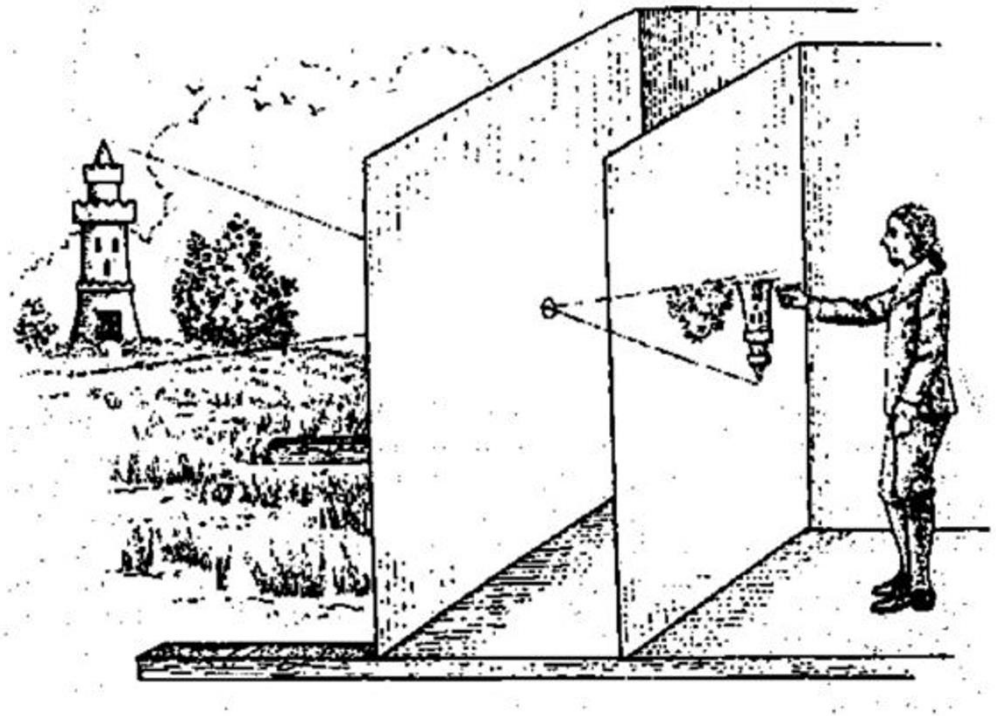
It’s a 3D World, After All



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.

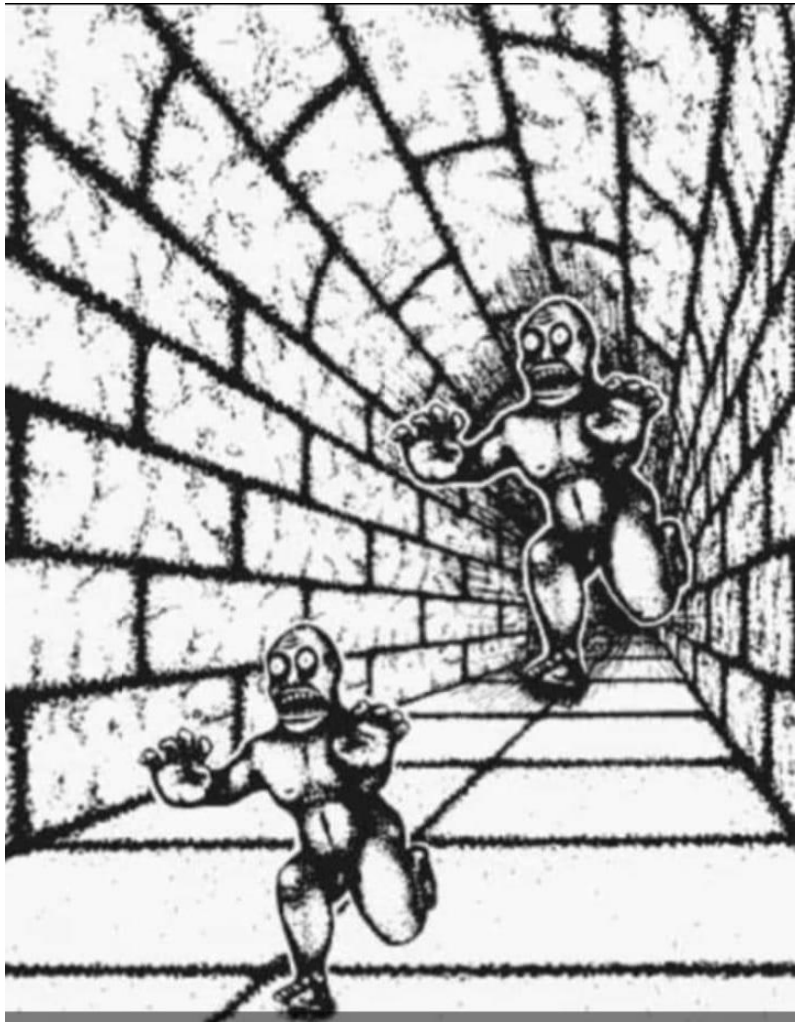
Historically, we have treated vision as a mapping from images to intermediate representations—classes, segmentation masks, or 3D reconstructions. But in the era of the Bitter Lesson, these distinct tasks are becoming qualitatively no different than edge detection: historical artifacts of scoping "solvable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

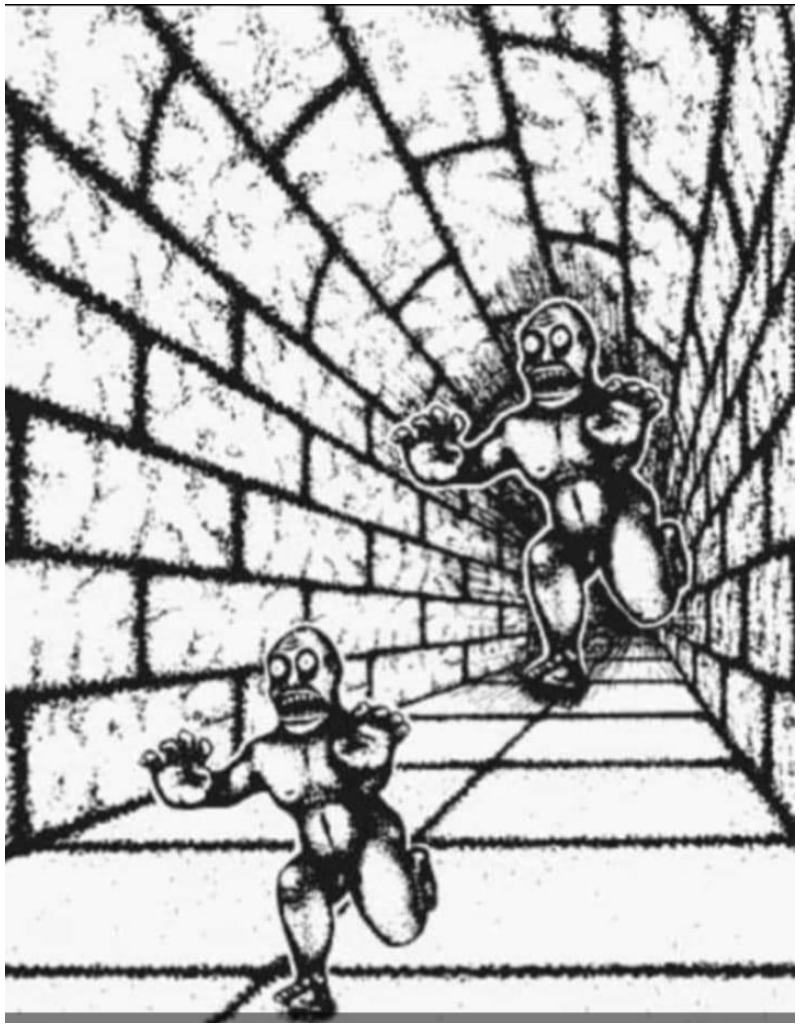
In this post, I argue that the future of computer vision is as part of end-to-end perception-action loops. The historical boundaries between computer vision, robot learning, and control will dissolve. Frontier research will no longer draw a boundary between "seeing" and "learning to act."

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!

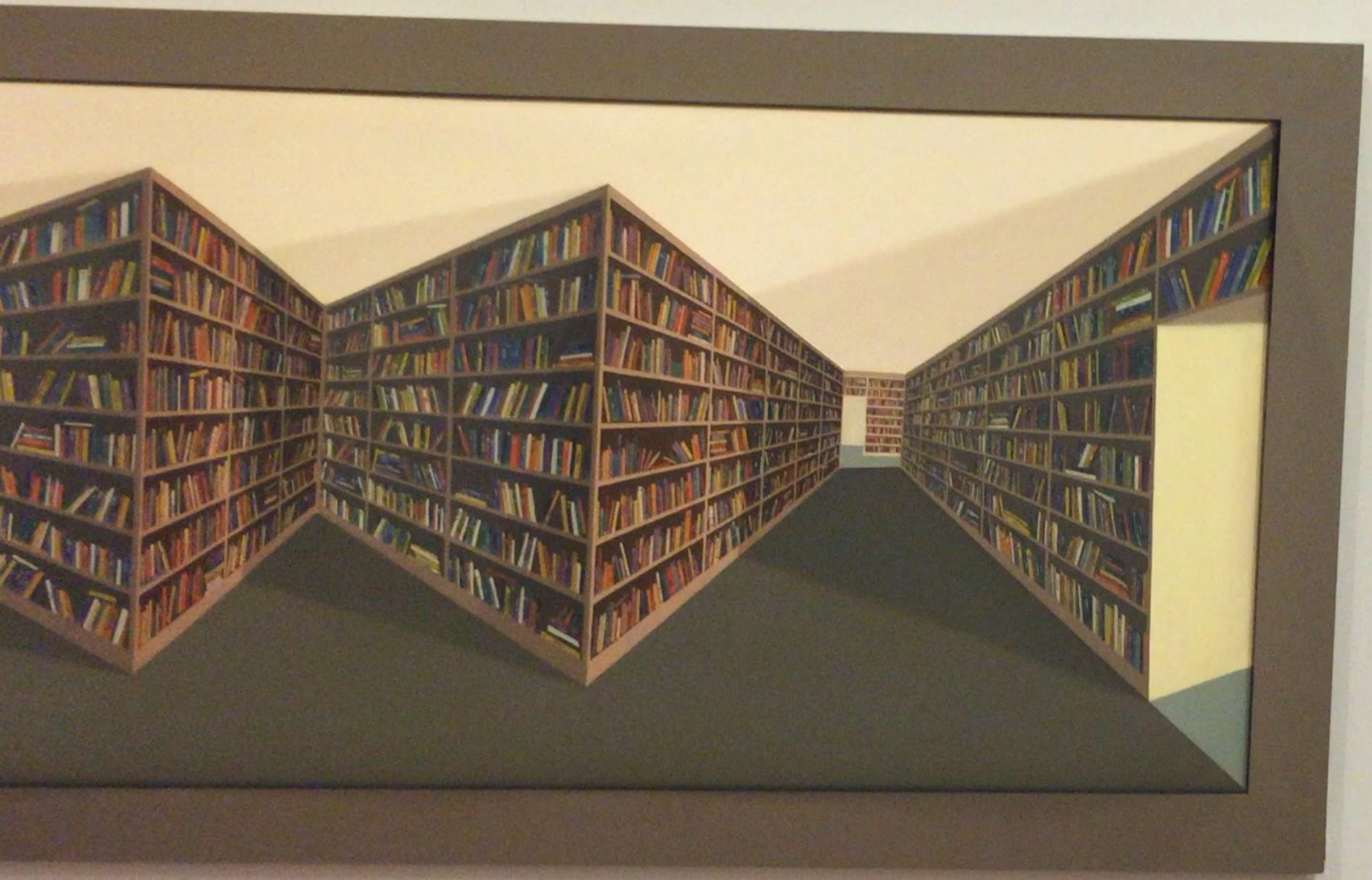


We’re good at seeing 3D!



And sometimes too good..

And sometimes too good..

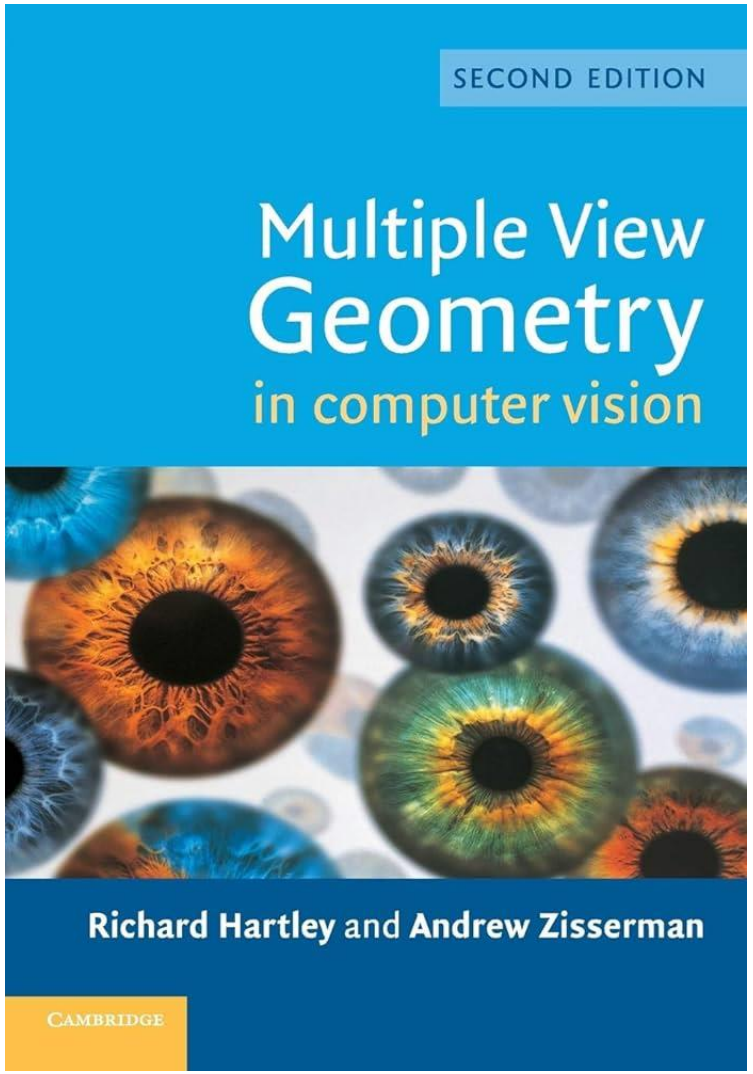


These ‘Reverspective’ paintings are mind-boggling. https://youtu.be/fzZy1FtYsVM

3D Computer Vision – Outline

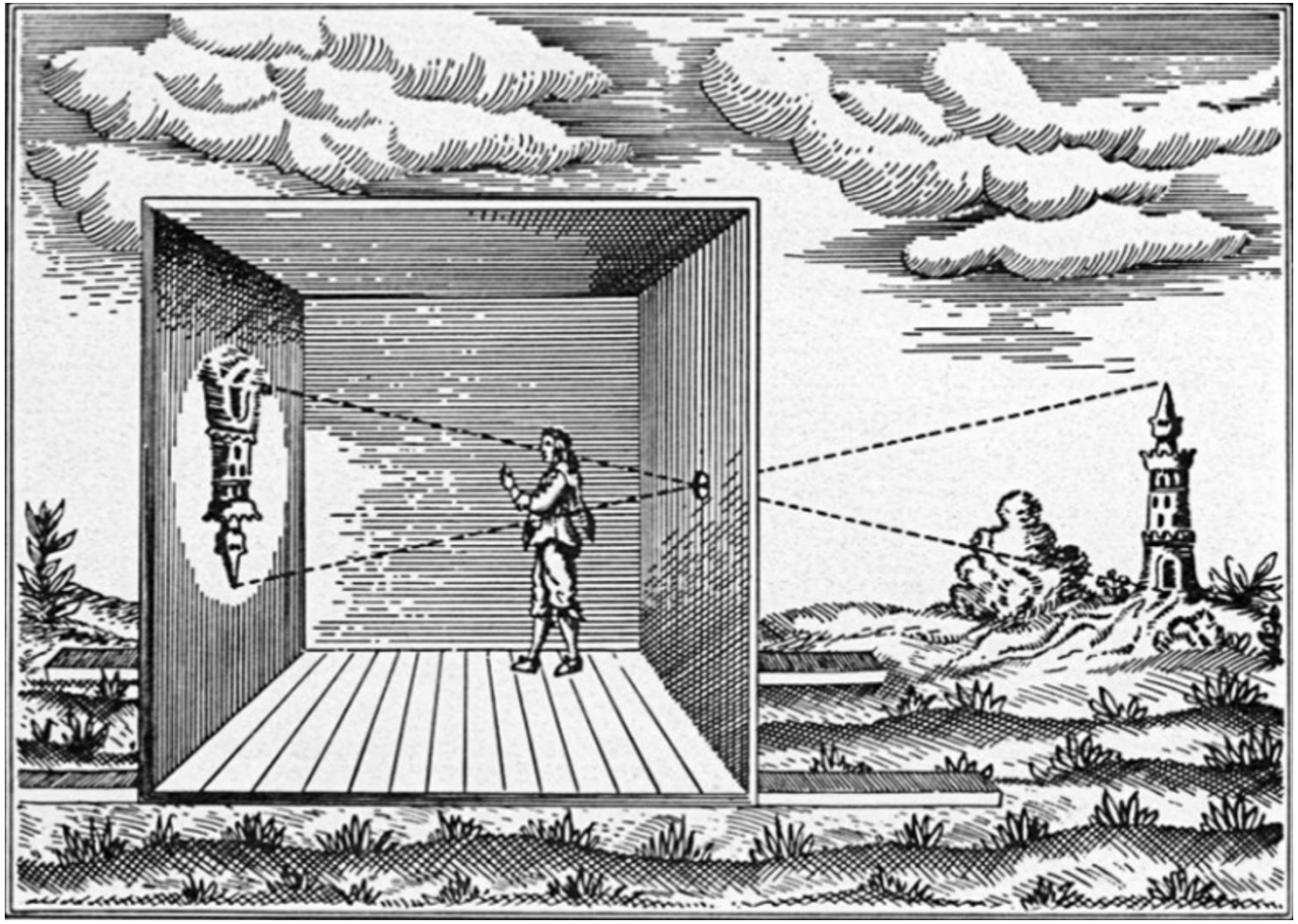
- Part 1 – Camera Model & Projection
- Part 2 – Multi-view Geometry
- Part 3 – Learning-based 3D Reconstruction
- Part 4 – 3D Representations & Rendering

Multi-view Geometry “Bible”



Part 1 – Camera Model & Projection

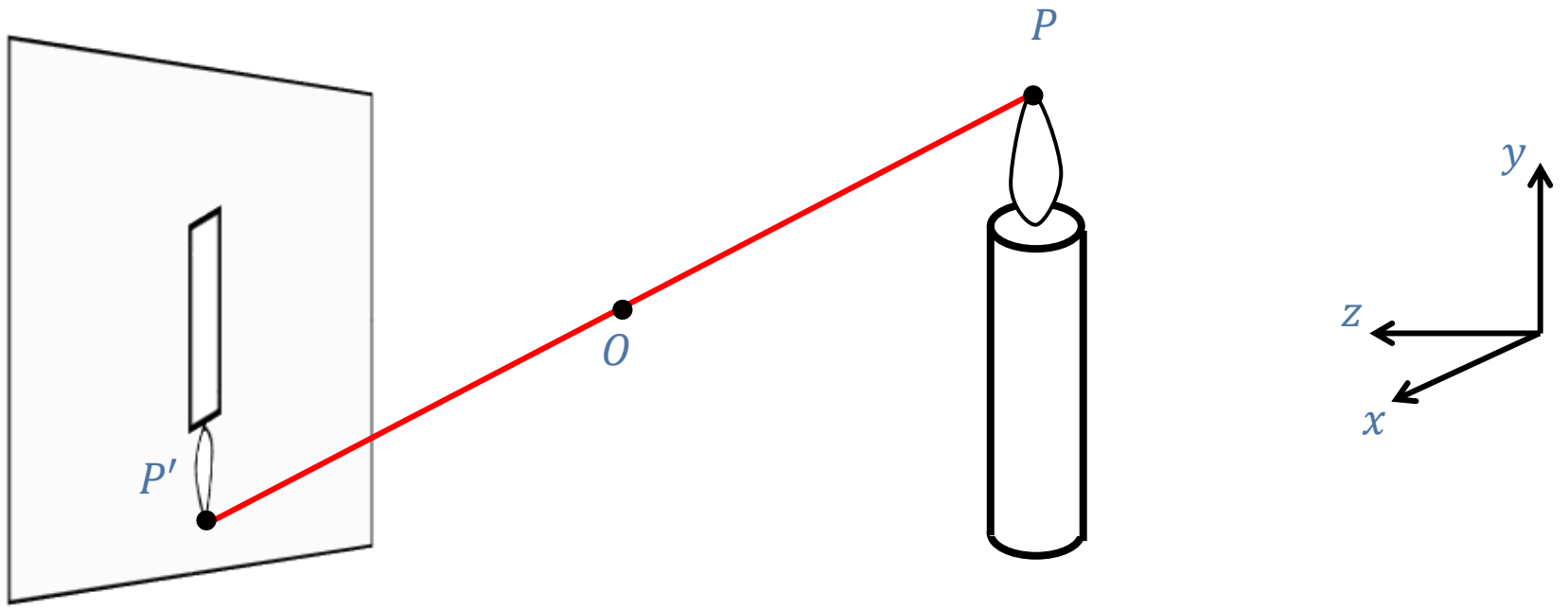
Image Formation – Camera Obscura



"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)

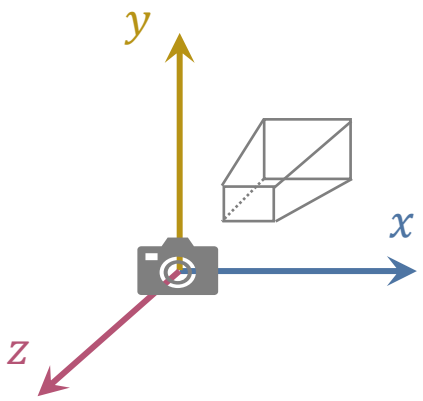
Pinhole Camera



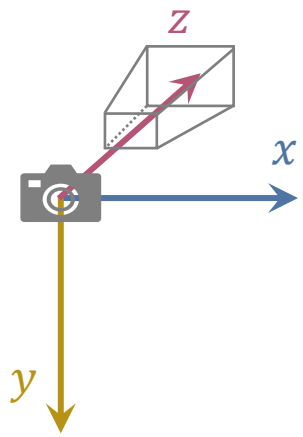
Canonical coordinate system

- 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

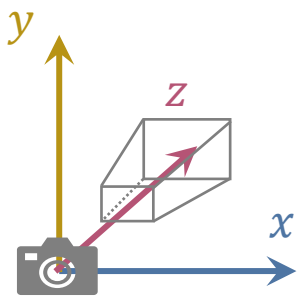
Everybody Agrees… Except on the Axes



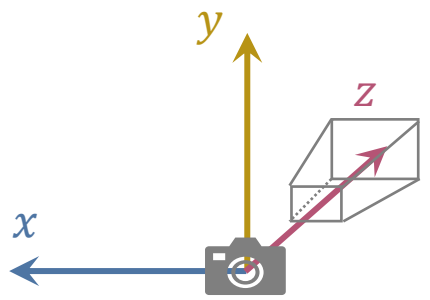
- OpenGL
- Blender
- ARKit
- Three.js
- Nerfstudio
- …



- OpenCV
- Open3D
- COLMAP
- gsplat
- …

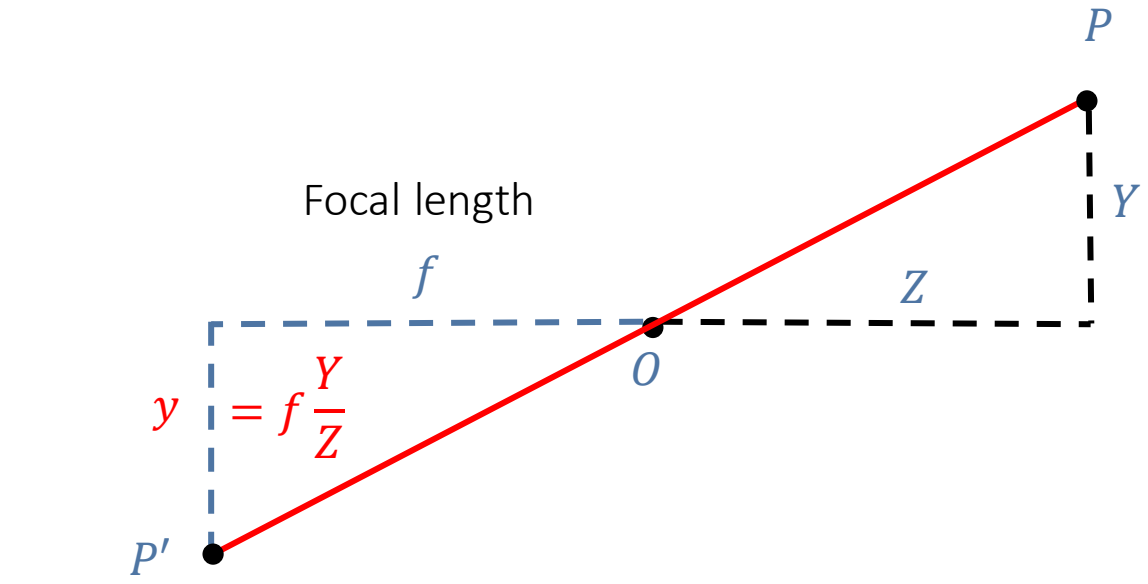


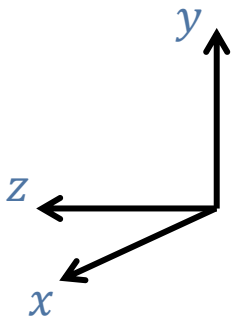
- Unity
- …



- PyTorch3D
- ?

Perspective Projection



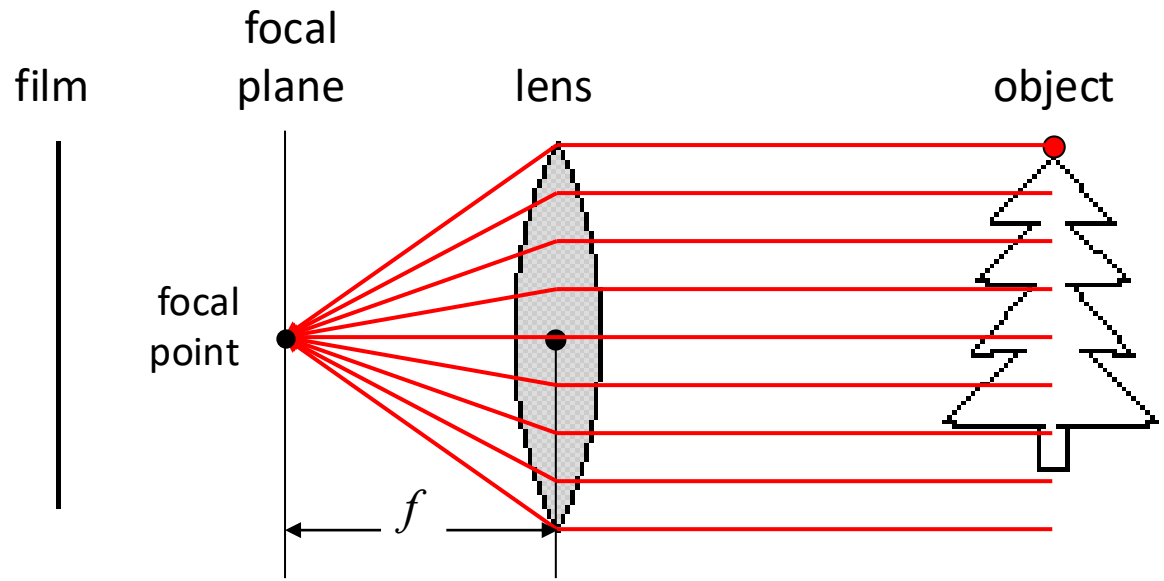


$$\left(f\frac{X}{Z}, f\frac{Y}{Z}\right) \leftarrow (X, Y, Z)$$

2D point 3D point

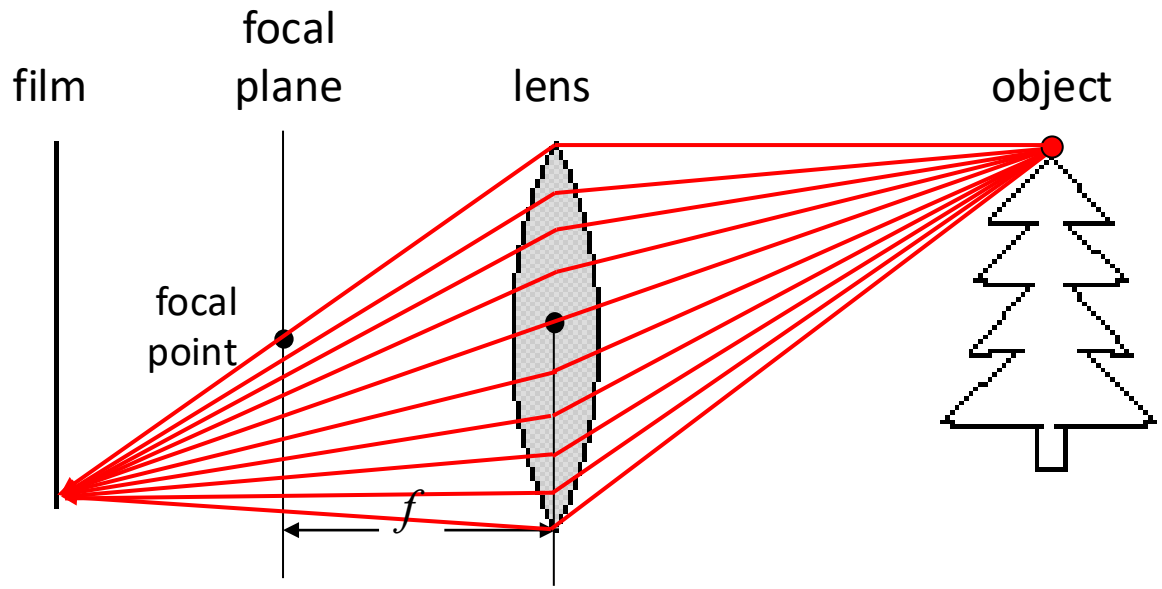
Focal Length of Lenses

- 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.



