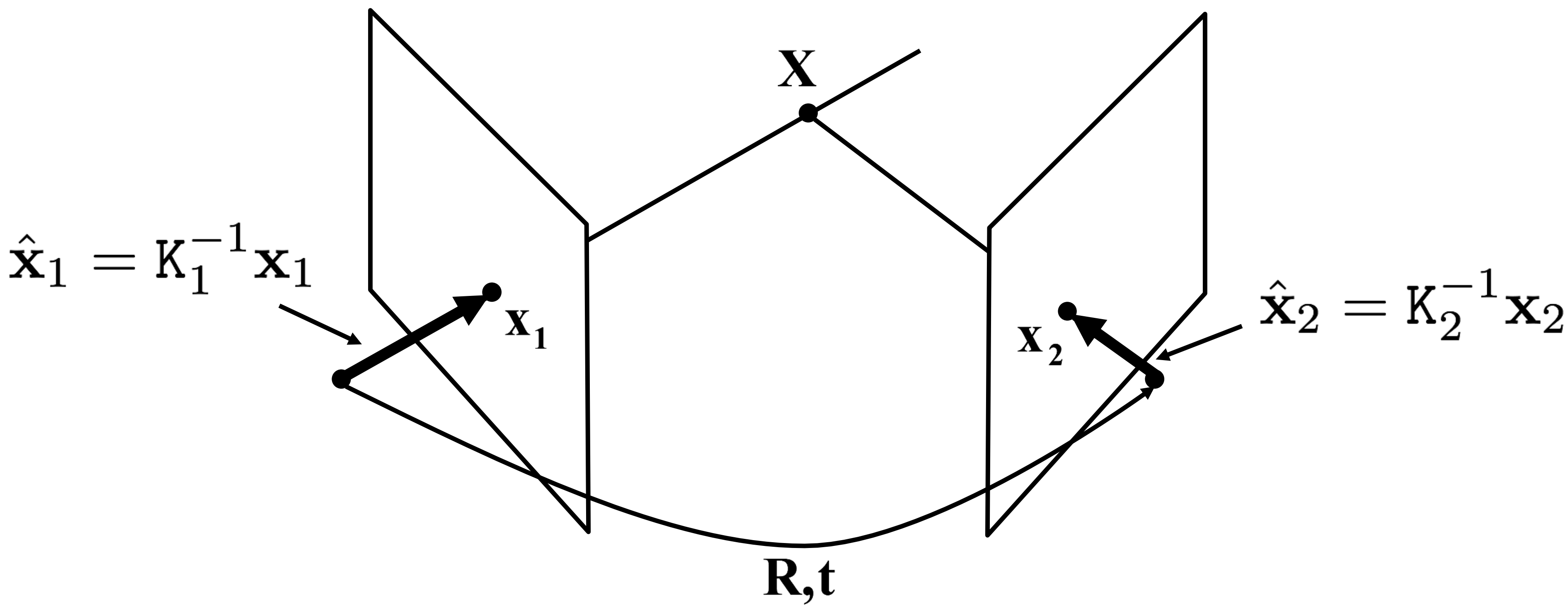
Epipolar Geometry



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

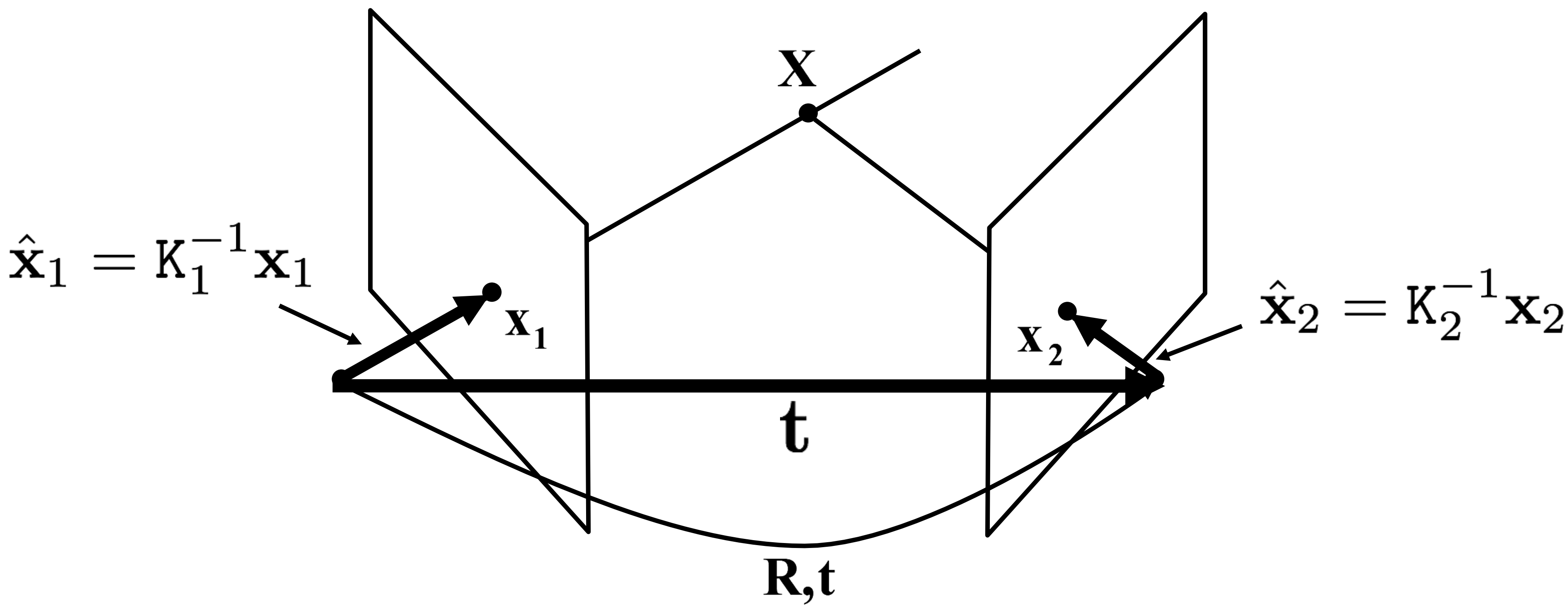
unknown

Epipolar Geometry



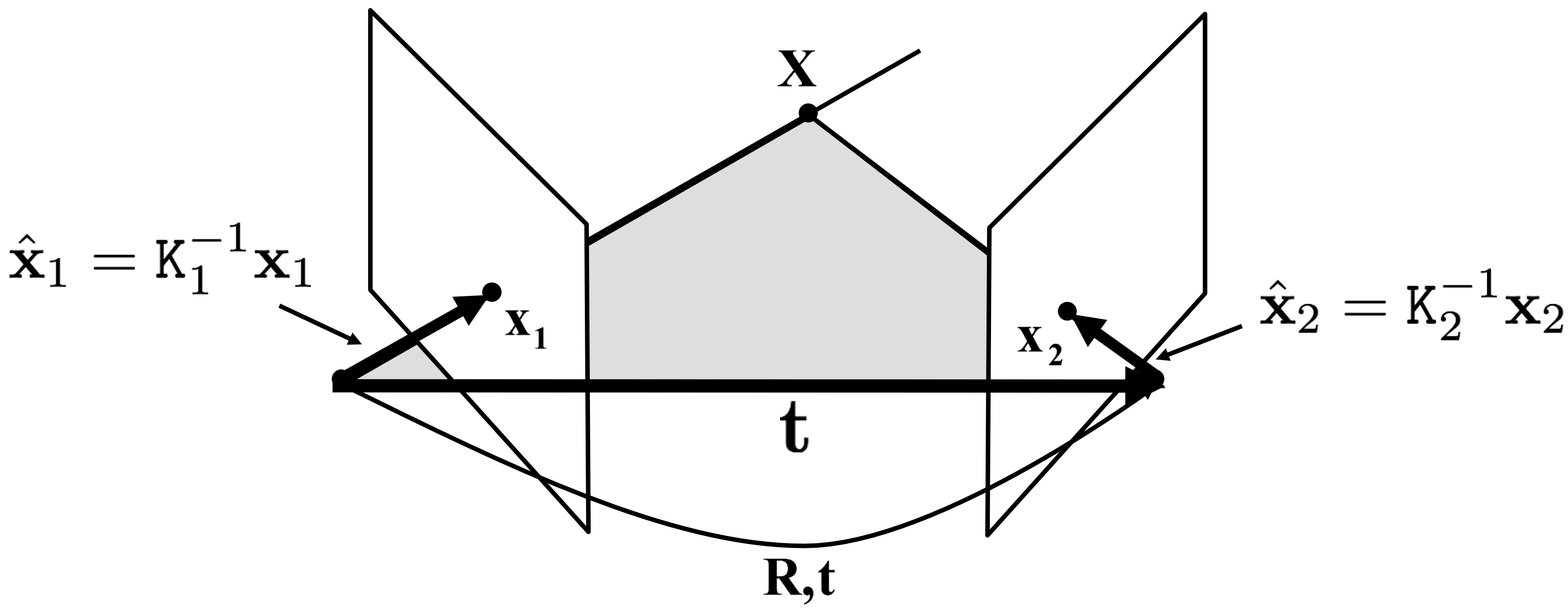
$$\lambda_2 \hat{\mathbf{x}}_2 = \mathtt{R}\lambda_1 \hat{\mathbf{x}}_1 + \mathbf{t}$$

Epipolar Geometry



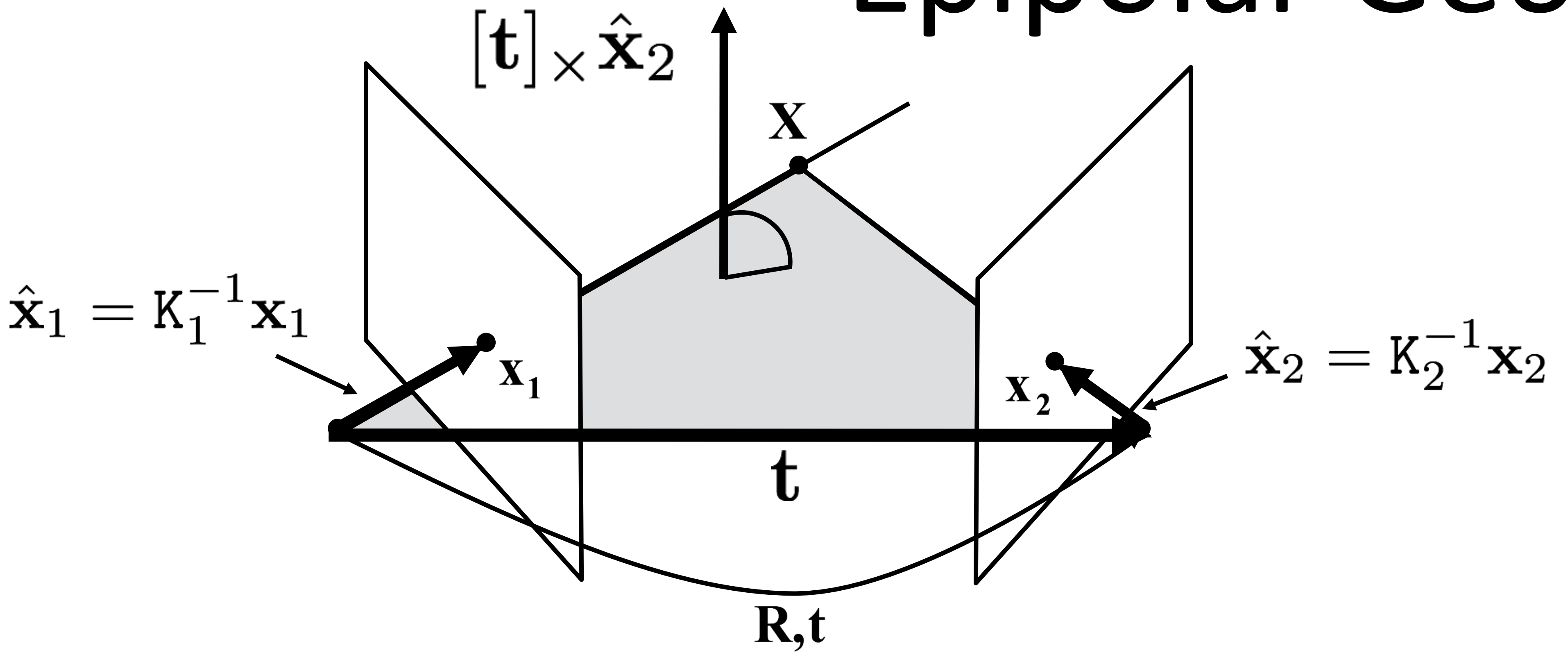
$$\lambda_2 \hat{\mathbf{x}}_2 = \mathtt{R}\lambda_1 \hat{\mathbf{x}}_1 + \mathbf{t}$$

Epipolar Geometry



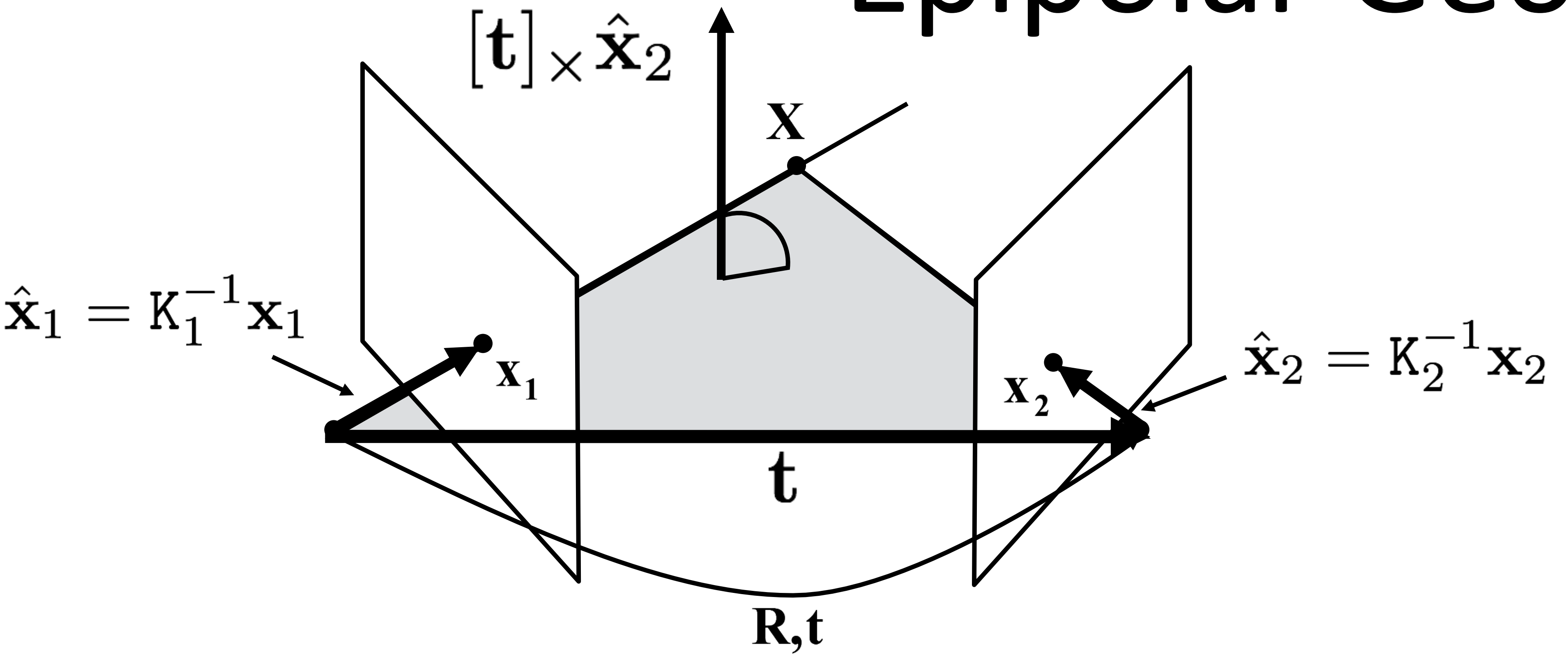
$$\lambda_2 \hat{\mathbf{x}}_2 = \mathtt{R}\lambda_1 \hat{\mathbf{x}}_1 + \mathbf{t}$$

Epipolar Geometry



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

Epipolar Geometry

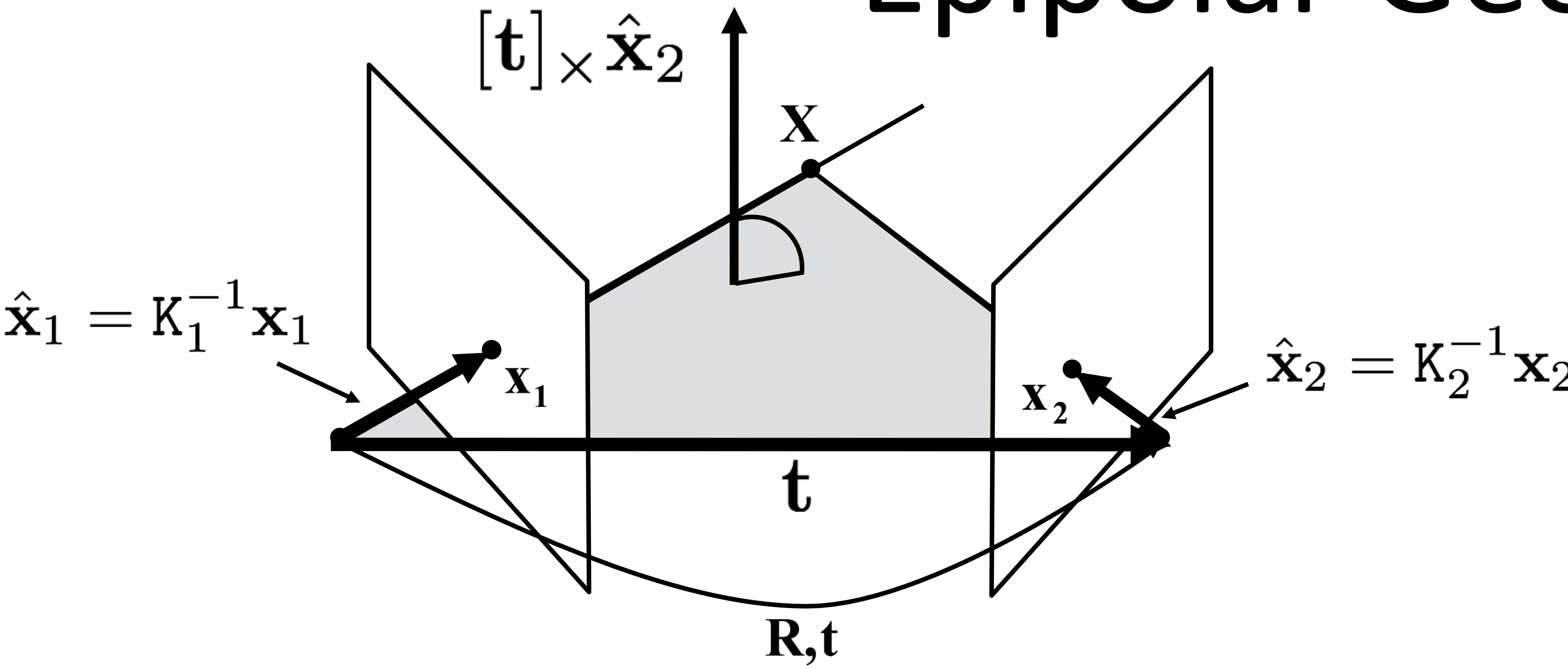


$$\lambda_2 \hat{\mathbf{x}}_2 = \mathtt{R}\lambda_1 \hat{\mathbf{x}}_1 + \mathbf{t}$$

0

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

Epipolar Geometry

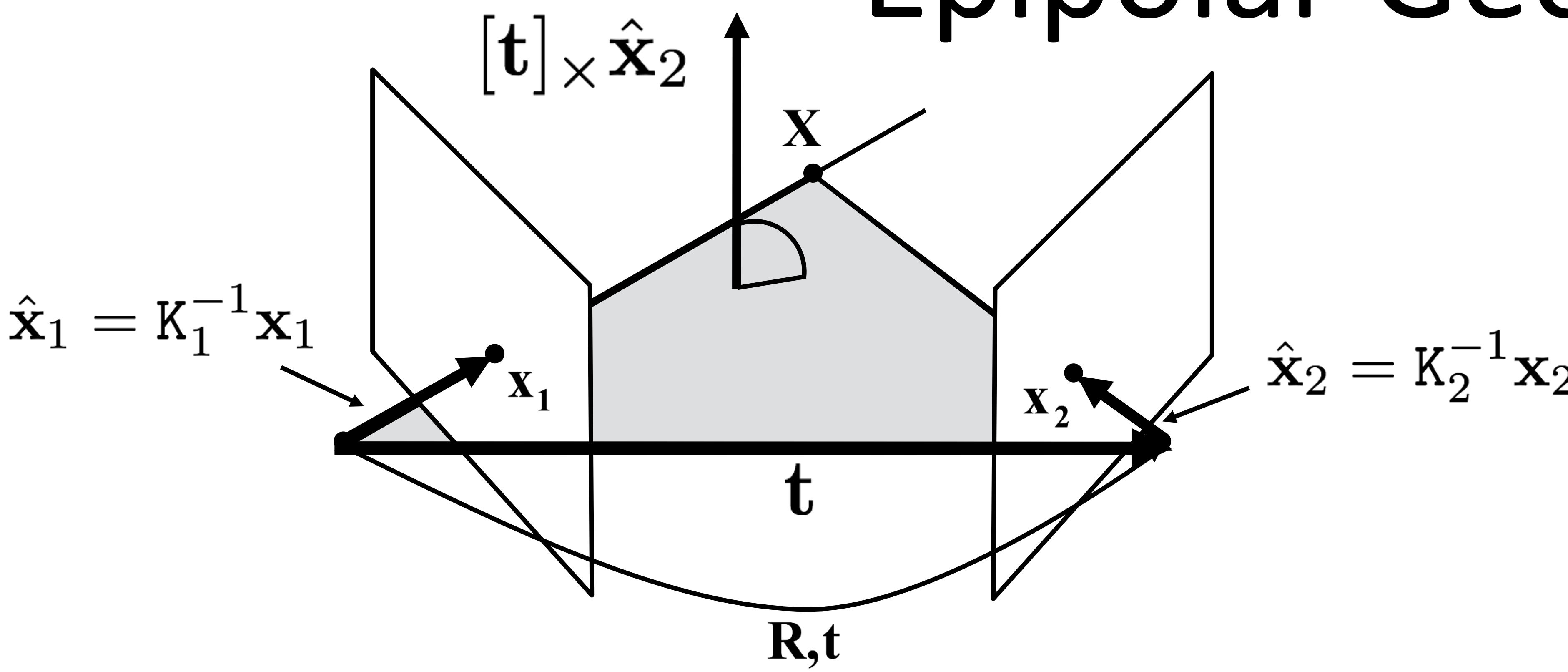


$$\lambda_2 \hat{\mathbf{x}}_2 = \mathtt{R}\lambda_1 \hat{\mathbf{x}}_1 + \mathbf{t}$$

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

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

Epipolar Geometry



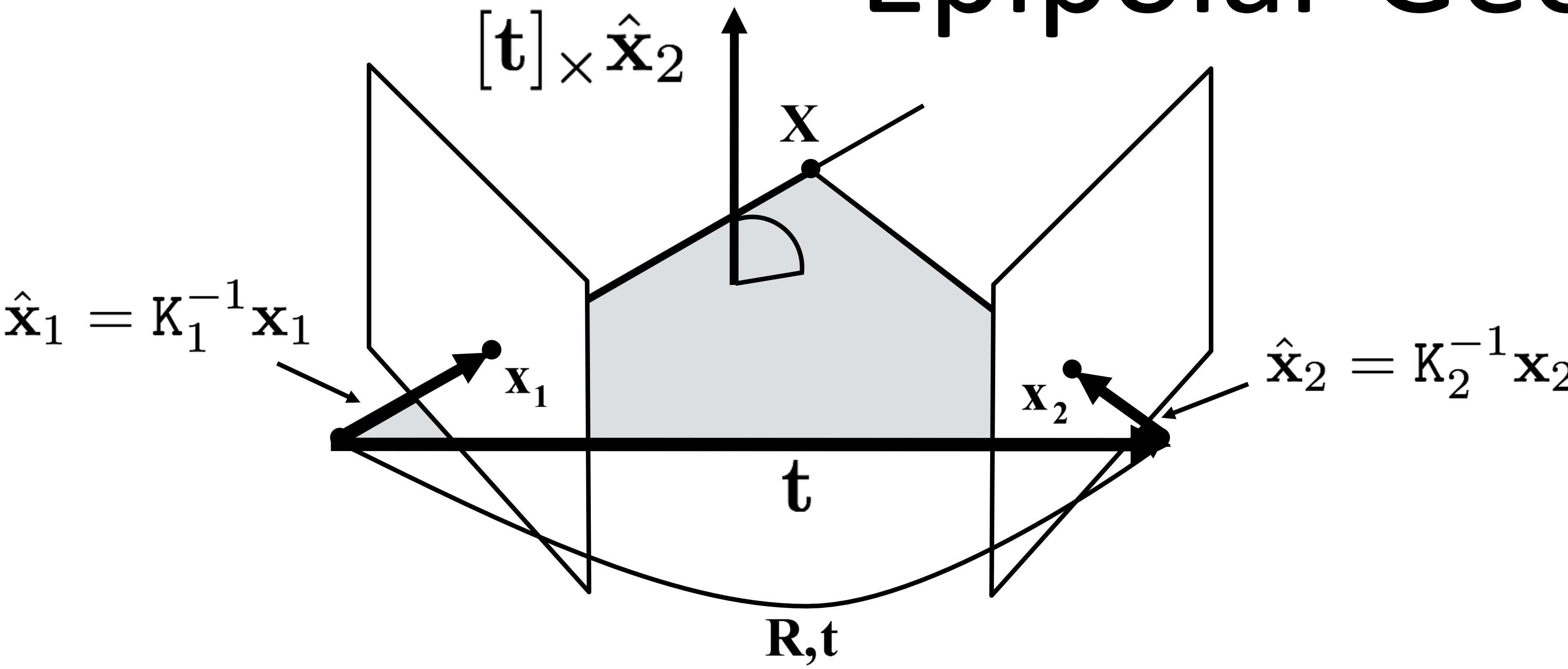
$$\lambda_2 \hat{\mathbf{x}}_2 = \mathtt{R}\lambda_1 \hat{\mathbf{x}}_1 + \mathbf{t}$$

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

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

scalar!

Epipolar Geometry



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

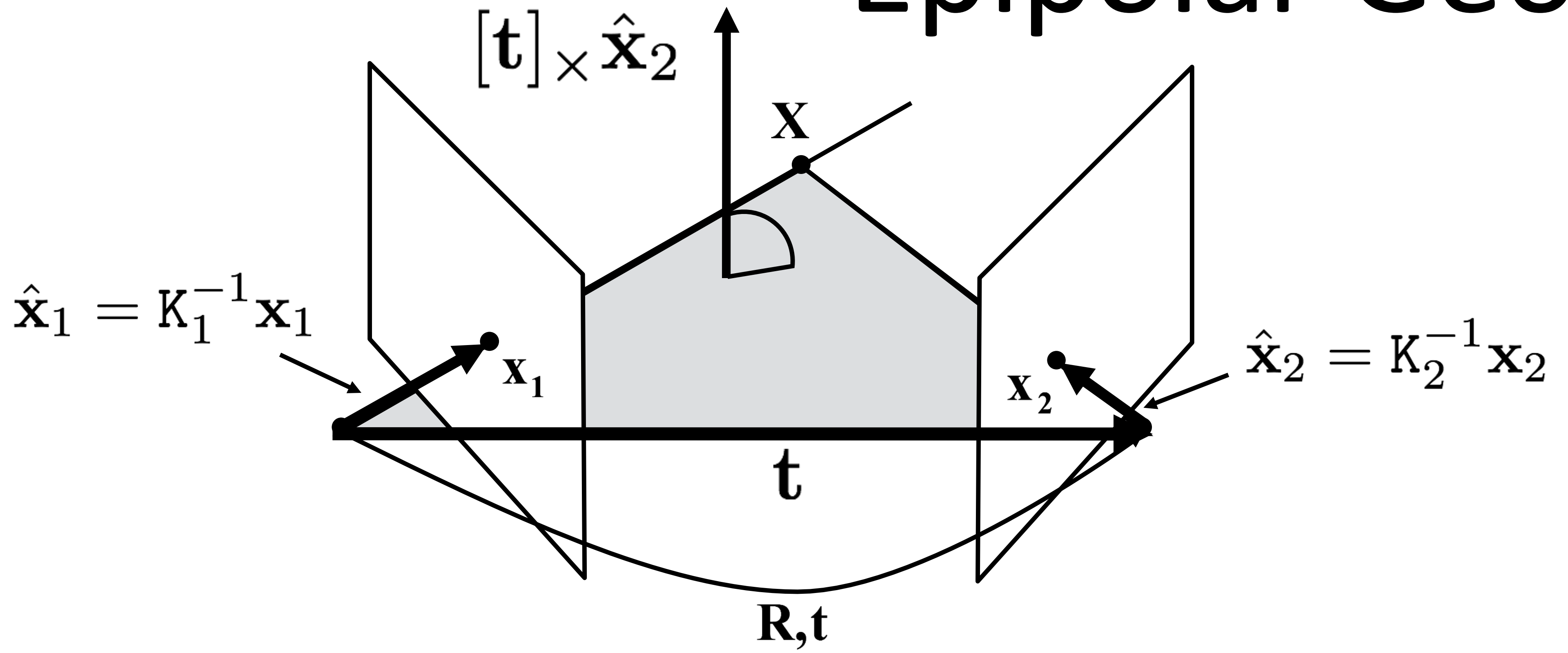
$$\Rightarrow \quad [\mathbf{t}]_\times \lambda_2 \hat{\mathbf{x}}_2 = [\mathbf{t}]_\times \mathtt{R}\lambda_1 \hat{\mathbf{x}}_1 + [\mathbf{t}]_\times \mathbf{t}$$

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

scalar!

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

Epipolar Geometry



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

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

$$\Rightarrow \quad \hat{\mathbf{x}}_2^T [\mathbf{t}]_\times \lambda_2 \hat{\mathbf{x}}_2 = \hat{\mathbf{x}}_2^T [\mathbf{t}]_\times \mathtt{R}\lambda_1 \hat{\mathbf{x}}_1$$ scalar!

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

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

The Essential Matrix

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

The Essential Matrix

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

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

- Essential matrix $\mathtt{E}$

The Essential Matrix

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

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

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

The Essential Matrix

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

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

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

The Essential Matrix

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

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

- Essential matrix $\mathtt{E}$
- $\mathtt{E}$ is 3x3 matrix, has 5 DoF (degrees-of-freedom)
- $\mathtt{E}$ has two equal singular values, third singular value is 0
- $\mathtt{E}$ has rank 2

The Fundamental Matrix

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

The Fundamental Matrix

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

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

The Fundamental Matrix

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

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

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

- Fundamental Matrix $\mathtt{F}$

The Fundamental Matrix

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

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

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

- Fundamental Matrix $\mathtt{F}$
- $\mathtt{F}$ has 7 DoF
- $\mathtt{F}$ has rank 2

The Fundamental Matrix

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

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

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

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

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

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

8-Point Linear Solver for $\mathtt{F}$

Objective

Given $n \geq 8$ image point correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{x}'_i\}$, determine the fundamental matrix $\mathtt{F}$ such that $\mathbf{x}'^{\mathsf{T}}_i \mathtt{F} \mathbf{x}_i = 0$.

Hartley and Zisserman. *Multiple View Geometry in Computer Vision*, 2nd edition, Cambridge University Press, 2004.

8-Point Linear Solver for $\mathtt{F}$

Objective

Given $n \geq 8$ image point correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{x}'_i\}$, determine the fundamental matrix $\mathtt{F}$ such that $\mathbf{x}'^{\mathsf{T}}_i \mathtt{F} \mathbf{x}_i = 0$.

Algorithm

(i) **Normalization:** Transform the image coordinates according to $\hat{\mathbf{x}}_i = \mathtt{T}\mathbf{x}_i$ and $\hat{\mathbf{x}}'_i = \mathtt{T}'\mathbf{x}'_i$, where $\mathtt{T}$ and $\mathtt{T}'$ are normalizing transformations consisting of a translation and scaling.

Hartley and Zisserman. *Multiple View Geometry in Computer Vision*, 2nd edition, Cambridge University Press, 2004.

8-Point Linear Solver for $\mathtt{F}$

Objective

Given $n \geq 8$ image point correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{x}'_i\}$, determine the fundamental matrix $\mathtt{F}$ such that $\mathbf{x}'^{\mathsf{T}}_i \mathtt{F} \mathbf{x}_i = 0$.

