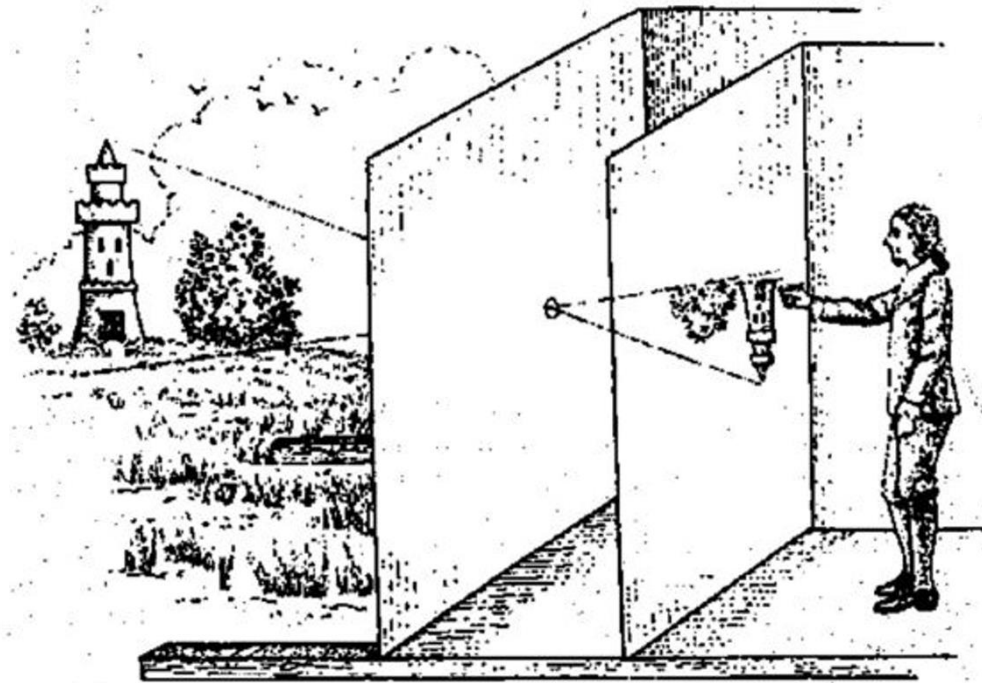


It's a 3D World, After All



3D Computer Vision

Elliott Wu

3D or Not 3D?

- Provocative blogpost by Vincent Sitzmann:
https://www.vincentsitzmann.com/blog/bitter_lesson_of_cv/
- Ongoing discussions:
<https://x.com/vincesitzmann/status/2023420051182022774?s=20>



VINCENT SITZMANN

Feb 1, 2026

The flavor of the bitter lesson for computer vision

I believe that computer vision as we know it is about to go away.

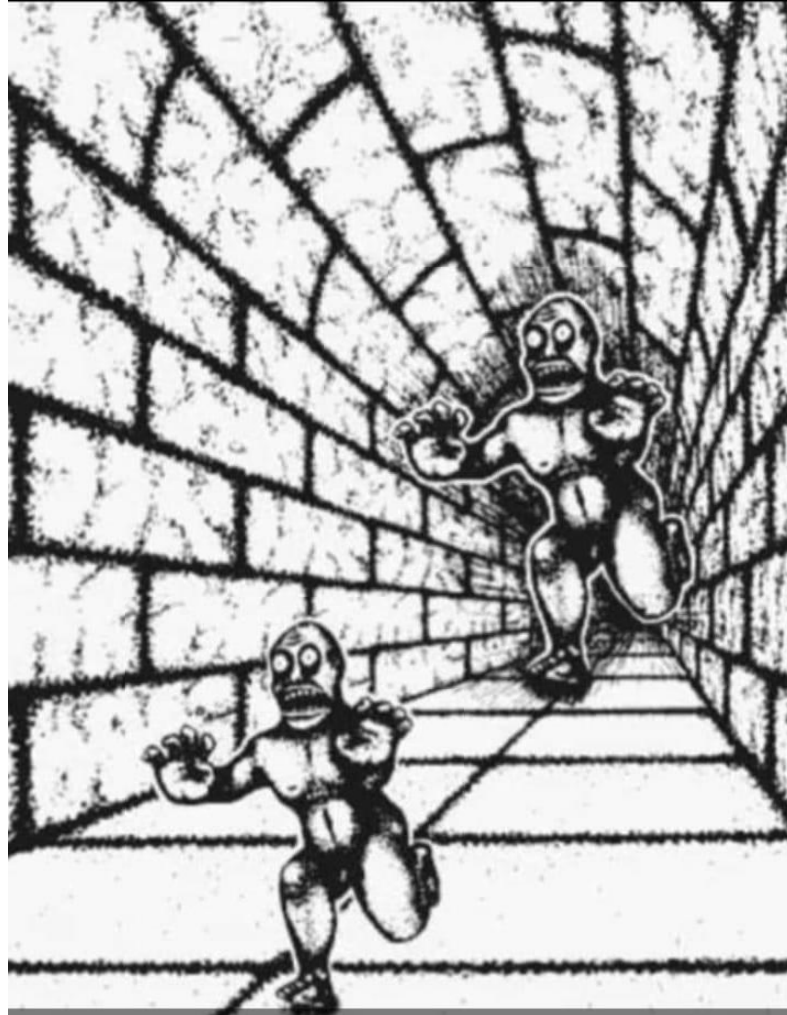
Historically, we have treated vision as a mapping from images to intermediate representations—classes, segmentation masks, or 3D reconstructions. But in the era of the Bitter Lesson, these distinct tasks are becoming qualitatively no different than edge detection: historical artifacts of scoping “solvable intermediate problems” rather than solving intelligence.

While the “LLM moment” in NLP clarified that language modeling is the ultimate objective, the vision community is still debating the flavor of its own revolution. We continue to fine-tune models for specific tasks like point tracking, segmentation, or 3D reconstruction—even as world models emerge, skirt all conventional intermediate representations, and directly solve a problem dramatically more general than everything our community has tackled in the past.

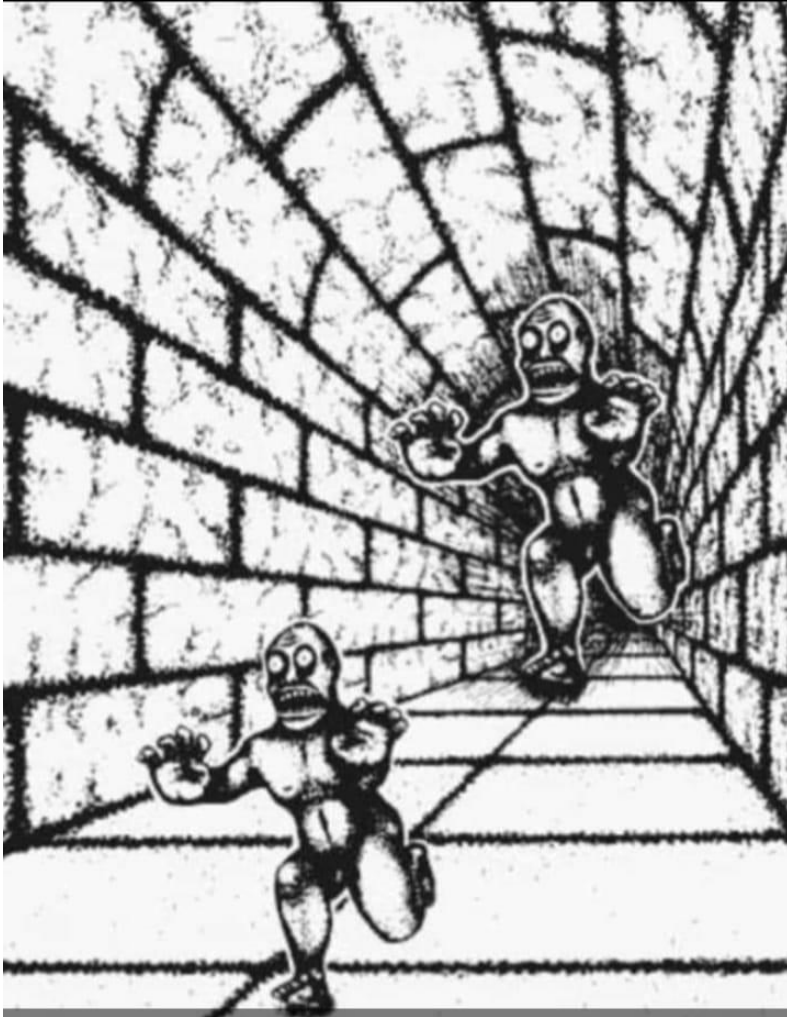
In this post, I argue that the future of computer vision is as part of end-to-end perception-action loops. The historical boundaries between computer vision, robot learning, and control will dissolve. Frontier research will no longer draw a boundary between “seeing” and “learning to act.”

As a special case, I will discuss the waning importance of 3D representations: I predict that just as we no longer hand-craft features for detection, we will soon stop using 3D as part of embodied intelligence.

We're good at seeing 3D!



We're good at seeing 3D!

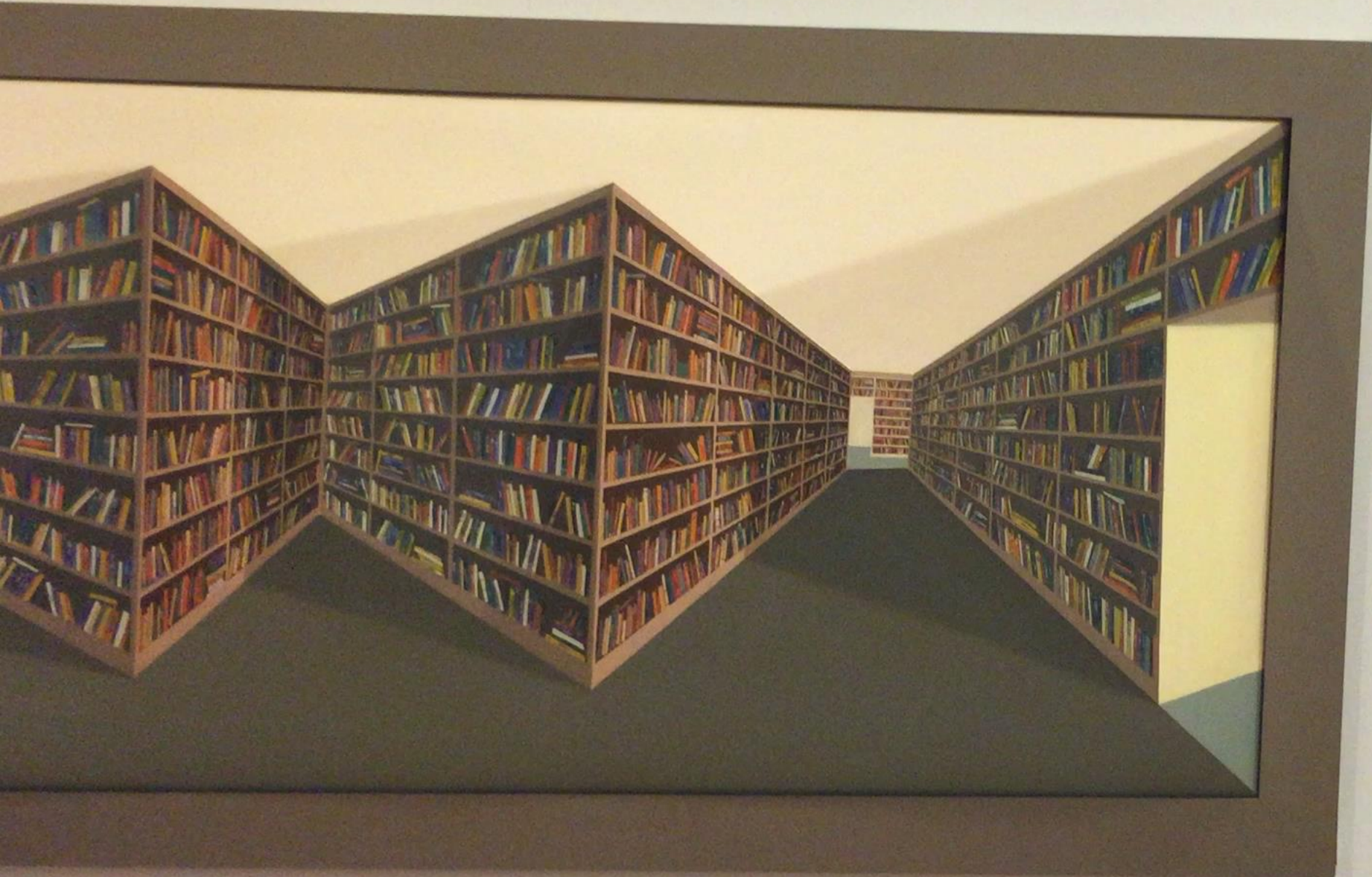


And sometimes too good..



And sometimes too good..





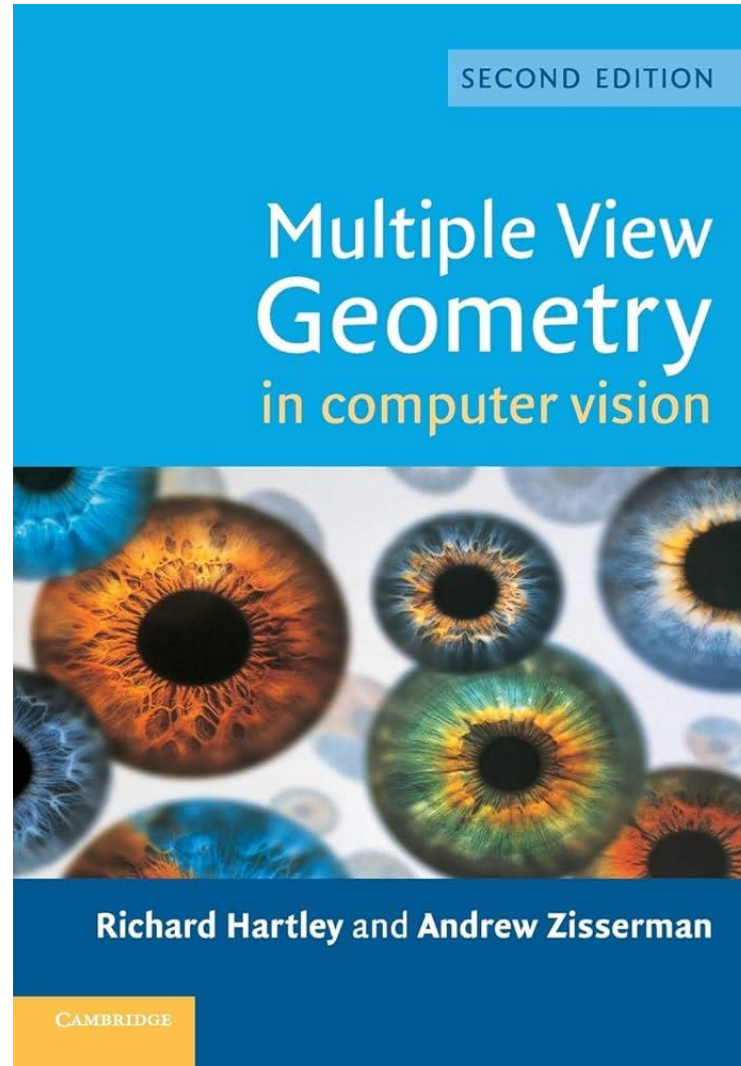


These 'Reverspective' paintings are mind-boggling. <https://youtu.be/fzZy1FtYsVM>

3D Computer Vision – Outline

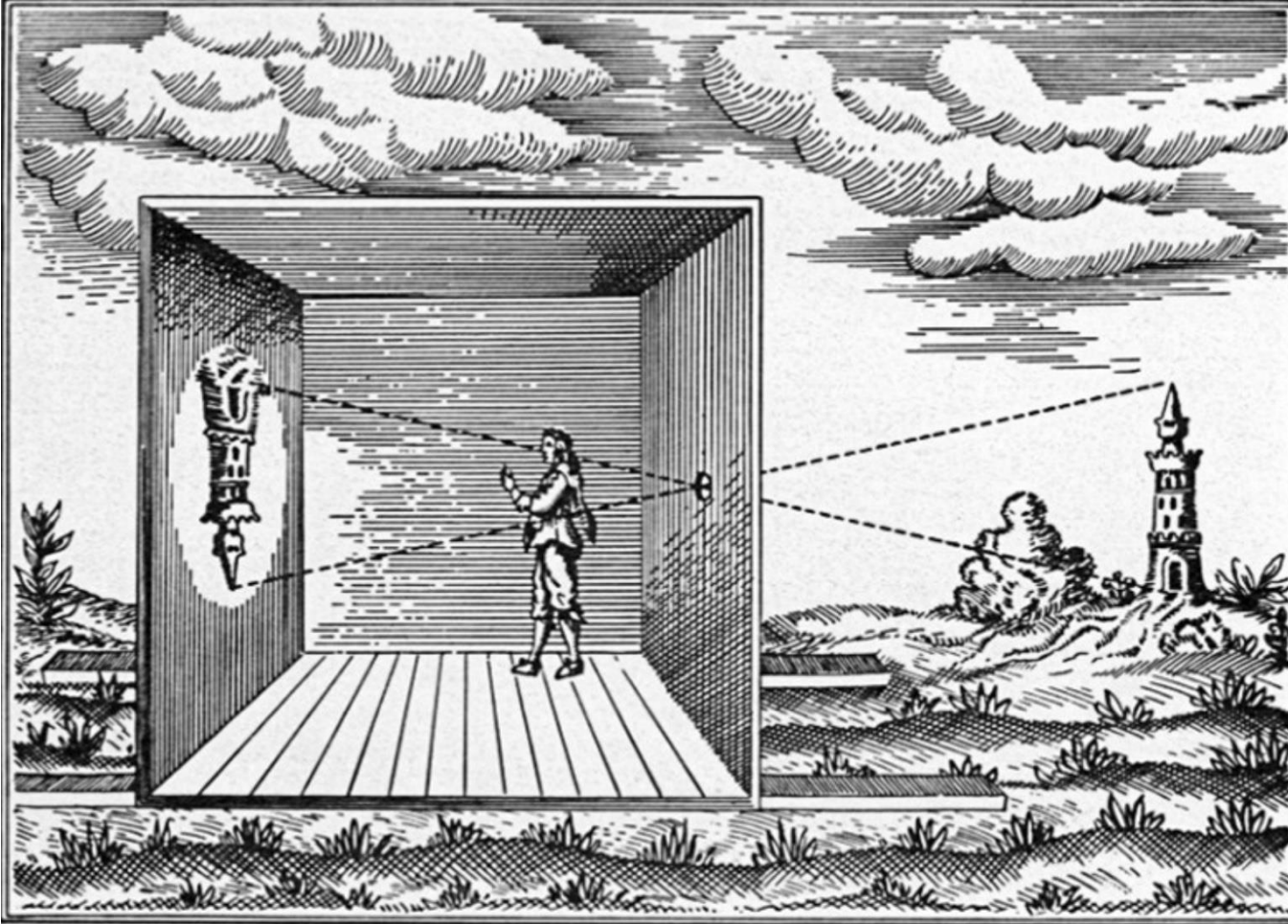
- Part 1 – Camera Model & Projection
- Part 2 – Multi-view Geometry
- Part 3 – 3D Representations & Rendering
- Part 4 – Learning-based 3D Modeling

Multi-view Geometry “Bible”



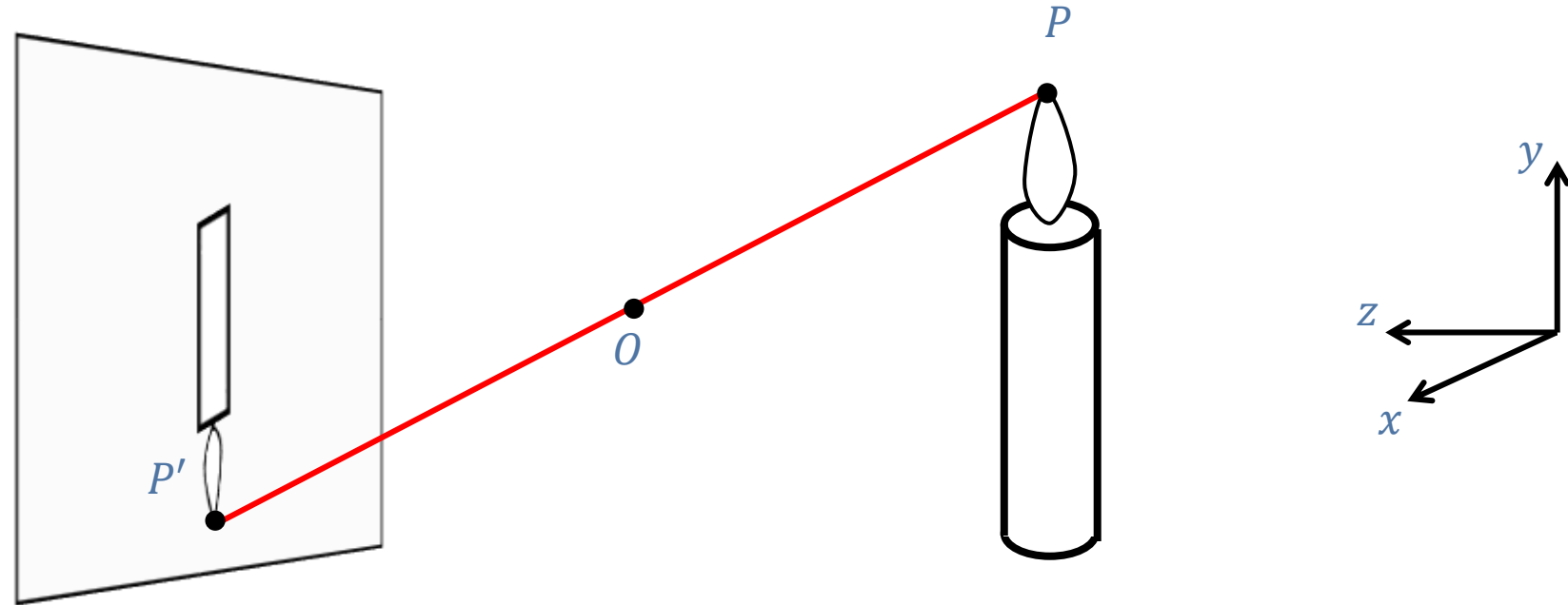
Part 1 – Camera Model & Projection

Image Formation – Camera Obscura



- Basic principle known to Mozi (470-390 BCE), Aristotle (384-322 BCE)
- Drawing aid for artists such as Leonardo da Vinci (1452-1519)

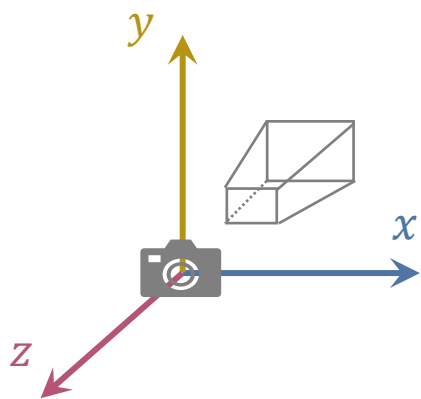
Pinhole Camera



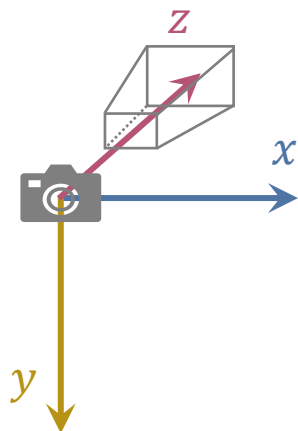
Canonical coordinate system

- The optical center (O) is at the origin
- The z axis is the optical axis perpendicular to the image plane
- The xy plane is parallel to the image plane, x and y axes are horizontal and vertical directions of the image plane

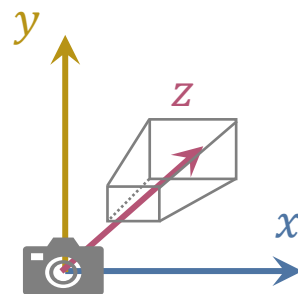
Everybody Agrees... Except on the Axes



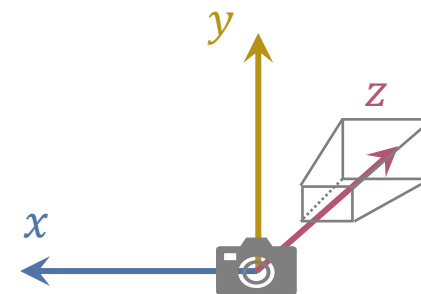
- OpenGL
- Blender
- ARKit
- Three.js
- Nerfstudio
- ...



- OpenCV
- Open3D
- COLMAP
- gsplat
- ...