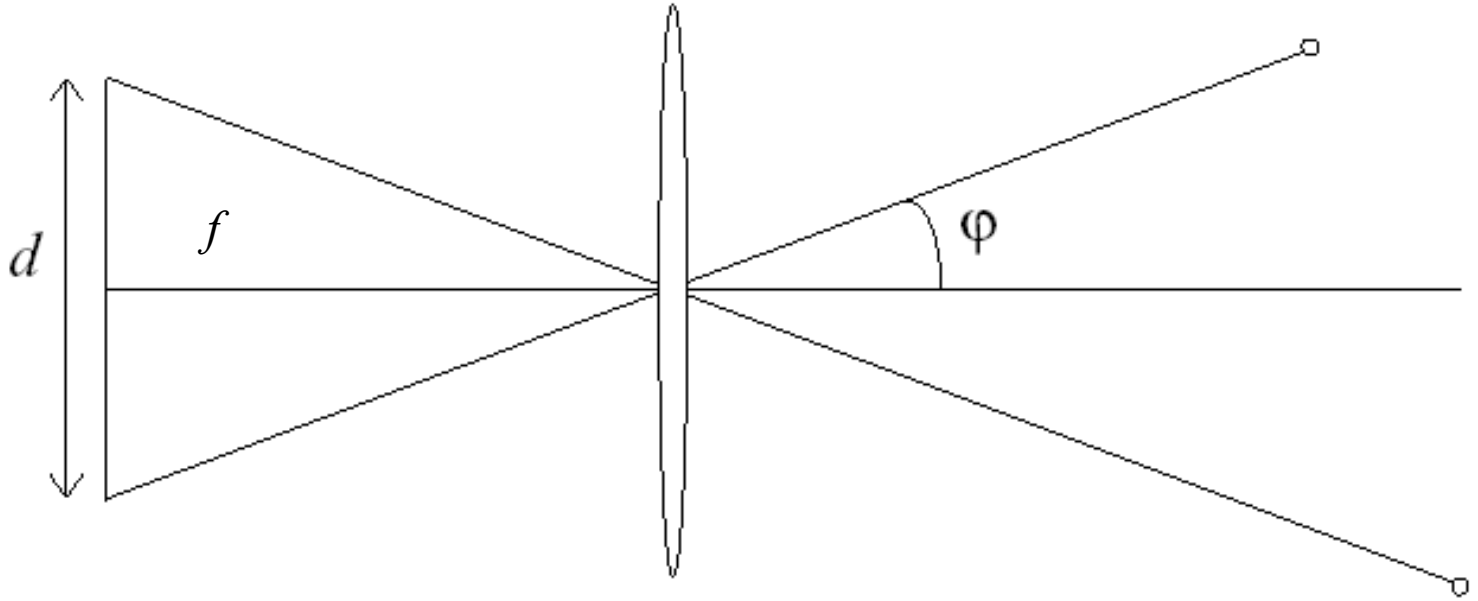
Focal Length of Lenses

- 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)

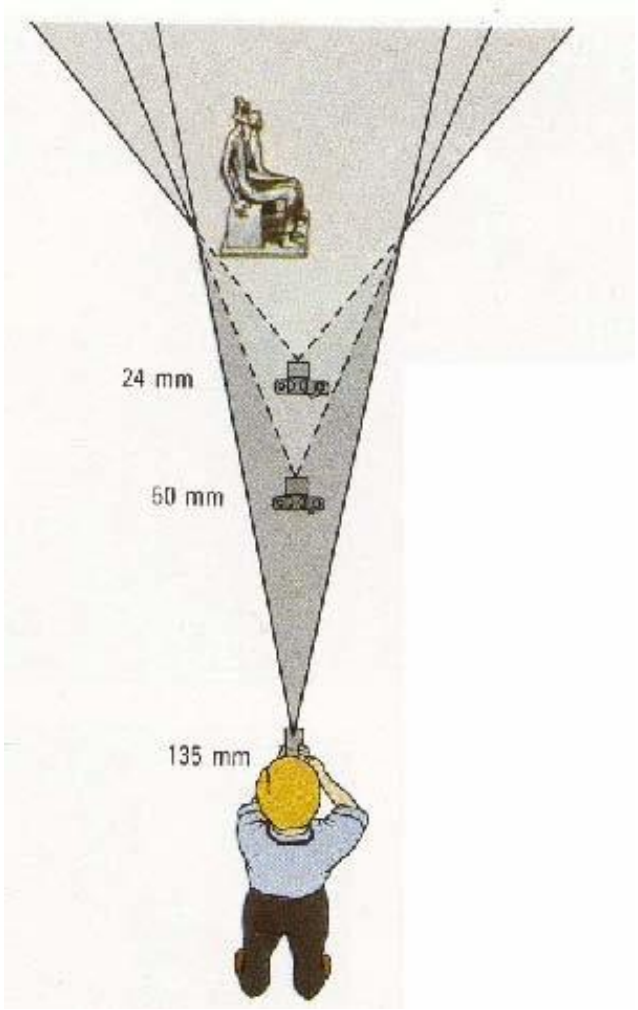
- The field of view is the angular extent of the world observed by the camera.
- Focal length (f), length of the sensor (d):



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

- 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

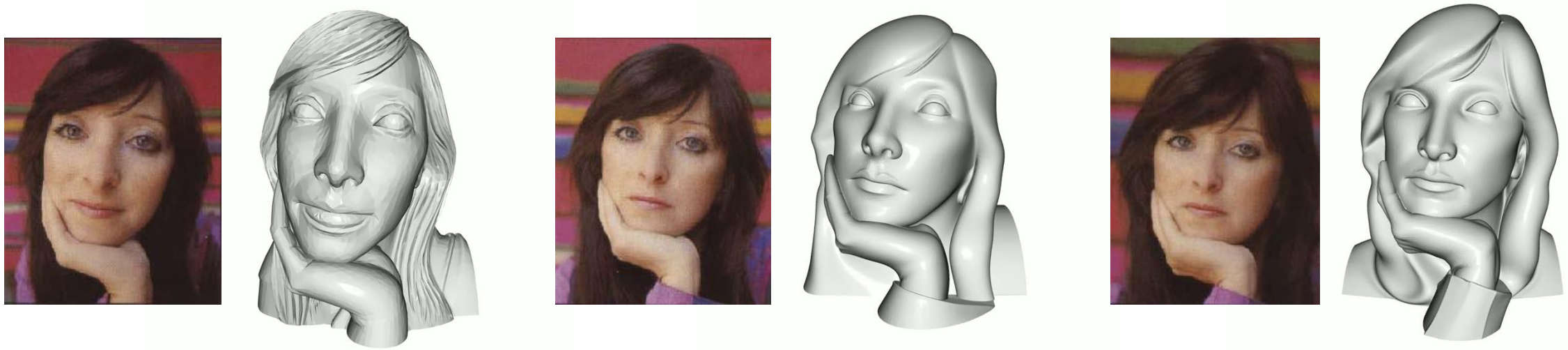
wide-angle

standard

telephoto

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

Field of View / Focal Length Ambiguity



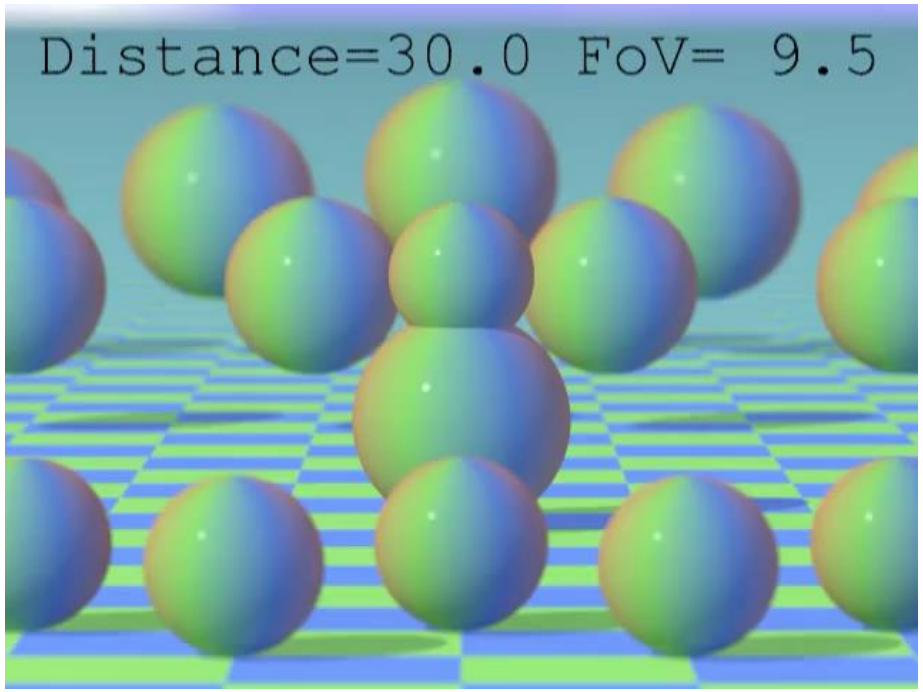
wide-angle standard telephoto

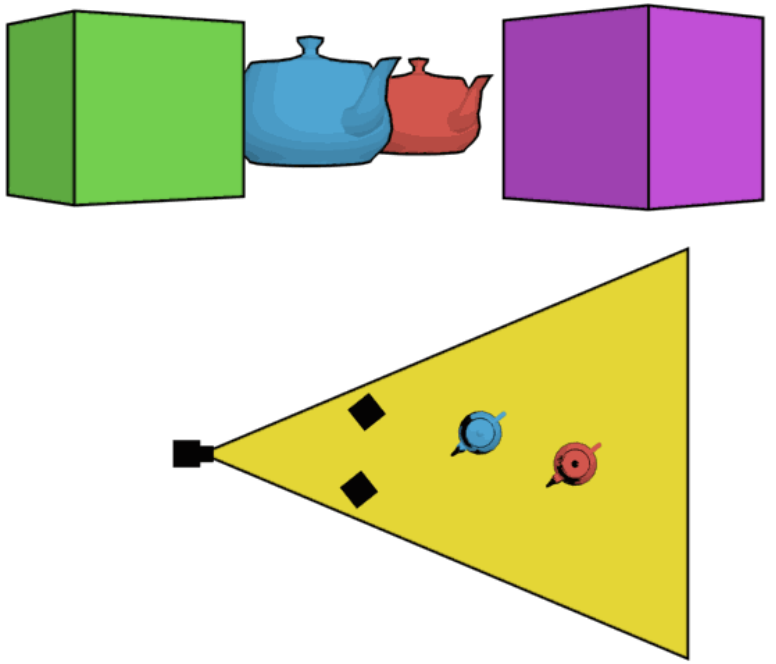
(reconstructed by Hunyuan3D V3.1)

- 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.

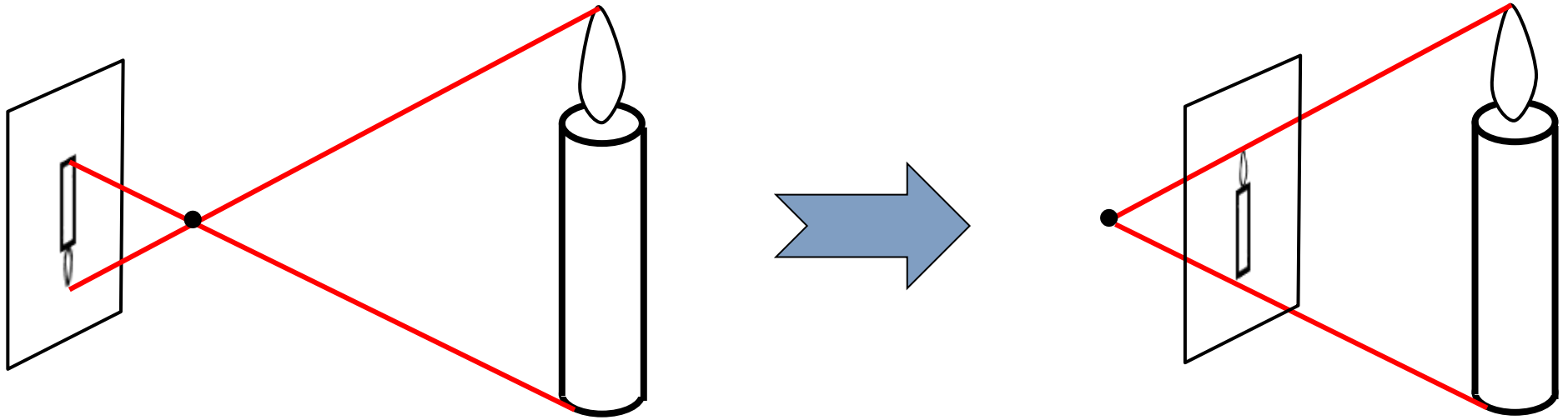




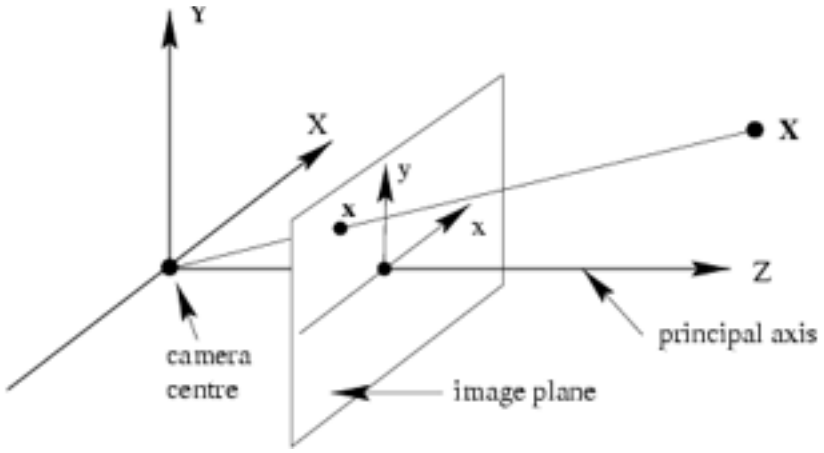
https://en.wikipedia.org/wiki/Dolly_zoom

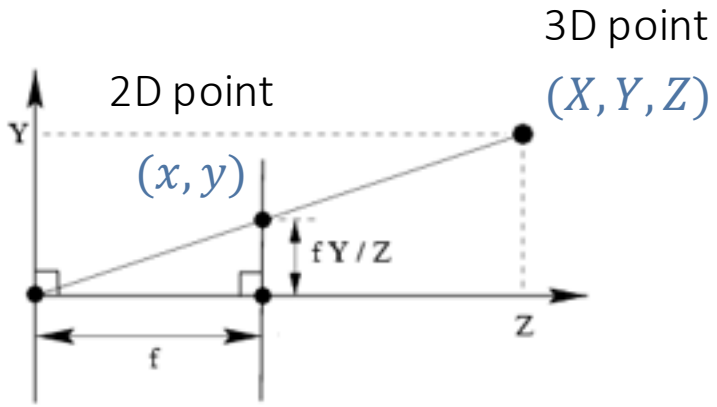
Perspective Projection

- 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:

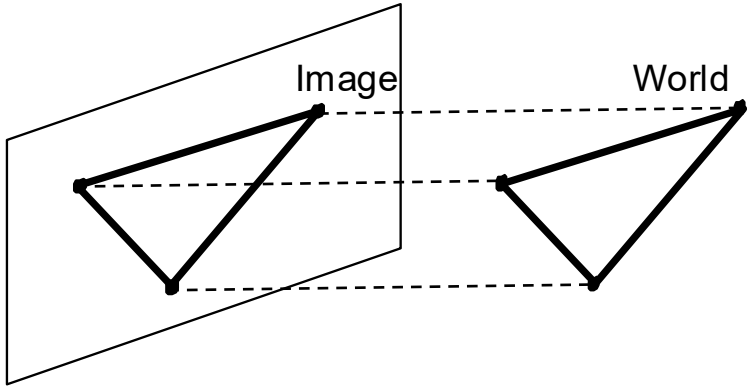
$$\begin{bmatrix} & & & \\ & & & \\ & & & \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \\ \\ \end{bmatrix} \Rightarrow \left(f\frac{X}{Z}, f\frac{Y}{Z} \right)$$

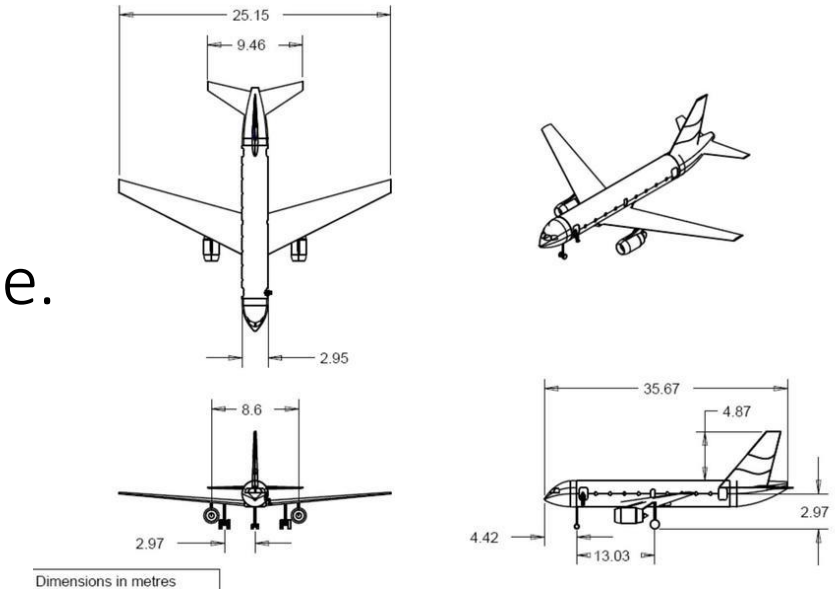
divide by the third coordinate

- 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.





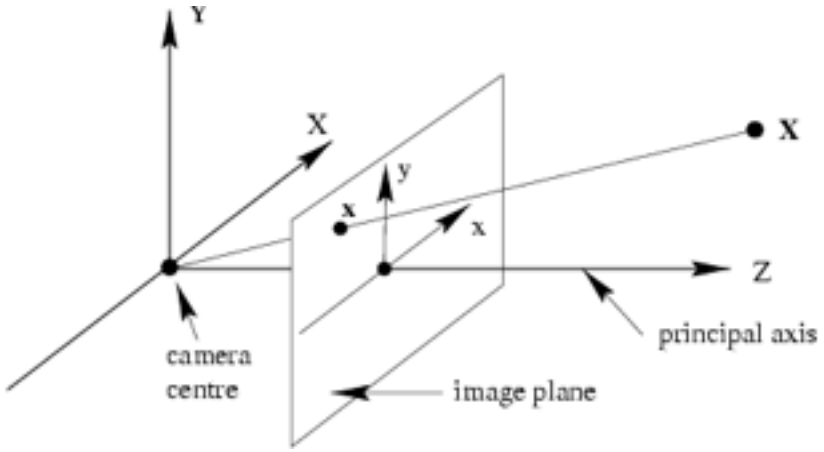
Three-view drawing

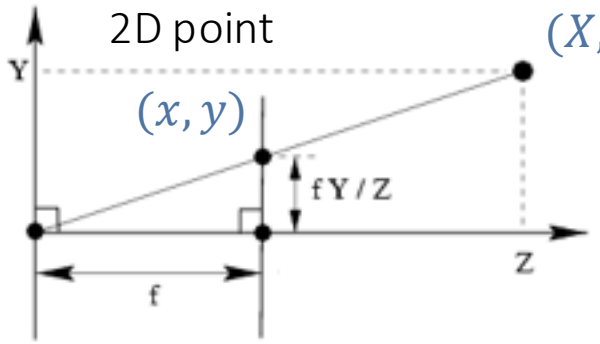
- Assuming projection along the z axis, what's the matrix?

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

Perspective Projection *in Normalized Coordinates*

(assuming camera looking toward z axis and no scaling)





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

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

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

$\boldsymbol{x}$: Homogeneous coordinates of image point

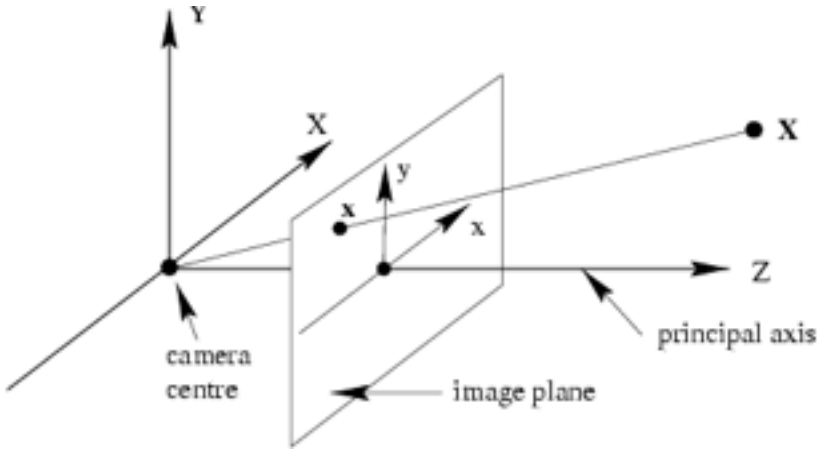
$\boldsymbol{P}$: Camera projection matrix

$\boldsymbol{X}$: Homogeneous coordinates of 3D point

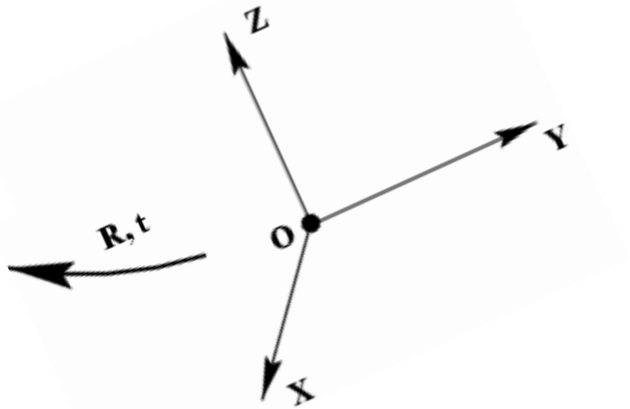
$$\boldsymbol{x \cong PX}$$

Equality up to scale

Camera Calibration



camera coordinate system



world coordinate system

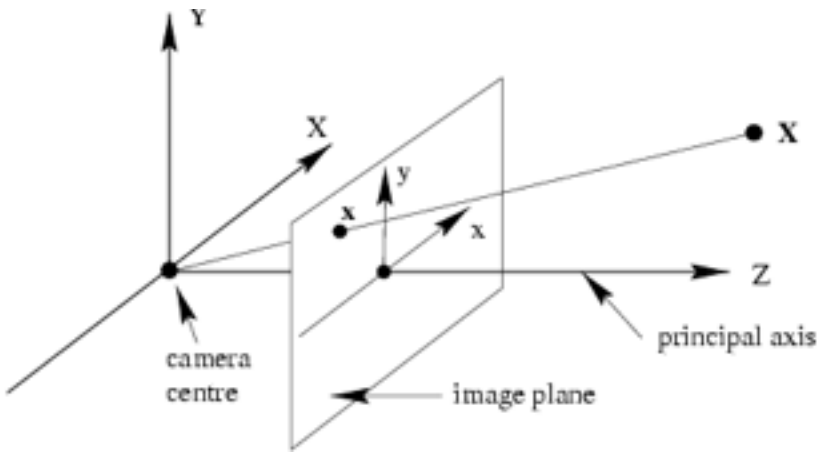
- **Camera calibration**: figuring out transformation from *world* coordinate system to *image* coordinate system

$$\begin{pmatrix} \text{2D point } \boldsymbol{x} \\ \text{(3x1)} \end{pmatrix} \cong \begin{pmatrix} \text{Camera to pixel coord. trans. matrix } \boldsymbol{K} \text{ (3x3)} \end{pmatrix} \begin{pmatrix} \textit{Canonical} \text{ projection matrix } [\boldsymbol{I} \mid \boldsymbol{0}] \text{ (3x4)} \end{pmatrix} \begin{pmatrix} \text{World to camera coord. trans. matrix } \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \text{ (4x4)} \end{pmatrix} \begin{pmatrix} \text{3D point } \boldsymbol{X} \\ \text{(4x1)} \end{pmatrix}$$

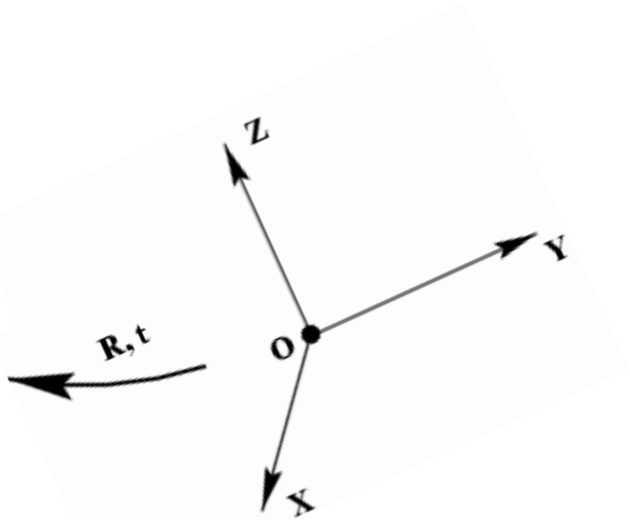
***Intrinsic* camera parameters**: focal length, principal point, scaling factors

***Extrinsic* camera parameters**: rotation, translation

Camera Calibration



camera coordinate system



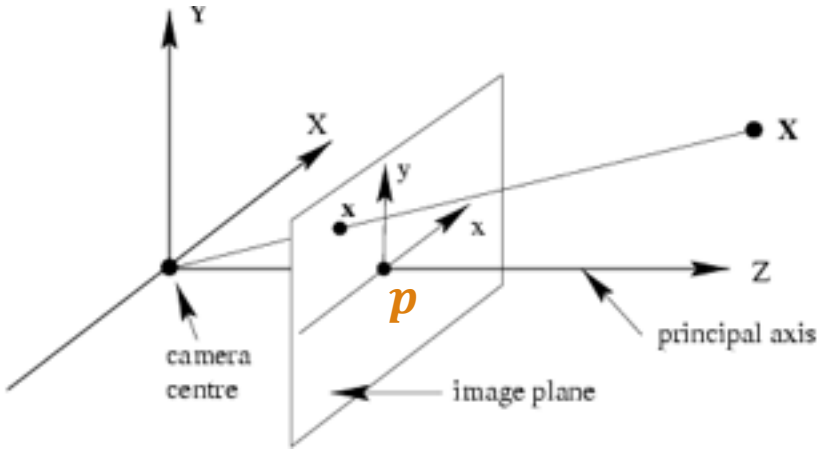
world coordinate system

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

$$\begin{bmatrix} \text{2D point} \\ \boldsymbol{x} \\ \text{(3x1)} \end{bmatrix} \cong \underbrace{\begin{bmatrix} \text{Camera to pixel coord. trans. matrix} \\ \boldsymbol{K} \text{ (3x3)} \end{bmatrix} \begin{bmatrix} \textit{Canonical} \\ \text{projection matrix} \\ [\boldsymbol{I} \mid \boldsymbol{0}] \text{ (3x4)} \end{bmatrix} \begin{bmatrix} \text{World to} \\ \text{camera coord.} \\ \text{trans. matrix} \\ \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \text{ (4x4)} \end{bmatrix}}_{\boldsymbol{P} = \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}]} \begin{bmatrix} \text{3D point} \\ \boldsymbol{X} \\ \text{(4x1)} \end{bmatrix}$$

General camera projection matrix

Intrinsic Parameters – Principal Point



camera coordinate system

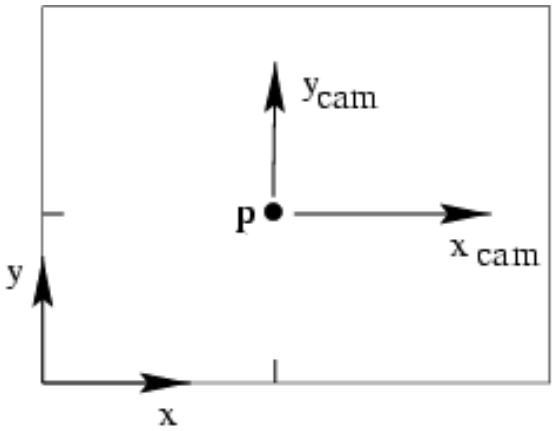
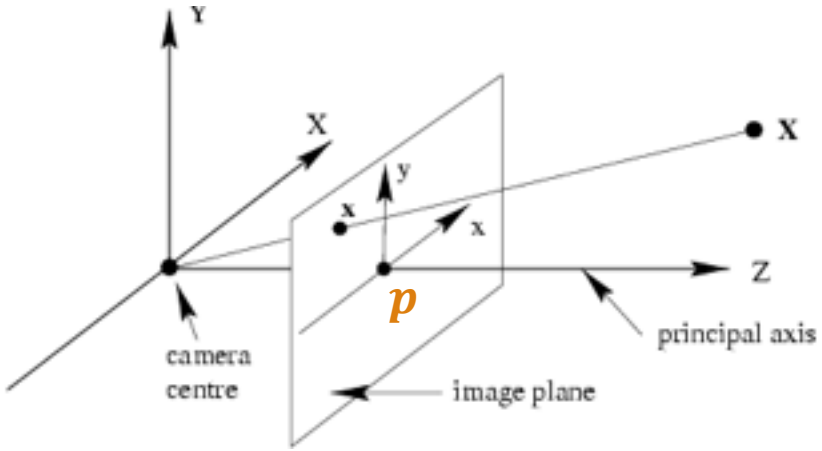


image coordinate system

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

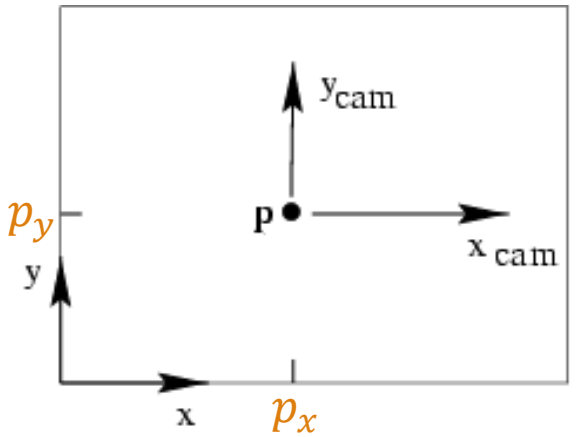


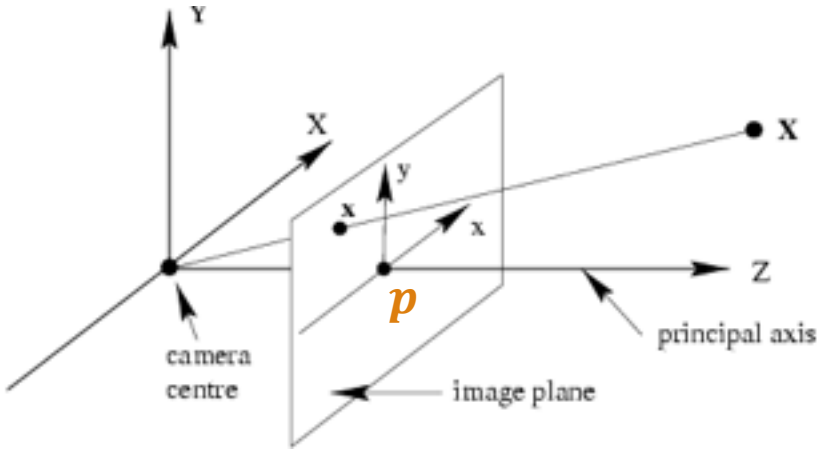
image coordinate system

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

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

calibration matrix $\boldsymbol{K}$

Intrinsic Parameters – Scaling



camera coordinate system

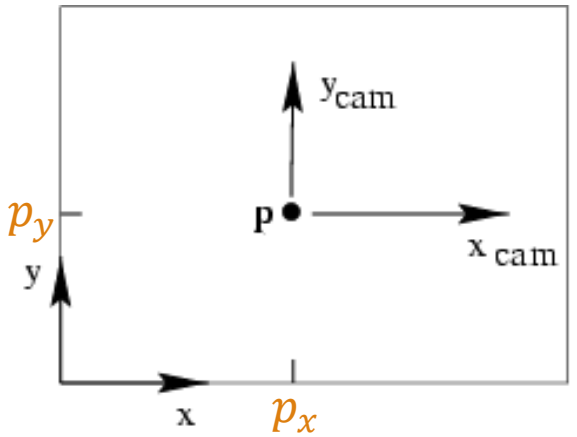


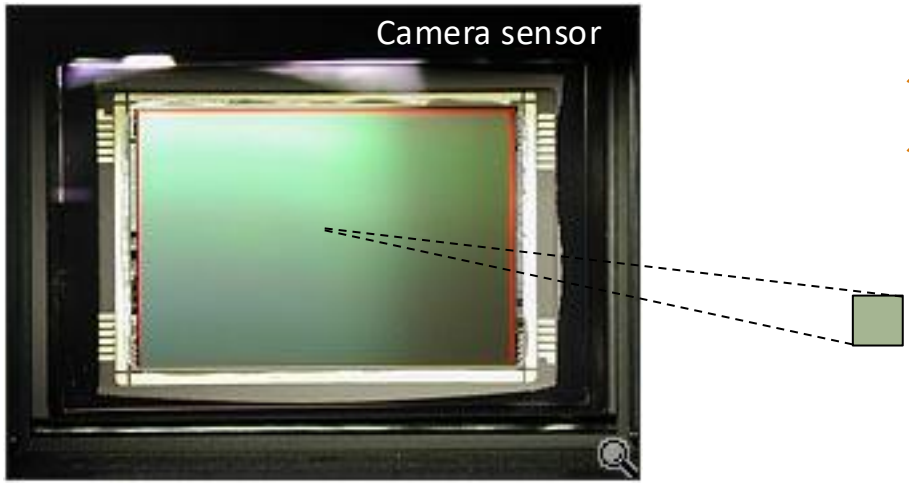
image coordinate system

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

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

pixels — pixels/m — m — calibration matrix $\boldsymbol{K}$

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 $\boldsymbol{K}$ in metric units

