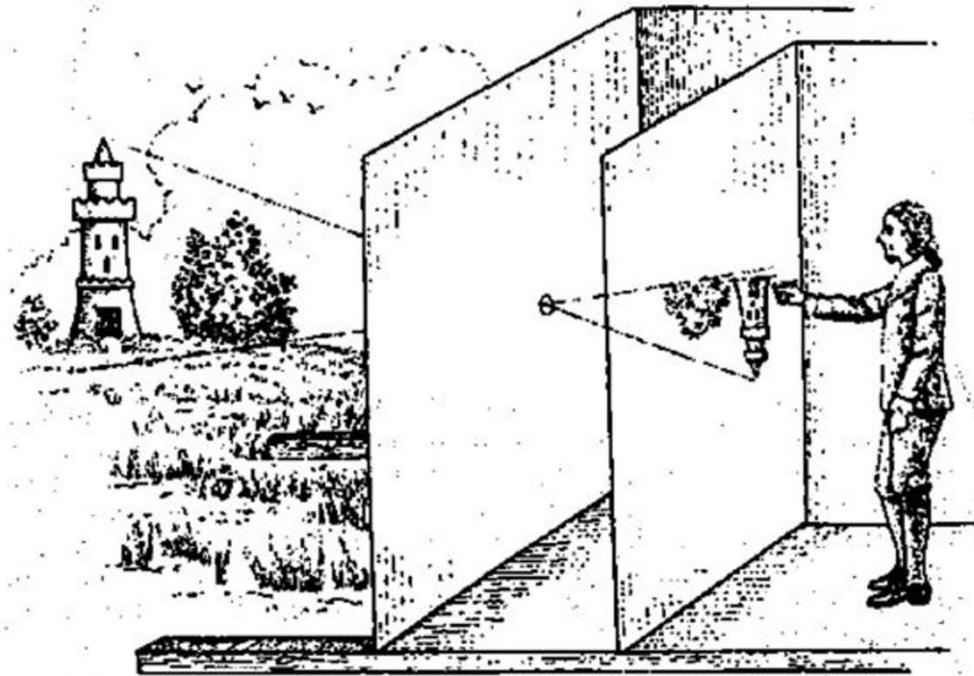


It's a 3D World, After All



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.

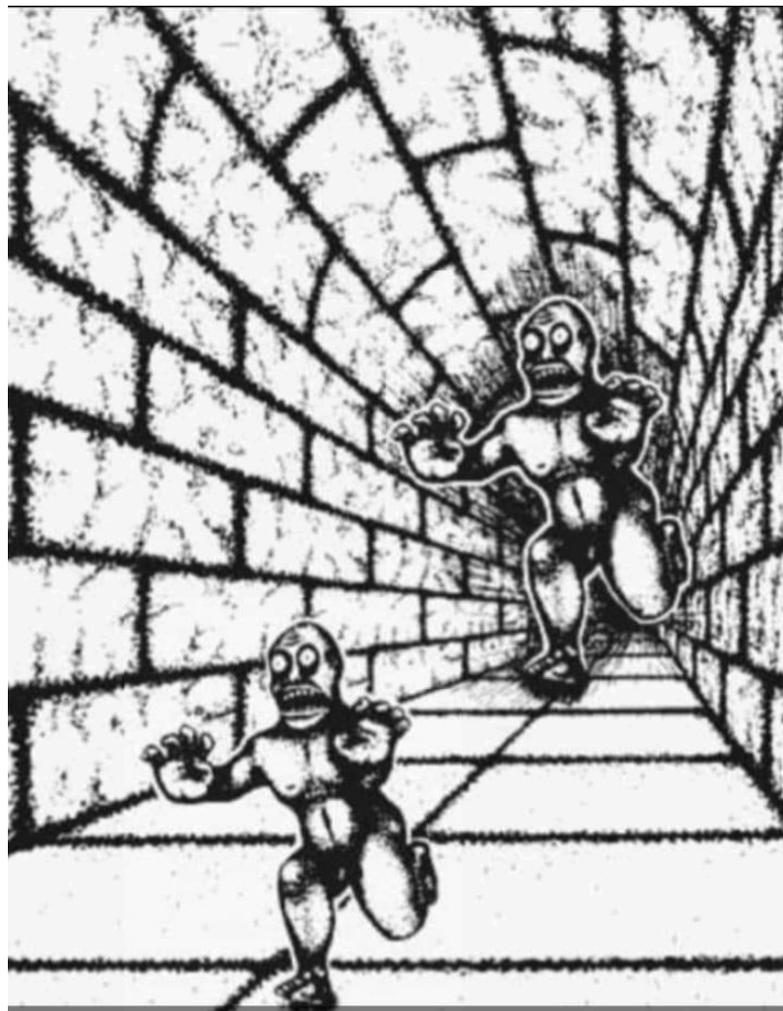
Historically, we have treated vision as a mapping from images to intermediate representations—classes, segmentation masks, or 3D reconstructions. But in the era of the Bitter Lesson, these distinct tasks are becoming qualitatively no different than edge detection: historical artifacts of scoping “solvable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

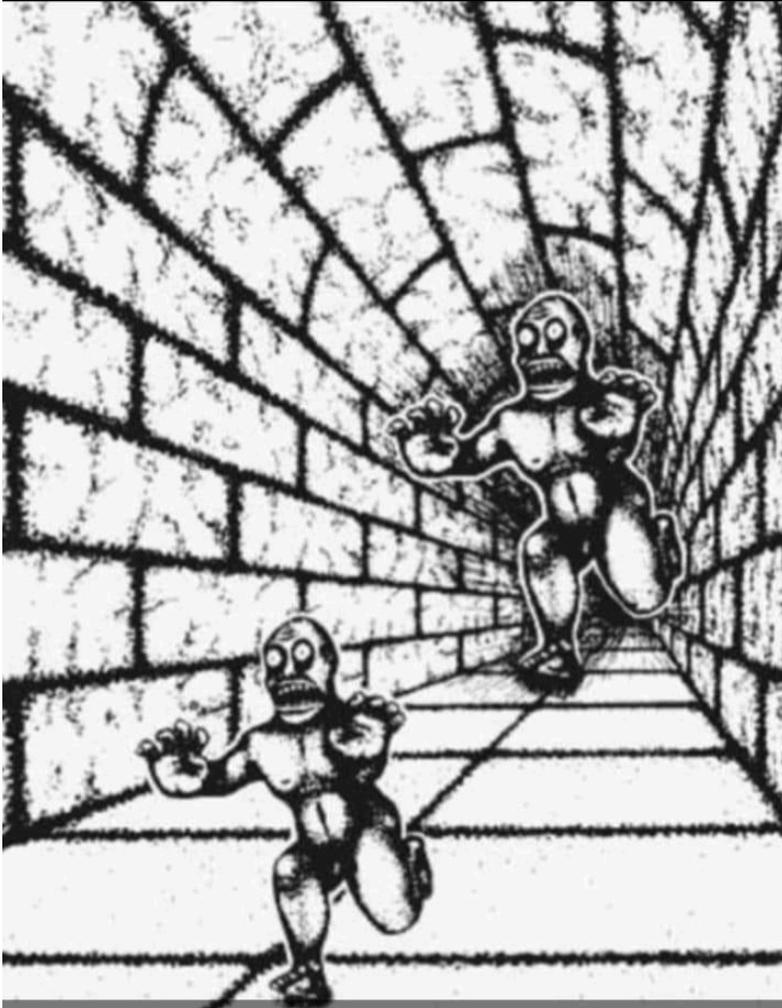
In this post, I argue that the future of computer vision is as part of end-to-end perception-action loops. The historical boundaries between computer vision, robot learning, and control will dissolve. Frontier research will no longer draw a boundary between “seeing” and “learning to act.”

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!

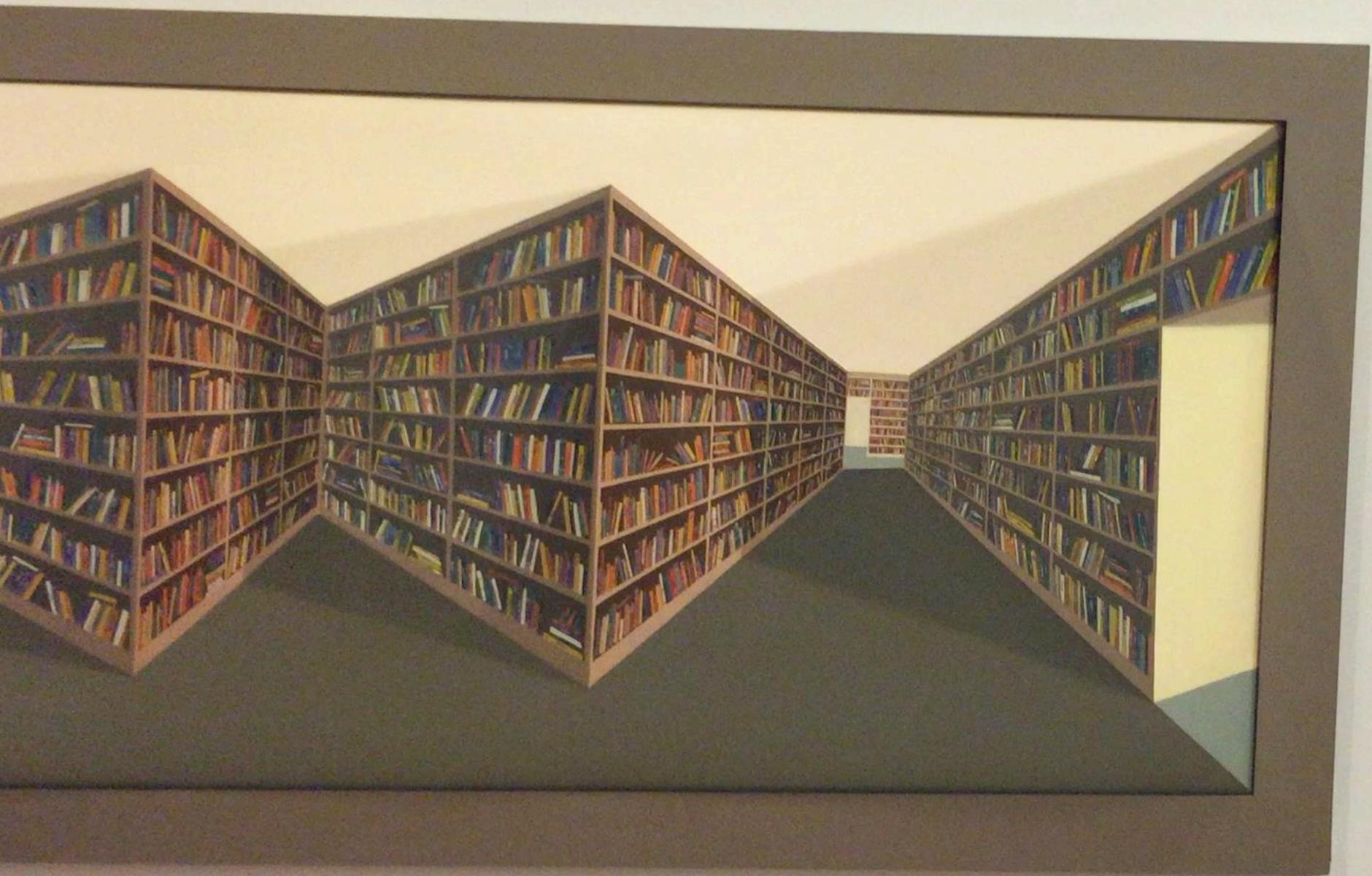


And sometimes too good..



And sometimes too good..





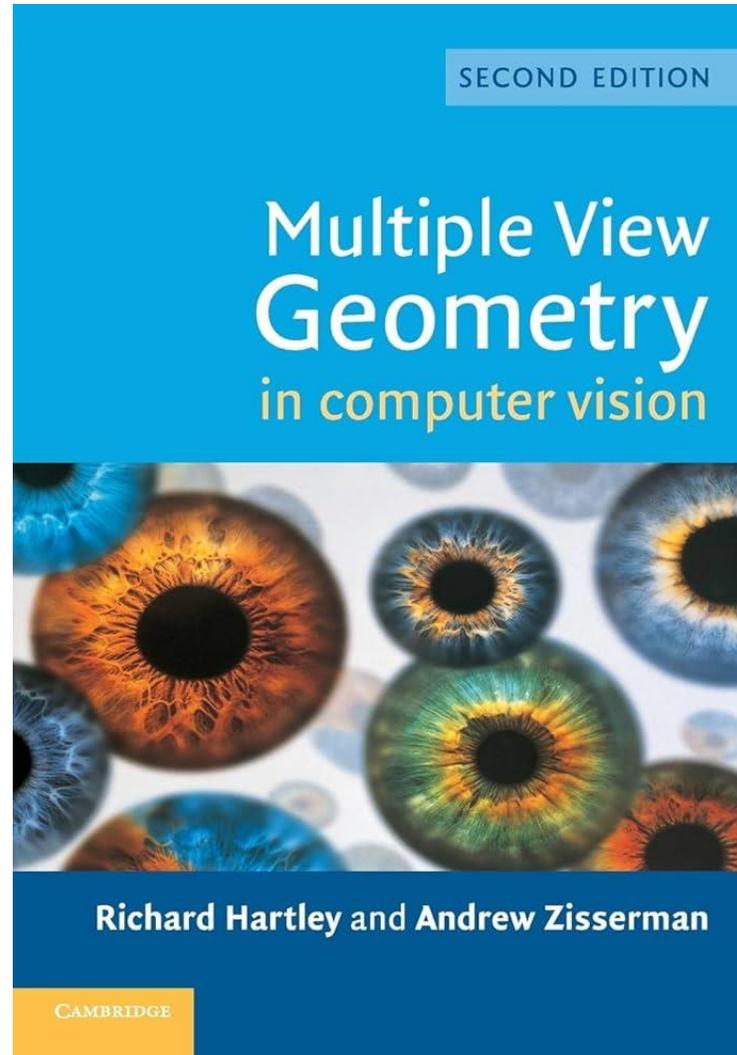


These 'Reverspective' paintings are mind-boggling. <https://youtu.be/fzZy1FtYsVM>

3D Computer Vision – Outline

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

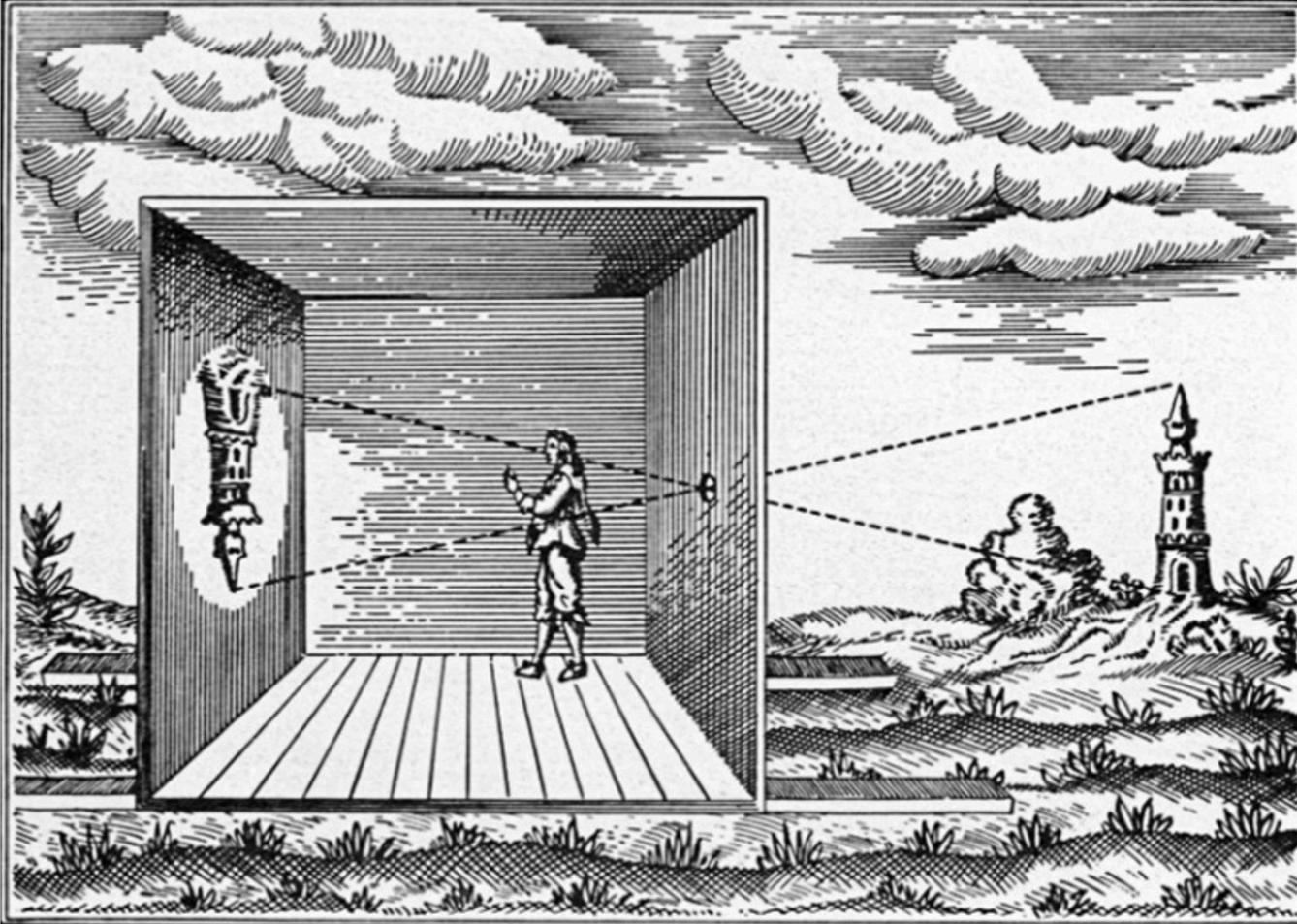
Multi-view Geometry “Bible”



Part 1 – Camera Model & Projection

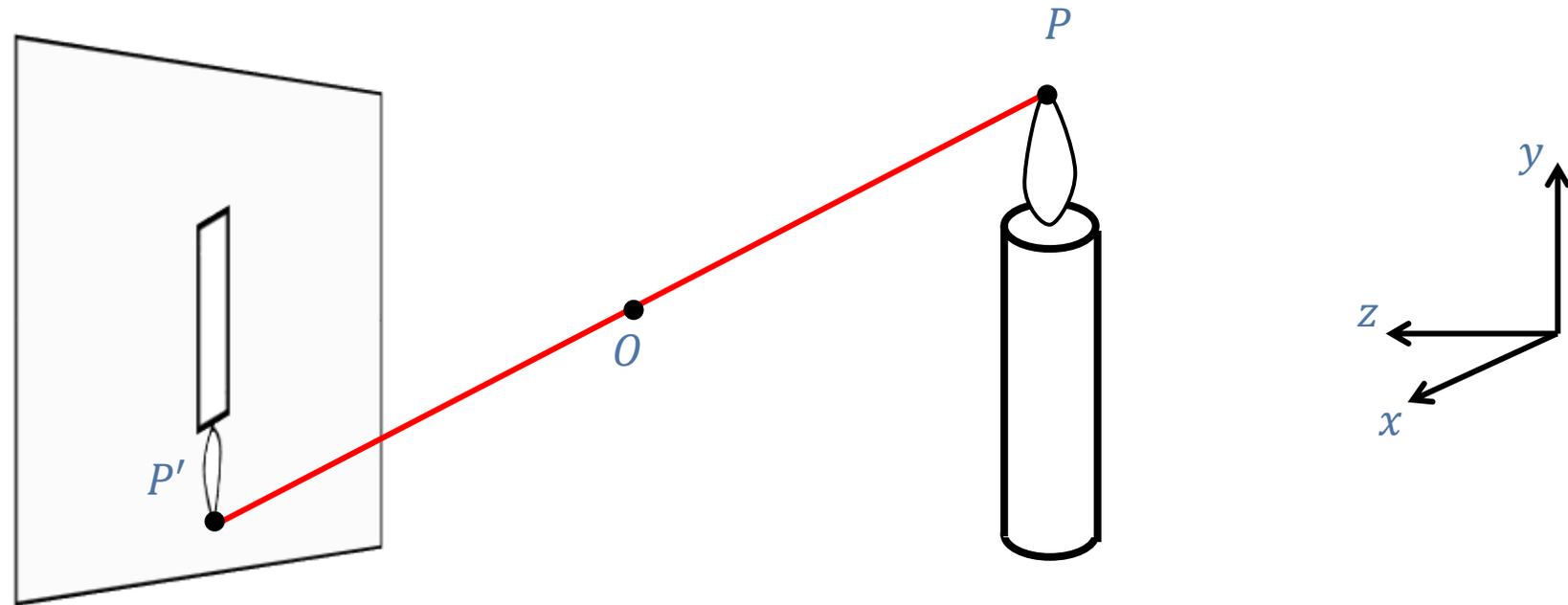
Image Formation – Camera Obscura

“dark chamber” in Latin



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

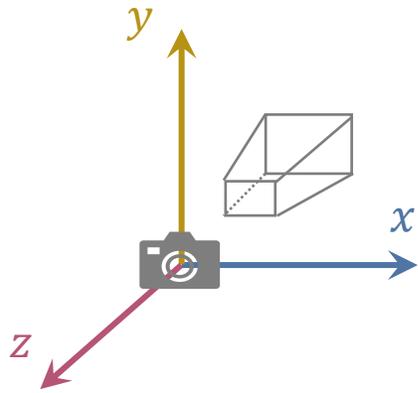
Pinhole Camera



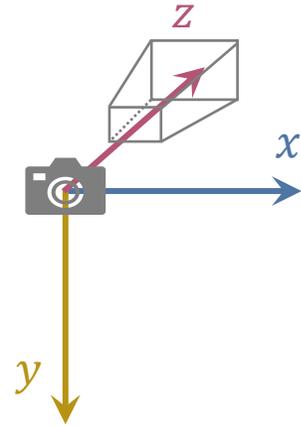
Canonical coordinate system

- 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

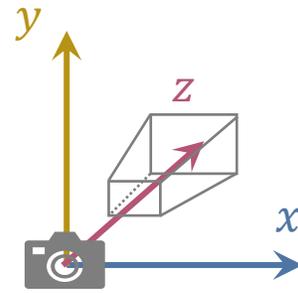
Everybody Agrees... Except on the Axes



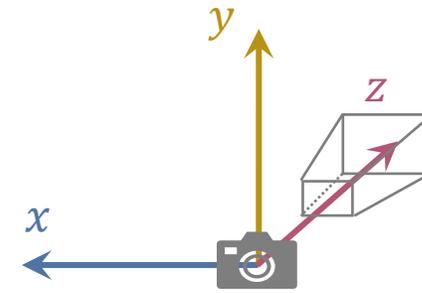
- OpenGL
- Blender
- ARKit
- Three.js
- Nerfstudio
- ...



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

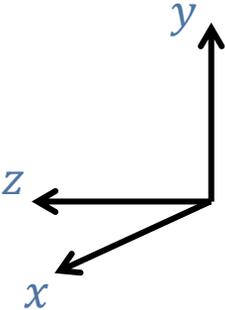
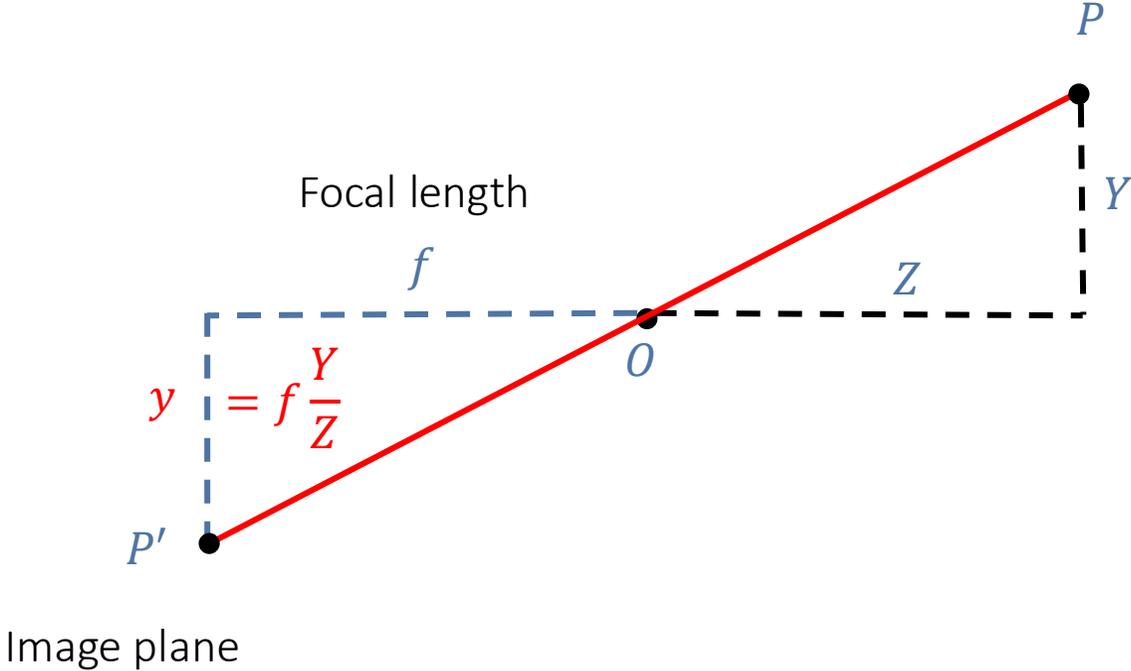


- Unity
- ...



- PyTorch3D
- ?