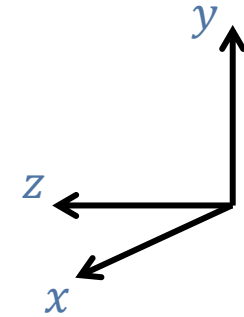
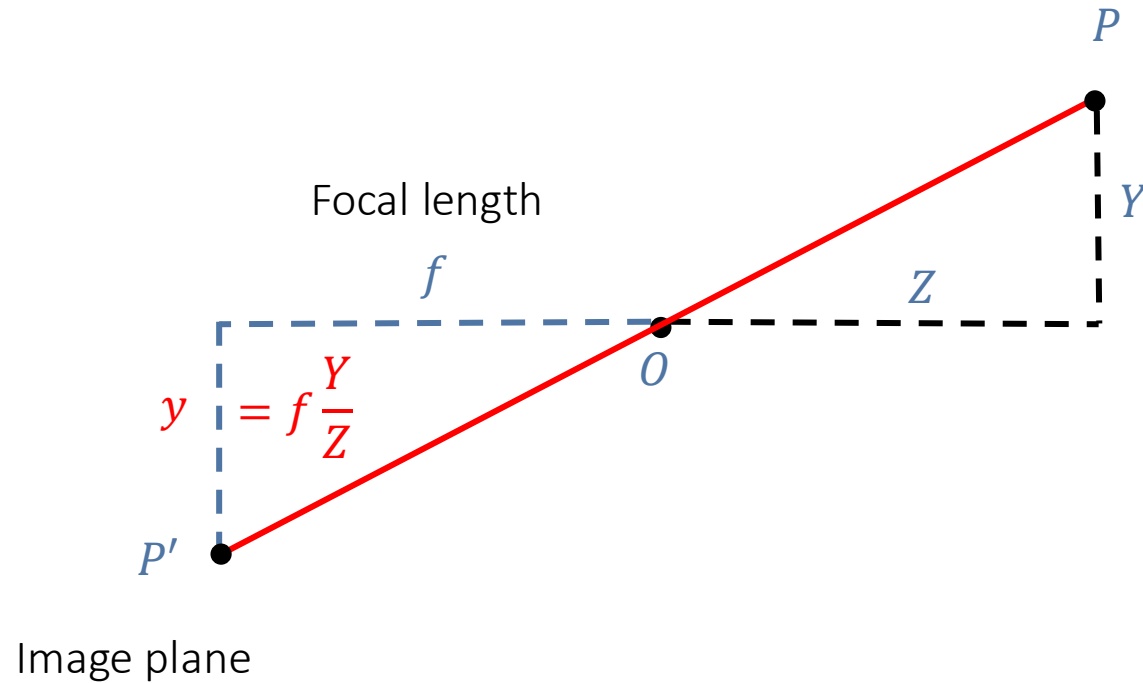


- Unity
- ...



- PyTorch3D
- ?

Perspective Projection



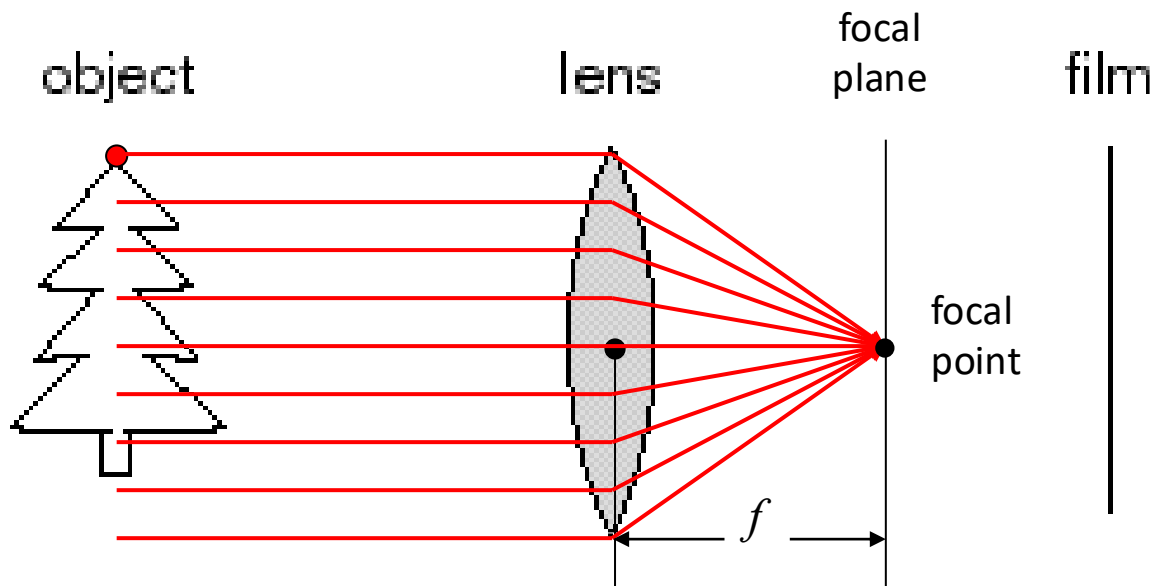
$$(X, Y, Z) \rightarrow \left(f \frac{X}{Z}, f \frac{Y}{Z} \right)$$

3D point

2D point

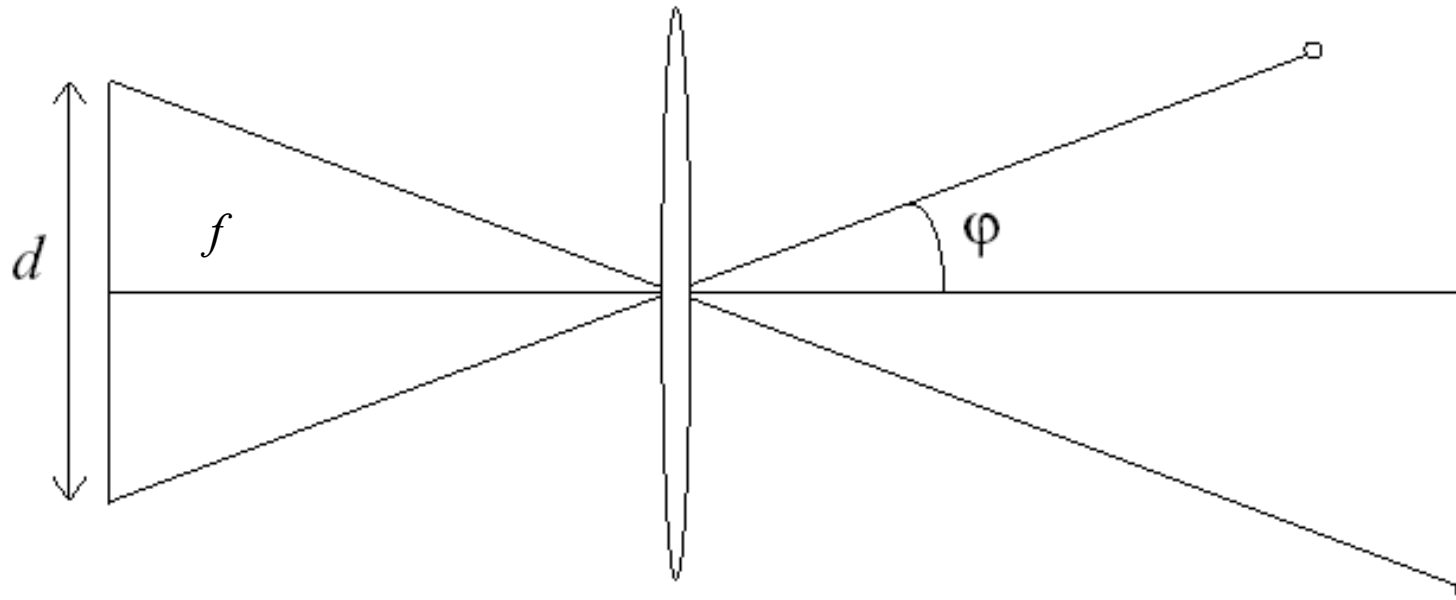
Focal Length of Lenses

- In practice, most cameras use lenses, and the focal length is determined by the physical properties of the lens, such as refraction index, thickness, curvature etc.



Field of View (FOV)

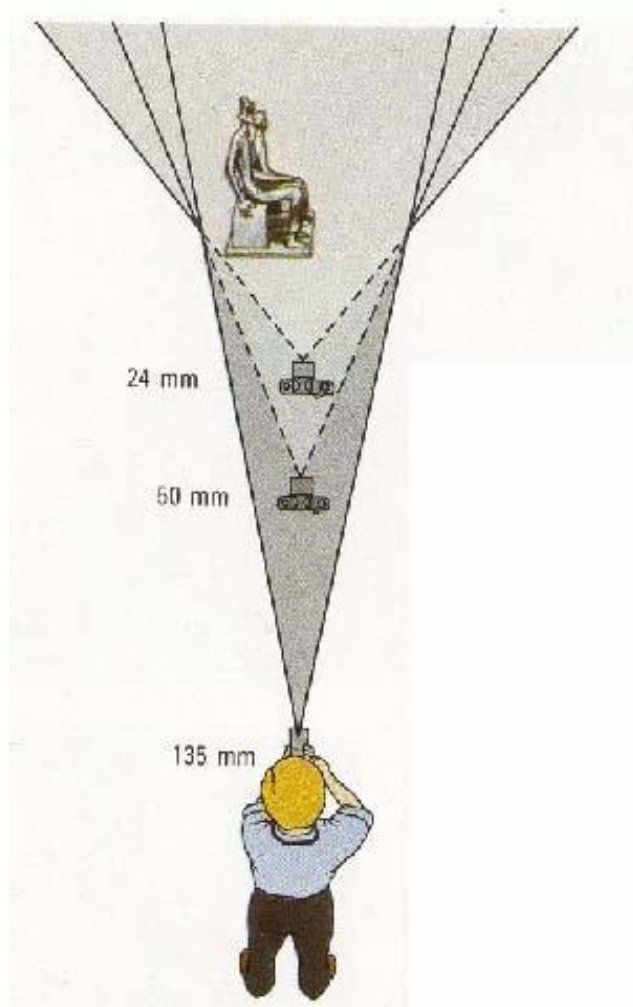
- The field of view is the angular extent of the world observed by the camera.
- Focal length (f), length of the sensor (d):



$$\varphi = \tan^{-1} \frac{d}{2f}$$

- Larger focal length = smaller FOV (assuming a constant sensor size)

Field of View / Focal Length Ambiguity



Large FOV, small f
Camera close to car



Small FOV, large f
Camera far from the car

Field of View / Focal Length Ambiguity



wide-angle



standard



telephoto

- What would happen when reconstructing 3D shapes from a single image with *unknown* focal length/FOV?

Field of View / Focal Length Ambiguity



wide-angle



standard



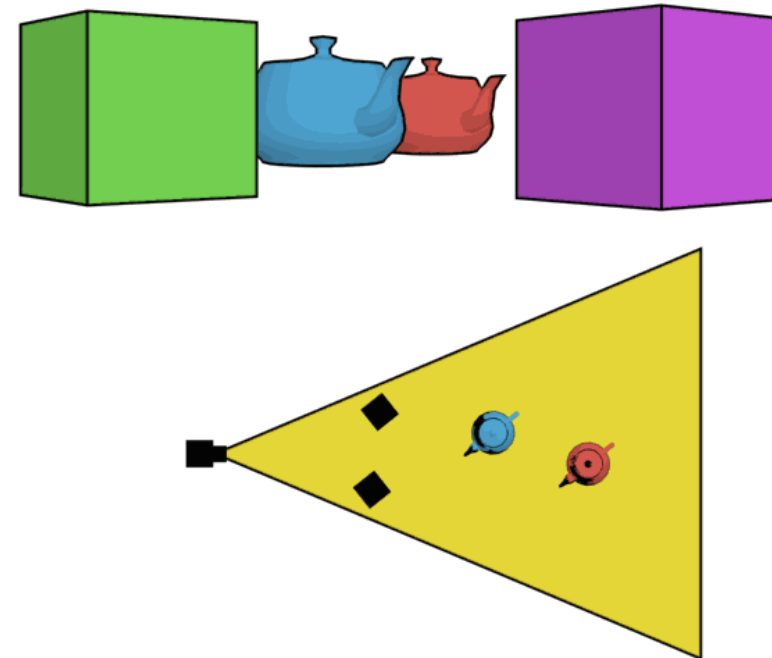
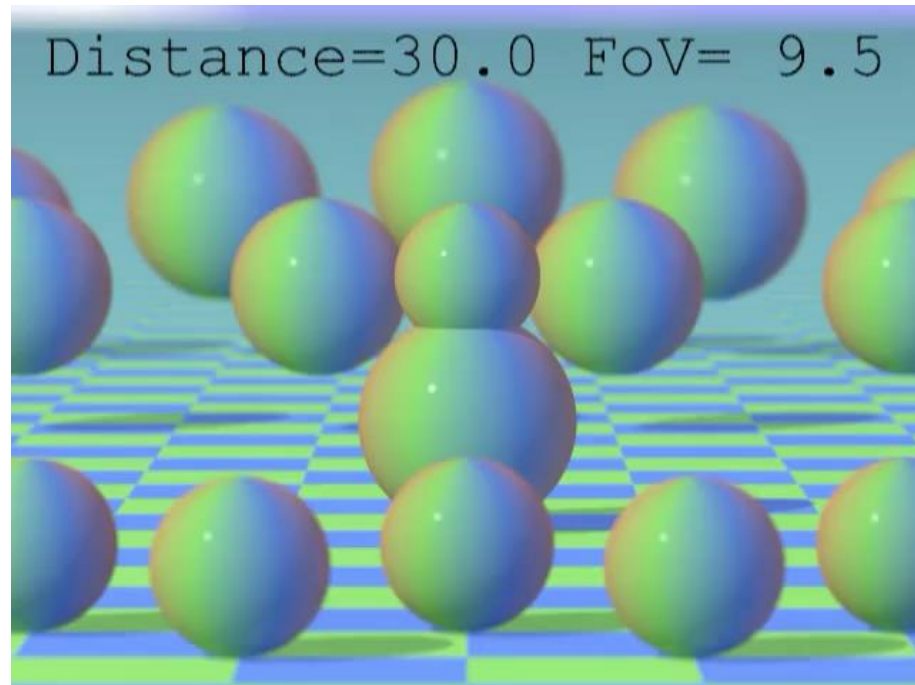
telephoto

(reconstructed by Hunyuan3D V3.1)

- What would happen when reconstructing 3D shapes from a single image with *unknown* focal length/FOV?

Dolly Zoom Effect

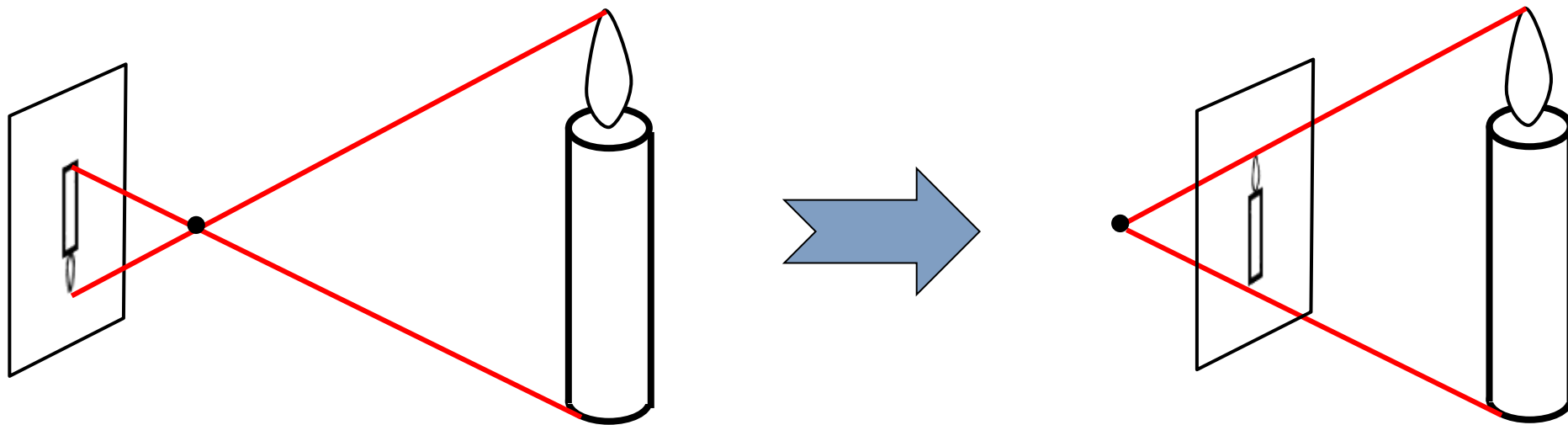
- Continuously adjusting the focal length, while dollying toward or away from the subject.



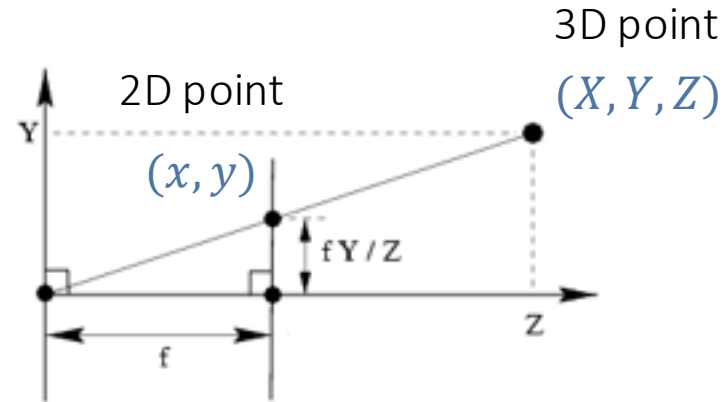
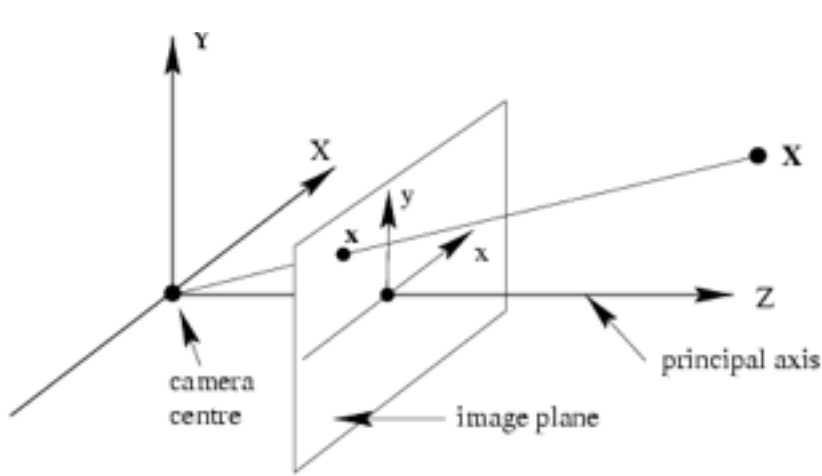
https://en.wikipedia.org/wiki/Dolly_zoom

Perspective Projection

- Instead of dealing with an image that is upside down, most of the time we will pretend that the image plane is in front of the camera center.



Perspective Projection



$$(X, Y, Z) \rightarrow (x, y)$$

$$x = f \frac{X}{Z}, y = f \frac{Y}{Z}$$

- Perspective projection is not a linear transformation!
- But most other transformations we will be dealing with are linear.
- Projective geometry provides a handy mathematical tool to unify them.

Homogeneous Coordinates

- To form homogeneous coordinates from Euclidean coordinates, append 1 as the last entry:

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

homogeneous coordinates
of a 2D point (x, y)

$$(X, Y, Z) \Rightarrow \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

homogeneous coordinates
of a 3D point (X, Y, Z)

- To convert homogeneous coordinates to Euclidean coordinates, divide by the last entry:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} \Rightarrow (X/W, Y/W, Z/W)$$

- All scalar multiples represent the same point! $\begin{bmatrix} x \\ y \\ w \end{bmatrix} \sim \lambda \begin{bmatrix} x \\ y \\ w \end{bmatrix}, \quad \lambda \neq 0$

Homogeneous Coordinates

2D	3D
2D point (x, y) $\mathbf{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$	3D point (x, y, z) $\mathbf{X} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$
2D line \mathbf{l} $\mathbf{l} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$ $\mathbf{l}^T \mathbf{x} = ax + by + c = 0$	3D plane $\boldsymbol{\pi}$ $\boldsymbol{\pi} = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$ $\boldsymbol{\pi}^T \mathbf{X} = aX + bY + cZ + d = 0$
2 points $\mathbf{x}_1, \mathbf{x}_2$ form a line \mathbf{l} $\mathbf{l} = \mathbf{x}_1 \times \mathbf{x}_2$	3 points $\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3$ form a plane $\boldsymbol{\pi}$ $\boldsymbol{\pi} = \mathbf{x}_1 \times \mathbf{x}_2 \times \mathbf{x}_3$
2 lines $\mathbf{l}_1, \mathbf{l}_2$ intersect at point \mathbf{x} $\mathbf{x} = \mathbf{l}_1 \times \mathbf{l}_2$	3 planes $\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \boldsymbol{\pi}_3$ intersect at point \mathbf{X} $\mathbf{X} = \boldsymbol{\pi}_1 \times \boldsymbol{\pi}_2 \times \boldsymbol{\pi}_3$

Perspective Projection Matrix

- Projection is a matrix multiplication using homogeneous coordinates:

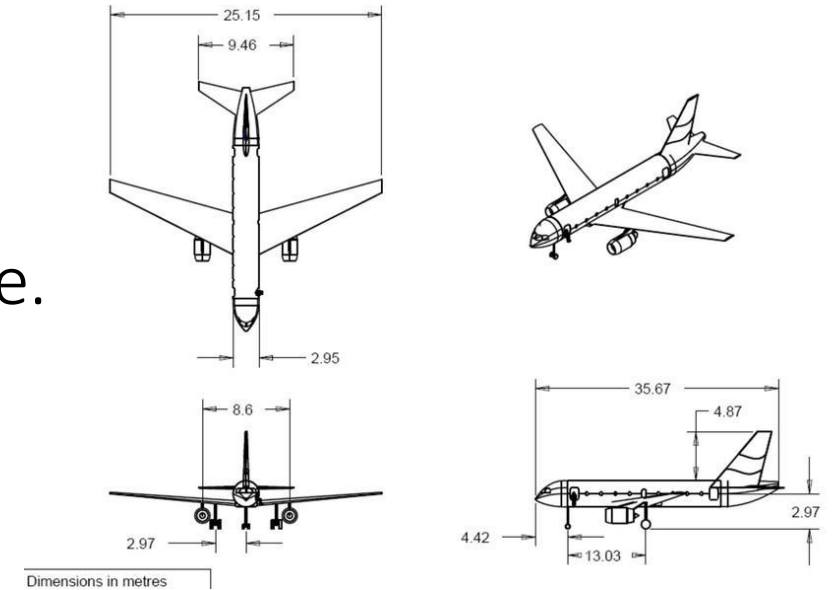
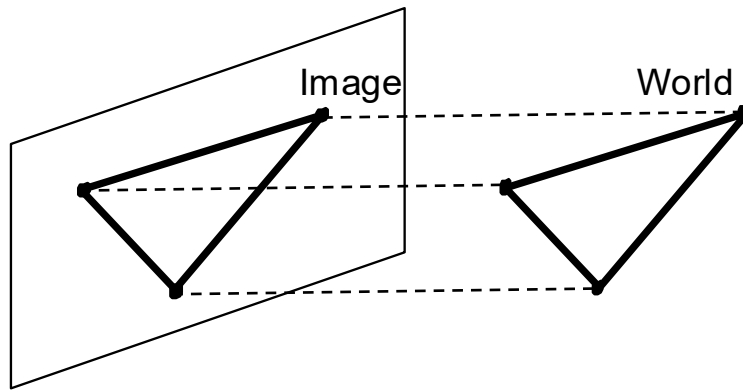
$$\begin{bmatrix} \\ \\ \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \\ \\ \end{bmatrix} \Rightarrow \left(f \frac{X}{Z}, f \frac{Y}{Z} \right)$$

divide by the third
coordinate

- What will happen if f is very large?

Orthographic Projection

- A form of *parallel projection* where the projection axis is orthogonal to image plane.