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

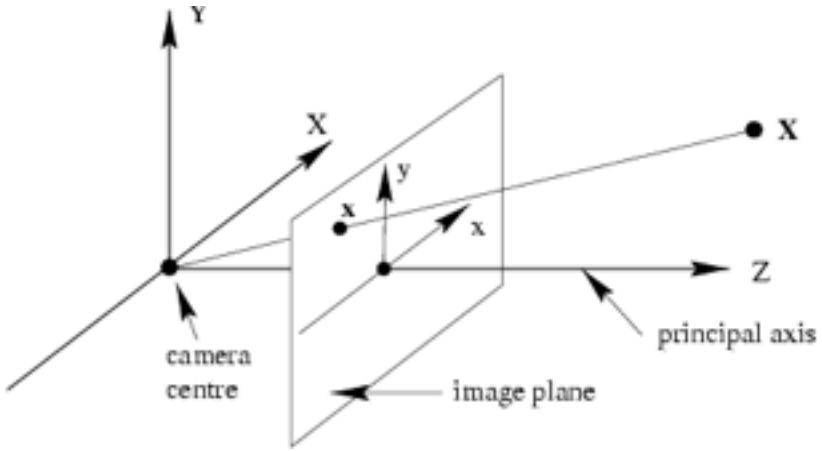
m

Calibration matrix $\boldsymbol{K}$ in pixel units

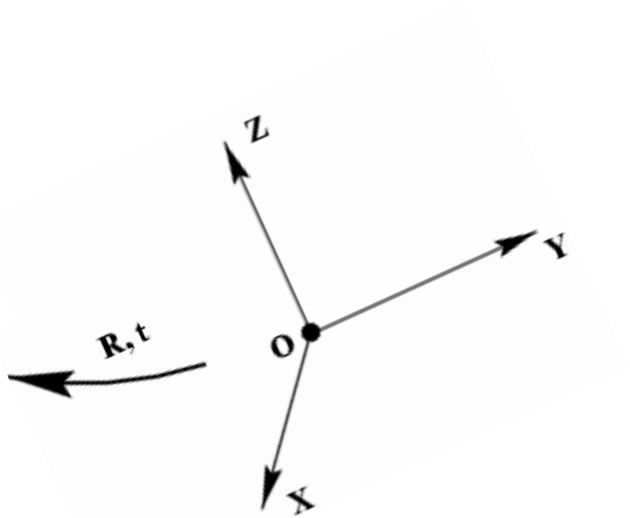
$$= \begin{bmatrix} \alpha_x & 0 & \beta_x \\ 0 & \alpha_y & \beta_y \\ 0 & 0 & 1 \end{bmatrix}$$

pixels

Camera Calibration



camera coordinate system



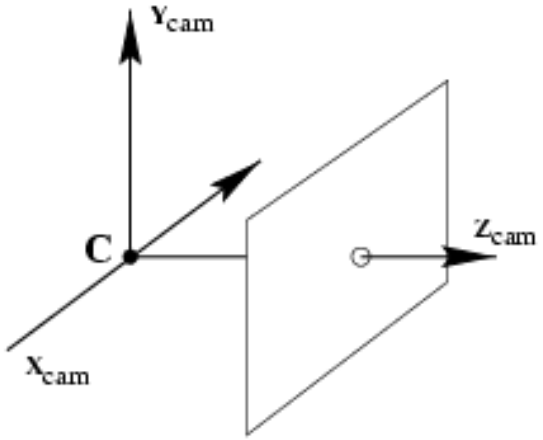
world coordinate system

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

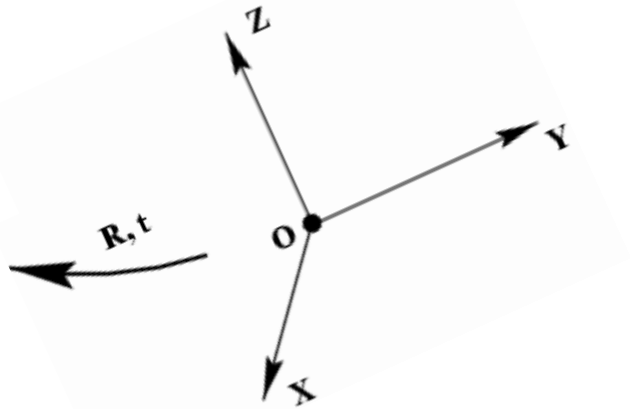
$$\begin{pmatrix} \text{2D point} \\ \boldsymbol{x} \\ \text{(3x1)} \end{pmatrix} \cong \underbrace{\begin{pmatrix} \text{Camera to pixel coord. trans. matrix} \\ \boldsymbol{K} \text{ (3x3)} \end{pmatrix} \begin{pmatrix} \textit{Canonical} \\ \text{projection matrix} \\ [\boldsymbol{I} \mid \boldsymbol{0}] \text{ (3x4)} \end{pmatrix} \begin{pmatrix} \text{World to} \\ \text{camera coord.} \\ \text{trans. matrix} \\ \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \text{ (4x4)} \end{pmatrix}}_{\boldsymbol{P} = \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}]} \begin{pmatrix} \text{3D point} \\ \boldsymbol{X} \\ \text{(4x1)} \end{pmatrix}$$

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:

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

coords. of point in normalized camera frame

3x3 rotation matrix

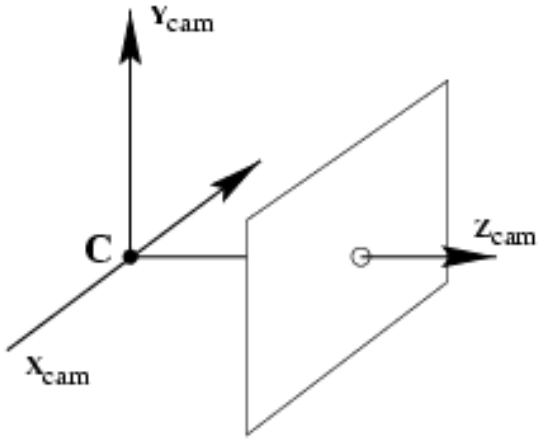
coords. of a point in world frame

coords. of camera center in world frame

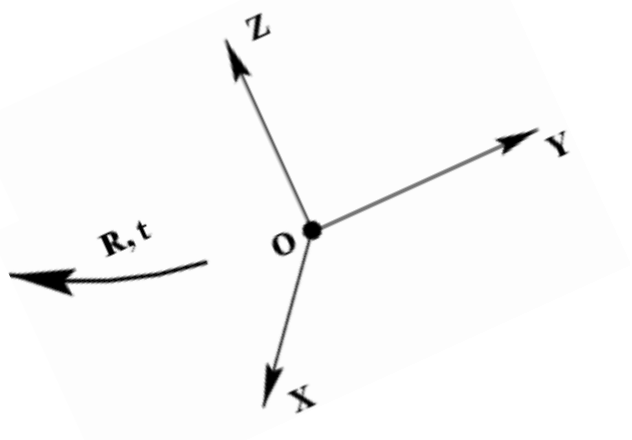
W2C: $\tilde{\boldsymbol{X}}_{\text{cam}} = \boldsymbol{R}\tilde{\boldsymbol{X}}_{\text{world}} + \boldsymbol{t}$

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

Extrinsic Parameters



camera coordinate system



world coordinate system

In non-homogeneous coordinates:

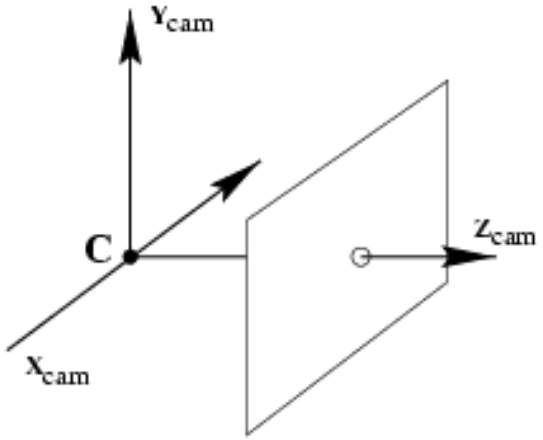
$$\widetilde{\boldsymbol{X}}_{\mathrm{cam}} = \boldsymbol{R}\widetilde{\boldsymbol{X}} + \boldsymbol{t}$$

In homogeneous coordinates:

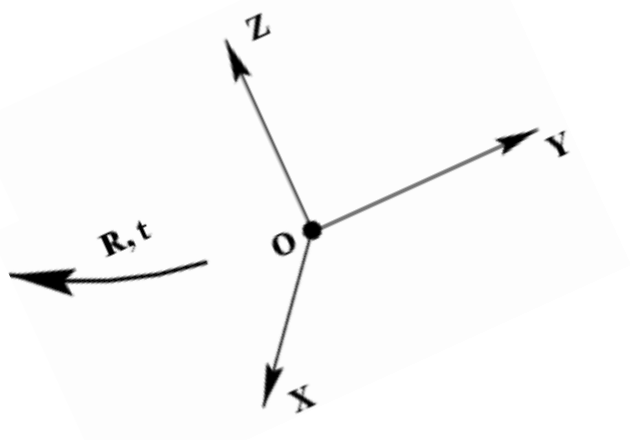
$$\begin{pmatrix} \widetilde{\boldsymbol{X}}_{\mathrm{cam}} \\ 1 \end{pmatrix} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^{\boldsymbol{T}} & 1 \end{bmatrix} \begin{pmatrix} \widetilde{\boldsymbol{X}} \\ 1 \end{pmatrix}$$

3D transformation matrix (4 x 4)

Extrinsic Parameters



camera coordinate system



world coordinate system

In non-homogeneous coordinates:

$$\widetilde{\boldsymbol{X}}_{\mathrm{cam}} = \boldsymbol{R}\widetilde{\boldsymbol{X}} + \boldsymbol{t}$$

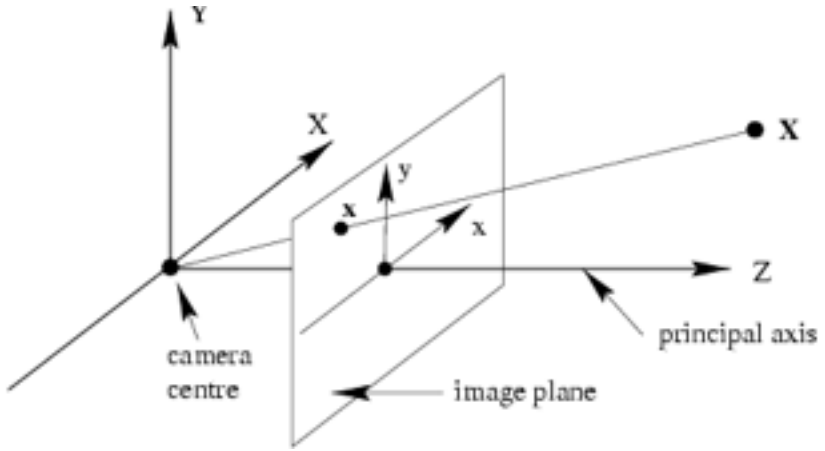
In homogeneous coordinates:

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

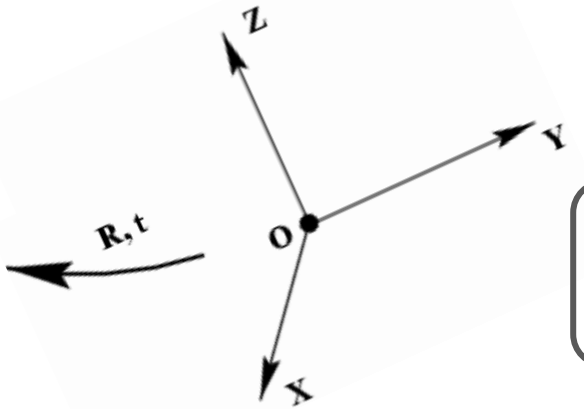
3D transformation matrix (4 x 4)

And then to image coordinates: $\boldsymbol{x} \cong \boldsymbol{K}[\boldsymbol{I}|\boldsymbol{0}]\boldsymbol{X}_{\mathrm{cam}}$

Camera Calibration



camera coordinate system



world coordinate system

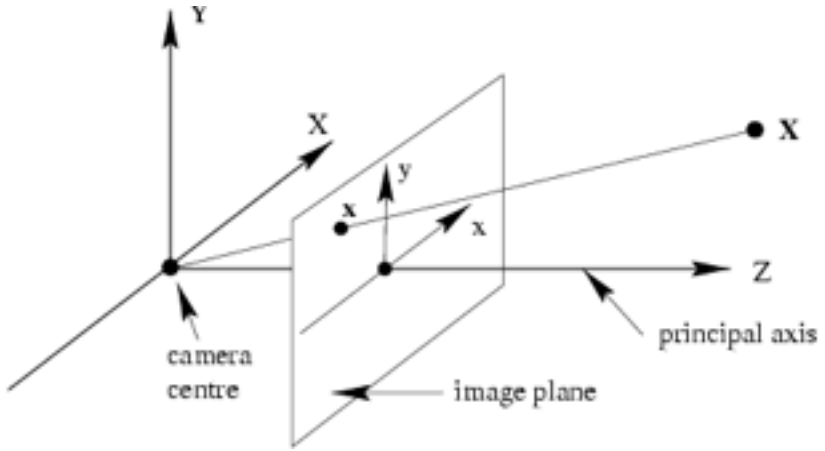
$$x \cong K[R|t]X$$

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

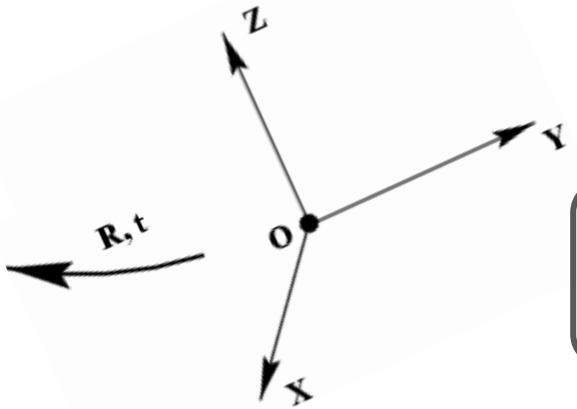
$$\begin{pmatrix} \text{2D point} \\ x \\ \text{(3x1)} \end{pmatrix} \cong \underbrace{\begin{pmatrix} \text{Camera to pixel coord. trans. matrix} \\ K \text{ (3x3)} \end{pmatrix} \begin{pmatrix} \textit{Canonical} \\ \text{projection matrix} \\ [I \mid 0] \text{ (3x4)} \end{pmatrix} \begin{pmatrix} \text{World to camera coord. trans. matrix} \\ \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \text{ (4x4)} \end{pmatrix}}_{P = K[R|t]} \begin{pmatrix} \text{3D point} \\ X \\ \text{(4x1)} \end{pmatrix}$$

General camera projection matrix

Camera Calibration



camera coordinate system



world coordinate system

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

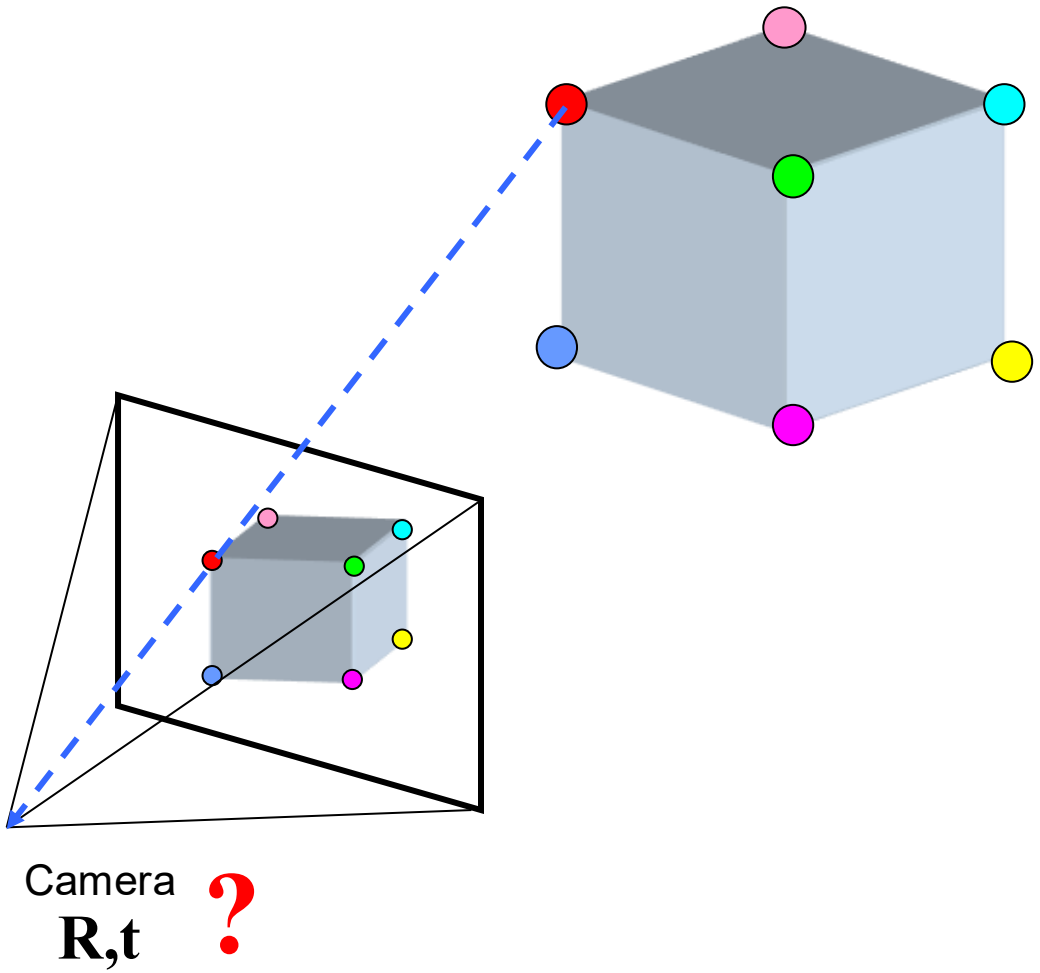
- **Camera calibration**: figuring out transformation from *world* coordinate system to *image* coordinate system

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

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

Perspective-n-Point (PnP)

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



Known:

- *n* 3D points X
- Corresponding 2D pixel coordinates x
- Camera intrinsics K

Solve for:

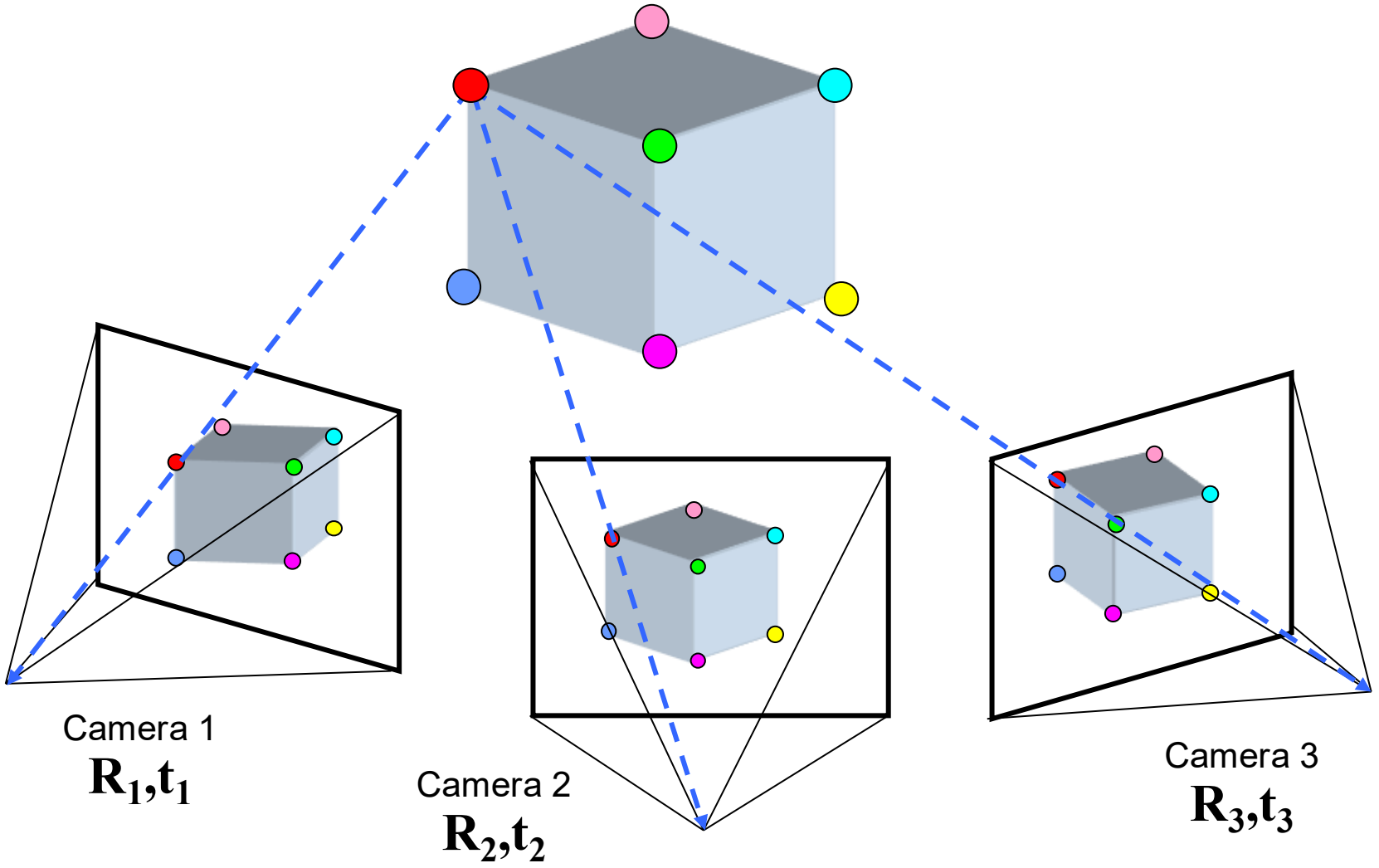
- Camera pose (extrinsics) R, t

Q: How many points at least do we need?

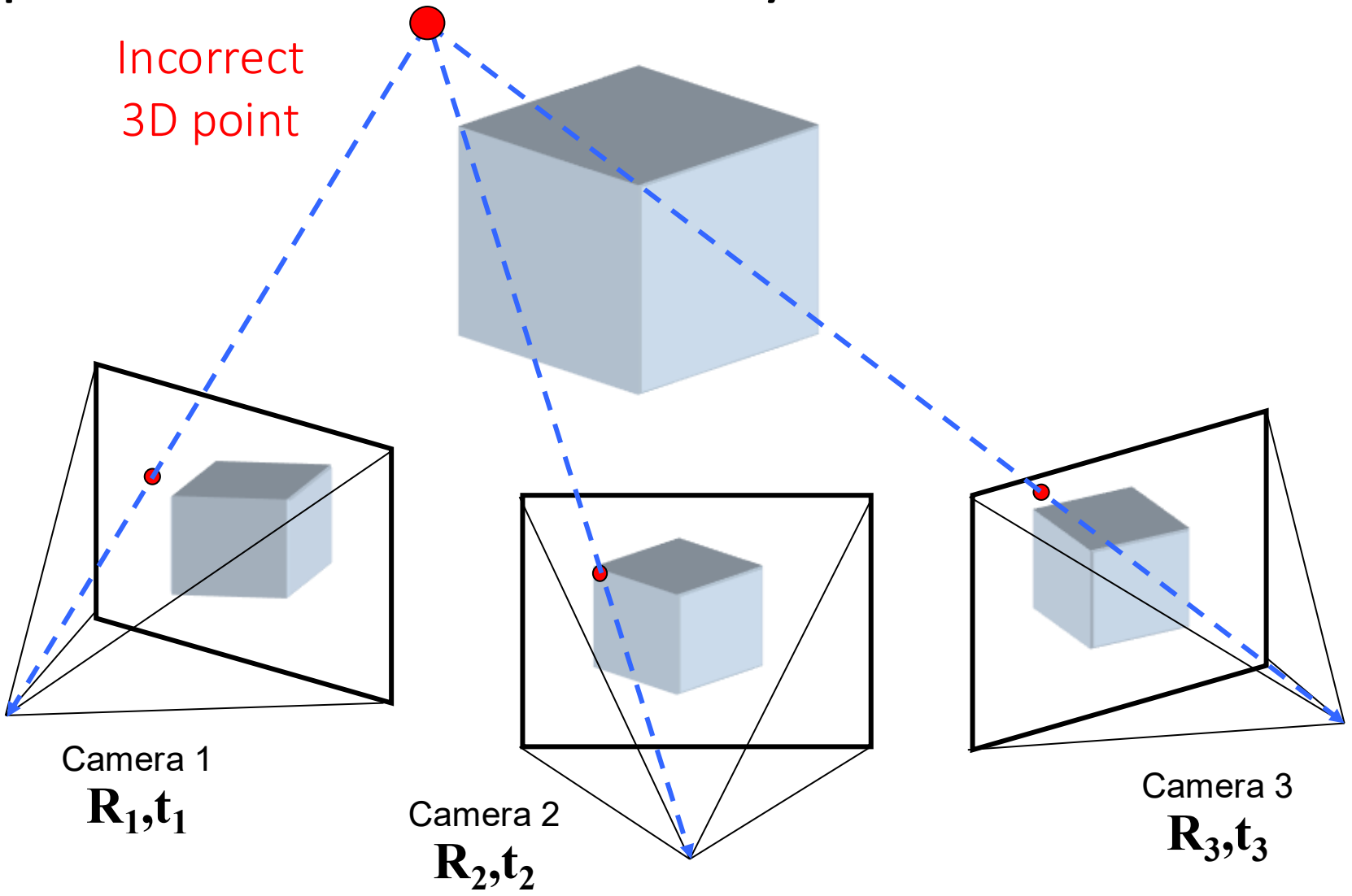
A: n ≥ 3 (in general). We have 6 DoF unknowns R, t. Each point provides 2 constraints (2 equations for x and y).