Perspective Projection



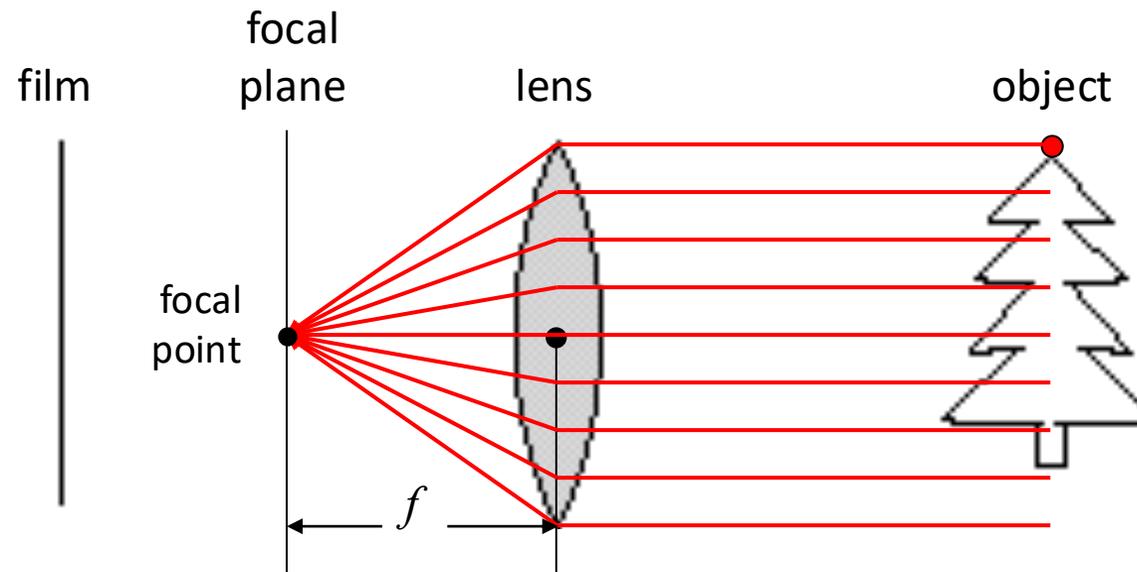
$$\left(f \frac{X}{Z}, f \frac{Y}{Z} \right) \leftarrow (X, Y, Z)$$

2D point

3D point

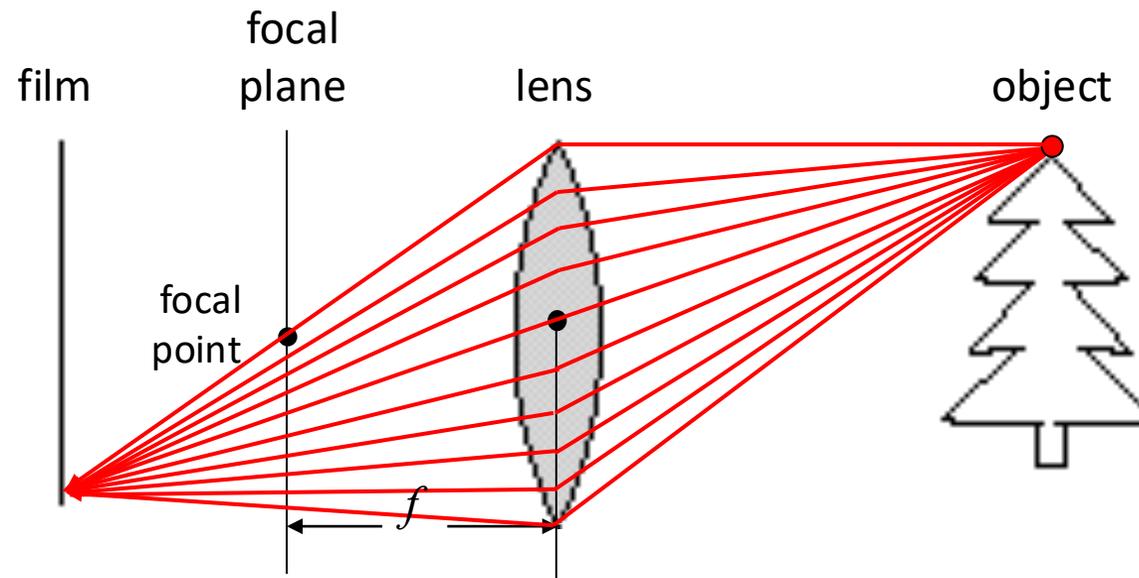
Focal Length of Lenses

- 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.



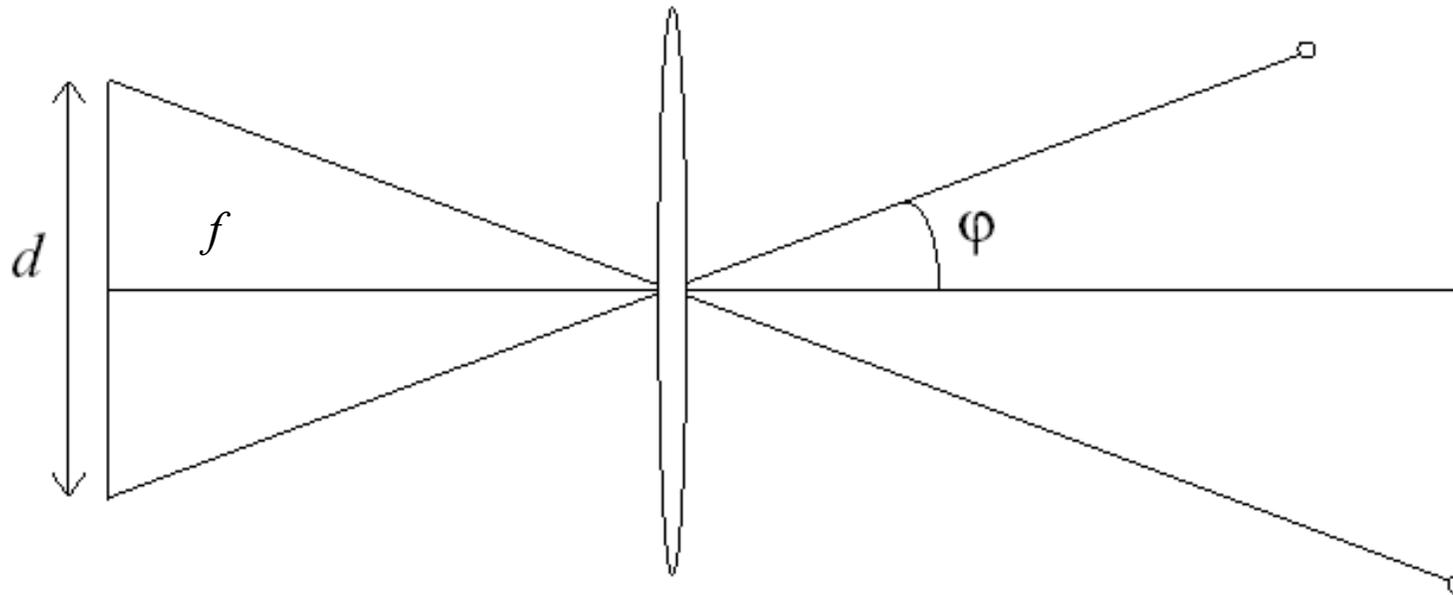
Focal Length of Lenses

- 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)

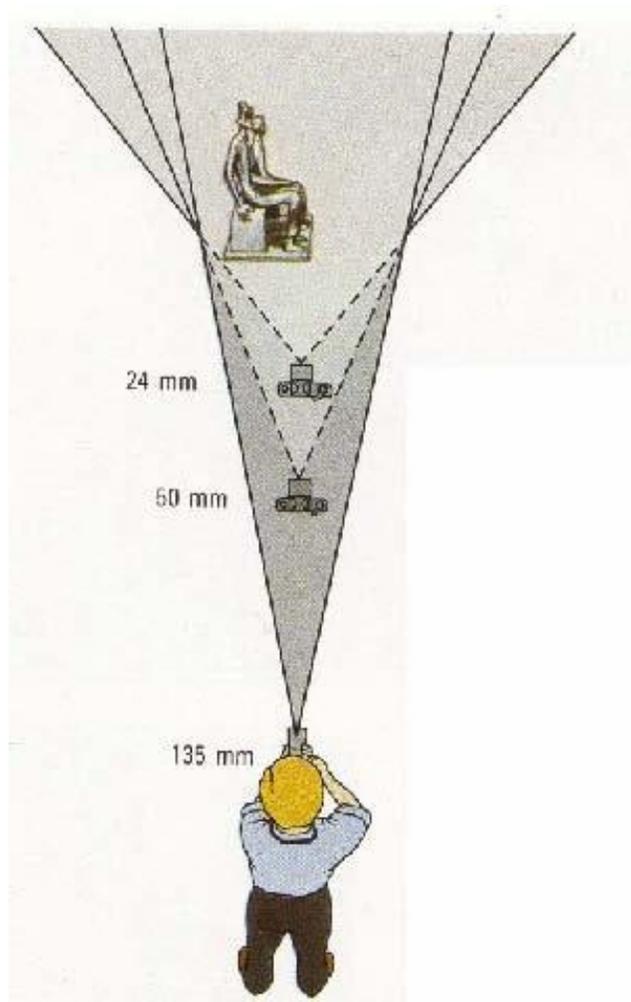
- The field of view is the angular extent of the world observed by the camera.
- Focal length (f), length of the sensor (d):



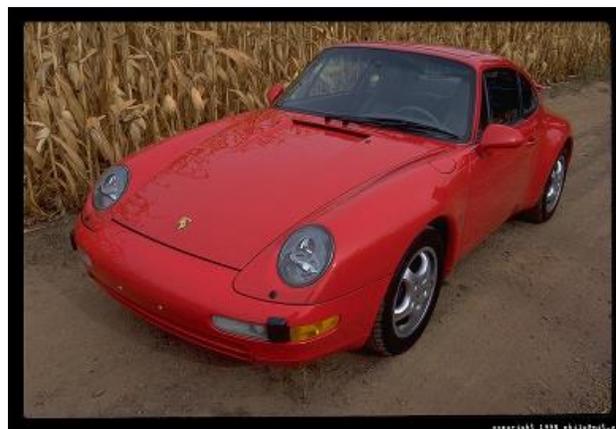
$$\varphi = \tan^{-1} \frac{d}{2f}$$

- 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



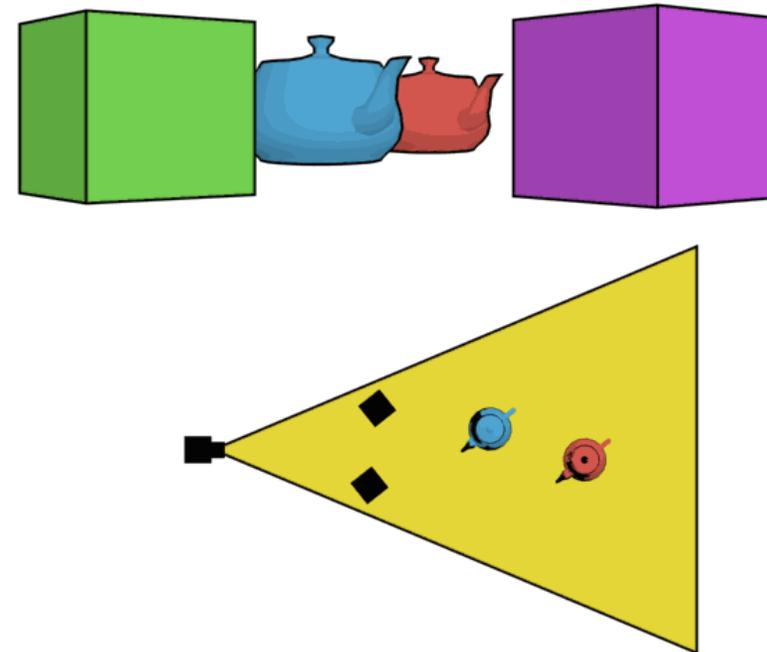
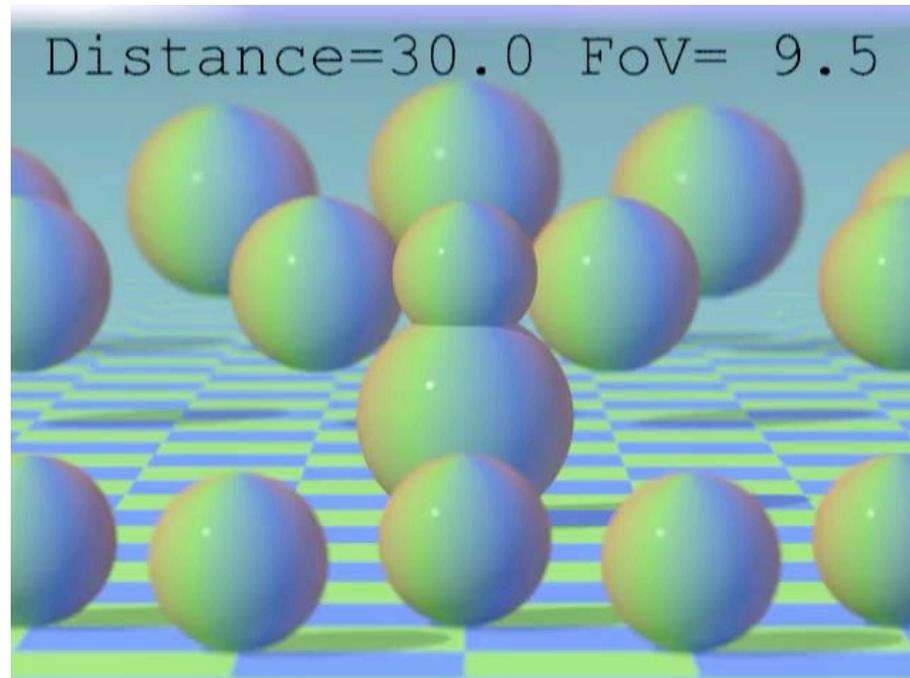
telephoto

(reconstructed by Hunyuan3D V3.1)

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

Dolly Zoom Effect

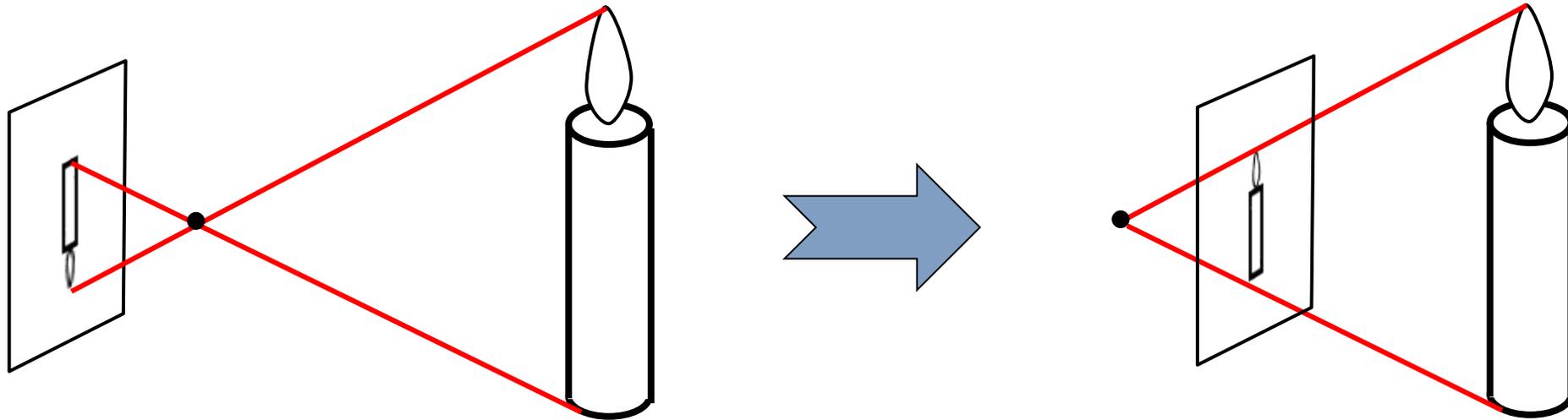
- Continuously adjusting the focal length, while dollying toward or away from the subject.



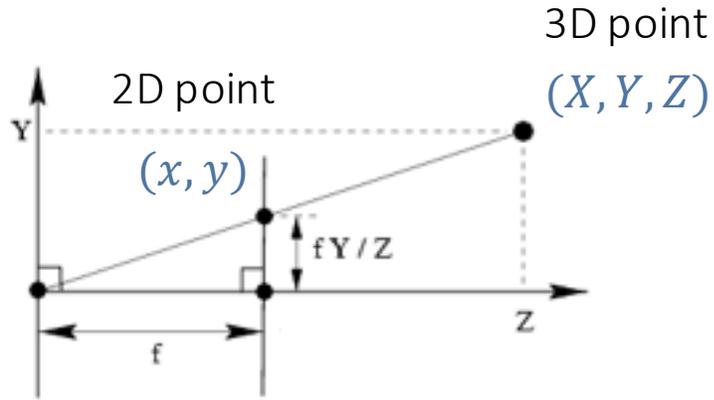
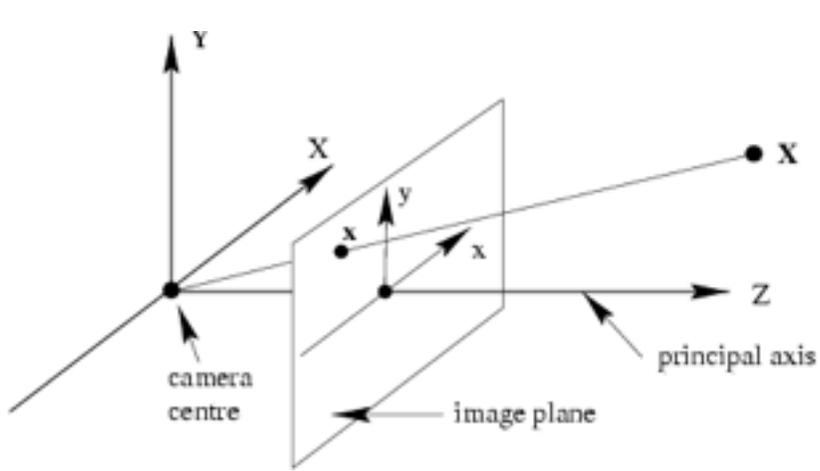
https://en.wikipedia.org/wiki/Dolly_zoom

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} $\mathbf{l} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$ $\mathbf{l}^\top \mathbf{x} = ax + by + c = 0$	3D plane $\boldsymbol{\pi}$ $\boldsymbol{\pi} = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$ $\boldsymbol{\pi}^\top \mathbf{X} = aX + bY + cZ + d = 0$
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:

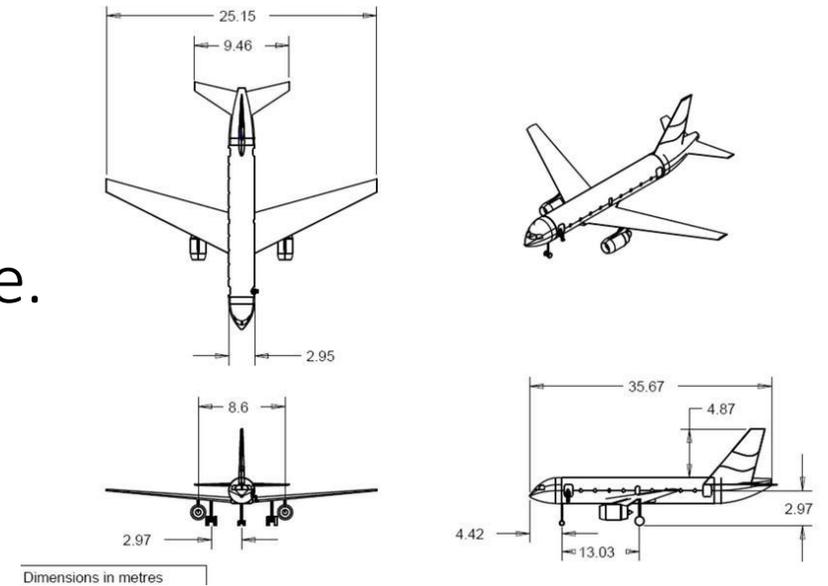
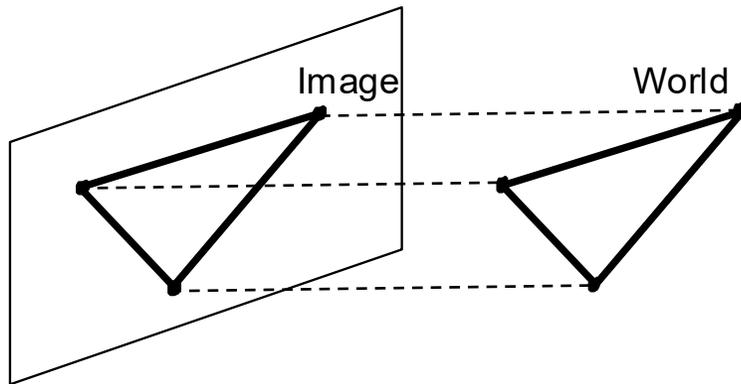
$$\begin{bmatrix} \\ \\ \\ \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \\ \\ \\ \end{bmatrix} \Rightarrow \left(f \frac{X}{Z}, f \frac{Y}{Z} \right)$$

divide by the third coordinate

- What will happen if f is very large?
 - Weak perspective projection $x \approx f \frac{X}{Z_0}$, where Z_0 is a constant global depth

Orthographic Projection

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



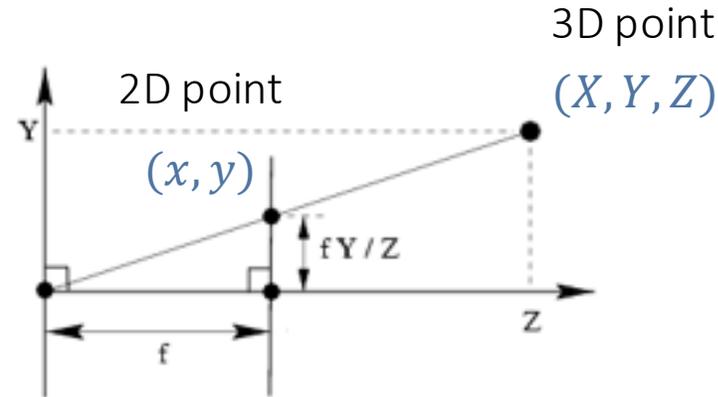
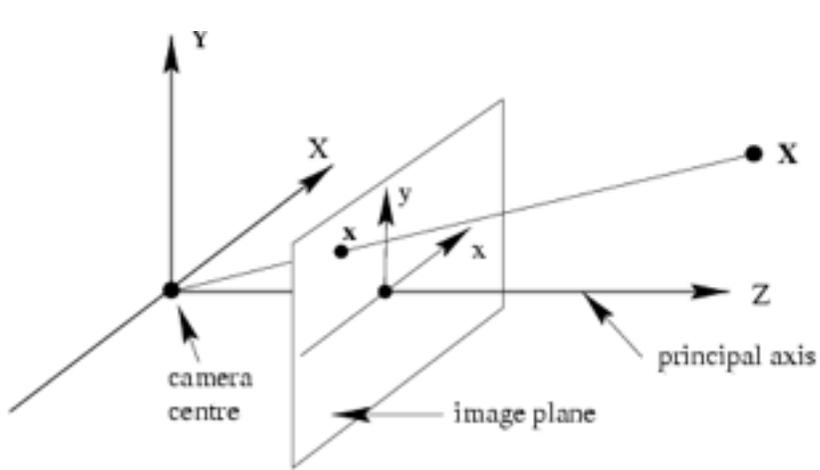
Three-view drawing

- 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)



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

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

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

Homogeneous coordinates of image point

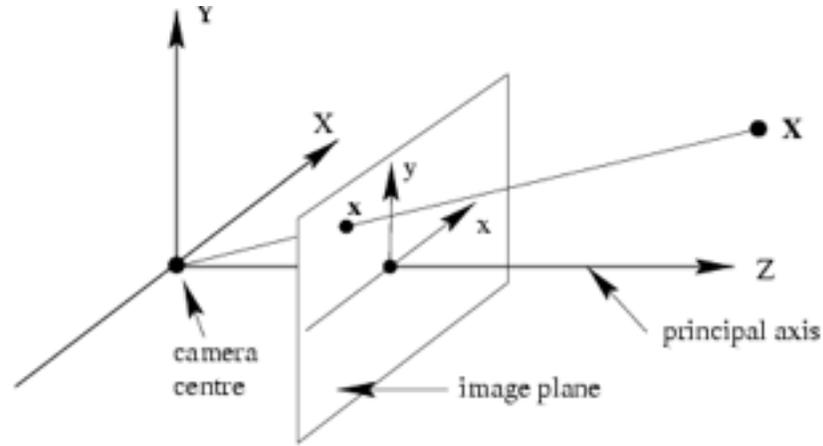
Camera projection matrix

Homogeneous coordinates of 3D point

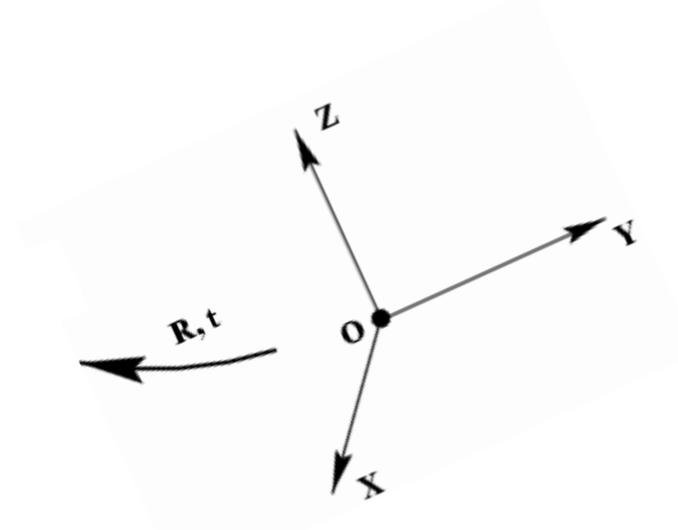
$$\mathbf{x} \cong \mathbf{P}\mathbf{X}$$

↑
Equality up to scale

Camera Calibration



camera coordinate system



world coordinate system

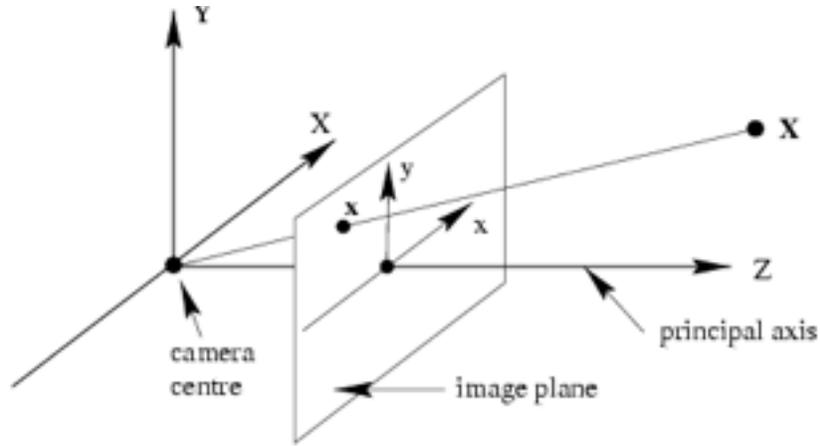
- **Camera calibration:** figuring out transformation from *world* coordinate system to *image* coordinate system

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

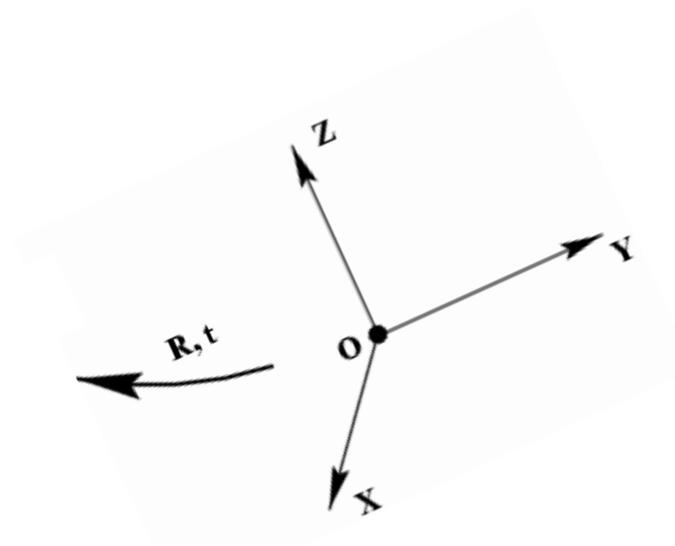
Intrinsic camera parameters: focal length, principal point, scaling factors