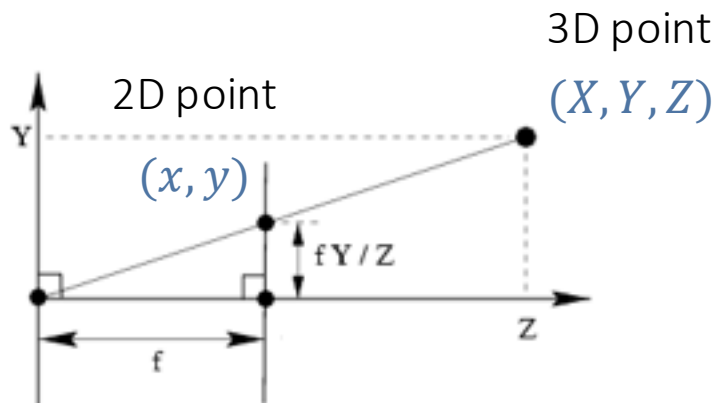
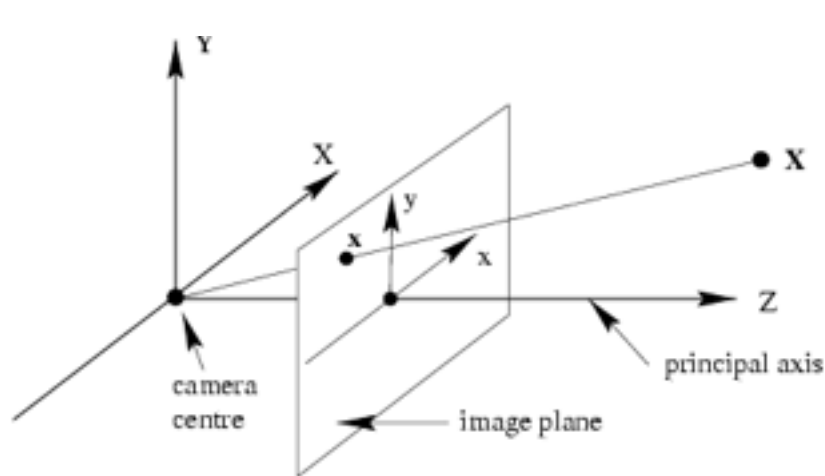
Three-view drawing

- Assuming projection along the z axis, what's the matrix?

$$\begin{bmatrix} \\ \\ \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \\ \\ \end{bmatrix}$$

Perspective Projection *in Normalized Coordinates*

(assuming camera looking toward z axis and no scaling)



$$(X, Y, Z) \rightarrow (x, y)$$

$$x = f \frac{X}{Z}, y = f \frac{Y}{Z}$$

$$\underbrace{\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}}_{\mathbf{x}} \cong \underbrace{\begin{pmatrix} fX \\ fY \\ Z \end{pmatrix}}_{\mathbf{P}} = \underbrace{\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\mathbf{P}} \underbrace{\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}}_{\mathbf{X}}$$

Homogeneous
coordinates of
image point

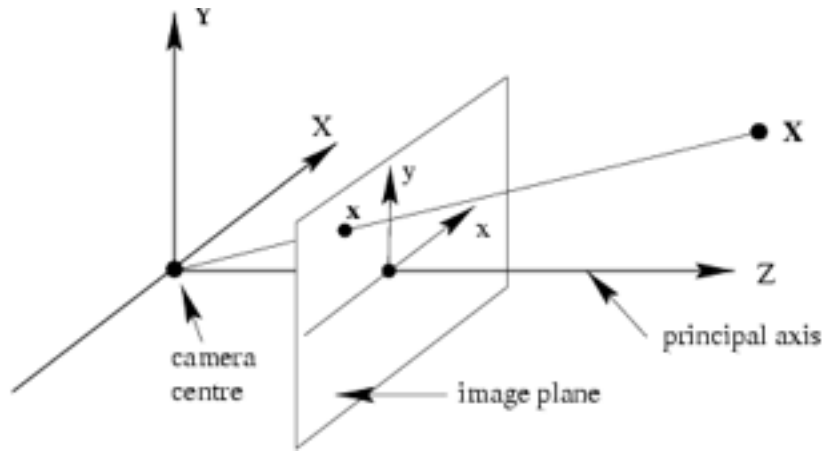
Camera
projection
matrix

Homogeneous
coordinates of
3D point

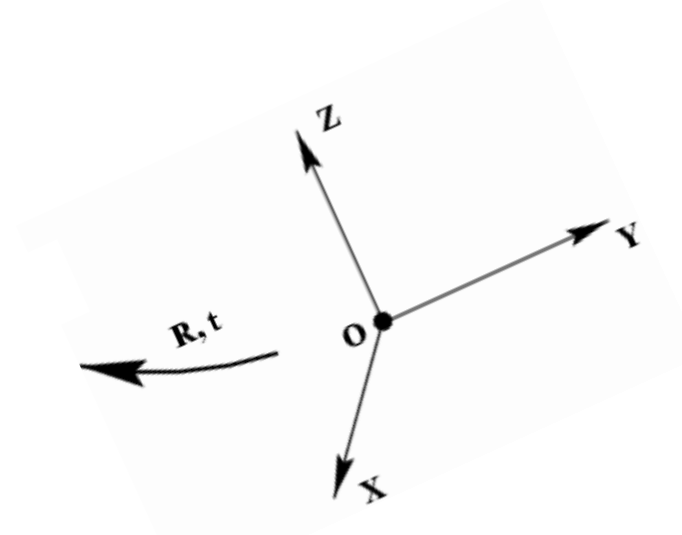
$$\mathbf{x} \cong \mathbf{P}\mathbf{X}$$

Equality up to scale

Camera Calibration



camera coordinate system



world coordinate system

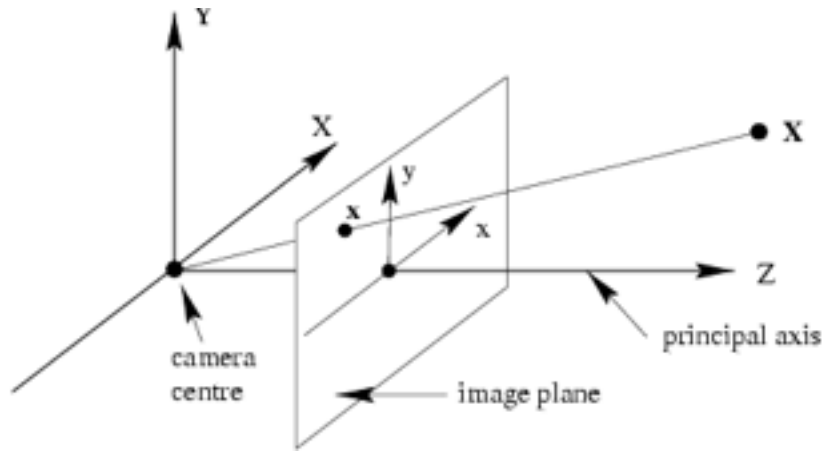
- **Camera calibration:** figuring out transformation from *world* coordinate system to *image* coordinate system

$$\begin{pmatrix} \text{2D point} \\ \mathbf{x} \\ (3 \times 1) \end{pmatrix} \cong \begin{pmatrix} \text{Camera to pixel coord. trans. matrix} \\ \mathbf{K} (3 \times 3) \end{pmatrix} \begin{pmatrix} \text{Canonical projection matrix} \\ [\mathbf{I} \mid \mathbf{0}] (3 \times 4) \end{pmatrix} \begin{pmatrix} \text{World to camera coord. trans. matrix} \\ \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} (4 \times 4) \end{pmatrix} \begin{pmatrix} \text{3D point} \\ \mathbf{X} \\ (4 \times 1) \end{pmatrix}$$

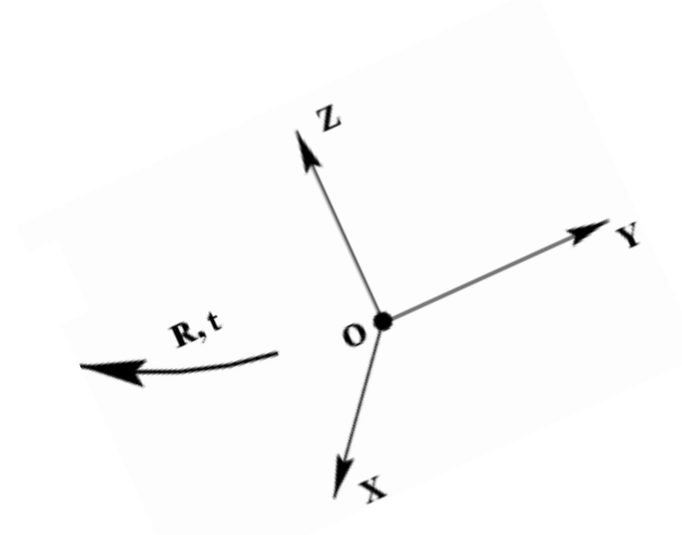
Intrinsic camera parameters: principal point, scaling factors

Extrinsic camera parameters: rotation, translation

Camera Calibration



camera coordinate system



world coordinate system

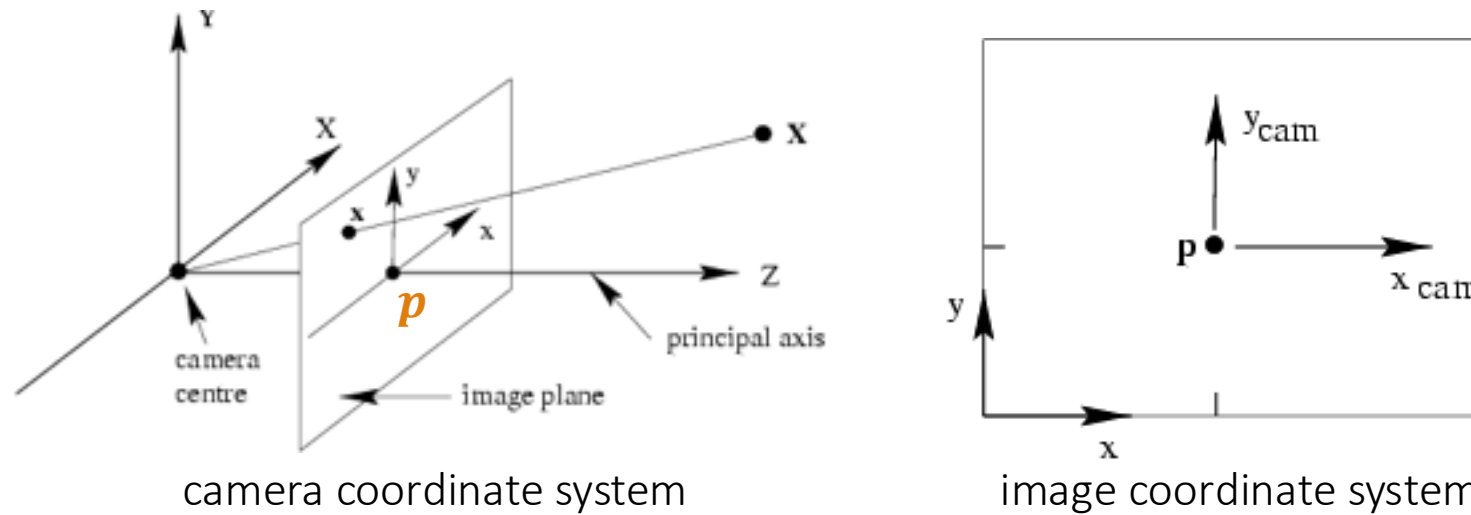
- **Camera calibration:** figuring out transformation from *world* coordinate system to *image* coordinate system

$$\begin{pmatrix} \text{2D point} \\ \mathbf{x} \\ (3 \times 1) \end{pmatrix} \cong \underbrace{\begin{pmatrix} \text{Camera to pixel coord. trans. matrix} \\ \mathbf{K} \ (3 \times 3) \end{pmatrix} \begin{pmatrix} \text{Canonical projection matrix} \\ [\mathbf{I} \mid \mathbf{0}] \ (3 \times 4) \end{pmatrix} \begin{pmatrix} \text{World to camera coord. trans. matrix} \\ \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \ (4 \times 4) \end{pmatrix}}_{\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]} \begin{pmatrix} \text{3D point} \\ \mathbf{X} \\ (4 \times 1) \end{pmatrix}$$

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

General camera projection matrix

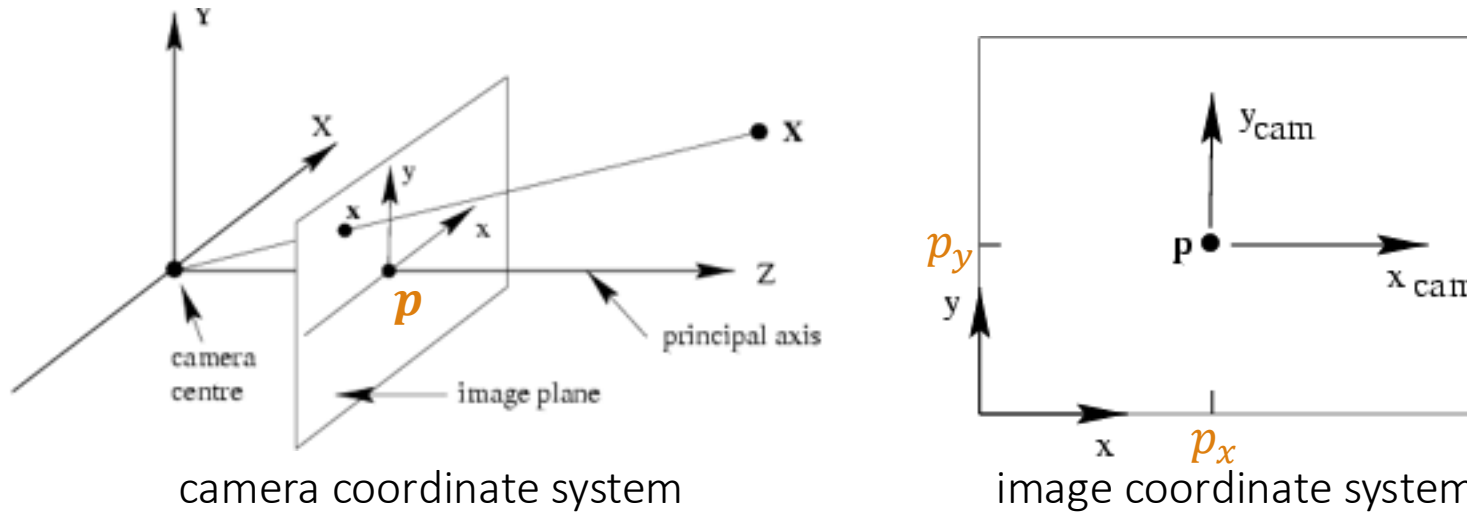
Intrinsic Parameters – Principal Point



Principal point (p): point where principal axis intersects the image plane

- In the *normalized* coordinate system, the **origin** is at the **principal point**.
- In the *image* coordinate system, the **origin** is in the **corner**.

Intrinsic Parameters – Principal Point

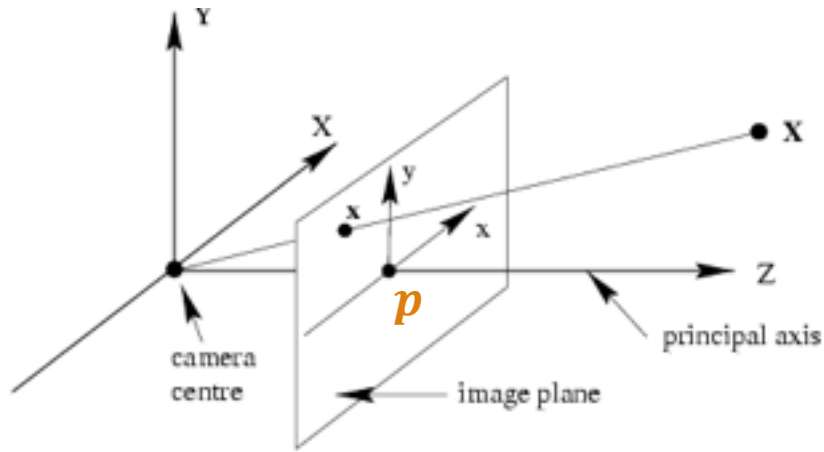


$$x = f \frac{X}{Z} + p_x, \quad y = f \frac{Y}{Z} + p_y$$

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \cong \begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

calibration matrix **K**

Intrinsic Parameters – Scaling



camera coordinate system

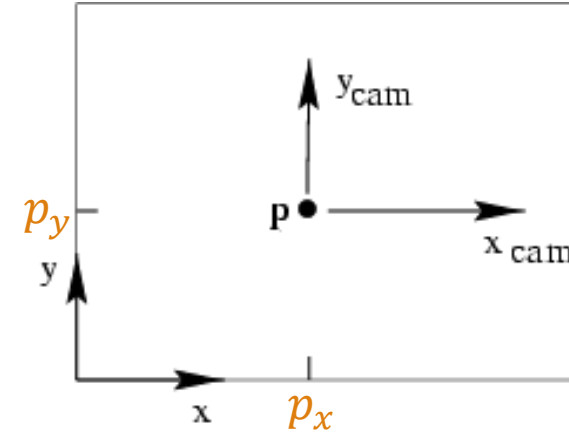


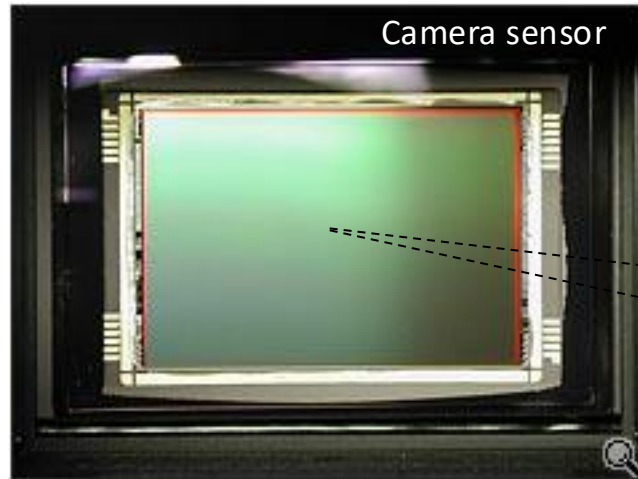
image coordinate system

$$x = f \frac{X}{Z} + p_x, \quad y = f \frac{Y}{Z} + p_y$$

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \xleftarrow[\text{pixels/m}]{\alpha} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \cong \begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

pixels m calibration matrix \mathbf{K}

Intrinsic Parameters – Scaling



m_x pixels/m in horizontal direction
 m_y pixels/m in vertical direction

Pixel size (m): $\frac{1}{m_x} \times \frac{1}{m_y}$

Scaling factors

$$\begin{bmatrix} m_x & 0 & 0 \\ 0 & m_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

pixels/m

Calibration matrix
 \mathbf{K} in metric units

$$\begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}$$

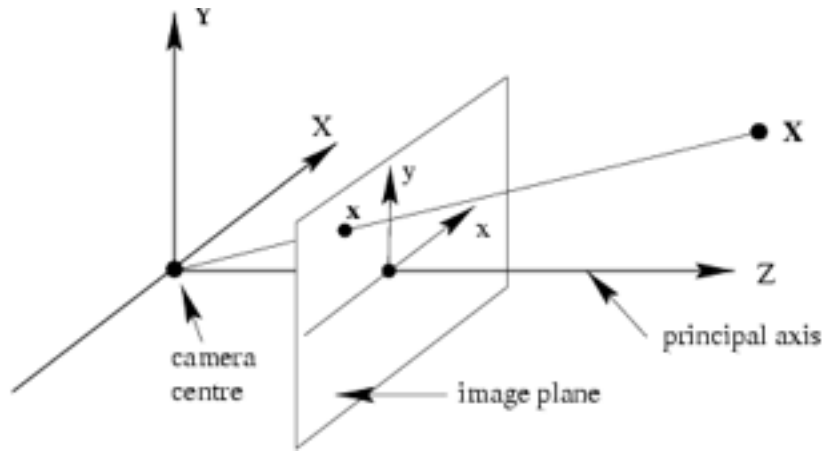
m

Calibration matrix
 \mathbf{K} in pixel units

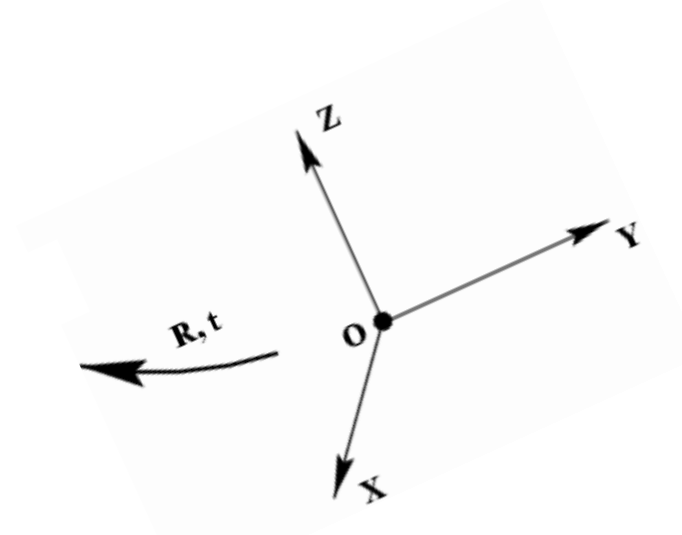
$$= \begin{bmatrix} \alpha_x & 0 & \beta_x \\ 0 & \alpha_y & \beta_y \\ 0 & 0 & 1 \end{bmatrix}$$

pixels

Camera Calibration



camera coordinate system



world coordinate system

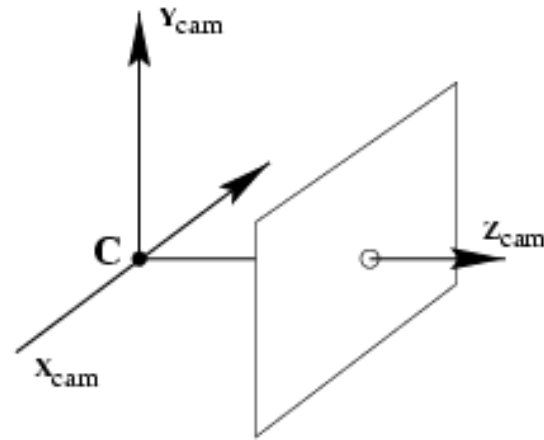
- **Camera calibration:** figuring out transformation from *world* coordinate system to *image* coordinate system

$$\begin{pmatrix} \text{2D point} \\ \mathbf{x} \\ (3 \times 1) \end{pmatrix} \cong \underbrace{\begin{pmatrix} \text{Camera to pixel coord. trans. matrix} \\ \mathbf{K} \ (3 \times 3) \end{pmatrix} \begin{pmatrix} \text{Canonical projection matrix} \\ [\mathbf{I} \mid \mathbf{0}] \ (3 \times 4) \end{pmatrix} \begin{pmatrix} \text{World to camera coord. trans. matrix} \\ \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \ (4 \times 4) \end{pmatrix}}_{\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]} \begin{pmatrix} \text{3D point} \\ \mathbf{X} \\ (4 \times 1) \end{pmatrix}$$

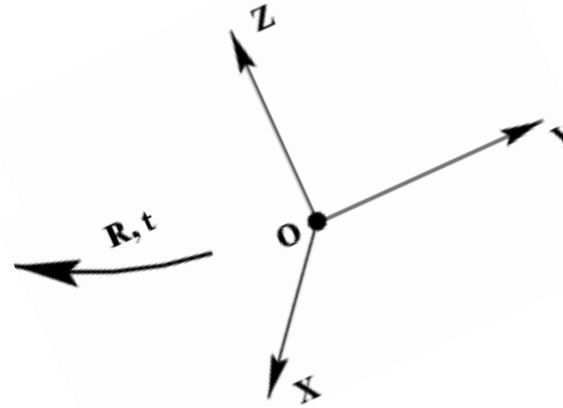
$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$$

General camera projection matrix

Extrinsic Parameters



camera coordinate system



world coordinate system

- In *non-homogeneous* coordinates, the transformation from world to normalized camera coordinate system is given by:

$$\tilde{\mathbf{X}}_{\text{cam}} = \mathbf{R}(\tilde{\mathbf{X}} - \tilde{\mathbf{C}}) = \mathbf{R}\tilde{\mathbf{X}} + \mathbf{t}$$

coords. of point in normalized camera frame

3x3 rotation matrix

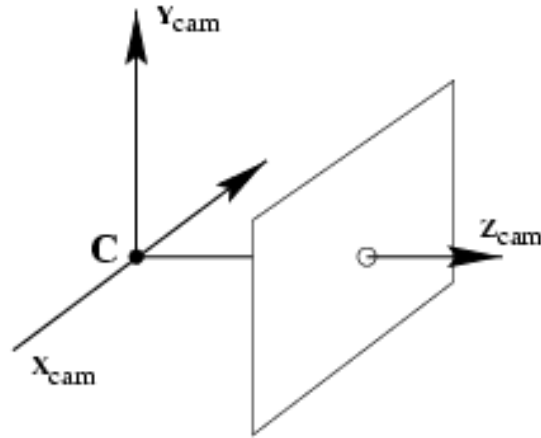
coords. of a point in world frame

coords. of camera center in world frame

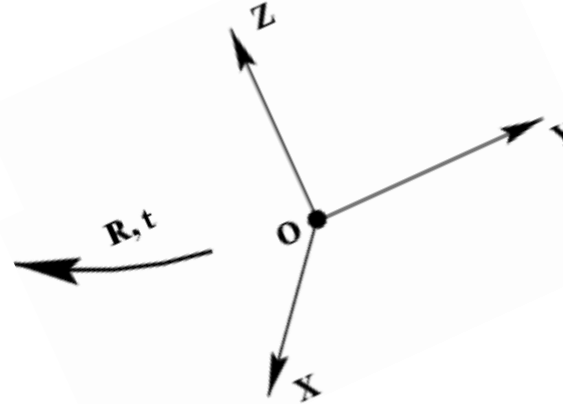
$$\text{W2C: } \tilde{\mathbf{X}}_{\text{cam}} = \mathbf{R}\tilde{\mathbf{X}}_{\text{world}} + \mathbf{t}$$

$$\text{C2W: } \tilde{\mathbf{X}}_{\text{world}} = \mathbf{R}^T \tilde{\mathbf{X}}_{\text{cam}} - \mathbf{R}^T \mathbf{t}$$

Extrinsic Parameters



camera coordinate system



world coordinate system

In non-homogeneous
coordinates:

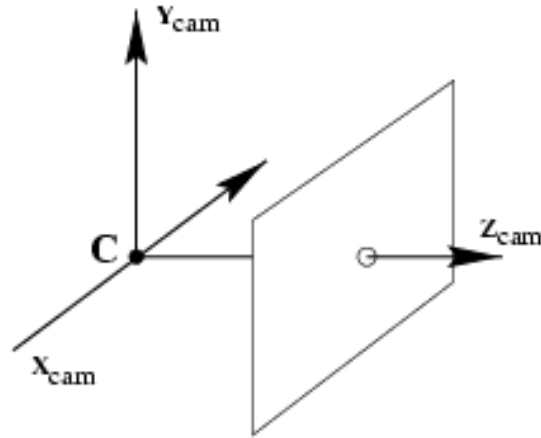
$$\tilde{X}_{cam} = R\tilde{X} + t$$

In homogeneous
coordinates:

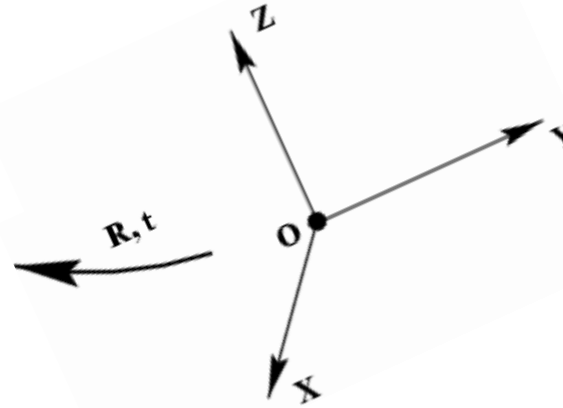
$$\begin{pmatrix} \tilde{X}_{cam} \\ 1 \end{pmatrix} = \begin{bmatrix} R & t \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{pmatrix} \tilde{X} \\ 1 \end{pmatrix}$$

3D transformation
matrix (4 x 4)

Extrinsic Parameters



camera coordinate system



world coordinate system

In non-homogeneous
coordinates:

$$\tilde{X}_{cam} = R\tilde{X} + t$$

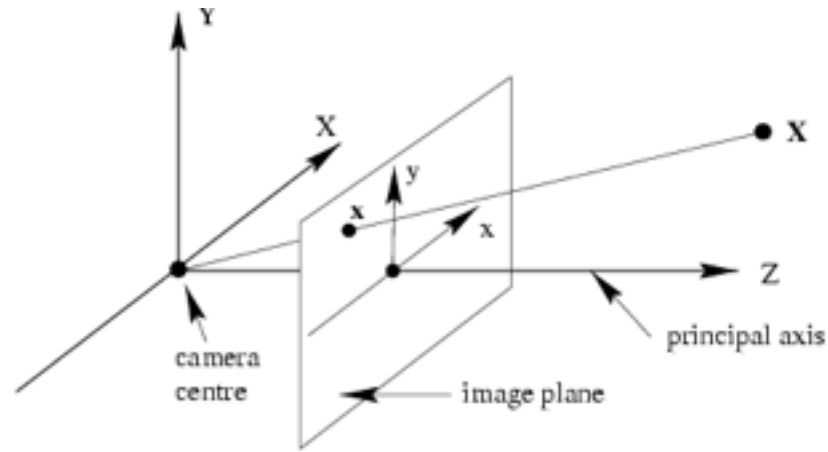
In homogeneous
coordinates:

$$X_{cam} = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} X$$

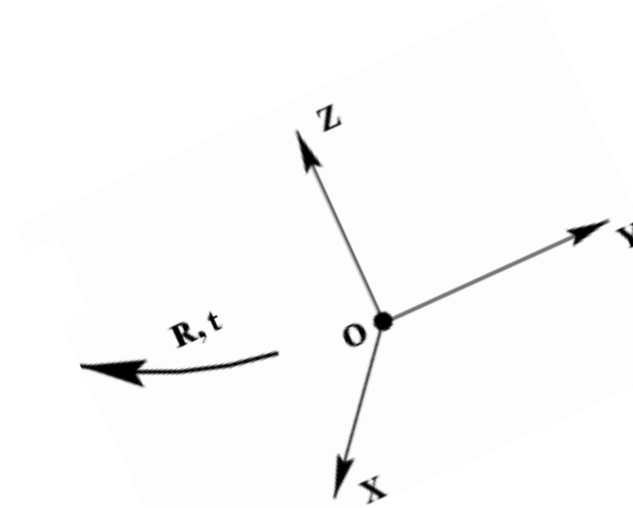
3D transformation
matrix (4 x 4)

And then to image coordinates: $x \cong K[I|0]X_{cam}$

Camera Calibration



camera coordinate system



world coordinate system

$$x \cong K[R|t]X$$

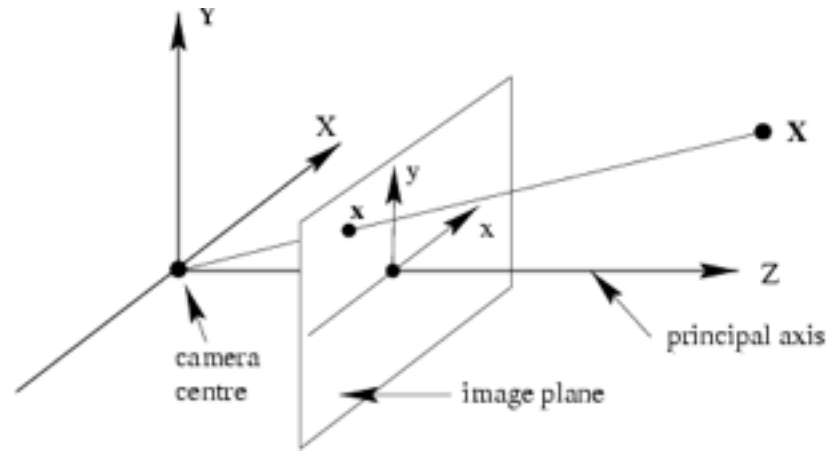
- **Camera calibration:** figuring out transformation from *world* coordinate system to *image* coordinate system

$$\begin{pmatrix} \text{2D point} \\ \mathbf{x} \\ (3 \times 1) \end{pmatrix} \cong \underbrace{\begin{pmatrix} \text{Camera to pixel coord. trans. matrix} \\ \mathbf{K} (3 \times 3) \end{pmatrix} \begin{pmatrix} \text{Canonical projection matrix} \\ [\mathbf{I} | \mathbf{0}] (3 \times 4) \end{pmatrix} \begin{pmatrix} \text{World to camera coord. trans. matrix} \\ \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} (4 \times 4) \end{pmatrix}}_{\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]} \begin{pmatrix} \text{3D point} \\ \mathbf{X} \\ (4 \times 1) \end{pmatrix}$$

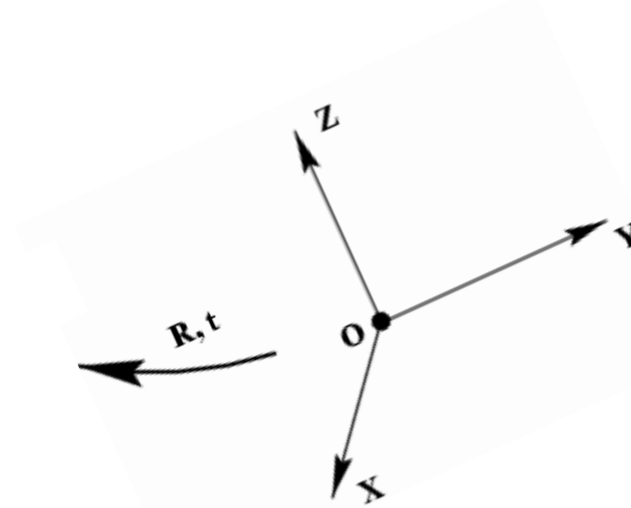
$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$$

General camera projection matrix

Camera Calibration



camera coordinate system



world coordinate system

$$x \cong K[R|t]X$$

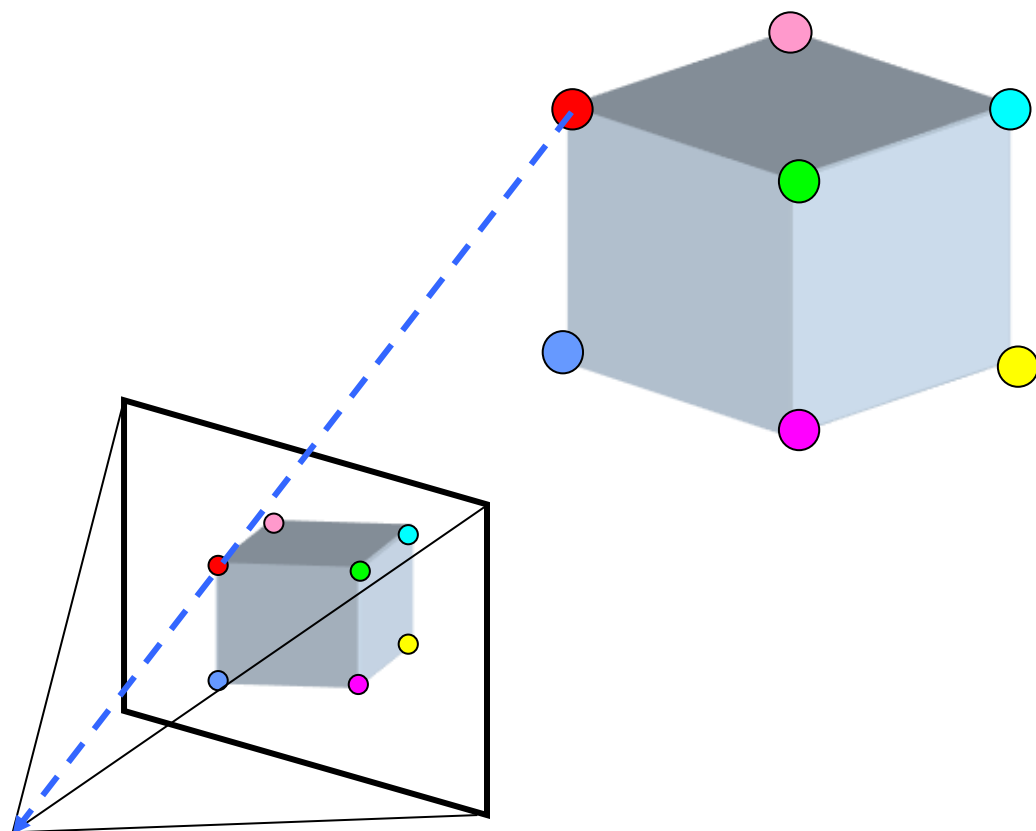
- **Camera calibration:** figuring out transformation from *world* coordinate system to *image* coordinate system

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \cong \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

← We could solve this as a linear system (DLT), then perform QR decomposition to get K , R , t .
(not robust in practice;
sensitive to noise)

Perspective-n-Point (PnP)

$$\mathbf{x} \cong \mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{X}$$



Camera
 \mathbf{R}, \mathbf{t} ?

Known:

- n 3D points \mathbf{X}
- Corresponding 2D pixel coordinates \mathbf{x}
- Camera intrinsics \mathbf{K}

Solve for:

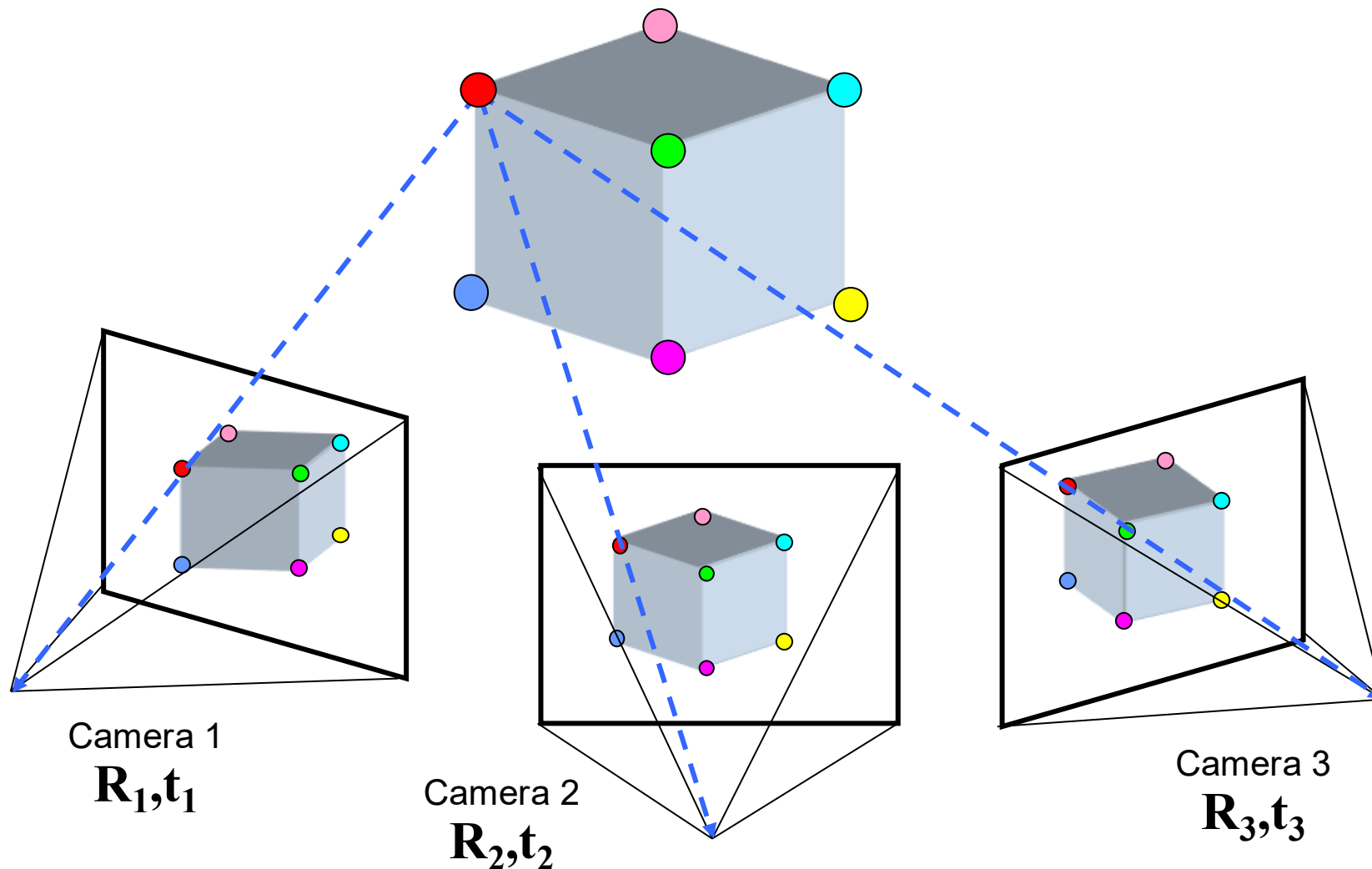
- Camera pose (extrinsics) \mathbf{R}, \mathbf{t}

Q: How many points at least do we need?

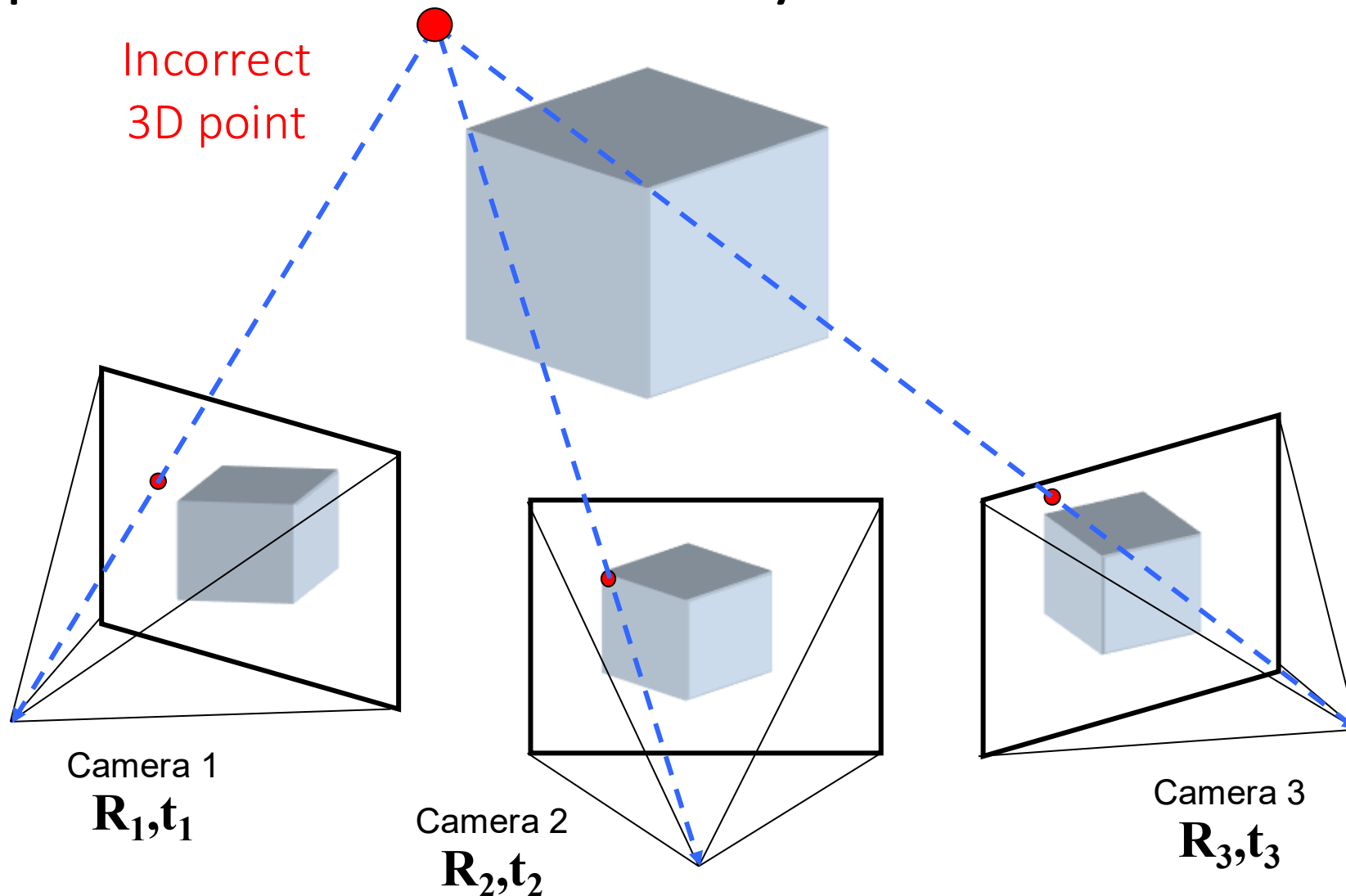
A: $n \geq 3$ (in general). We have 6 DoF unknowns \mathbf{R}, \mathbf{t} . Each point provides 2 constraints (2 equations for \mathbf{x} and \mathbf{y}).