Part 2 – Multi-view Geometry

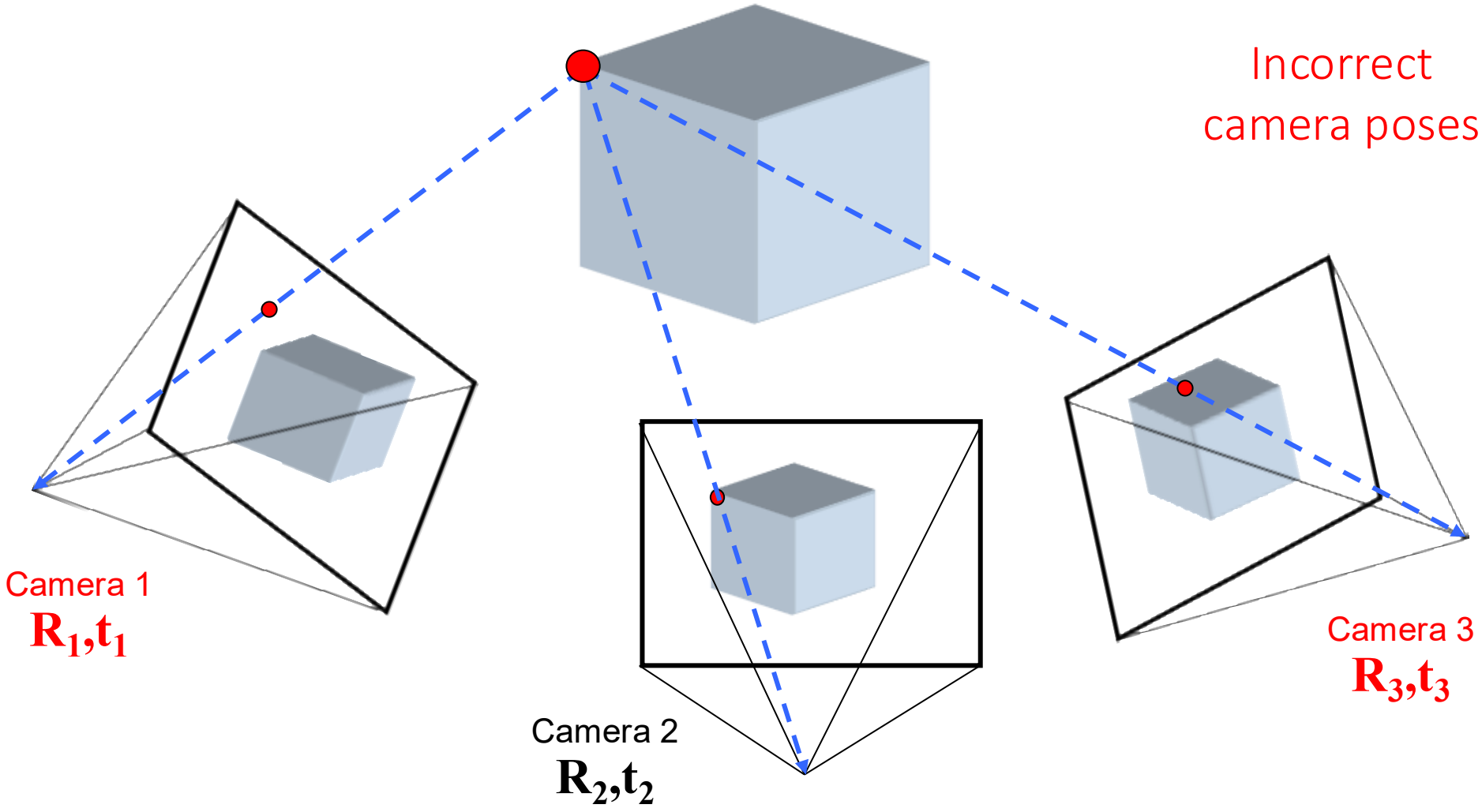
Multiple View Geometry



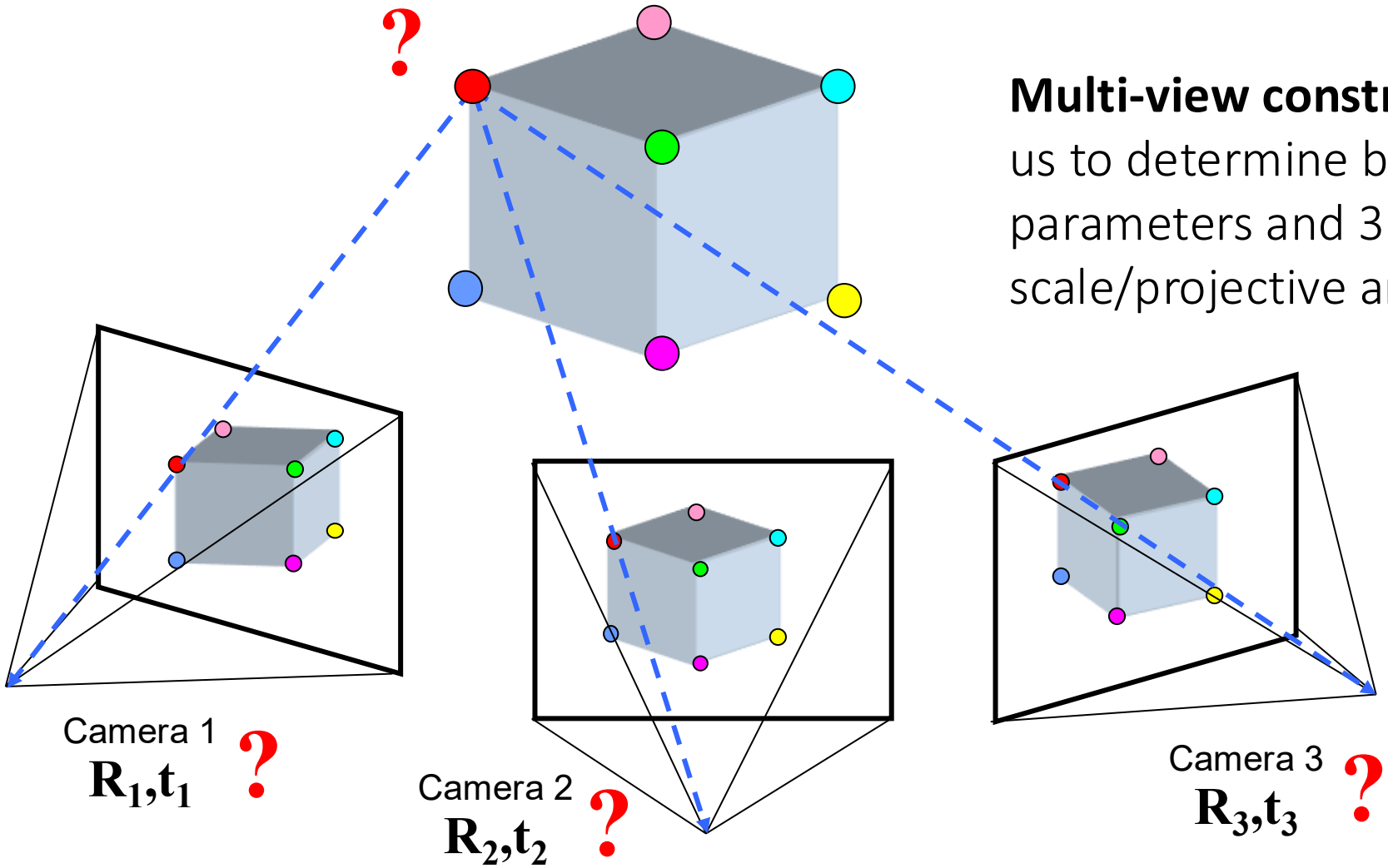
Multiple View Geometry



Multiple View Geometry

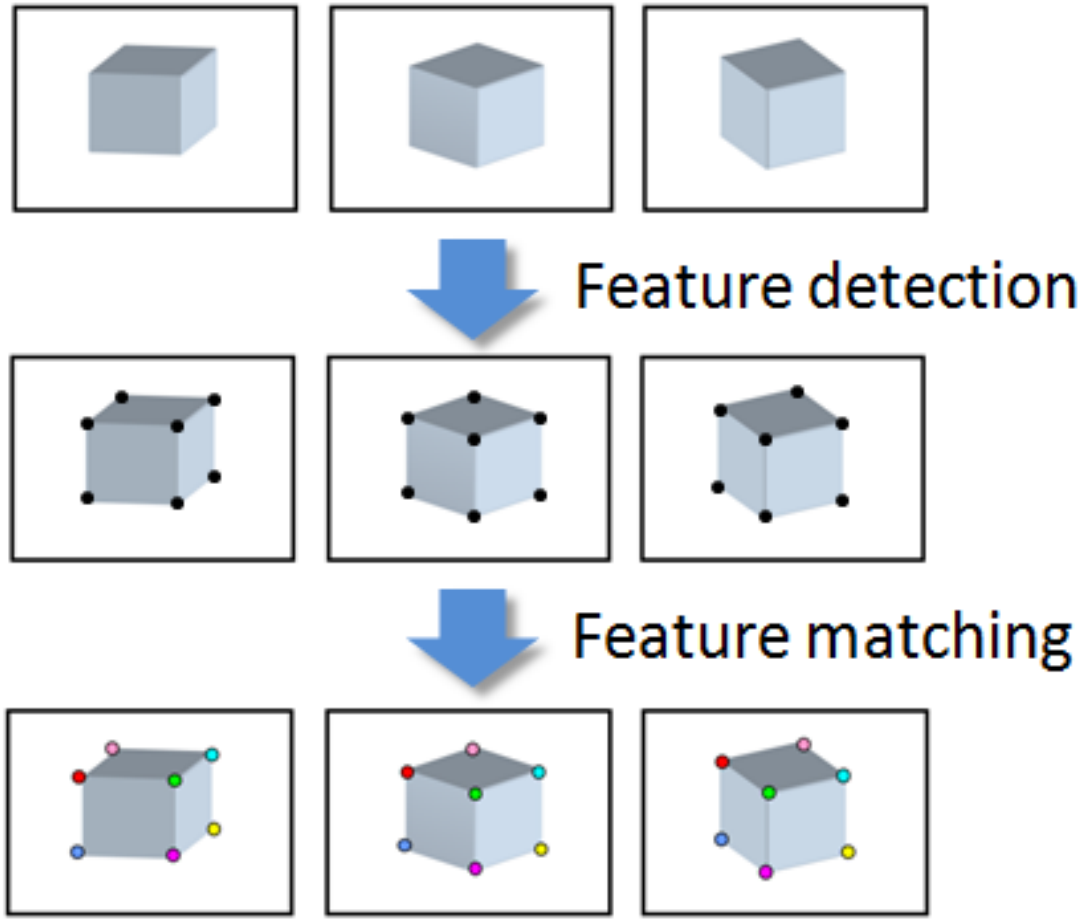


Multiple View Geometry

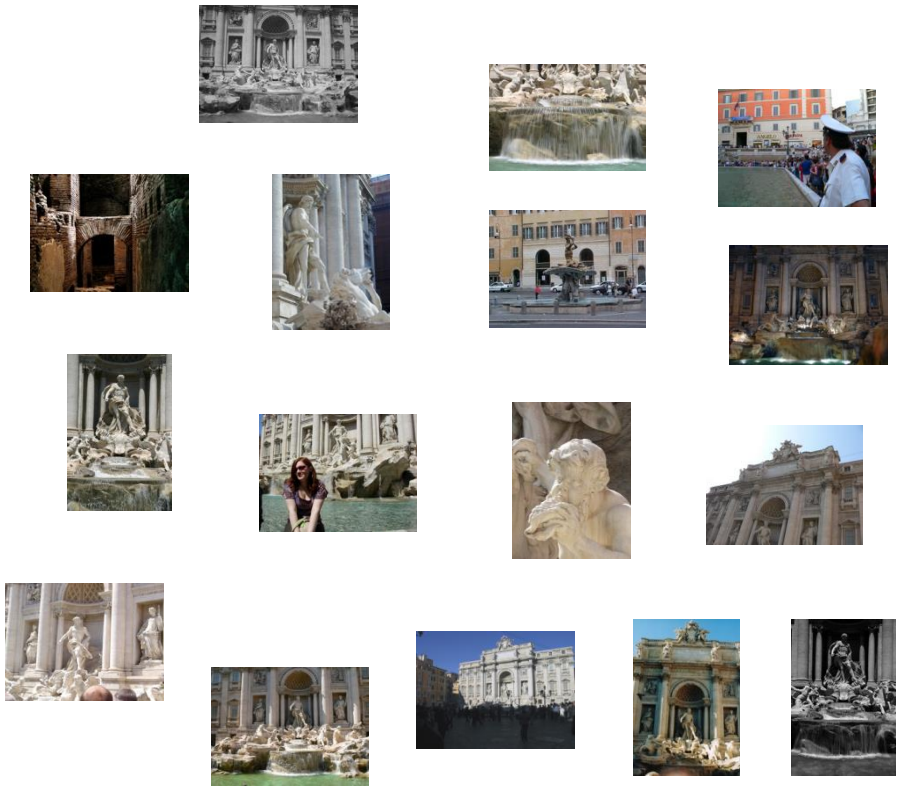


Multi-view constraints allow us to determine both camera parameters and 3D (up to scale/projective ambiguities)

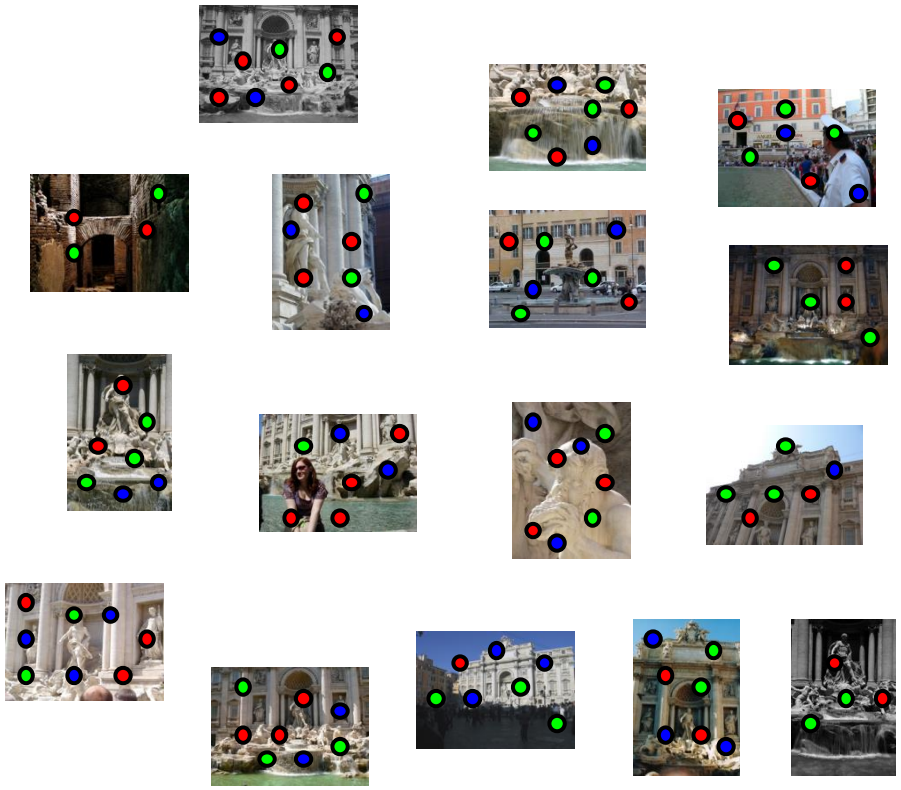
Estimating Correspondences



Estimating Correspondences

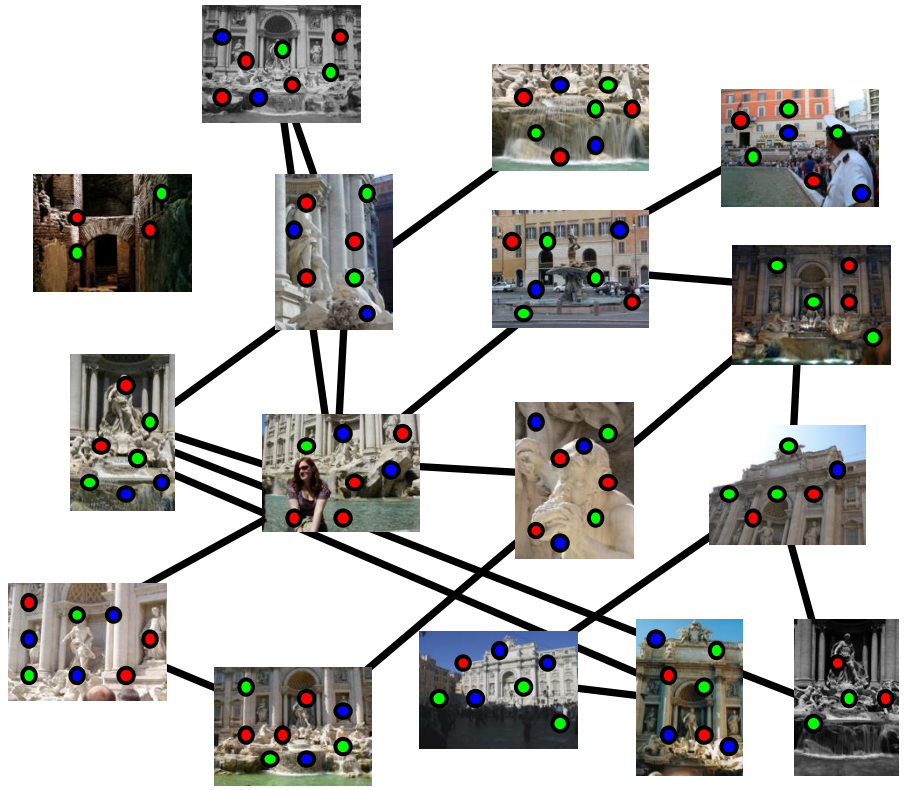


Estimating Correspondences



- **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]

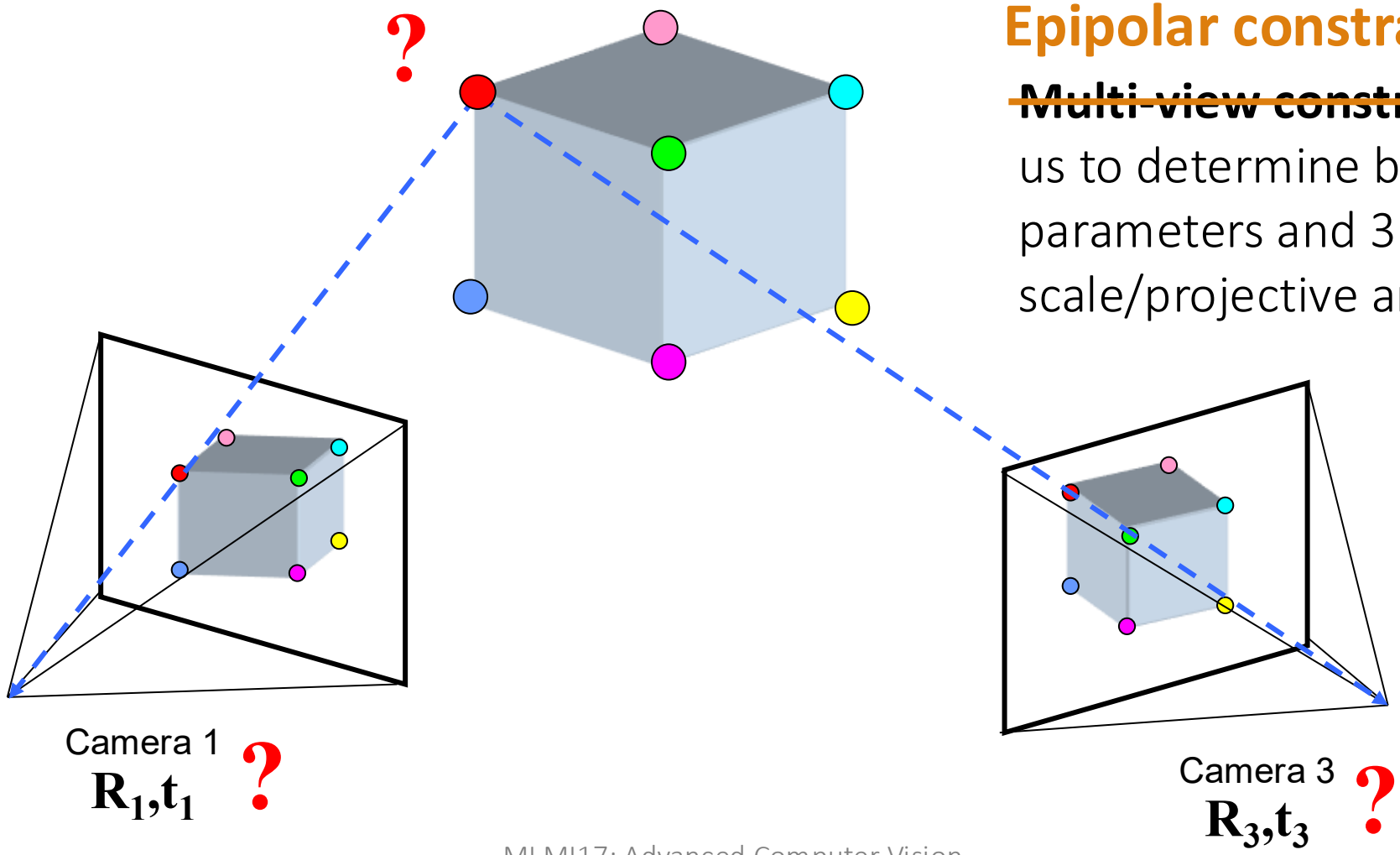
Student Feedback Survey

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

Let's Start with Two Views – Epipolar Geometry

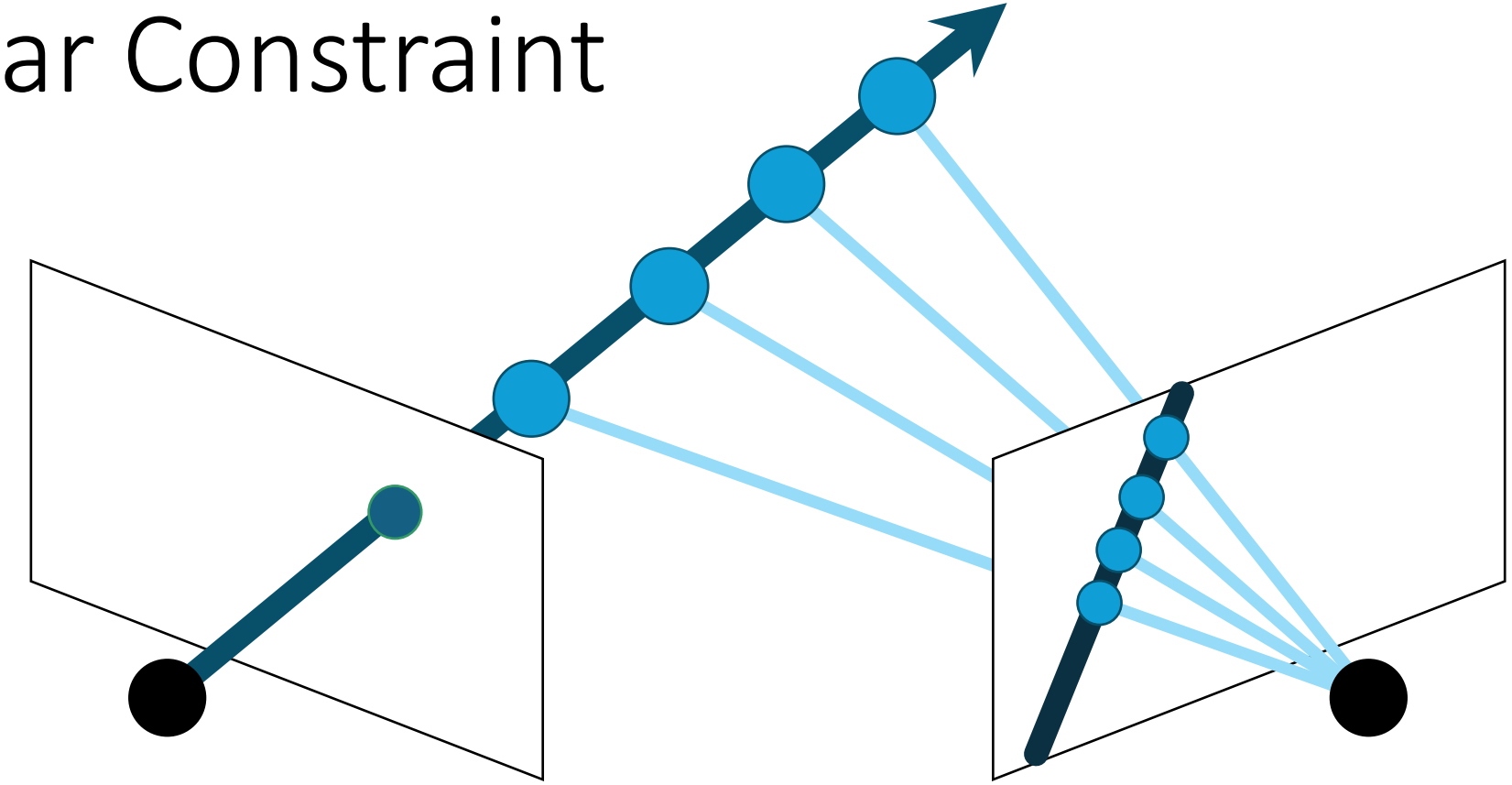
"epi-pole" ≈ upon the pole/pivot



Epipolar constraints

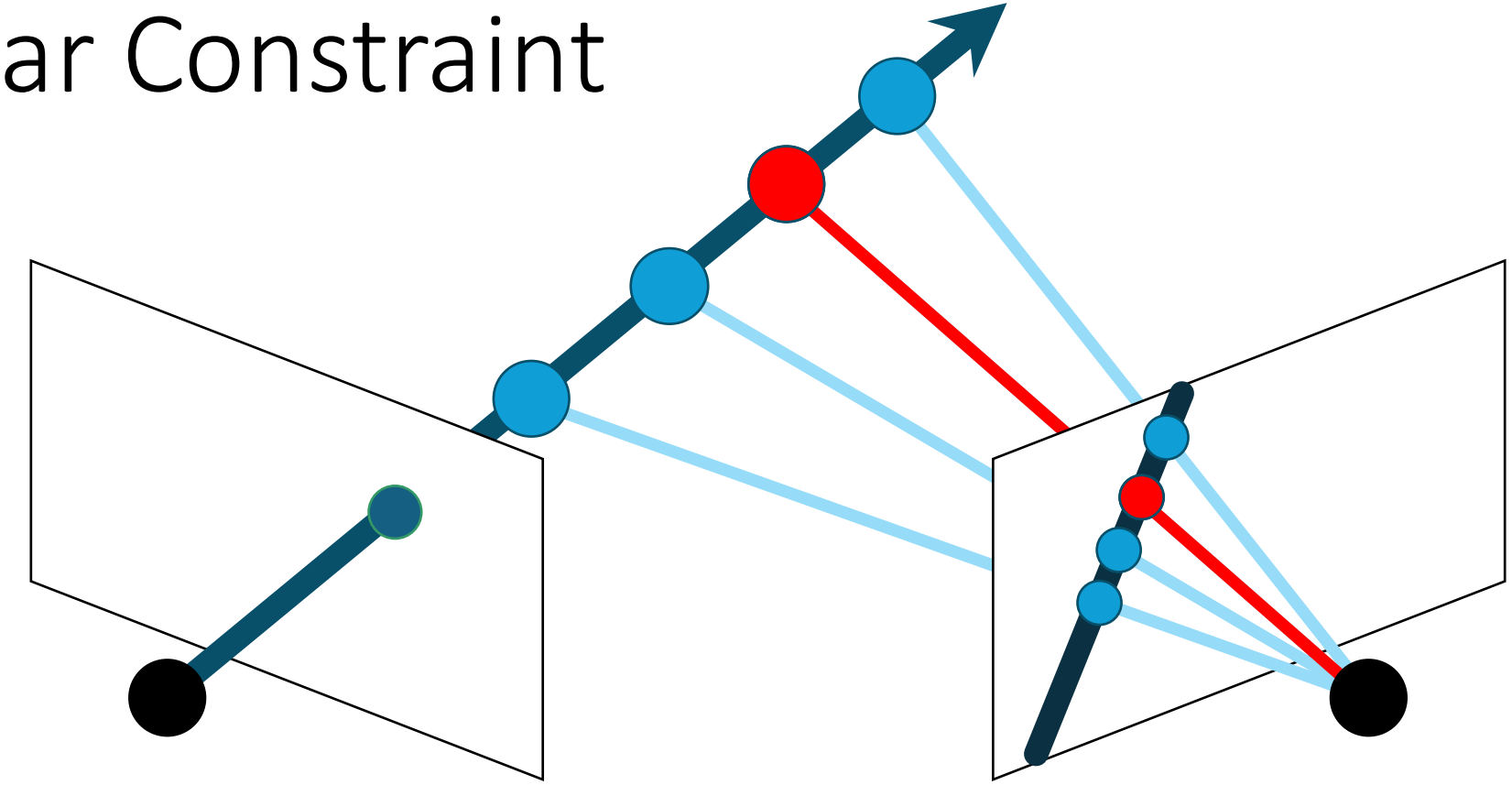
~~**Multi-view constraints**~~ allow us to determine both camera parameters and 3D (up to scale/projective ambiguities)

Epipolar Constraint



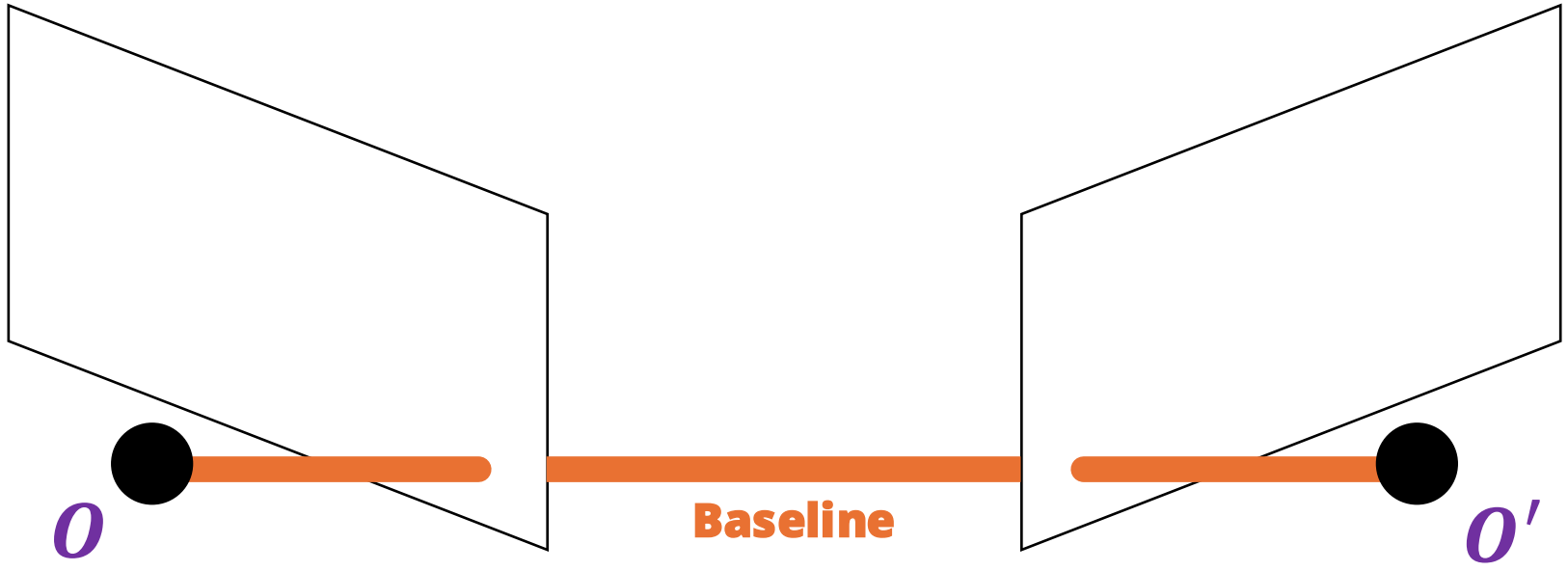
- Projecting points on a ray to another camera forms a **line**, i.e., the corresponding point on the second image *must* lie on this line