Algorithm

(i) **Normalization:** Transform the image coordinates according to $\hat{\mathbf{x}}_i = \mathtt{T}\mathbf{x}_i$ and $\hat{\mathbf{x}}'_i = \mathtt{T}'\mathbf{x}'_i$, where $\mathtt{T}$ and $\mathtt{T}'$ are normalizing transformations consisting of a translation and scaling.

(ii) Find the fundamental matrix $\hat{\mathtt{F}}'$ corresponding to the matches $\hat{\mathbf{x}}_i \leftrightarrow \hat{\mathbf{x}}'_i$ by

Hartley and Zisserman. *Multiple View Geometry in Computer Vision*, 2nd edition, Cambridge University Press, 2004.

8-Point Linear Solver for $\mathtt{F}$

Objective

Given $n \geq 8$ image point correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{x}'_i\}$, determine the fundamental matrix $\mathtt{F}$ such that $\mathbf{x}'^{\mathsf{T}}_i \mathtt{F} \mathbf{x}_i = 0$.

Algorithm

(i) **Normalization:** Transform the image coordinates according to $\hat{\mathbf{x}}_i = \mathtt{T}\mathbf{x}_i$ and $\hat{\mathbf{x}}'_i = \mathtt{T}'\mathbf{x}'_i$, where $\mathtt{T}$ and $\mathtt{T}'$ are normalizing transformations consisting of a translation and scaling.

(ii) Find the fundamental matrix $\hat{\mathtt{F}}'$ corresponding to the matches $\hat{\mathbf{x}}_i \leftrightarrow \hat{\mathbf{x}}'_i$ by

  (a) **Linear solution:** Determine $\hat{\mathtt{F}}$ from the singular vector corresponding to the smallest singular value of $\hat{\mathtt{A}}$, where $\hat{\mathtt{A}}$ is composed from the matches $\hat{\mathbf{x}}_i \leftrightarrow \hat{\mathbf{x}}'_i$ as defined in (11.3).

Hartley and Zisserman. *Multiple View Geometry in Computer Vision*, 2nd edition, Cambridge University Press, 2004.

8-Point Linear Solver for $\mathtt{F}$

<u>Objective</u>

Given $n \geq 8$ image point correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{x}'_i\}$, determine the fundamental matrix $\mathtt{F}$ such that $\mathbf{x}_i'^{\mathsf{T}} \mathtt{F} \mathbf{x}_i = 0$.

<u>Algorithm</u>

(i) **Normalization:** Transform the image coordinates according to $\hat{\mathbf{x}}_i = \mathtt{T}\mathbf{x}_i$ and $\hat{\mathbf{x}}'_i = \mathtt{T}'\mathbf{x}'_i$, where $\mathtt{T}$ and $\mathtt{T}'$ are normalizing transformations consisting of a translation and scaling.

(ii) Find the fundamental matrix $\hat{\mathtt{F}}'$ corresponding to the matches $\hat{\mathbf{x}}_i \leftrightarrow \hat{\mathbf{x}}'_i$ by

(a) **Linear solution:** Determine $\hat{\mathtt{F}}$ from the singular vector corresponding to the smallest singular value of $\hat{\mathtt{A}}$, where $\hat{\mathtt{A}}$ is composed from the matches $\hat{\mathbf{x}}_i \leftrightarrow \hat{\mathbf{x}}'_i$ as defined in (11.3).

(b) **Constraint enforcement:** Replace $\hat{\mathtt{F}}$ by $\hat{\mathtt{F}}'$ such that $\det \hat{\mathtt{F}}' = 0$ using the SVD (see section 11.1.1).

Hartley and Zisserman. *Multiple View Geometry in Computer Vision*, 2nd edition, Cambridge University Press, 2004.

8-Point Linear Solver for $\mathtt{F}$

Objective

Given $n \geq 8$ image point correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{x}'_i\}$, determine the fundamental matrix F such that $\mathbf{x}'^{\mathsf{T}}_i \mathtt{F} \mathbf{x}_i = 0$.

Algorithm

(i) **Normalization:** Transform the image coordinates according to $\hat{\mathbf{x}}_i = \mathtt{T}\mathbf{x}_i$ and $\hat{\mathbf{x}}'_i = \mathtt{T}'\mathbf{x}'_i$, where $\mathtt{T}$ and $\mathtt{T}'$ are normalizing transformations consisting of a translation and scaling.

(ii) Find the fundamental matrix $\hat{\mathtt{F}}'$ corresponding to the matches $\hat{\mathbf{x}}_i \leftrightarrow \hat{\mathbf{x}}'_i$ by

(a) **Linear solution:** Determine $\hat{\mathtt{F}}$ from the singular vector corresponding to the smallest singular value of $\hat{\mathtt{A}}$, where $\hat{\mathtt{A}}$ is composed from the matches $\hat{\mathbf{x}}_i \leftrightarrow \hat{\mathbf{x}}'_i$ as defined in (11.3).

rank 2 constraint!

(b) **Constraint enforcement:** Replace $\hat{\mathtt{F}}$ by $\hat{\mathtt{F}}'$ such that $\det \hat{\mathtt{F}}' = 0$ using the SVD (see section 11.1.1).

Hartley and Zisserman. *Multiple View Geometry in Computer Vision*, 2nd edition, Cambridge University Press, 2004.

8-Point Linear Solver for $\mathtt{F}$

<u>Objective</u>

Given $n \geq 8$ image point correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{x}'_i\}$, determine the fundamental matrix $\mathtt{F}$ such that $\mathbf{x}'^{\mathsf{T}}_i \mathtt{F} \mathbf{x}_i = 0$.

<u>Algorithm</u>

(i) **Normalization:** Transform the image coordinates according to $\hat{\mathbf{x}}_i = \mathtt{T}\mathbf{x}_i$ and $\hat{\mathbf{x}}'_i = \mathtt{T}'\mathbf{x}'_i$, where $\mathtt{T}$ and $\mathtt{T}'$ are normalizing transformations consisting of a translation and scaling.

(ii) Find the fundamental matrix $\hat{\mathtt{F}}'$ corresponding to the matches $\hat{\mathbf{x}}_i \leftrightarrow \hat{\mathbf{x}}'_i$ by

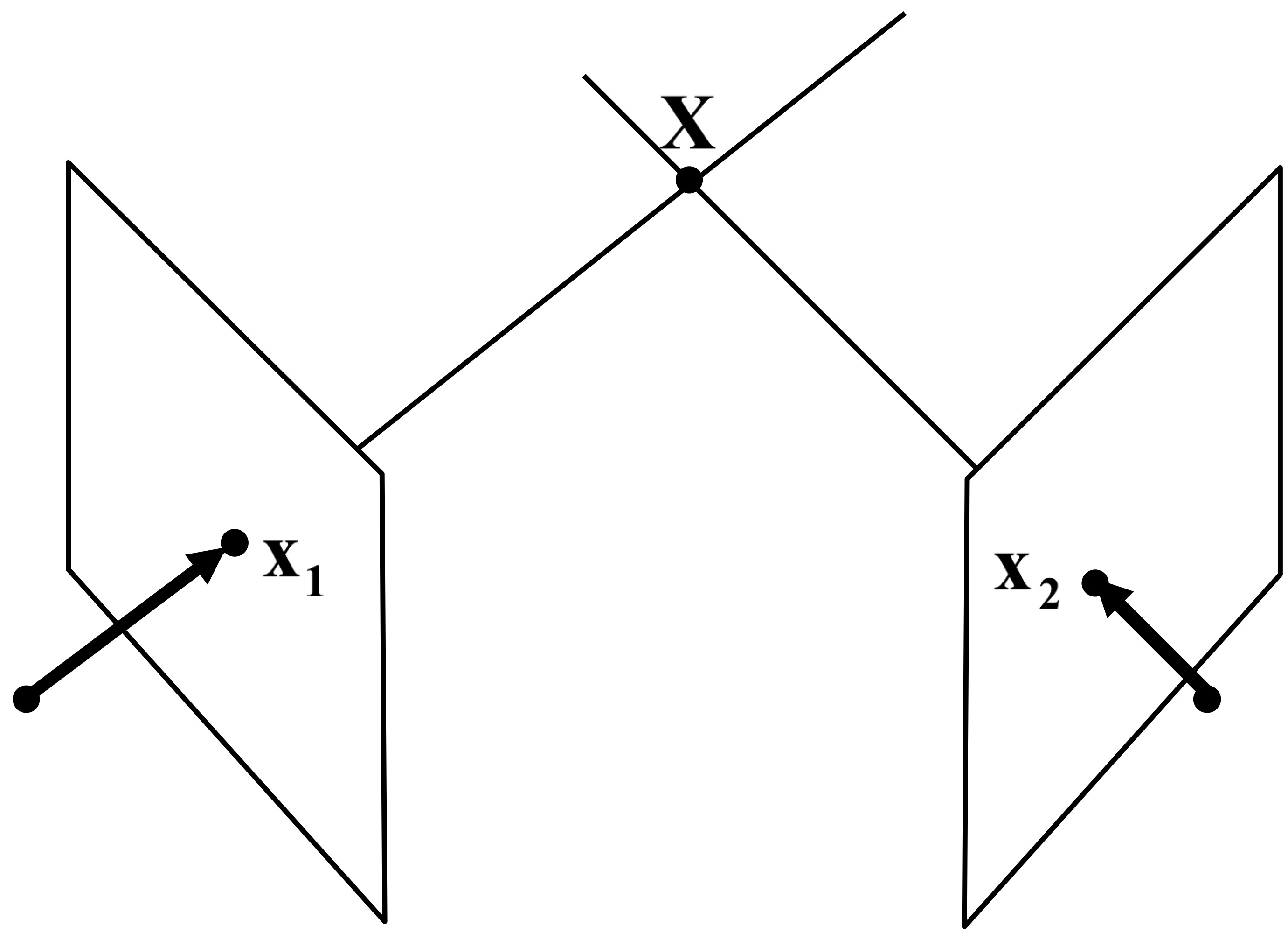
  (a) **Linear solution:** Determine $\hat{\mathtt{F}}$ from the singular vector corresponding to the smallest singular value of $\hat{\mathtt{A}}$, where $\hat{\mathtt{A}}$ is composed from the matches $\hat{\mathbf{x}}_i \leftrightarrow \hat{\mathbf{x}}'_i$ as defined in (11.3).

  (b) **Constraint enforcement:** Replace $\hat{\mathtt{F}}$ by $\hat{\mathtt{F}}'$ such that $\det \hat{\mathtt{F}}' = 0$ using the SVD (see section 11.1.1).

(iii) **Denormalization:** Set $\mathtt{F} = \mathtt{T}'^{\mathsf{T}} \hat{\mathtt{F}}' \mathtt{T}$. Matrix $\mathtt{F}$ is the fundamental matrix corresponding to the original data $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$.

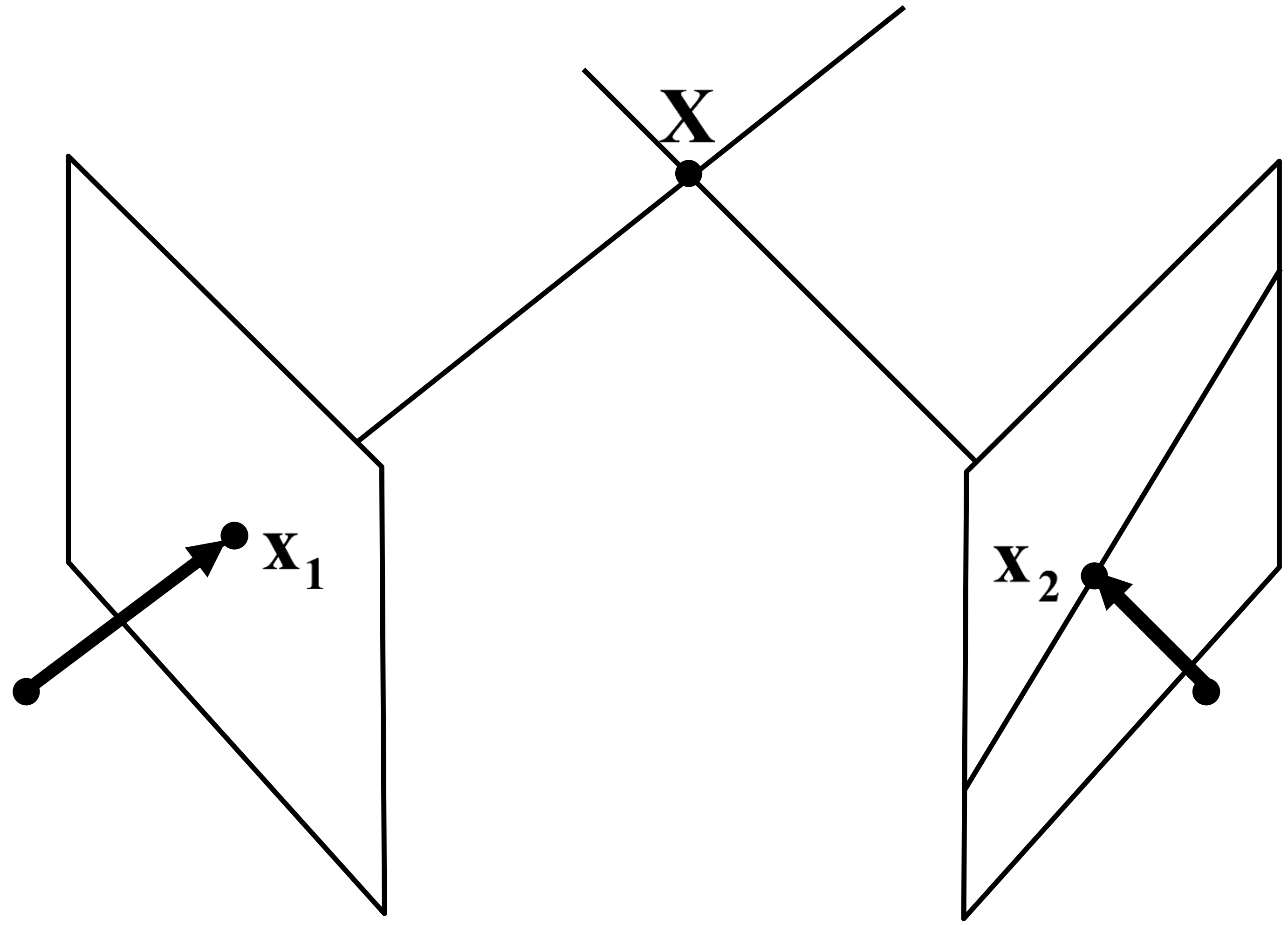
Hartley and Zisserman. *Multiple View Geometry in Computer Vision*, 2nd edition, Cambridge University Press, 2004.

Geometric Interpretation



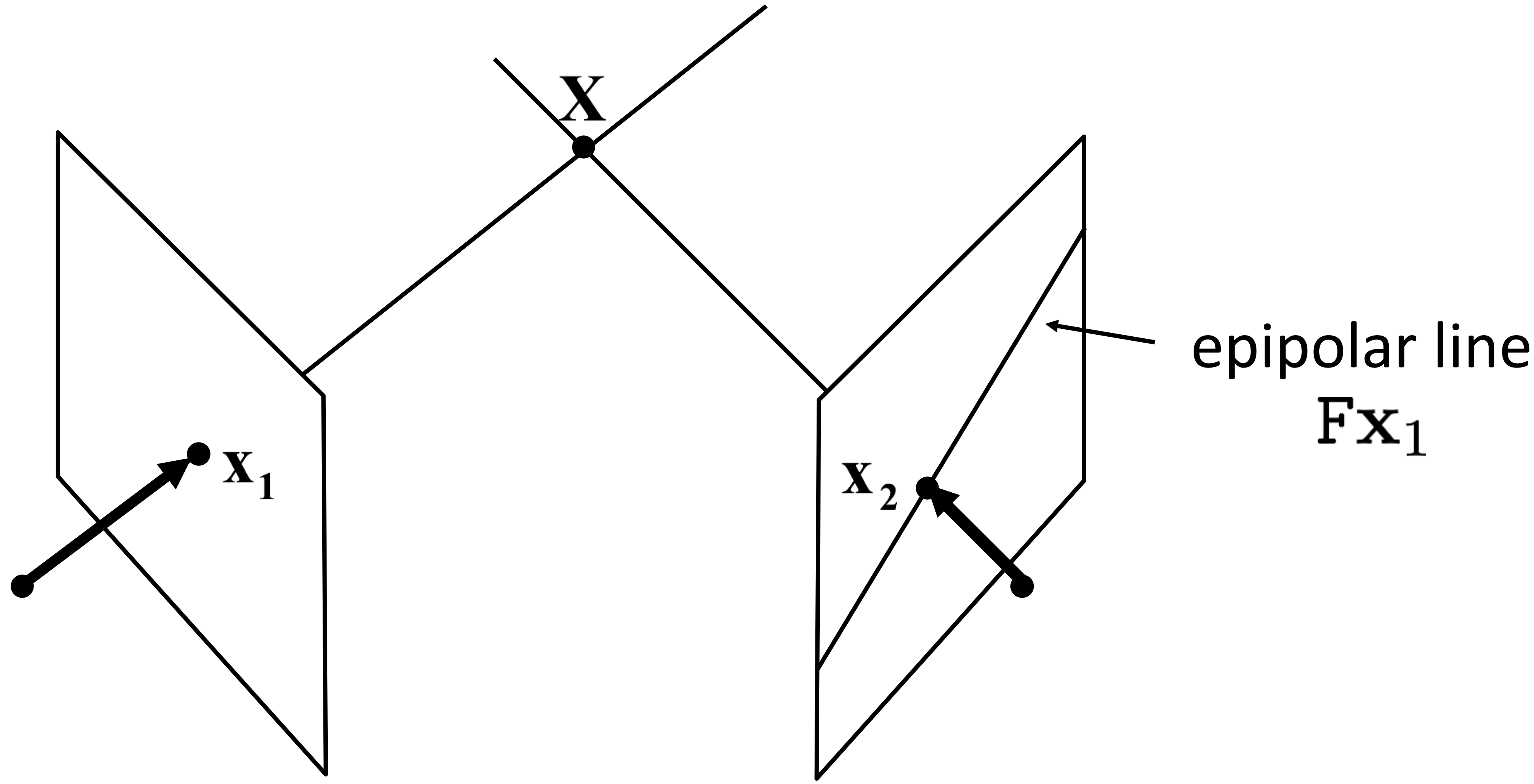
$$0 = \mathbf{x}_2^T \mathbf{F} \mathbf{x}_1$$

Geometric Interpretation



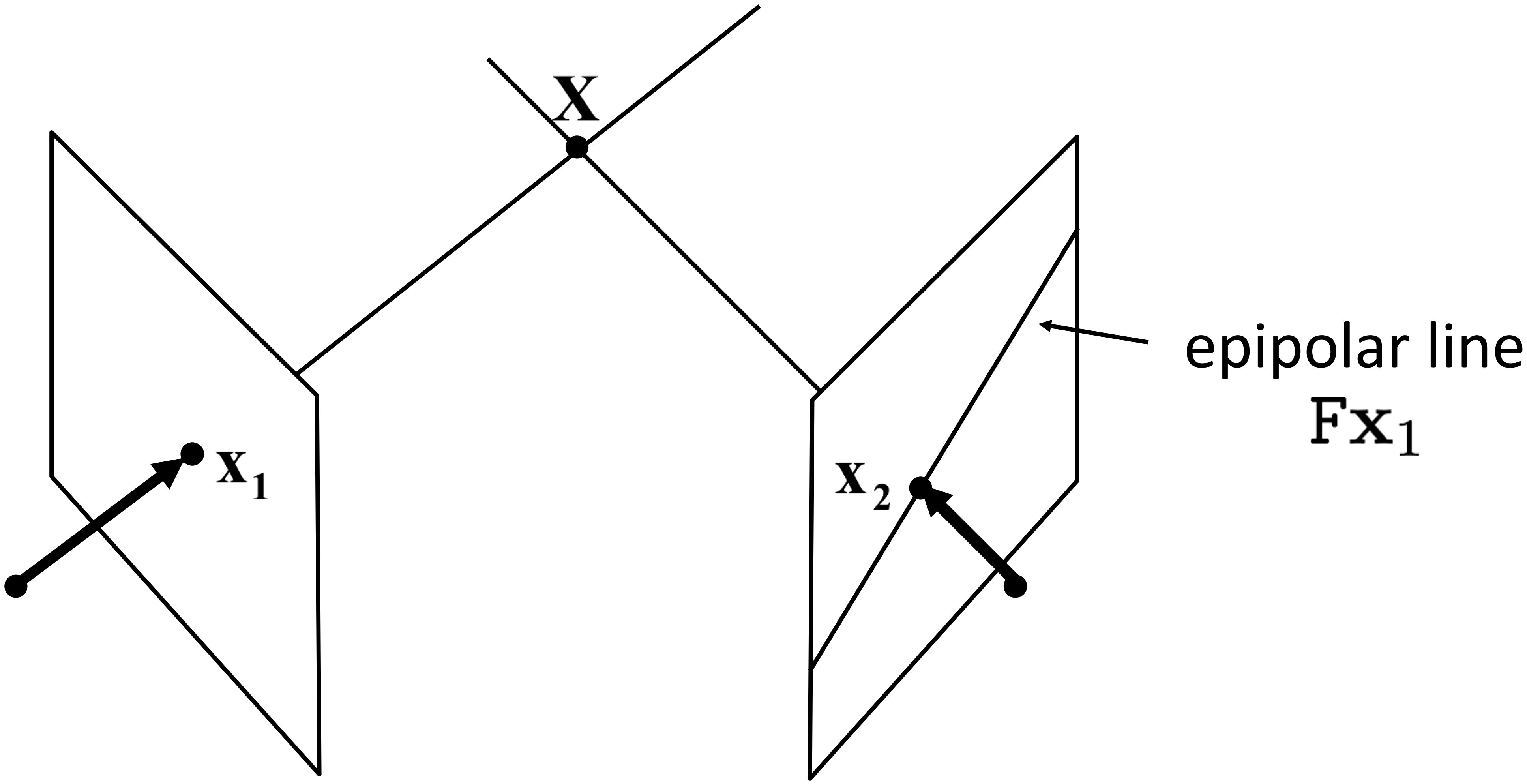
$$0 = \mathbf{x}_2^T \mathbf{F} \mathbf{x}_1$$