Extrinsic camera parameters: rotation, translation

Camera Calibration



camera coordinate system



world coordinate system

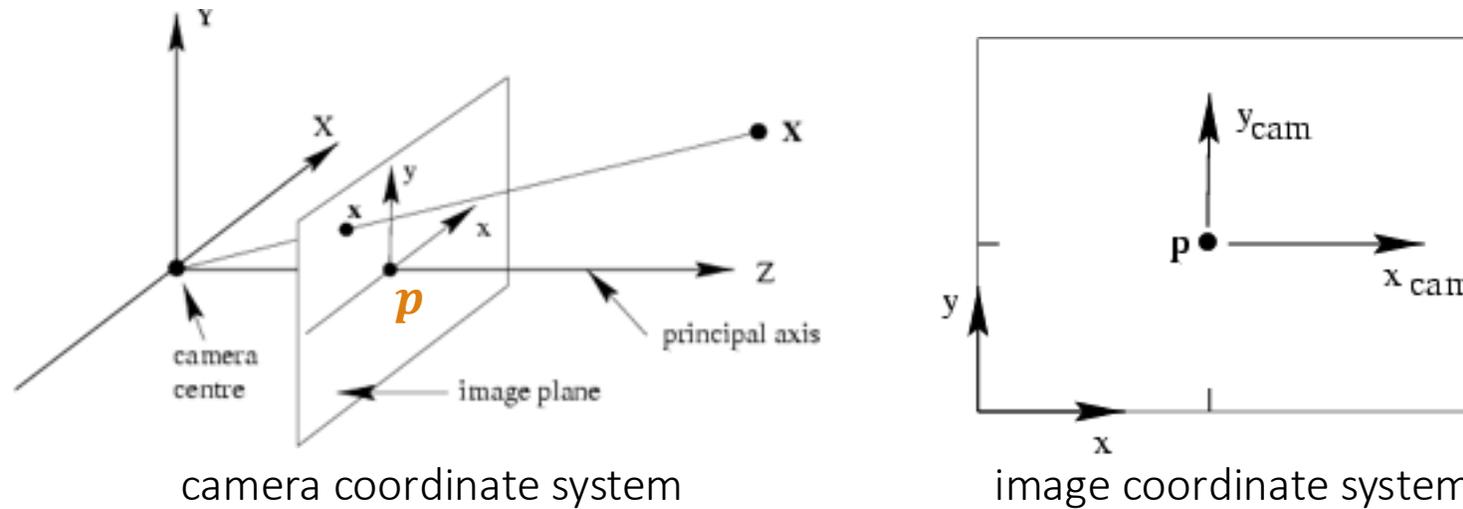
- **Camera calibration:** figuring out transformation from *world* coordinate system to *image* coordinate system

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

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

General camera projection matrix

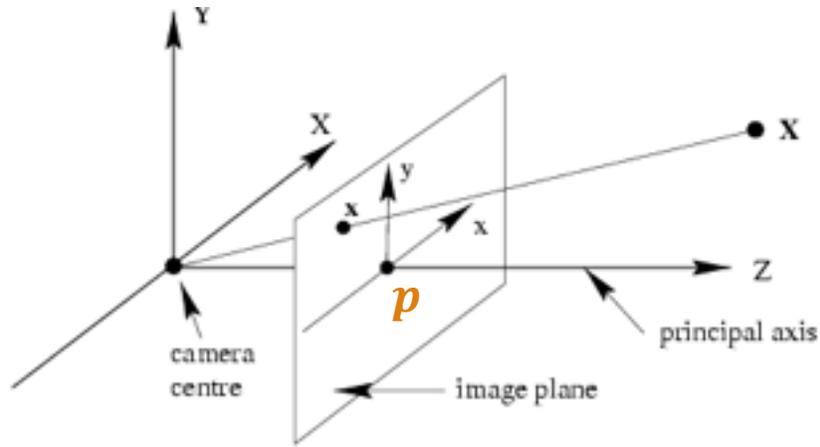
Intrinsic Parameters – Principal Point



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



camera coordinate system

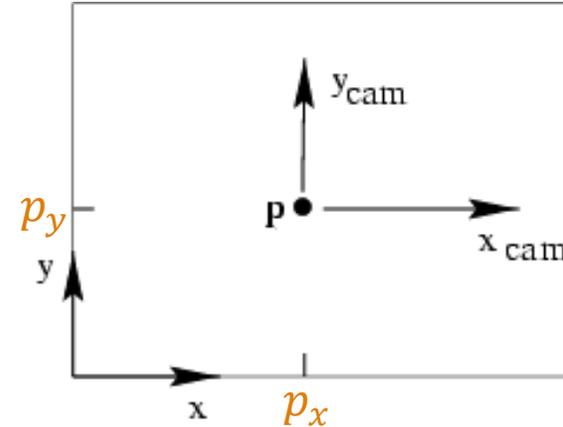


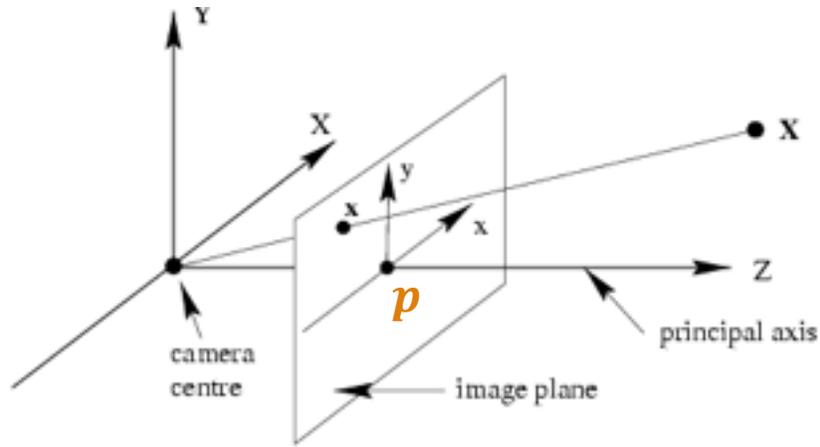
image coordinate system

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

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

calibration matrix \mathbf{K}

Intrinsic Parameters – Scaling



camera coordinate system

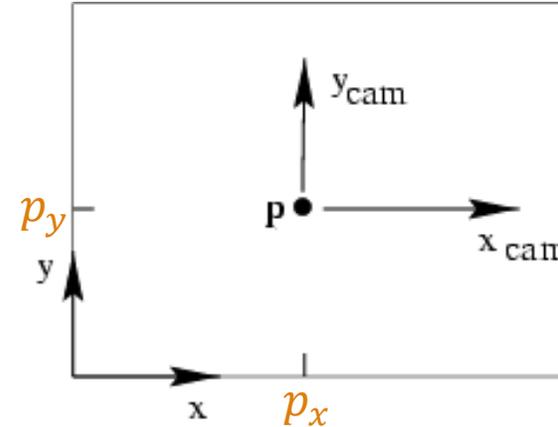


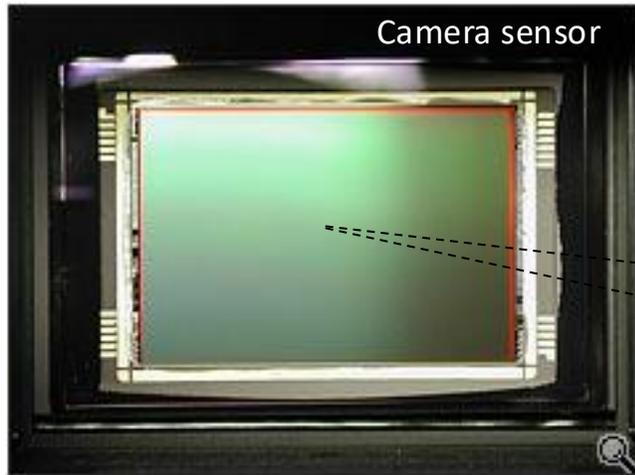
image coordinate system

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

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

pixels m calibration matrix \mathbf{K}

Intrinsic Parameters – Scaling



Camera sensor

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
 K in metric units

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

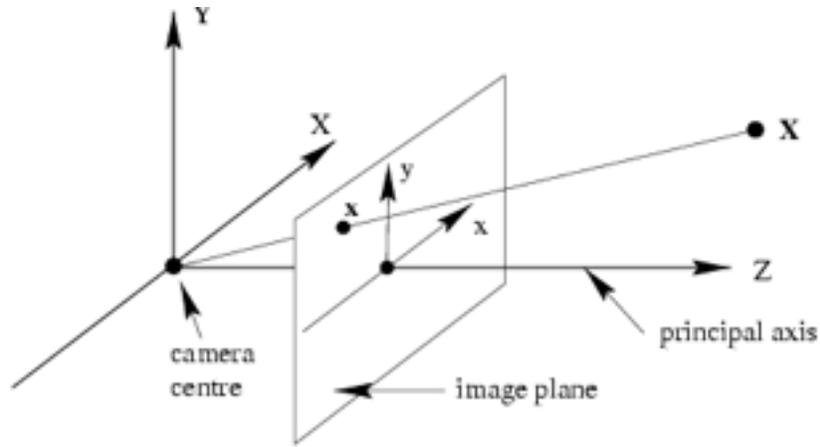
m

Calibration matrix
 K in pixel units

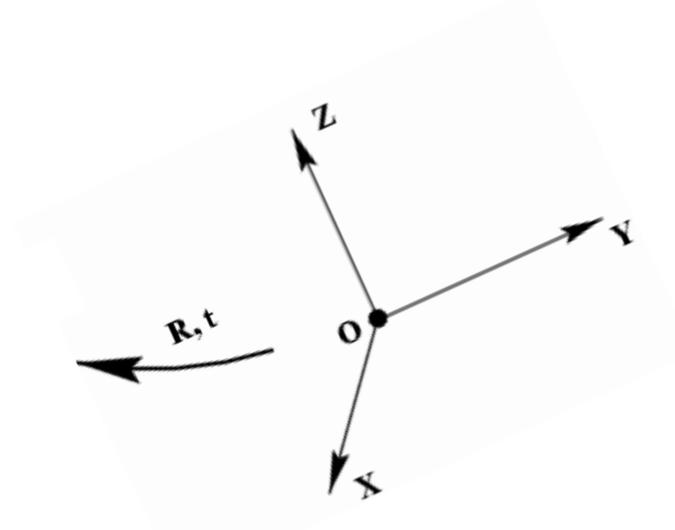
$$= \begin{bmatrix} \alpha_x & 0 & \beta_x \\ 0 & \alpha_y & \beta_y \\ 0 & 0 & 1 \end{bmatrix}$$

pixels

Camera Calibration



camera coordinate system



world coordinate system

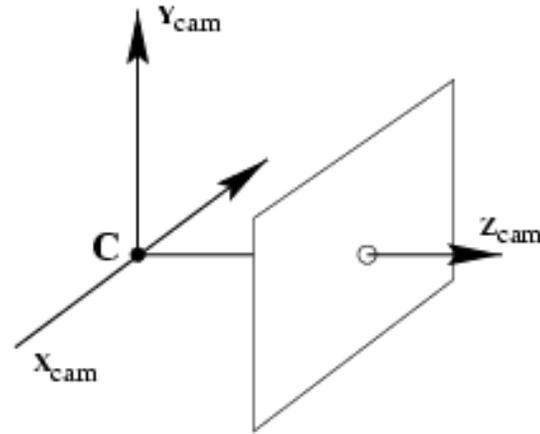
- **Camera calibration:** figuring out transformation from *world* coordinate system to *image* coordinate system

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

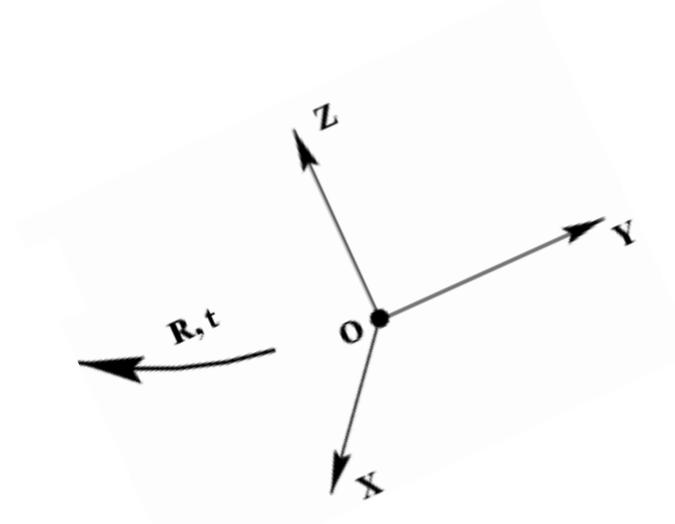
$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$$

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:

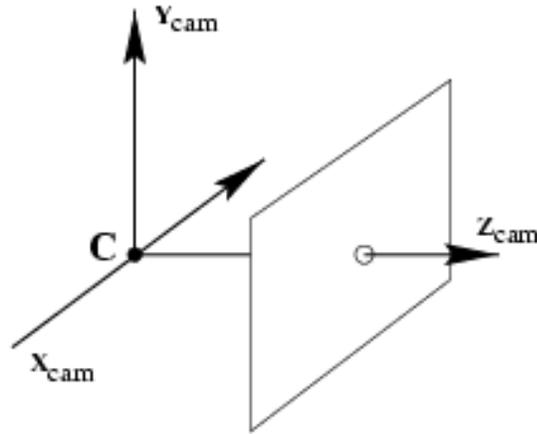
$$\tilde{X}_{cam} = R(\tilde{X} - \tilde{C}) = R\tilde{X} + t$$

coords. of point in normalized camera frame → \tilde{X}_{cam}
 3x3 rotation matrix → R
 coords. of a point in world frame → \tilde{X}
 coords. of camera center in world frame → \tilde{C}

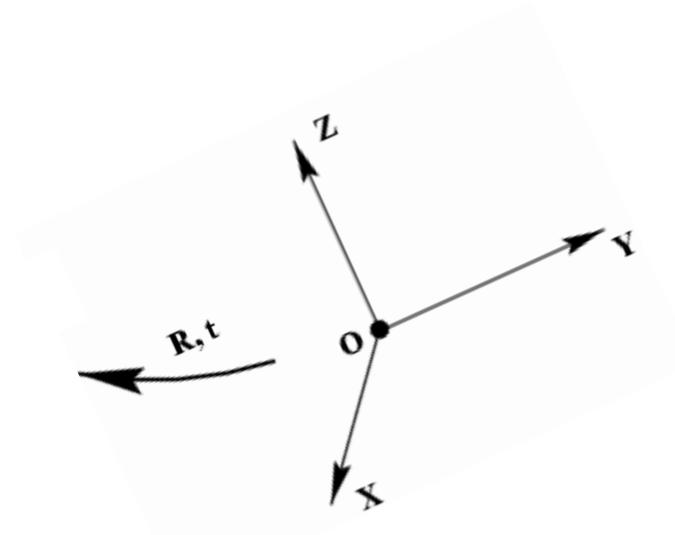
W2C: $\tilde{X}_{cam} = R\tilde{X}_{world} + t$

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

Extrinsic Parameters



camera coordinate system



world coordinate system

In non-homogeneous
coordinates:

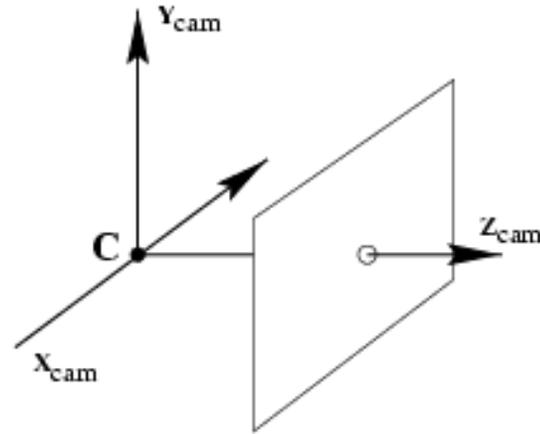
$$\tilde{\mathbf{X}}_{\text{cam}} = \mathbf{R}\tilde{\mathbf{X}} + \mathbf{t}$$

In homogeneous
coordinates:

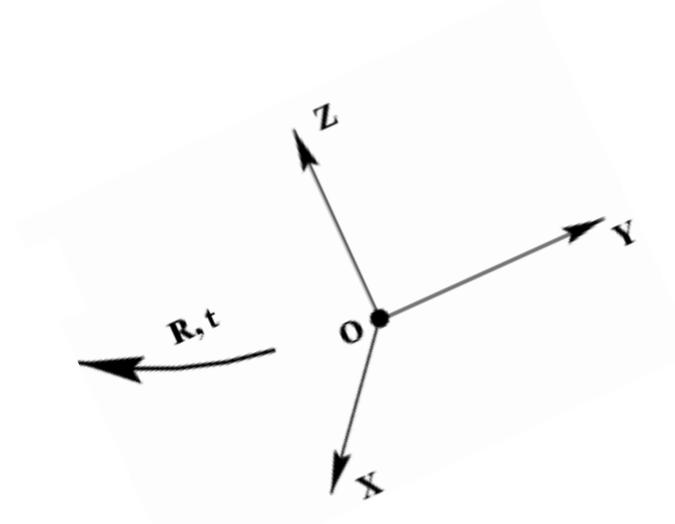
$$\begin{pmatrix} \tilde{\mathbf{X}}_{\text{cam}} \\ 1 \end{pmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{pmatrix} \tilde{\mathbf{X}} \\ 1 \end{pmatrix}$$

3D transformation
matrix (4 x 4)

Extrinsic Parameters



camera coordinate system



world coordinate system

In non-homogeneous coordinates:

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

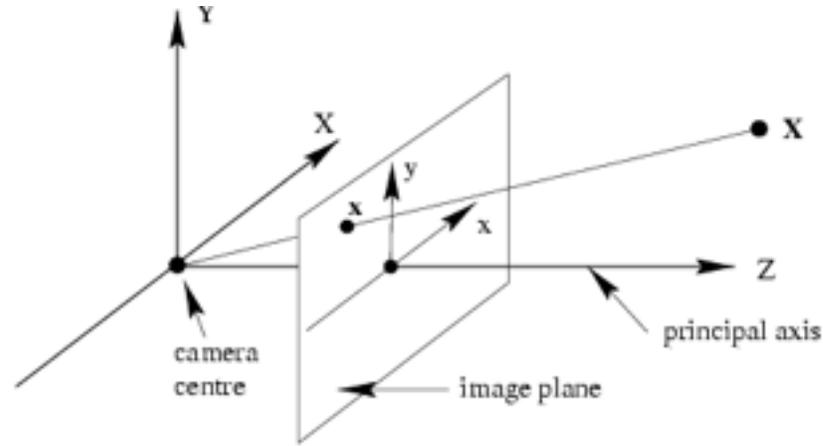
In homogeneous coordinates:

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

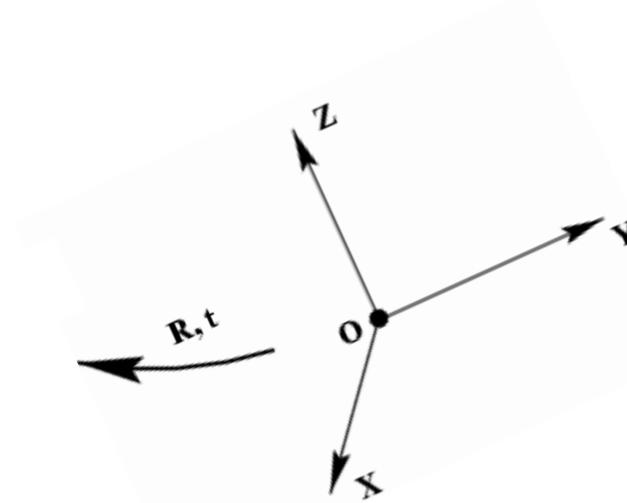
3D transformation matrix (4 x 4)

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

Camera Calibration



camera coordinate system



world coordinate system

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

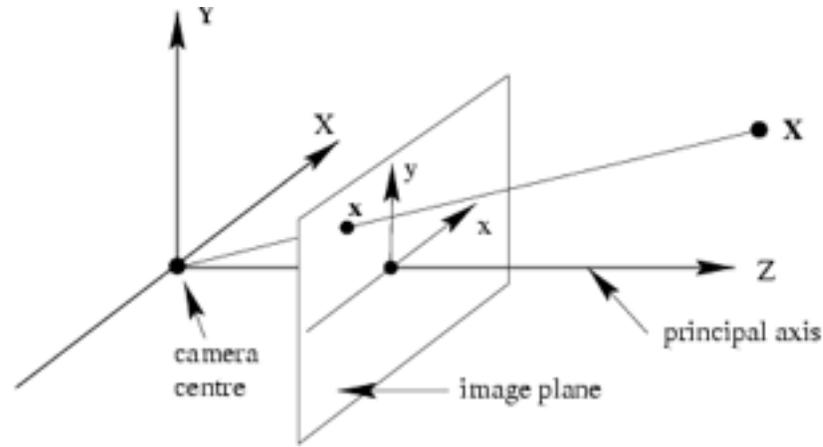
- **Camera calibration:** figuring out transformation from *world* coordinate system to *image* coordinate system

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

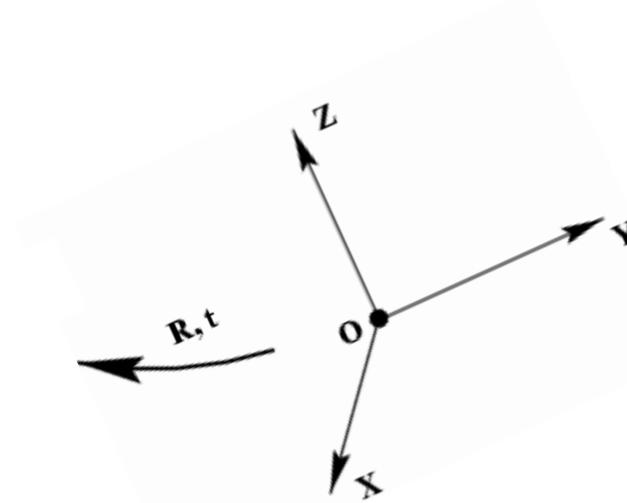
$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$$

General camera projection matrix

Camera Calibration



camera coordinate system



world coordinate system

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

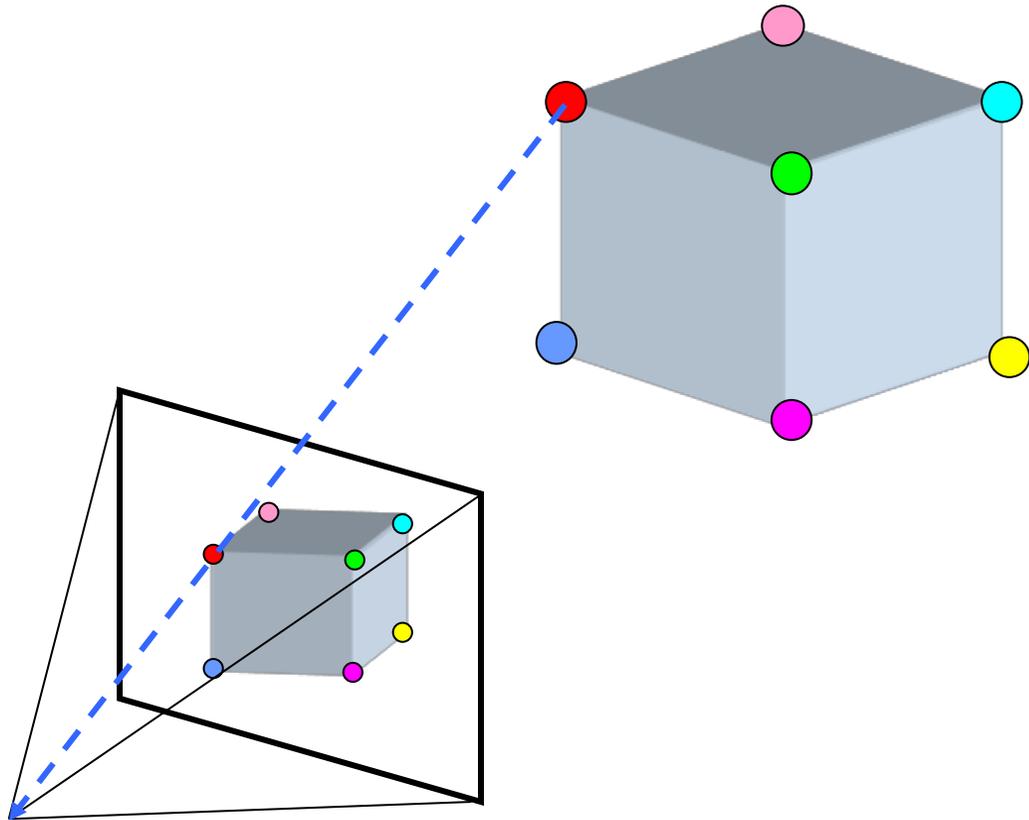
- **Camera calibration:** figuring out transformation from *world* coordinate system to *image* coordinate system

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

← 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}).