Epipolar Constraint



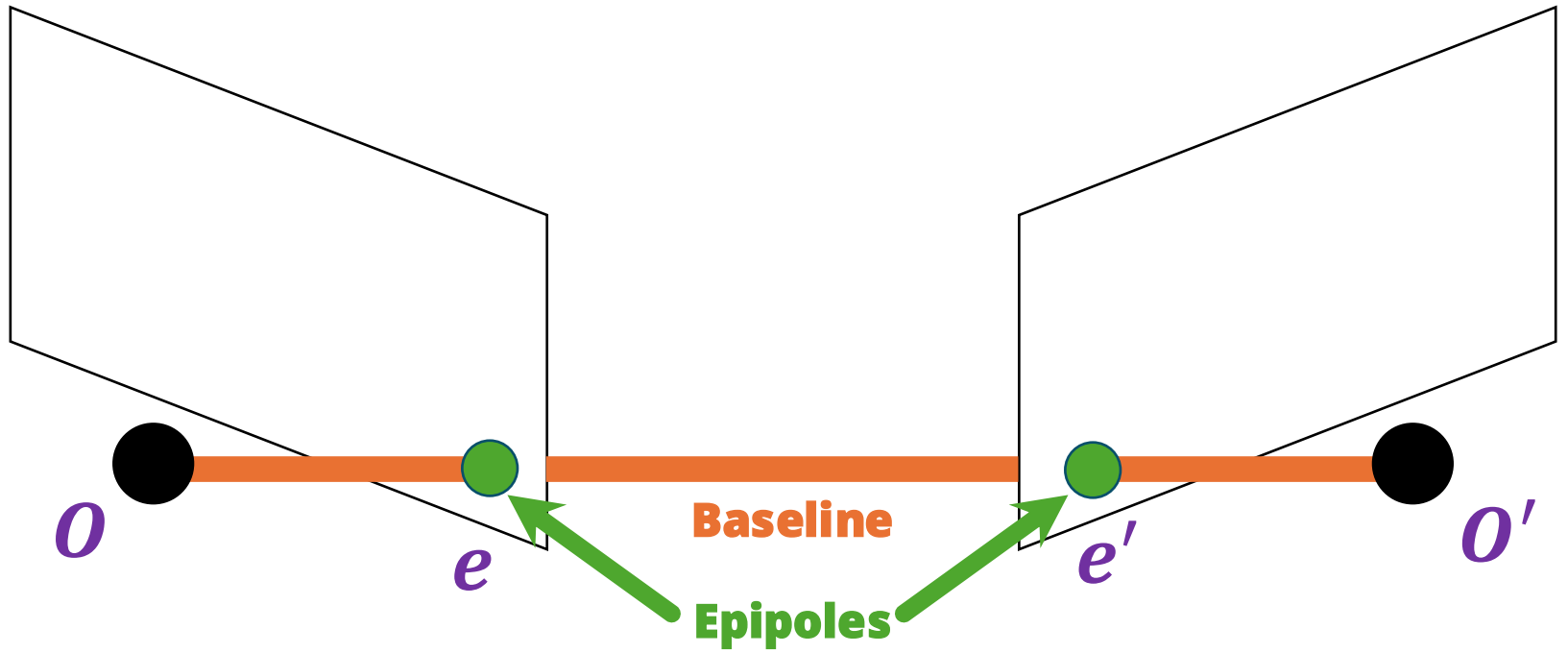
- Projecting points on a ray to another camera forms a **line**, i.e., the corresponding point on the second image *must* lie on this line
- Hence, if the *matched* 2D point is found along this line, we can determine its 3D location

Epipolar Geometry



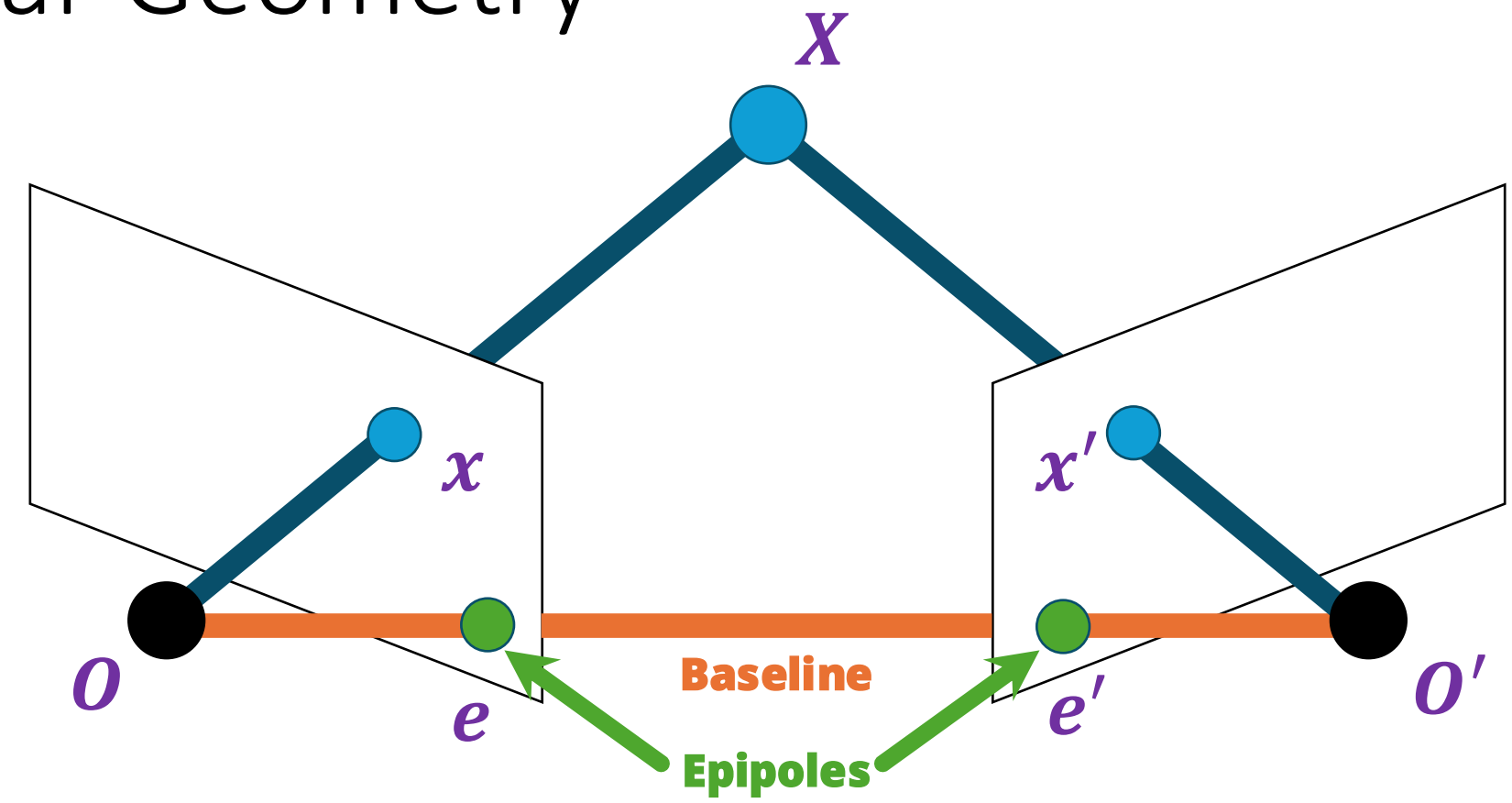
- Suppose we have two cameras with centers O, O'
- The **baseline** is the line connecting the origins

Epipolar Geometry



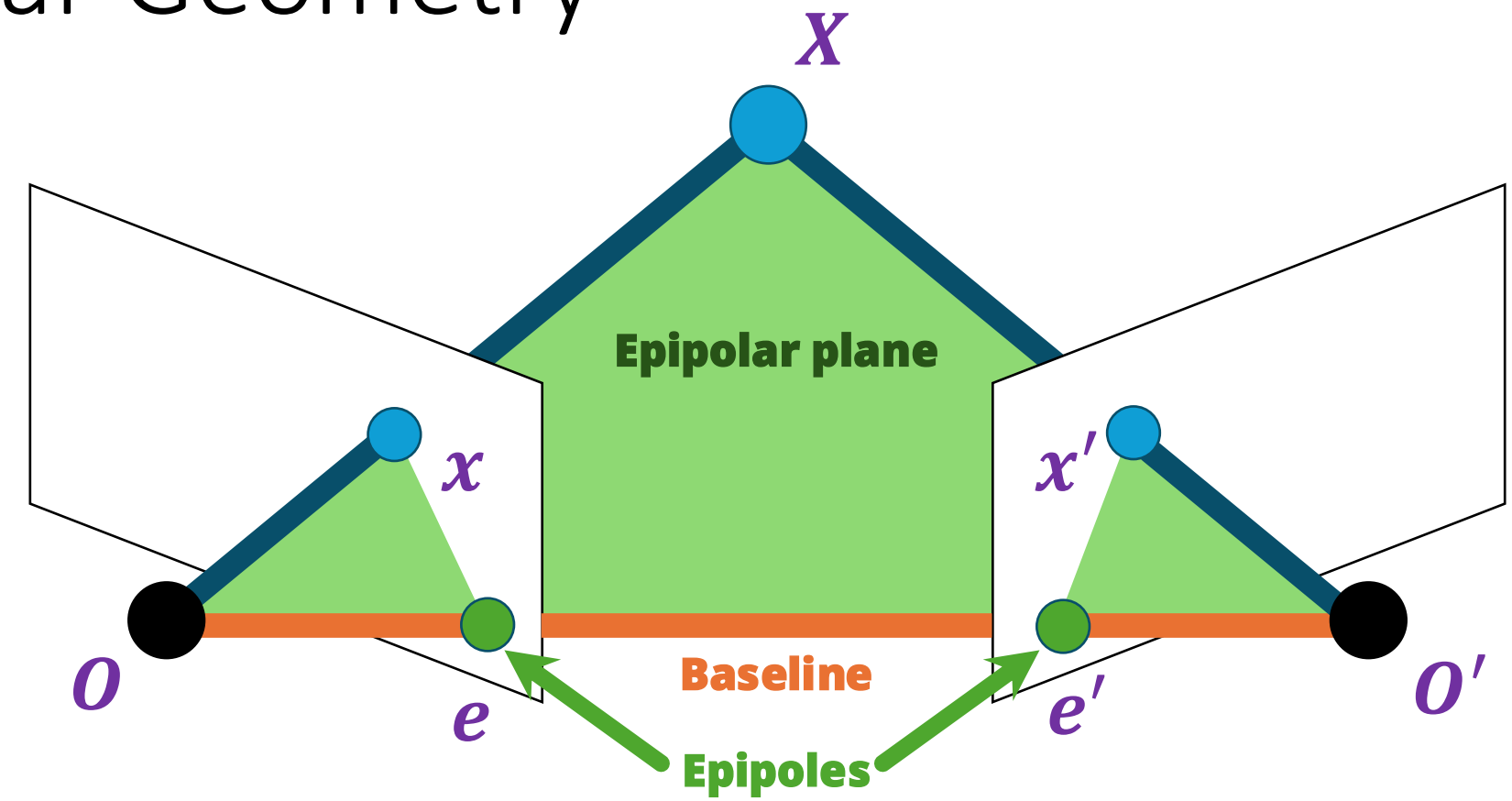
- **Epipoles** e, e' are where the baseline intersects the image planes, or projections of the other camera in each view
- **Epipoles** may lie outside the images

Epipolar Geometry



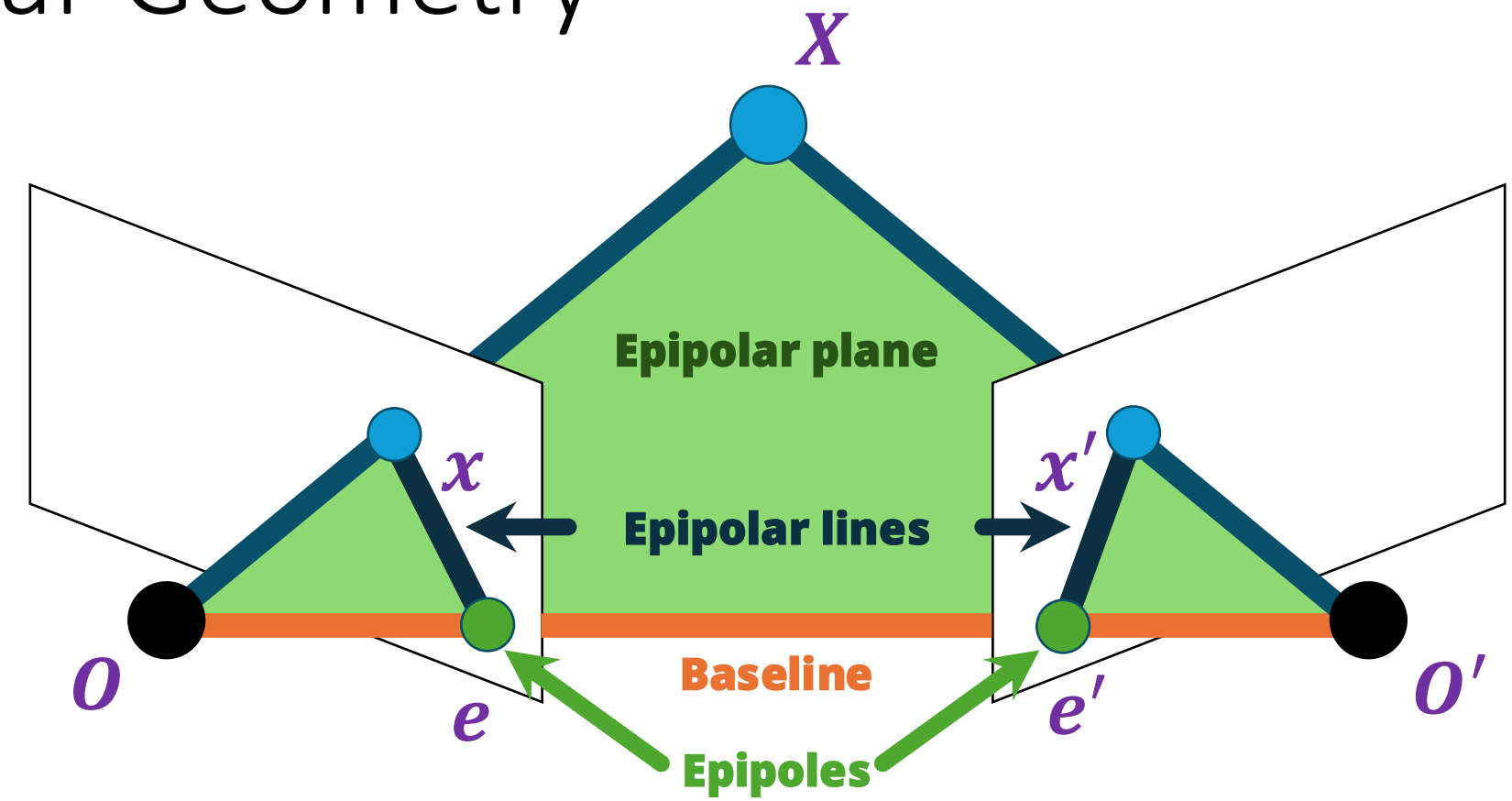
- Consider a **point** X, which projects to x and x'

Epipolar Geometry



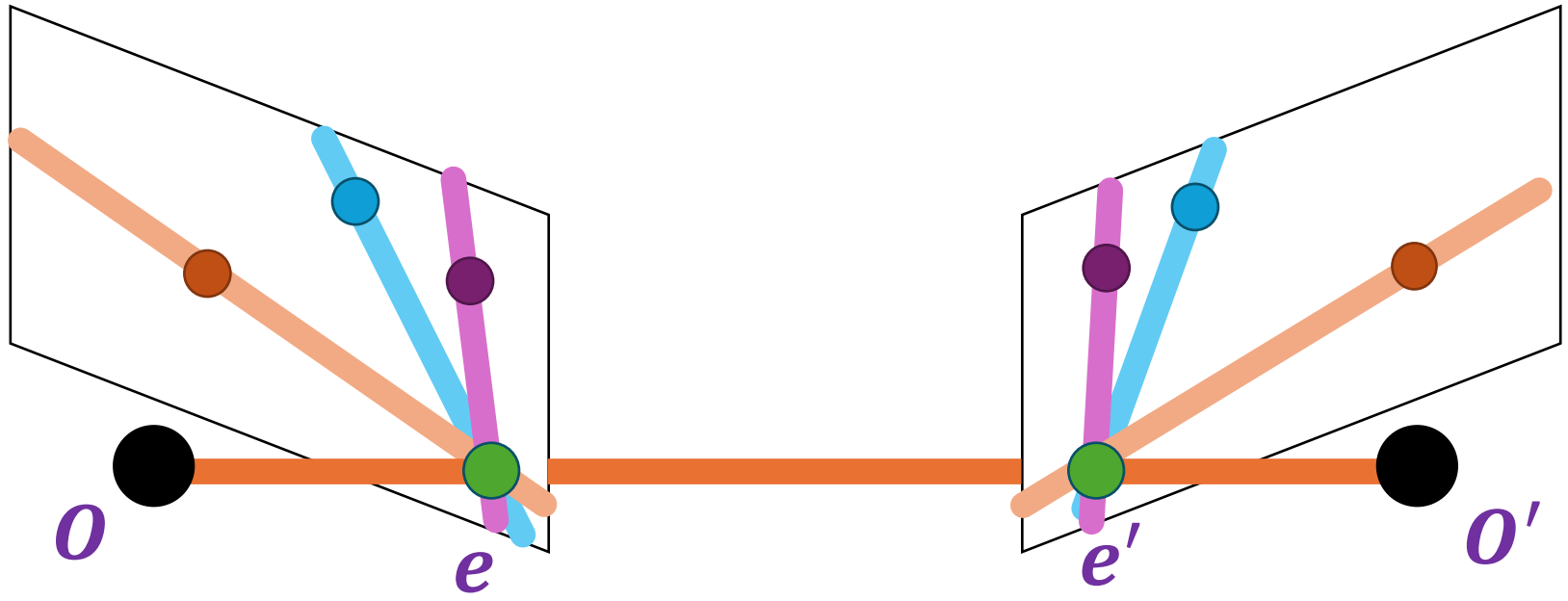
- The plane formed by X, O, and O' is called an **epipolar plane**
- There is a family of planes passing through O and O'

Epipolar Geometry



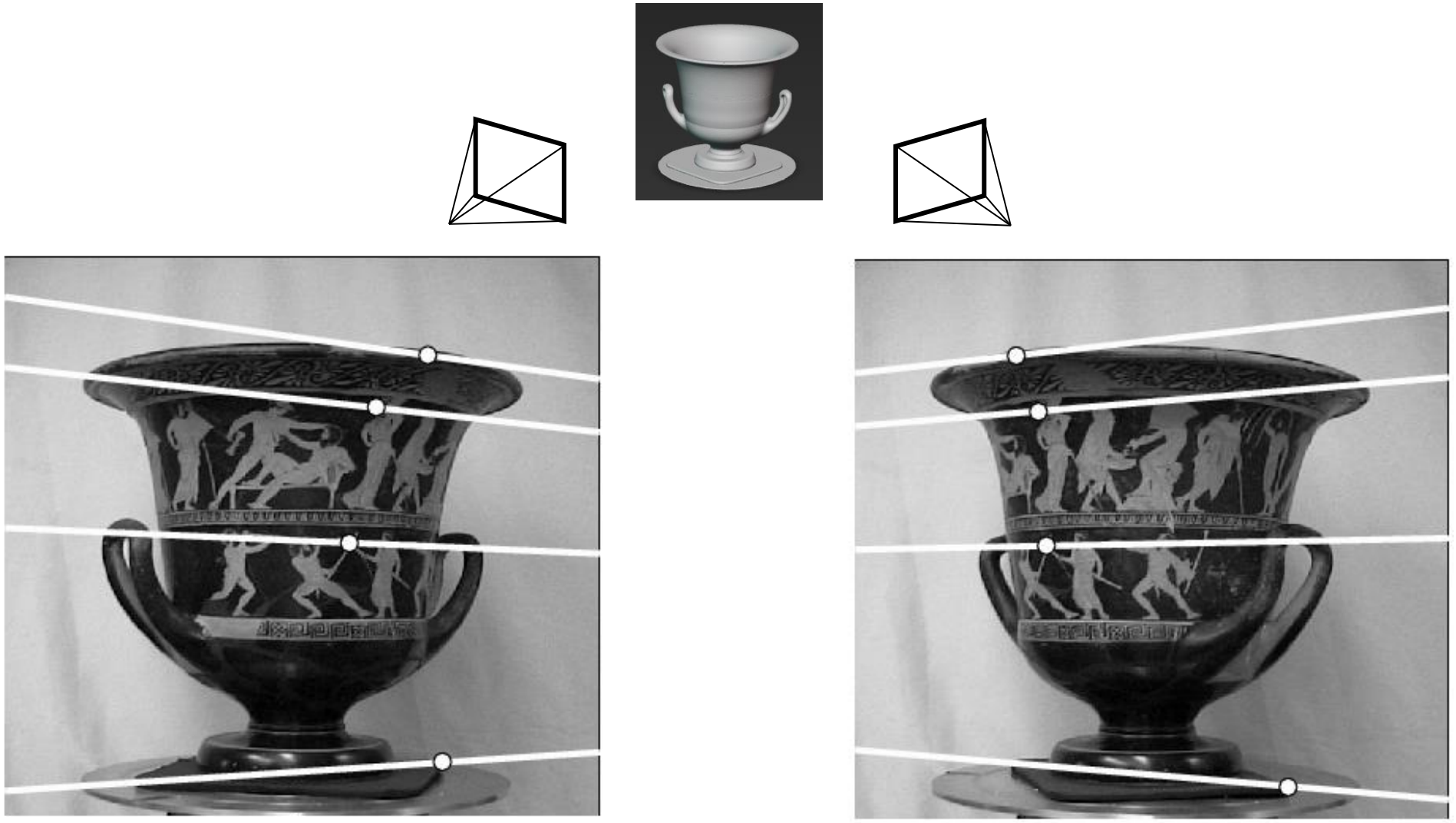
- **Epipolar lines** connect the epipoles to the projections of X
- Equivalently, they are intersections of the epipolar plane with the image planes – thus, they come in matching pairs

Epipolar Geometry

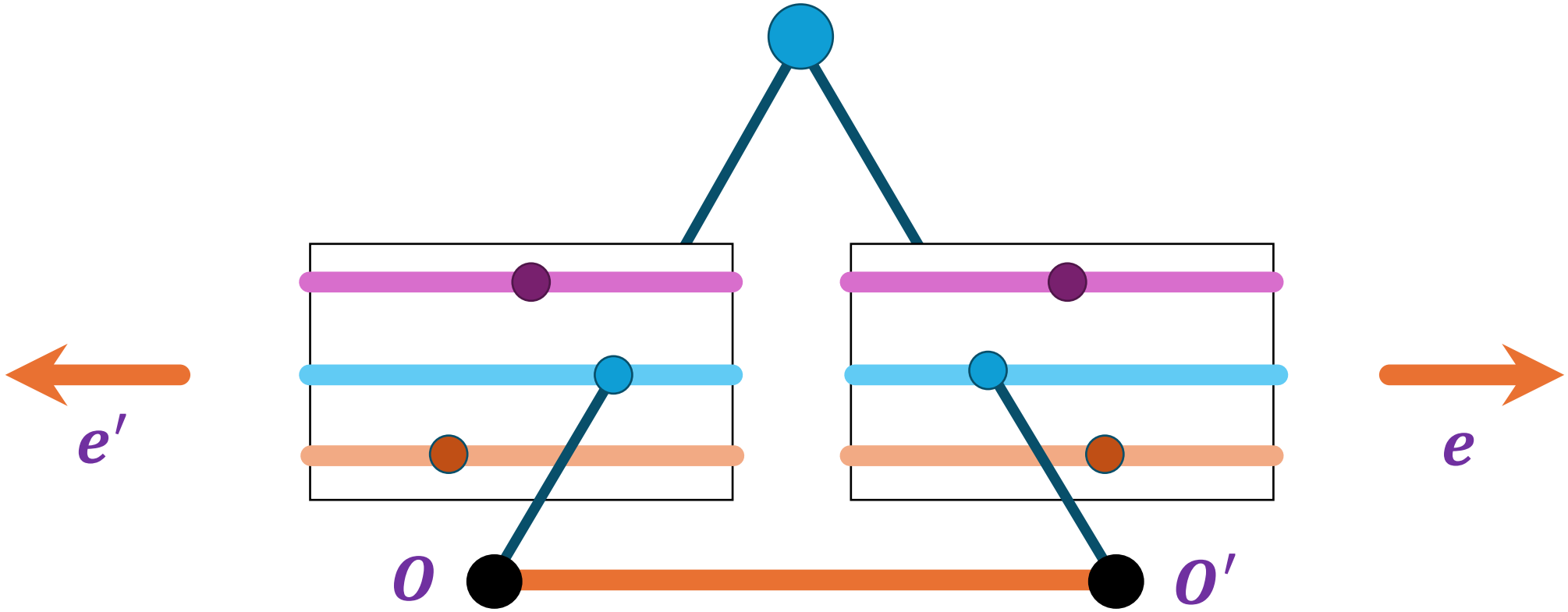


- All **epipolar lines** pass through the **epipoles**

Example of Epipolar Lines

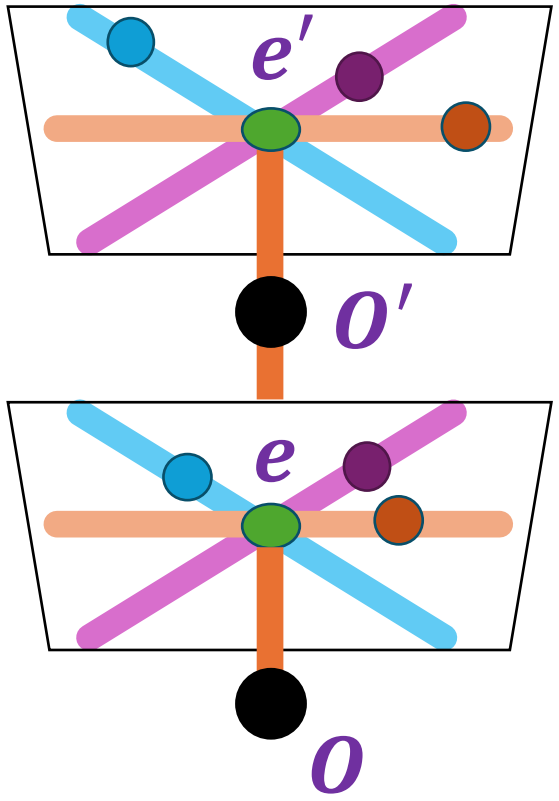


Parallel to Image Plane



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

Perpendicular to Image Plane

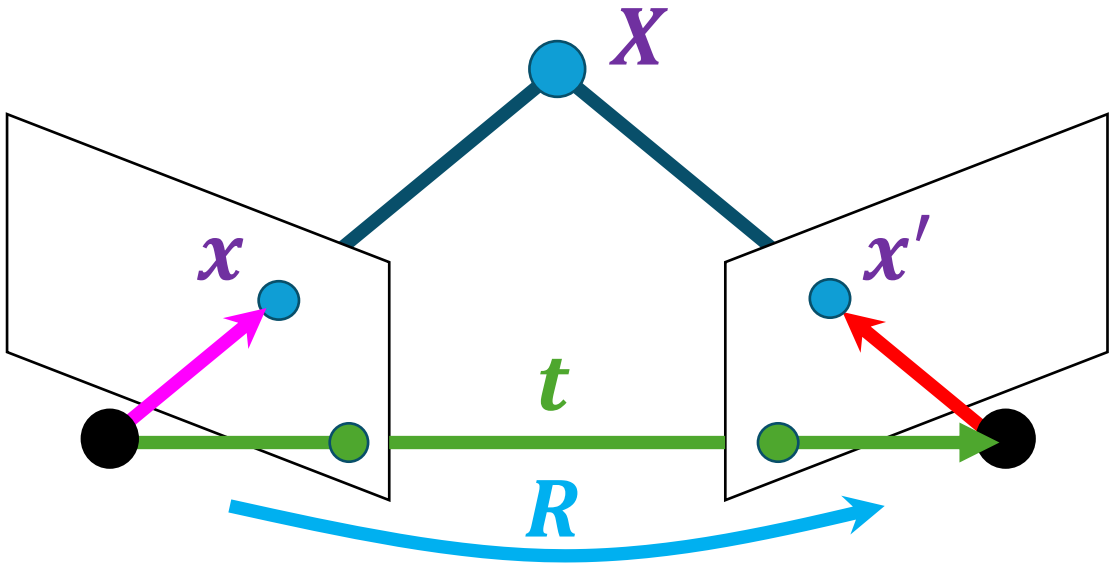


- Where are the epipoles and what do the epipolar lines look like?