Geometric Interpretation



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

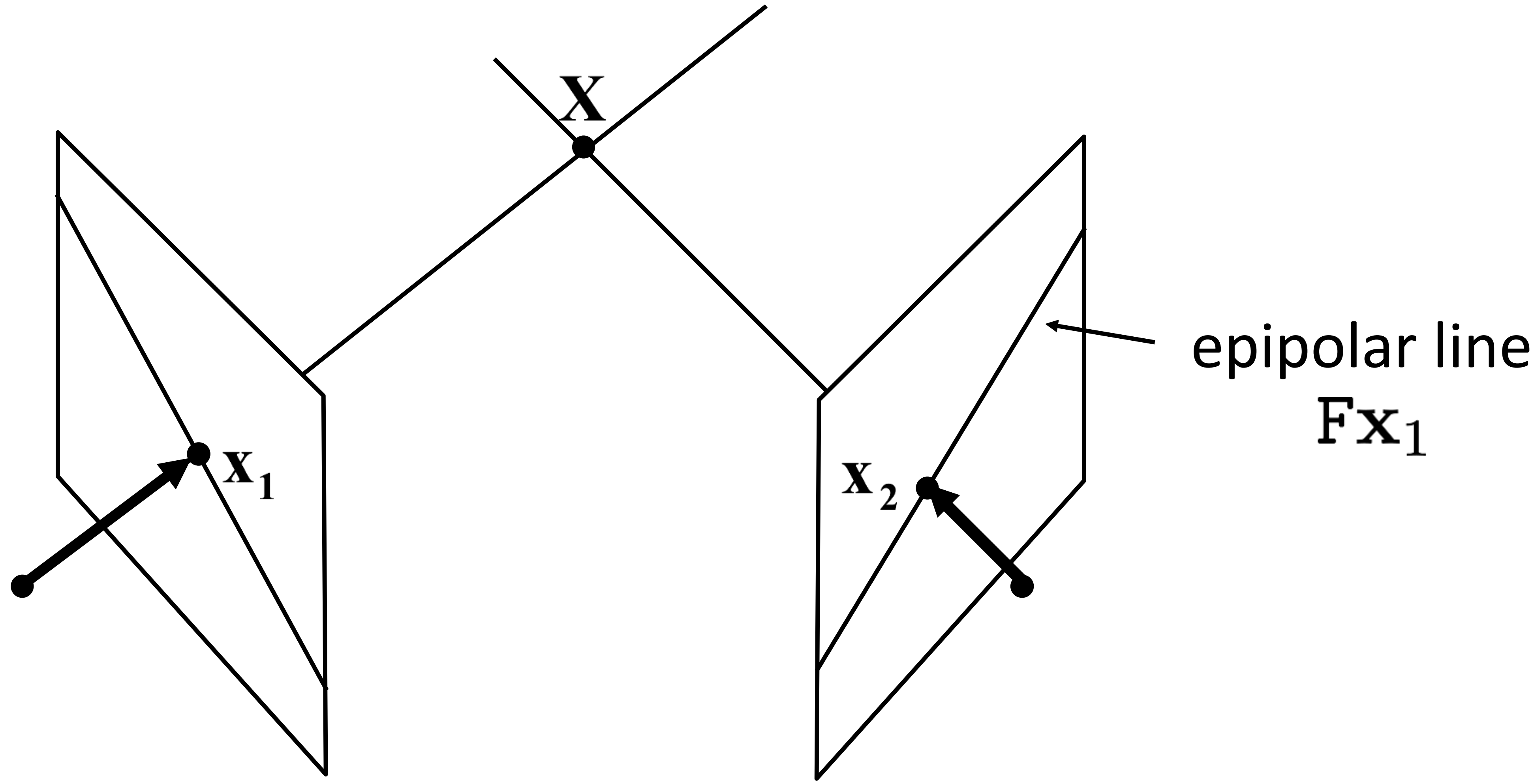
Geometric Interpretation



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

- $\mathtt{F}$ maps points in first image to lines in second image

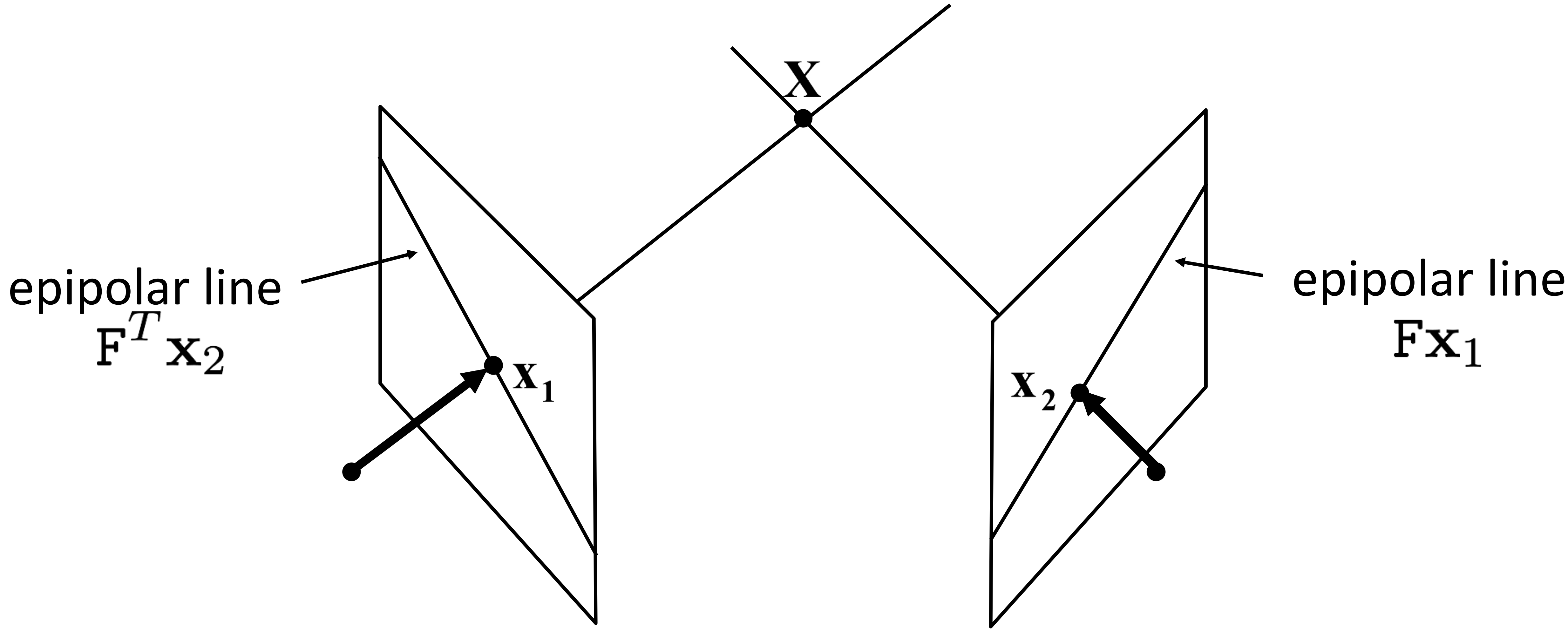
Geometric Interpretation



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

- $\mathtt{F}$ maps points in first image to lines in second image

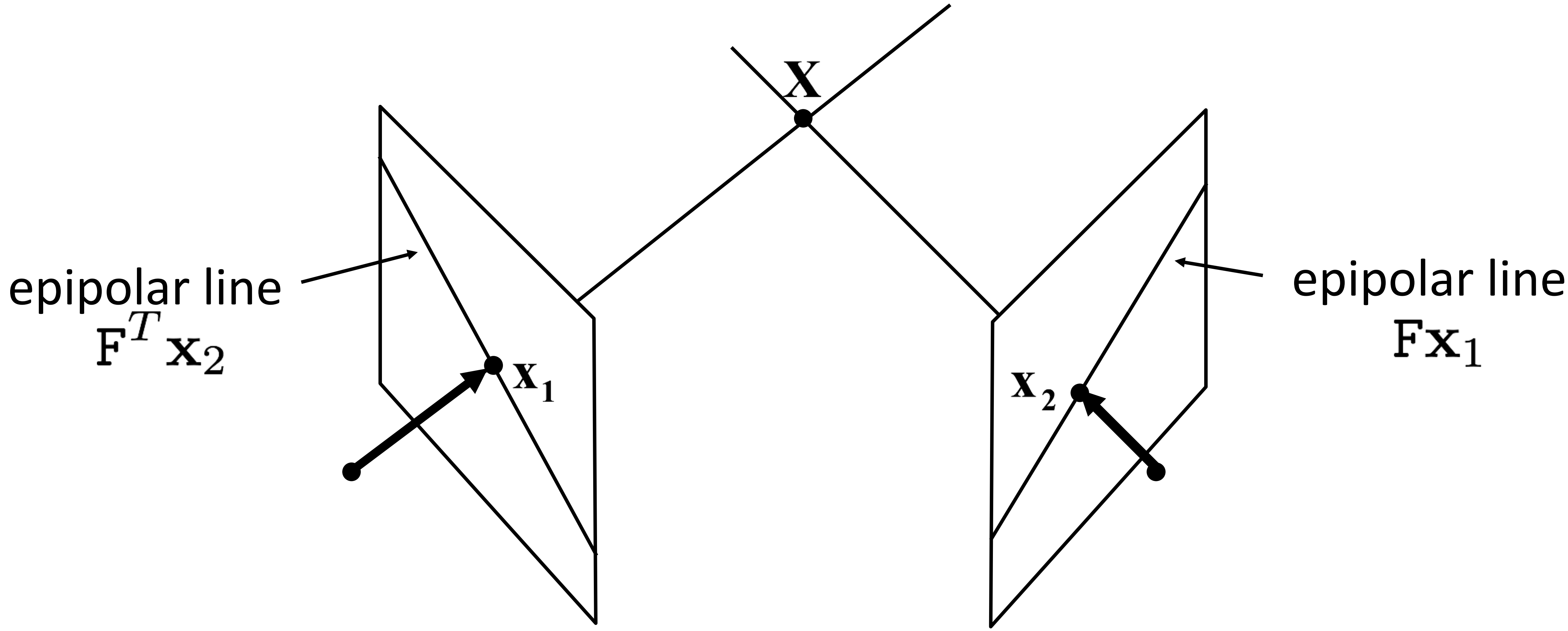
Geometric Interpretation



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

- $\mathtt{F}$ maps points in first image to lines in second image

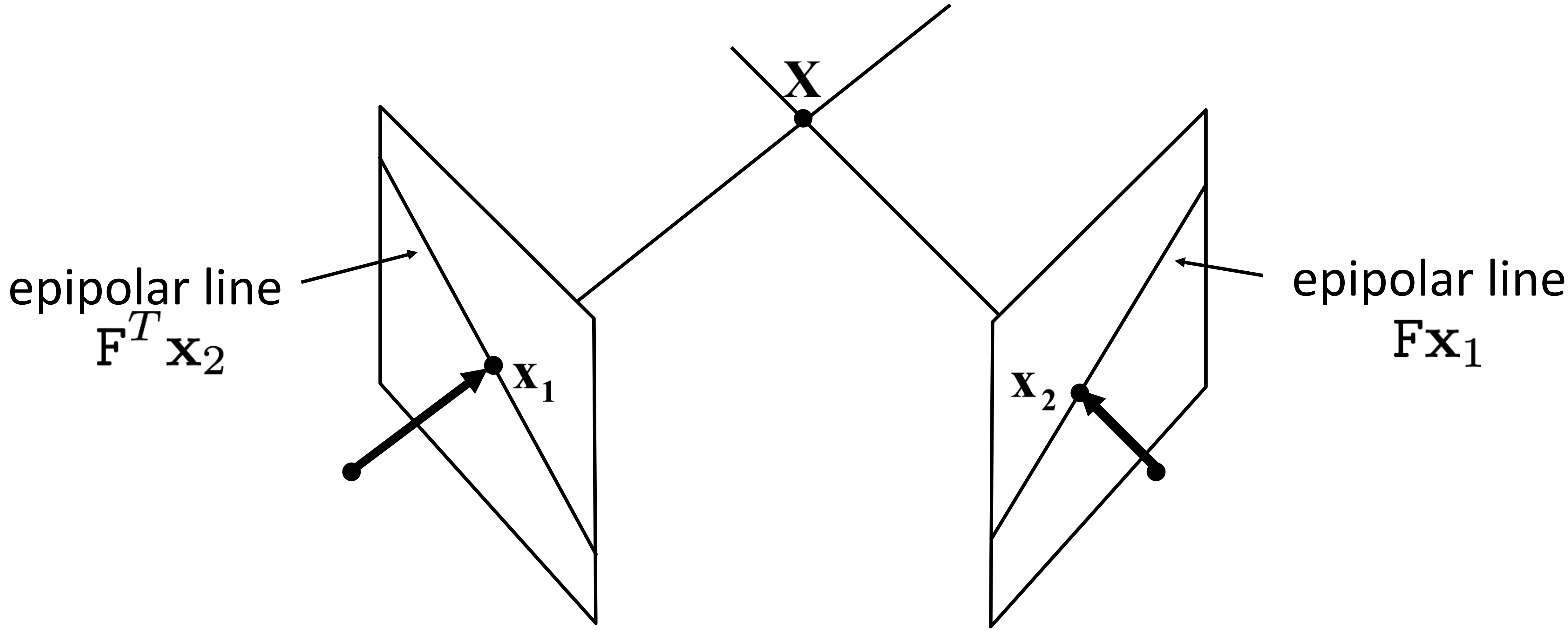
Geometric Interpretation



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

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

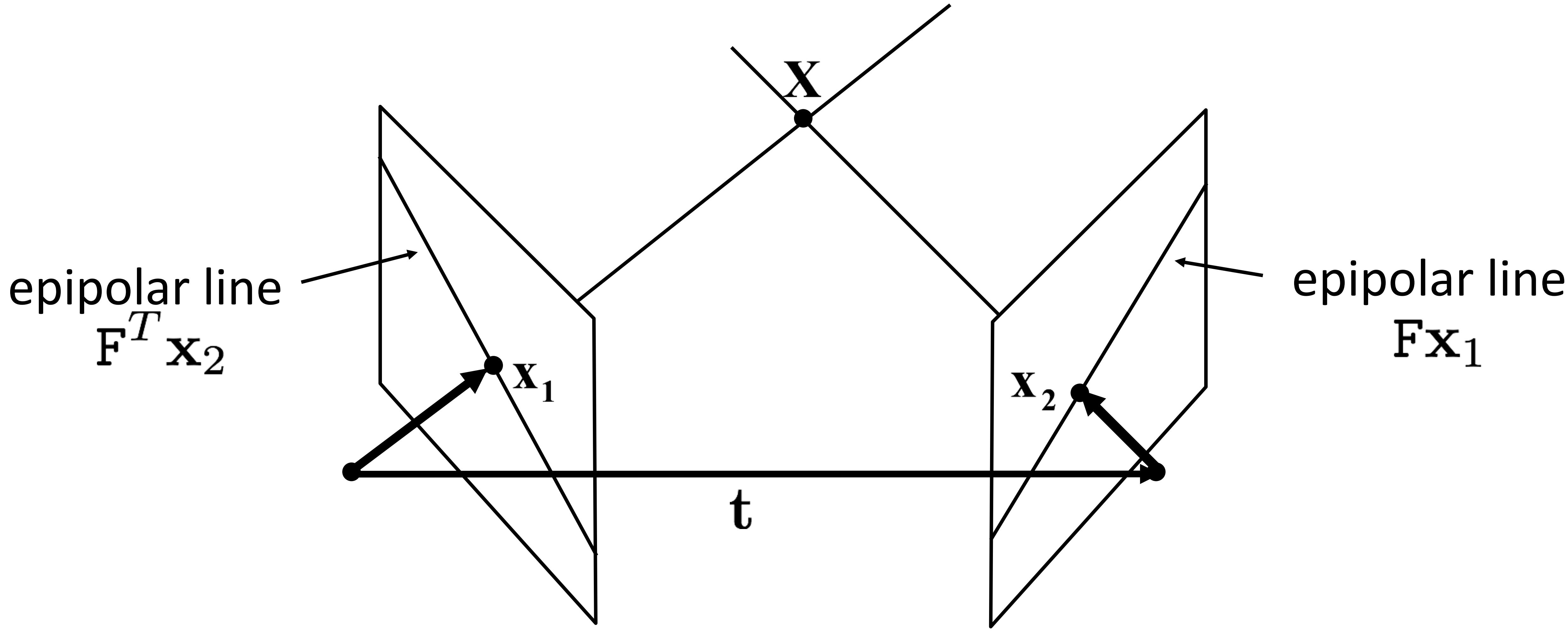
Geometric Interpretation



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

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

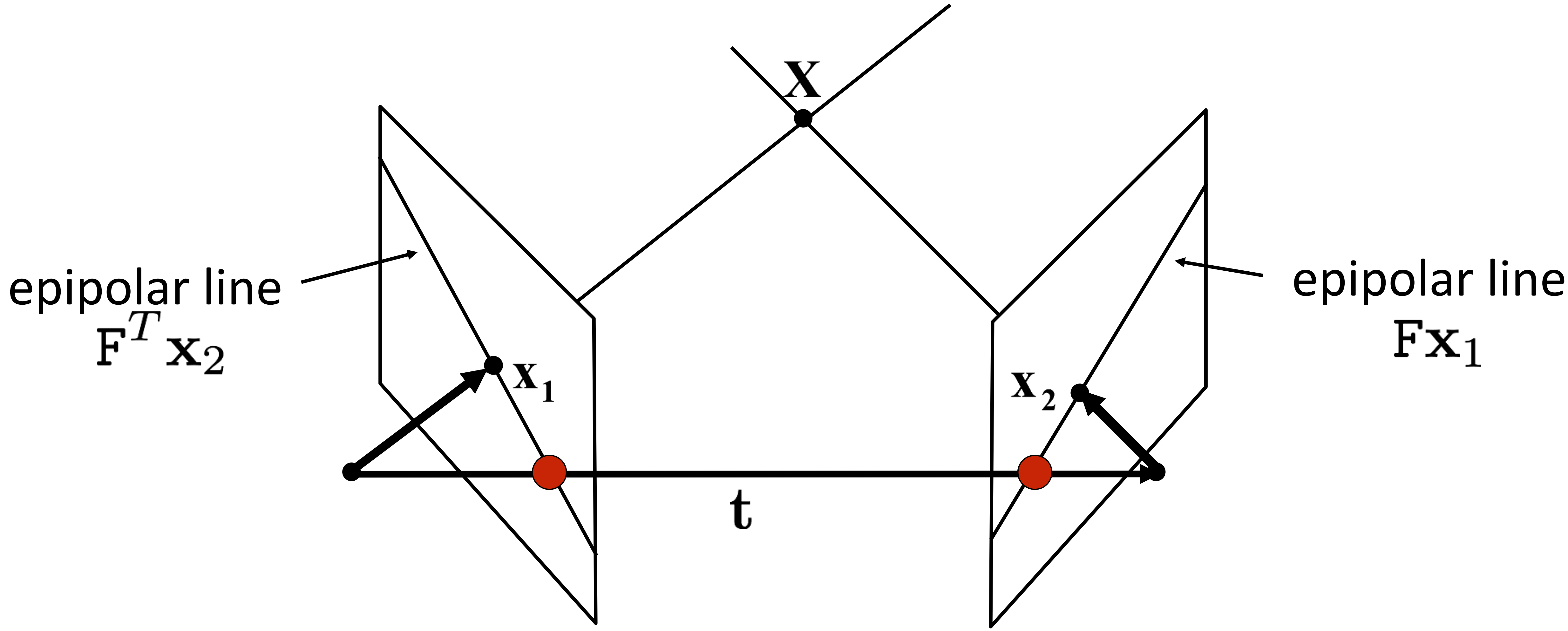
Geometric Interpretation



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

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

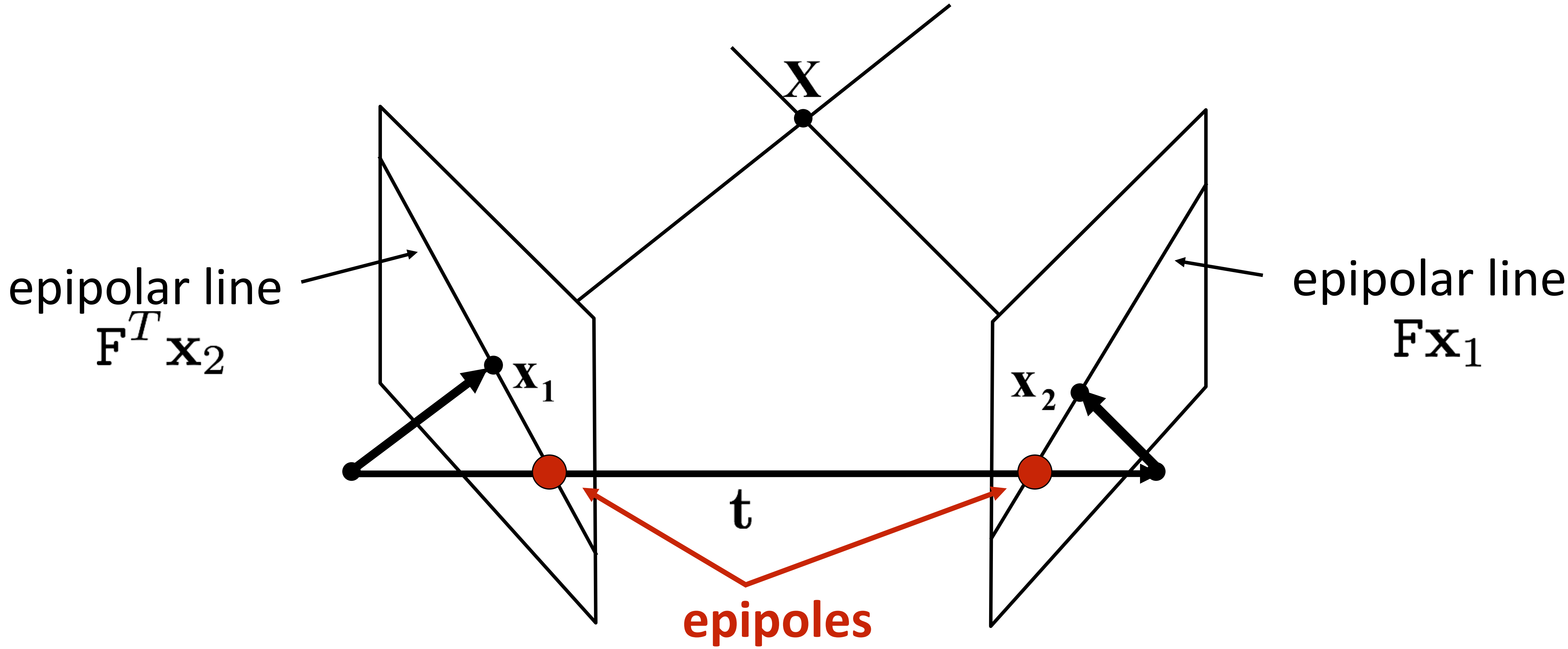
Geometric Interpretation



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

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

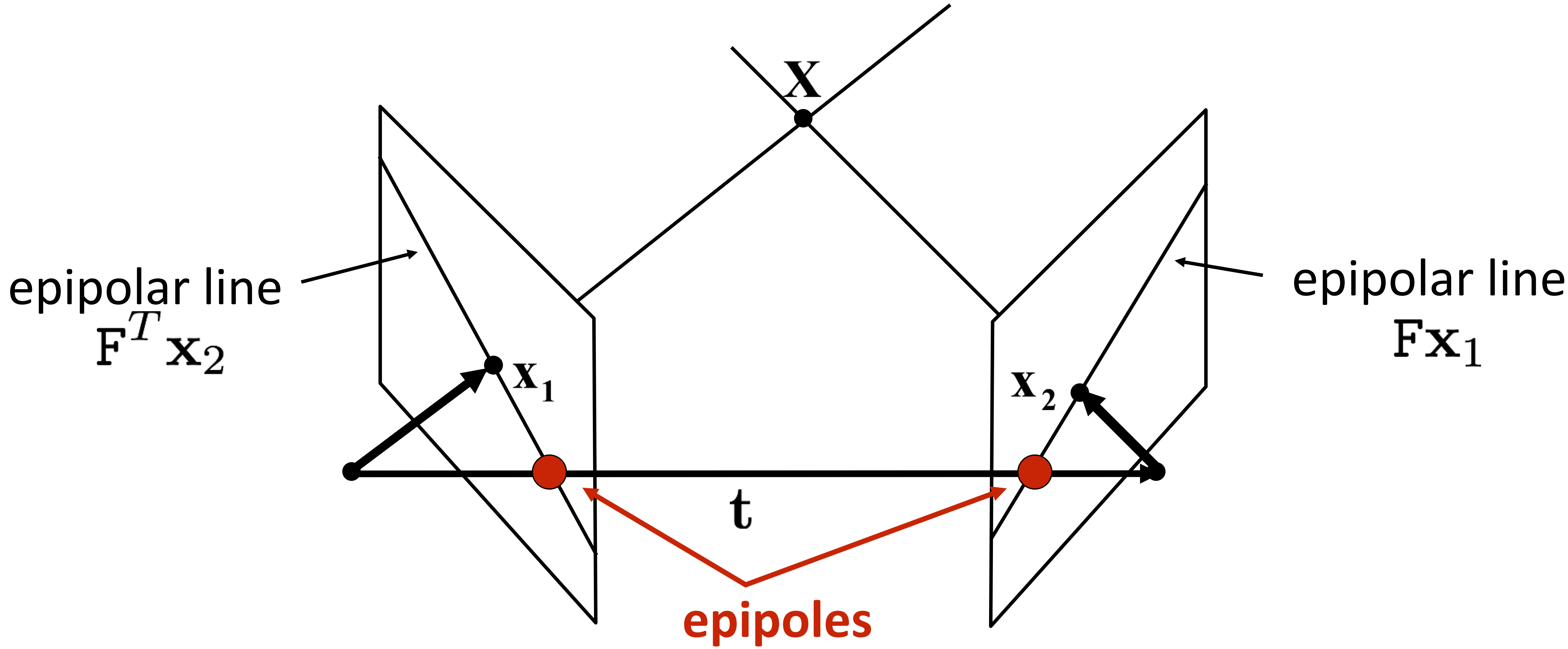
Geometric Interpretation



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

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

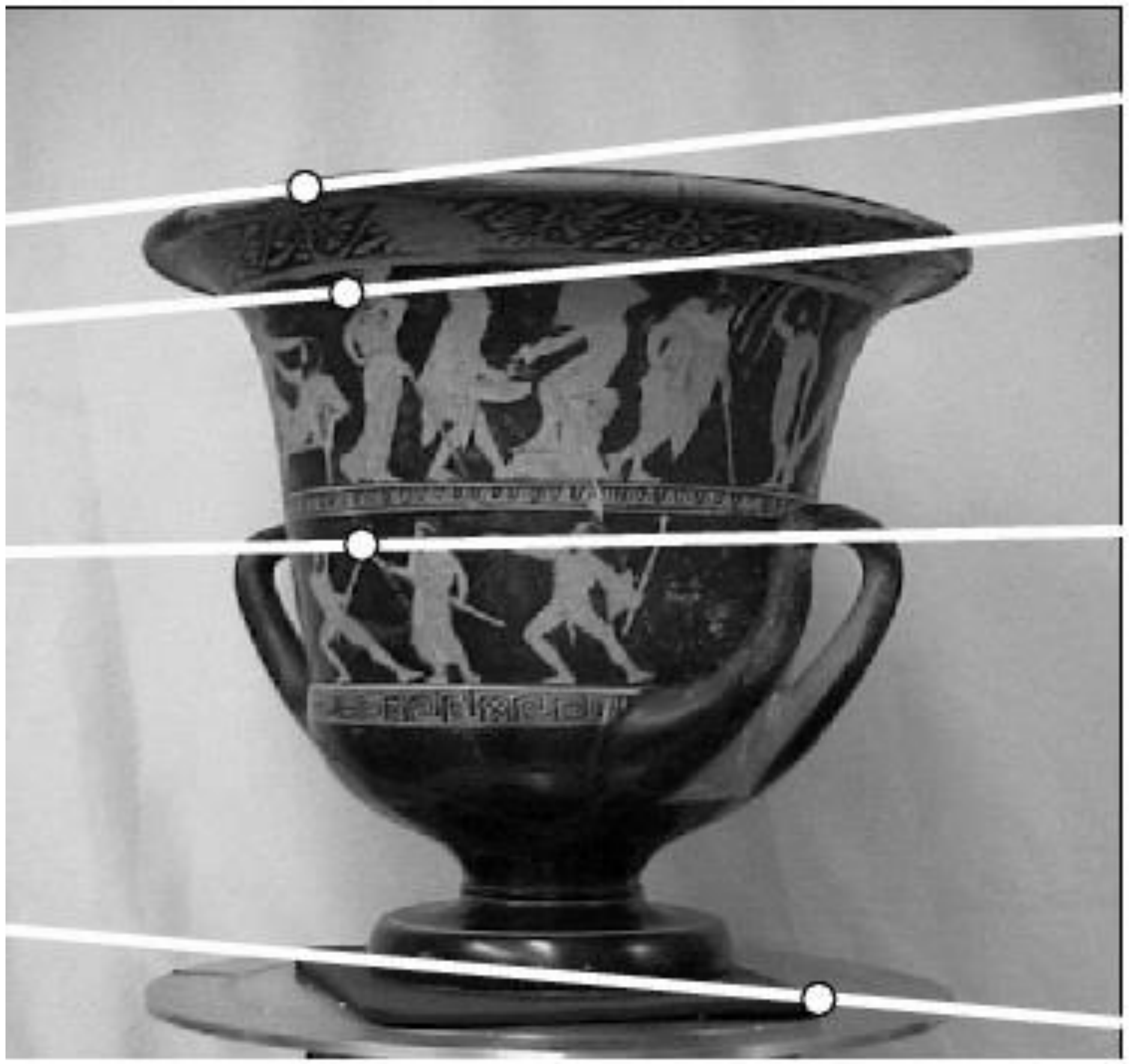
Geometric Interpretation



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

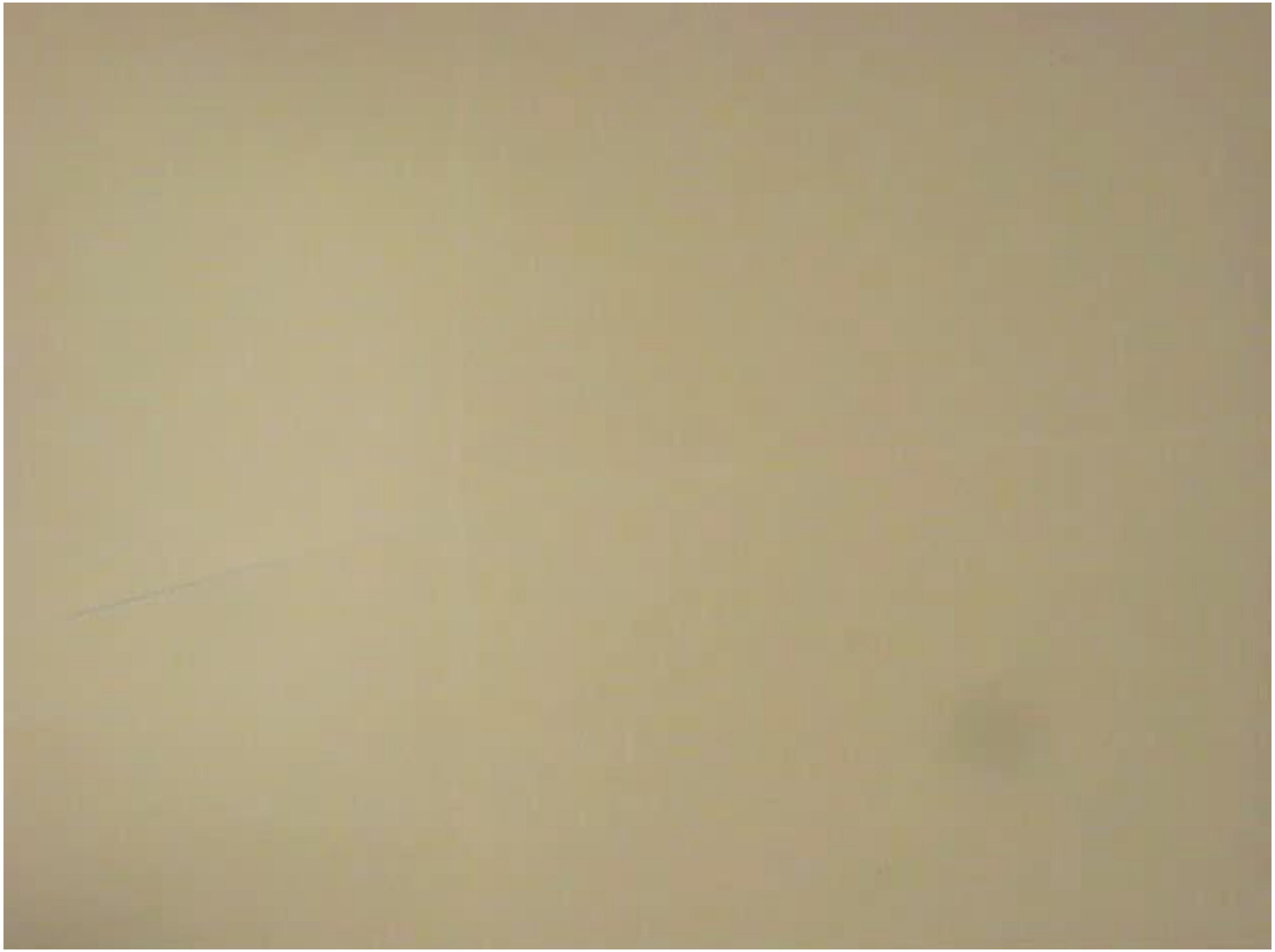
- $\mathtt{F}$ maps points in first image to lines in second image
- $\mathtt{F}^{\mathtt{T}}$ maps points in second image to lines in first image
- Lines are called **epipolar lines**
- All epipolar lines intersect in the **epipoles**