Part 2 – Multi-view Geometry

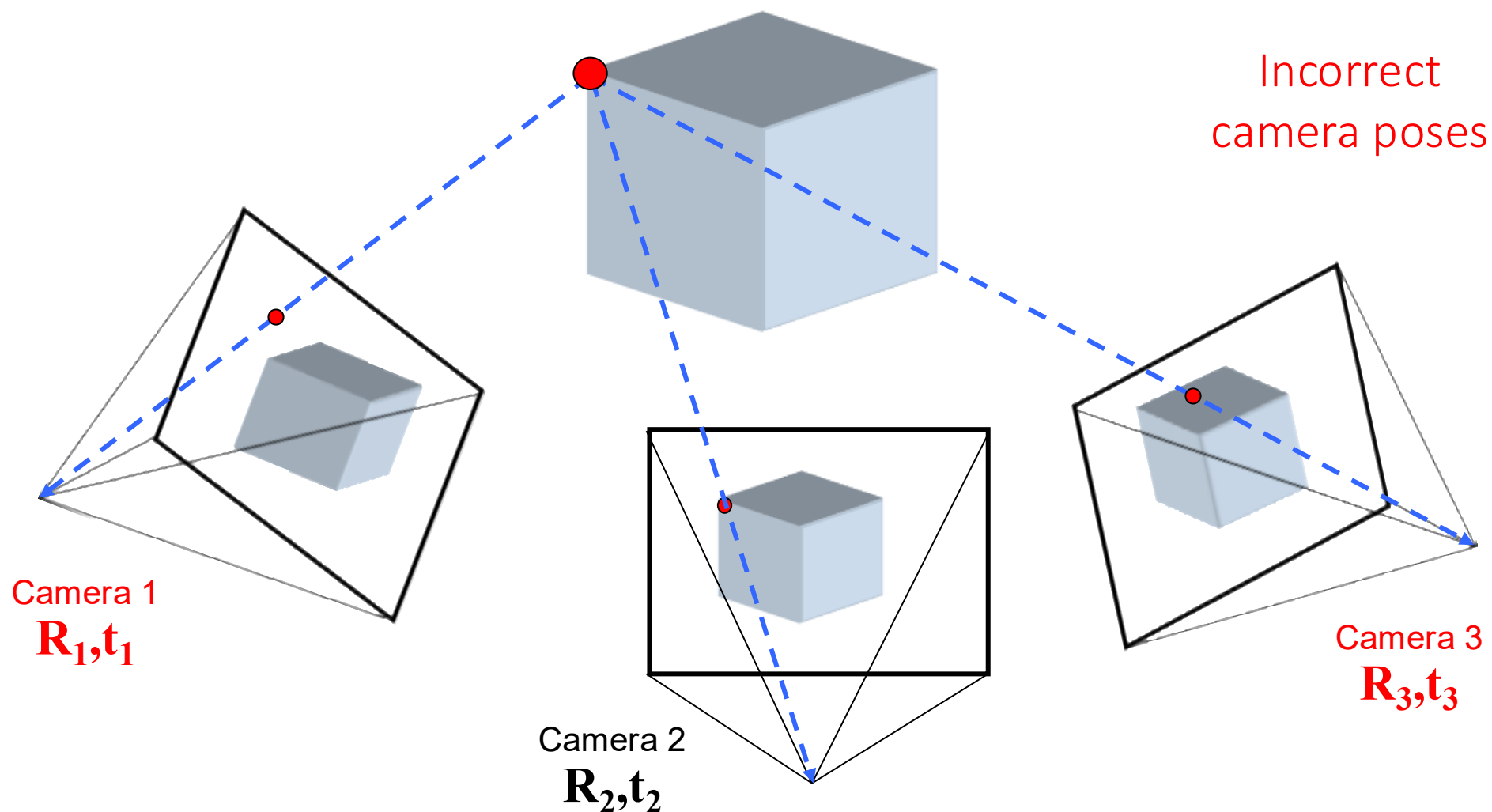
Multiple View Geometry



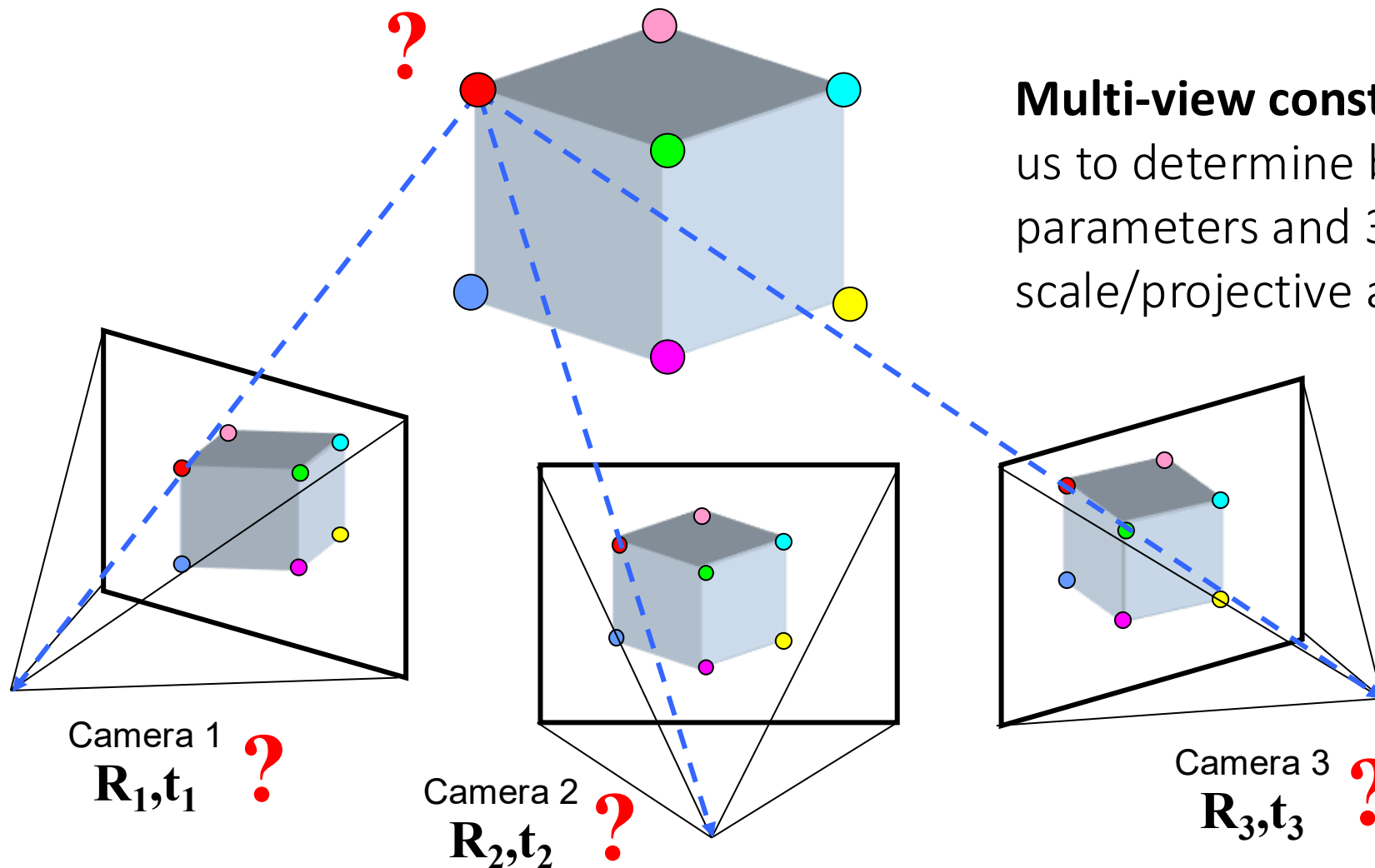
Multiple View Geometry



Multiple View Geometry

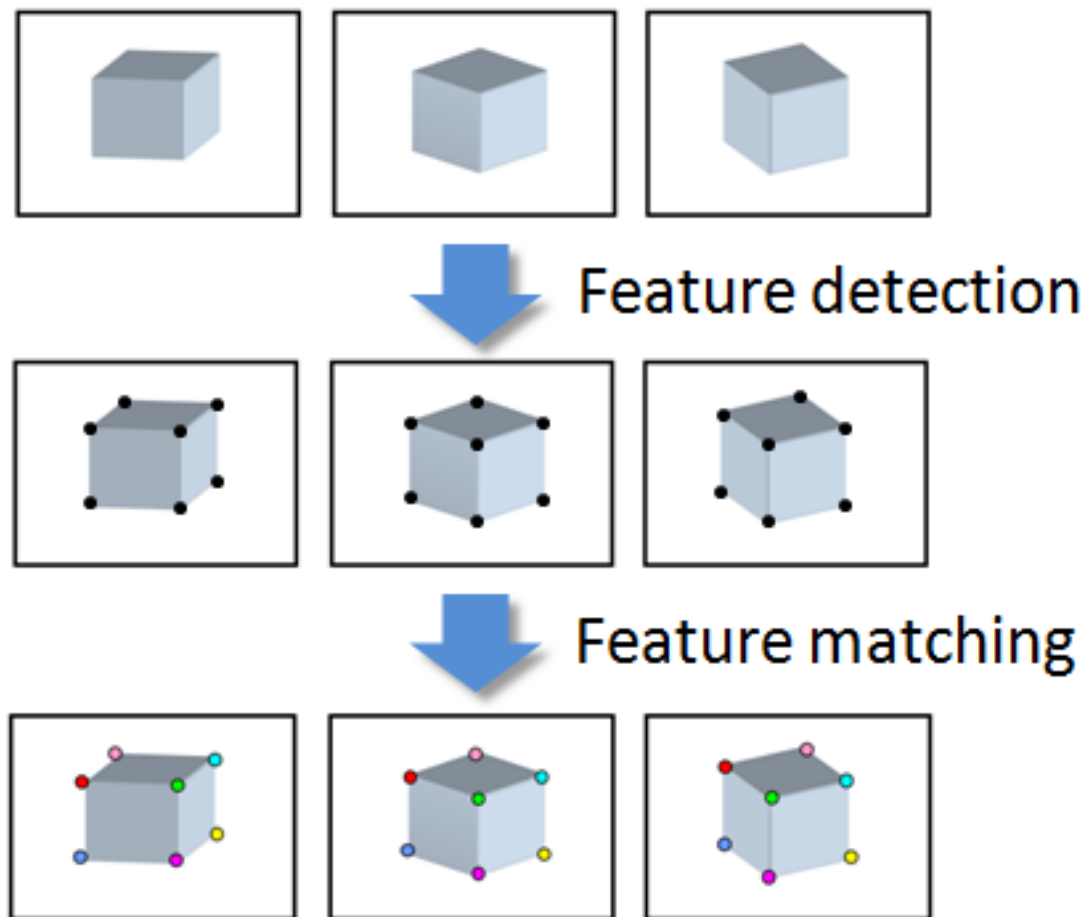


Multiple View Geometry

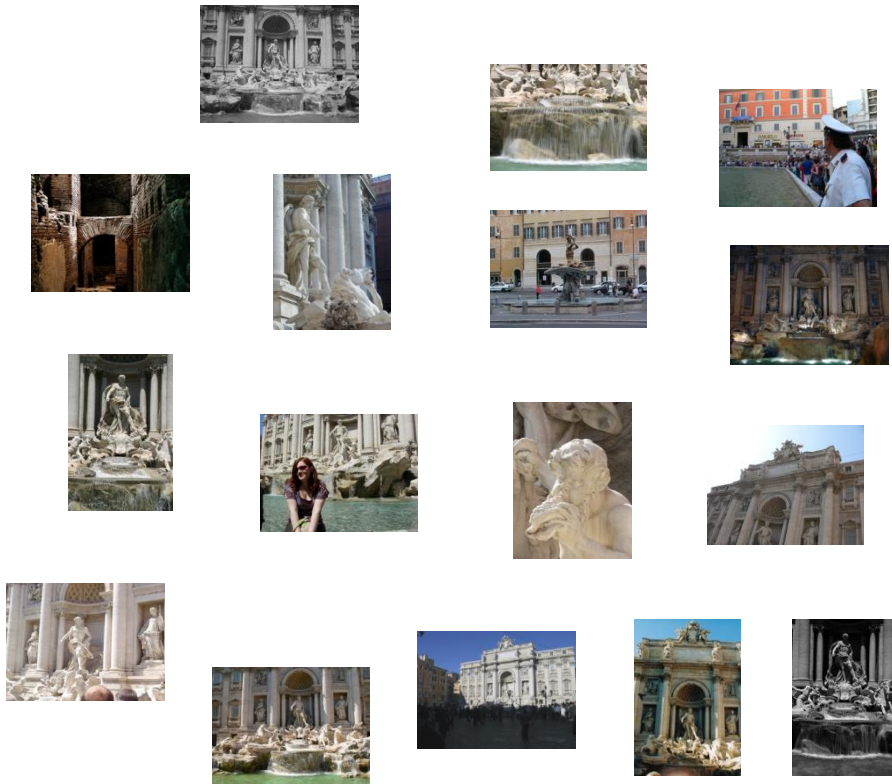


Multi-view constraints allow us to determine both camera parameters and 3D (up to scale/projective ambiguities)

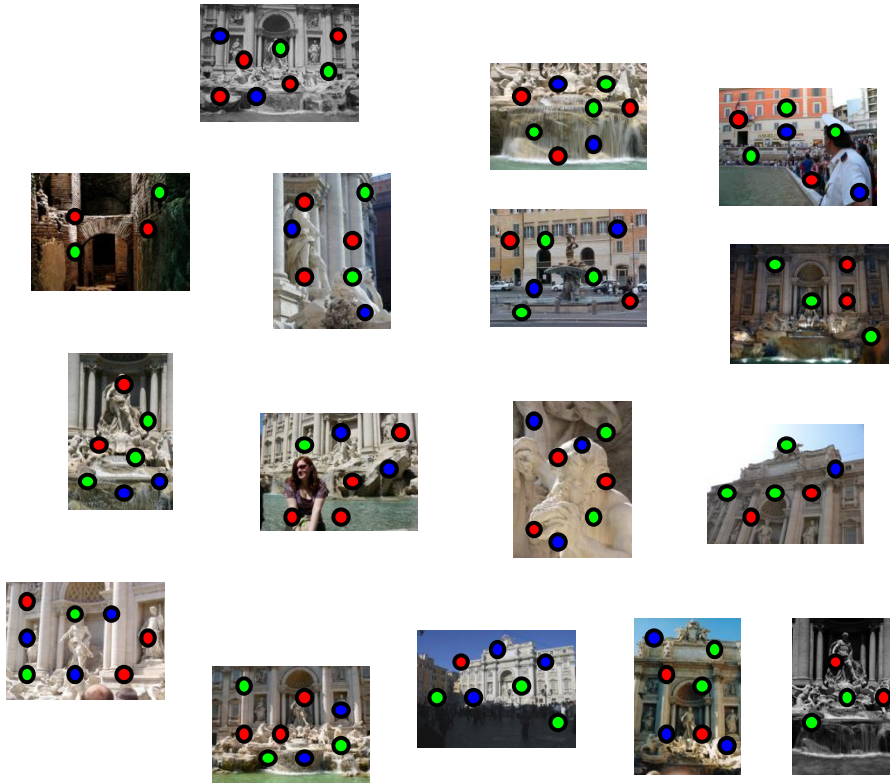
Estimating Correspondences



Estimating Correspondences



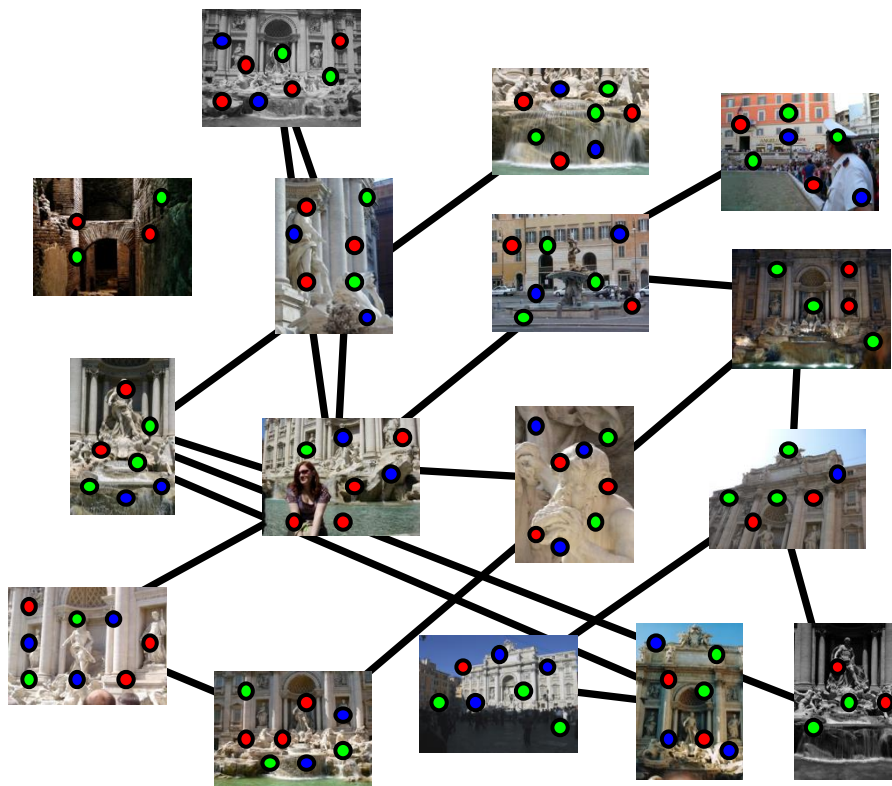
Estimating Correspondences



- **Detect feature points**

- SIFT descriptors [Lowe, 2004]
- Learned features – SuperPoint [DeTone, 2018])

Estimating Correspondences



- **Detect feature points**

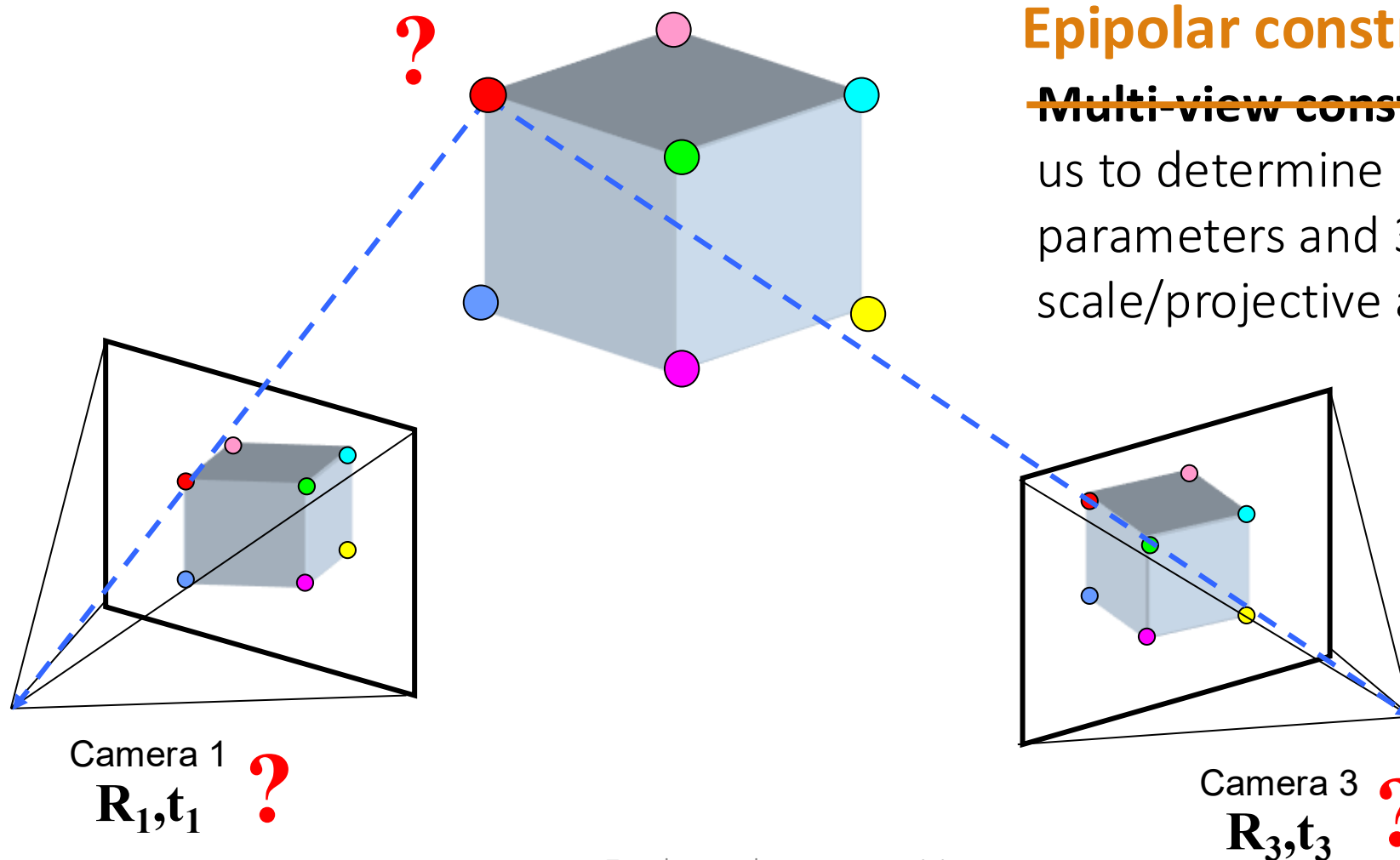
- SIFT descriptors [Lowe, 2004]
- Learned features – SuperPoint [DeTone, 2018])

- **Match features** between pair of images

- Use RANSAC to filter outliers
- Learned matching – SuperGlue [Sarlin, 2020]

Let's Start with Two Views – Epipolar Geometry

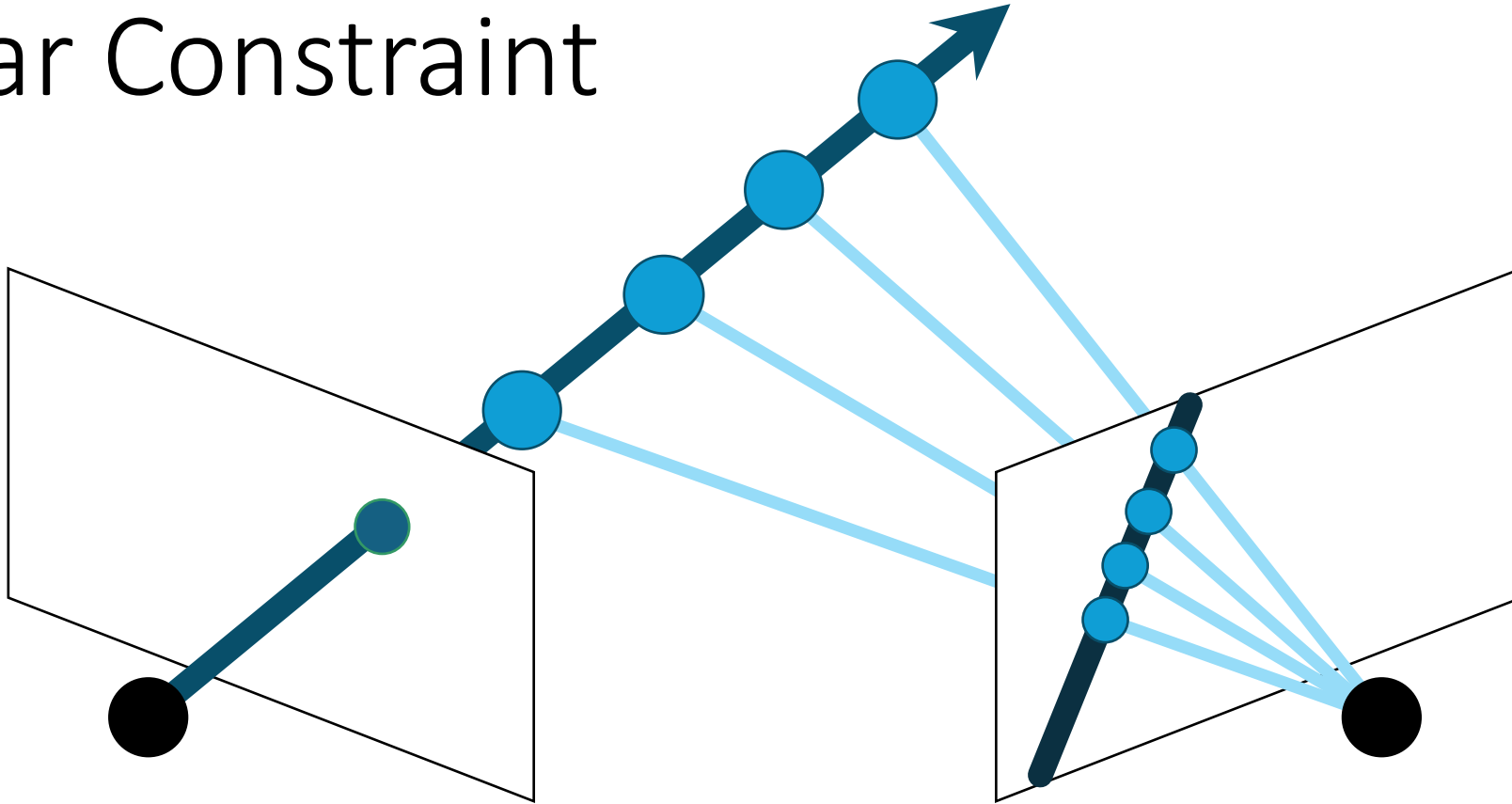
“epi-pole” \approx upon the pole/pivot



Epipolar constraints

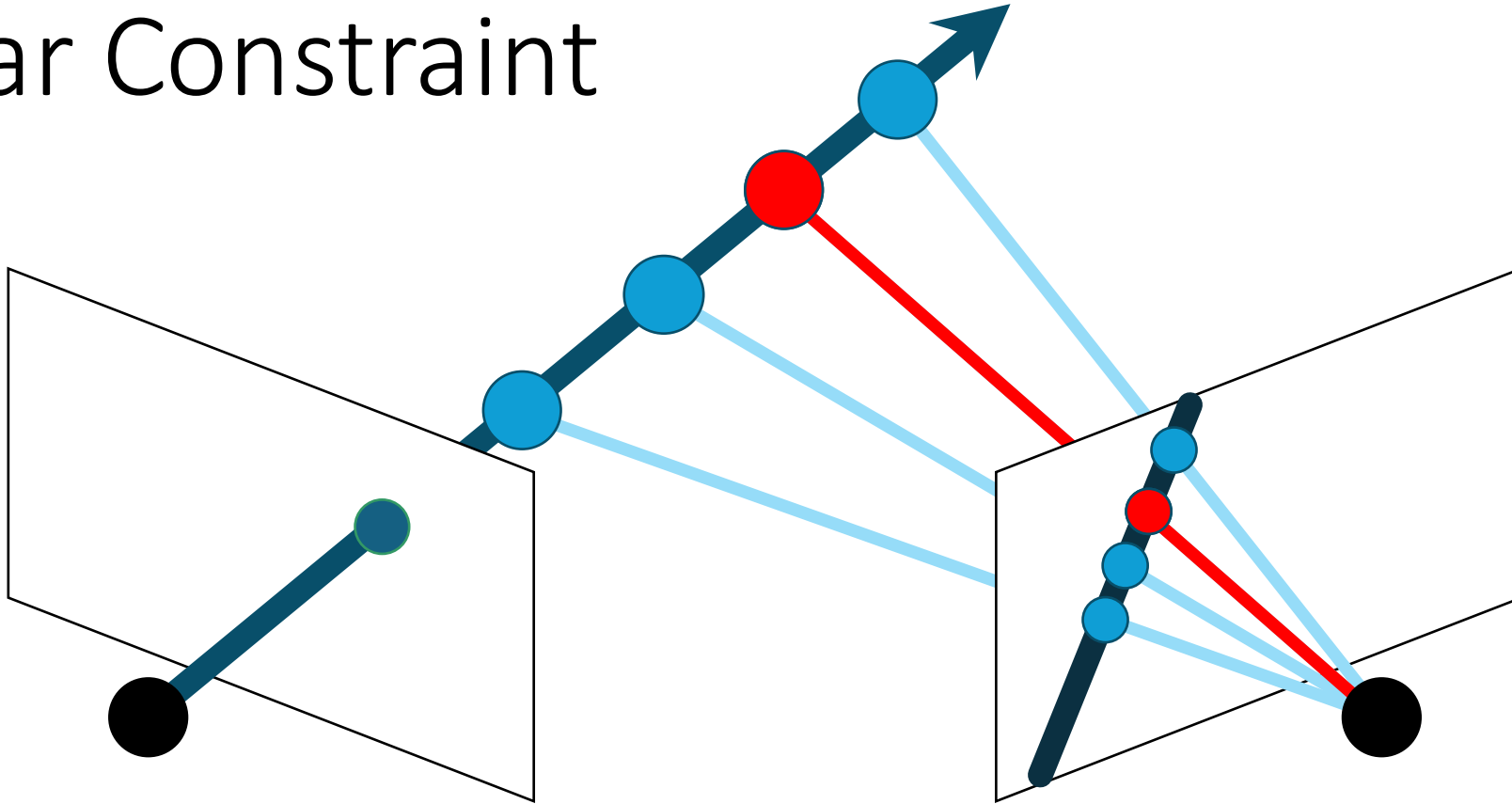
~~Multi-view constraints~~ allow us to determine both camera parameters and 3D (up to scale/projective ambiguities)

Epipolar Constraint



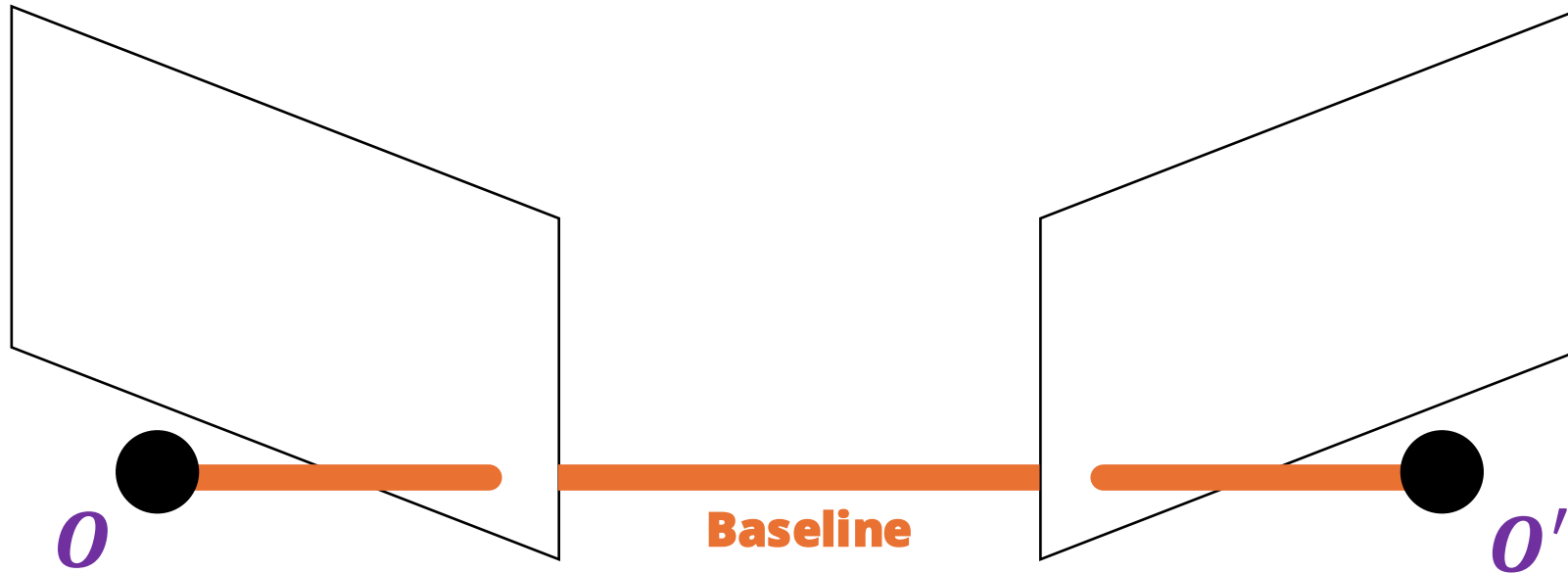
- Projecting points on a ray to another camera forms a **line**, i.e., the corresponding point on the second image *must* lie on this line