- 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 $K[I \mid 0]$ 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:

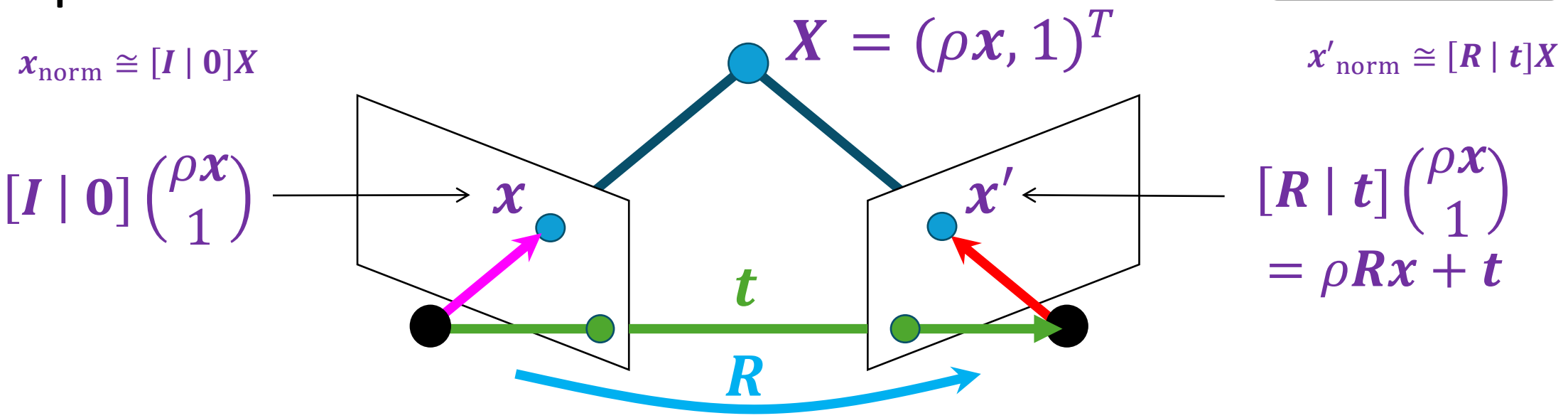
$$x_{\text{norm}} = K^{-1}x_{\text{pixel}} \cong [I \mid 0]X, \qquad x'_{\text{norm}} = K'^{-1}x'_{\text{pixel}} \cong [R \mid t]X$$

Epipolar Constraint: Calibrated Case

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

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

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



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

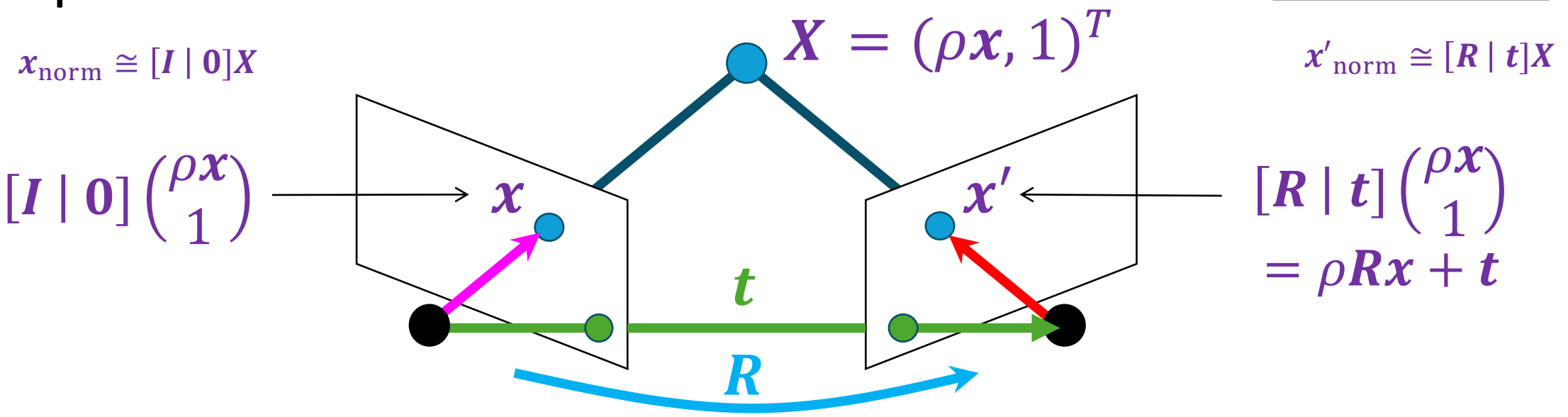
- This means the three vectors x', Rx, and t are linearly dependent, i.e., lying on the same plane
- This constraint can be written using the *triple product*

Epipolar Constraint: Calibrated Case

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

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

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



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

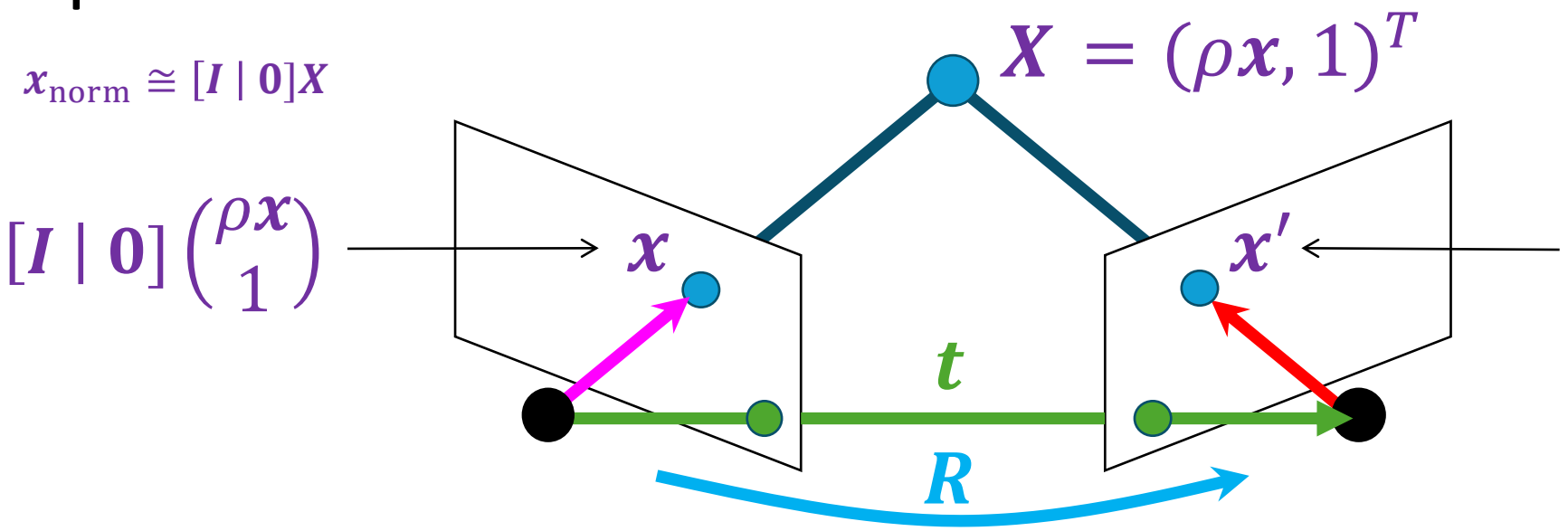
Recall: $a \times b = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = [a_\times]b$ ⇒ skew-symmetric matrix: $A^T = -A$

Epipolar Constraint: Calibrated Case

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

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

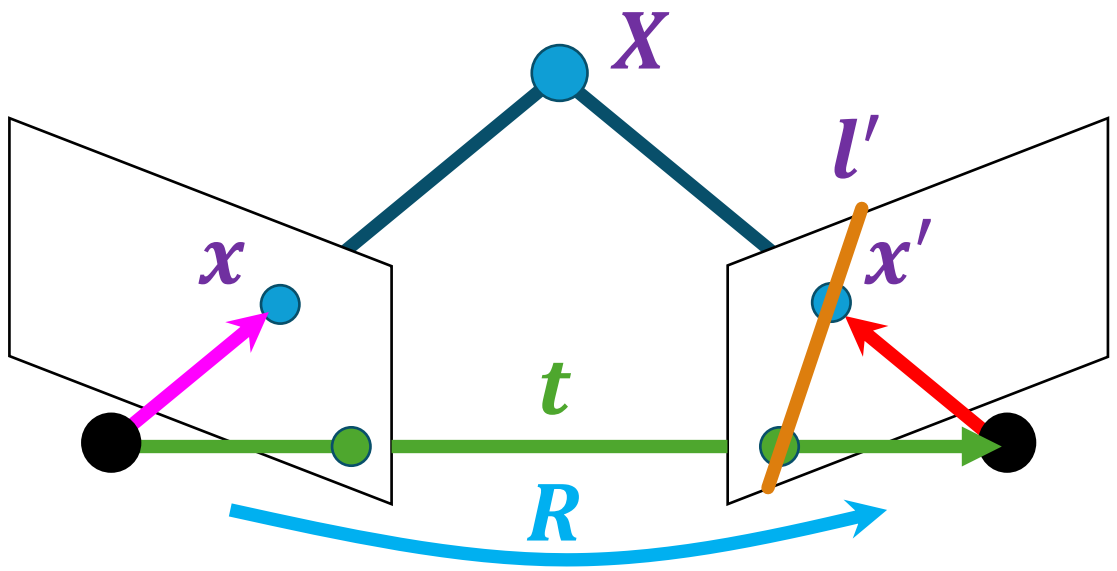
$x'_{\text{norm}} \cong [R \mid t]X$



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

Essential Matrix

The Essential Matrix



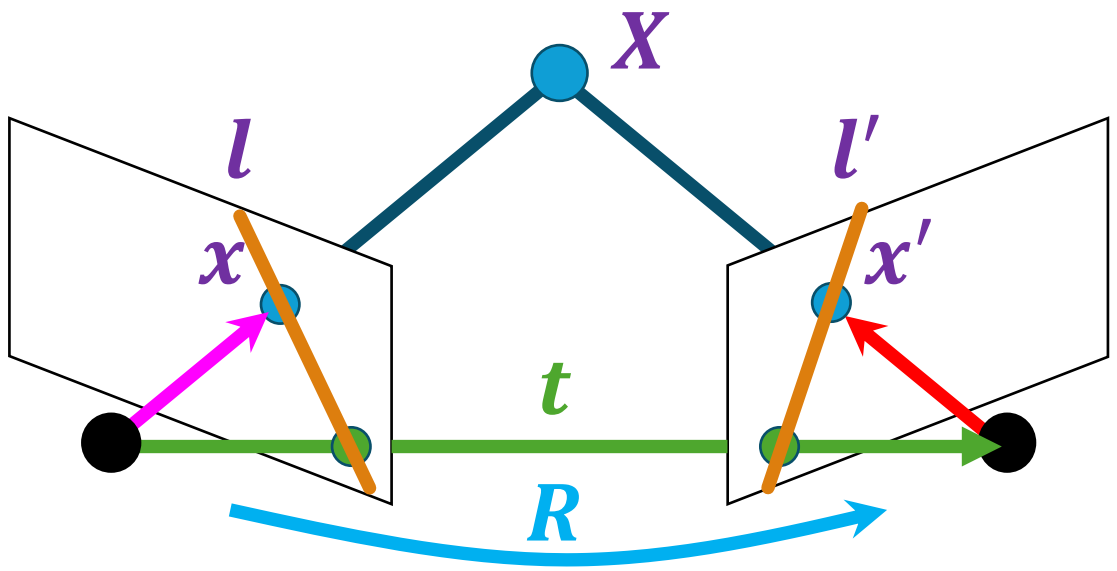
$$l' = E^T x$$

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

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

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

The Essential Matrix



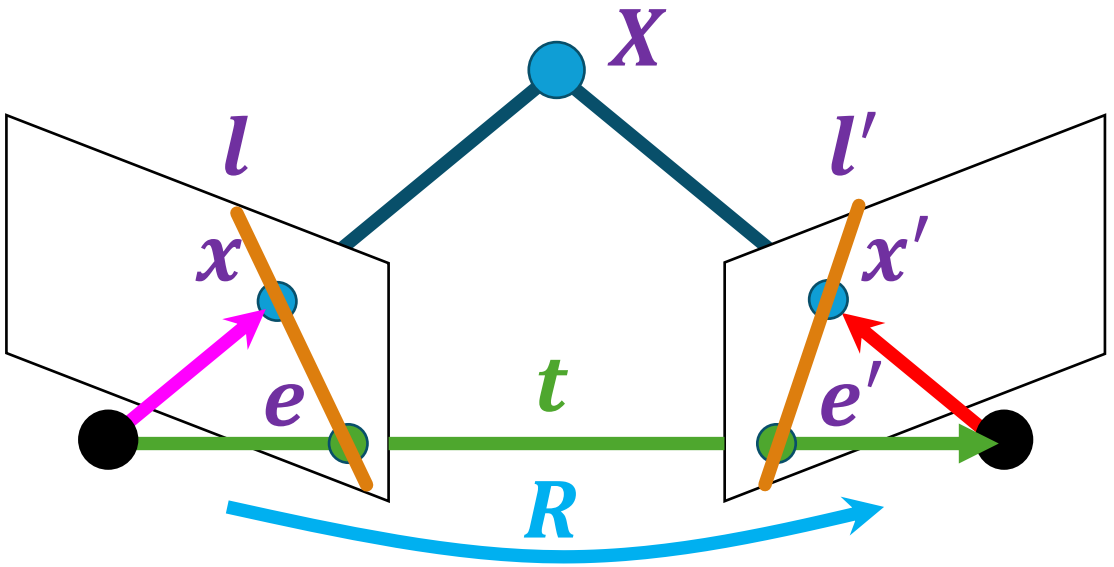
$$\boldsymbol{l} = \boldsymbol{E}^T \boldsymbol{x}'$$

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

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

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

The Essential Matrix



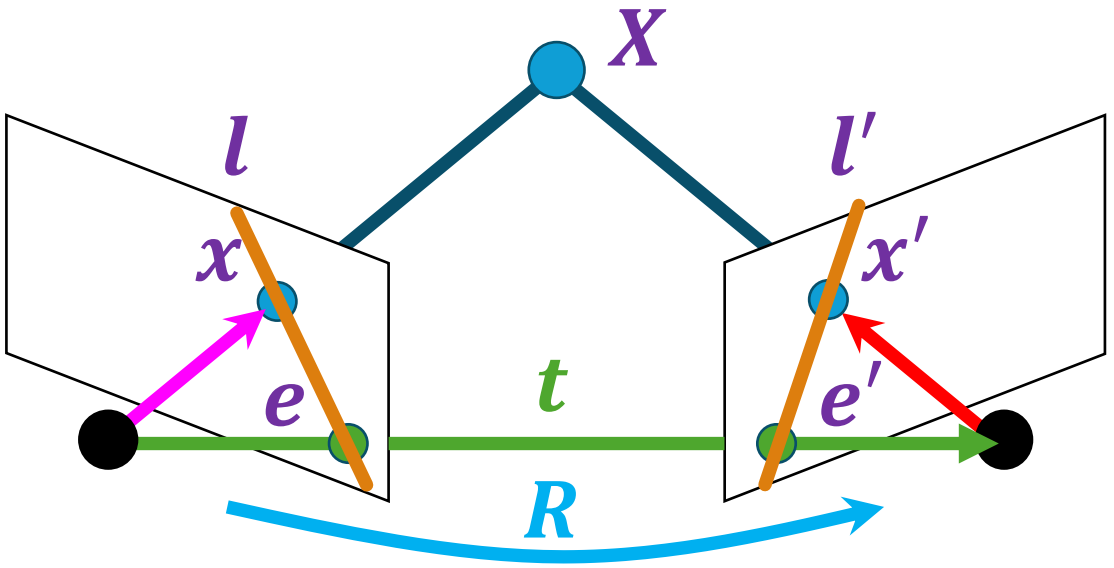
$$\boldsymbol{x'^T E x} = 0$$

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

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

Epipolar Constraint: Uncalibrated

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



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

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

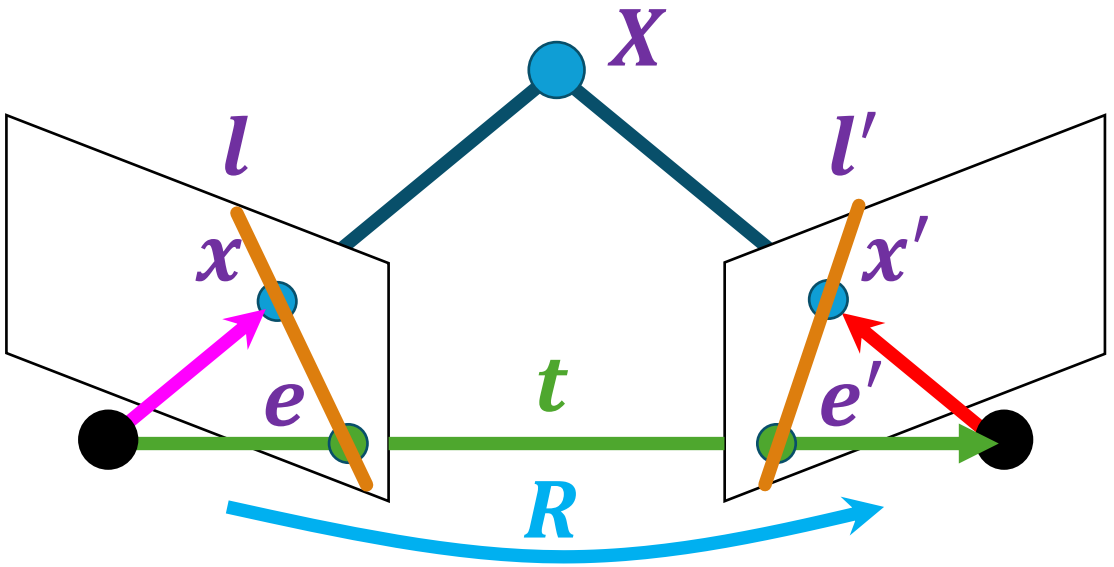
- What if camera intrinsics K, K' are unknown?
- We can write the epipolar constraint:

$$x'^T_{\text{norm}} E x_{\text{norm}} = 0$$

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

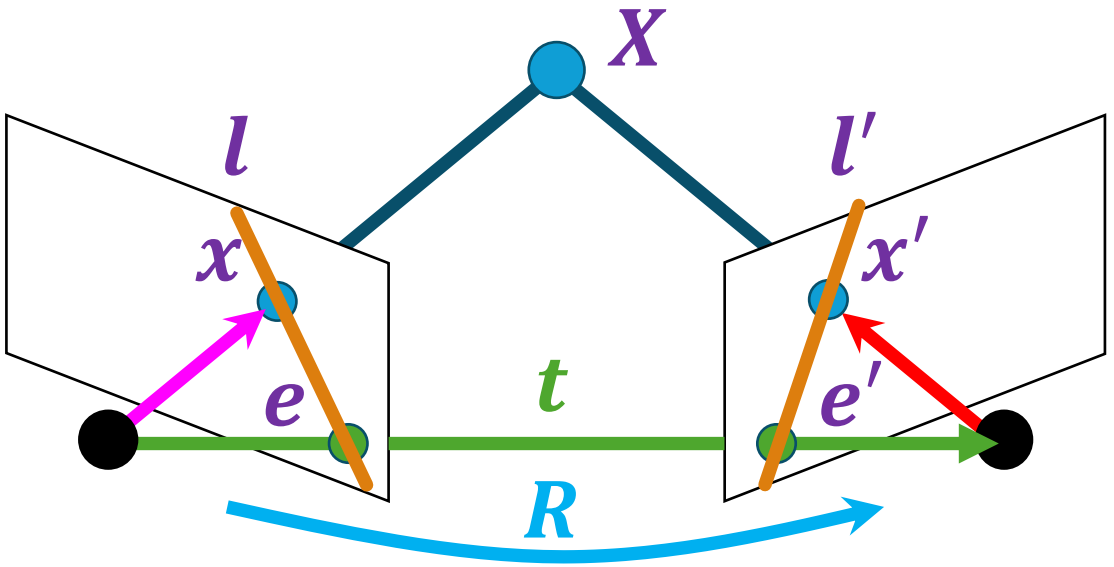
$$F = K'^{-T} E K^{-1}$$

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

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

The Fundamental Matrix

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



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

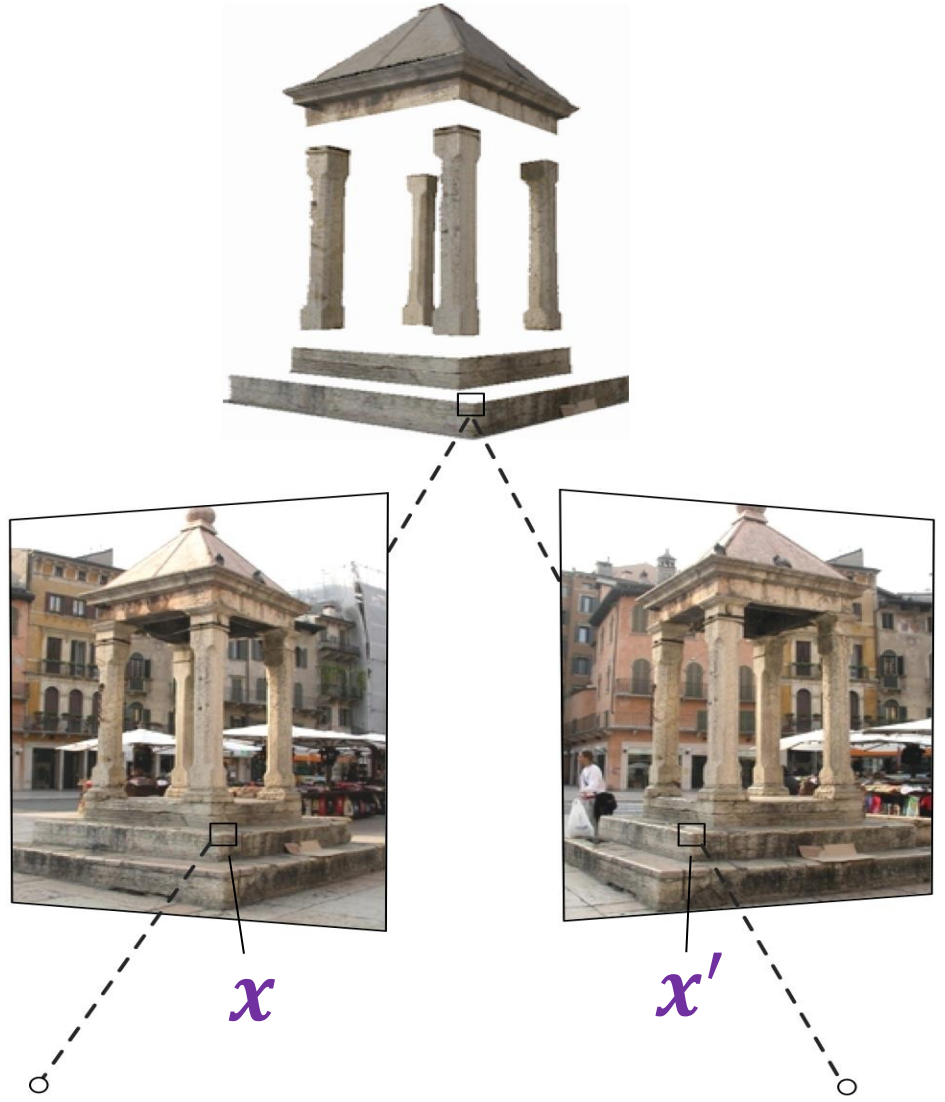
$$(x', y', 1)\begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = 0$$

- 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?
  - $F = K'^{-T}[t_\times]RK^{-1}$! $[t_\times]$ 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?

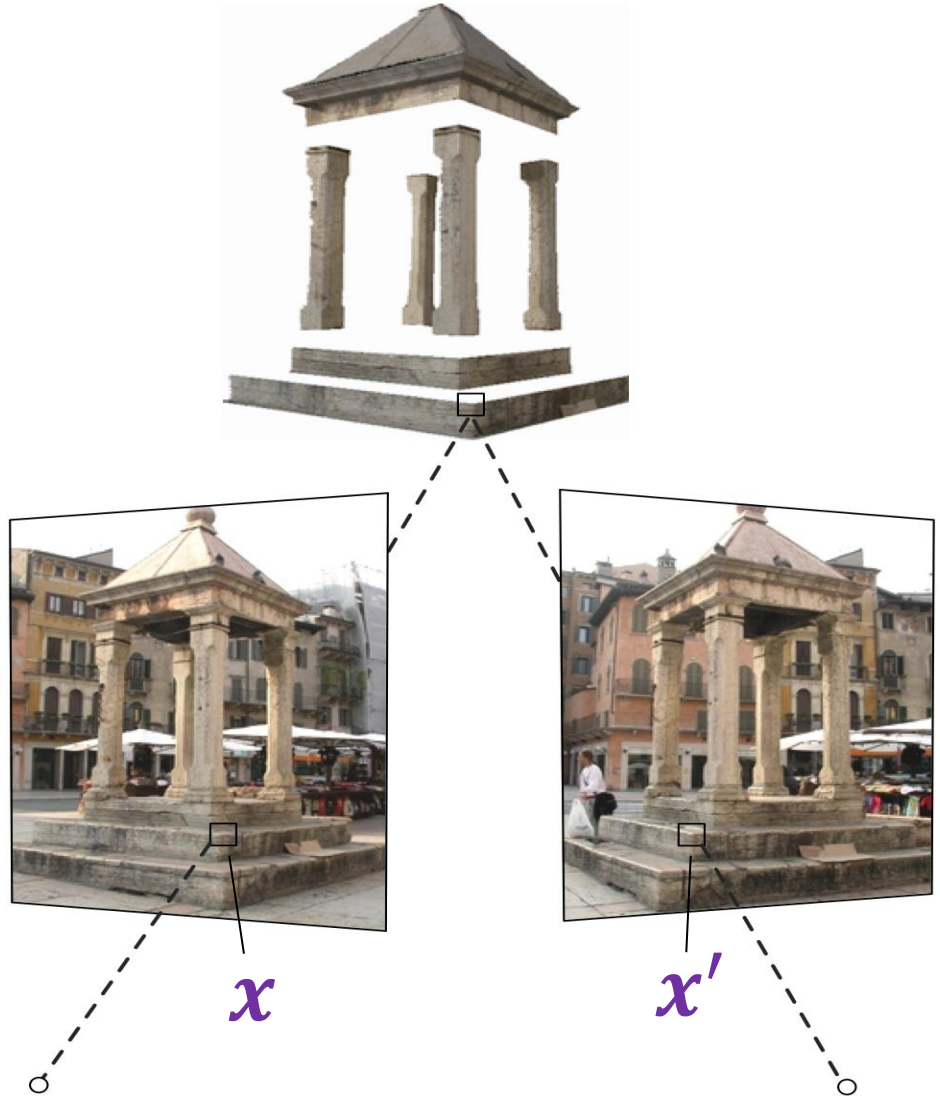
- **Given:** $\boldsymbol{F}$, $\boldsymbol{x}$, $\boldsymbol{x'}$
- **Q:** does there exist a 3D point that projects to $\boldsymbol{x}$ and $\boldsymbol{x'}$?
- **A:** Yes, if $\mathrm{residual}(\boldsymbol{x'}^T \boldsymbol{F} \boldsymbol{x})$ is sufficiently low

- Note: the interpretation of $\mathrm{residual}(\boldsymbol{x'}^T \boldsymbol{F} \boldsymbol{x})$ is the distance (geometric or algebraic) between $\boldsymbol{x}$ and $\boldsymbol{l} = \boldsymbol{F}^T \boldsymbol{x'}$, or $\boldsymbol{x'}$ and $\boldsymbol{l'} = \boldsymbol{F}\boldsymbol{x}$



How Can We Use the Epipolar Constraint?

- **Given:** $\boldsymbol{F}$
- **Q:** how do we find $\boldsymbol{R}$, $\boldsymbol{t}$?
- **A:**
  - **Step 0:** estimate $\boldsymbol{K}$, $\boldsymbol{K'}$ if not known (self-calibration)
  - **Step 1:** compute $\boldsymbol{E = K'^T F K}$
  - **Step 2:** since $\boldsymbol{E = [t_\times]R}$, perform SVD on $\boldsymbol{E = U\Sigma V^T}$. We have 4 solutions $\boldsymbol{(R_1, \pm t)}$, $\boldsymbol{(R_2, \pm t)}$, where $\boldsymbol{t = U[:, 3]}$, $\boldsymbol{R_1 = UWV^T}$, $\boldsymbol{R_2 = UW^TV^T}$, and $\boldsymbol{W}$ 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 $\boldsymbol{x_i} = (x_i, y_i, 1)^T$ and $\boldsymbol{x_i'} = (x_i', y_i', 1)^T$
- **Constraint:** $\boldsymbol{x'^T F x} = 0$
- **Q:** $\boldsymbol{F}$?

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

How to Estimate the Fundamental Matrix?

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

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

Homogeneous least squares to find $\boldsymbol{f}$:

$$\arg\min_{\|\boldsymbol{f}\|=1} \|\boldsymbol{A}\boldsymbol{f}\|_2^2$$ ⟹ Least eigenvector of $\boldsymbol{A^T A}$

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

A Small Trick to Enforce Rank-2 Constraint

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

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

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

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

The Fundamental Matrix Song

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

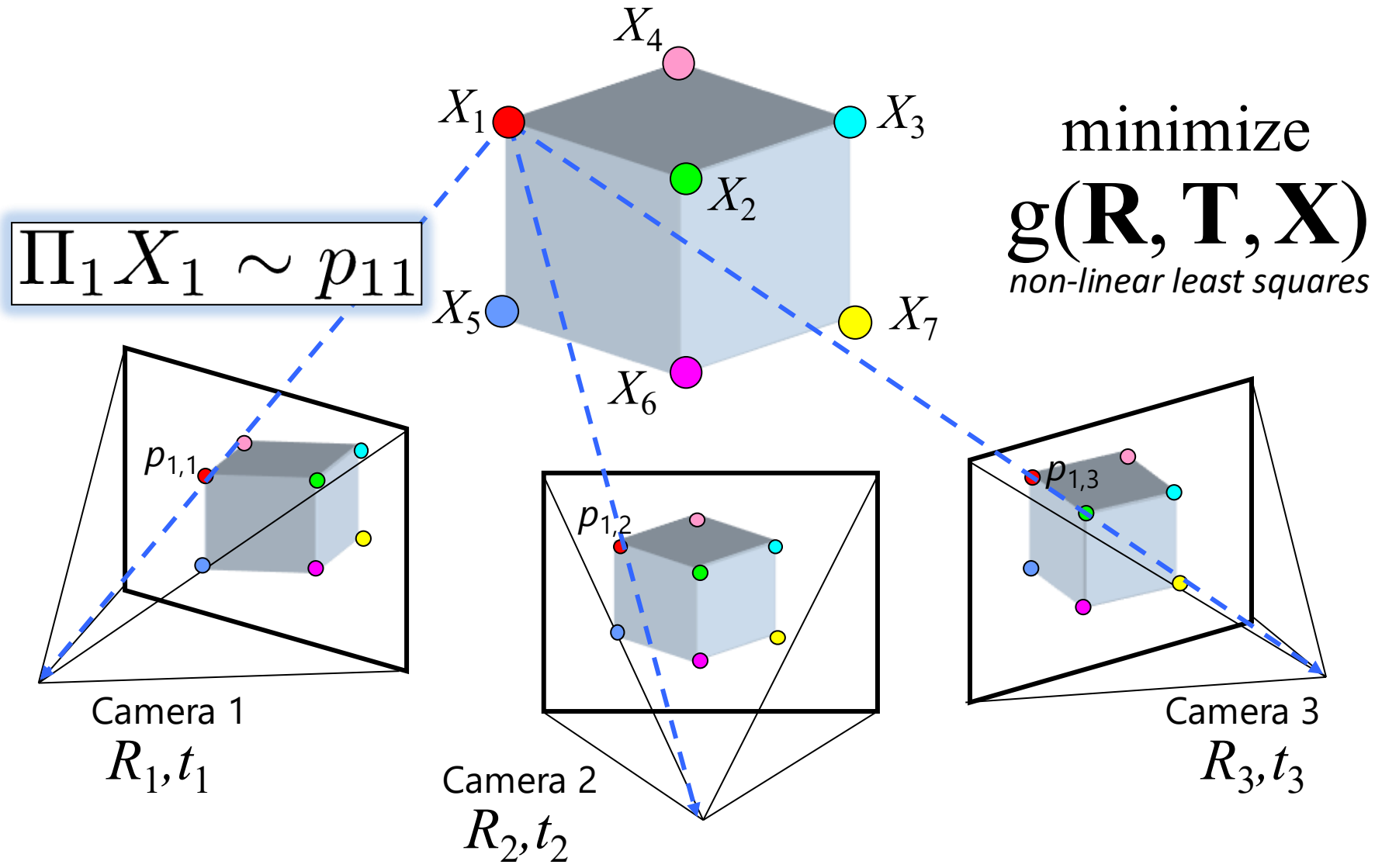
The Fundamental Matrix Song – Live!