Epipolar Lines



Brief Recap

Brief Recap



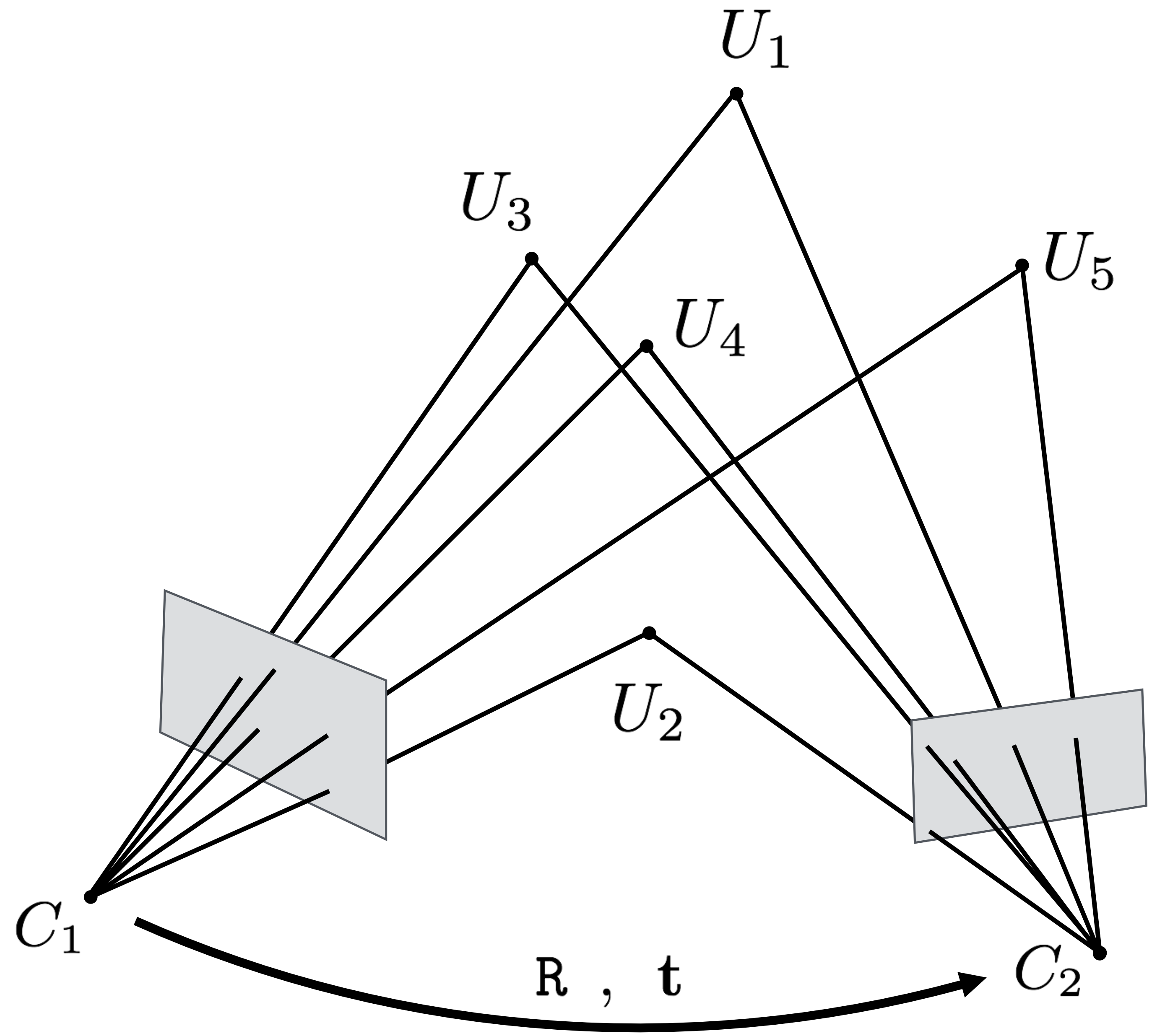
http://danielwedge.com/fmatrix/

Finding More Matches

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

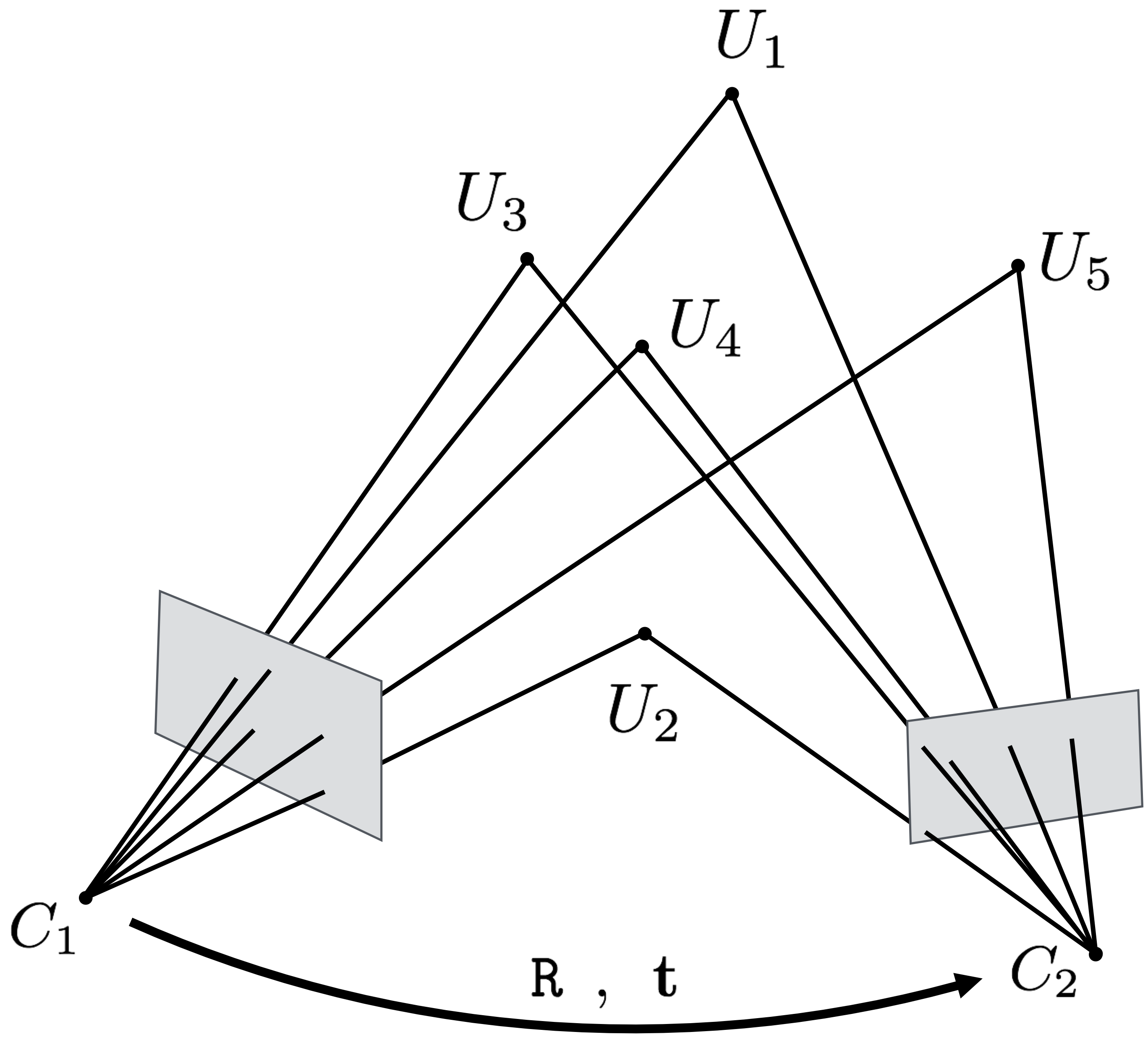
slide credit: Marc Pollefeys

Relative Pose Estimation



- Compute E / F

Relative Pose Estimation



- Compute E / F
- Decompose E / F to obtain rotation and translation

Relative Pose Estimation

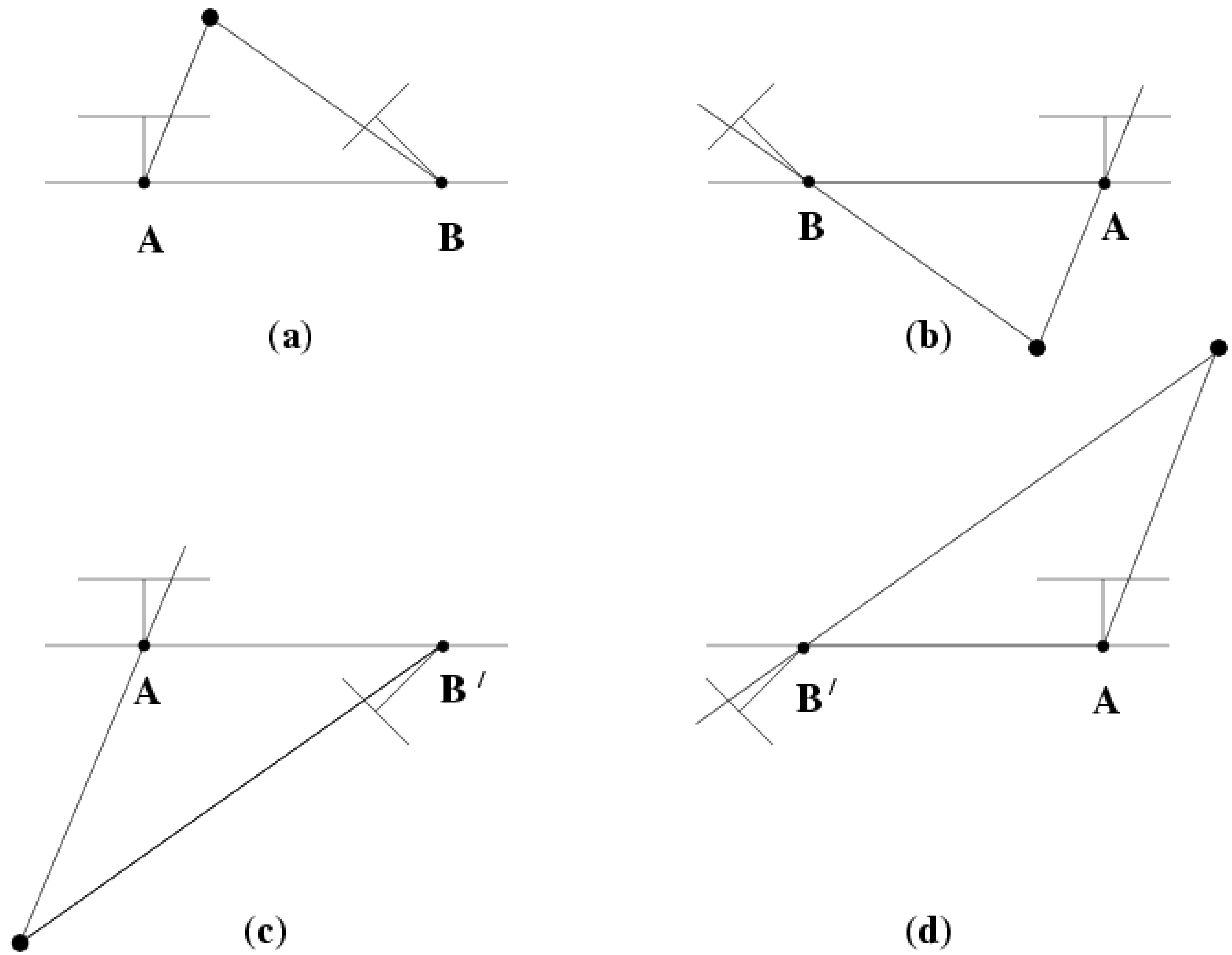


figure from Hartley and Zisserman, 2004

Today

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

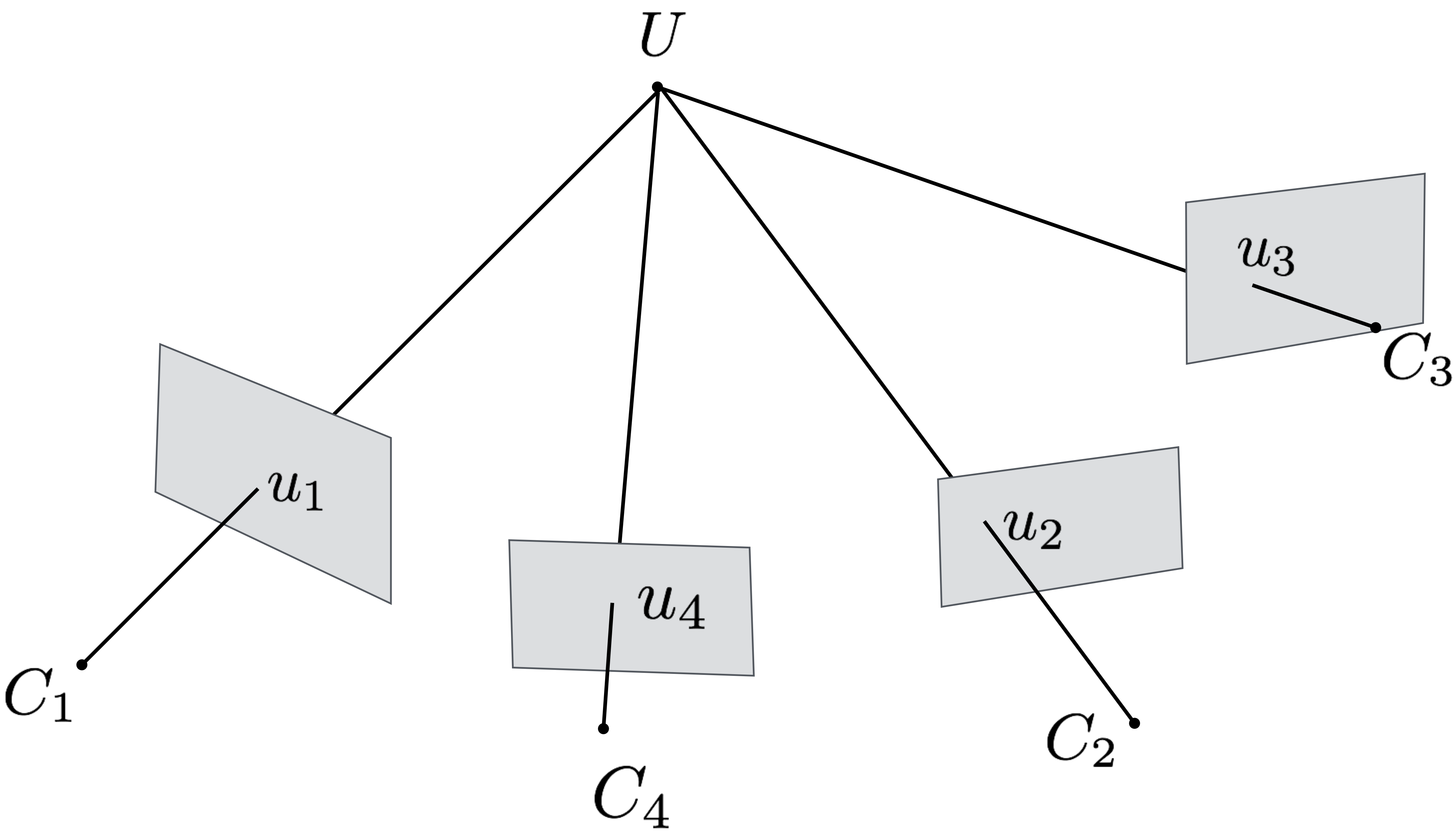
Initialize motion from two views

Initialize structure from two views

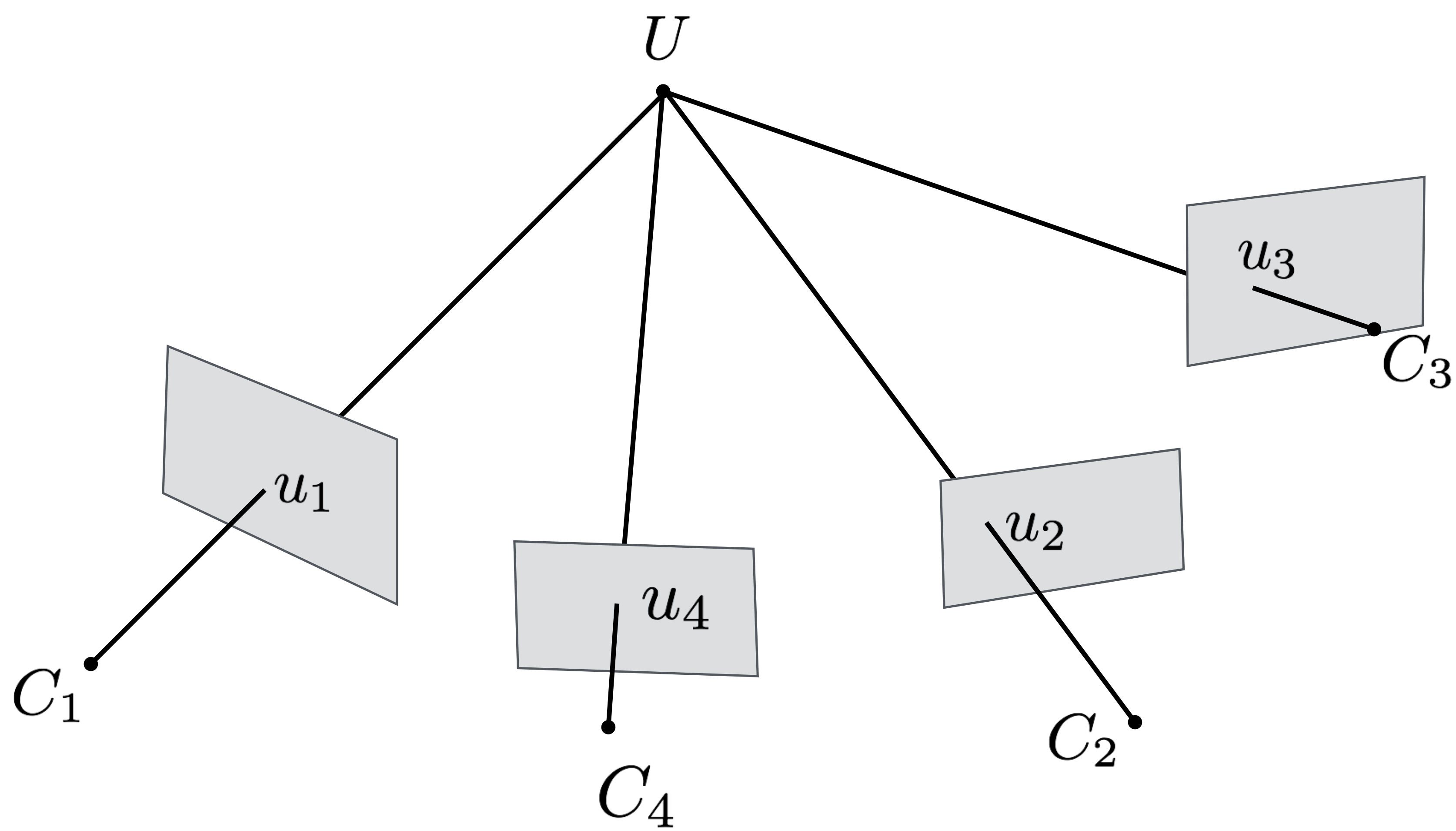
Extend motion

Extend structure

Triangulation

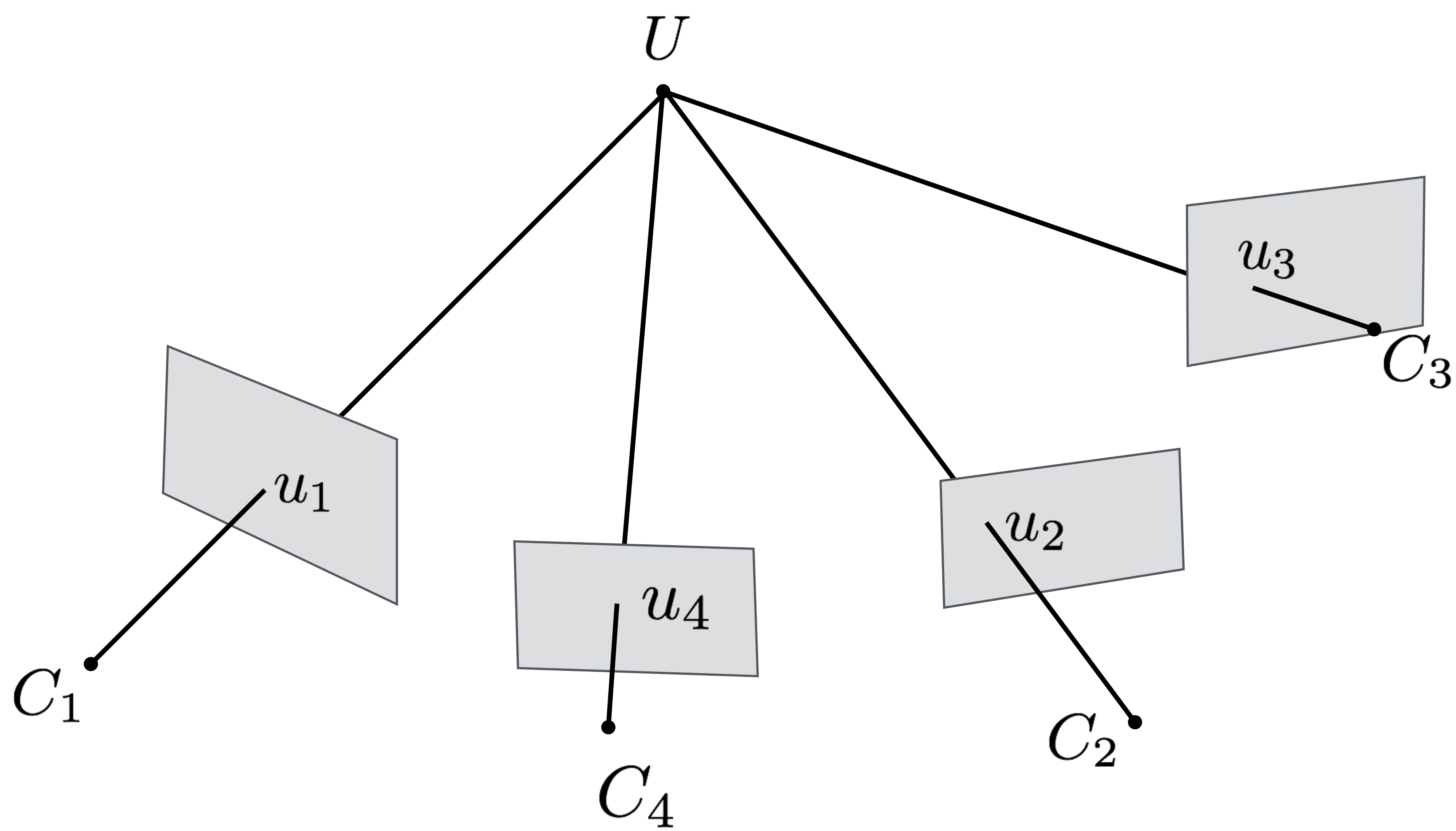


Triangulation using RANSAC



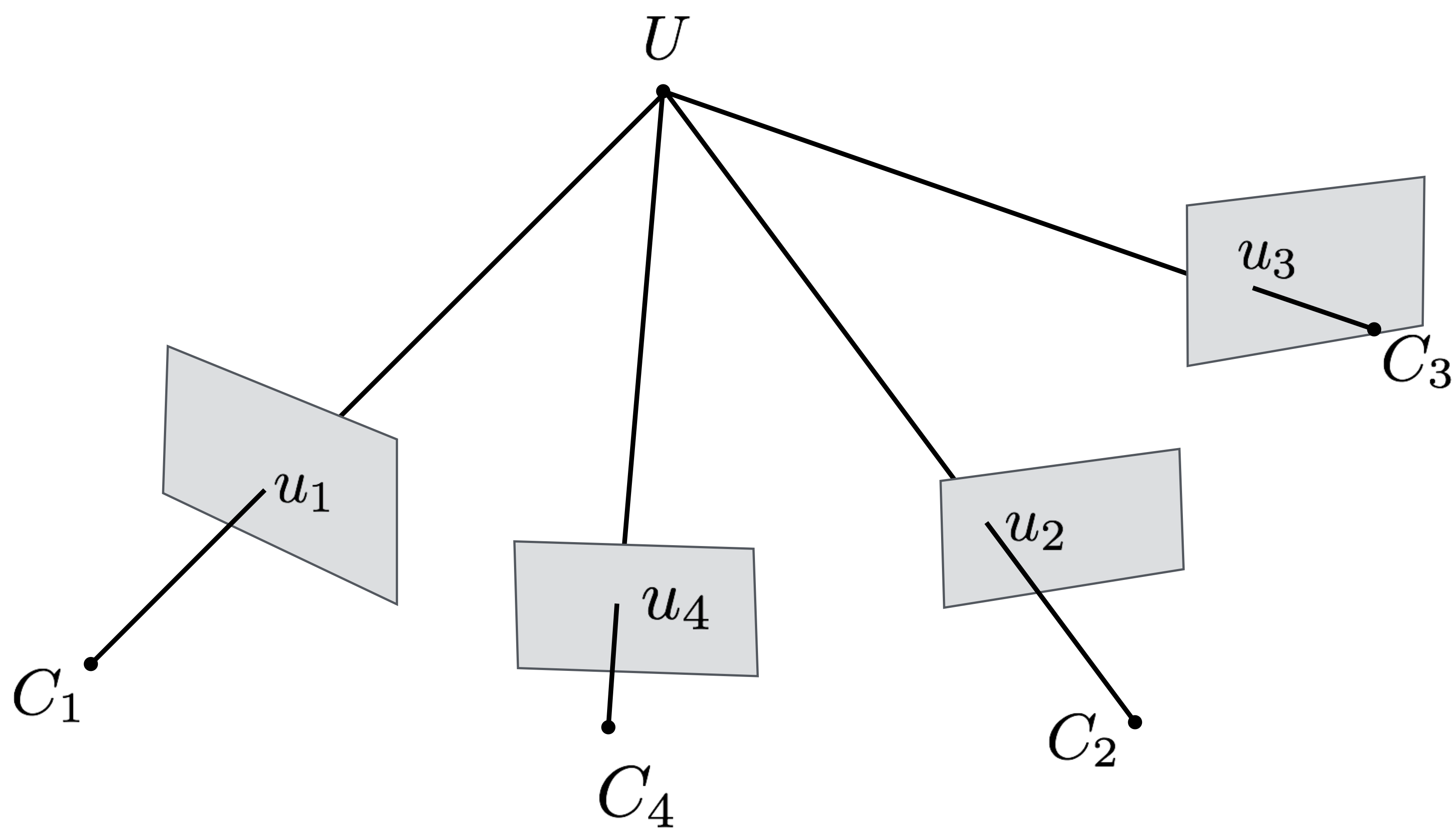
- Given: Projection matrices, track of 2D features

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

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

Minimal Solver for Triangulation

Perspective projection in homogeneous coordinates

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

Minimal Solver for Triangulation

Perspective projection in homogeneous coordinates

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

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

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

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:

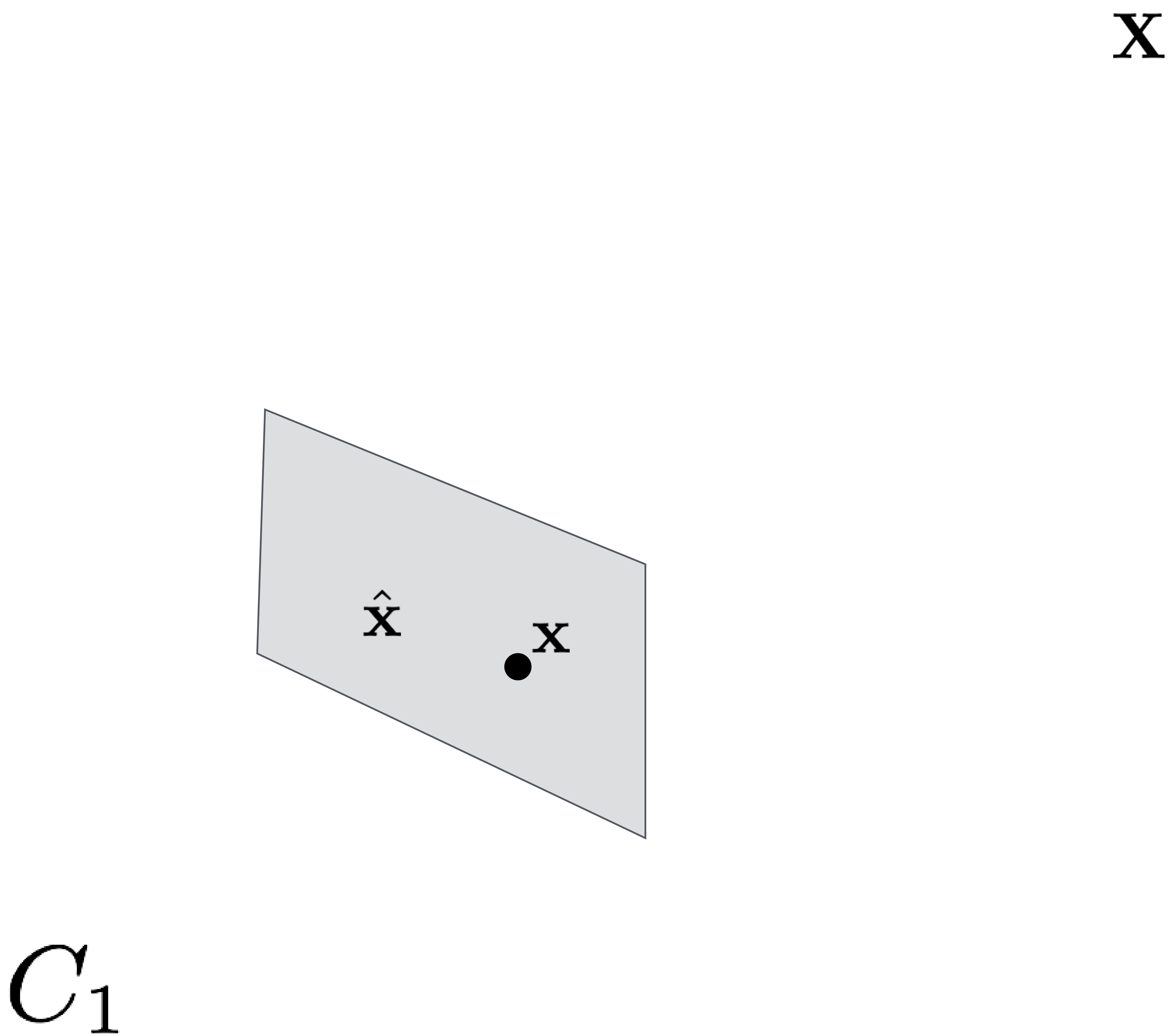
$$\begin{pmatrix} \mathbf{P}_3 x - \mathbf{P}_1 \\ \mathbf{P}_3 y - \mathbf{P}_2 \end{pmatrix} \mathbf{X} = \mathbf{0}$$

Minimal Solver for Triangulation

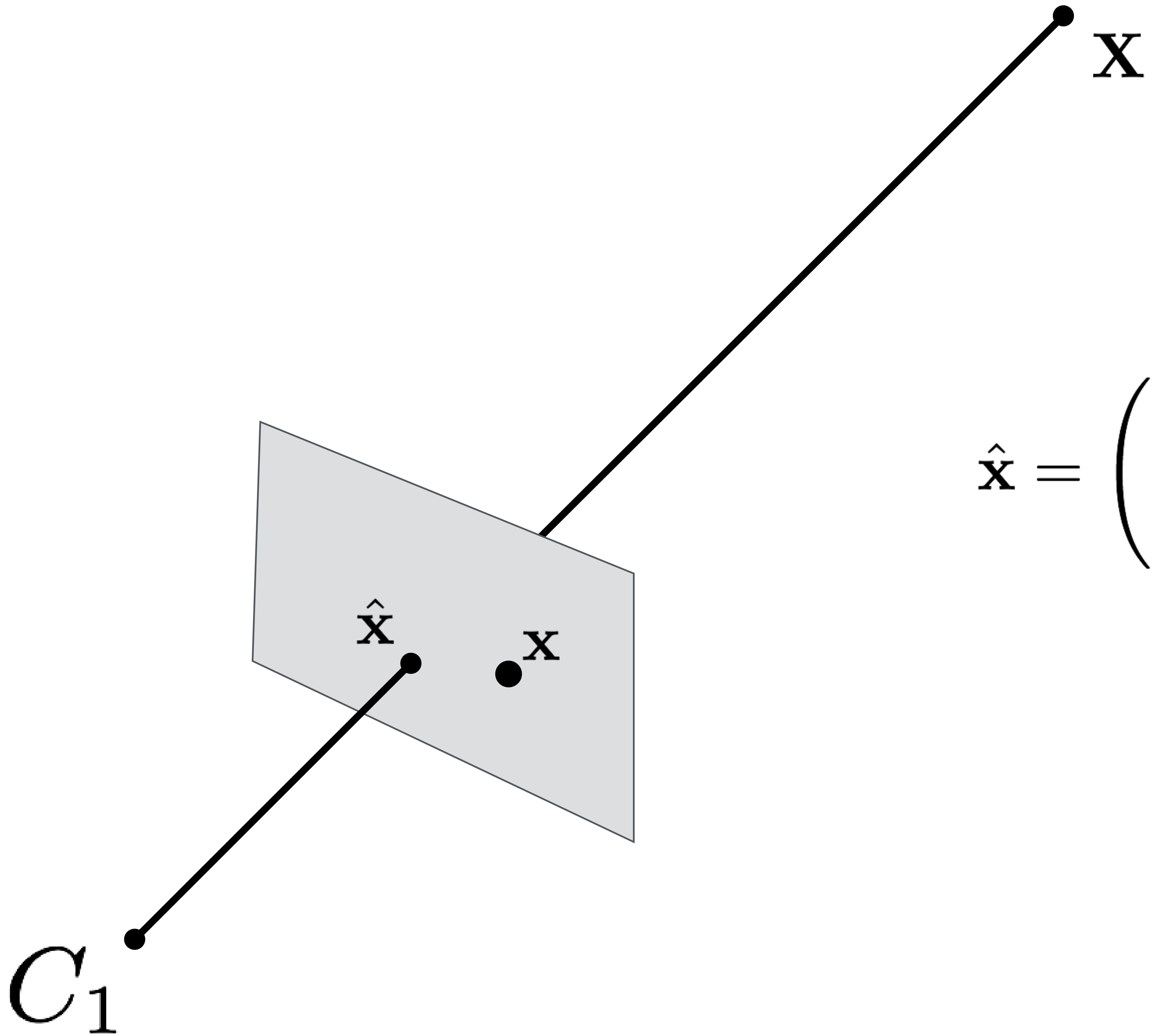
Need two images to solve for the 4 unknowns

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

The Reprojection Error

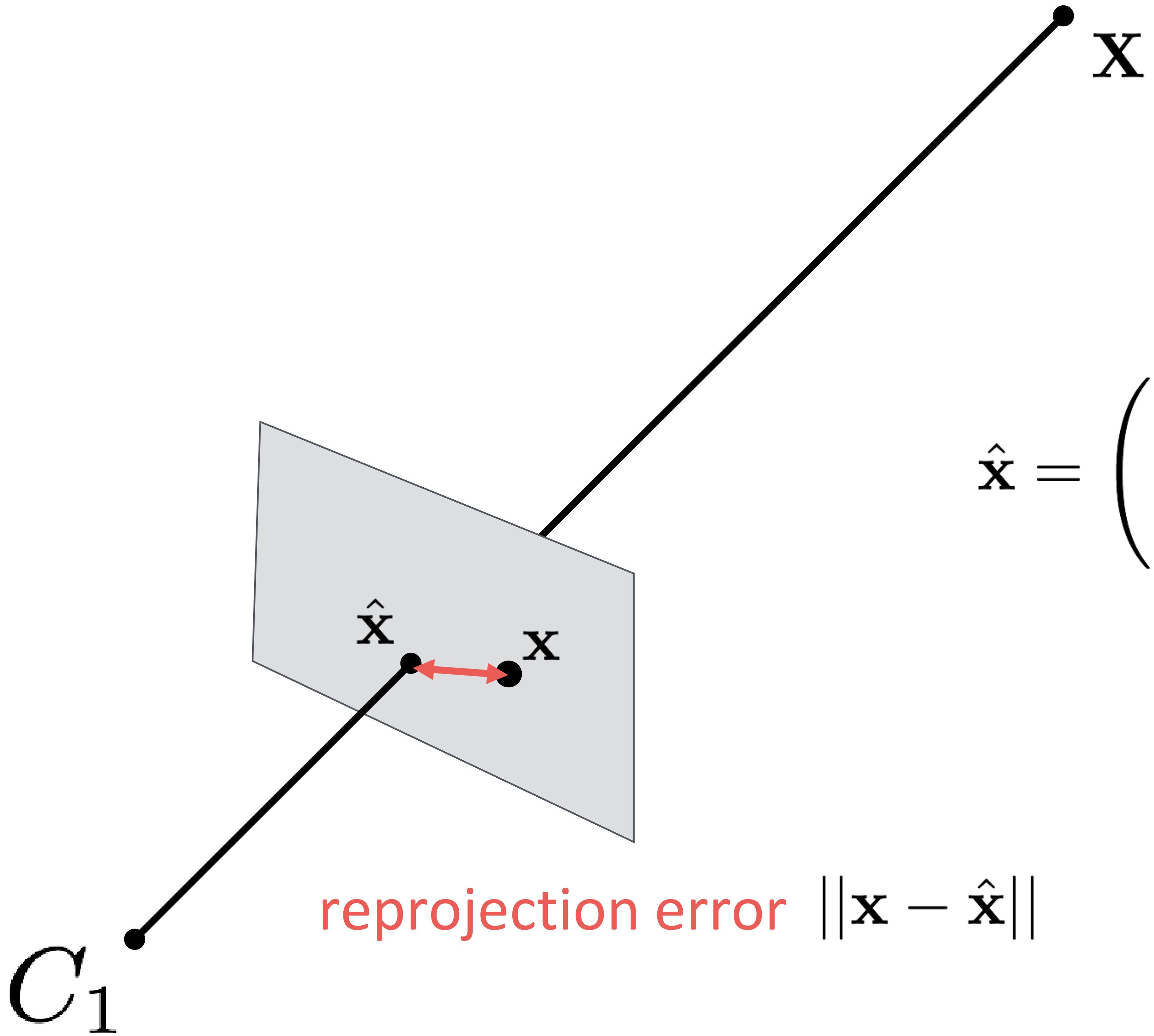


The Reprojection Error



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

The Reprojection Error



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

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:

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

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

Least Squares Solution

Maximum likelihood estimate:

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

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

Cost function non-linear …

Least Squares Solution

Maximum likelihood estimate:

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

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

Cost function non-linear …

… but we have initial guess from RANSAC

Least Squares Solution

Maximum likelihood estimate:

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

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

Cost function non-linear …

… but we have initial guess from RANSAC

… use Gradient Descent for minimization

Today

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

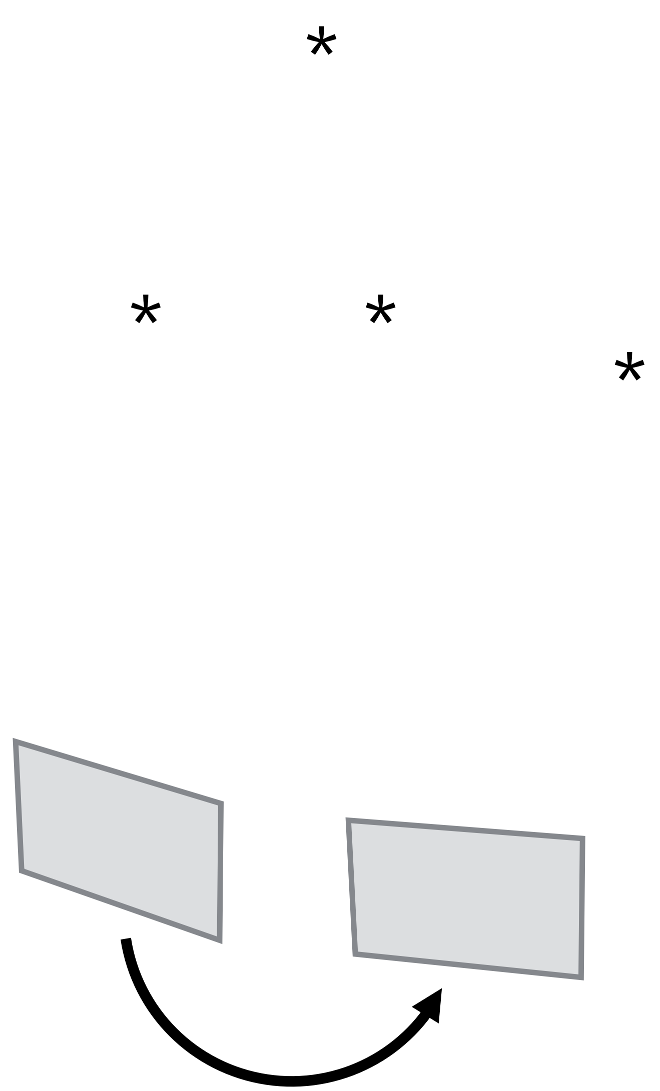
Initialize motion from two views

Initialize structure from two views

Extend motion

Extend structure

Sequential Structure-from-Motion

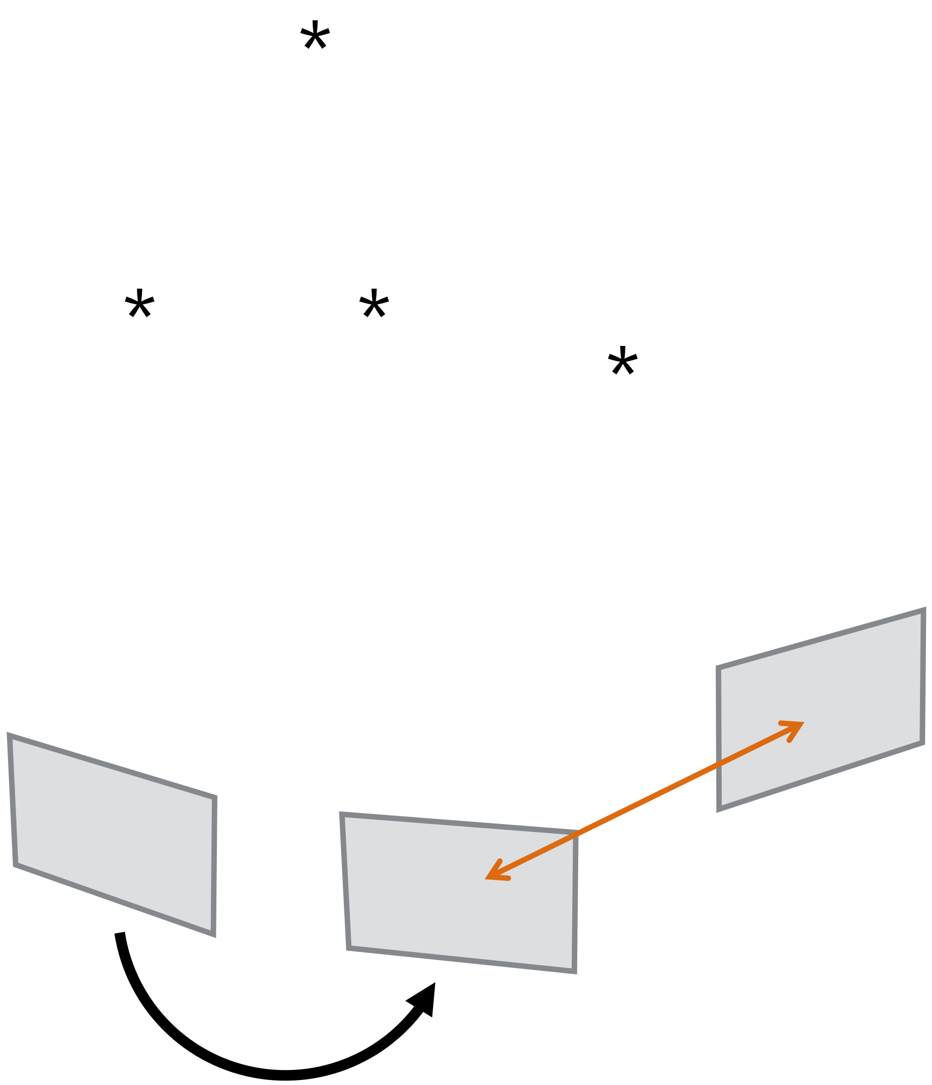


$\mathrm{R}, \mathbf{t}$ with $||\mathbf{t}|| = 1$

Initialize motion from two views

Initialize structure from two views

Sequential Structure-from-Motion



$$\mathbf{R}, \mathbf{t} \text{ with } ||\mathbf{t}|| = 1$$

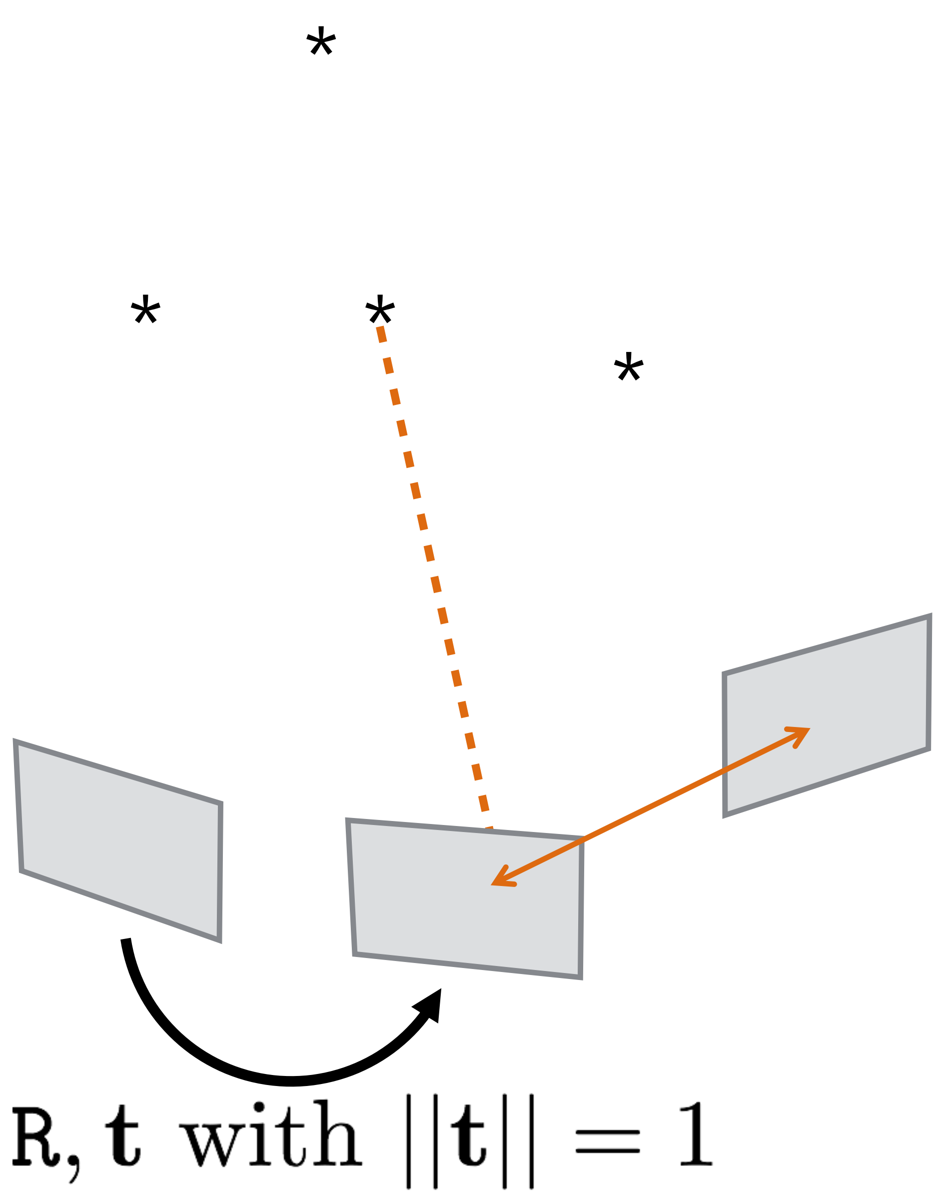
Match features

Initialize motion from two views

Initialize structure from two views

Extend motion

Sequential Structure-from-Motion



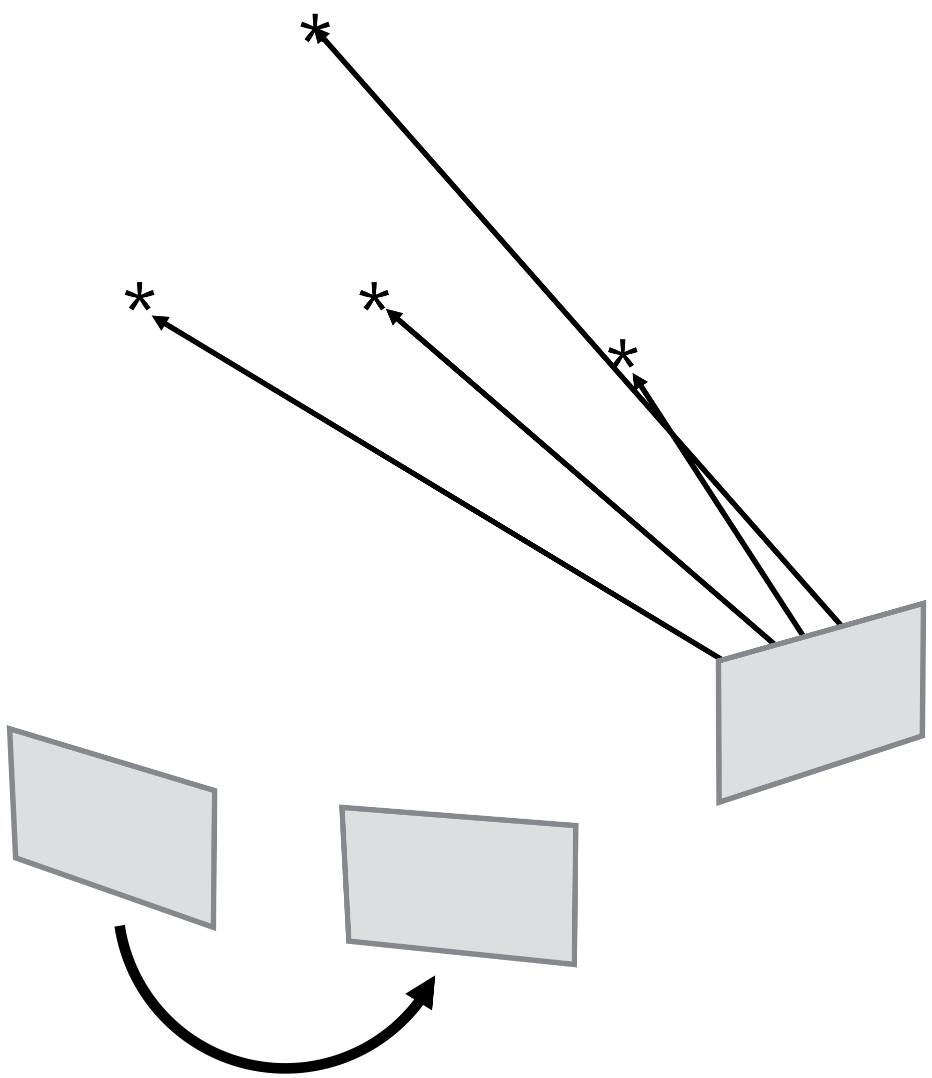
Initialize motion from two views

Initialize structure from two views

Extend motion

Transfer matches to 3D

Sequential Structure-from-Motion



$$\mathtt{R}, \mathbf{t} \text{ with } ||\mathbf{t}|| = 1$$

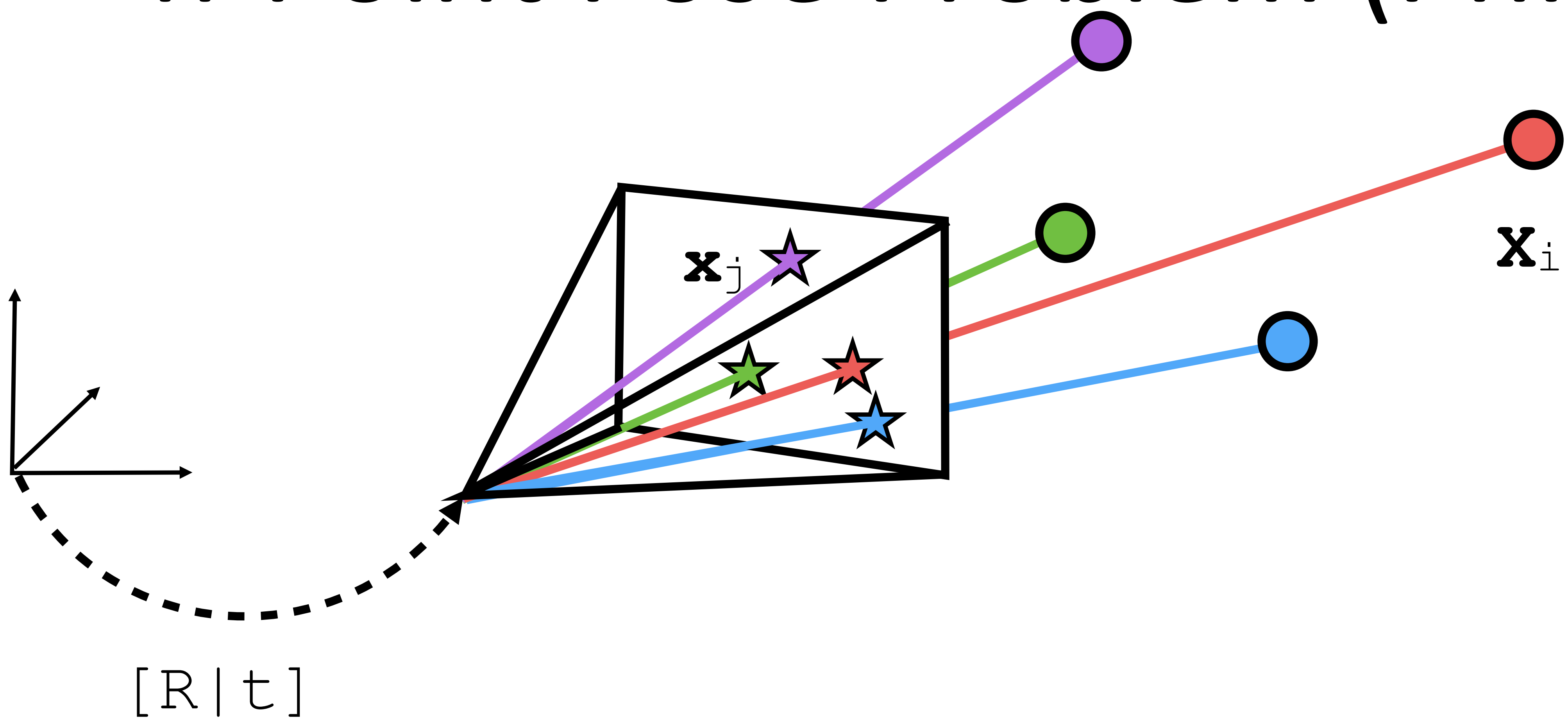
Camera pose for third camera

Initialize motion from two views

Initialize structure from two views

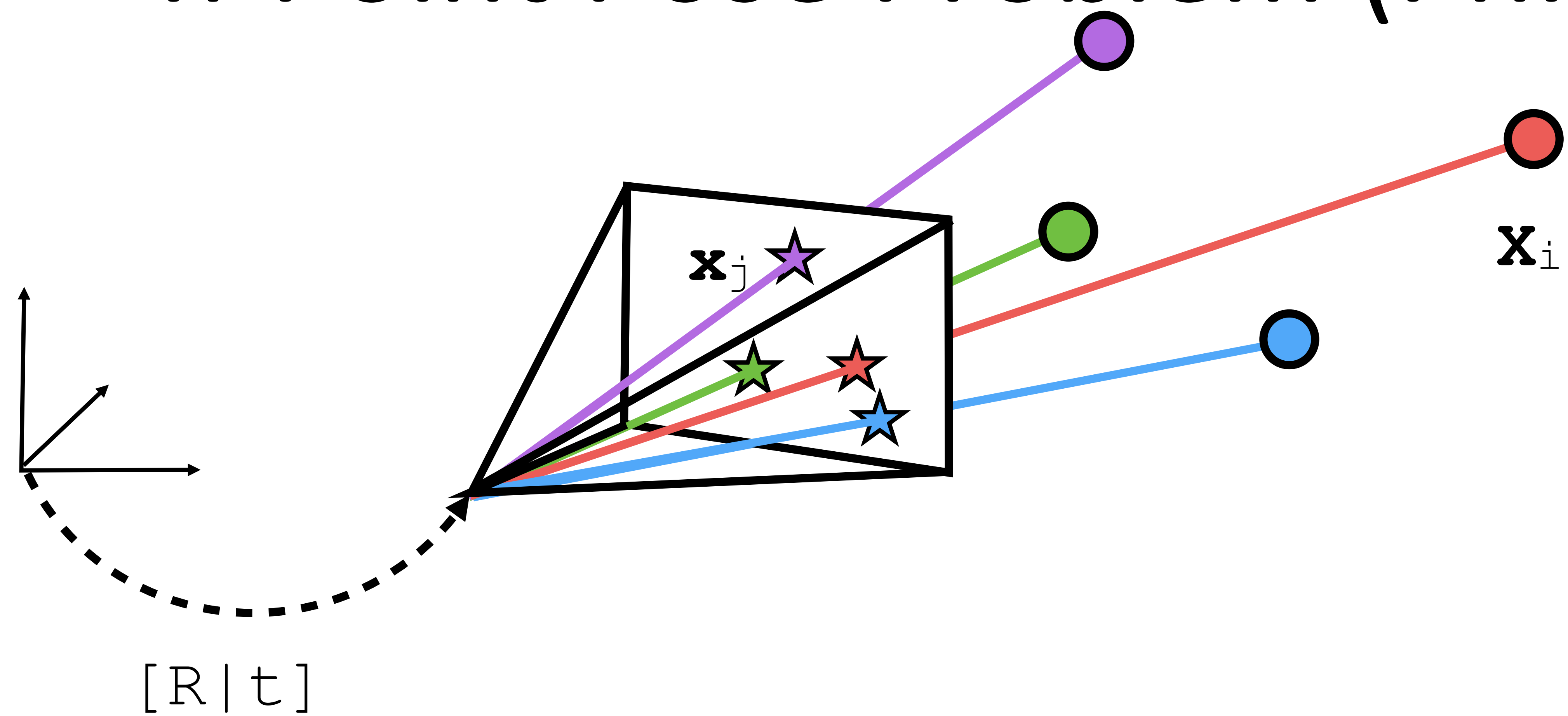
Extend motion

n-Point Pose Problem (PnP)



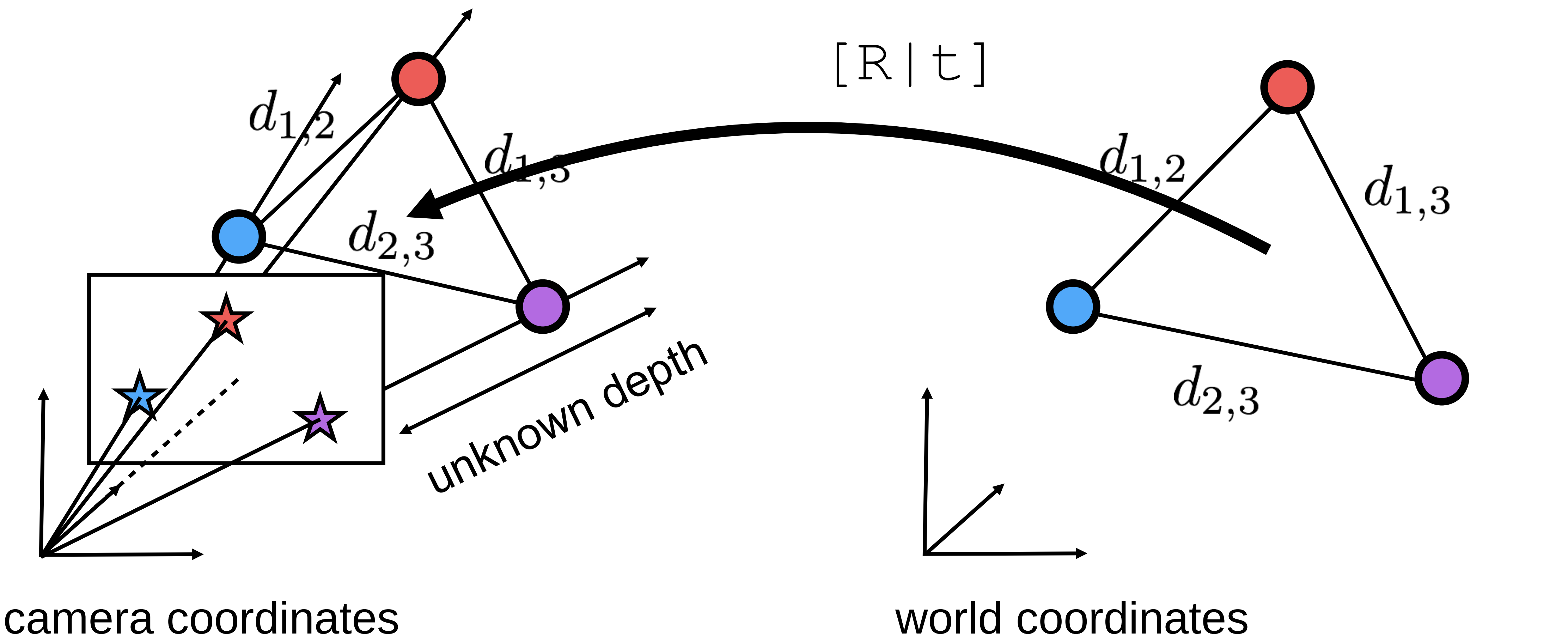
- Given: *n* 2D-3D correspondences $(\mathbf{x}_i,\ \mathbf{X}_i)$
- Compute pose $[R|t]$ s.t. $K[R|t]\mathbf{X}_i = \alpha_i \mathbf{x}_i, \alpha_i > 0$

n-Point Pose Problem (PnP)



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

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: ~$2\mu s$ total [Kneip et al., CVPR'11] [code]
- Up to four solutions: Disambiguate using 4th point

Unknown Focal Length

P4Pf: Estimate focal length and pose from 4 matches [Bujnak et al., CVPR'08] [code]

- Solve system of **multivariate** polynomials
- Recover variables as Eigenvectors of 10×10 matrix
- Usually returns multiple solutions
  - Disambiguate using 5th point

Unknown Intrinsics

General projection equation:

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

Unknown Intrinsics

General projection equation:

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

6-point DLT algorithm, similar to homography DLT:

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

Unknown Intrinsics

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

$$\left[\begin{array}{ccc} \mathbf{0}^\top & -w_i\mathbf{X}_i{}^\top & y_i\mathbf{X}_i{}^\top \\ w_i\mathbf{X}_i{}^\top & \mathbf{0}^\top & -x_i\mathbf{X}_i{}^\top \end{array}\right]\left(\begin{array}{c} \mathbf{P}^1 \\ \mathbf{P}^2 \\ \mathbf{P}^3 \end{array}\right) = \mathbf{0}$$

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:

$$\left[\begin{array}{ccc} \mathbf{0}^\top & -w_i\mathbf{X}_i{}^\top & y_i\mathbf{X}_i{}^\top \\ w_i\mathbf{X}_i{}^\top & \mathbf{0}^\top & -x_i\mathbf{X}_i{}^\top \end{array}\right] \begin{pmatrix} \mathbf{P}^1 \\ \mathbf{P}^2 \\ \mathbf{P}^3 \end{pmatrix} = \mathbf{0}$$

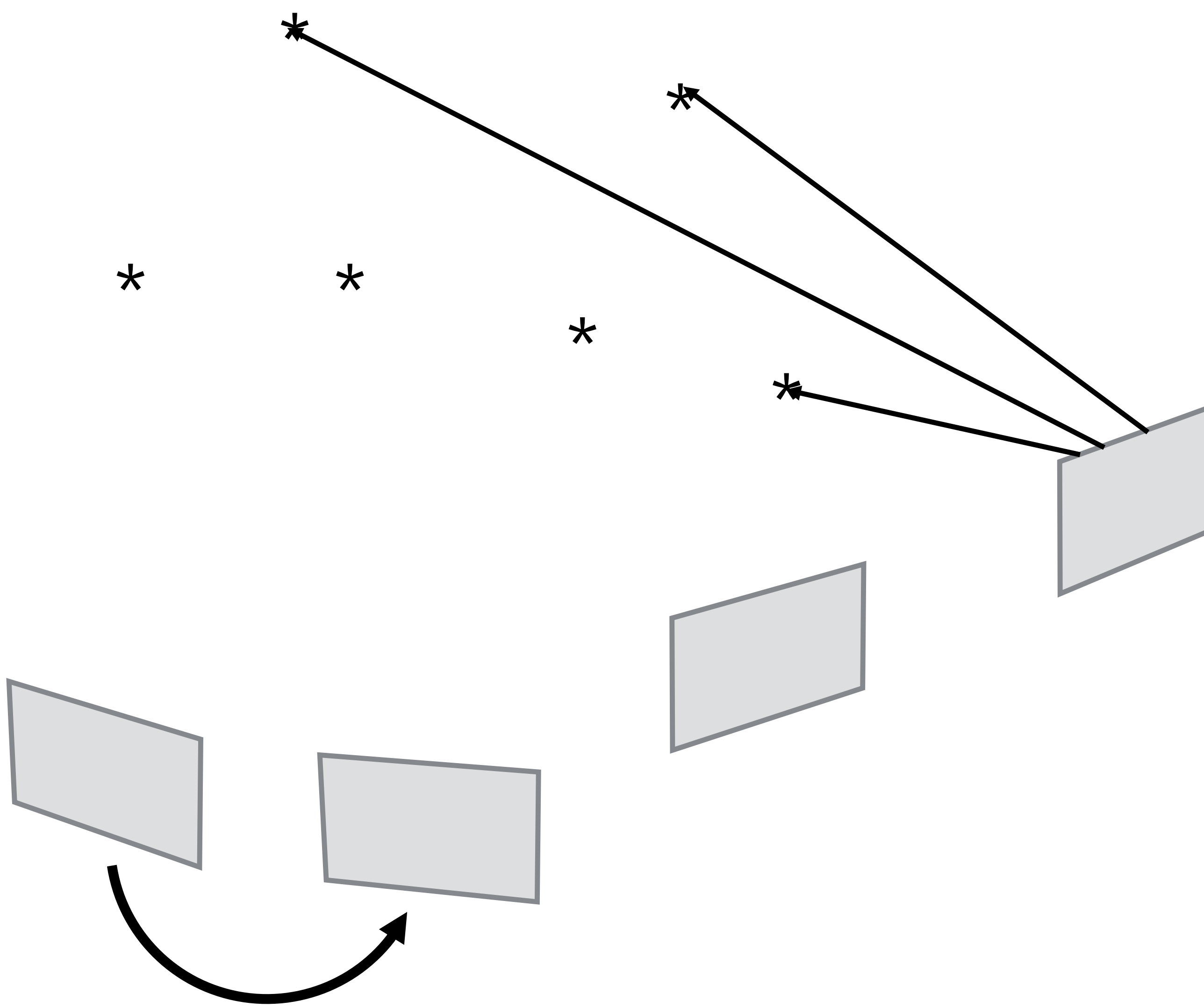
12 unknowns (11 DoF): 6 points for minimal solution

Linear least squares solution for >6 points

Don’t forget normalization (normalized DLT)