Student Feedback Survey

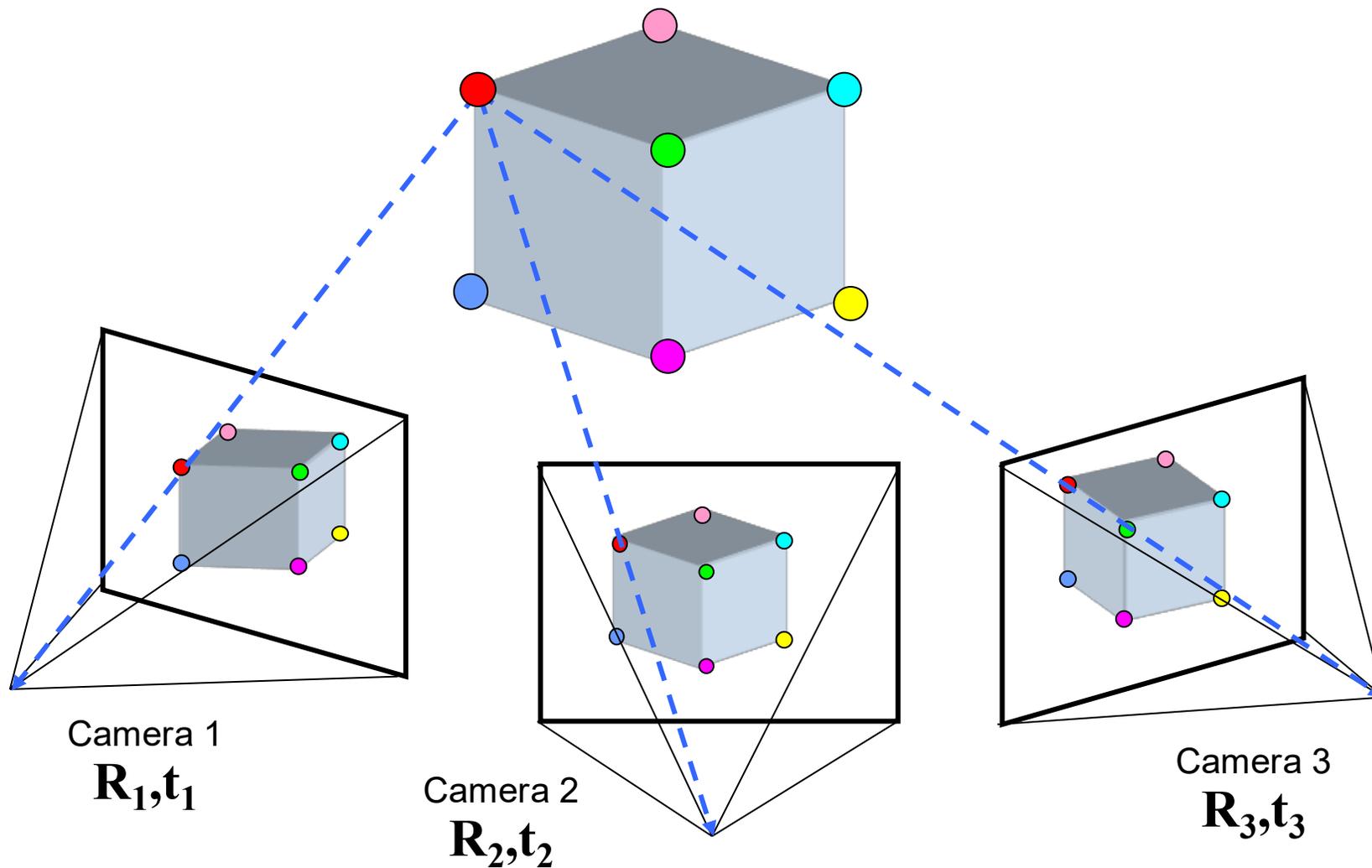


3D Computer Vision – Outline

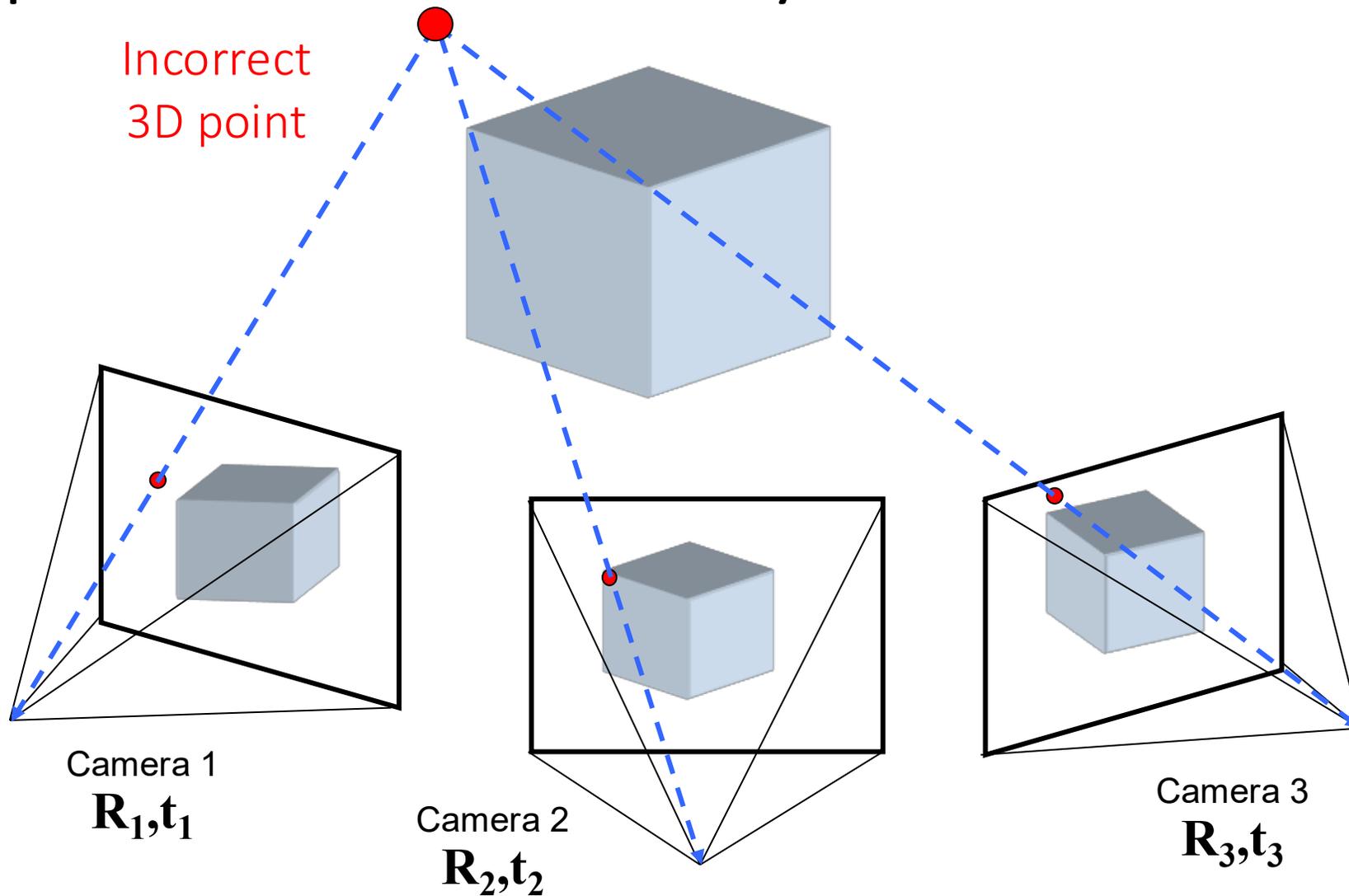
- Part 1 – Camera Model & Projection
- Part 2 – Multi-view Geometry
- Part 3 – Learning-based 3D Reconstruction
- Part 4 – 3D Representations & Rendering

Part 2 – Multi-view Geometry

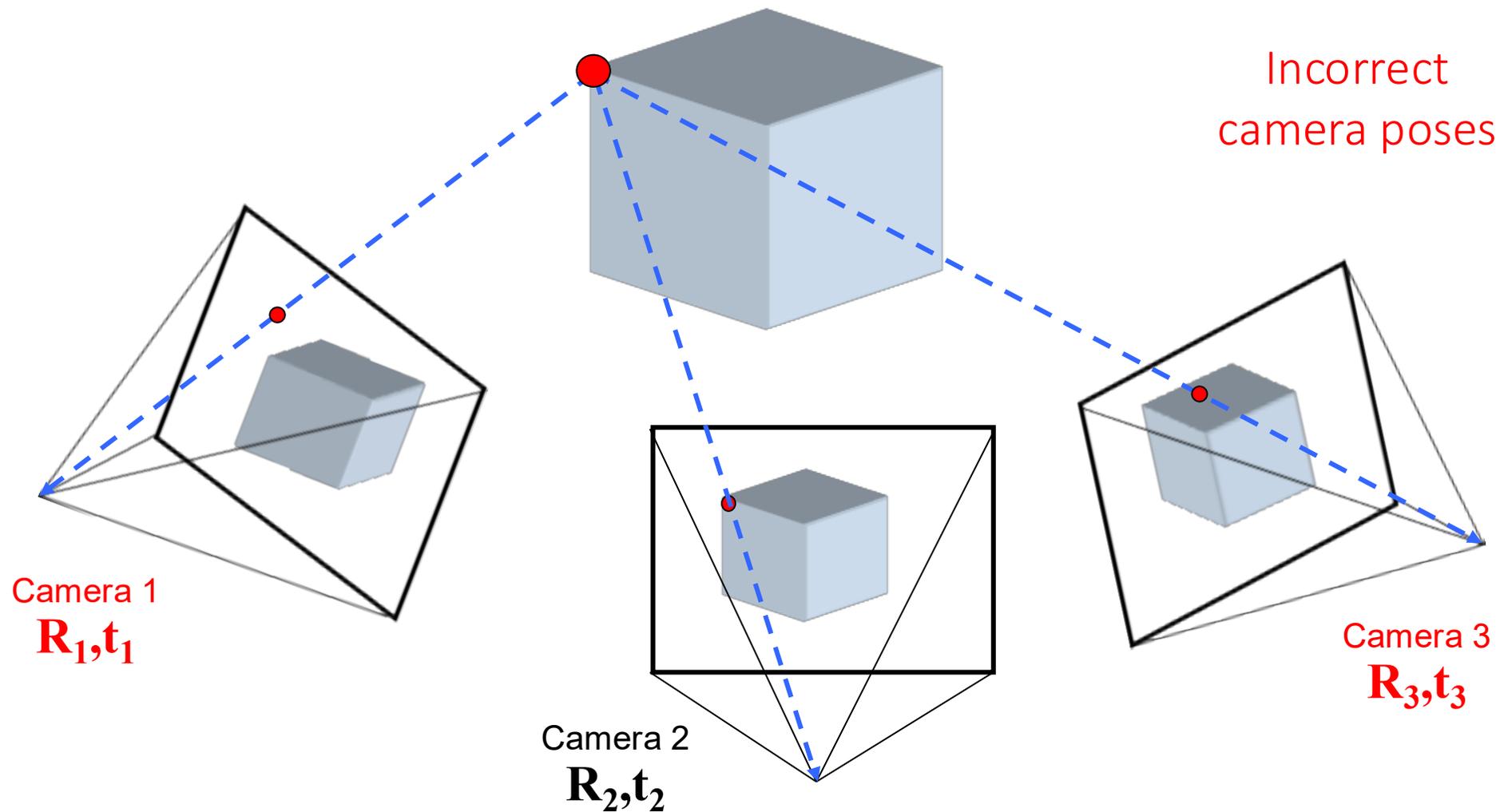
Multiple View Geometry



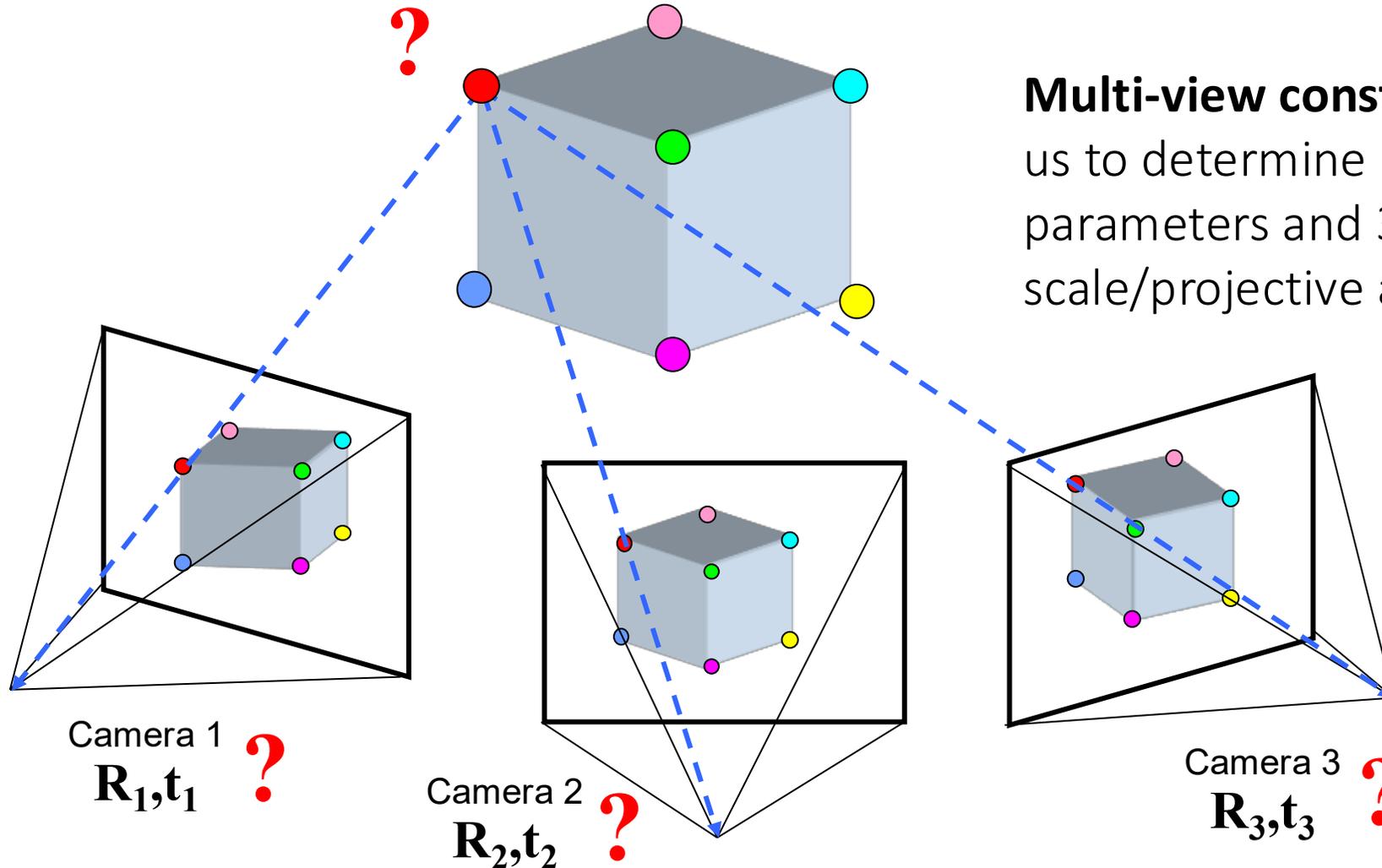
Multiple View Geometry



Multiple View Geometry

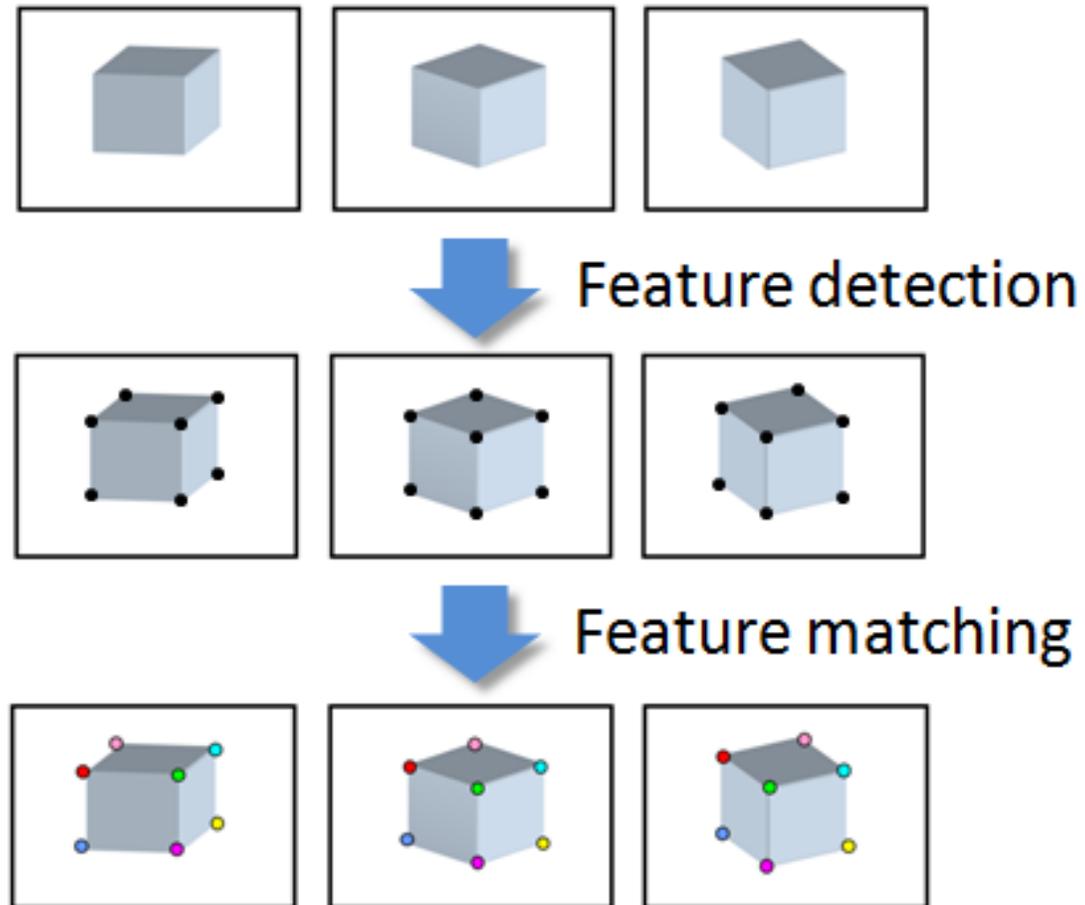


Multiple View Geometry

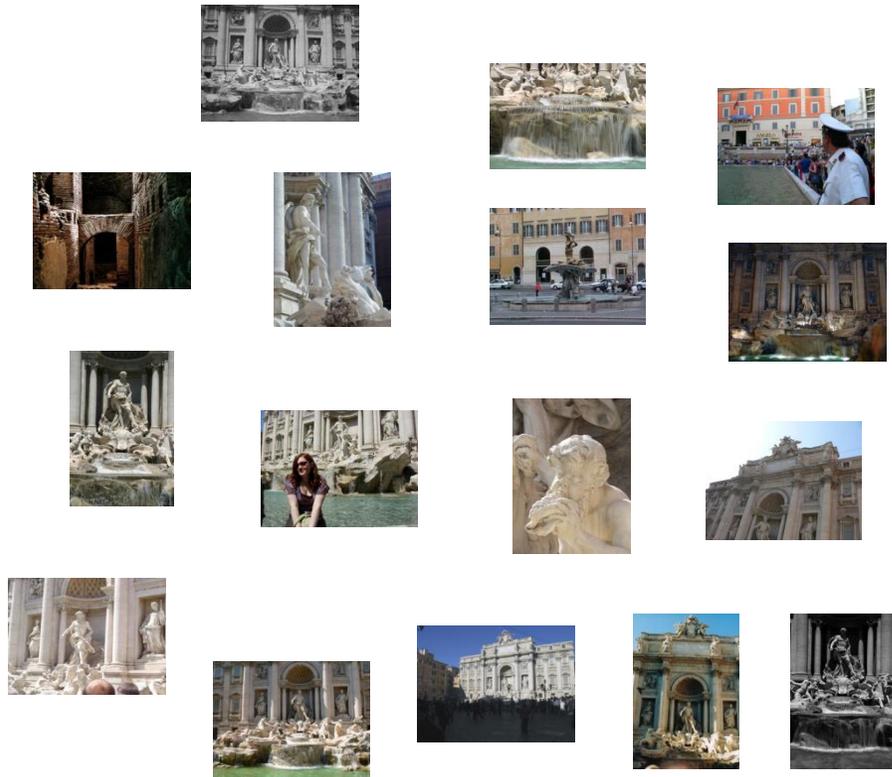


Multi-view constraints allow us to determine both camera parameters and 3D (up to scale/projective ambiguities)

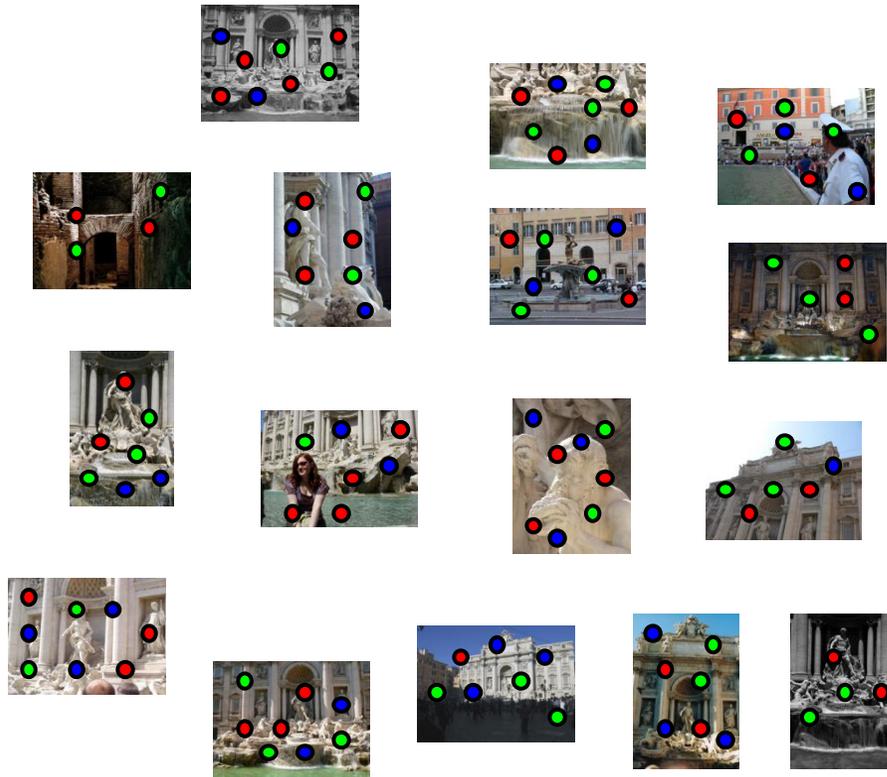
Estimating Correspondences



Estimating Correspondences



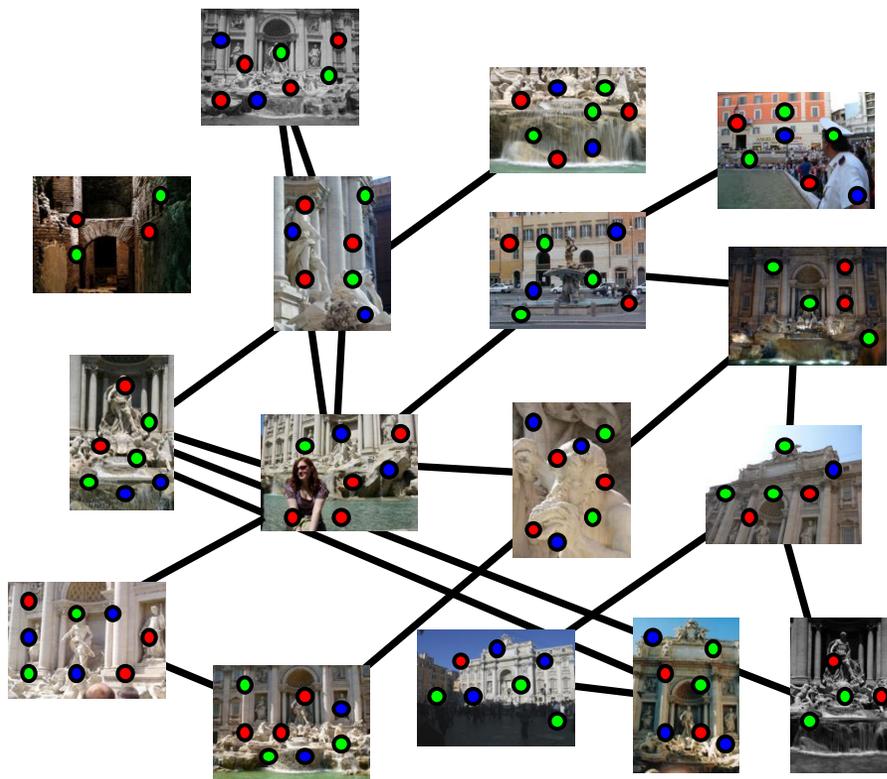
Estimating Correspondences



- **Detect feature points**

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

Estimating Correspondences



- **Detect feature points**

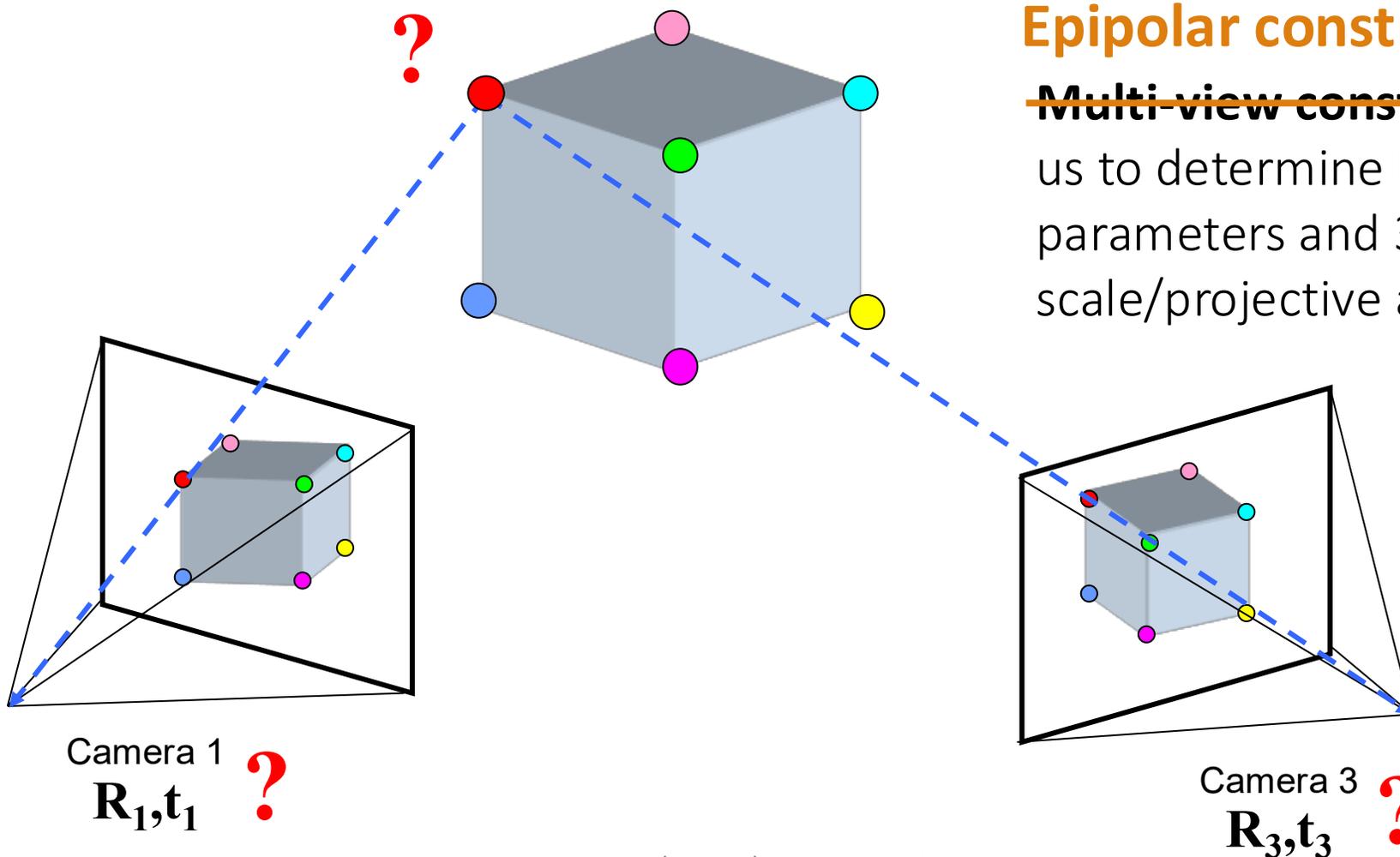
- SIFT descriptors [Lowe, 2004]
- Learned features – SuperPoint [DeTone, 2018])