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

Bundle Adjustment



Bundle Adjustment

- Minimize reprojection errors:

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

indicator variable: is point *i* visible in image *j* ?

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

Incremental Structure from Motion (SfM)

- Photo Tourism (Snavely et al., SIGGAPH'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 $\boldsymbol{x} \cong \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}]\boldsymbol{X}$
  - Intrinsics $\boldsymbol{K}$, extrinsics $[\boldsymbol{R}|\boldsymbol{t}]$
  - Camera calibration
- Epipolar geometry
  - Epipoles, epipolar plane, epipolar lines
  - Essential matrix $\boldsymbol{E}$ => $\boldsymbol{x}'^{T}_{\mathrm{norm}}\boldsymbol{E}\boldsymbol{x}_{\mathrm{norm}} = 0$ (calibrated)
  - Fundamental matrix $\boldsymbol{F}$ => $\boldsymbol{x}'^{T}\boldsymbol{F}\boldsymbol{x} = 0$ (uncalibrated)
- Structure from Motion (SfM)
  - Bundle adjustment

Coursework 2 Released

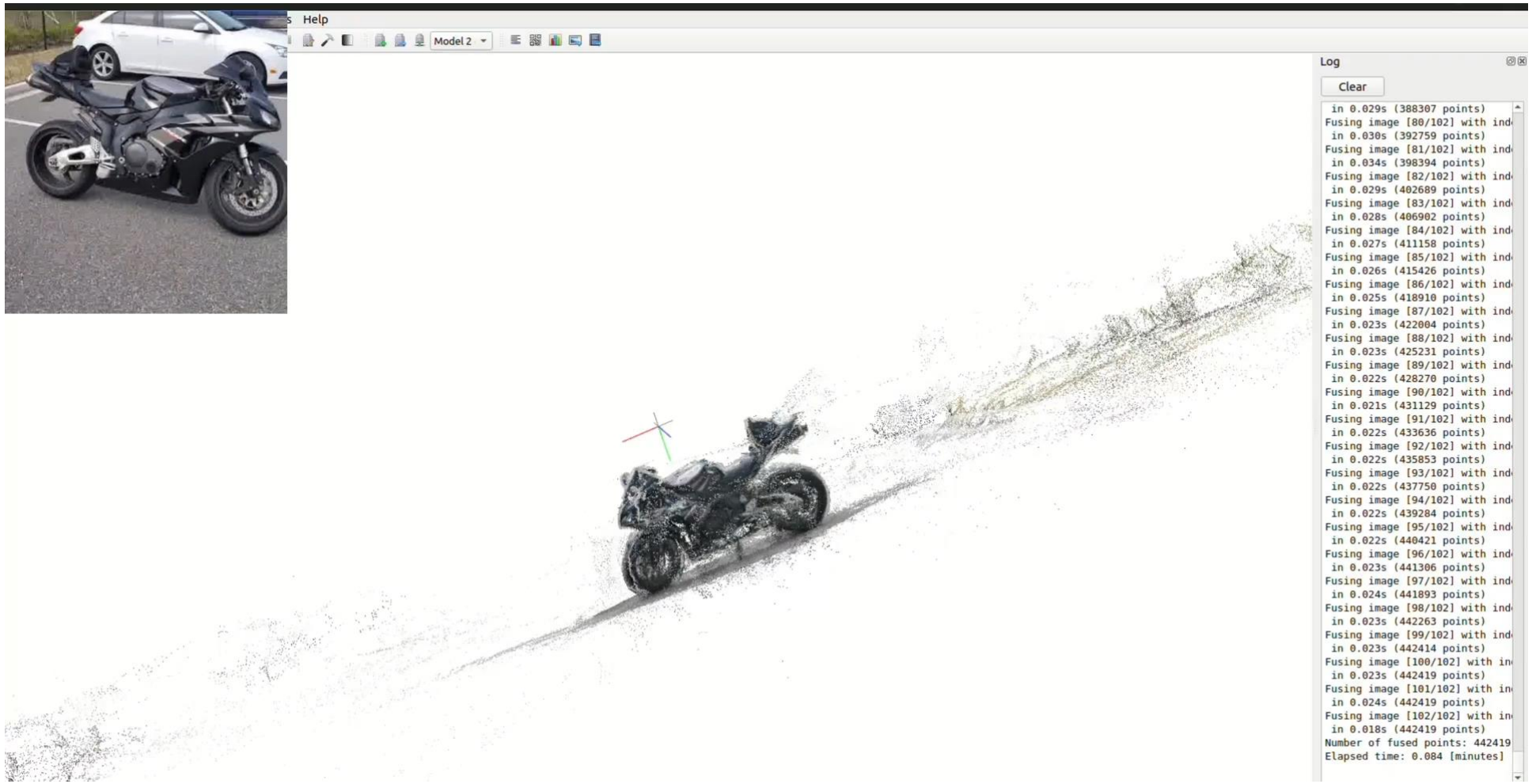
- PDF: https://cambridgecvcourses.github.io/assets/CW/CW2.pdf
- Codebase: https://github.com/CambridgeCVCourses/CW2
- **Due**: Tuesday, 24 March 2026, 12:00 PM (noon)
- Office hours: Fridays, 3-4PM (BE4-54)

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 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†, Yohann Cabon†, Boris Chidlovskii† and Jerome Revaud†
*Aalto University
shuzhe.wang@aalto.fi
†Naver
firstname.last

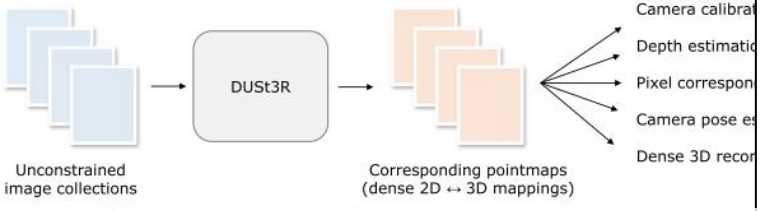


Figure 1. **Overview**: Given an unconstrained image collection, *i.e.* a set of photographs w
proposed method **DUSt3R** outputs a set of corresponding *pointmaps*, from which we can st
quantities normally difficult to estimate all at once, such as the camera parameters, pixel cor
3D reconstruction. 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 shadi

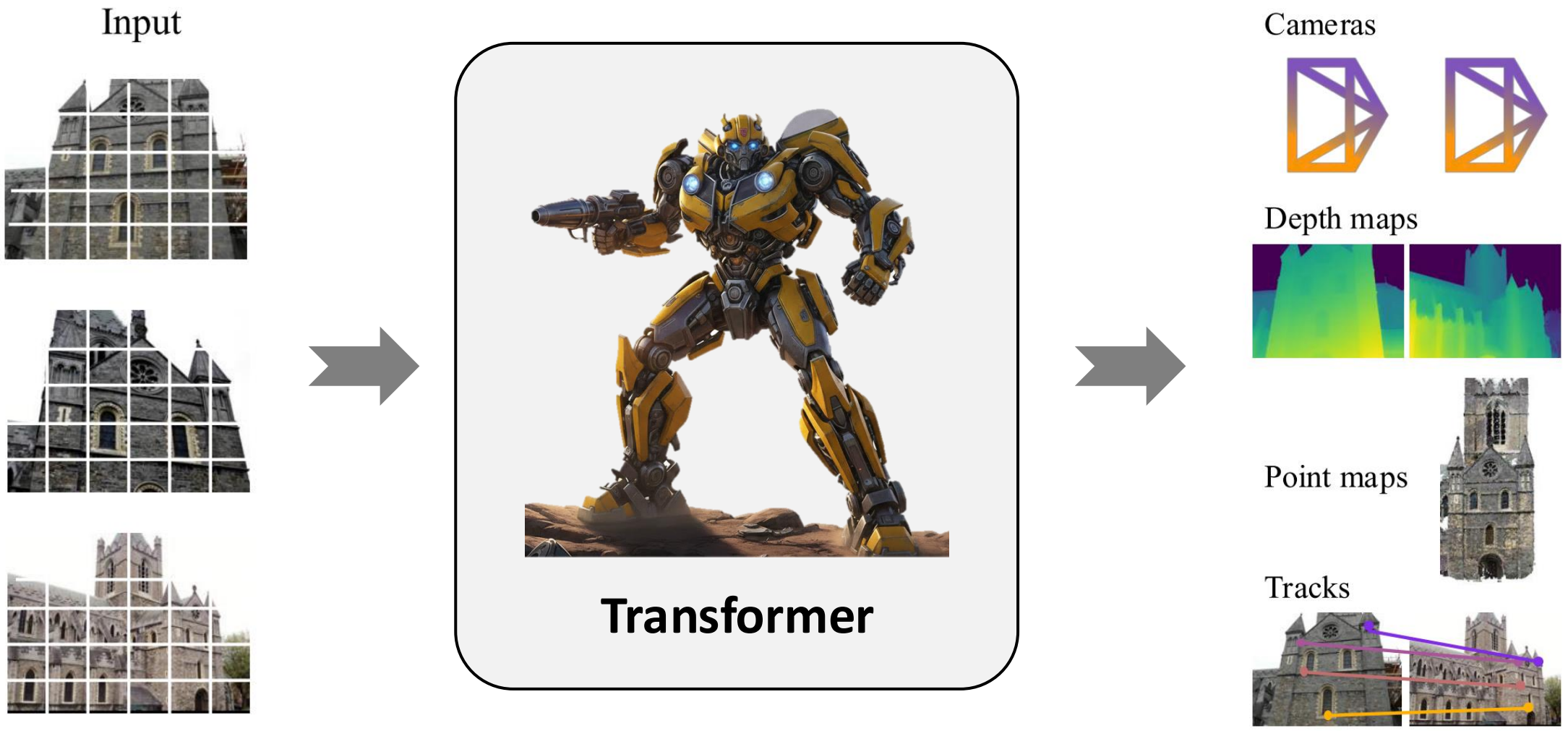
VGGT: Visual Geometry Grounded Transformer

Jianyuan Wang[1,2] Minghao Chen[1,2] Nikita Karaev[1,2] Andrea Vedaldi[1,2]
Christian Rupprecht[1] David Novotny[2]
[1]Visual Geometry Group, University of Oxford [2]Meta AI

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.

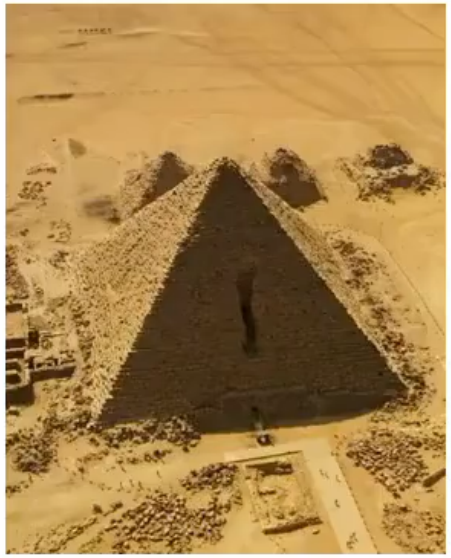
Feed-forward 3D in seconds!!

Feed-forward 3D Reconstruction



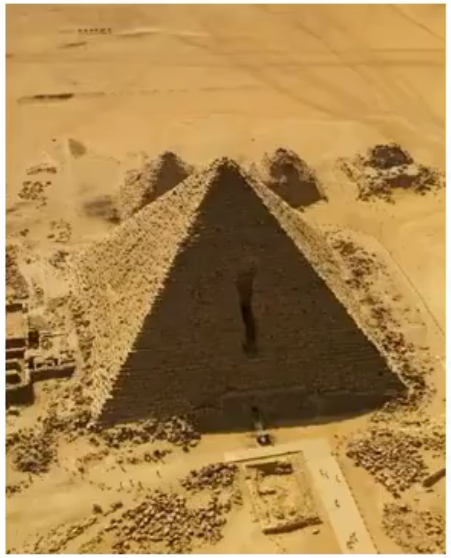
Feed-forward 3D Reconstruction

32 Views



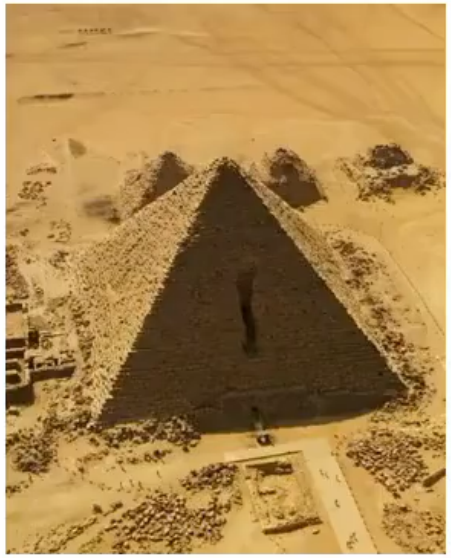
Feed-forward 3D Reconstruction

32 Views



Feed-forward 3D Reconstruction

32 Views

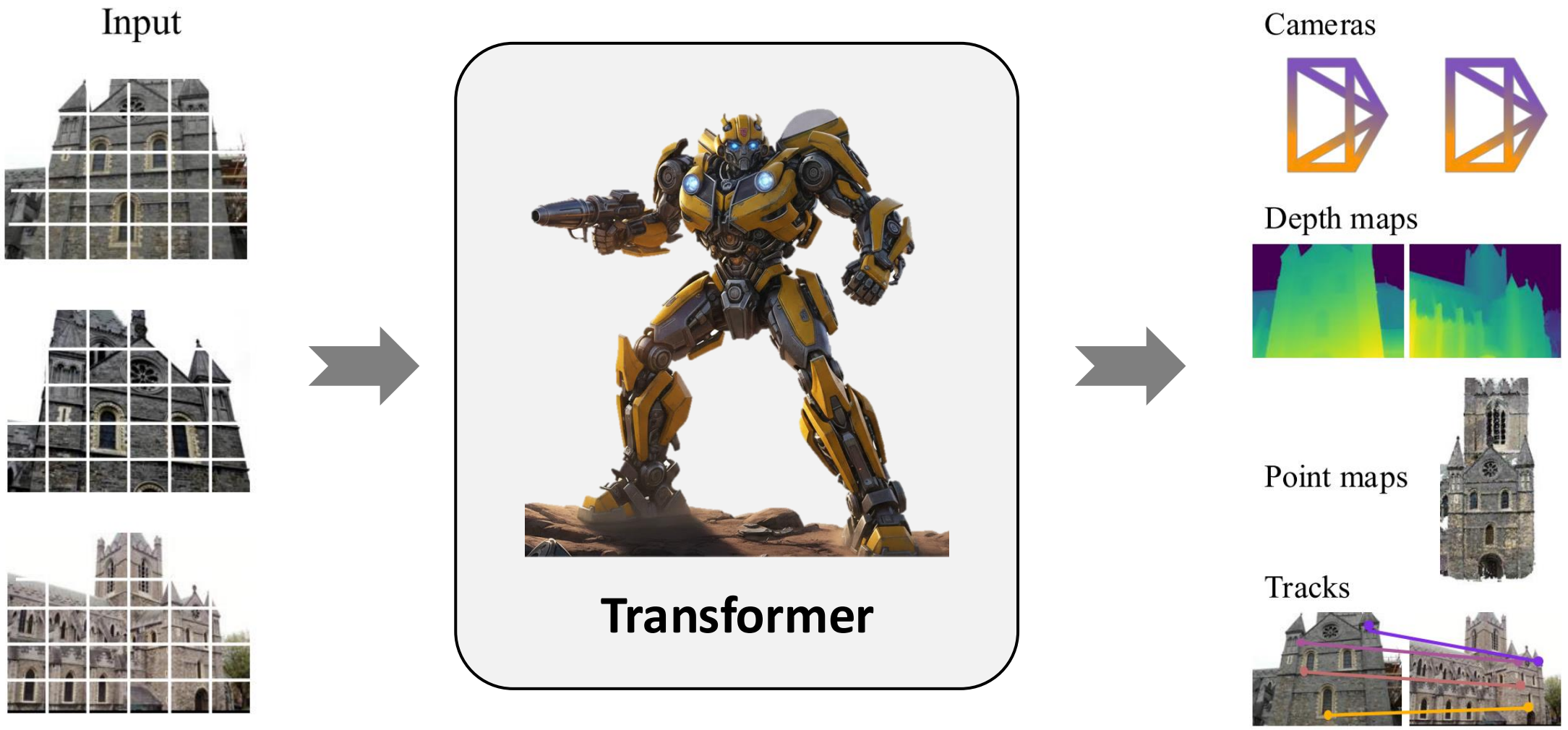


Feed-forward 3D Reconstruction

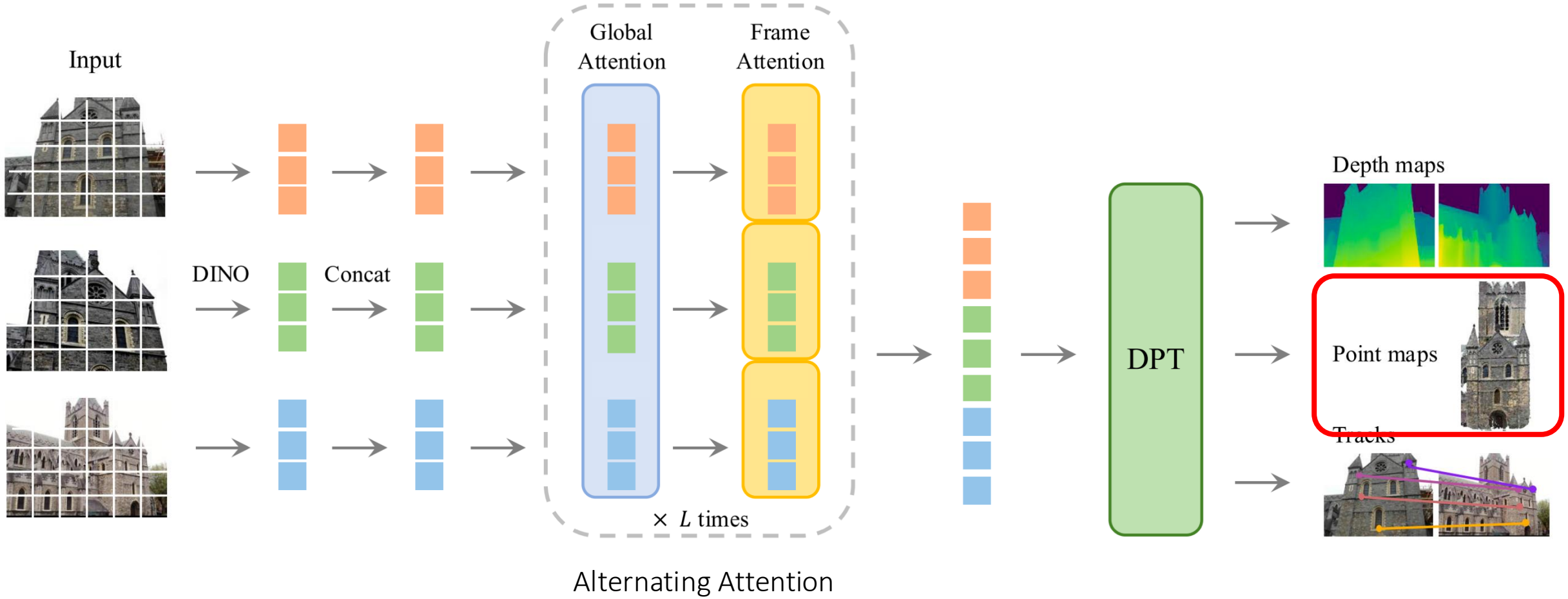
Methods	Re10K *(unseen)* AUC@30 ↑	CO3Dv2 AUC@30 ↑	Time
Colmap+SPSG [92]	45.2	25.3	∼ 15s
PixSfM [66]	49.4	30.1	> 20s
PoseDiff [124]	48.0	66.5	∼ 7s
DUSt3R [129]	67.7	76.7	∼ 7s
MASt3R [62]	76.4	81.8	∼ 9s
VGGSfM v2 [125]	78.9	83.4	∼ 10s
MV-DUSt3R [111] ‡	71.3	69.5	∼ 0.6s
CUT3R [127] ‡	75.3	82.8	∼ 0.6s
FLARE [156] ‡	78.8	83.3	∼ 0.5s
Fast3R [141] ‡	72.7	82.5	∼ **0.2s**
Ours (Feed-Forward)	85.3	88.2	∼ **0.2s**
Ours (with BA)	**93.5**	**91.8**	∼ 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 ‡ represent concurrent work.

Feed-forward 3D Reconstruction



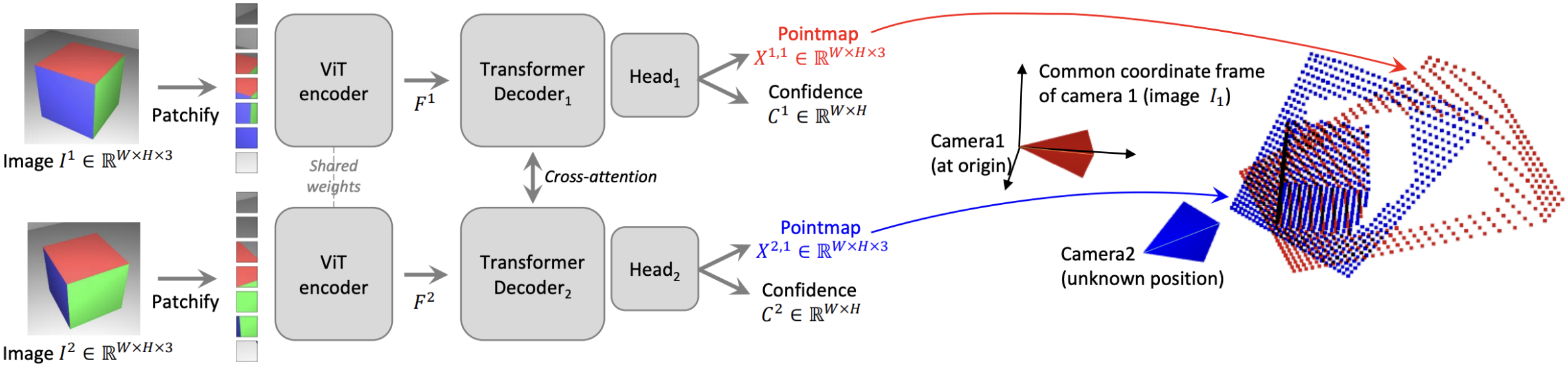
Feed-forward 3D Reconstruction



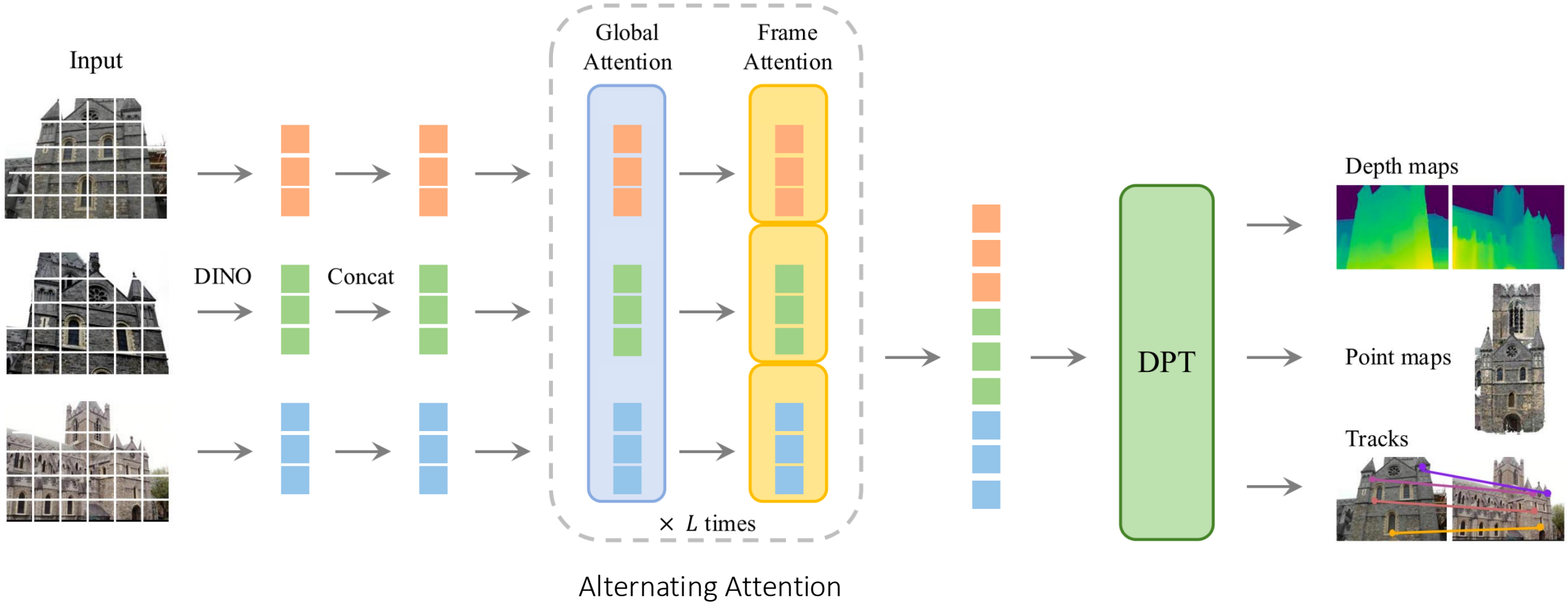
DPT. R. Ranftl et al. ICCV 2021.

VGGT. J. Wang et al. CVPR 2025 (Best Paper Award).

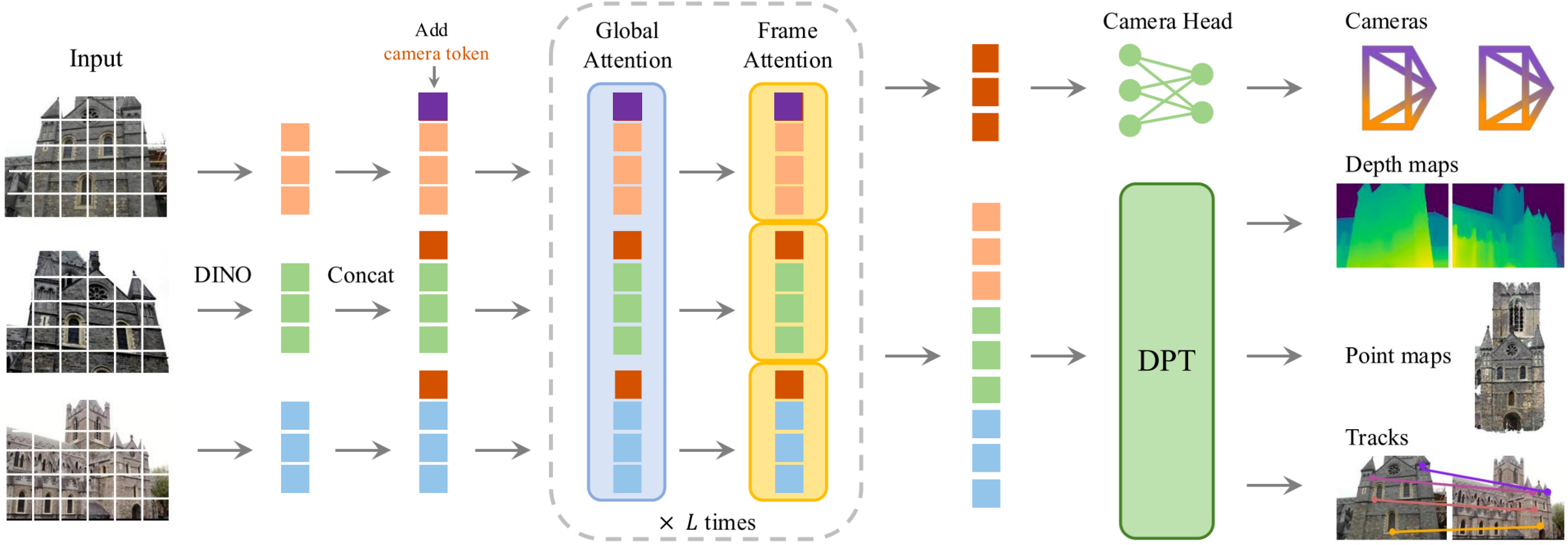
Point Maps from DUSt3R



Feed-forward 3D Reconstruction



Feed-forward 3D Reconstruction



Of Course This Is Obvious, Right?

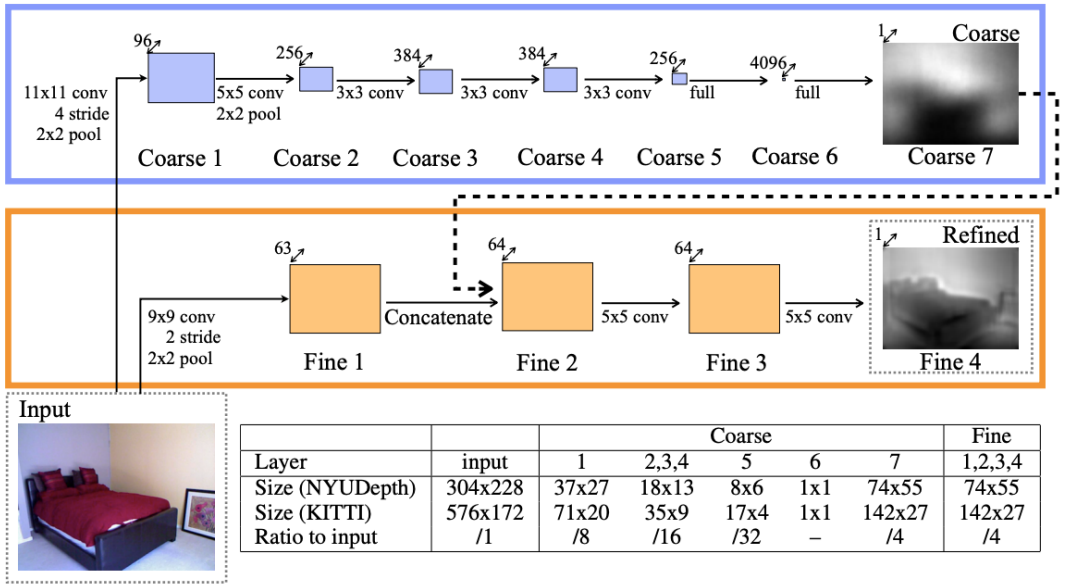
- No, not at all!
- We have long thought that supervised learning would not conquer 3D
  - We thought we would at least need some 3D inductive biases
- Early evidence of supervised 3D learning:
  - Monocular depth estimation (2.5D)

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



		Coarse					Fine
Layer	input	1	2,3,4	5	6	7	1,2,3,4
Size (NYUDepth)	304x228	37x27	18x13	8x6	1x1	74x55	74x55
Size (KITTI)	576x172	71x20	35x9	17x4	1x1	142x27	142x27
Ratio to input	/1	/8	/16	/32	–	/4	/4

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

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

Of Course This Is Obvious, Right?