Degenerate if all points in single plane

Sequential Structure-from-Motion



$\mathtt{R}, \mathbf{t}$ with $||\mathbf{t}|| = 1$

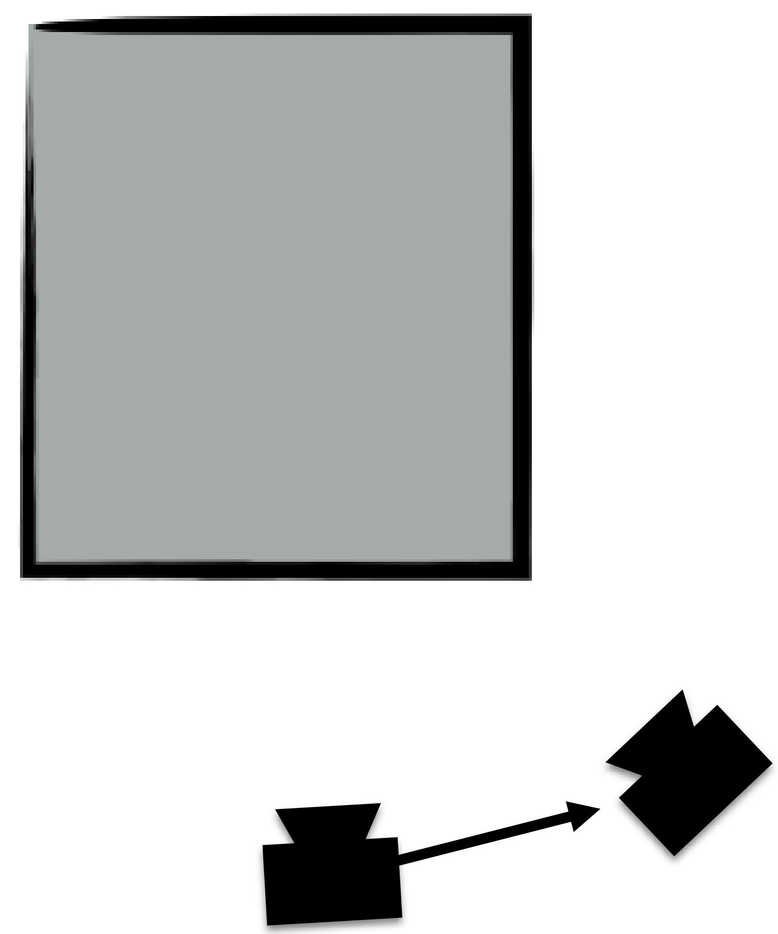
Initialize motion from two views

Initialize structure from two views

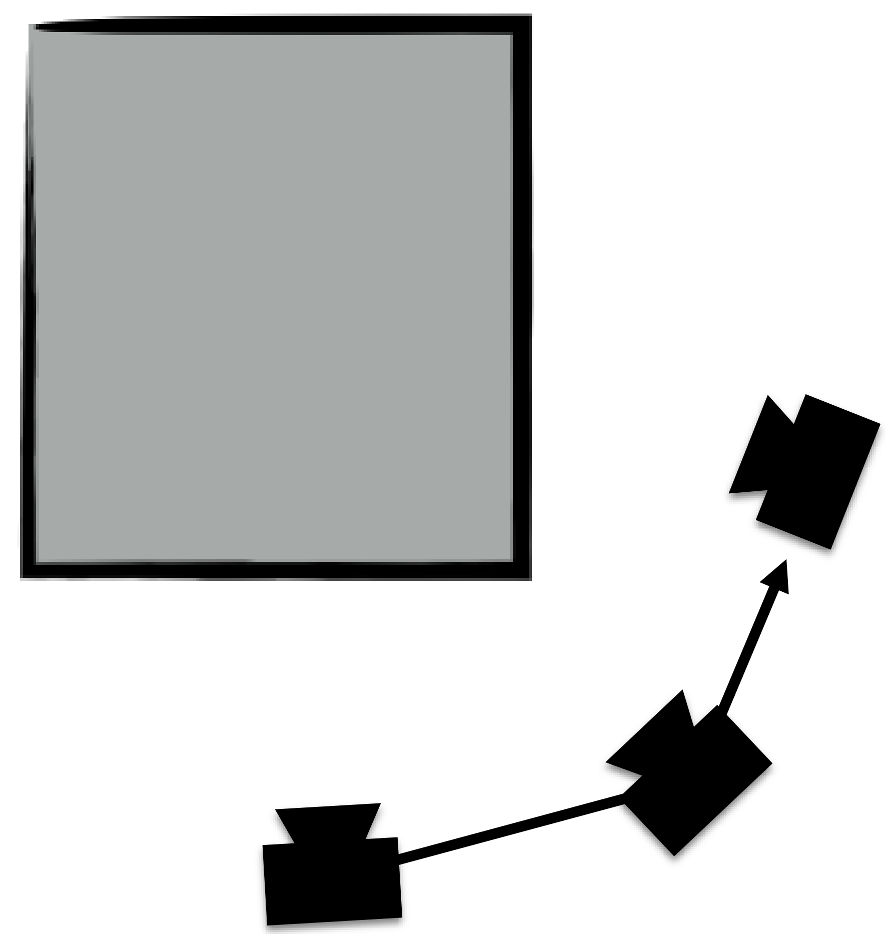
Extend motion

Extend structure

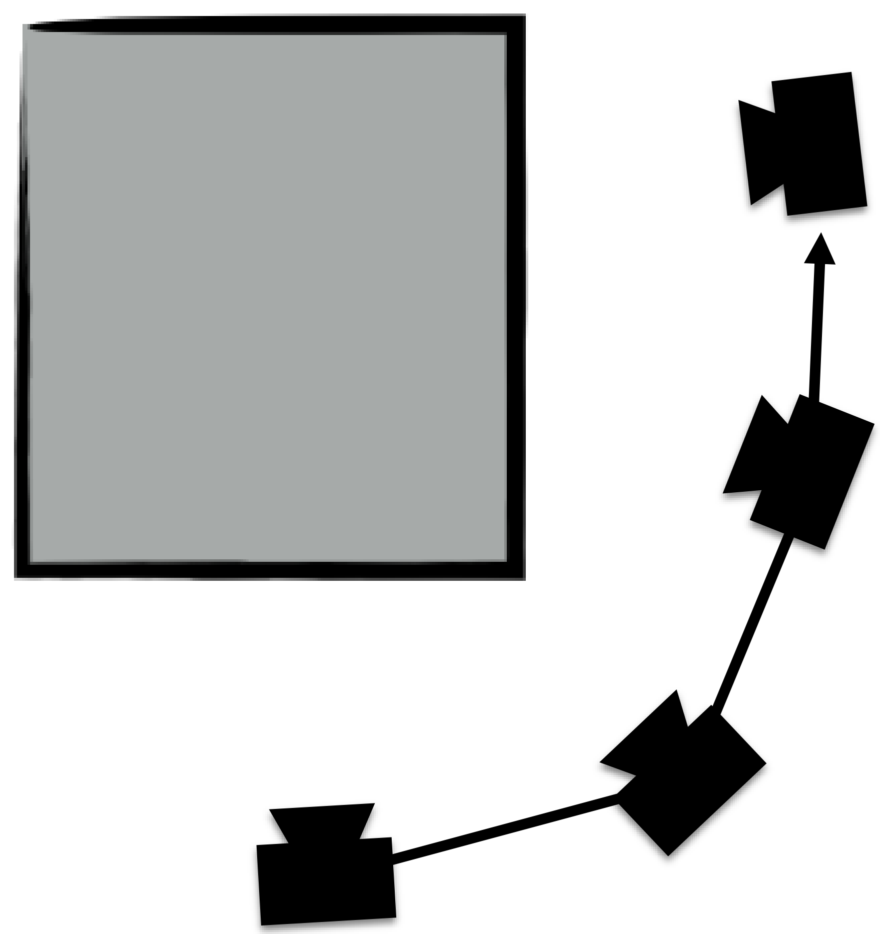
Sequential Structure-from-Motion



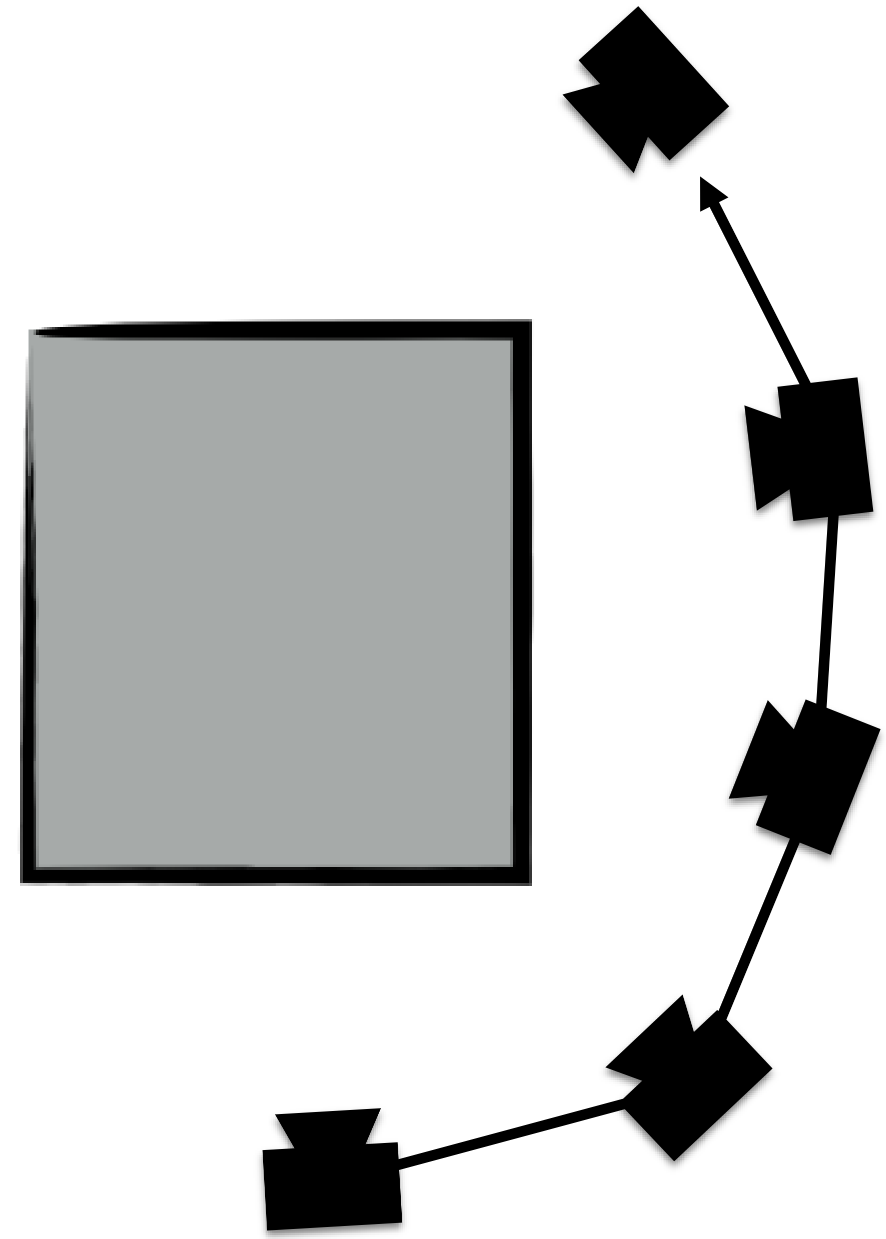
Sequential Structure-from-Motion



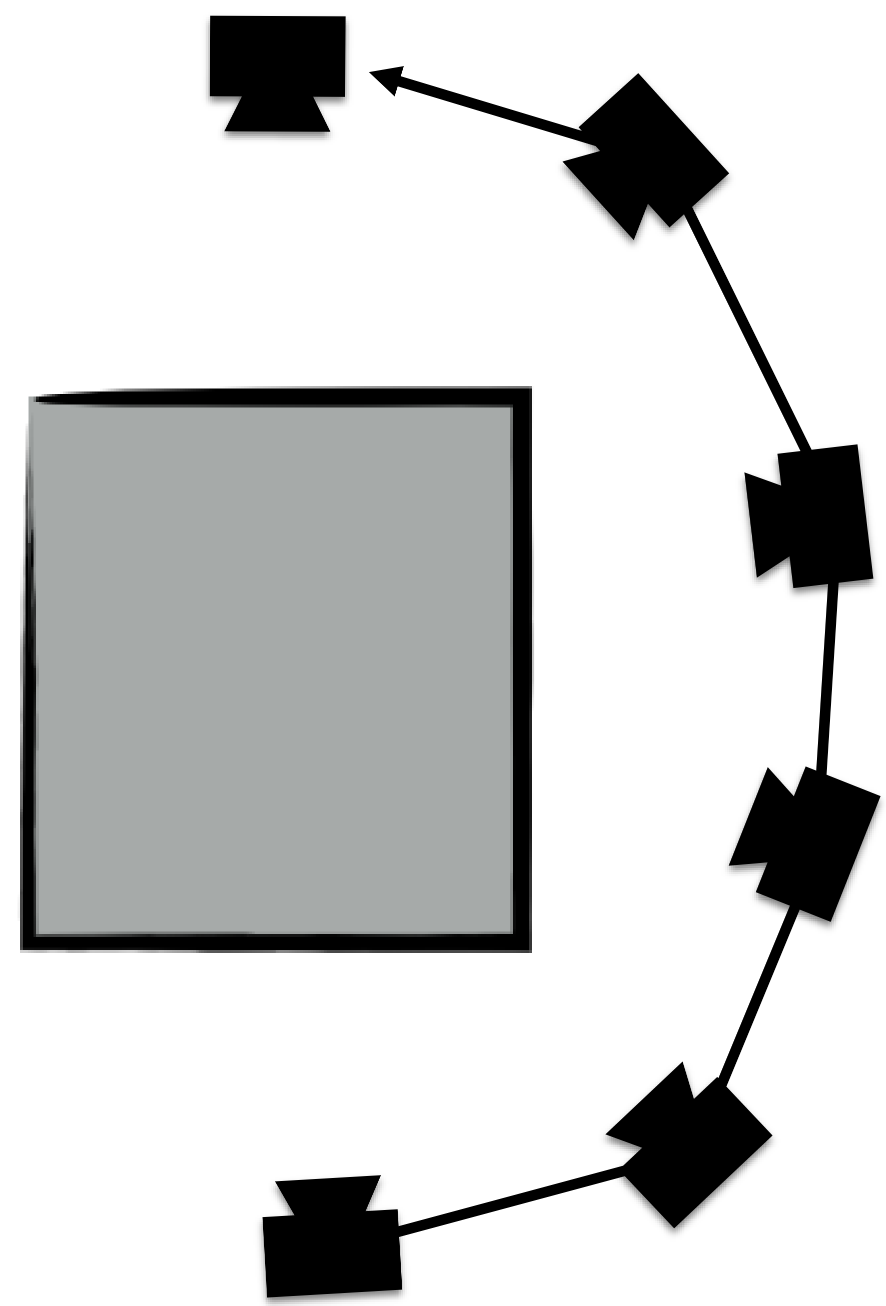
Sequential Structure-from-Motion



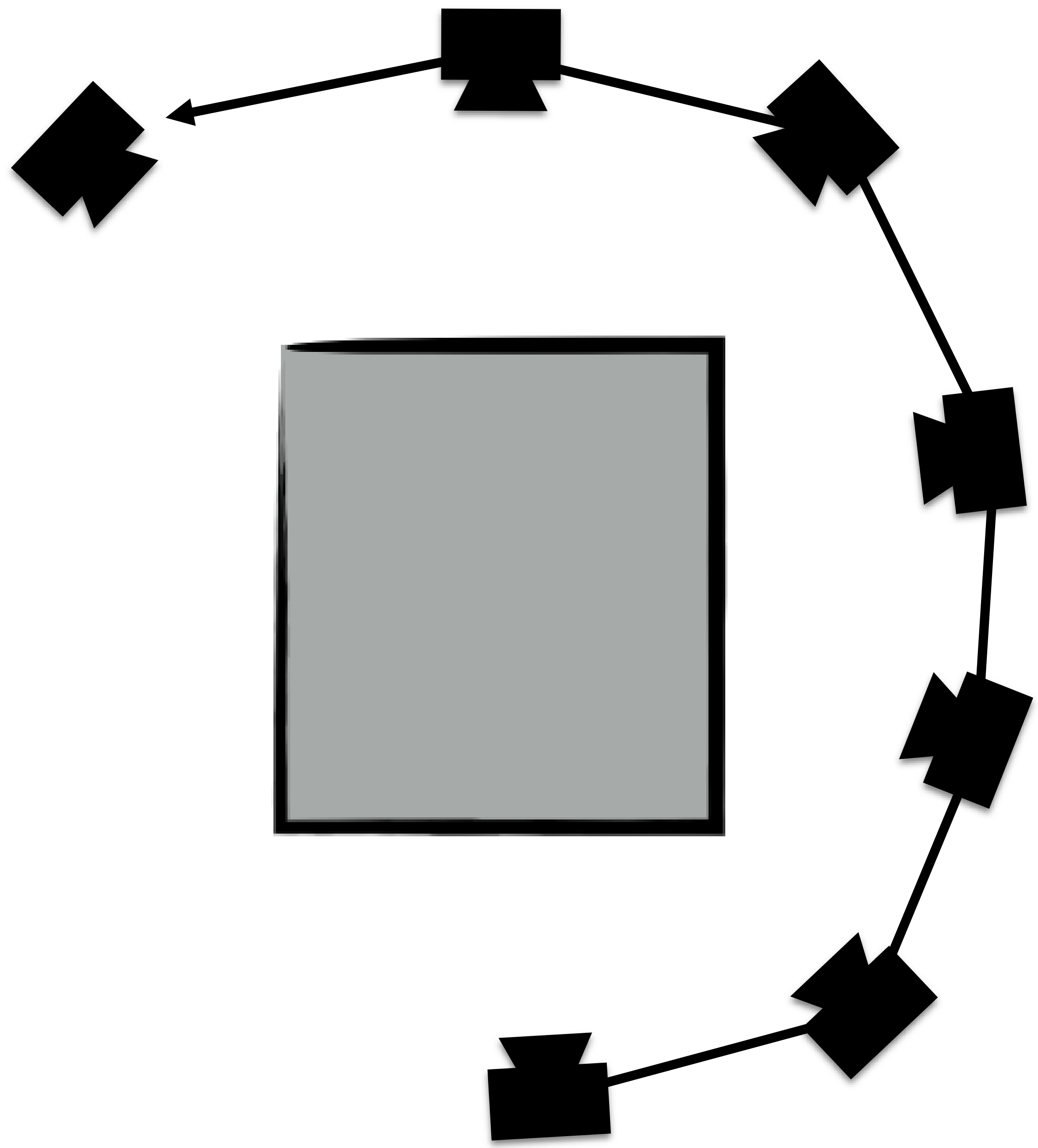
Sequential Structure-from-Motion



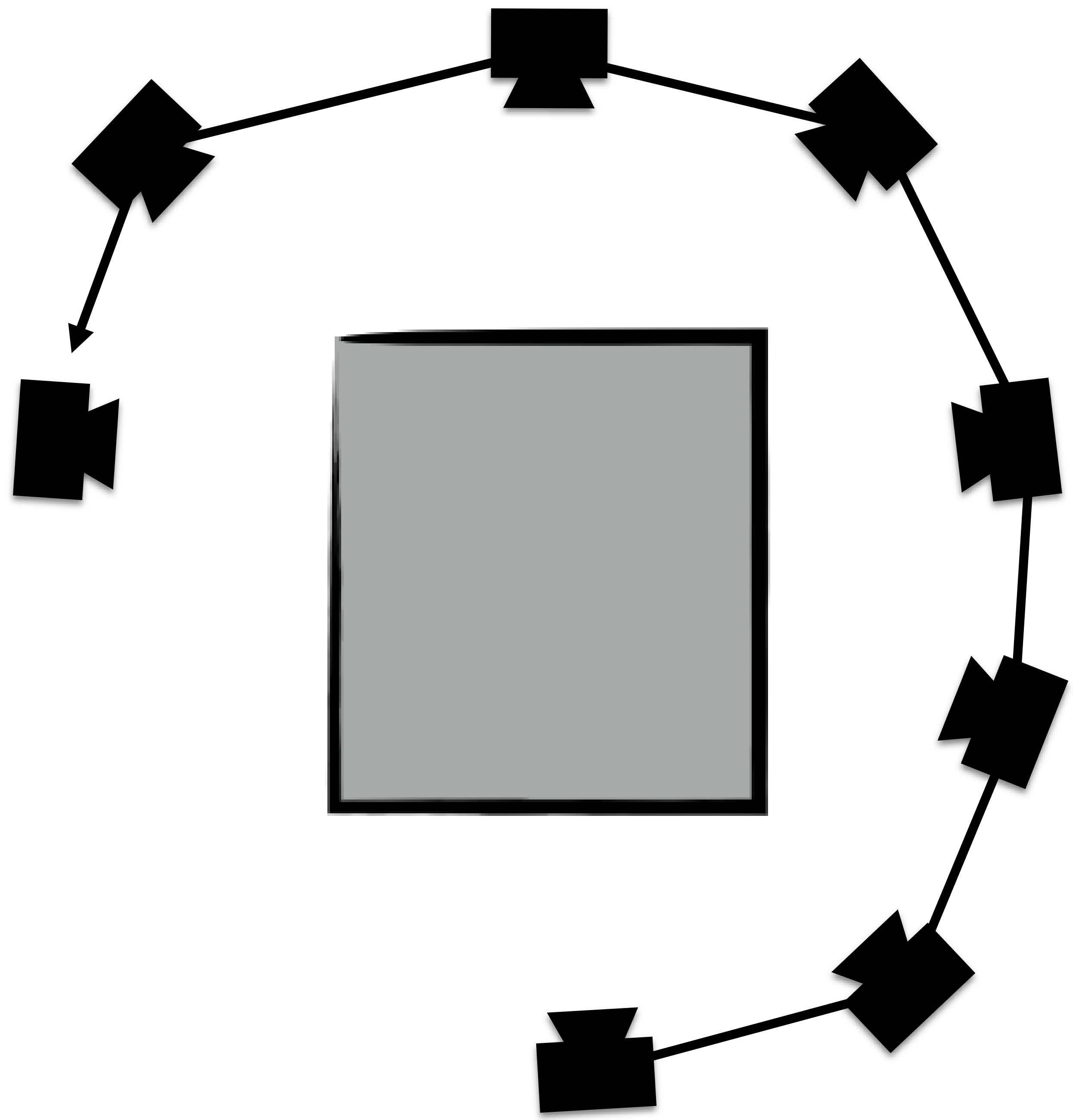
Sequential Structure-from-Motion



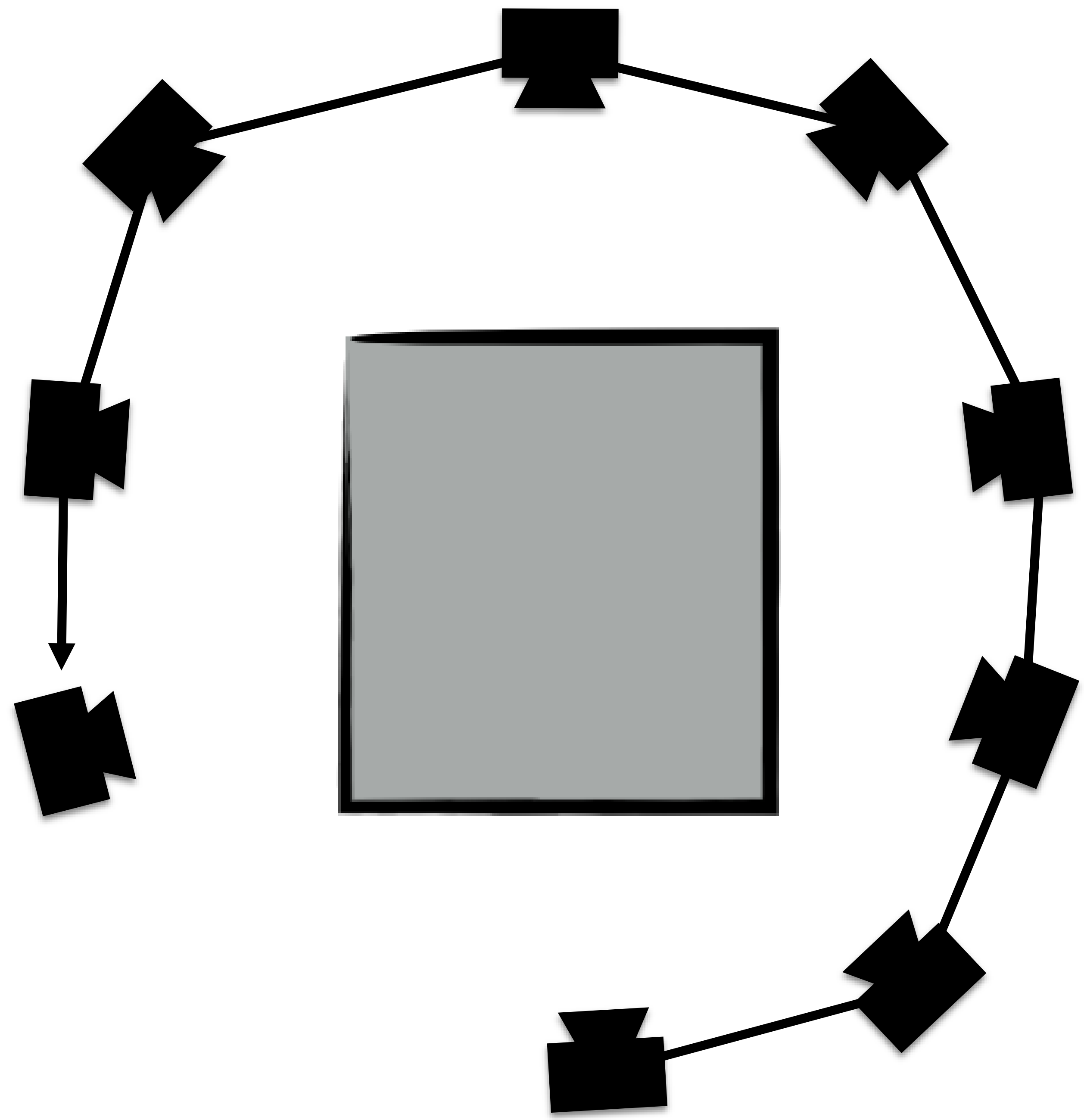
Sequential Structure-from-Motion



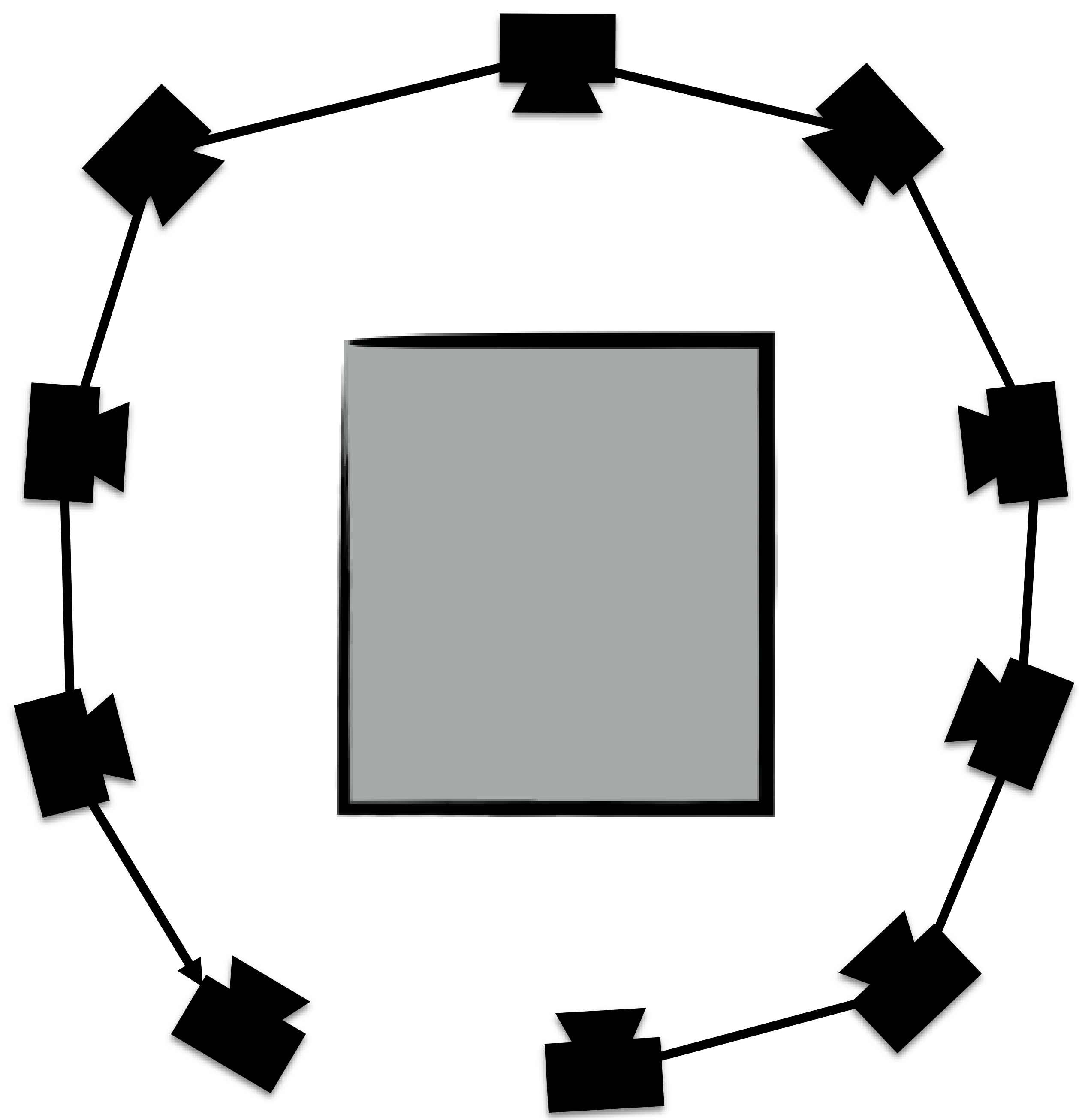
Sequential Structure-from-Motion



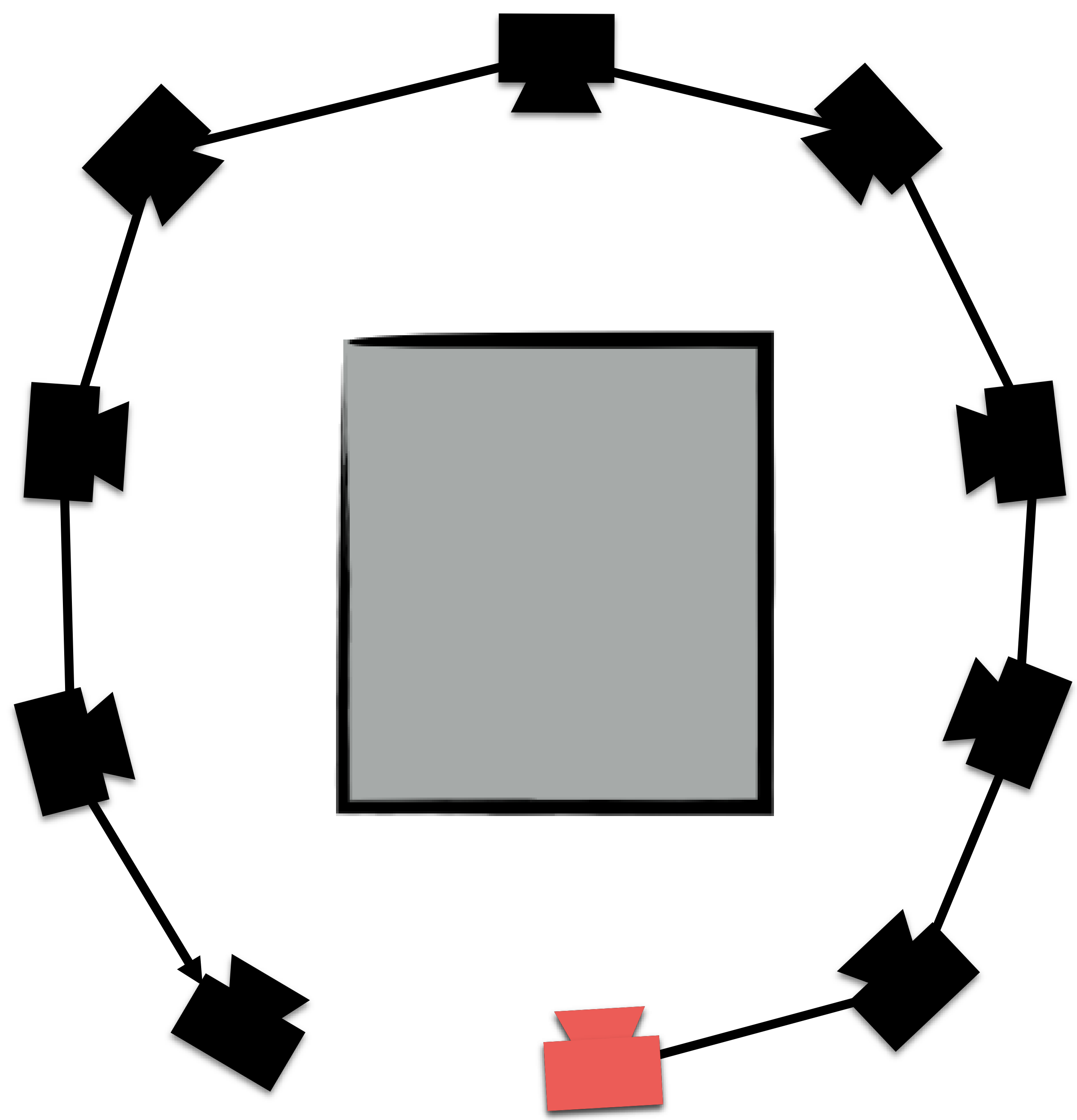
Sequential Structure-from-Motion



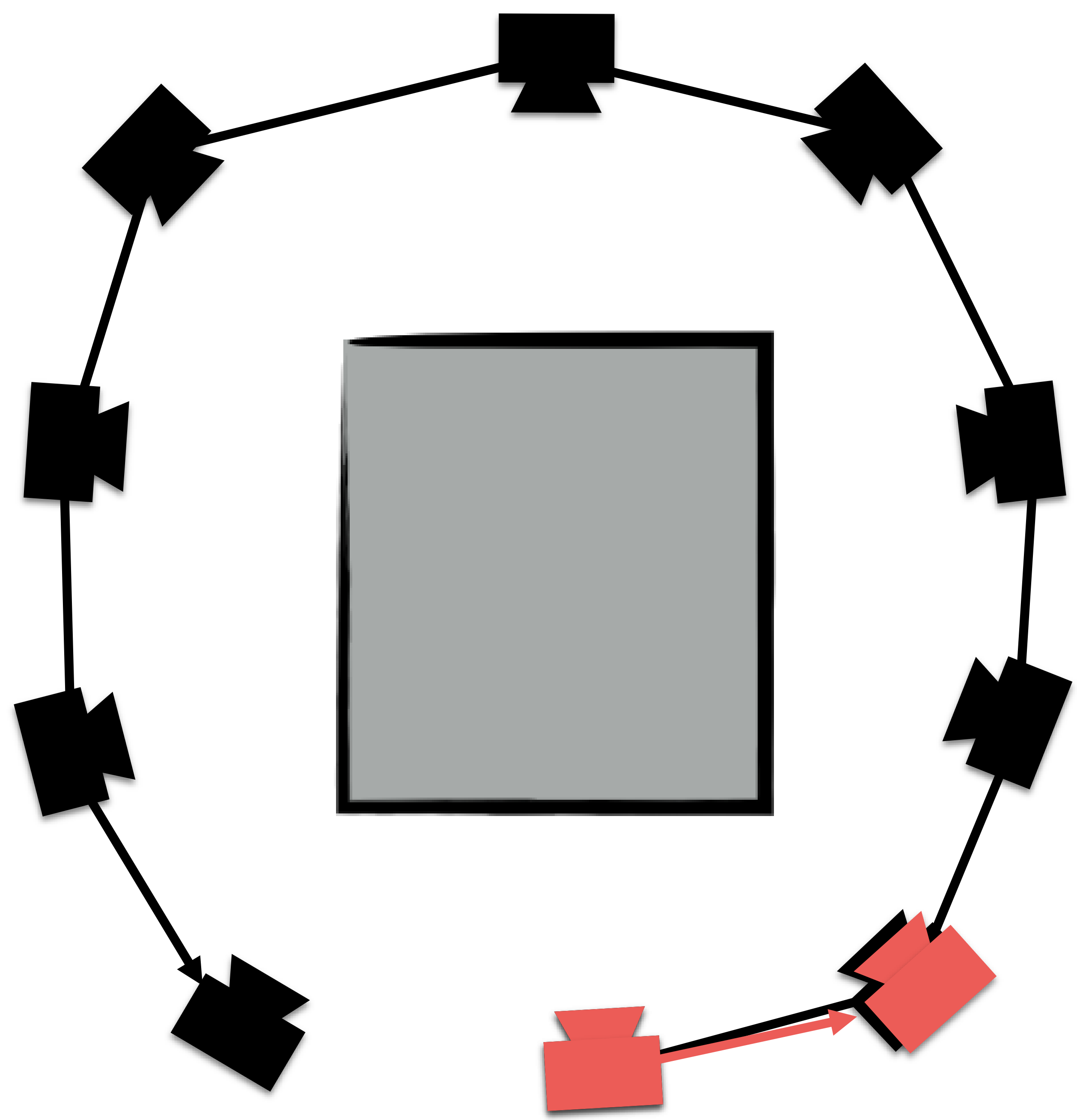
Sequential Structure-from-Motion



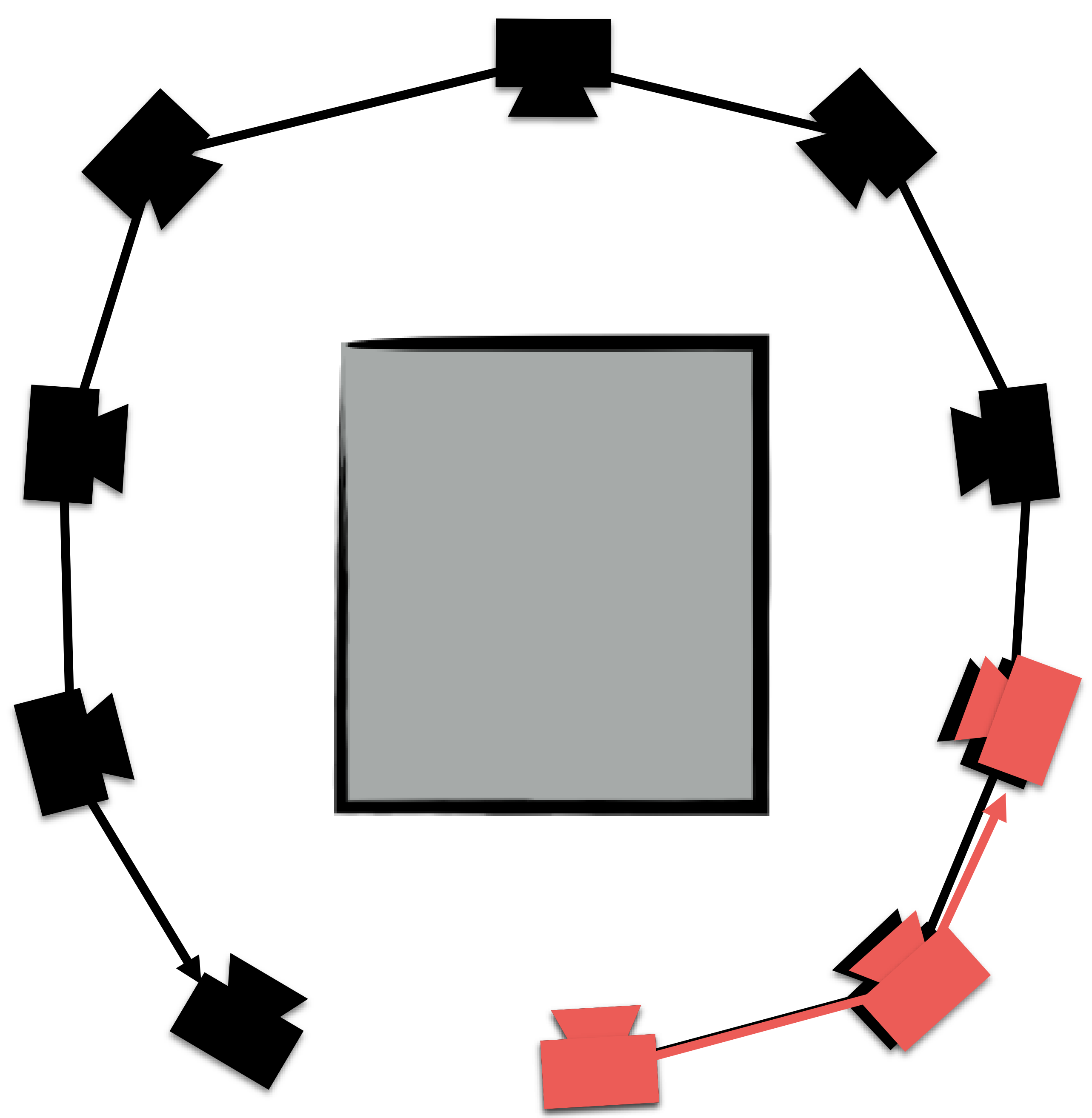
Sequential Structure-from-Motion



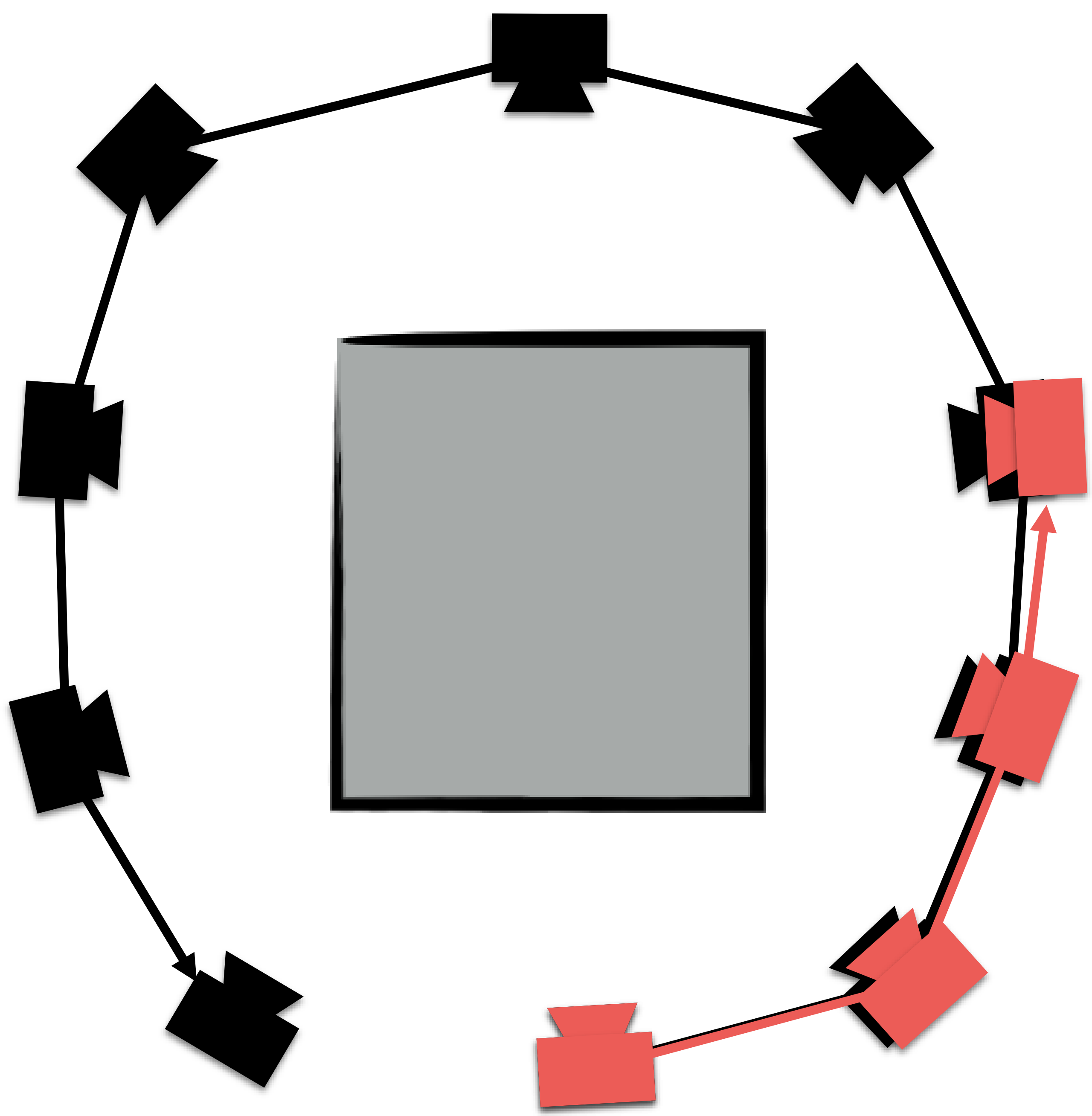
Sequential Structure-from-Motion



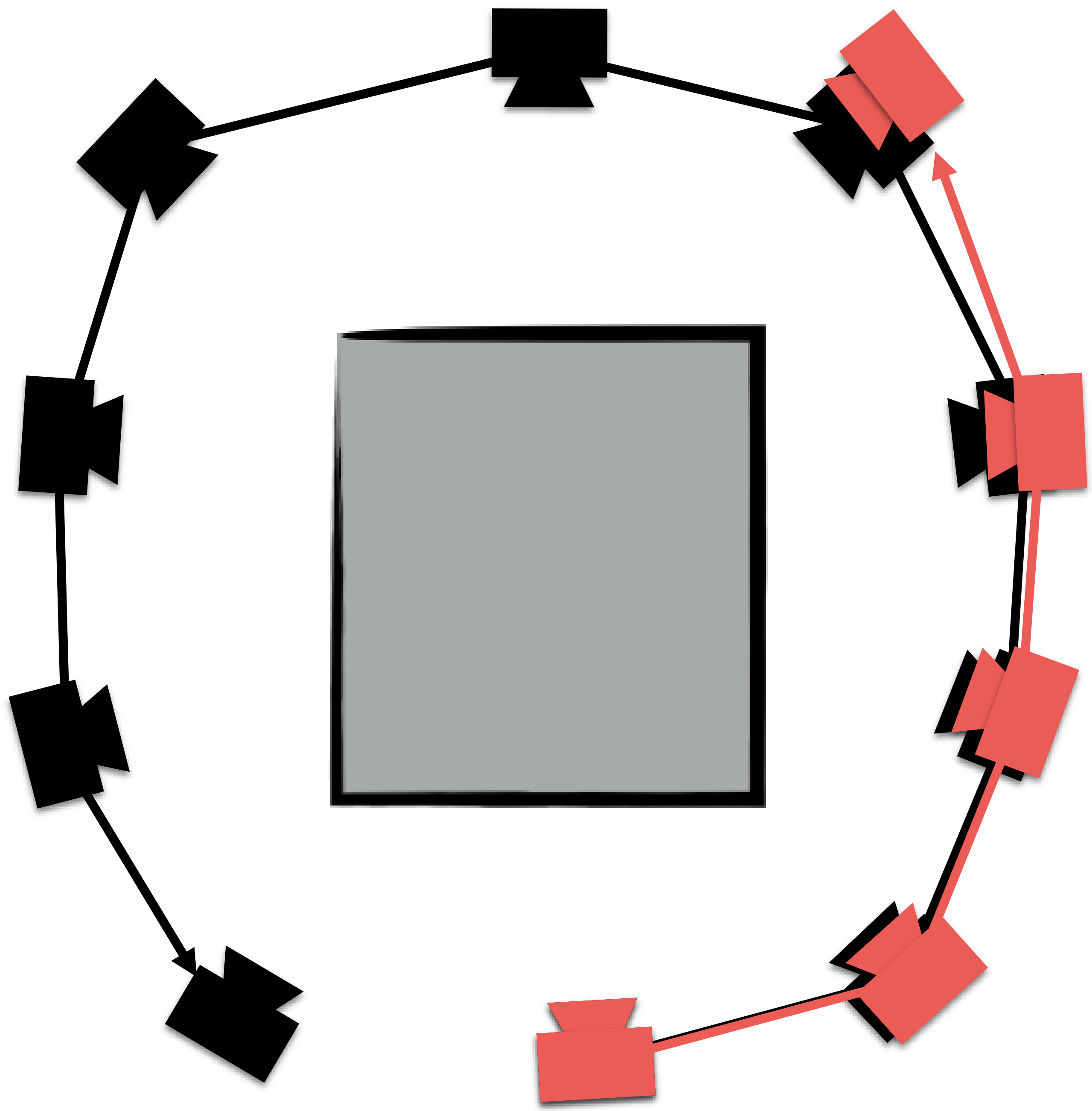
Sequential Structure-from-Motion



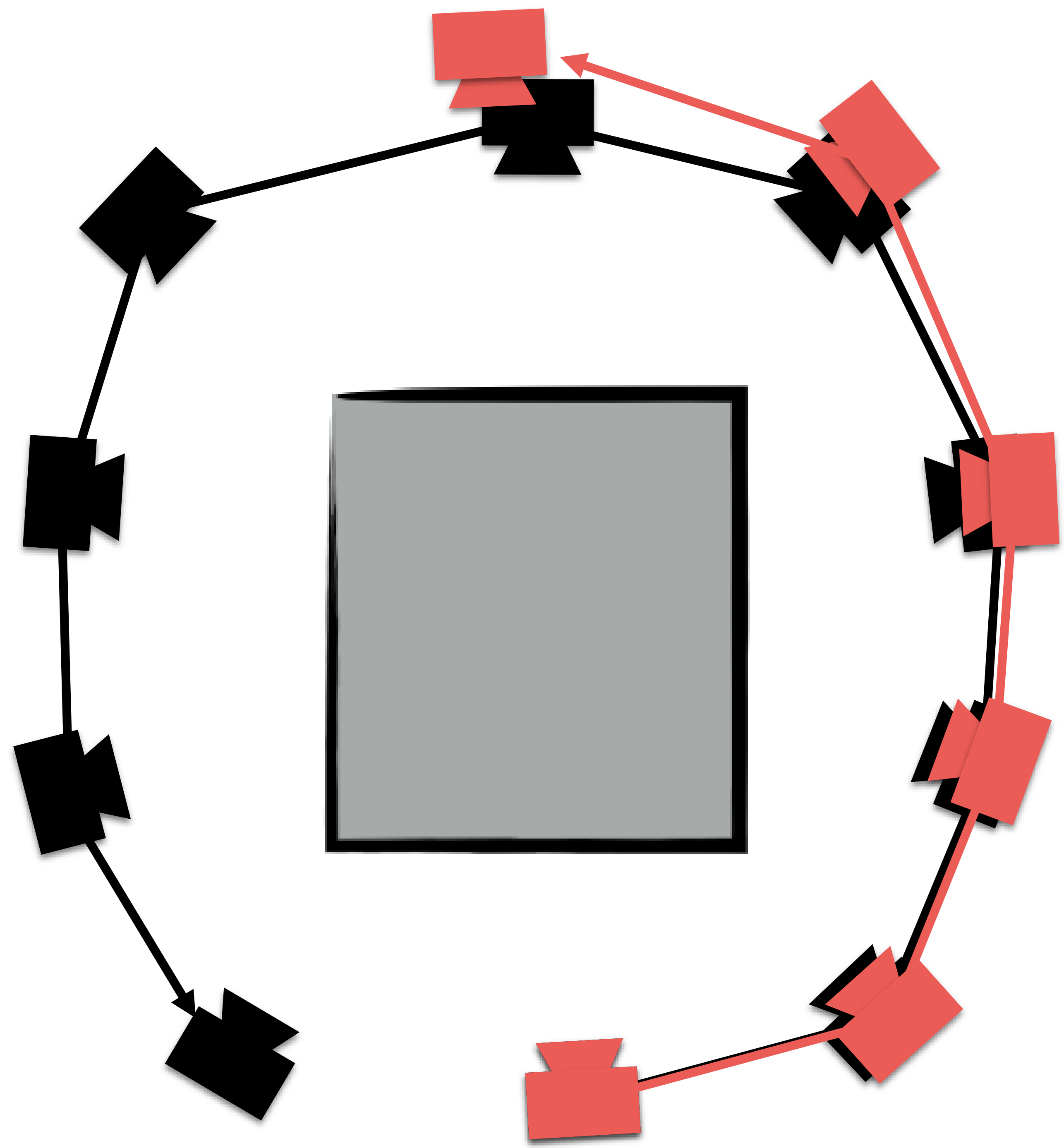
Sequential Structure-from-Motion



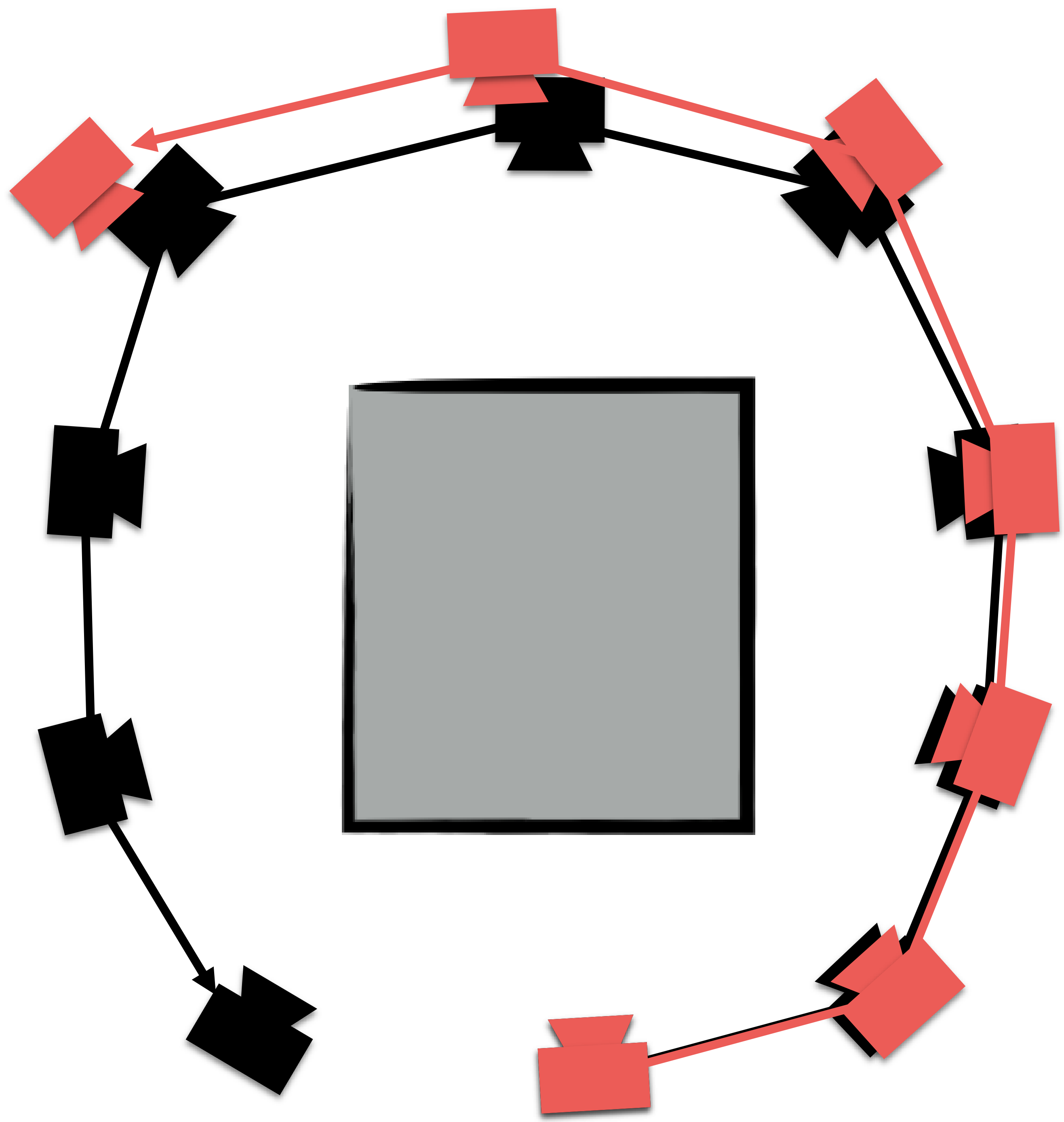
Sequential Structure-from-Motion



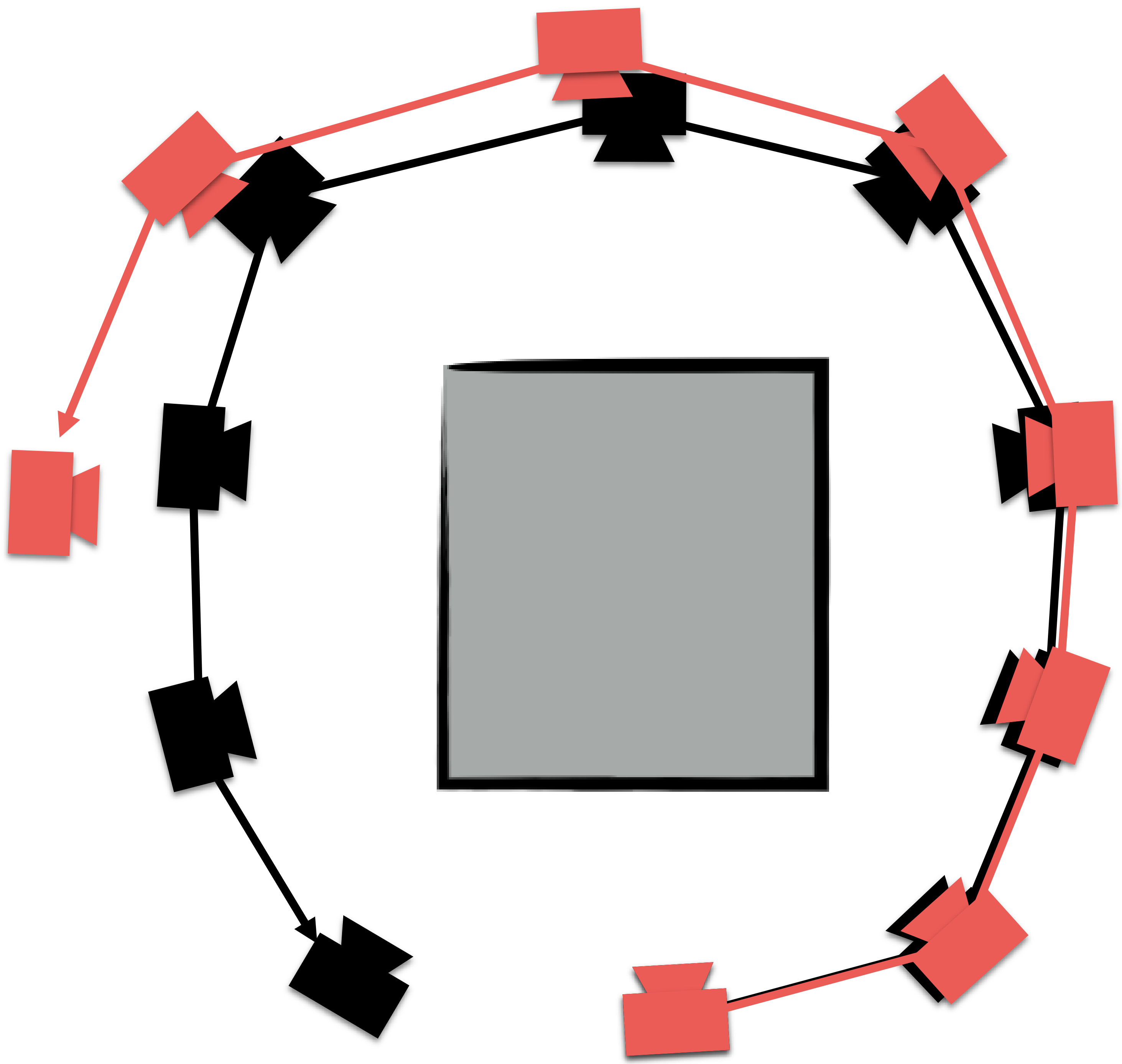
Sequential Structure-from-Motion



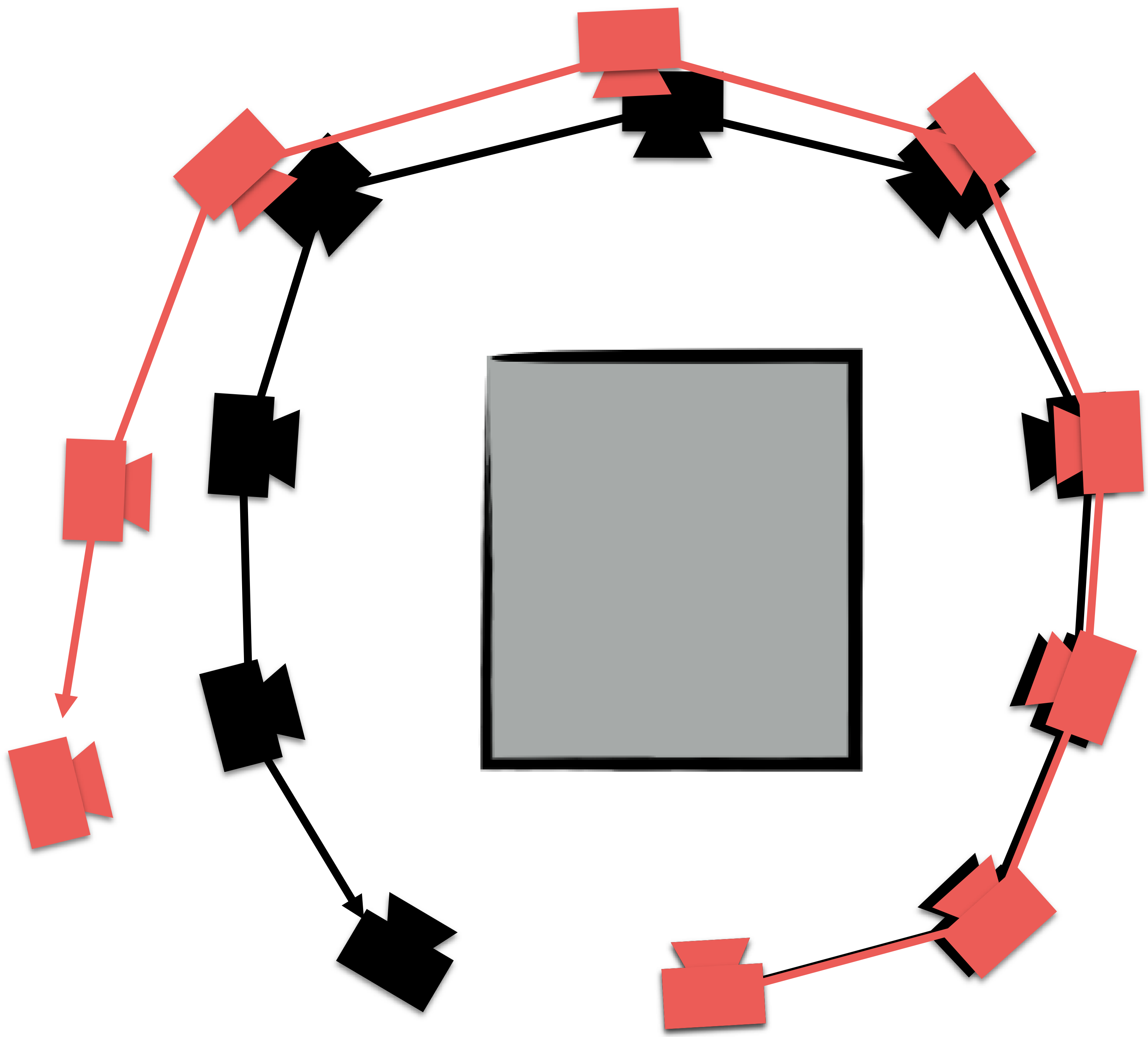