- **Match features** between pair of images

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

Let's Start with Two Views – Epipolar Geometry

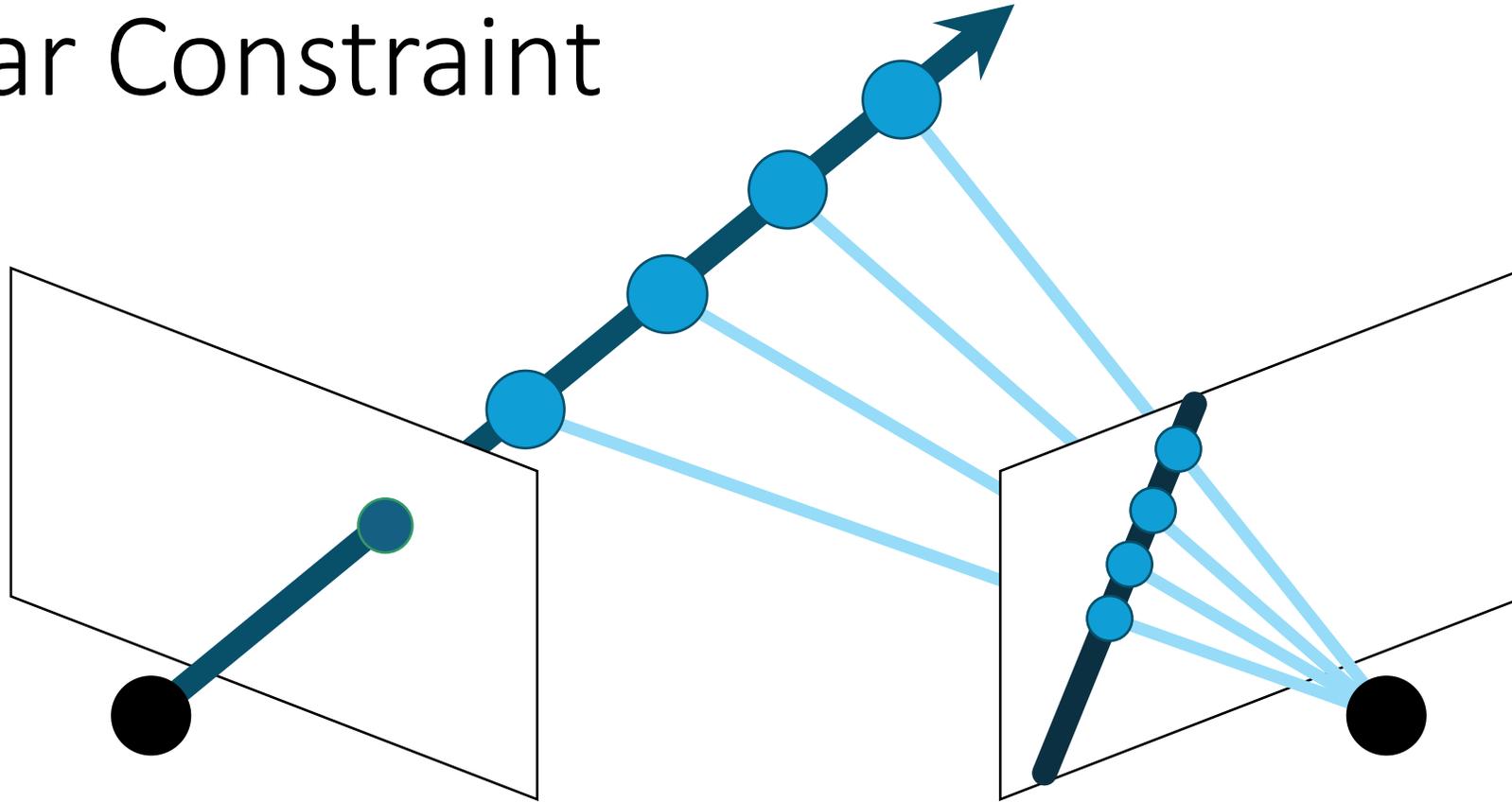
“epi-pole” \approx upon the pole/pivot



Epipolar constraints

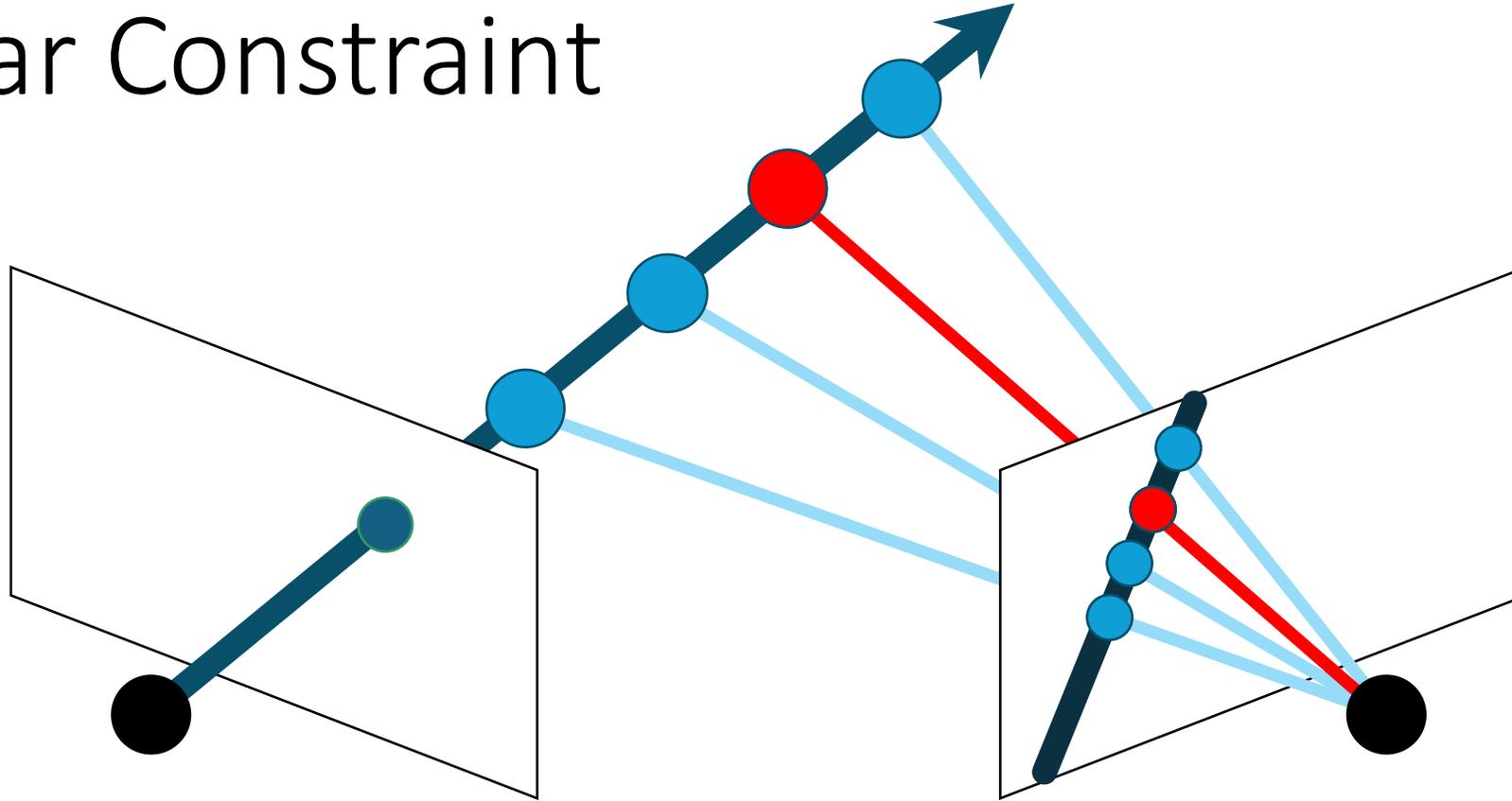
~~Multi-view constraints~~ allow us to determine both camera parameters and 3D (up to scale/projective ambiguities)

Epipolar Constraint



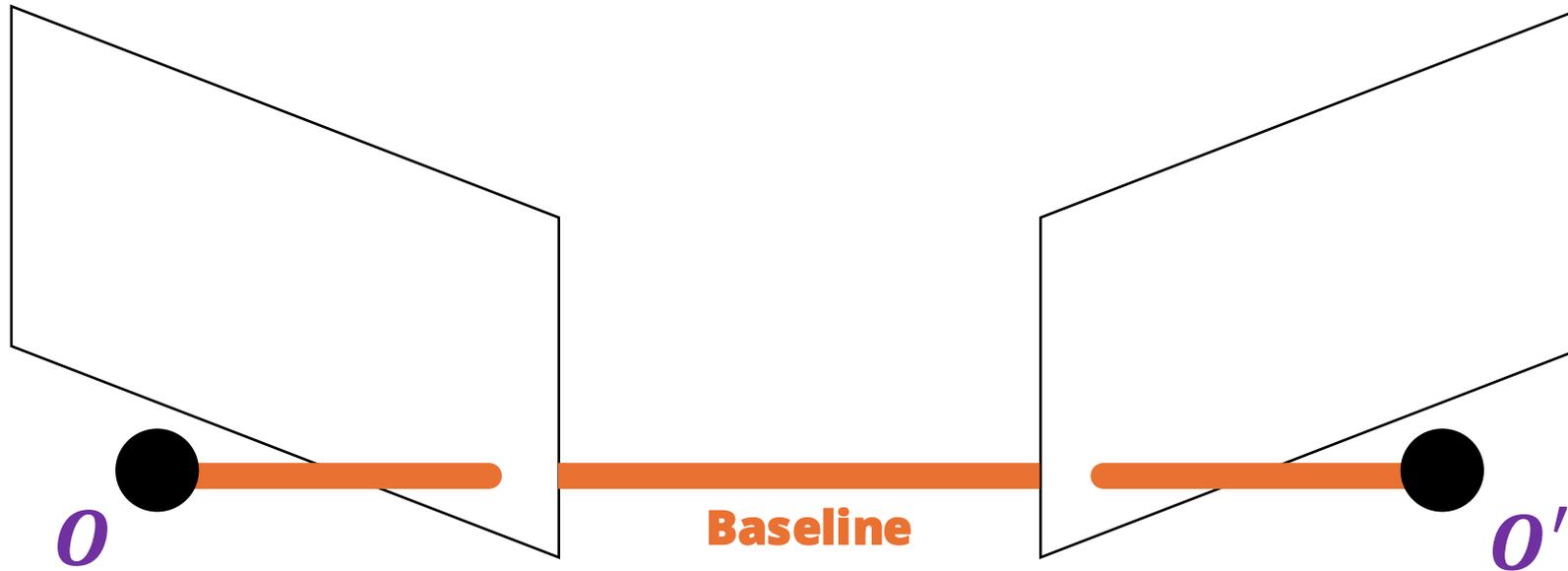
- Projecting points on a ray to another camera forms a **line**, i.e., the corresponding point on the second image *must* lie on this line

Epipolar Constraint



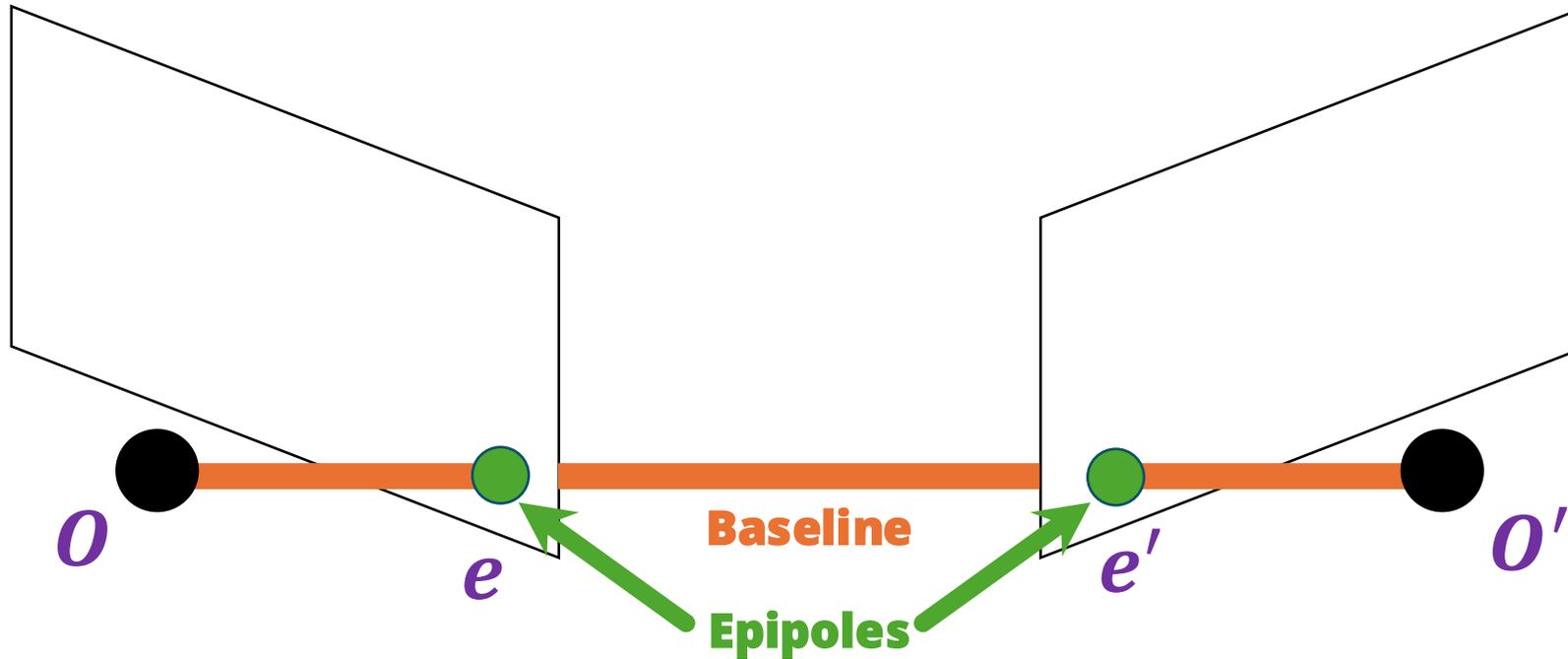
- Projecting points on a ray to another camera forms a **line**, i.e., the corresponding point on the second image *must* lie on this line
- Hence, if the *matched* 2D point is found along this line, we can determine its 3D location

Epipolar Geometry



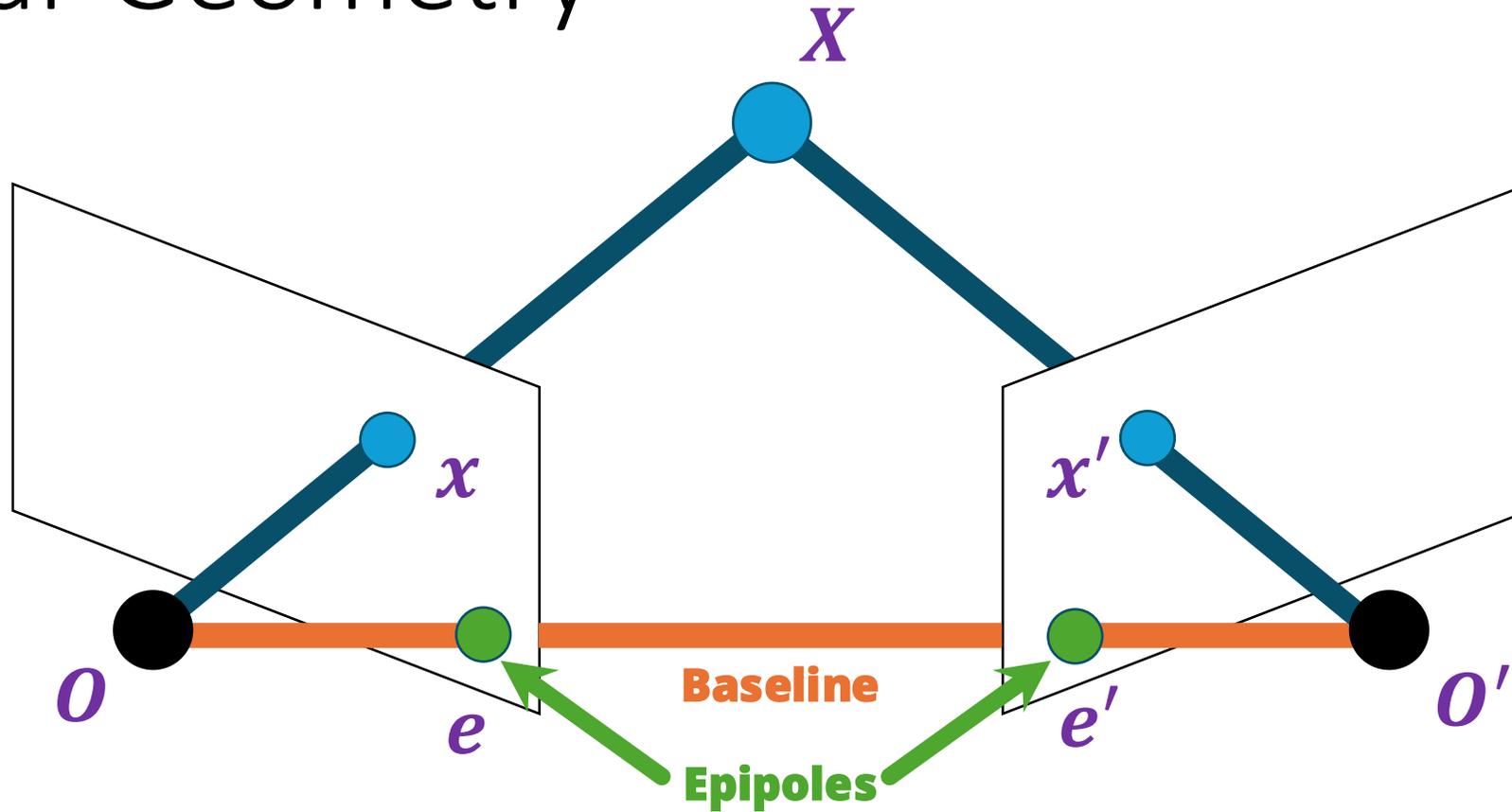
- Suppose we have two cameras with centers O, O'
- The **baseline** is the line connecting the origins

Epipolar Geometry



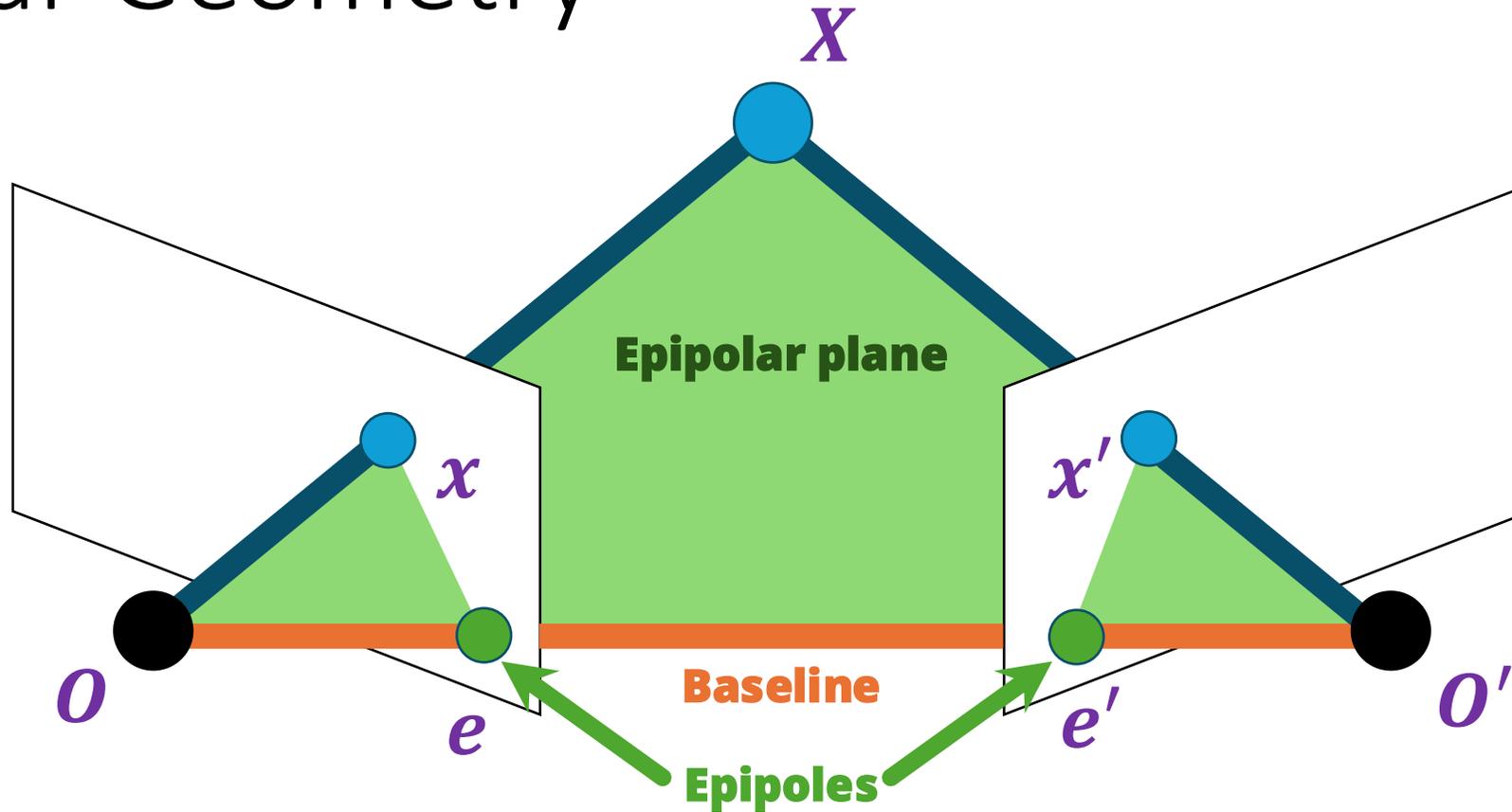
- **Epipoles** e, e' are where the baseline intersects the image planes, or projections of the other camera in each view
- **Epipoles** may lie outside the images

Epipolar Geometry



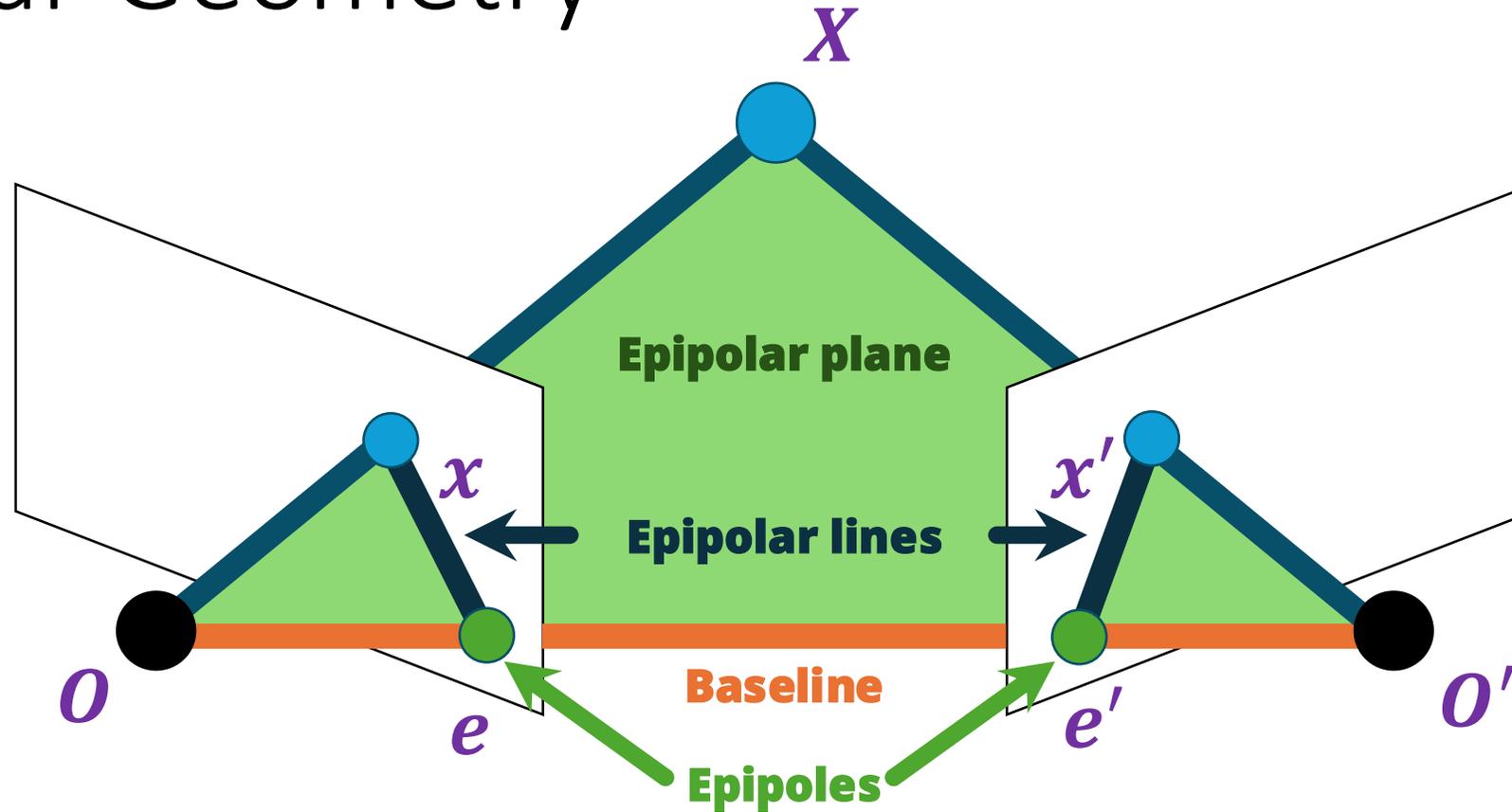
- Consider a **point** X , which projects to x and x'