Epipolar Constraint



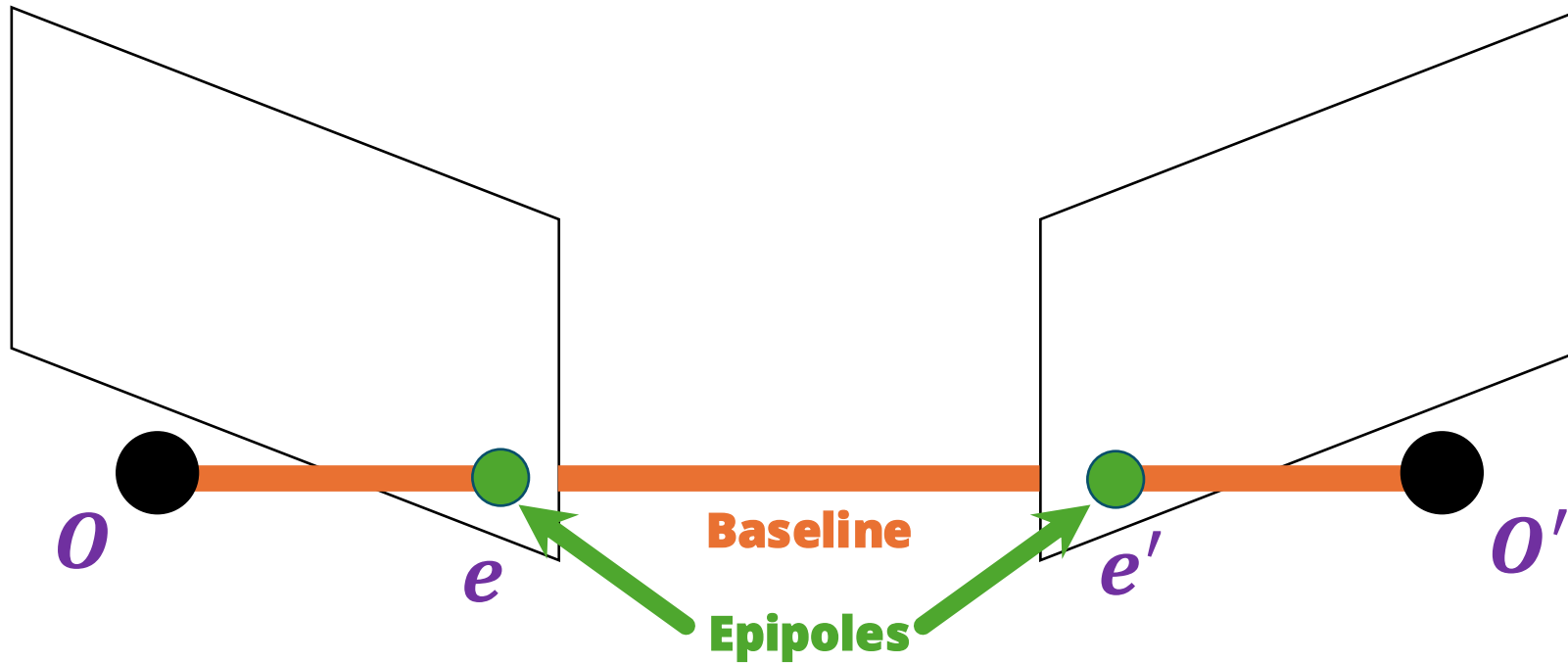
- Projecting points on a ray to another camera forms a **line**, i.e., the corresponding point on the second image *must* lie on this line
- Hence, if the *matched* 2D point is found along this line, we can determine its 3D location

Epipolar Geometry



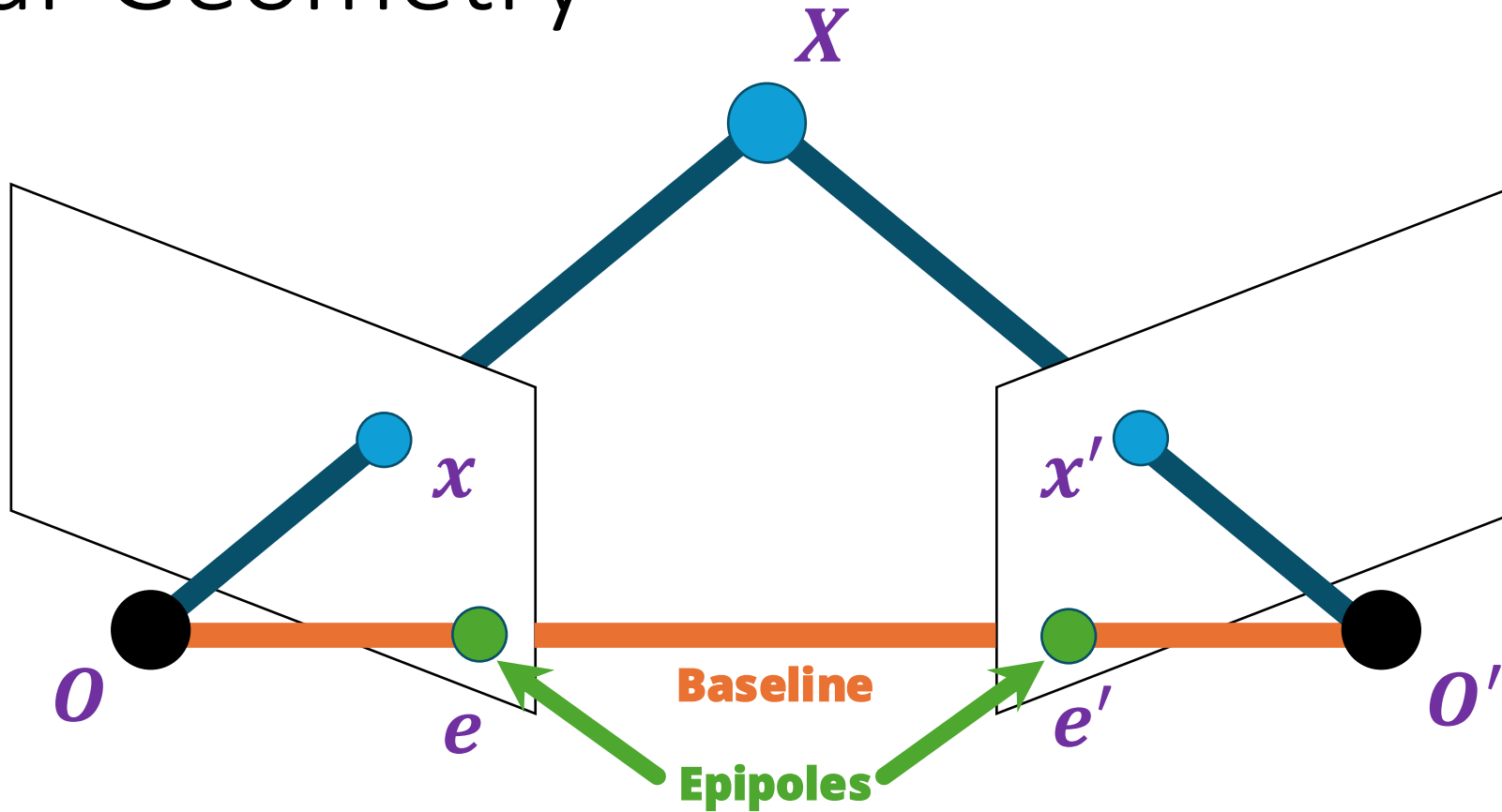
- Suppose we have two cameras with centers O, O'
- The **baseline** is the line connecting the origins

Epipolar Geometry



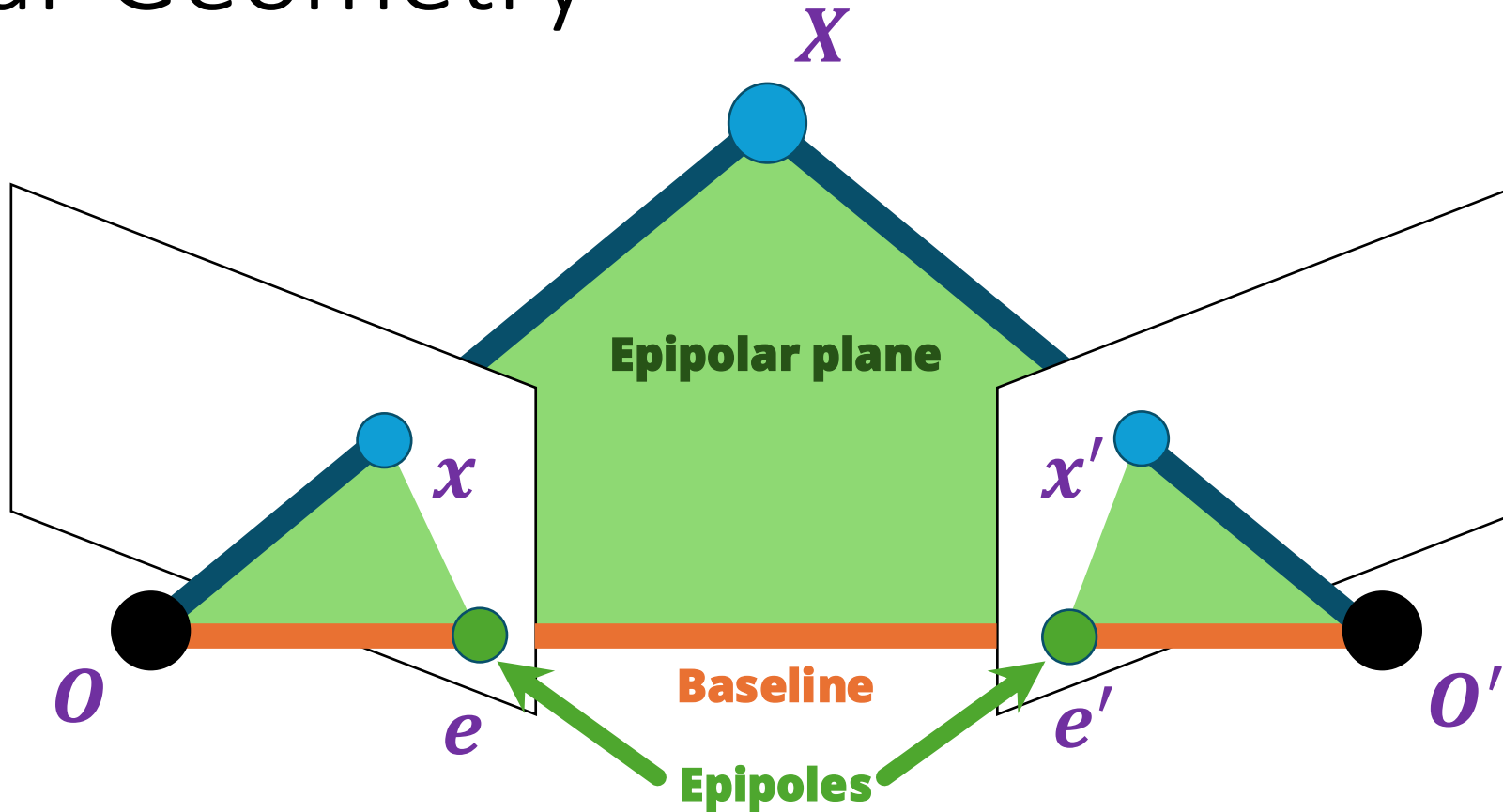
- **Epipoles** e, e' are where the baseline intersects the image planes, or projections of the other camera in each view

Epipolar Geometry



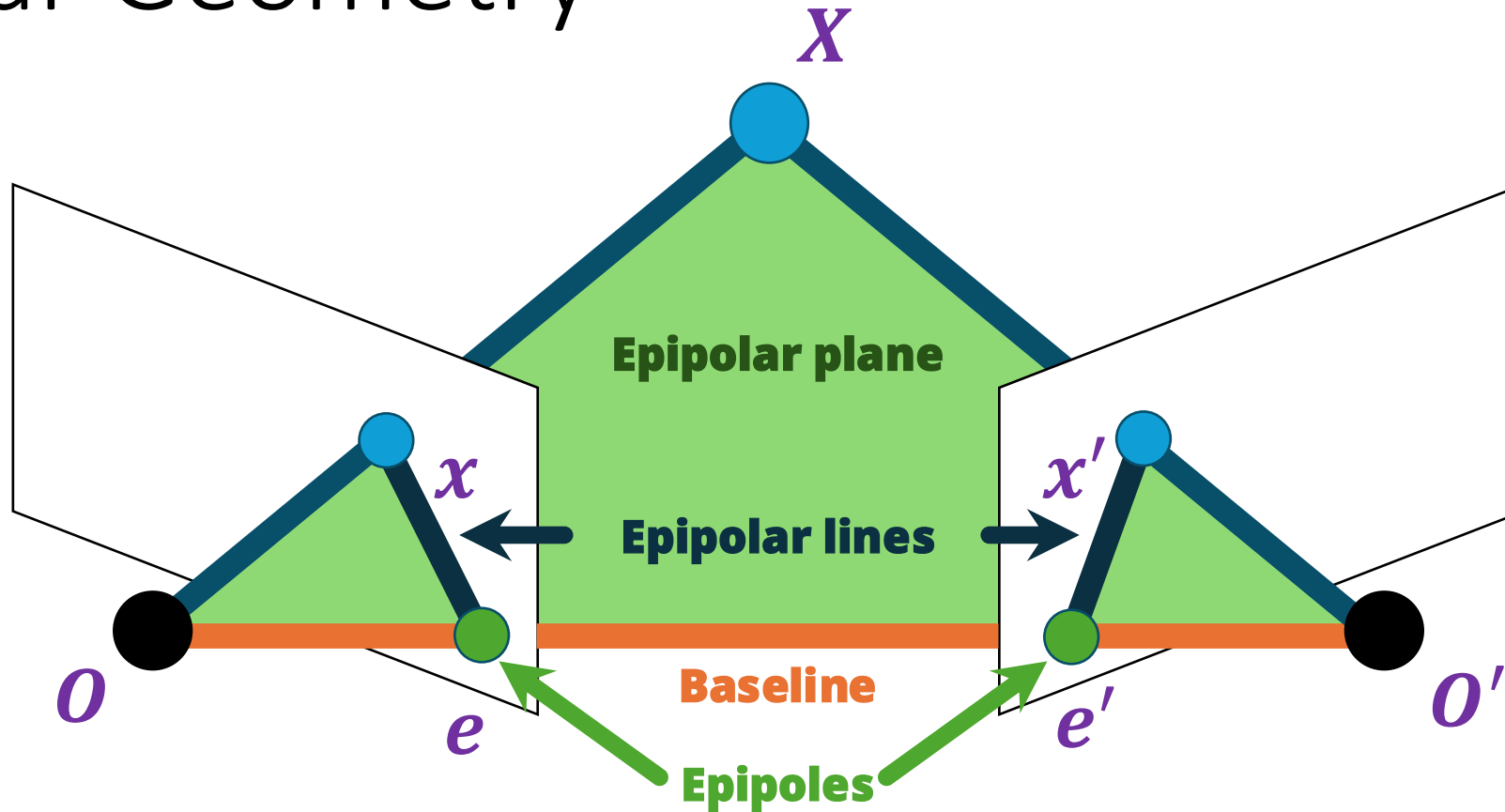
- Consider a **point** X , which projects to x and x'

Epipolar Geometry



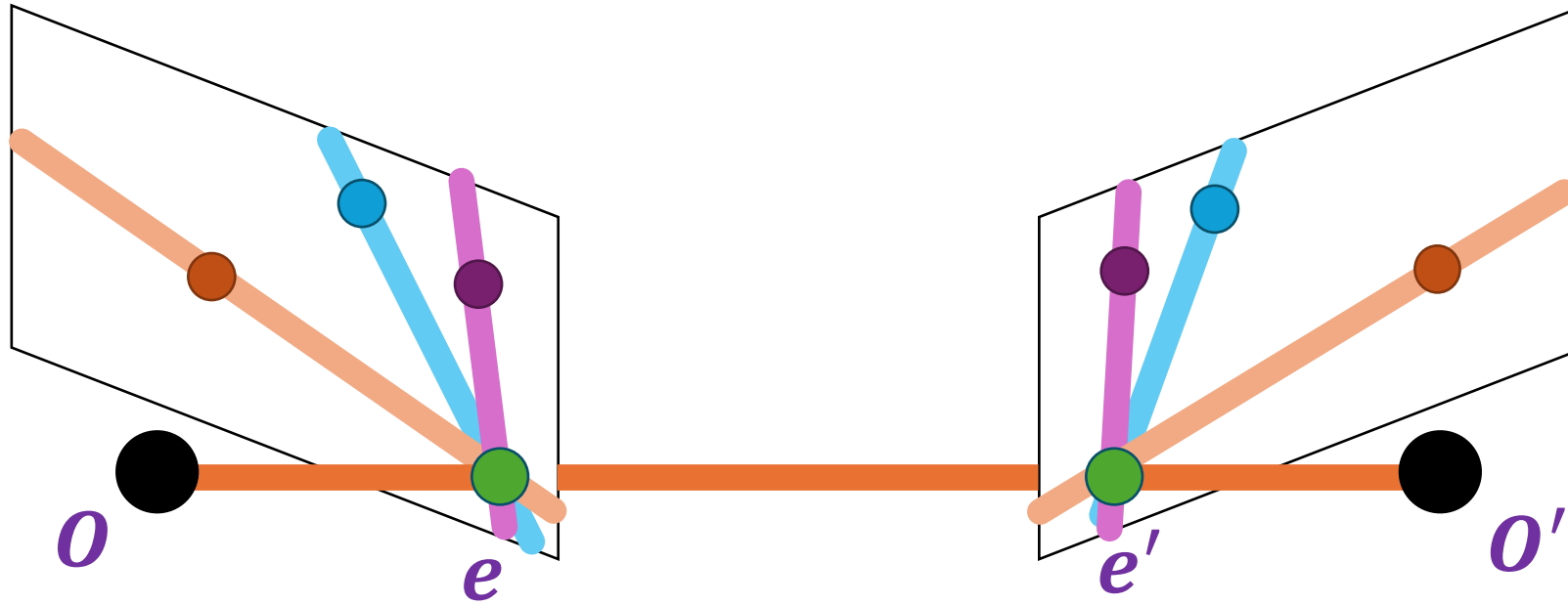
- The plane formed by X , O , and O' is called an **epipolar plane**
- There is a family of planes passing through O and O'

Epipolar Geometry



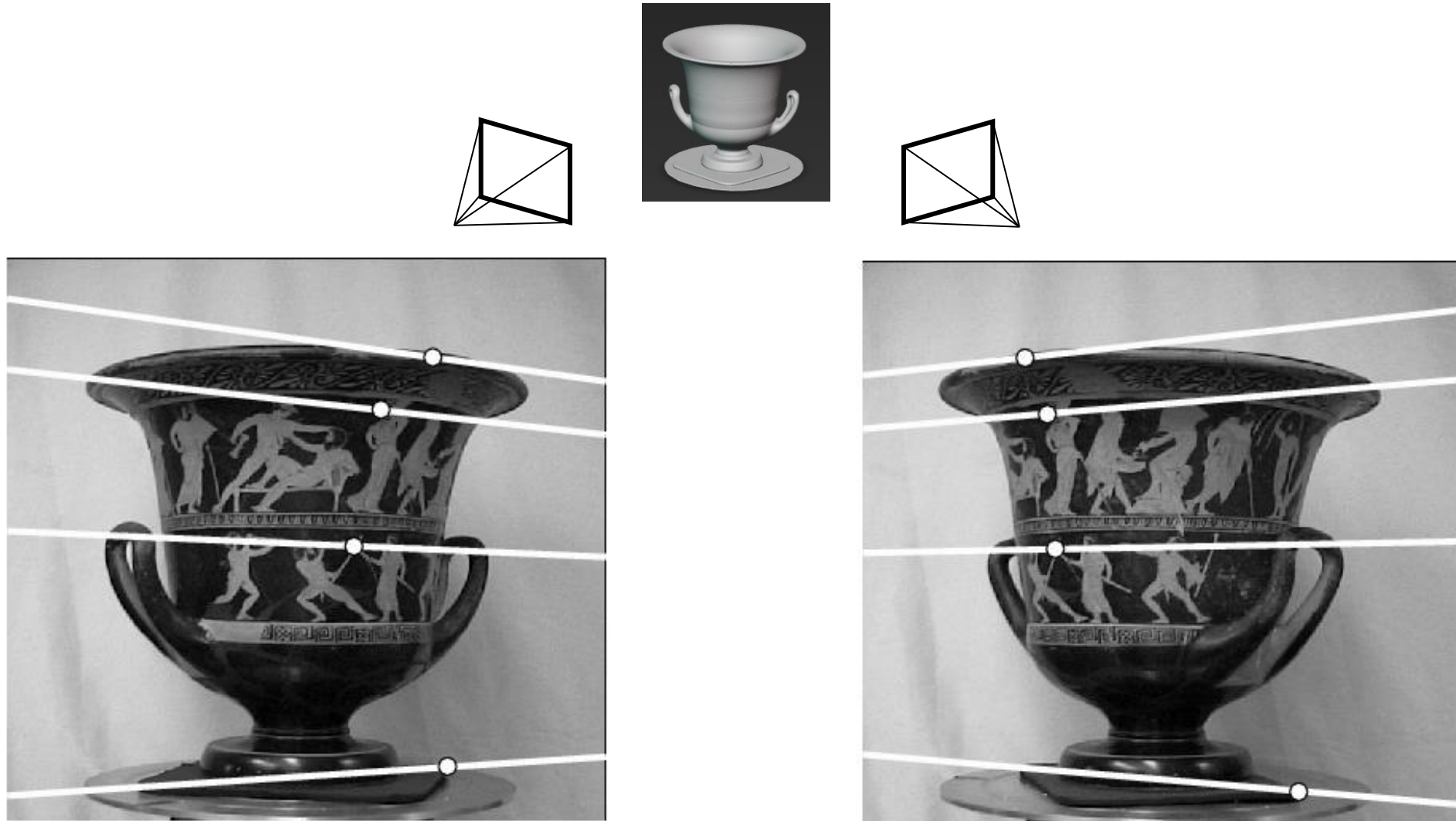
- **Epipolar lines** connect the epipoles to the projections of X
- Equivalently, they are intersections of the epipolar plane with the image planes – thus, they come in matching pairs

Epipolar Geometry

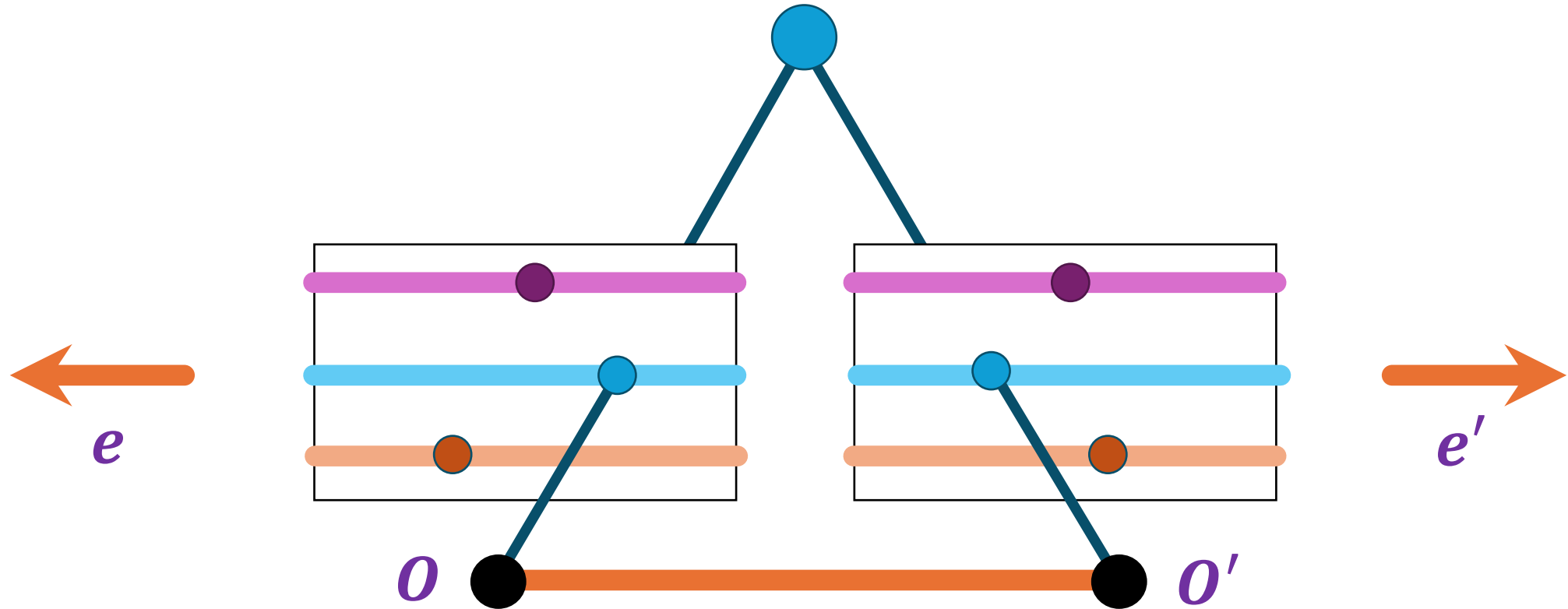


- All **epipolar lines** pass through the **epipoles**
- **Epipoles** can lie outside of the image