Sequential Structure-from-Motion



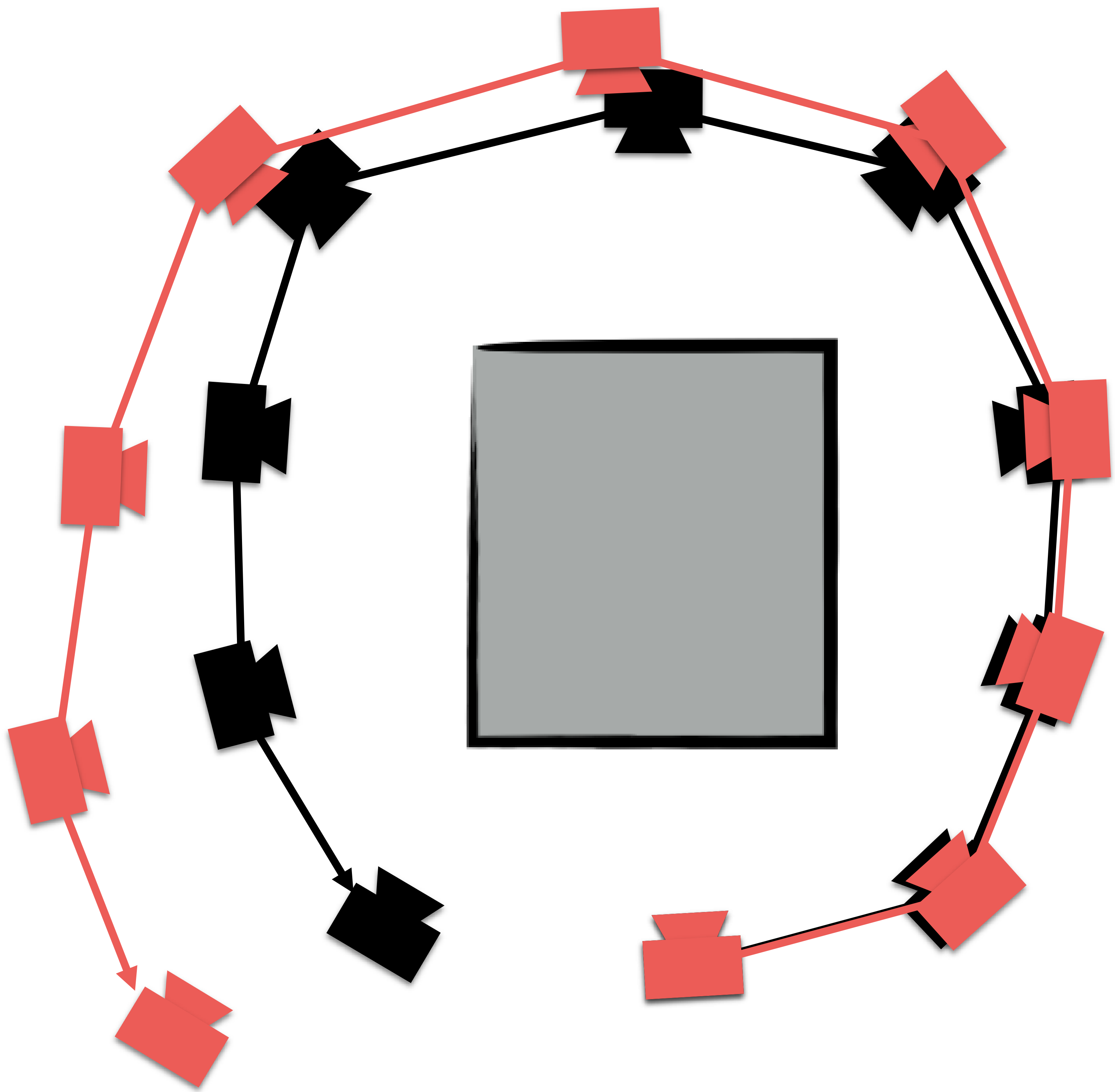
Sequential Structure-from-Motion



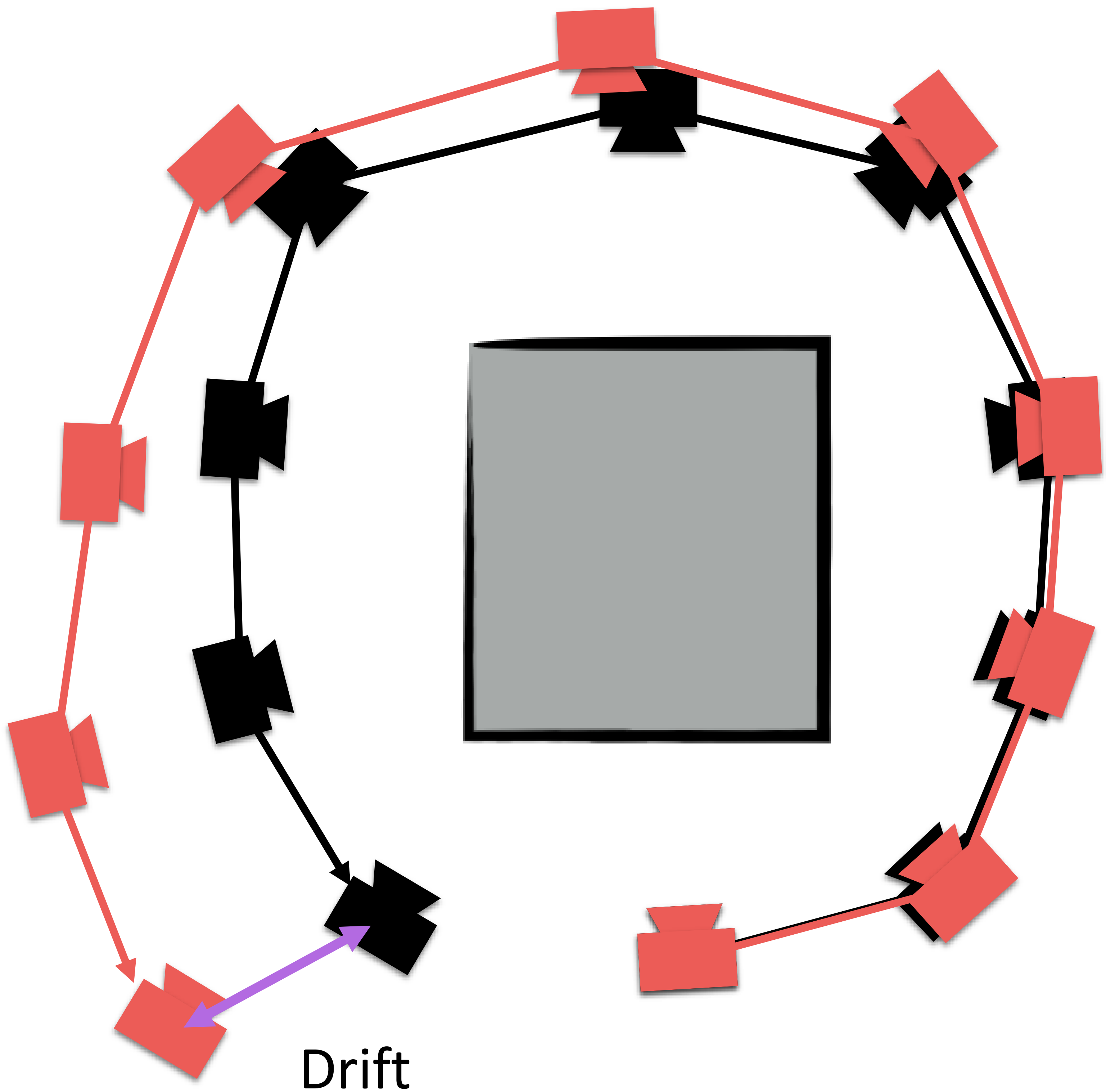
Sequential Structure-from-Motion



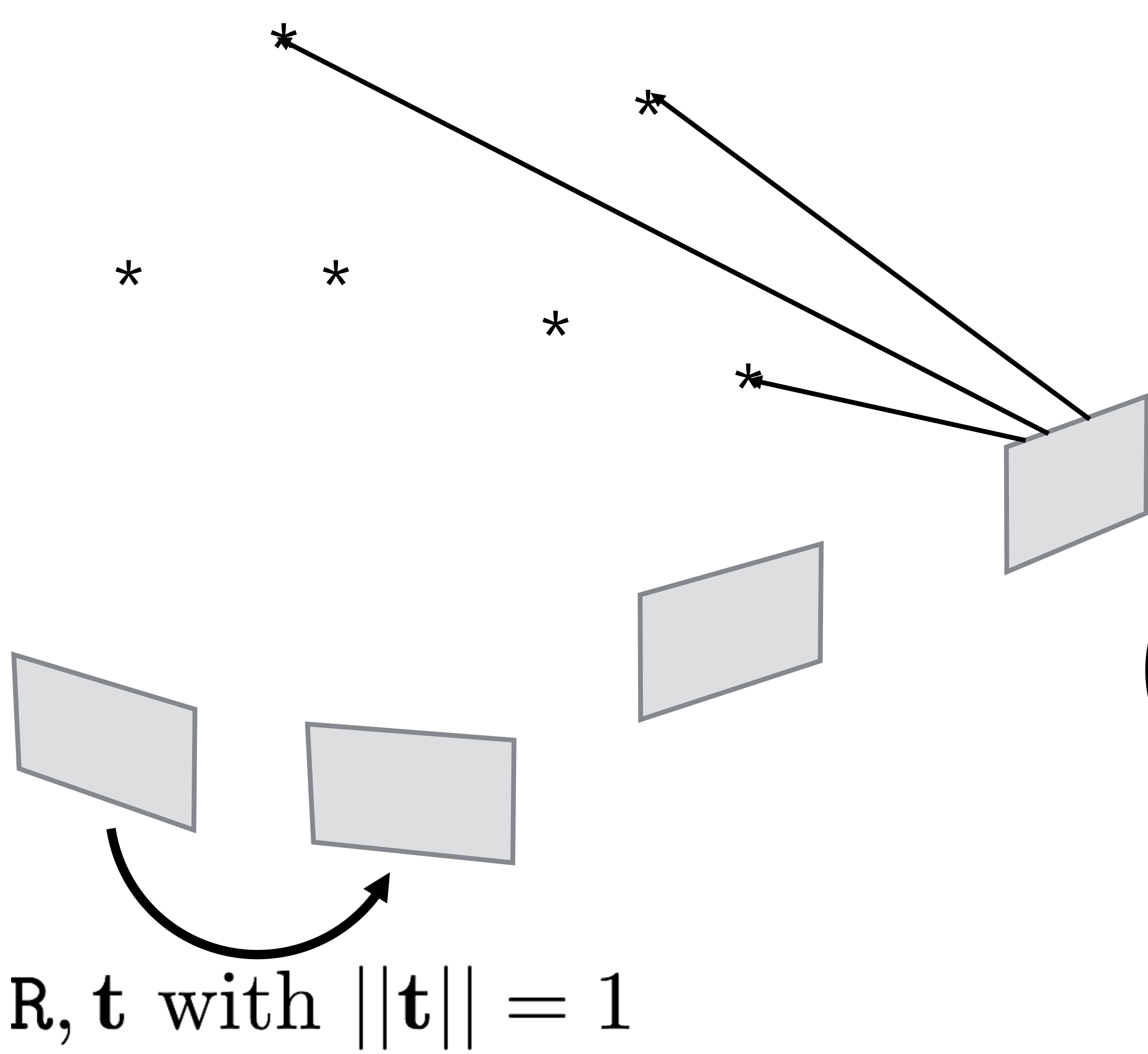
Sequential Structure-from-Motion

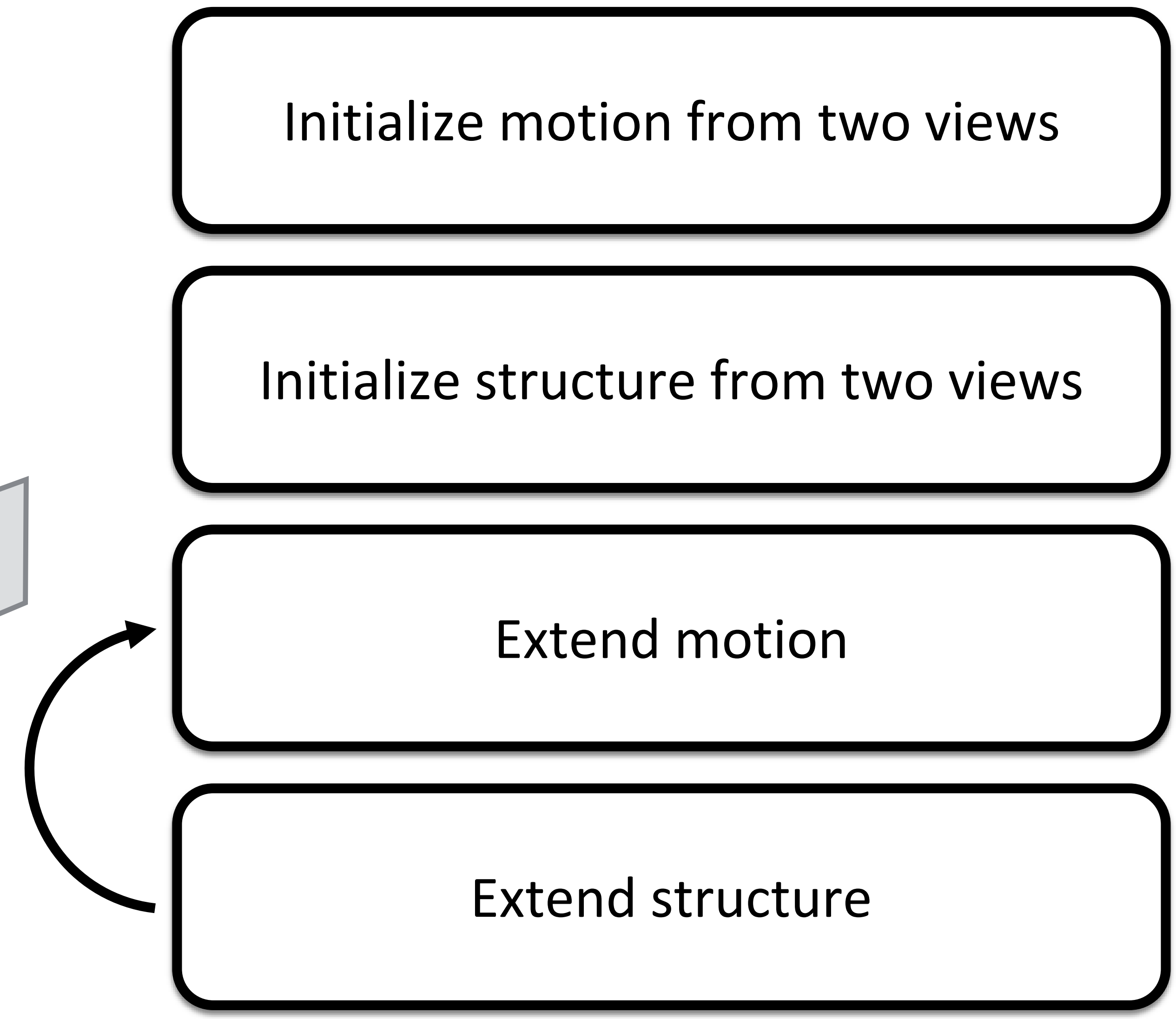


Sequential Structure-from-Motion

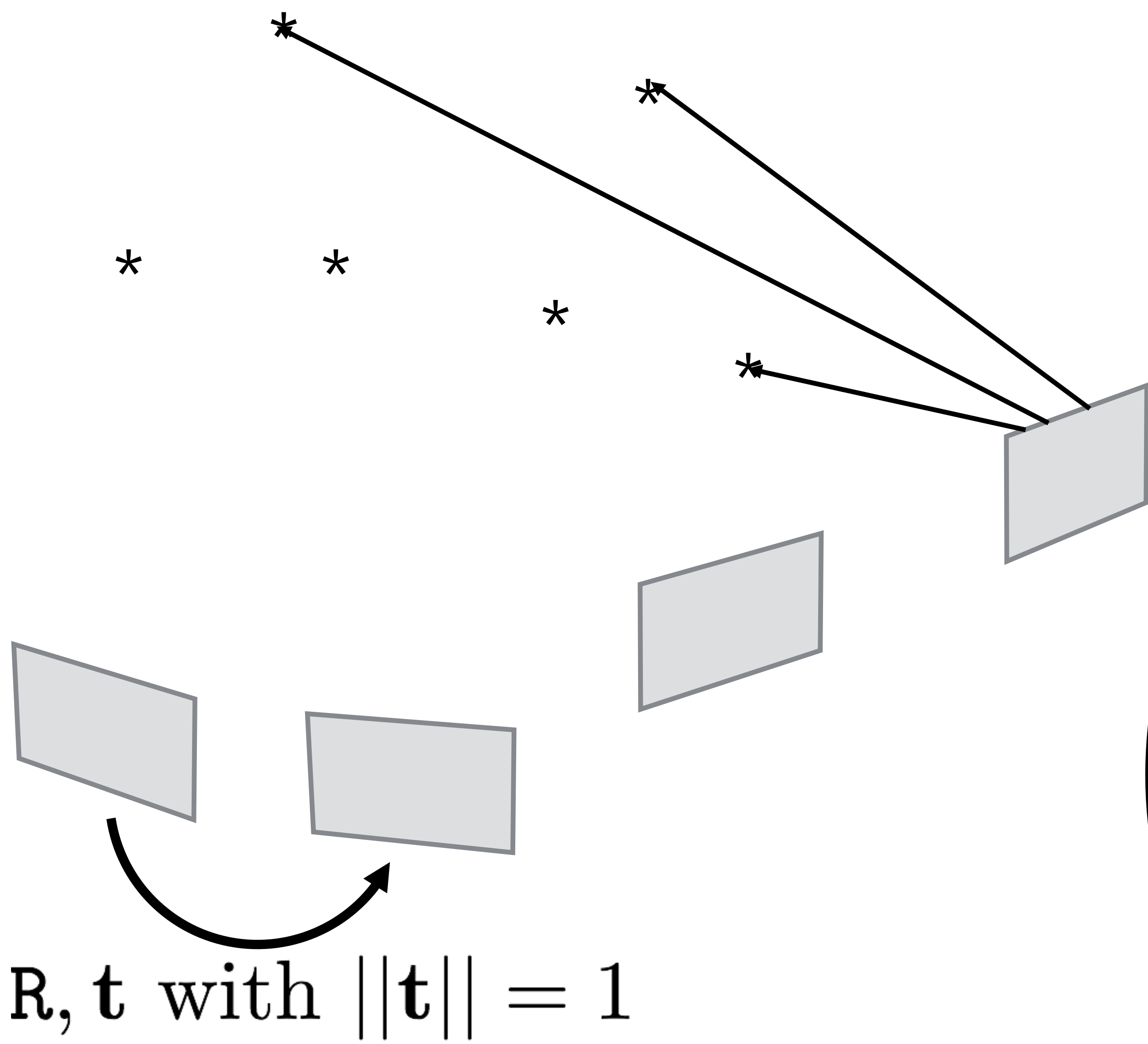


Sequential Structure-from-Motion





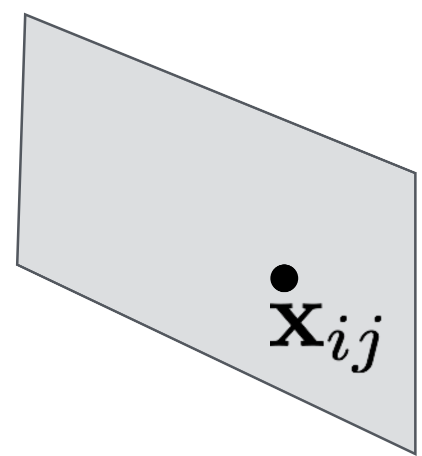
Sequential Structure-from-Motion



- Initialize motion from two views
- Initialize structure from two views
- Extend motion
- Extend structure
- **Bundle Adjustment**

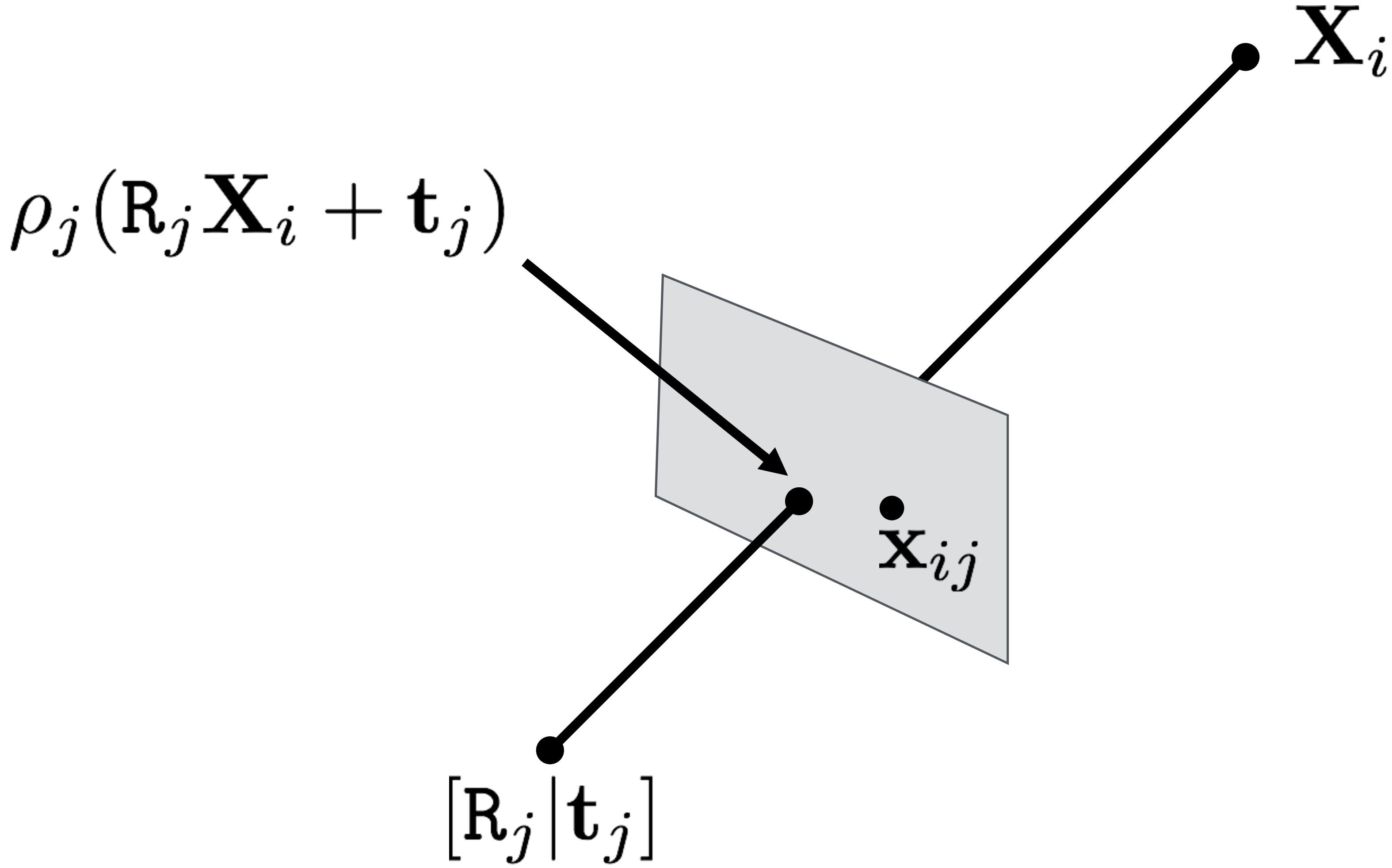
Bundle Adjustment

$\mathbf{X}_i$

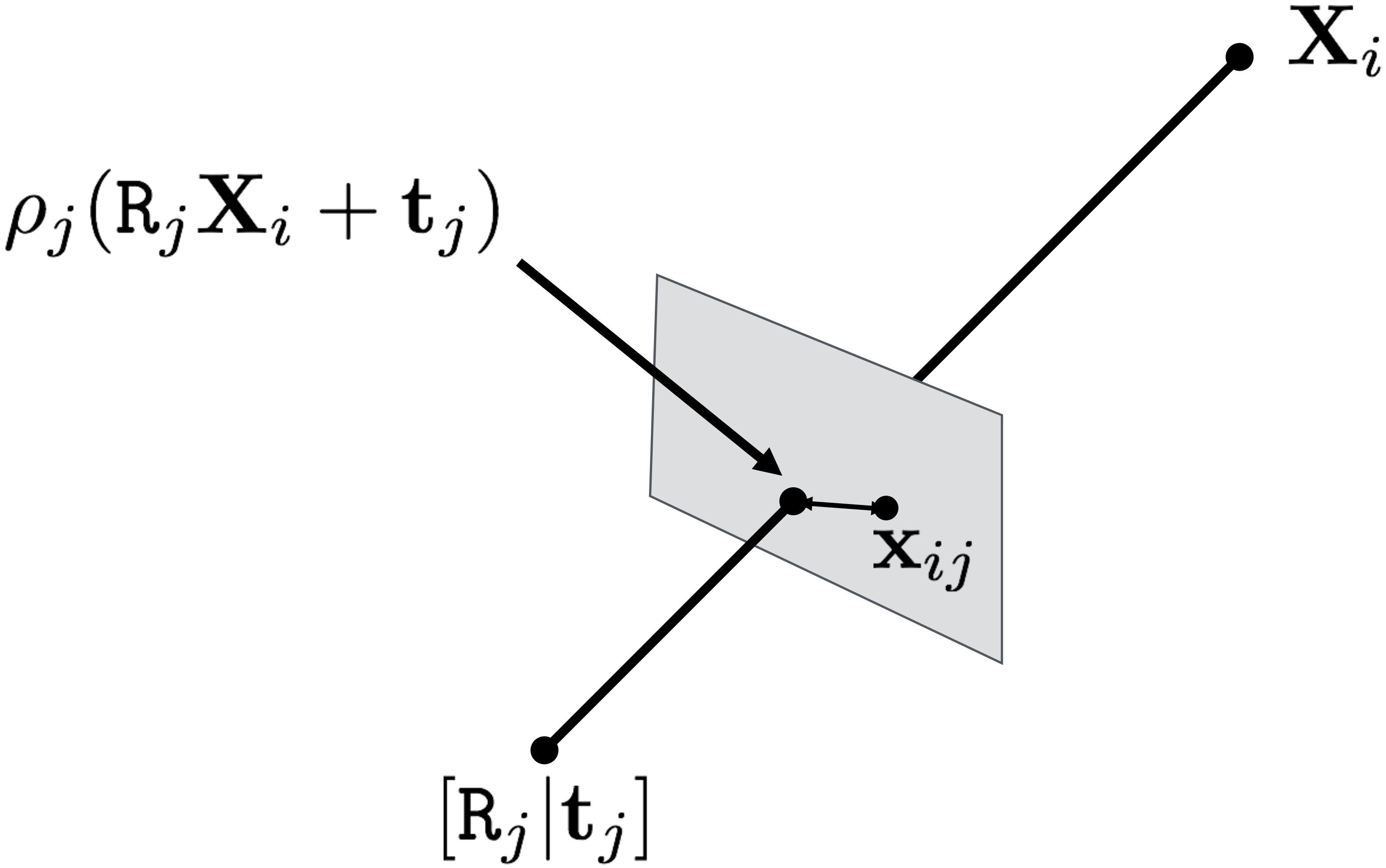


$[\mathrm{R}_j|\mathbf{t}_j]$

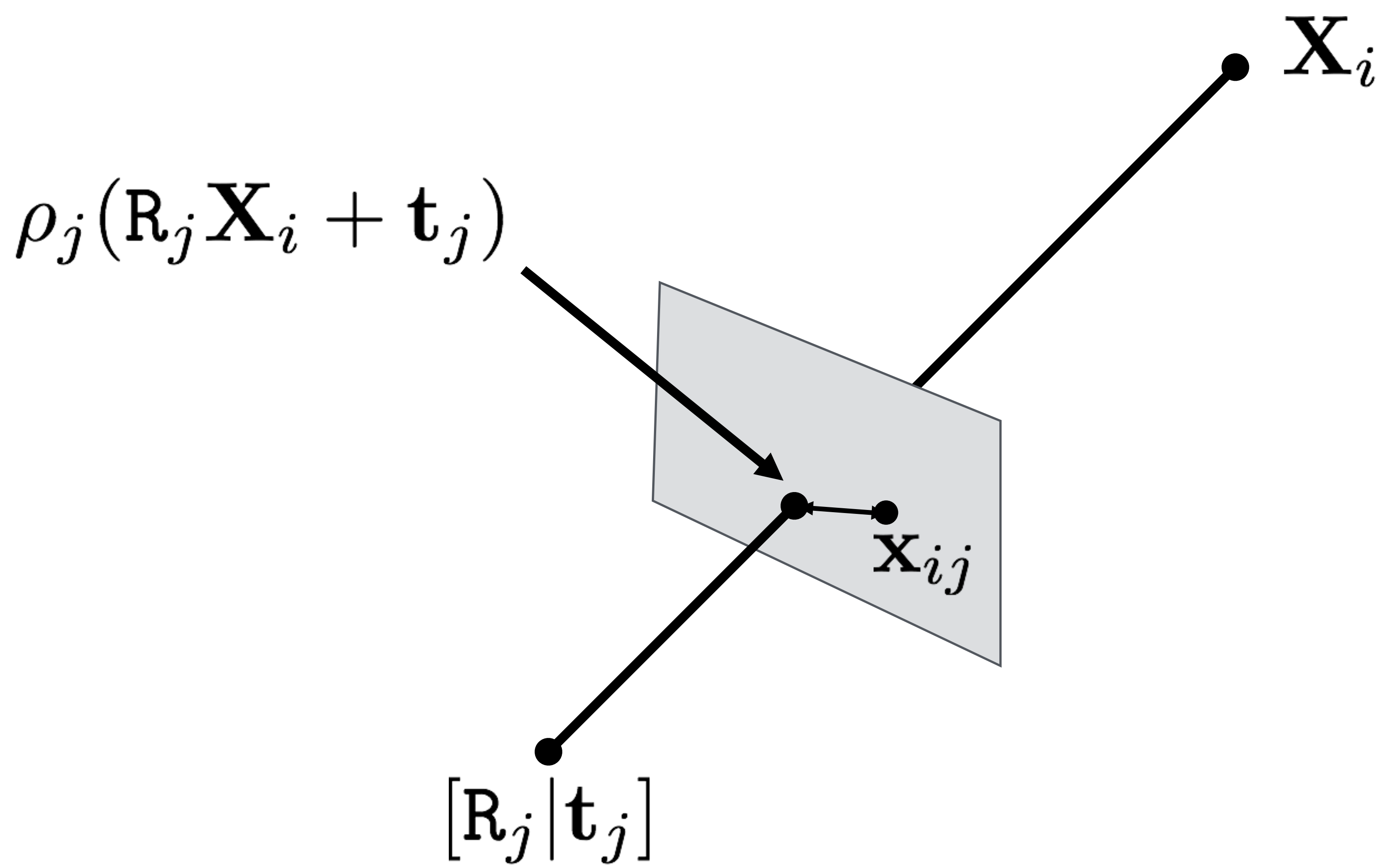
Bundle Adjustment



Bundle Adjustment

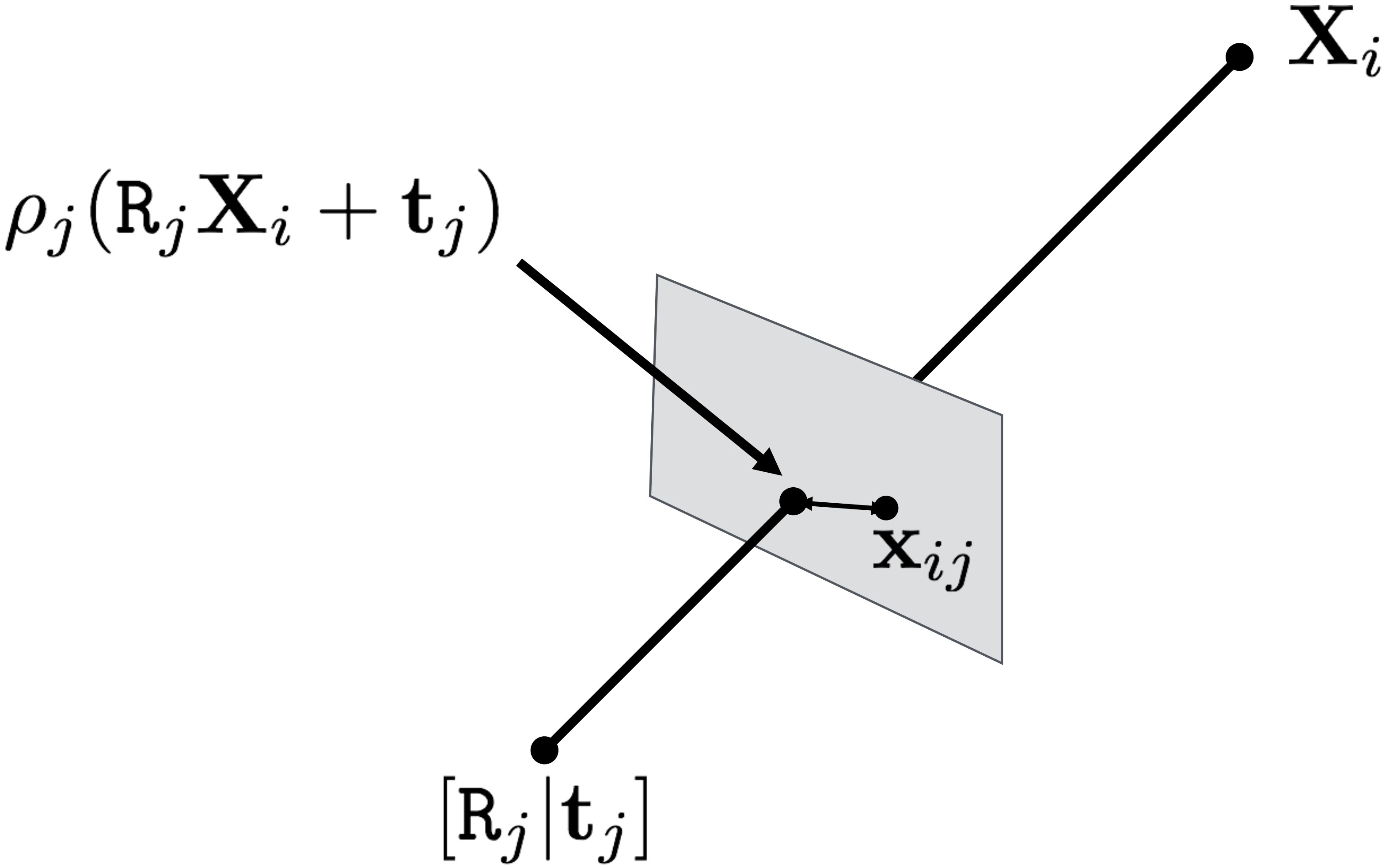


Bundle Adjustment



$$\underset{\text{camera poses, 3D points}}{\operatorname{argmin}} \sum_i \sum_j \Delta_{ij} ||\mathbf{x}_{ij} - \rho_j(\mathtt{R}_j\mathbf{X}_i + \mathbf{t}_j)||^2$$

Bundle Adjustment



$$\underset{\text{camera poses, 3D points}}{\operatorname{argmin}} \sum_i \sum_j \Delta_{ij} ||\mathbf{x}_{ij} - \rho_j(\mathtt{R}_j\mathbf{X}_i + \mathbf{t}_j)||^2$$

point i visible in image j?

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

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

$\mathrm{J} = \frac{\partial f(\mathbf{X})}{\partial \mathbf{X}}$: Jacobian

η : Step size

Slow convergence near minimum point!