Epipolar Geometry



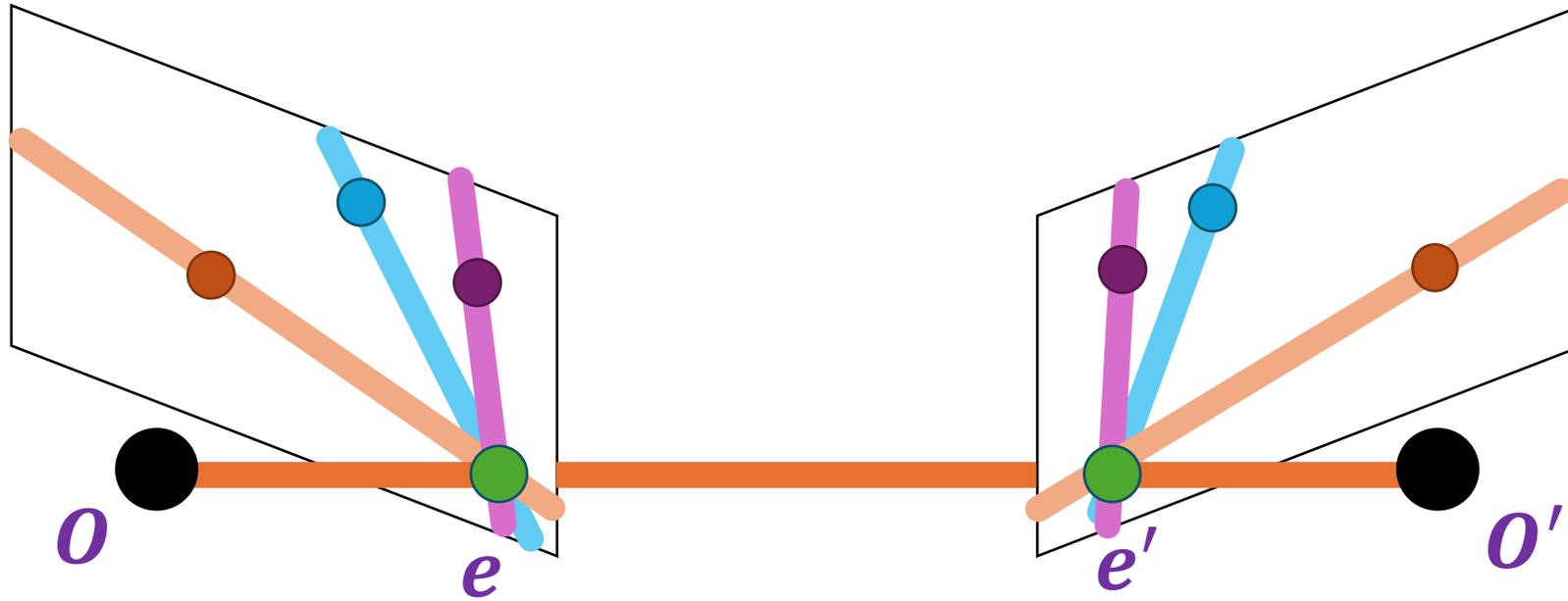
- The plane formed by X , O , and O' is called an **epipolar plane**
- There is a family of planes passing through O and O'

Epipolar Geometry



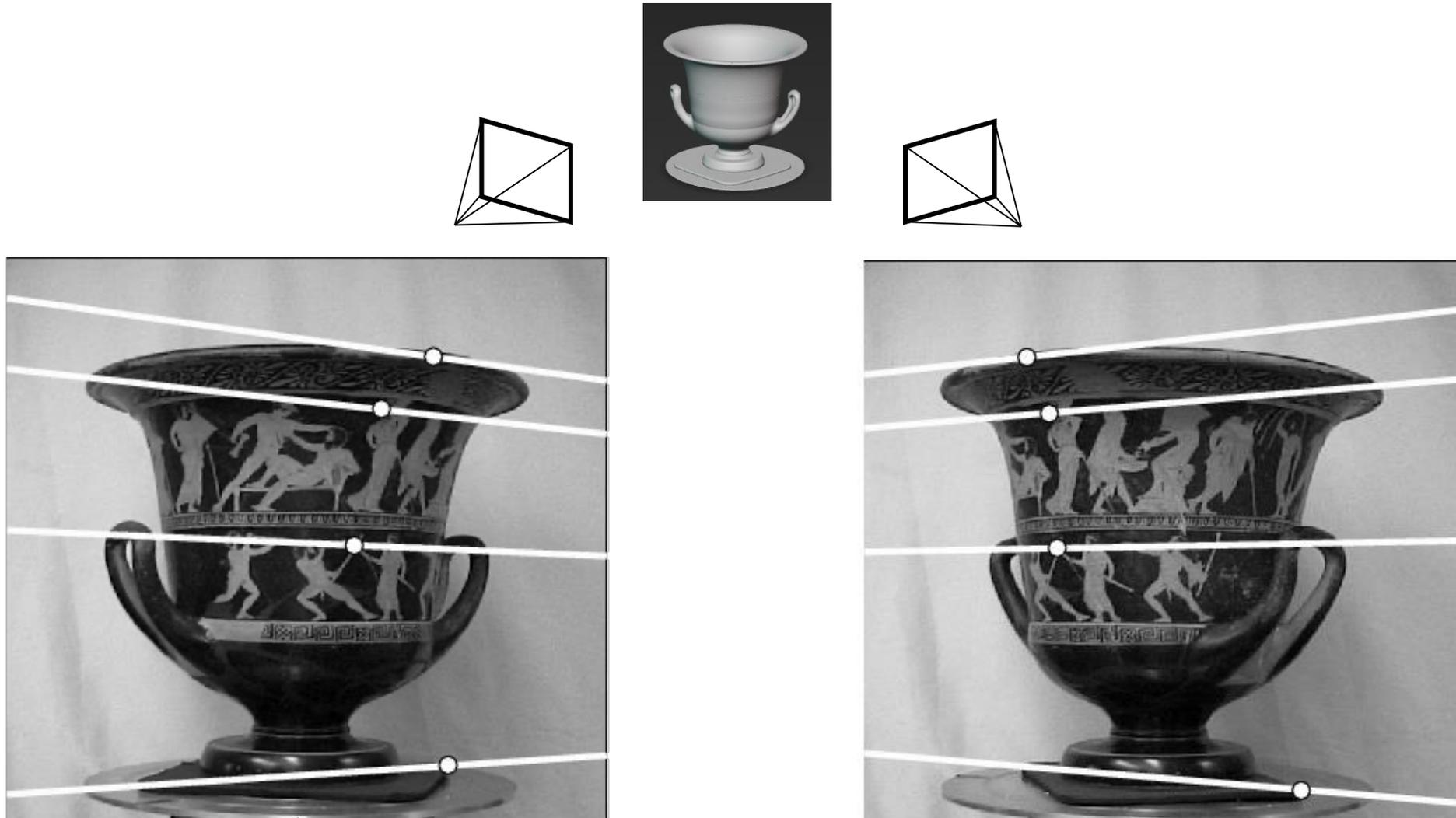
- **Epipolar lines** connect the epipoles to the projections of X
- Equivalently, they are intersections of the epipolar plane with the image planes – thus, they come in matching pairs

Epipolar Geometry

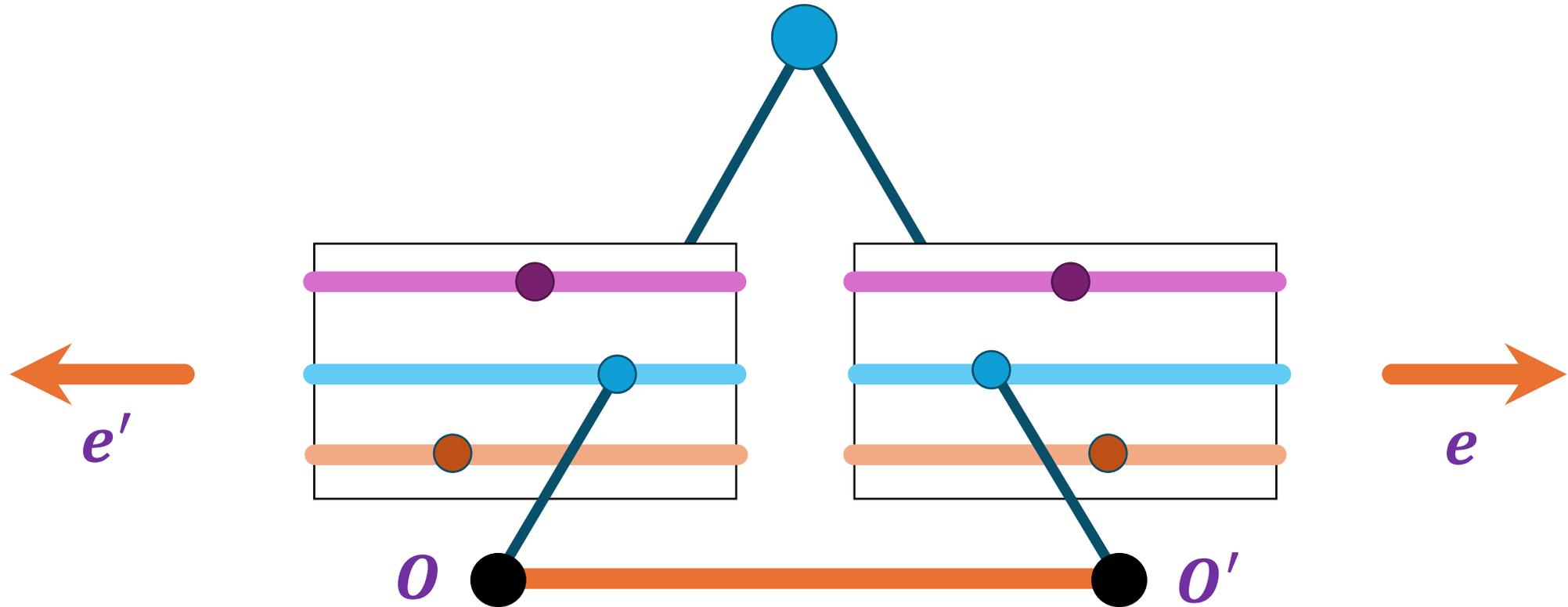


- All **epipolar lines** pass through the **epipoles**

Example of Epipolar Lines

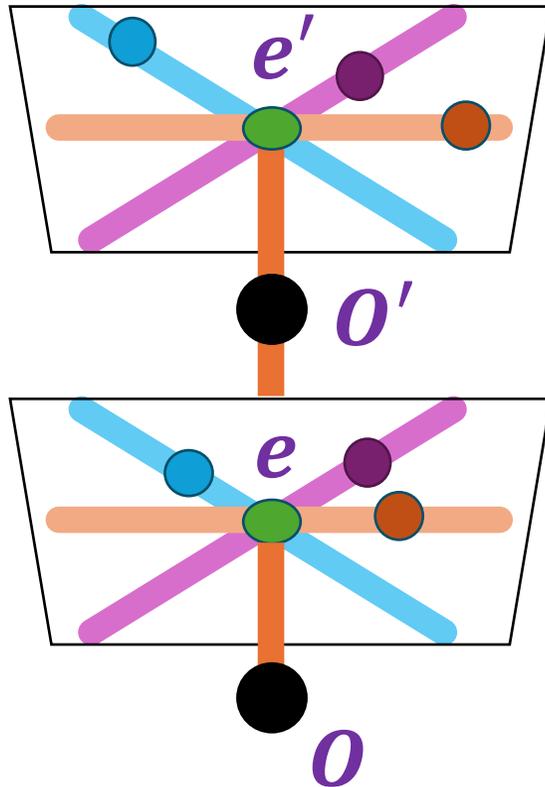


Parallel to Image Plane



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

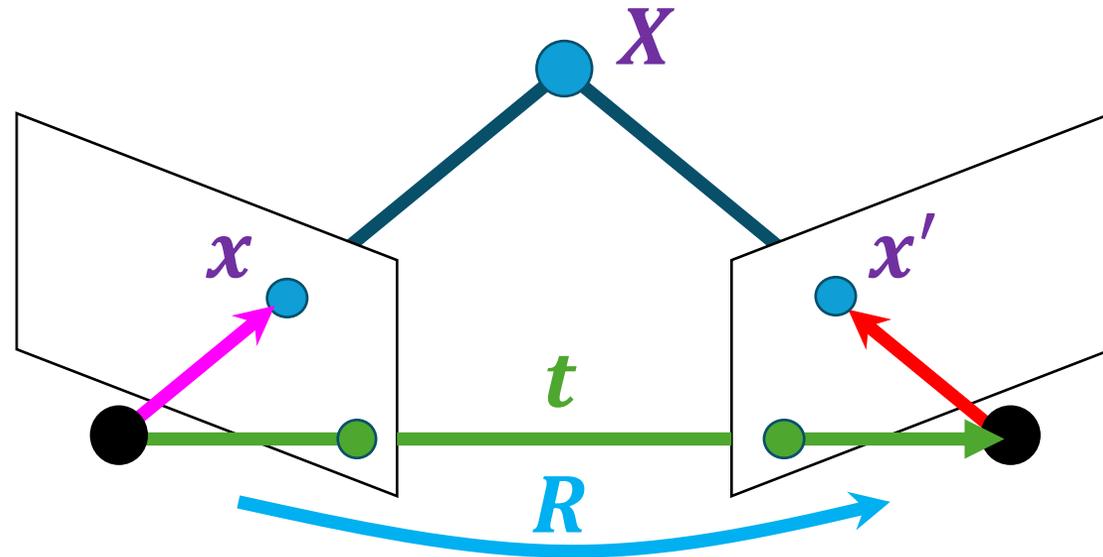
Perpendicular to Image Plane



- 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 $K[I | \mathbf{0}]$ and $K'[R | t]$
- We can pre-multiply the projection matrices (and the image points) by the inverse calibration matrices to get *normalized* image coordinates:

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

Epipolar Constraint: Calibrated Case

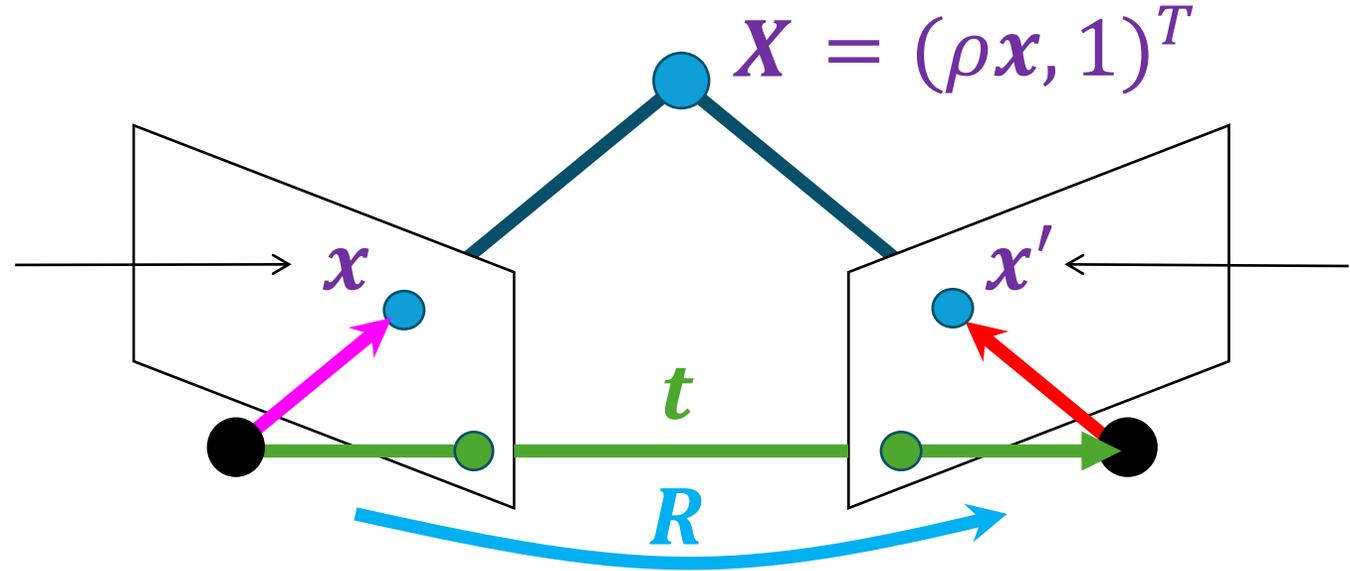
$$x \cong K[R|t]X$$

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

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

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

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



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

- 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

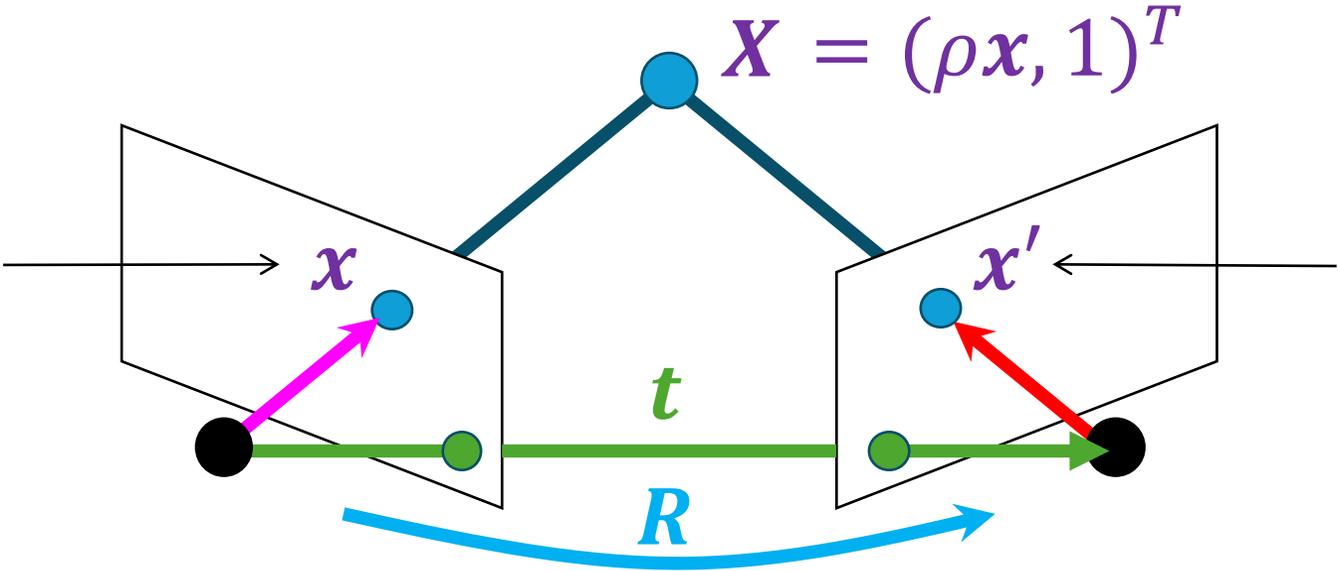
$$x \cong K[R|t]X$$

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

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

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

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



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

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

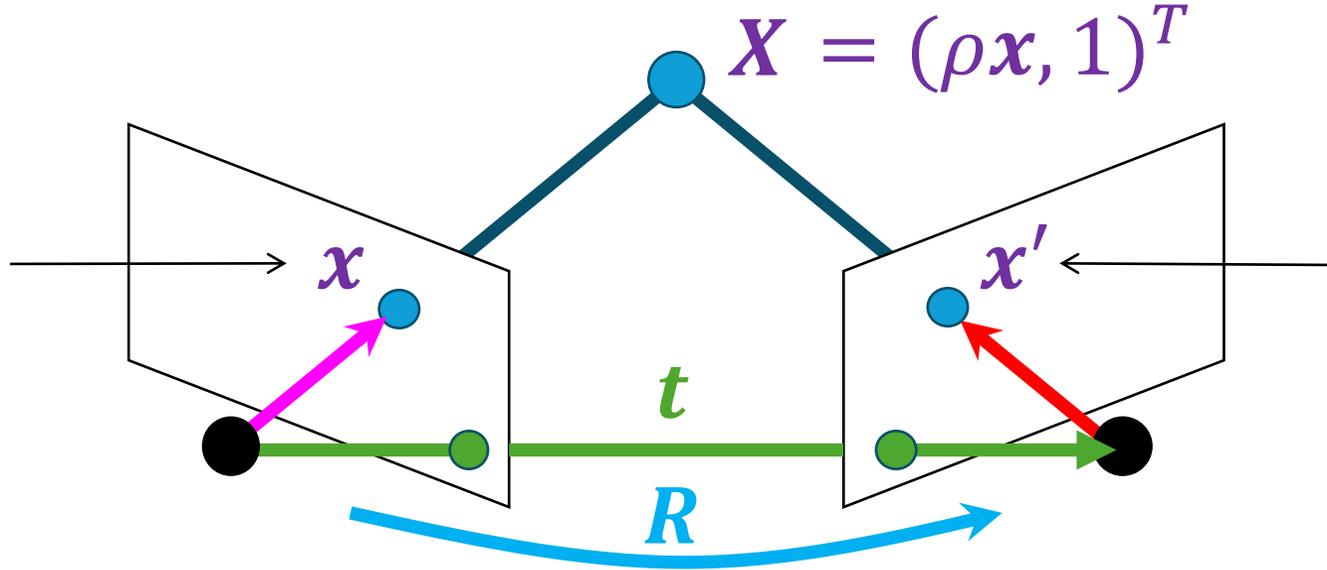
Epipolar Constraint: Calibrated Case

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

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

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

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



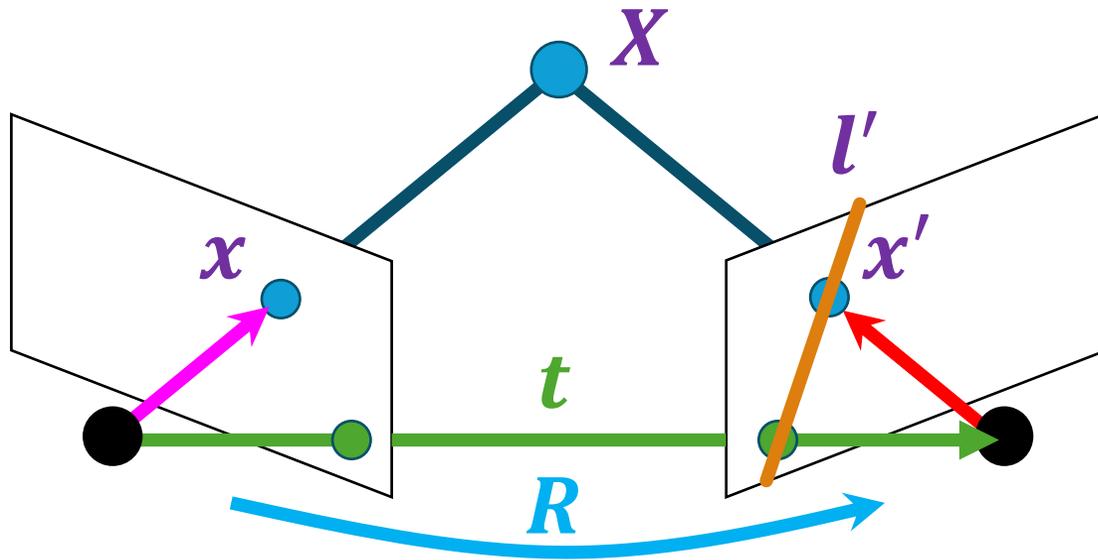
$$[R | t] \begin{pmatrix} \rho x \\ 1 \end{pmatrix} = \rho Rx + t$$

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



Essential Matrix

The Essential Matrix



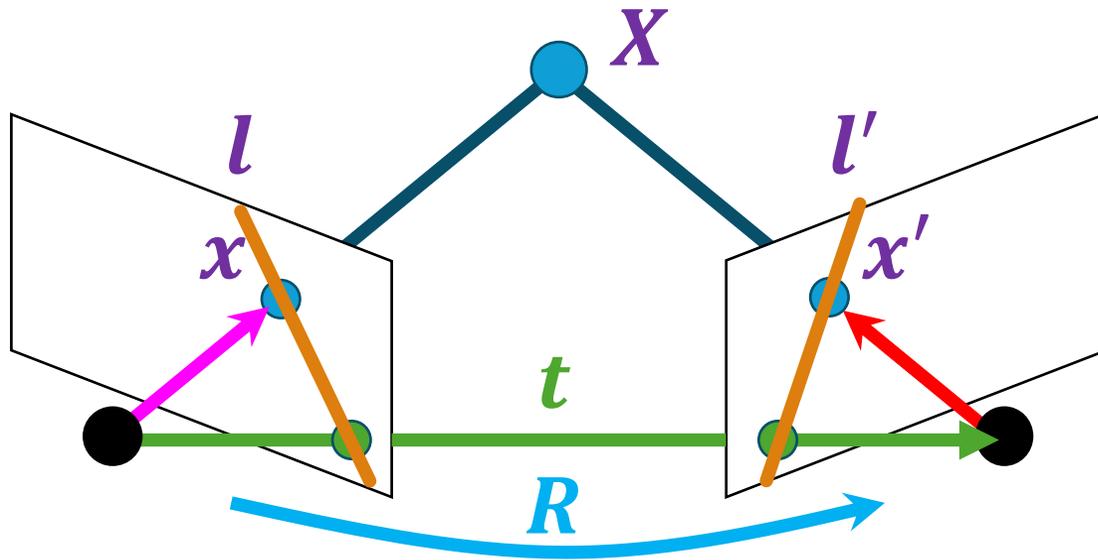
$$l' = E^T x$$

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

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

- $E x$ 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



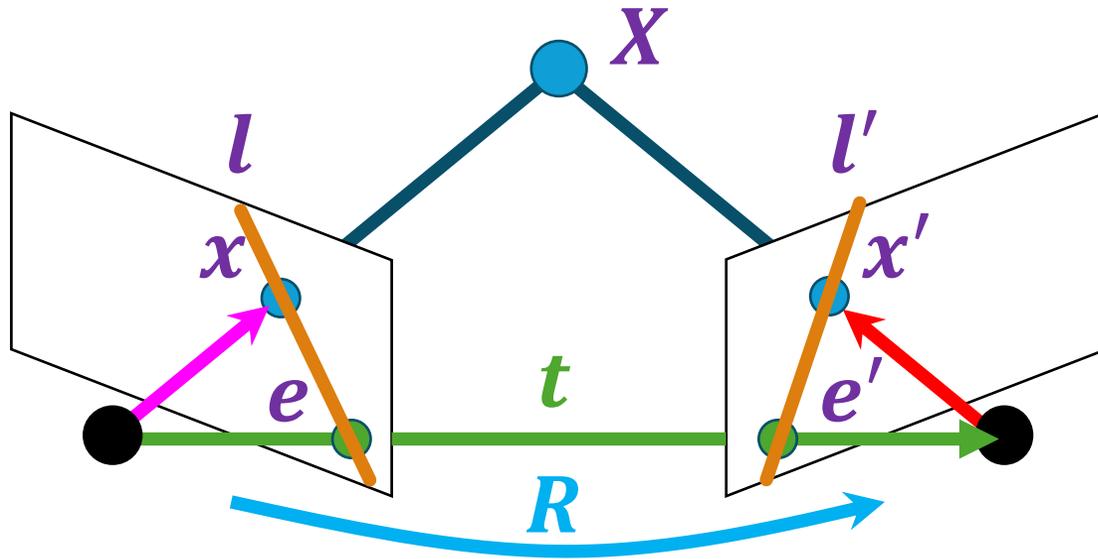
$$l = E^T x'$$

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

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

- $E x$ 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



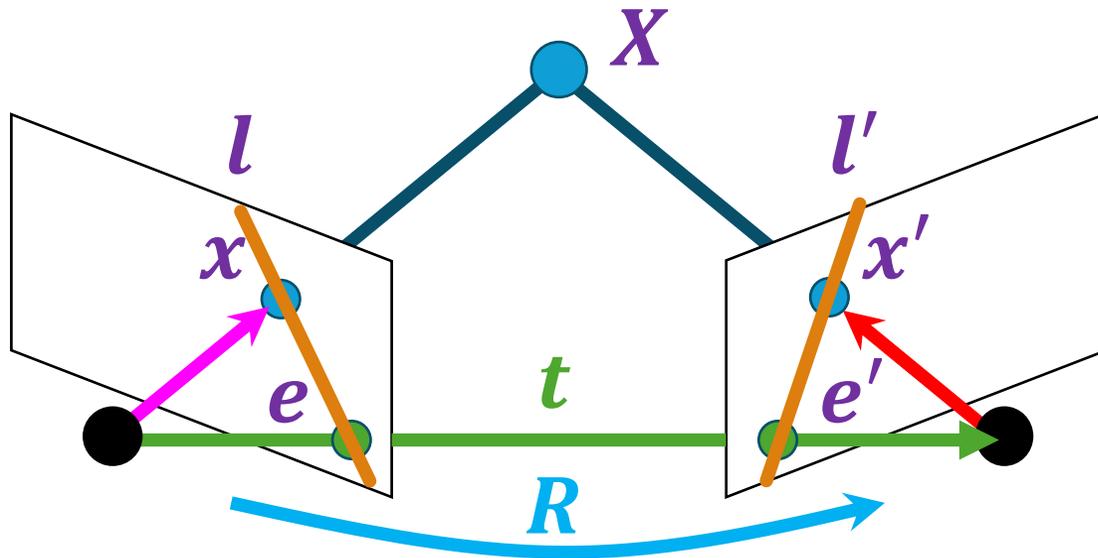
$$\mathbf{x}'^T \mathbf{E} \mathbf{x} = 0$$