Example of Epipolar Lines

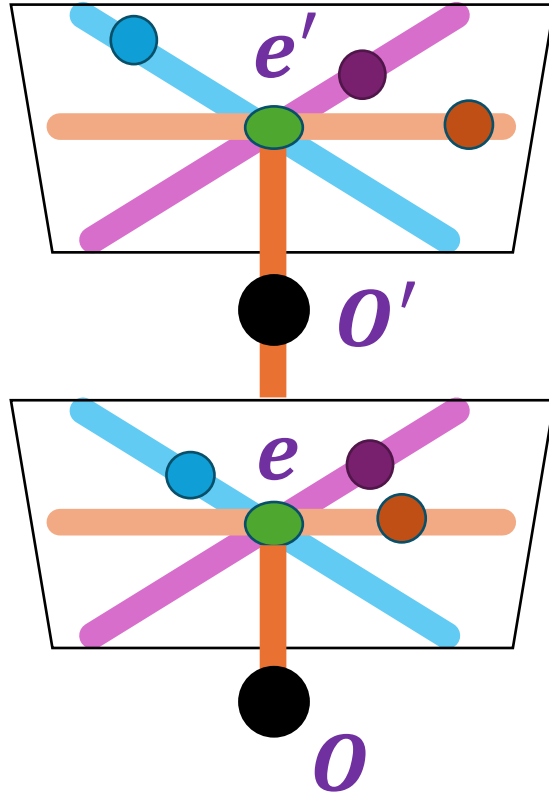


Parallel to Image Plane



- Where are the epipoles and what do the epipolar lines look like?
- Epipoles ***infinitely*** far away, epipolar lines parallel

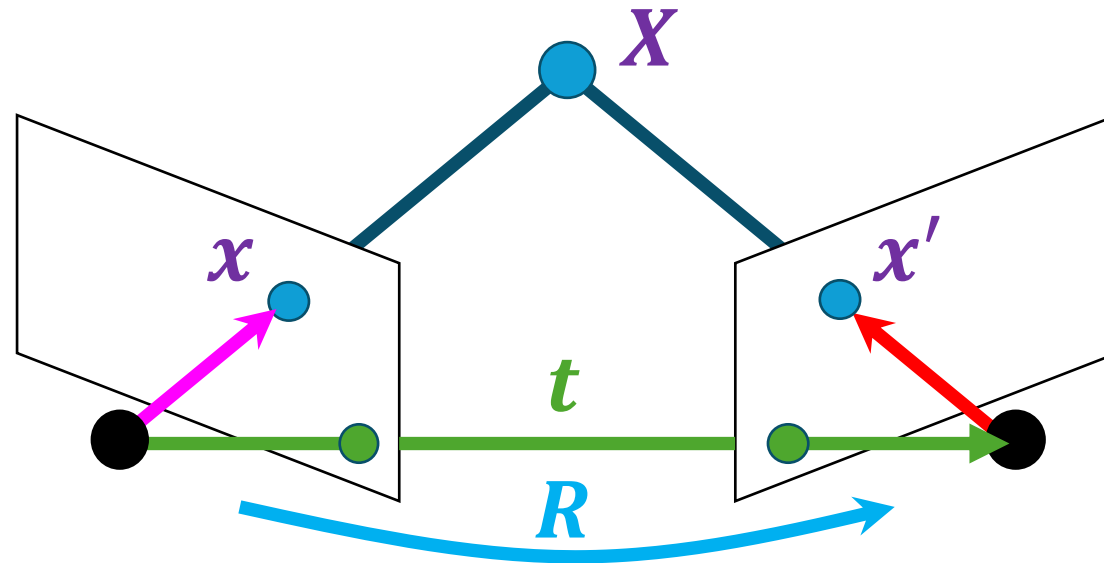
Perpendicular to Image Plane



- Where are the epipoles and what do the epipolar lines look like?
- Epipole is “focus of expansion” and coincides with the principal point of the camera
- Epipolar lines go out from principal point

Epipolar Constraint: Calibrated Case

$$x \cong K[R|t]X$$



- Assume the intrinsic and extrinsic parameters of the cameras are known (*calibrated*), and world coordinate system is set to that of the first camera
- Then the projection matrices are given by $K[I \mid \mathbf{0}]$ and $K'[R \mid t]$
- We can pre-multiply the projection matrices (and the image points) by the inverse calibration matrices to get *normalized* image coordinates:

$$x_{\text{norm}} = K^{-1}x_{\text{pixel}} \cong [I \mid \mathbf{0}]X, \quad x'_{\text{norm}} = K'^{-1}x'_{\text{pixel}} \cong [R \mid t]X$$

Epipolar Constraint: Calibrated Case

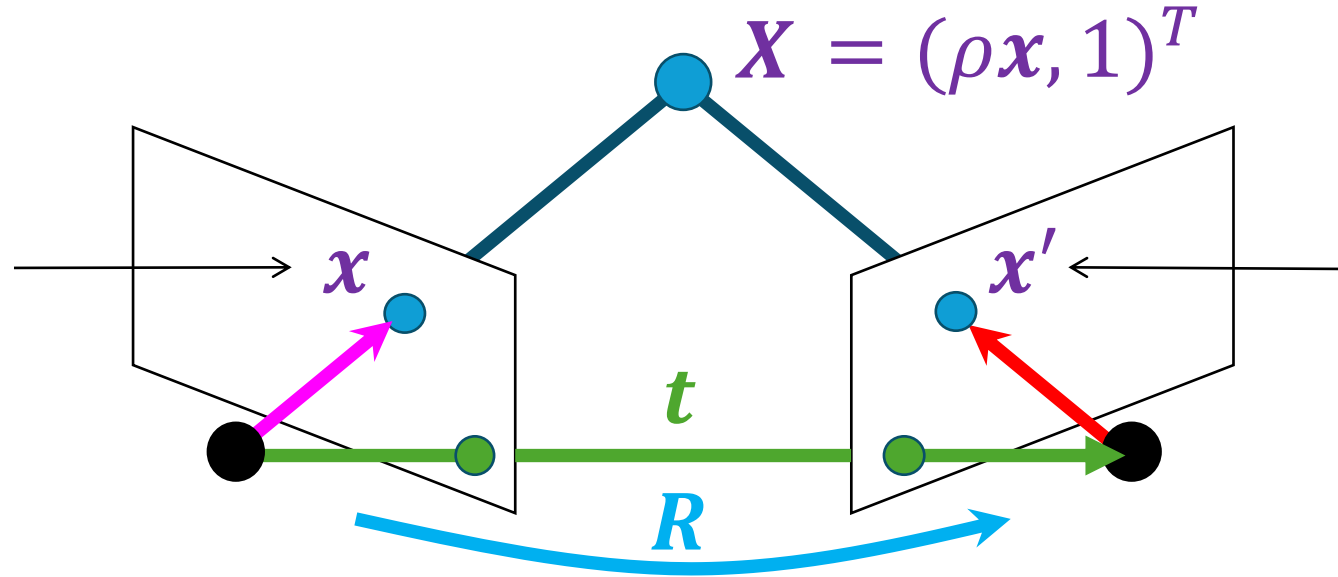
$$x \cong K[R|t]X$$

$$x_{\text{norm}} \cong [I | 0]X$$

$$x'_{\text{norm}} \cong [R | t]X$$

$$[I | 0] \begin{pmatrix} \rho x \\ 1 \end{pmatrix}$$

$$[R | t] \begin{pmatrix} \rho x \\ 1 \end{pmatrix} = \rho Rx + t$$



$$x' \cong \rho Rx + t \Rightarrow x' \cdot [t \times (Rx)] = 0$$

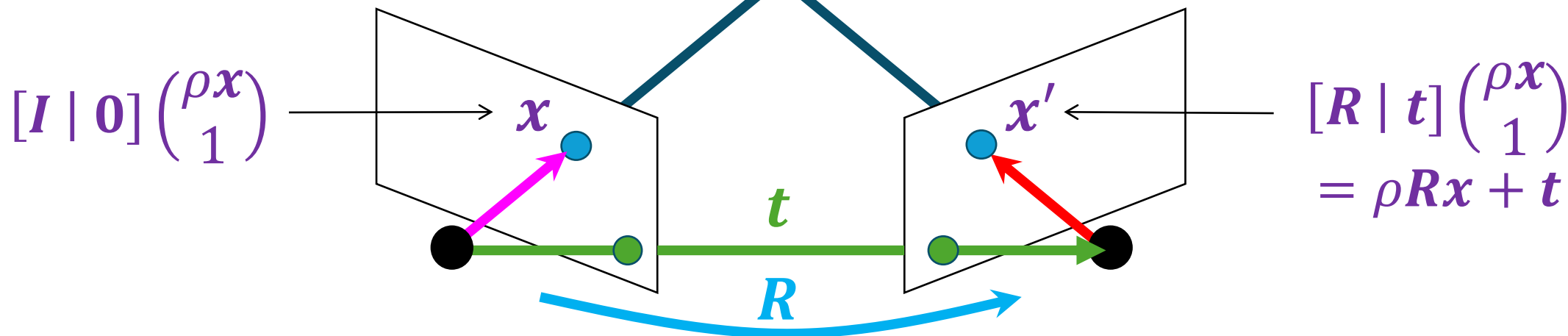
- This means the three vectors x' , Rx , and t are linearly dependent, i.e., lying on the same plane
- This constraint can be written using the *triple product*

Epipolar Constraint: Calibrated Case

$$x \cong K[R|t]X$$

$$x_{\text{norm}} \cong [I | 0]X$$

$$x'_{\text{norm}} \cong [R | t]X$$



$$x' \cong \rho Rx + t \Rightarrow x' \cdot [t \times (Rx)] = 0 \Rightarrow x'^T [t_{\times}] Rx = 0$$

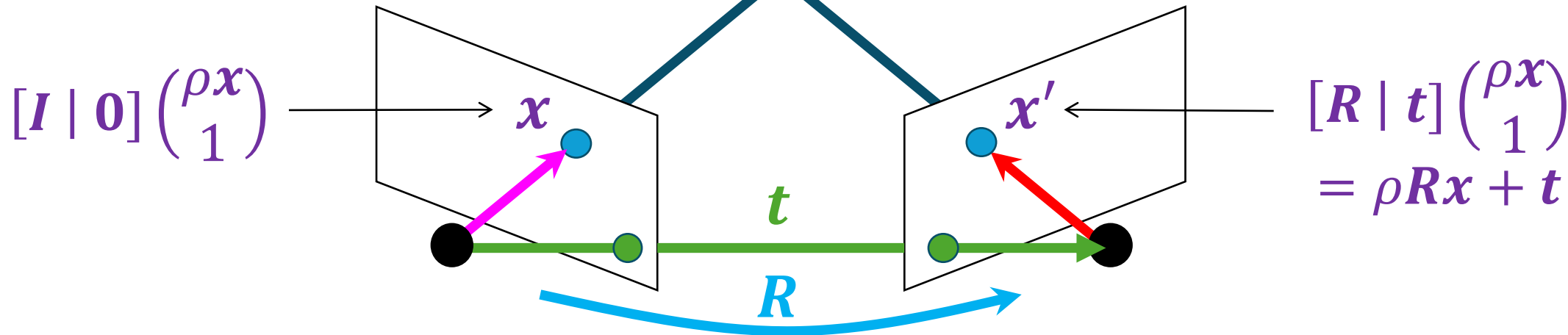
Recall: $a \times b = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = [a_{\times}]b \Rightarrow$ skew-symmetric matrix: $A^T = -A$

Epipolar Constraint: Calibrated Case

$$x \cong K[R|t]X$$

$$x_{\text{norm}} \cong [I | 0]X$$

$$x'_{\text{norm}} \cong [R | t]X$$

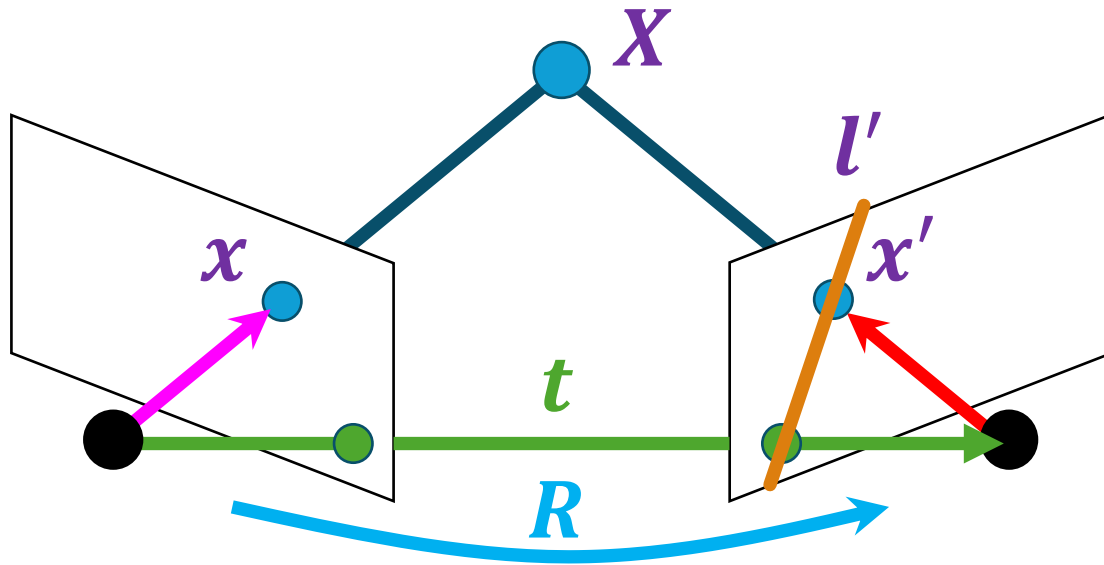


$$x' \cong \rho Rx + t \Rightarrow x' \cdot [t \times (Rx)] = 0 \Rightarrow x'^T [t_{\times}] Rx = 0 \Rightarrow x'^T E x = 0$$



Essential Matrix

The Essential Matrix



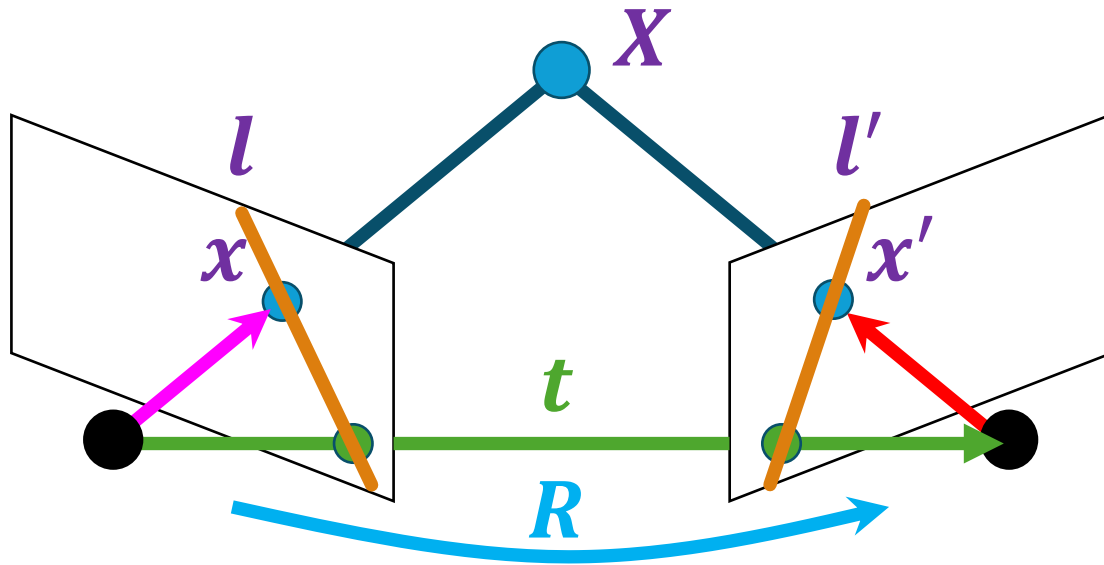
$$l' = E^T x$$

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

$$(x', y', 1) \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = 0$$

- $E x$ is the **epipolar line** associated with x ($l' = E x$)
- Recall: a line is given by $a x + b y + c = 0$ or $l^T x = 0$ in homogeneous coordinates, where $l = (a, b, c)^T$ and $x = (x, y, 1)^T$
- $x'^T E x = x'^T l' = 0$ means x' lies on the epipolar line l'

The Essential Matrix



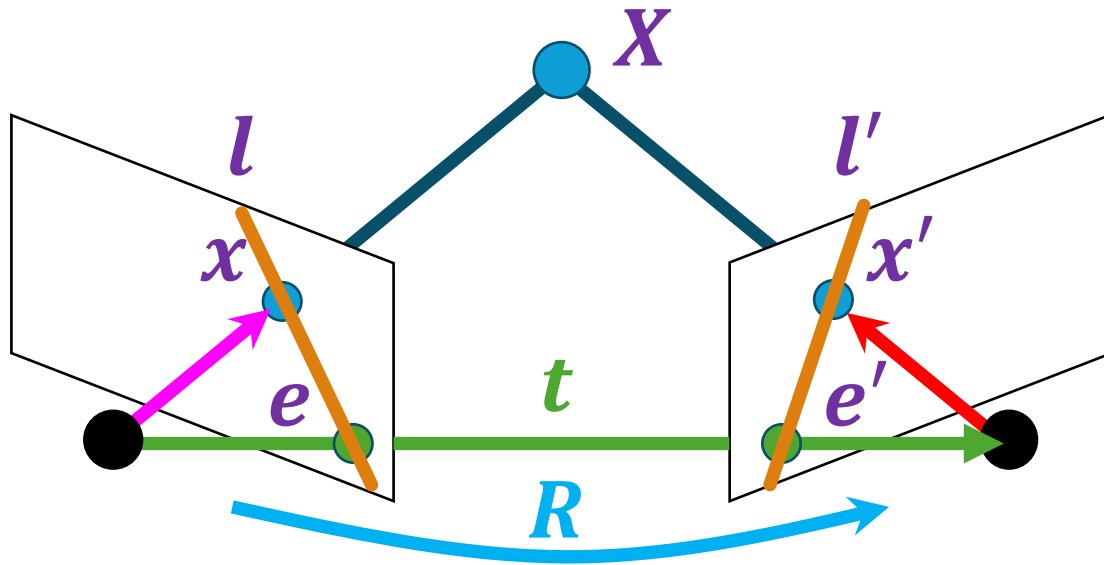
$$l = E^T x'$$

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

$$(x', y', 1) \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = 0$$

- $E x$ is the **epipolar line** associated with x ($l' = E x$)
- $x'^T E x = x'^T l' = 0$ means x' lies on the epipolar line l'
- Equivalently, $E^T x'$ is the **epipolar line** associated with x' ($l = E^T x'$)
- $x'^T E x = l x = 0$ means x lies on the epipolar line l

The Essential Matrix



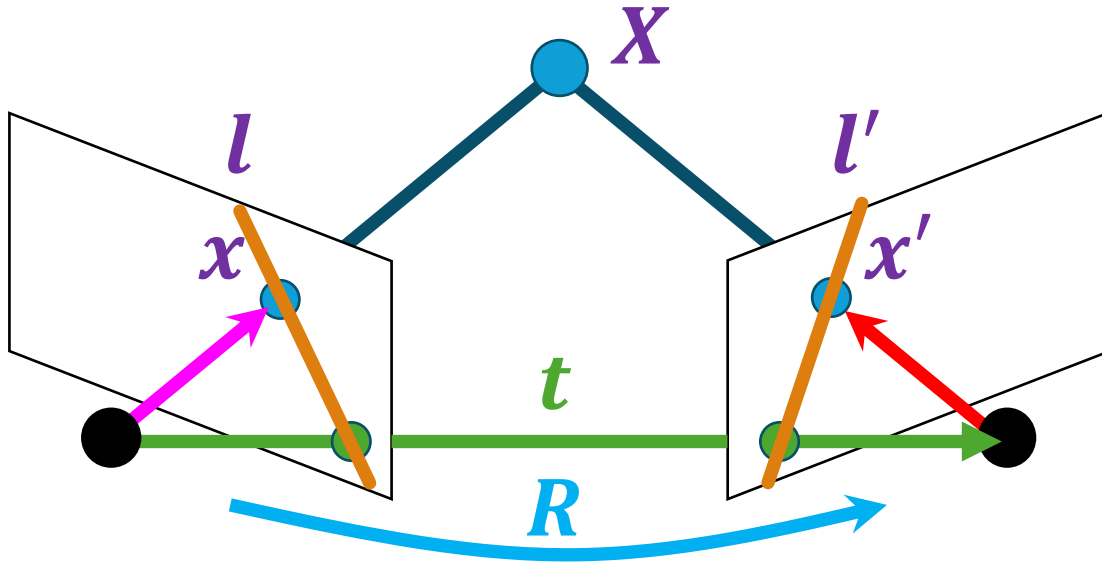
$$\mathbf{x}'^T \mathbf{E} \mathbf{x} = 0$$

$$(x', y', 1) \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = 0$$

- $\mathbf{E} \mathbf{x}$ is the **epipolar line** associated with \mathbf{x} ($\mathbf{l}' = \mathbf{E} \mathbf{x}$)
- $\mathbf{E}^T \mathbf{x}'$ is the **epipolar line** associated with \mathbf{x}' ($\mathbf{l} = \mathbf{E}^T \mathbf{x}'$)
- $\mathbf{E} \mathbf{e} = \mathbf{0}$ and $\mathbf{E}^T \mathbf{e}' = \mathbf{0}$, where \mathbf{e}, \mathbf{e}' are the epipoles
- \mathbf{E} is singular (rank two) and has five degrees of freedom => why?
 - $\mathbf{E} = [\mathbf{t}_\times] \mathbf{R}$; $[\mathbf{t}_\times]$ is skew-symmetric; has rank 2
 - \mathbf{R} : 3 DoF, \mathbf{t} : 3 DoF, but we lost 1 DoF due to scale; move along \mathbf{t} doesn't change \mathbf{l}

Epipolar Constraint: Uncalibrated

$$x \cong K[R|t]X$$



$$x'^T E x = 0$$

$$(x', y', 1) \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = 0$$

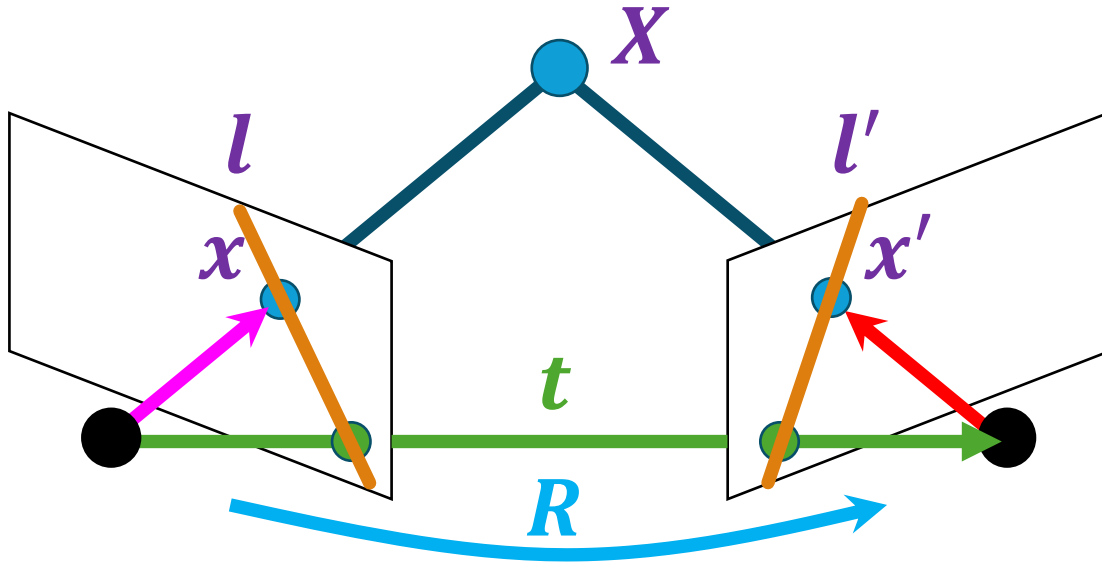
- What if camera intrinsics K, K' are unknown?
- We can write the epipolar constraint:

$$x'_{\text{norm}}^T E x_{\text{norm}} = 0$$

$$\text{where } x_{\text{norm}} = K^{-1}x, x'_{\text{norm}} = K'^{-1}x'$$

Epipolar Constraint: Uncalibrated

$$x \cong K[R|t]X$$



$$x'^T E x = 0$$

$$(x', y', 1) \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = 0$$

- What if camera intrinsics K, K' are unknown?
- We can write the epipolar constraint:

Fundamental Matrix

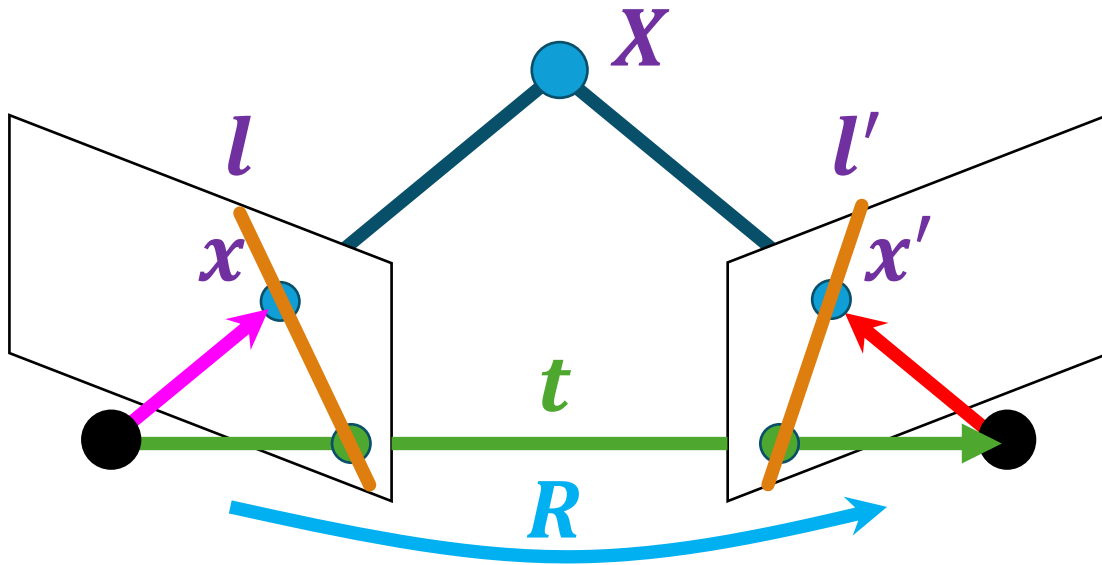
$$F = K'^{-T} E K^{-1}$$

$$x'_{\text{norm}}^T E x_{\text{norm}} = x'^T \boxed{K'^{-T} E K^{-1}} x = 0$$

where $x_{\text{norm}} = K^{-1}x, x'_{\text{norm}} = K'^{-1}x'$

The Fundamental Matrix

$$x \cong K[R|t]X$$



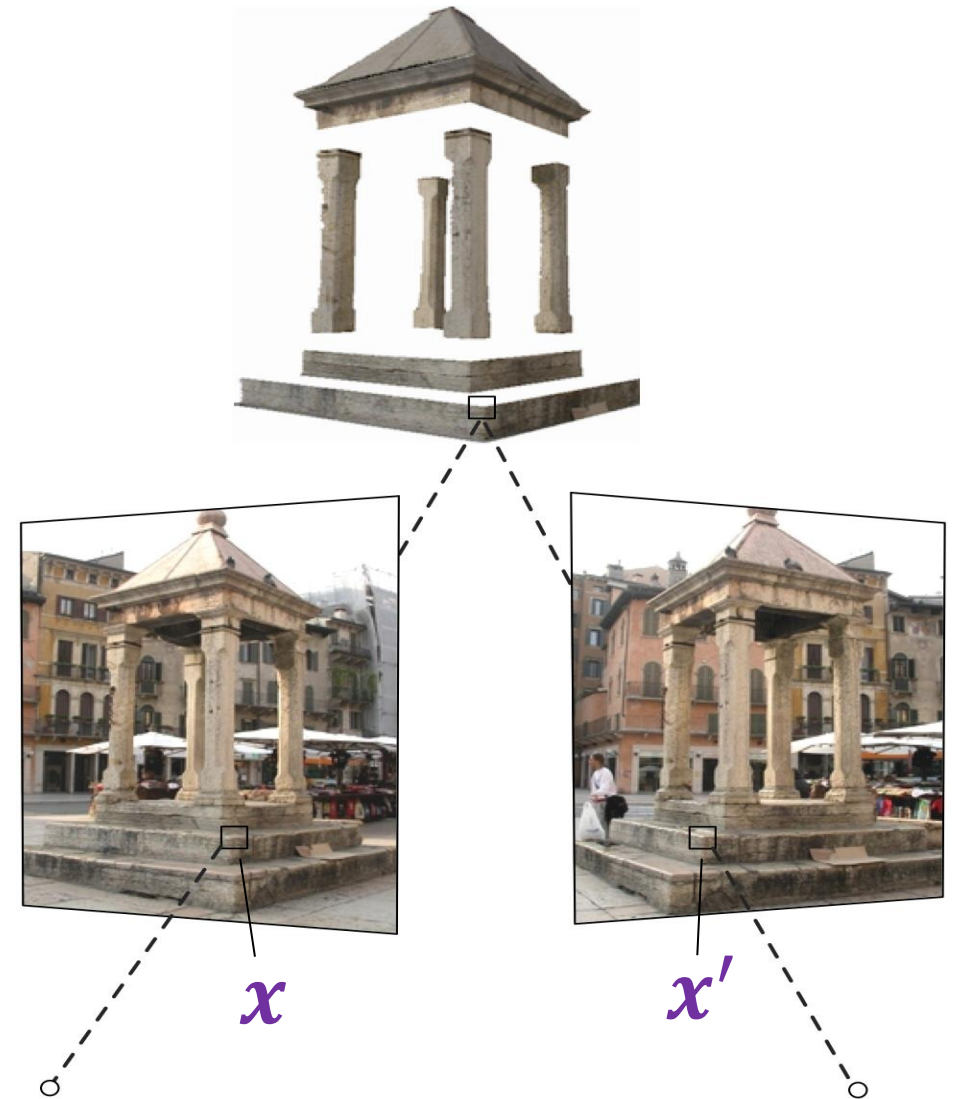
$$x'^T F x = 0$$

$$(x', y', 1) \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = 0$$

- Fx is the **epipolar line** associated with x ($l' = Fx$)
- $F^T x'$ is the **epipolar line** associated with x' ($l = F^T x'$)
- $Fe = 0$ and $F^T e' = 0$, where e, e' are the epipoles
- F is singular (rank two) and has seven degrees of freedom => why?
 - $F = K'^{-T}[t_x]RK^{-1}$! $[t_x]$ is skew-symmetric; has rank 2
 - 9 elements, but we lost 1 DoF to scale, and 1 DoF to rank constraint

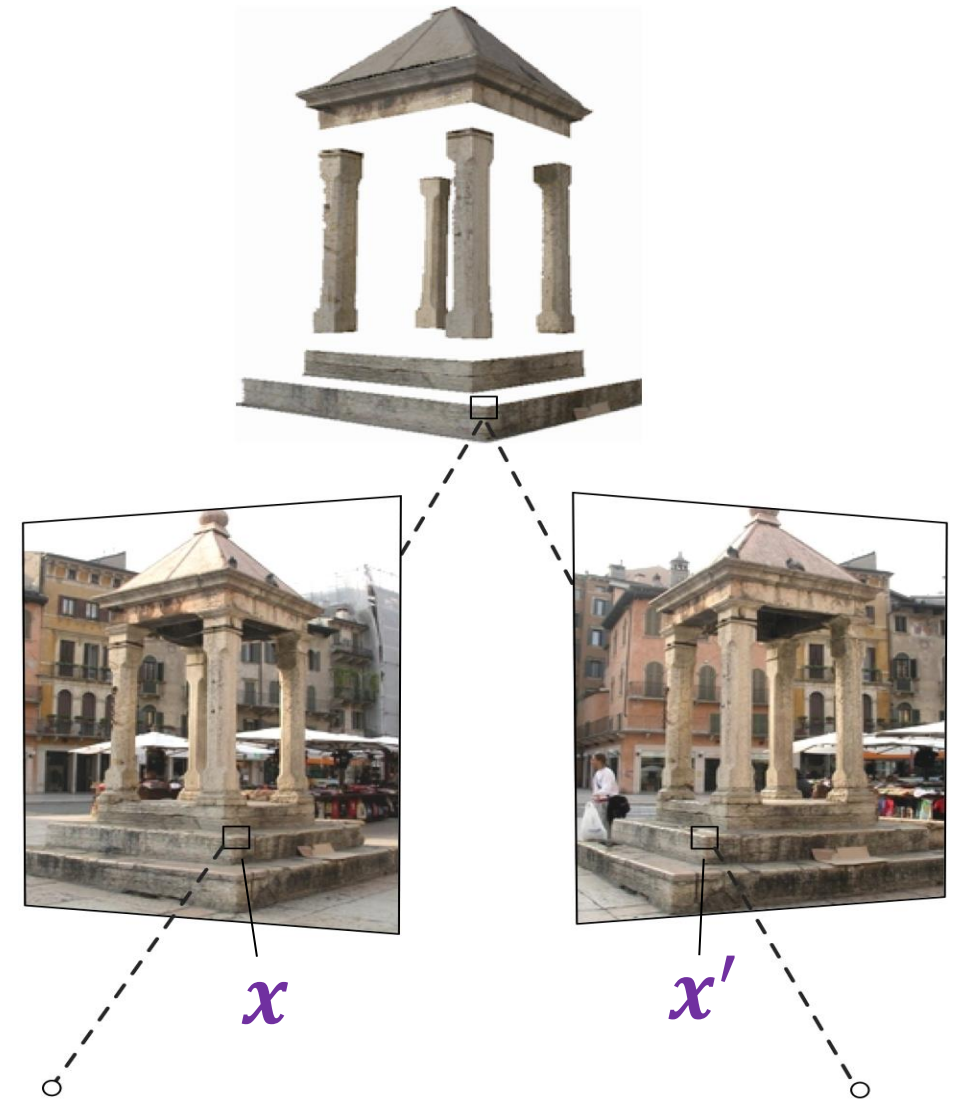
How Can We Use the Epipolar Constraint?

- **Given:** $F, \mathbf{x}, \mathbf{x}'$
- **Q:** does there exist a 3D point that projects to \mathbf{x} and \mathbf{x}' ?
- **A:** Yes, if $\text{residual}(\mathbf{x}'^T F \mathbf{x})$ is sufficiently low
- Note: the interpretation of $\text{residual}(\mathbf{x}'^T F \mathbf{x})$ is the distance (geometric or algebraic) between \mathbf{x} and $\mathbf{l} = F^T \mathbf{x}'$, or \mathbf{x}' and $\mathbf{l}' = F \mathbf{x}$



How Can We Use the Epipolar Constraint?

- **Given:** F
- **Q:** how do we find R, t ?
- **A:**
 - **Step 0:** estimate K, K' if not known (self-calibration)
 - **Step 1:** compute $E = K'^T F K$
 - **Step 2:** since $E = [t_\times] R$, perform SVD on $E = U \Sigma V^T$. We have 4 solutions $(R_1, \pm t)$, $(R_2, \pm t)$, where $t = U[:, 3]$, $R_1 = U W V^T$, $R_2 = U W^T V^T$, and W is the matrix that rotates 90° about z-axis.
 - **Step 3:** pick the one that gives 3D points in front of both cameras (cheirality check).



How to Estimate the Fundamental Matrix?

- **Given:** correspondences $\mathbf{x}_i = (x_i, y_i, 1)^T$ and $\mathbf{x}'_i = (x'_i, y'_i, 1)^T$
- **Constraint:** $\mathbf{x}'^T \mathbf{F} \mathbf{x} = 0$

$$(x', y', 1) \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = 0 \quad \Rightarrow \quad (x'x, x'y, x', y'x, y'y, y', x, y, 1) \begin{pmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix} = 0$$

How to Estimate the Fundamental Matrix?

- **Given:** correspondences $\mathbf{x}_i = (x_i, y_i, 1)^T$ and $\mathbf{x}'_i = (x'_i, y'_i, 1)^T$
- **Constraint:** $\mathbf{x}'^T \mathbf{F} \mathbf{x} = 0$

$$\overbrace{\begin{bmatrix} x'_1 x_1 & x'_1 y_1 & x'_1 & y'_1 x_1 & y'_1 y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_n x_n & x'_n y_n & x'_n & y'_n x_n & y'_n y_n & y'_n & x_n & y_n & 1 \end{bmatrix}}^{\mathbf{A}} \begin{pmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix} = \mathbf{0}$$

Homogeneous least squares to find \mathbf{f} :

$$\arg \min_{\|\mathbf{f}\|=1} \|\mathbf{A} \mathbf{f}\|_2^2 \longrightarrow \text{Least eigenvector of } \mathbf{A}^T \mathbf{A}$$

This is known as the “*eight-point algorithm*” introduced by Christopher Longuet-Higgins (1981).

Small Trick to Enforce Rank-2 Constraint

- We know \mathbf{F} must be singular/rank 2. How do we force that?
- **Solution:** take SVD of the initial estimate and throw out the smallest singular value

$$\mathbf{F}_{\text{init}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$



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



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

$$\mathbf{F} = \mathbf{U}\mathbf{\Sigma}'\mathbf{V}^T$$

The Fundamental Matrix Song



By Daniel Wedge (2008): <https://danielwedge.com/fmatrix/>

The Fundamental Matrix Song – Live!

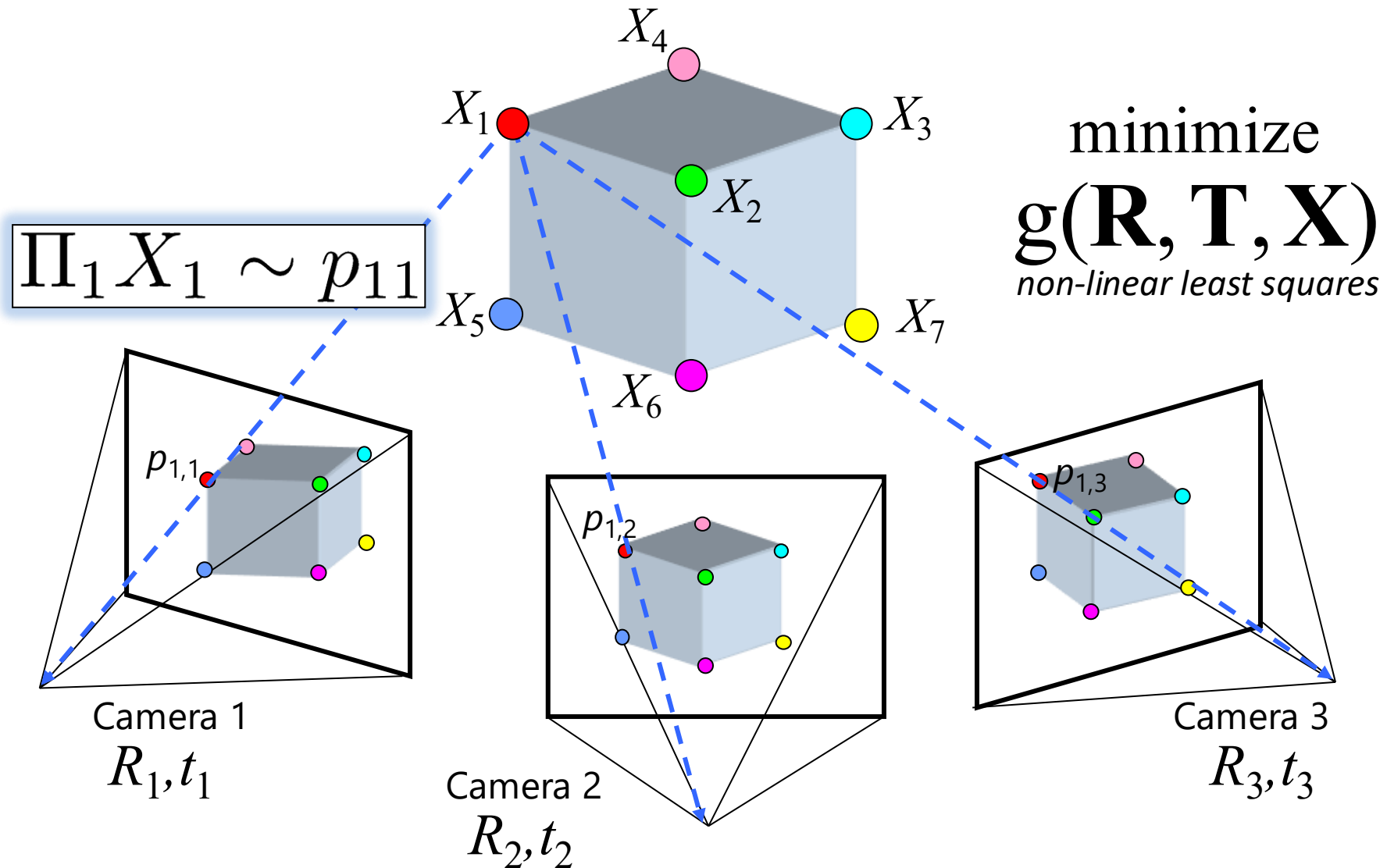


Daniel Wedge and the CVPR house band at CVPR 2023 in Vancouver

More Than Two Views?

- The geometry of three views is described by a $3 \times 3 \times 3$ tensor called the *trifocal tensor*
- The geometry of four views is described by a $3 \times 3 \times 3 \times 3$ tensor called the *quadrifocal tensor*
- After this it starts to get complicated...
- Or we can pose it as an optimization problem

Structure from Motion (SfM)



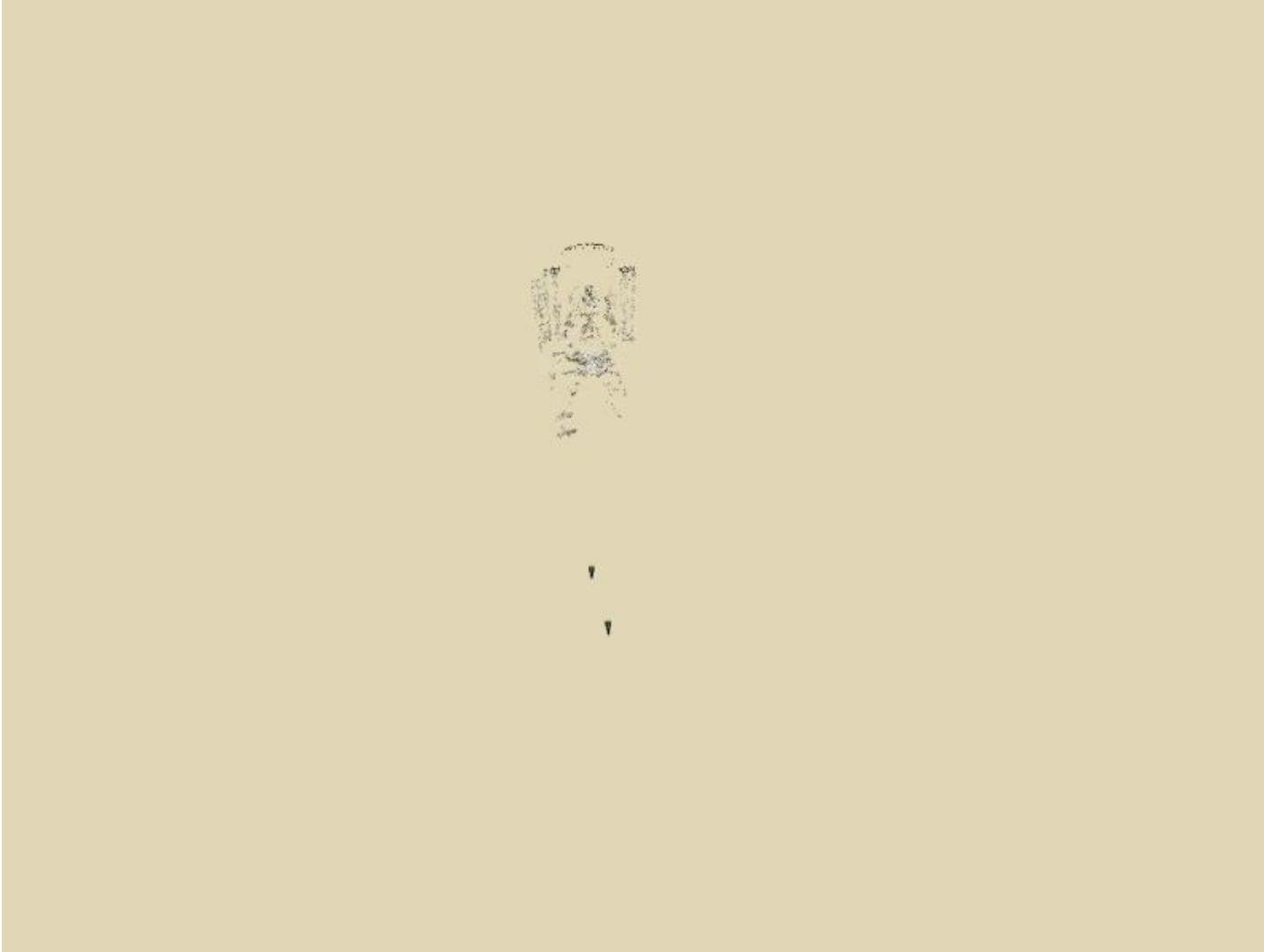
Bundle Adjustment

- Minimize reprojection errors:

$$g(\mathbf{X}, \mathbf{R}, \mathbf{T}) = \sum_{i=1}^m \sum_{j=1}^n \underbrace{w_{ij}}_{\substack{\text{indicator variable:} \\ \text{is point } i \text{ visible in image } j?}} \cdot \left\| \underbrace{\mathbf{P}(\mathbf{x}_i, \mathbf{R}_j, \mathbf{t}_j)}_{\substack{\text{predicted} \\ \text{image location}}} - \underbrace{\begin{bmatrix} u_{i,j} \\ v_{i,j} \end{bmatrix}}_{\substack{\text{observed} \\ \text{image location}}} \right\|^2$$

- Optimized using non-linear least squares, e.g., Levenberg-Marquardt algorithm
- Susceptible to local minima; requires careful initialization

Incremental Structure from Motion



- Photo Tourism (Snavely et al., SIGGRAPH'06)
- Trevis Fountain, Rome
 - 466 Internet photos
 - > 100,000 3D points
 - Very large optimization problem

Practical SfM Tools

- COLMAP (by Schönberger et al.): <https://colmap.github.io/>
- nerfstudio COLMAP Python wrapper:
https://docs.nerf.studio/quickstart/custom_dataset.html
- Still, with thousands of images, it can take many hours or even days!
- There's a family of methods optimized for efficiency and continuous streams, often referred to as *Simultaneous Localization and Mapping* (**SLAM**), particularly useful for robot navigation for instance
 - One classic, widely-used system is *ORB-SLAM* (Mur-Artal et al., 2015)
- We will discuss learning-based approaches later

Part 1&2 Summary – Multi-view Geometry

- Camera model and projection
 - Homogeneous coordinates
 - Pinhole camera, perspective projection $\mathbf{x} \cong \mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{X}$
 - Intrinsics \mathbf{K} , extrinsics $[\mathbf{R}|\mathbf{t}]$
 - Camera calibration
- Epipolar geometry
 - Epipolar plane, epipolar lines
 - Essential matrix $\mathbf{E} \Rightarrow \mathbf{x}'^T_{\text{norm}} \mathbf{E} \mathbf{x}_{\text{norm}} = 0$ (calibrated)
 - Fundamental matrix $\mathbf{F} \Rightarrow \mathbf{x}'^T \mathbf{F} \mathbf{x} = 0$ (uncalibrated)
- Structure from Motion (SfM)
 - Bundle adjustment

Part 3 – 3D Representations & Rendering

Classical 3D representations

- Voxels
- Meshes

Forward rendering

- Volumetric rendering
- Surface based rendering
- Shading models
- Raytracing

Photometric reconstruction

- Recovering appearance from images

Inverse rendering

- Recovering geometry from images using differentiable rendering

Differentiable rendering

- Differentiable mesh rendering via blurring

Differentiable rendering

- Volumetric rendering
- Neural fields
- 3D Gaussian Splatting

Part 4 – Learning-based 3D Modeling

Feed-forward 3D reconstruction

- Object reconstruction: Large Reconstruction Models (LRMs), SAM3D
- Scene reconstruction: VGGT

Learning 3D from 2D images

- Novel view synthesis: RayZer
- Distillation: SDS (DreamFusion)

3D applications and open challenges

- Dynamic 3D
- Robotics