slide credit: Gim Hee Lee

Newton’s Method

2nd order approximation (quadratic Taylor expansion):

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

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

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

slide credit: Gim Hee 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 (2^{nd} order derivatives) and optimization might get stuck at saddle point!

slide credit: Gim Hee 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: Gim Hee Lee

Levenberg-Marquardt

Regularized Gauss-Newton with damping factor

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

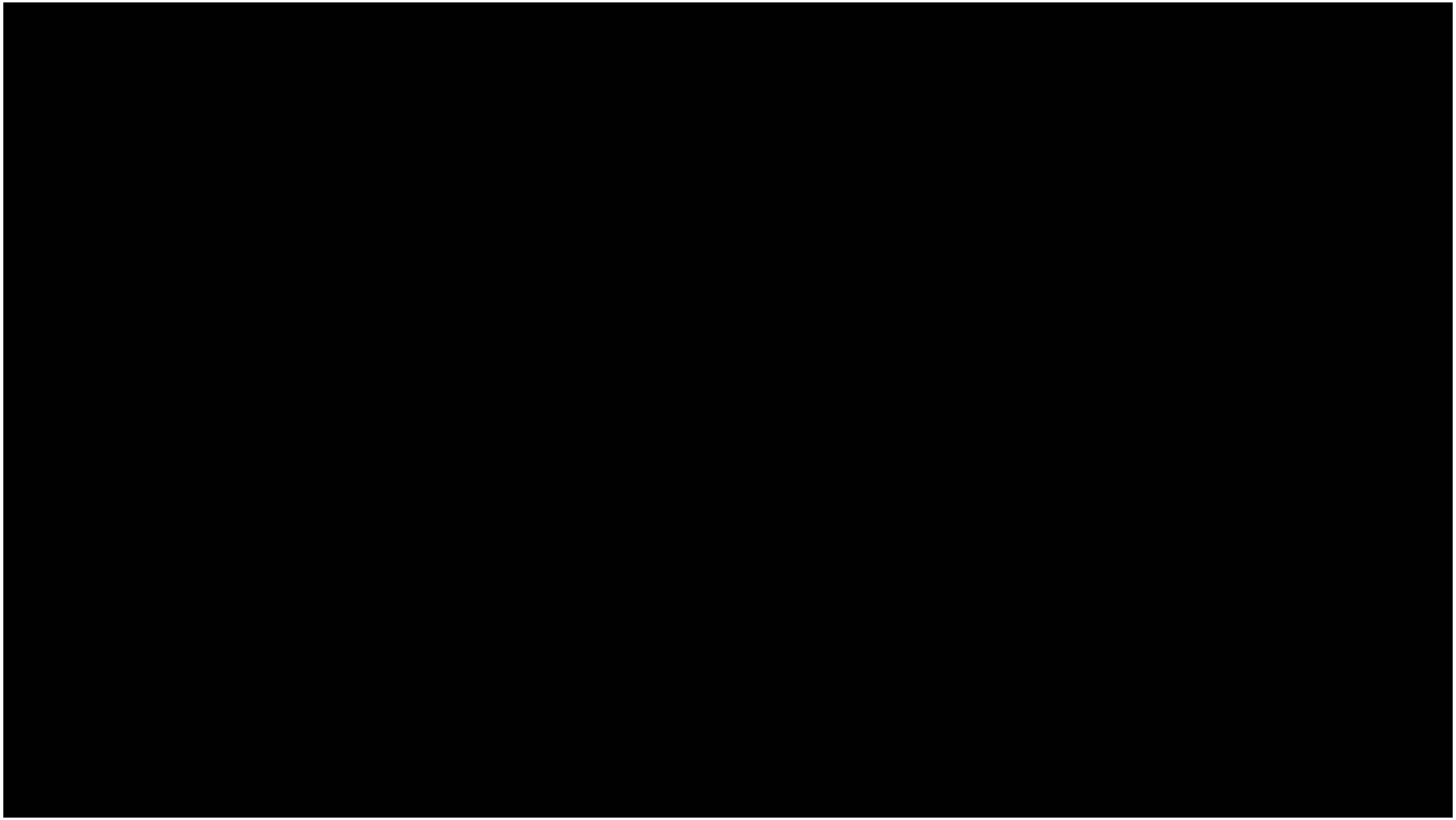
$\lambda \to 0$: Gauss-Newton (when convergence is rapid)
$\lambda \to \infty$: Gradient descent (when convergence is slow)

Adapt λ during optimization:

- Decrease λ when function value decreases
- Increase λ otherwise

slide credit: Gim Hee Lee

Bundle Adjustment



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

Next Lecture

Jan. 20	Introduction, Linear classifiers and filtering	
Jan. 23	Filtering, gradients, scale	Lab 1
Jan. 27	Local features	
Jan. 30	Learning a classifier	Lab 2
Feb. 3	Convolutional neural networks	
Feb. 6	More convolutional neural networks	
Feb. 10	Robust model fitting and RANSAC	Lab 3
Feb. 13	Image registration	
Feb. 17	Camera Geometry	Lab 4
Feb. 20	More camera geometry	
Feb. 24	**Generative neural networks**	
Feb. 27	Generative neural networks	
Mar. 2	Visual Localization & Feature Learning	
Mar. 9	No lecture	