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

- $\mathbf{E} \mathbf{x}$ 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

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



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

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

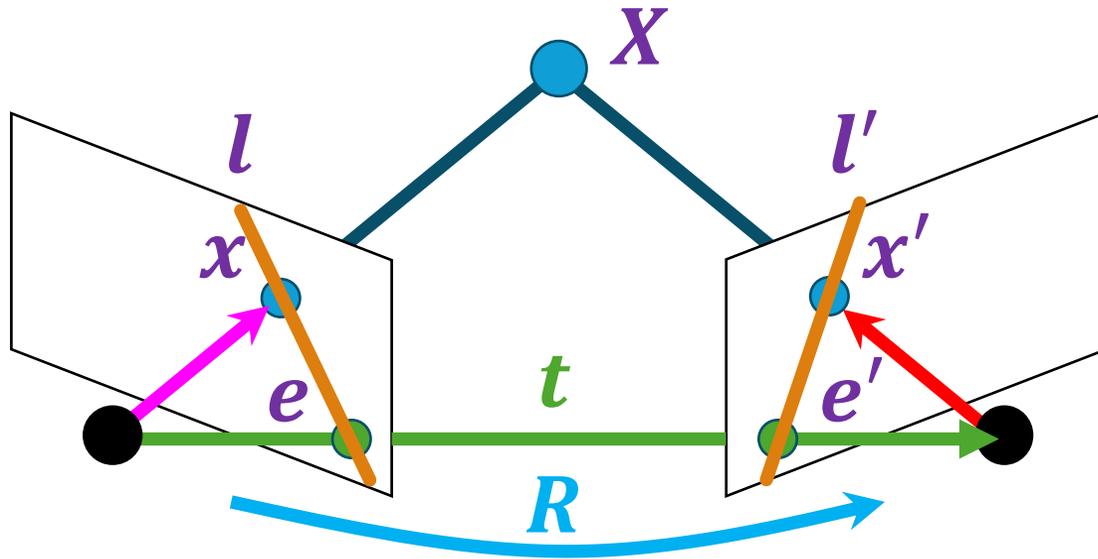
- What if camera intrinsics K, K' are unknown?
- We can write the epipolar constraint:

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

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

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

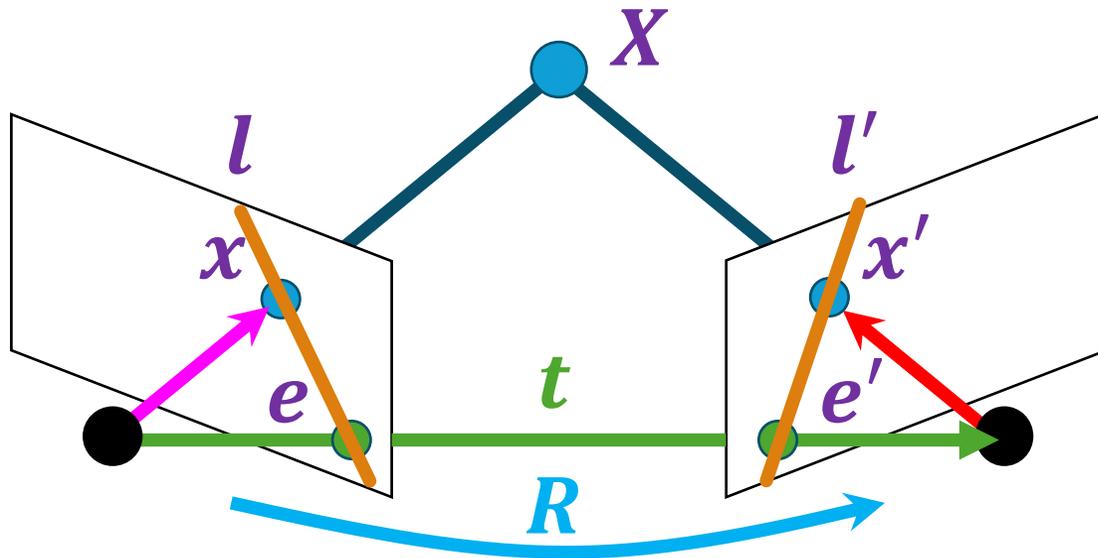
$$F = K'^{-T} E K^{-1}$$

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

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

The Fundamental Matrix

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



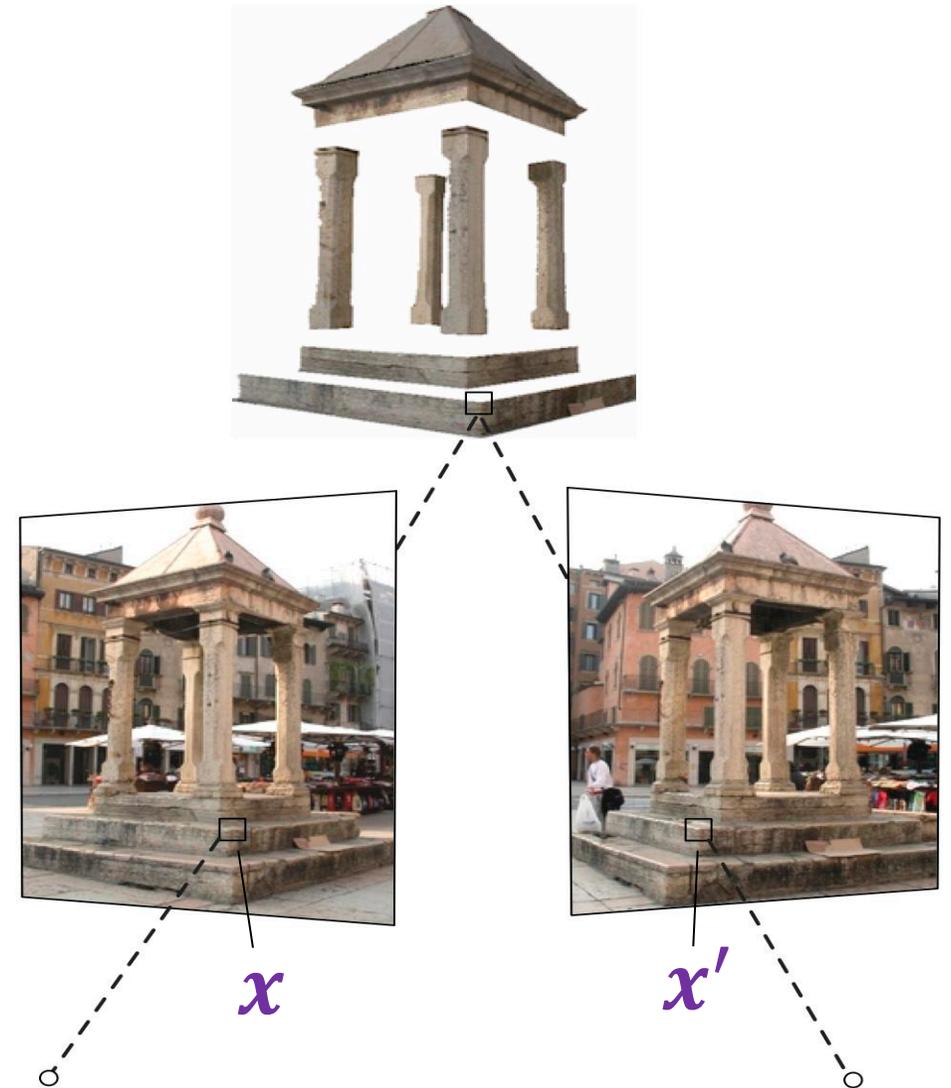
$$x'^T F x = 0$$

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

- 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?
 - $F = K'^{-T} [t_x] R K^{-1}$! $[t_x]$ is skew-symmetric; has rank 2
 - 9 elements, but we lost 1 DoF to scale, and 1 DoF to rank constraint

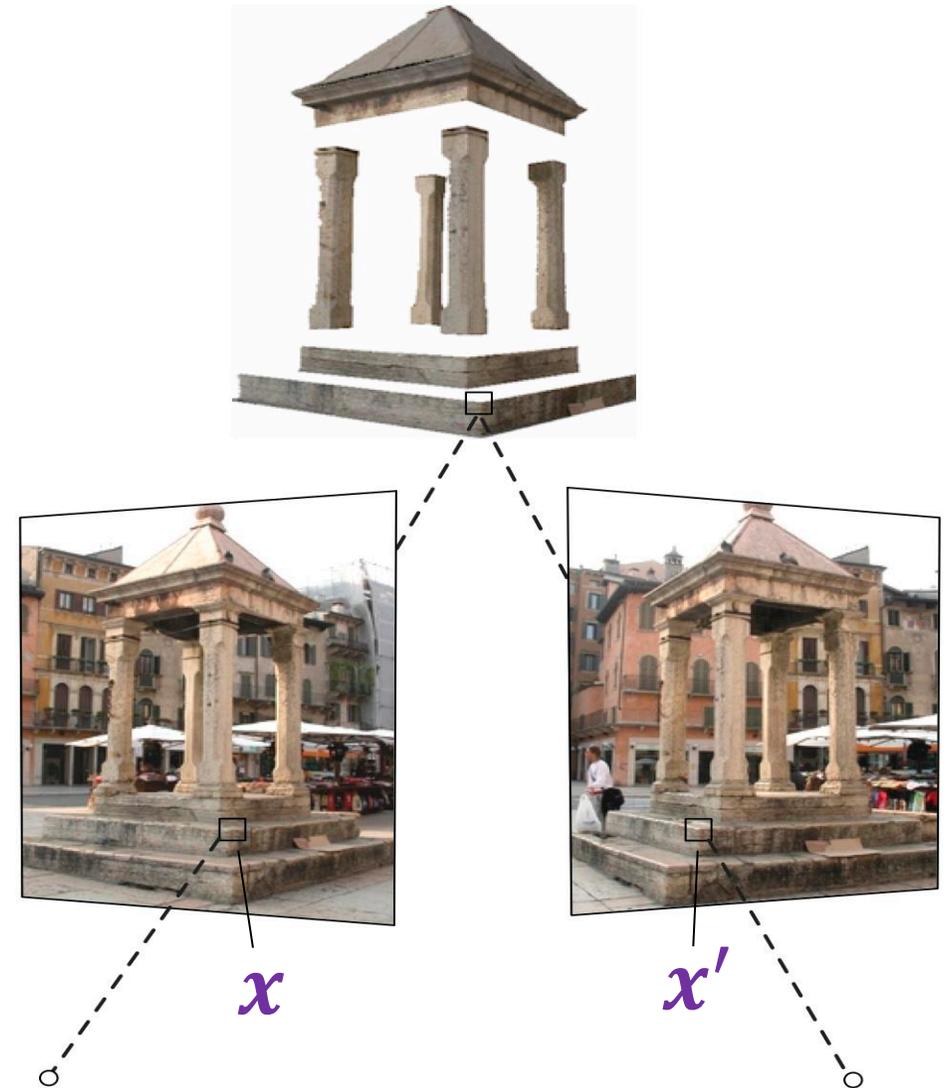
How Can We Use the Epipolar Constraint?

- **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 $E = K'^T FK$
 - **Step 2:** since $E = [t_x]R$, perform SVD on $E = U\Sigma V^T$. We have 4 solutions $(R_1, \pm t), (R_2, \pm t)$, where $t = U[:, 3]$, $R_1 = UWV^T$, $R_2 = UW^T V^T$, 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:** $\mathbf{x}'^T \mathbf{F} \mathbf{x} = 0$
- **Q:** \mathbf{F} ?

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

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:** $\mathbf{x}'^T \mathbf{F} \mathbf{x} = 0$
- **Q:** \mathbf{F} ?

$$\begin{array}{c}
 \mathbf{A} \\
 \left[\begin{array}{cccccccc|c}
 x'_1 x_1 & x'_1 y_1 & x'_1 & y'_1 x_1 & y'_1 y_1 & y'_1 & x_1 & y_1 & 1 \\
 \vdots & \vdots \\
 x'_n x_n & x'_n y_n & x'_n & y'_n x_n & y'_n y_n & y'_n & x_n & y_n & 1
 \end{array} \right]
 \begin{array}{c}
 f_{11} \\
 f_{12} \\
 f_{13} \\
 f_{21} \\
 f_{22} \\
 f_{23} \\
 f_{31} \\
 f_{32} \\
 f_{33}
 \end{array}
 = \mathbf{0}
 \end{array}$$

Homogeneous least squares to find \mathbf{f} :

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

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

A Small Trick to Enforce Rank-2 Constraint

- We know 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

$$F_{\text{init}} = U\Sigma V^T$$



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



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



$$F = U\Sigma'V^T$$

The Fundamental Matrix Song



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

The Fundamental Matrix Song – Live!

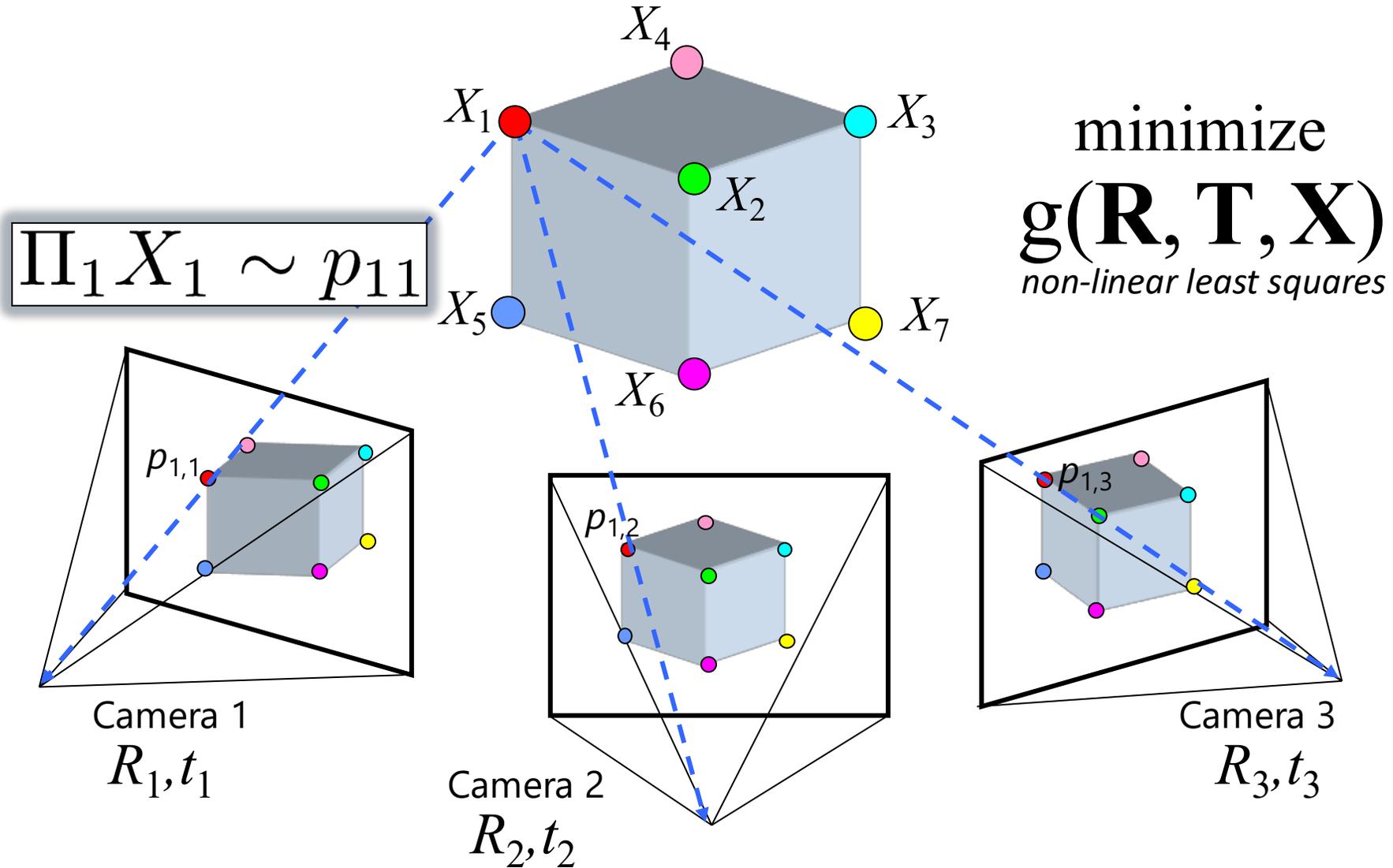


Daniel Wedge and the CVPR house band at CVPR 2023 in Vancouver

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
- Optimization is slow; 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)

Part 1&2 Summary – Multi-view Geometry

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

3D Computer Vision – Outline

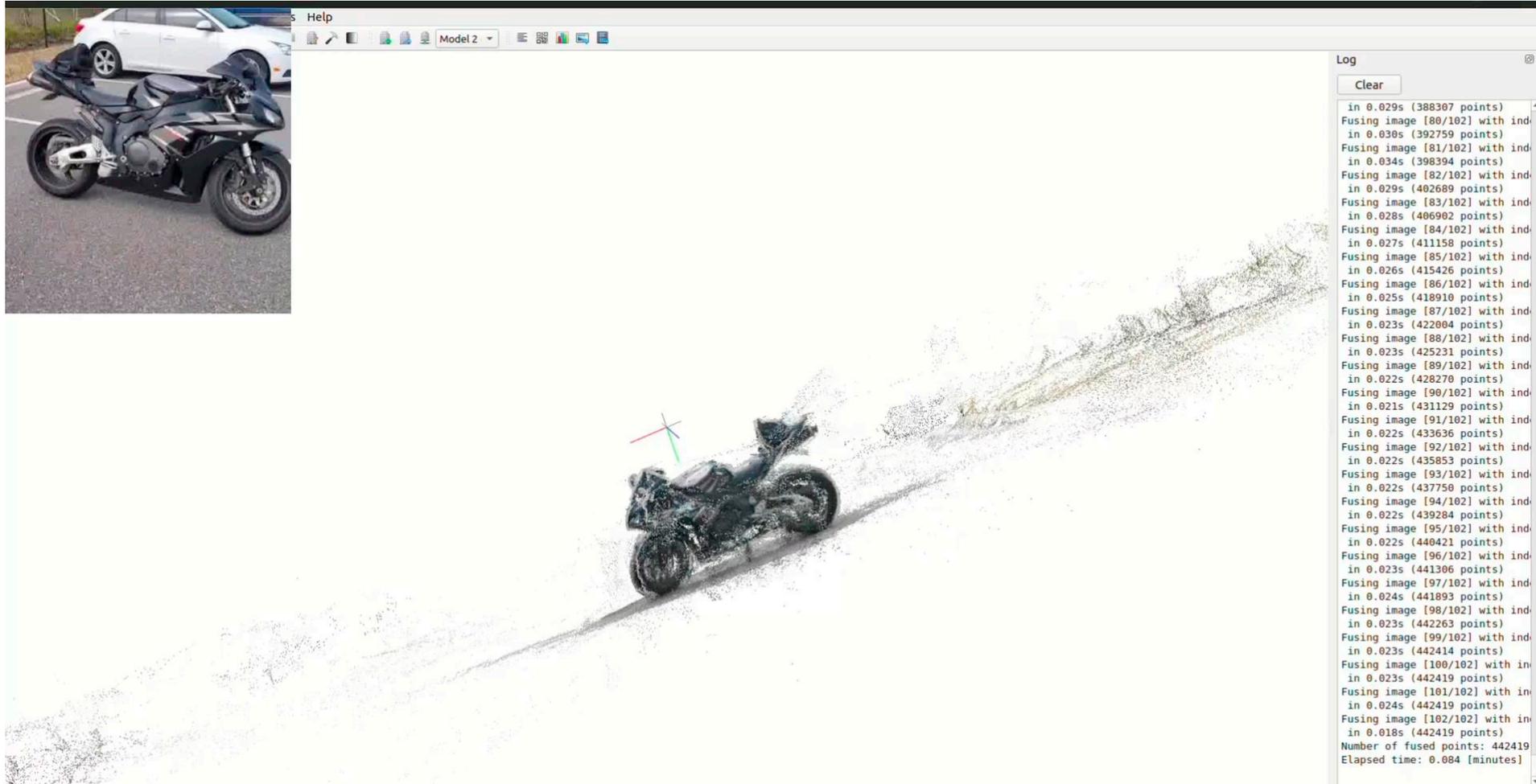
- Part 1 – Camera Model & Projection
- Part 2 – Multi-view Geometry
- Part 3 – Learning-based 3D Reconstruction
- Part 4 – 3D Representations & Rendering

Coursework 2 Released

- PDF: <https://cambridgecvcourses.github.io/assets/CW/CW2.pdf>
- Codebase: <https://github.com/CambridgeCVCourses/CW2>
- **Due:** Tuesday, 24 March 2026, 12:00 PM (noon)
- Office hours: Fridays, 3-4PM (BE4-54)

Part 3 – Learning-based 3D Reconstruction

Optimization is Slow



- 102 images
- 50+ mins for COLMAP (with GPU)

Dense reconstruction using COLMAP MVS – 102 images. <https://learnopencv.com/dust3r-geometric-3d-vision/>

DUST3R: Geometric 3D Vision Made Easy

Shuzhe Wang*, Vincent Leroy†, Yann Cabon†, Boris Chidlovskii† and Jerome Revaud†

*Aalto University

†Naver

shuzhe.wang@aalto.fi

firstname.last

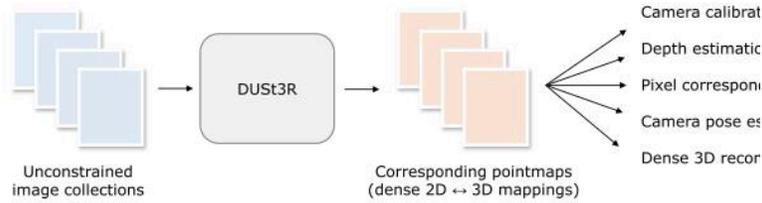


Figure 1. **Overview:** Given an unconstrained image collection, *i.e.* a set of photographs w proposed method **DUST3R** outputs a set of corresponding *pointmaps*, from which we can t quantities normally difficult to estimate all at once, such as the camera parameters, pixel cor 3D reconstruction. Note that DUST3R also works for a single input image (*e.g.* achieving in show **qualitative examples** on the DTU, Tanks and Temples and ETH-3D datasets [1, 51, 10]. For each sample, from *left to right*: input image, colored point cloud, and rendered with shadir

VGGT: Visual Geometry Grounded Transformer

Jianyuan Wang^{1,2}

Minghao Chen^{1,2}

Nikita Karaev^{1,2}

Andrea Vedaldi^{1,2}

Christian Rupprecht¹

David Novotny²

¹Visual Geometry Group, University of Oxford

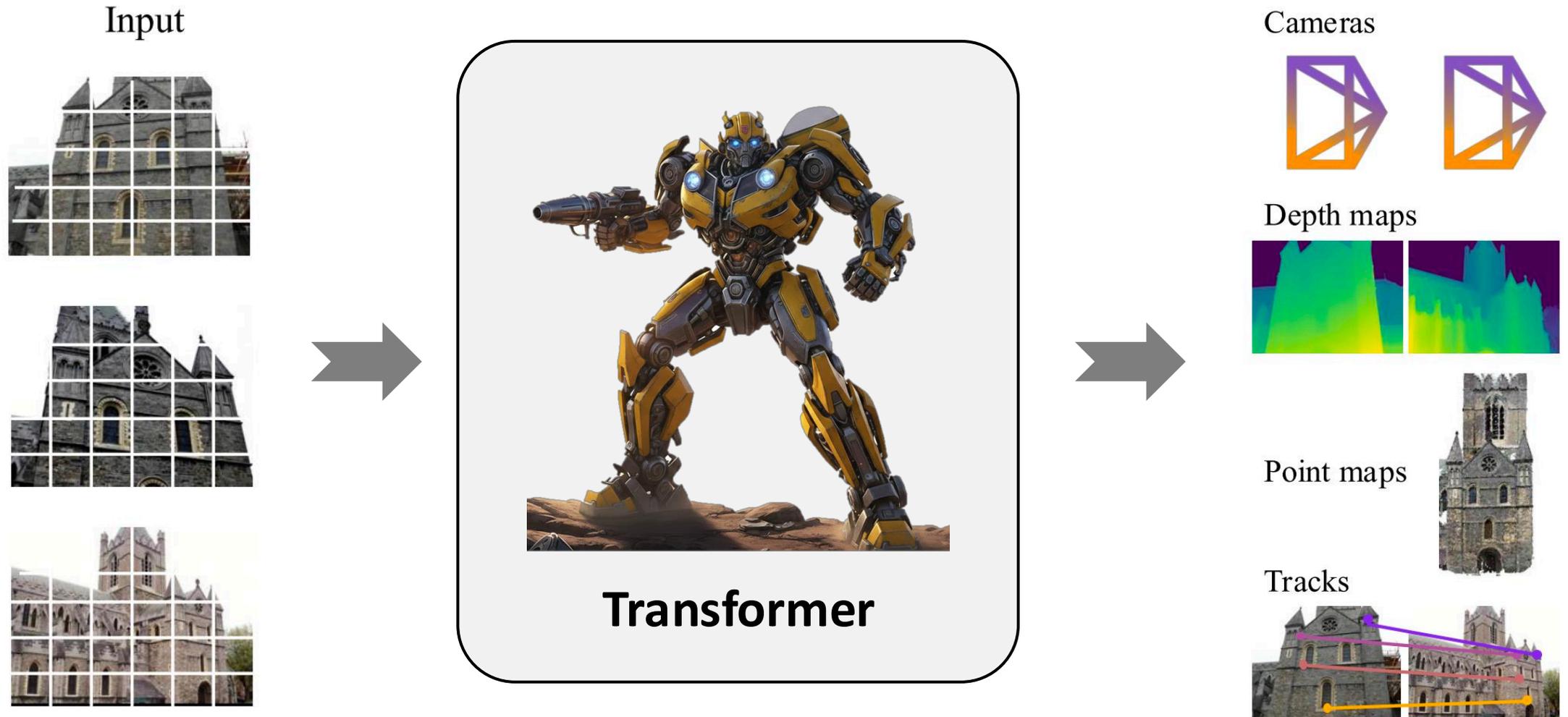
²Meta AI

Feed-forward 3D in seconds!!



Figure 1. **VGGT** is a large feed-forward transformer with minimal 3D-inductive biases trained on a trove of 3D-annotated data. It accepts up to hundreds of images and predicts cameras, point maps, depth maps, and point tracks for all images at once in less than a second, which often outperforms optimization-based alternatives without further processing.

Feed-forward 3D Reconstruction



Feed-forward 3D Reconstruction

32 Views



Feed-forward 3D Reconstruction

32 Views



Feed-forward 3D Reconstruction

32 Views

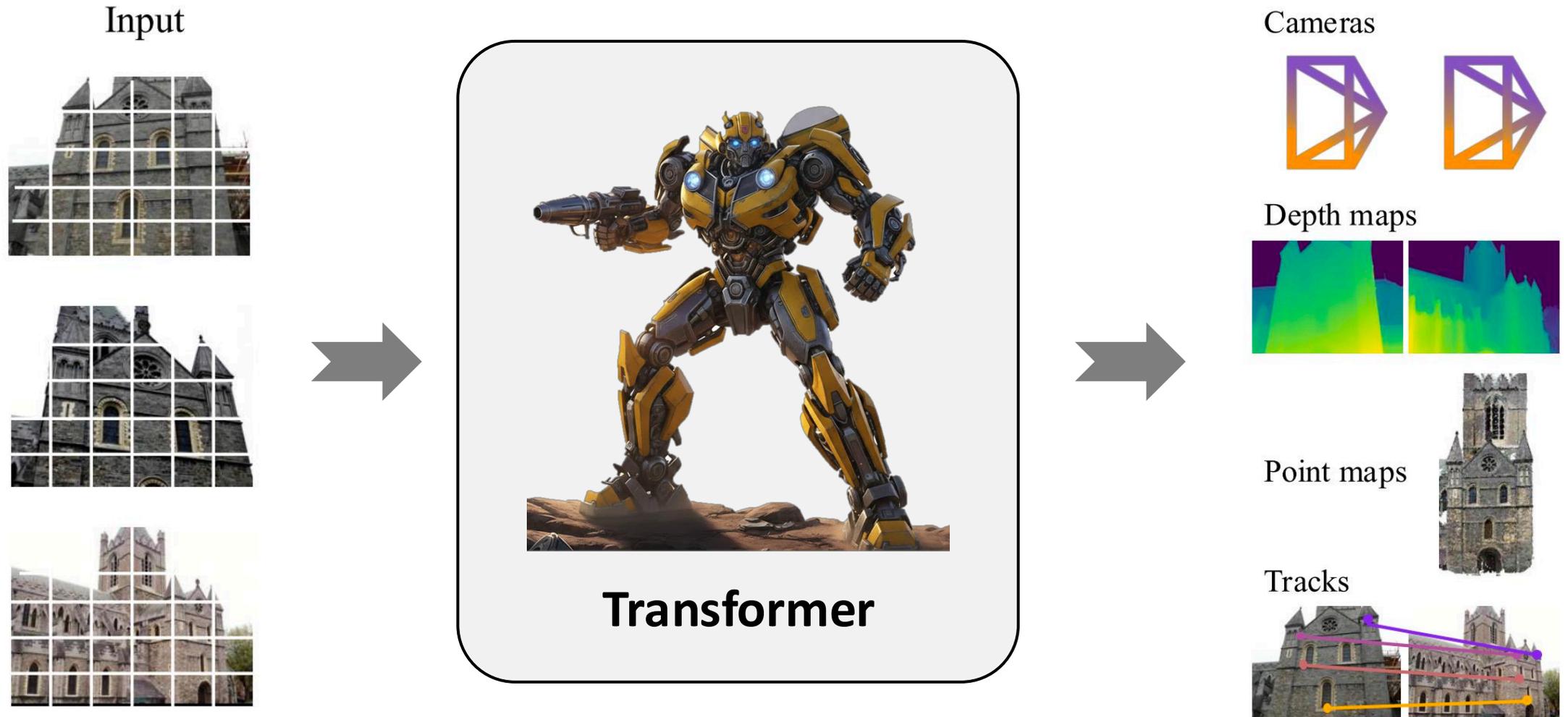


Feed-forward 3D Reconstruction

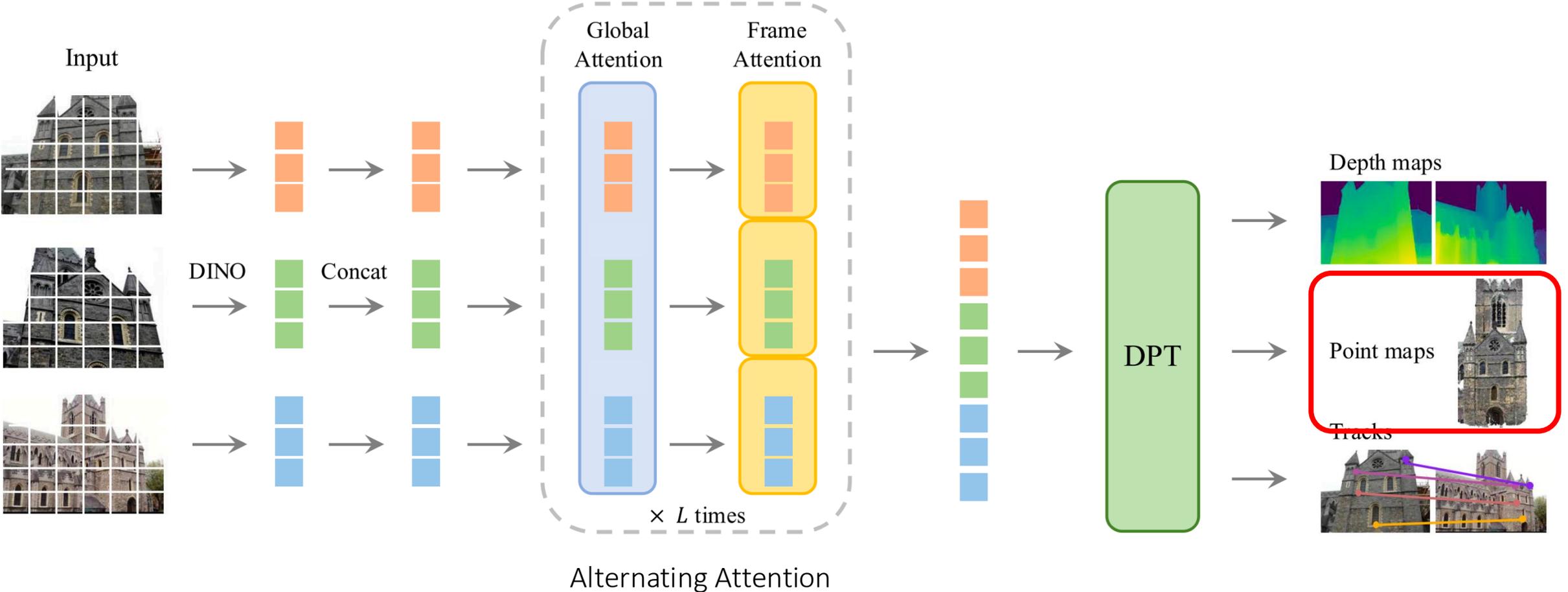
Methods	Re10K (<i>unseen</i>) AUC@30 \uparrow	CO3Dv2 AUC@30 \uparrow	Time
Colmap+SPSG [92]	45.2	25.3	\sim 15s
PixSfM [66]	49.4	30.1	$>$ 20s
PoseDiff [124]	48.0	66.5	\sim 7s
DUS _t 3R [129]	67.7	76.7	\sim 7s
MASt3R [62]	76.4	81.8	\sim 9s
VGGSfM v2 [125]	78.9	83.4	\sim 10s
MV-DUS _t 3R [111] \ddagger	71.3	69.5	\sim 0.6s
CUT3R [127] \ddagger	75.3	82.8	\sim 0.6s
FLARE [156] \ddagger	78.8	83.3	\sim 0.5s
Fast3R [141] \ddagger	72.7	82.5	\sim 0.2s
Ours (Feed-Forward)	<u>85.3</u>	<u>88.2</u>	\sim 0.2s
Ours (with BA)	93.5	91.8	\sim 1.8s

Table 1. **Camera Pose Estimation on RealEstate10K [161] and CO3Dv2 [88]** with 10 random frames. All metrics the higher the better. None of the methods were trained on the Re10K dataset. Runtime were measured using one H100 GPU. Methods marked with \ddagger represent concurrent work.

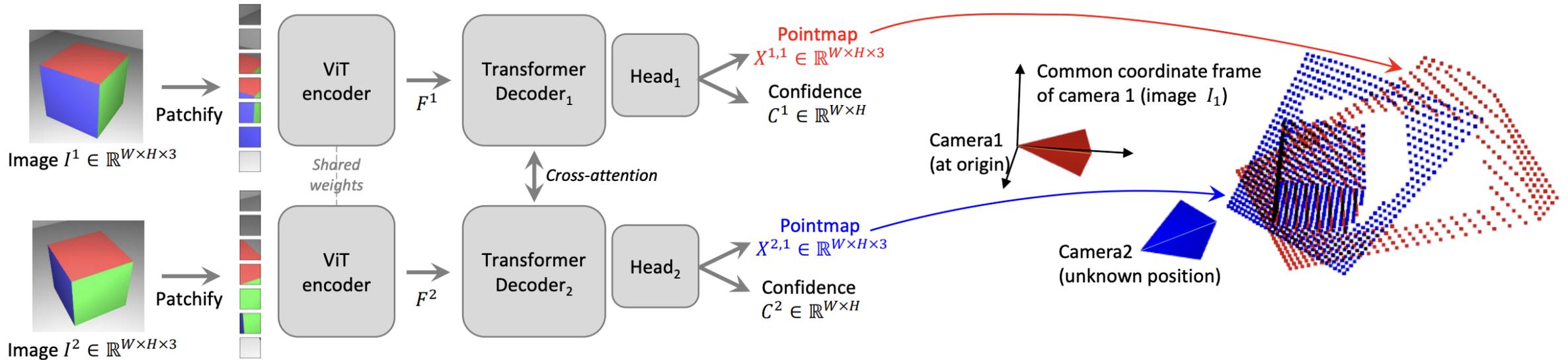
Feed-forward 3D Reconstruction



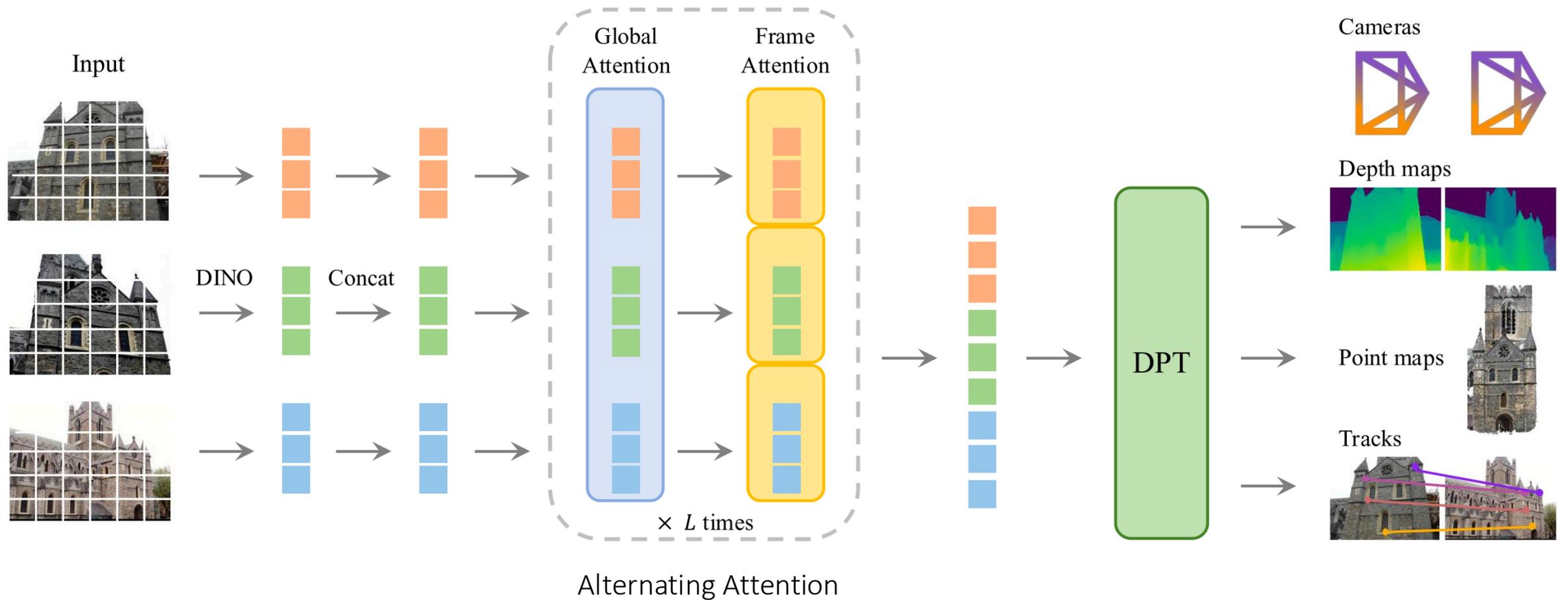
Feed-forward 3D Reconstruction



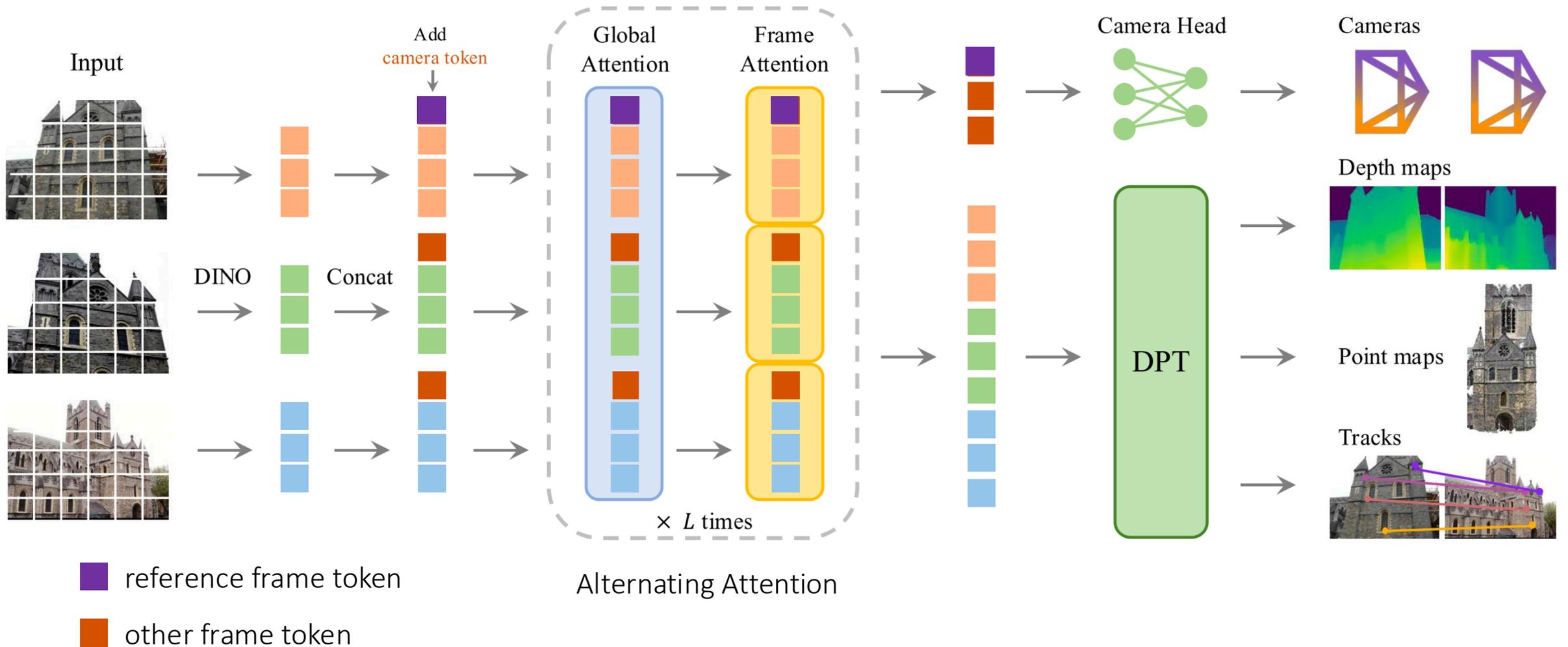
Point Maps from DUSSt3R



Feed-forward 3D Reconstruction



Feed-forward 3D Reconstruction



Of Course This is Obvious, Right?

- No, not at all!
- We have long thought that supervised learning would not conquer 3D
 - We thought we would at least need some 3D inductive biases
- Early evidence of supervised 3D learning:
 - Monocular depth estimation (2.5D)

Depth Map Prediction from a Single Image using a Multi-Scale Deep Network

David Eigen
deigen@cs.nyu.edu

Christian Puhrsch
cpuhrsch@nyu.edu

Rob Fergus
fergus@cs.nyu.edu

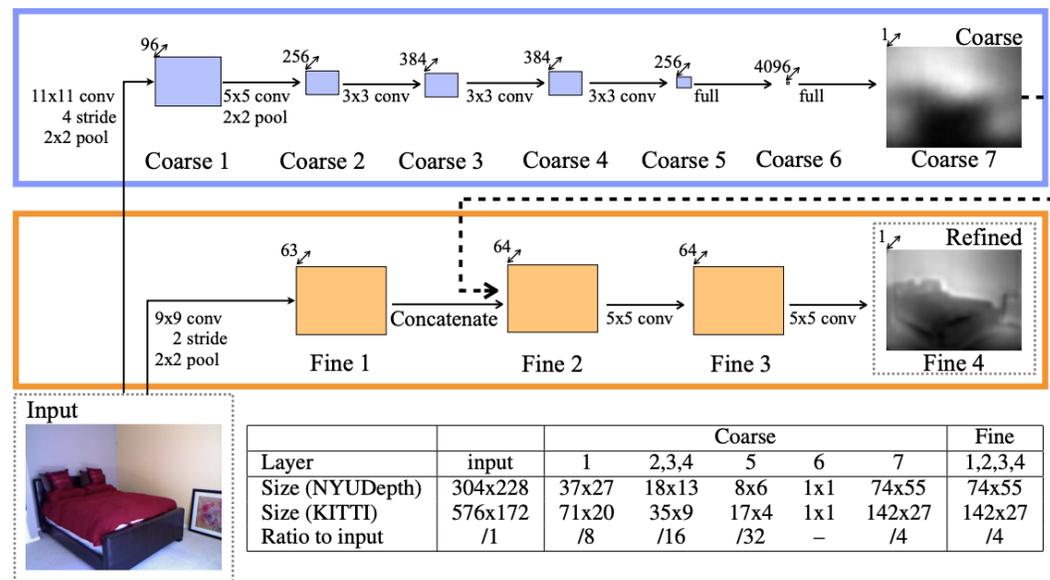


Figure 1: Model architecture.

Of Course This is Obvious, Right?

- No, not at all!
- We have long thought that supervised learning would not conquer 3D
 - We thought we would at least need some 3D inductive biases
- Early evidence of supervised 3D learning:
 - Monocular depth estimation (2.5D)
 - 6DoF pose regression (eg, PoseNet)
 - 7 indoor + 5 outdoor scenes

ICCV 2015

PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization

Alex Kendall

Matthew Grimes

Roberto Cipolla

University of Cambridge

agk34, mkg30, rc10001 @cam.ac.uk



King's College

Old Hospital

Shop Façade

St Mary's Church

Of Course This is Obvious, Right?

- No, not at all!
- We have long thought that supervised learning would not conquer 3D
 - We thought we would at least need some 3D inductive biases
- Early evidence of supervised 3D learning:
 - Monocular depth estimation (2.5D)
 - 6DoF pose regression (eg, PoseNet)
 - 7 indoor + 5 outdoor scenes
- We still believed we would never have enough 3D training data
 - We then looked into unsupervised 3D learning using reprojection losses

Of Course This is Obvious, Right?

- No, not at all!
- We have long thought that s
 - We thought we would at least
- Early evidence of supervised
 - Monocular depth estimation
 - 6DoF pose regression (eg, PoseNet)
 - 7 indoor + 5 outdoor scenes
- We still believed we would reach the goal
 - We then looked into unsupervised

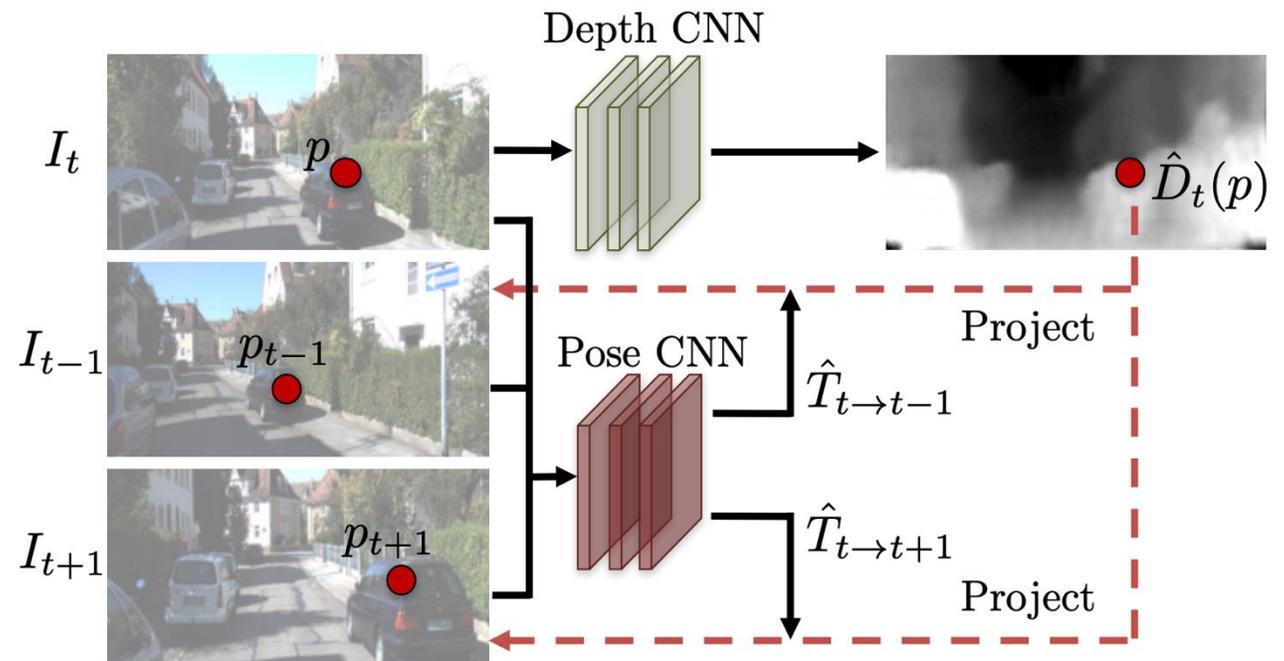
Unsupervised Learning of Depth and Ego-Motion from Video

Tinghui Zhou*
UC Berkeley

Matthew Brown
Google

Noah Snavely
Google

David G. Lowe
Google



ICCV 2017

Of Course This is Obvious, Right?

- No, not at all!
- We have long thought that supervised learning would not conquer 3D
 - We thought we would at least need some 3D inductive biases
- Early evidence of supervised 3D learning:
 - Monocular depth estimation (2.5D)
 - 6DoF pose regression (eg, PoseNet)
 - 7 indoor + 5 outdoor scenes
- We still believed we would never have enough 3D training data
 - We then looked into unsupervised 3D learning using reprojection losses
- Until very recently when we have large-scale 3D datasets

Of Course This is Obvious, Right?

- Preparing training data was a HUGE effort in VGGT
 - 17 large datasets of both synthetic and real-captured 3D scenes (800K-1M scenes)
- And traditional SfM is not dead (yet) :)

Training Data. The model was trained using a large and diverse collection of datasets, including: Co3Dv2 [88], BlendMVS [146], DL3DV [69], MegaDepth [64], Kubric [41], WildRGB [135], ScanNet [18], HyperSim [89], Mapillary [71], Habitat [107], Replica [104], MVS-Synth [50], PointOdyssey [159], Virtual KITTI [7], Aria Synthetic Environments [82], Aria Digital Twin [82], and a synthetic dataset of artist-created assets similar to Objaverse [20]. These datasets span various domains, including indoor and outdoor environments, and encompass synthetic and real-world scenarios. The 3D annotations for these datasets are derived from multiple sources, such as direct sensor capture, synthetic engines, or SfM techniques [95]. The combination of our datasets is broadly comparable to those of MAST3R [30] in size and diversity.

Meanwhile, 3D Generative Models Are Also Going Wild!

Hunyuan3D v2.5 by Tencent Hunyuan - Open

Hunyuan 3D: Free C Text-to-3D

Hunyuan3D by Tencent Hunyuan generates
3D assets. Hunyuan Tencent 3D creates a
seconds. Try Hunyuan 3D Studio f

✓ Apache 2.0 License ✓ Runs on 6GB VRAM

3D



See Output

SAM 3D: 3Dfy Anything in Images

SAM 3D Team, Xingyu Chen*, Fu-Jen Chu*, Pierre C... J Liang*, Alexander Sax*, Hao Tang*,
Weiyao Wang*, Michelle Guo, Thibaut Hardin, Xiang... Ziqi Ma*, Anushka Sagar,
Bowen Song*, Xiaodong Wang, Jianing Yang*, Bow... Matt
Feiszli†§, Jitendra Malik†§

Meta Superintelligence Labs

*Core Contributor (Alphabetical, Equal Contribu

We present SAM 3D, a generative model for v
geometry, texture, and layout from a single im
and scene clutter are common and visual reco
this with a human- and model-in-the-loop p
providing visually grounded 3D reconstructi
in a modern, multi-stage training framewo
alignment, breaking the 3D “data barrier”
at least a 5 : 1 win rate in human preference
code and model weights, an online demo, r
reconstruction.

Demo: <https://www.aidemos.meta.com/se>

Code: <https://github.com/facebookresearch>

Website: <https://ai.meta.com/sam3d>



Figure 1 SAM 3D converts a single image into a composable 3D scene made of individual objects. Our method predicts per-object geometry, texture, and layout, enabling full scene reconstruction. Bottom: high-quality 3D assets recovered for each object.



SAM 3D: 3Dfy Anything in Images. Meta SAM 3D Team. 2025.

As of Today (Feb 2026)

- The best closed-source model but available via an API is (perhaps) **Hunyuan3D-V3.1**: <https://3d.hunyuanglobal.com/>
- The best almost open-source model is **SAM 3D** (no training code): <https://ai.meta.com/blog/sam-3d/>
 - Excels at scene level reconstruction with occlusion
- The best open-source model is **TRELLIS 2**: <https://microsoft.github.io/TRELLIS.2/>
 - Reconstructs from object-centric images but produces more detailed geometry

3D Computer Vision – Outline

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