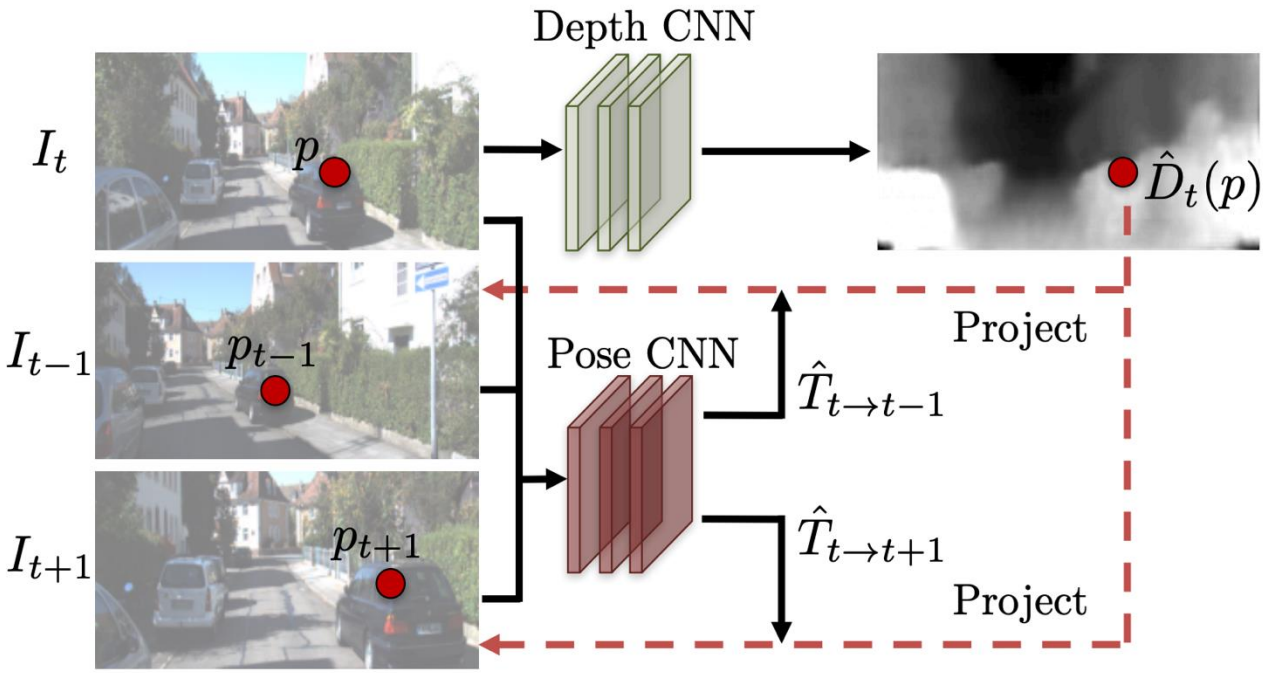
- No, not at all!
- We have long thought that supervised learning would not conquer 3D
  - We thought we would at least need some 3D inductive biases
- Early evidence of supervised 3D learning:
  - Monocular depth estimation (2.5D)
  - 6DoF pose regression (eg, PoseNet)
    - 7 indoor + 5 outdoor scenes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

Unsupervised Learning of Depth and Ego-Motion from Video

Tinghui Zhou* UC Berkeley
Matthew Brown Google
Noah Snavely Google
David G. Lowe Google



SfMLearner. T. Zhou et al. ICCV 2017. 107

Of Course This Is Obvious, Right?

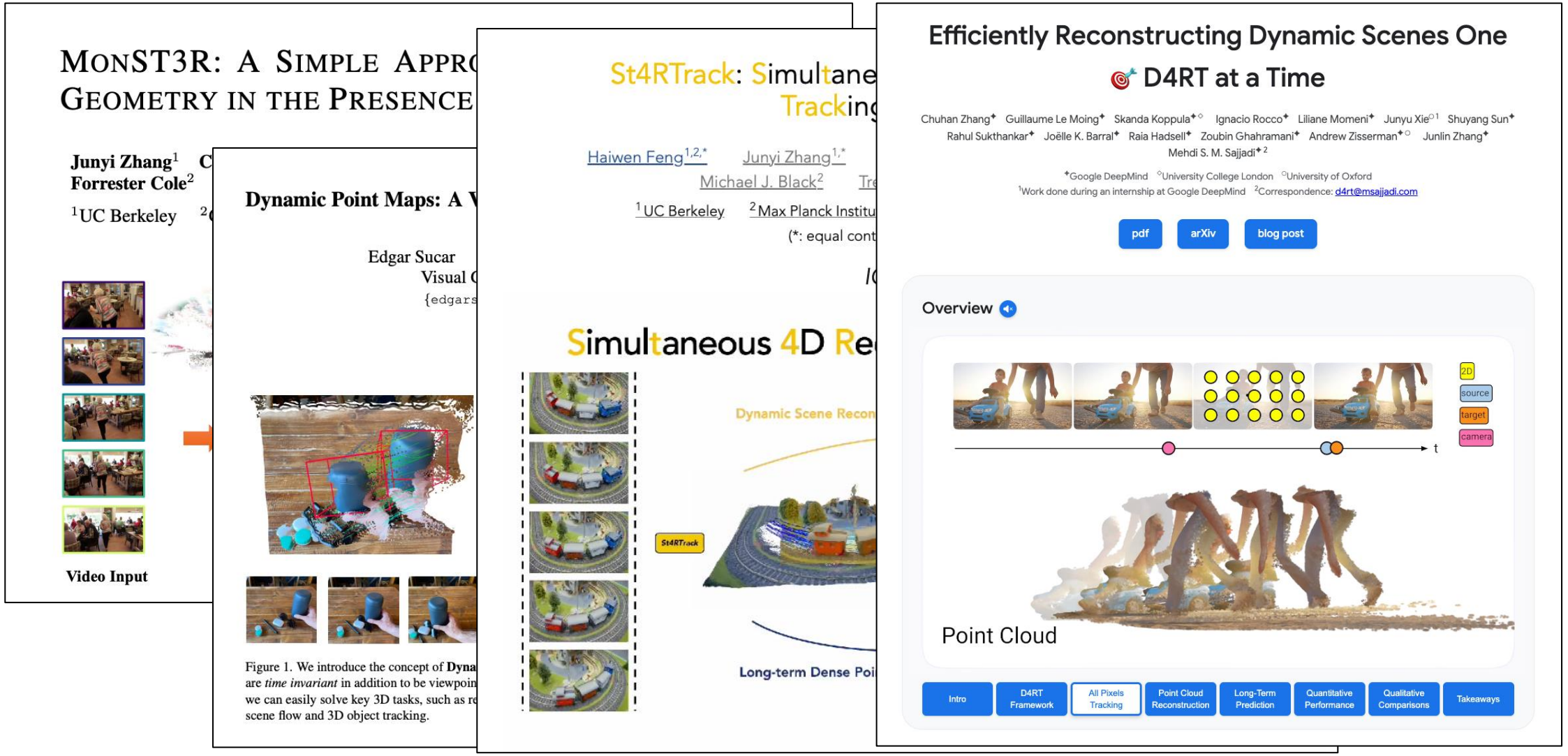
- No, not at all!
- We have long thought that supervised learning would not conquer 3D
  - We thought we would at least need some 3D inductive biases
- Early evidence of supervised 3D learning:
  - Monocular depth estimation (2.5D)
  - 6DoF pose regression (eg, PoseNet)
    - 7 indoor + 5 outdoor scenes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

Dynamic Point Maps and Tracking



Efficient Dense Tracking of All Pixels

Efficiently Reconstructing Dynamic Scenes One D4RT at a Time. Zhang et al. CVPR 2026.

Meanwhile, 3D Generative Models Are Also on Fire!

Hunyuan3D v2.5 by Tencent Hunyuan - Open

Hunyuan 3D: Free

Text-to-3D

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

Apache 2.0 License Runs on 6GB VRA

3D 3D

See Outpu

SAM 3D: 3Dfy Anything in Images

SAM 3D Team, Xingyu Chen*, Fu-Jen Chu*, Pierre
Alexander Sax*, Hao Tang*, Weiyao Wang*, Michelle Guo, Thibaut Hardin, Xiang
Anushka Sagar, Bowen Song°, Xiaodong Wang, Jianing Yang°, Bo
Feiszli†§, Jitendra Malik†§

Meta Superintelligence Labs
*Core Contributor (Alphabetical, Equal Contribut

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

Demo: https://www.aidemos.meta.com/se
Code: https://github.com/facebookresear
Website: https://ai.meta.com/sam3d

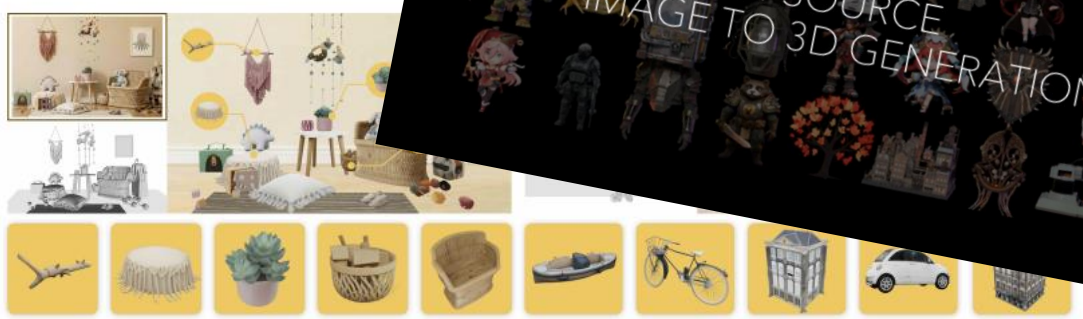


Figure 1 SAM 3D converts a single image into a composable 3D scene made of individual objects. Our method predicts per-object geometry, texture, and layout, enabling full scene reconstruction. Bottom: high-quality 3D assets recovered for each object.

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

As of Today (March 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

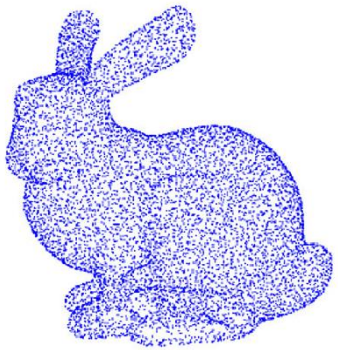
- The best open-source model is **TRELLIS 2**: https://microsoft.github.io/TRELLIS.2/
  - Reconstructs from object-centric images but produces more detailed geometry

3D Computer Vision – Outline

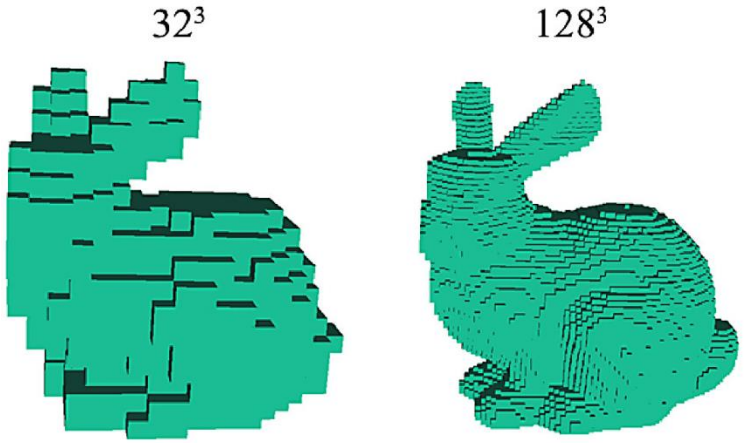
- Part 1 – Camera Model & Projection
- Part 2 – Multi-view Geometry
- Part 3 – Learning-based 3D Reconstruction
- Part 4 – 3D Representations & Rendering

Part 4 – 3D Representations & Rendering

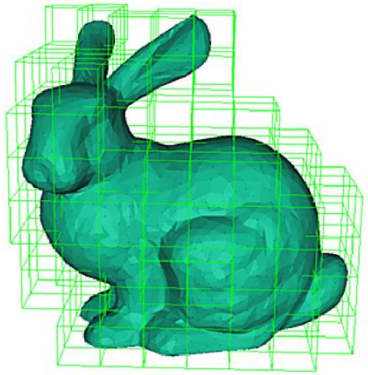
3D Shape Representations



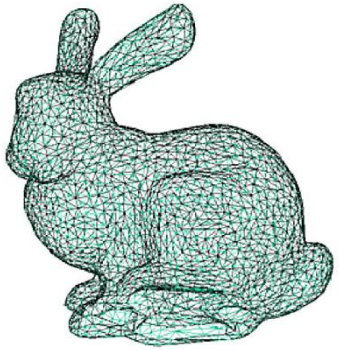
point cloud



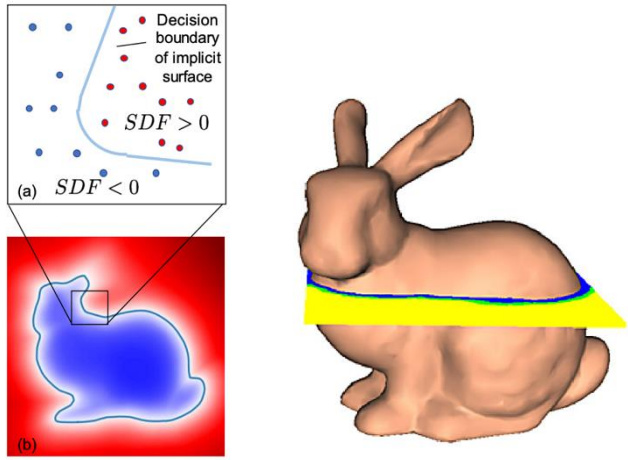
voxels



octree



mesh



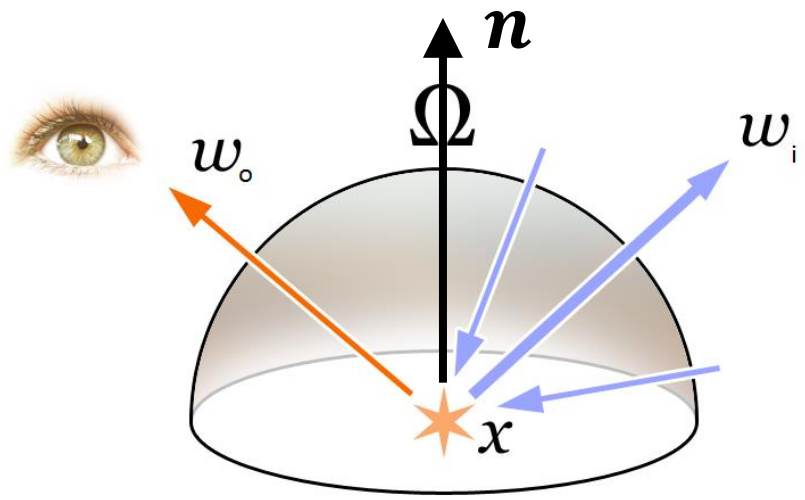
signed distance function (SDF) / level set

Illustration Source: G. Fahim 117

The Rendering Equation

- How much light (of wavelength λ) is leaving a point x in the direction of ω_o at time t?

$$L_o(x, \omega_o, \lambda, t) = \underbrace{L_e(x, \omega_o, \lambda, t)}_{\text{emitted radiance (glowing things)}} + \underbrace{L_r(x, \omega_o, \lambda, t)}_{\text{reflected radiance}}$$

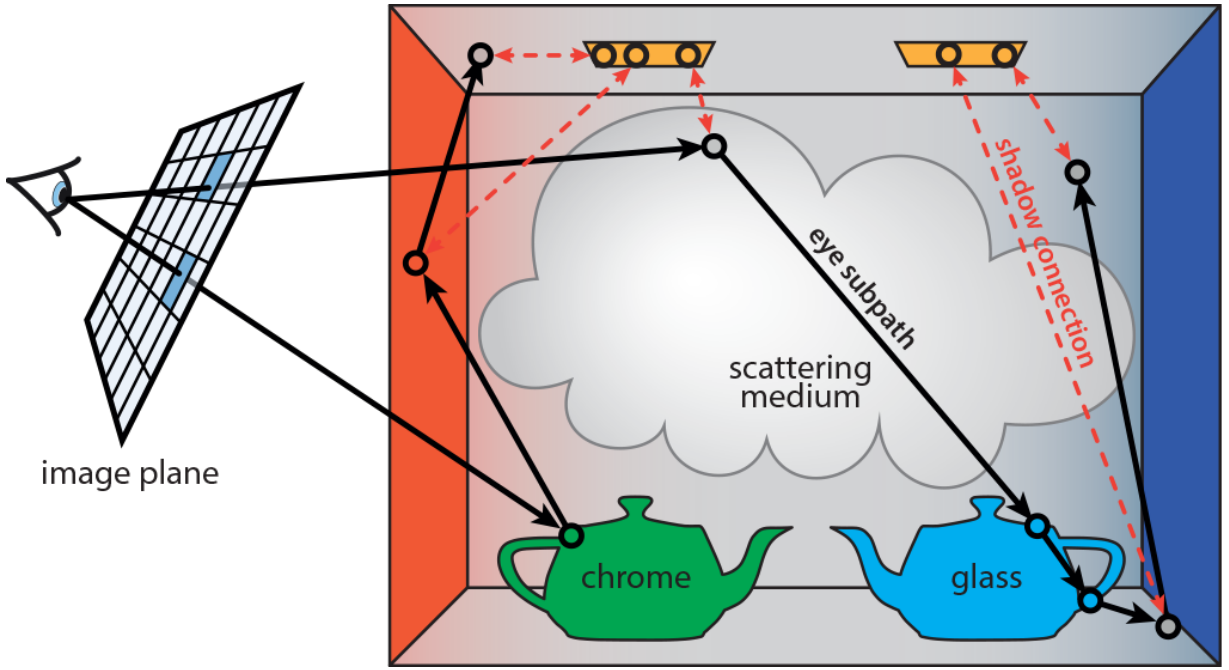


$$L_r(x, \omega_o, \lambda, t) = \int_{\Omega} \underbrace{f_i(x, \omega_i, \omega_o, \lambda, t)}_{\text{bidirectional reflectance distribution function (BRDF)}} \overbrace{L_i(x, \omega_i, \lambda, t)}^{\text{incoming radiance at } x \text{ from direction } \omega_i} (\omega_i \cdot \underbrace{\boldsymbol{n}}_{\text{surface normal}}) d\omega_i$$

D. Immel et al. SIGGRAPH. 1986.
Sources: J. Kajiya. SIGGRAPH. 1986.

Ray Tracing

- Trace the camera rays and calculate multiple bounces
- To render realistic illumination effects, we need to integrate all incoming directions at each bounce using Monte-Carlo sampling (Path Tracing)
- Expensive and complicated



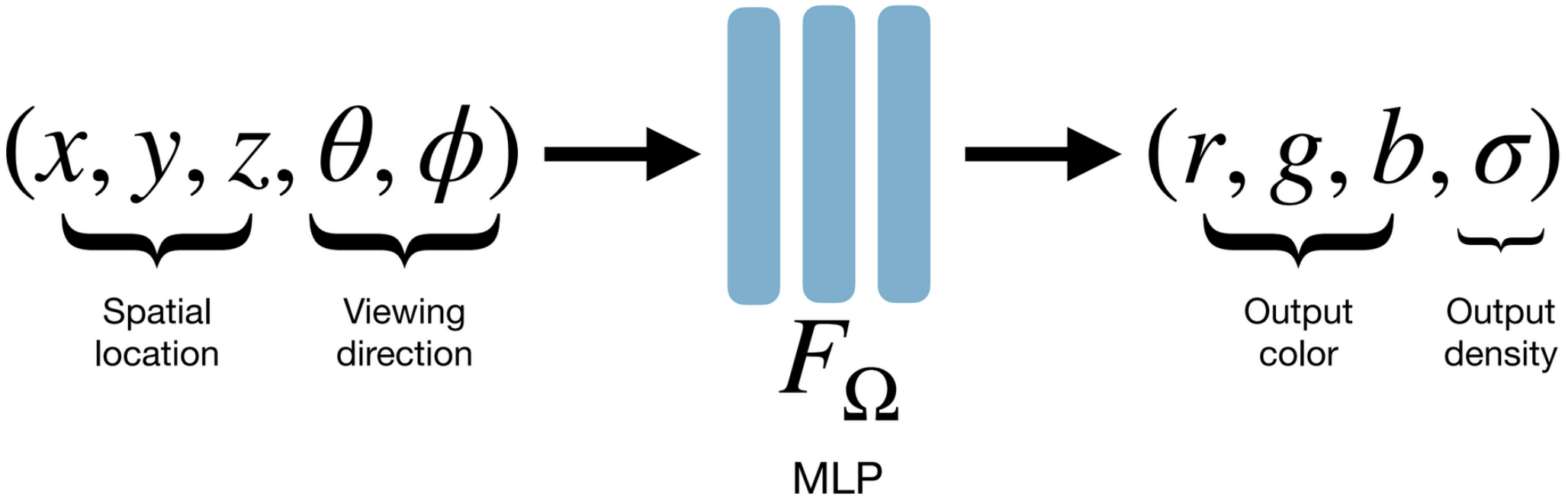
$$L_r(x, \omega_o, \lambda, t) = \int_\Omega \underbrace{f_i(x, \omega_i, \omega_o, \lambda, t)}_{\text{bidirectional reflectance distribution function (BRDF)}} \overbrace{L_i(x, \omega_i, \lambda, t)}^{\text{incoming radiance at } x \text{ from direction } \omega_i} (\omega_i \cdot \underbrace{\boldsymbol{n}}_{\text{surface normal}}) d\omega_i$$

Illustration Source: P. Christensen. 2016. 119

Ray Tracing vs Path Tracing in 6 Games. https://www.youtube.com/watch?v=lixD81ToGcg

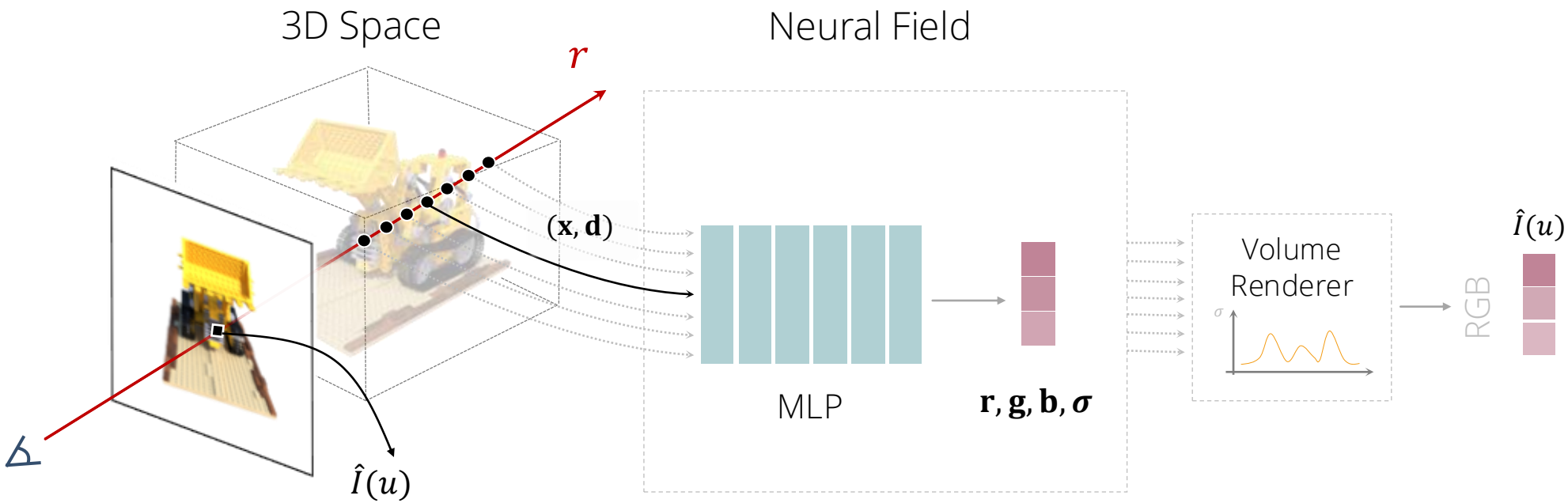
Neural Radiance Fields (NeRF)

- Representing a 3D scene as a continuous 5D function, which can be implemented as a neural network



NeRF. B. Mildenhall et al. ECCV 2020.

Neural Radiance Fields (NeRF)



NeRF. B. Mildenhall et al. ECCV 2020.
Slide Source: C. Rupprecht

Volume Rendering

- Rendering a ray $\boldsymbol{r}(t) = \boldsymbol{o} + t\boldsymbol{d}$

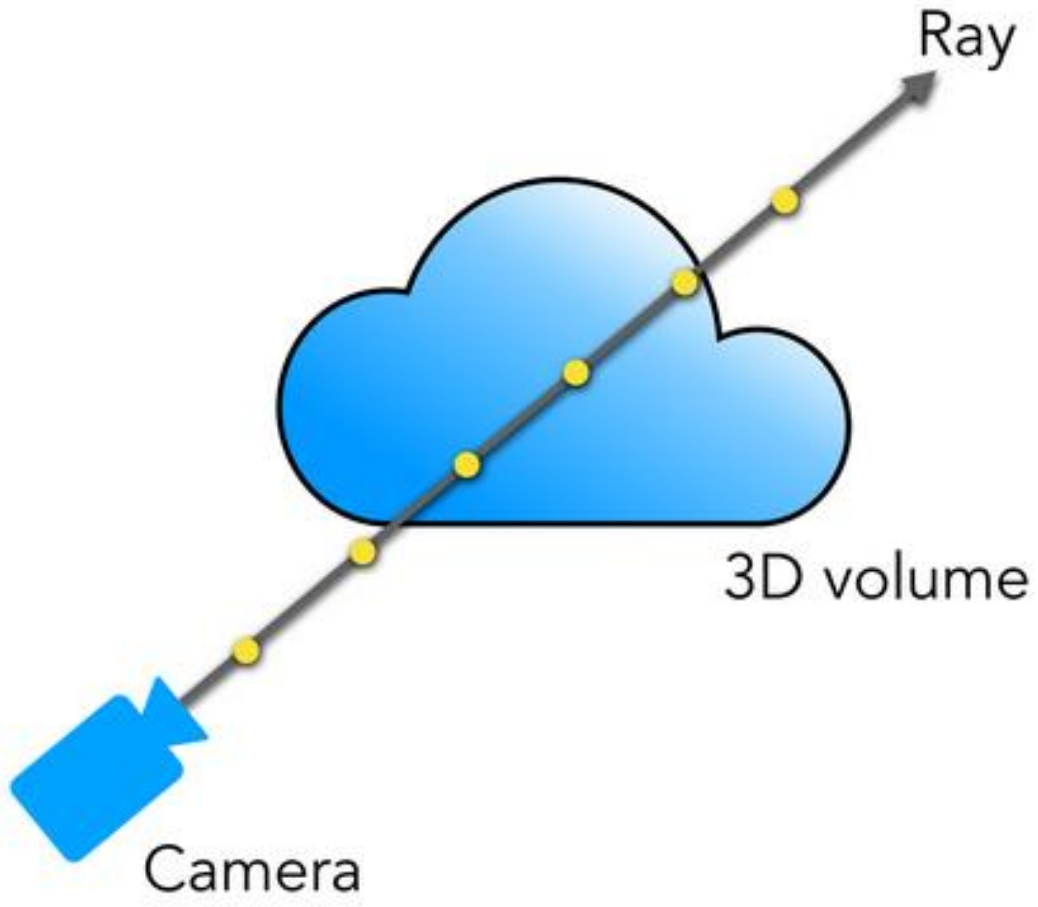
$$\boldsymbol{C} \approx \sum_{i=1}^{N} T_i \alpha_i \boldsymbol{c}_i$$

- Visibility depends on opacity before

$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_i)$$

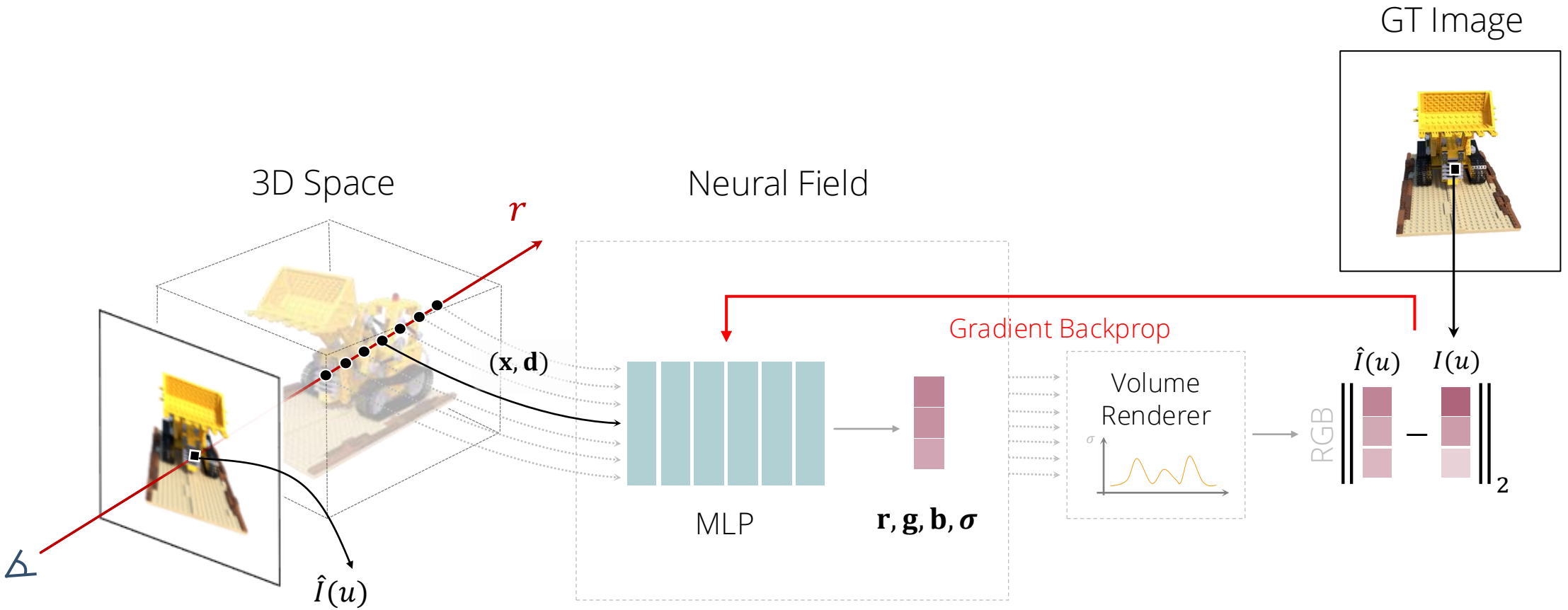
- Opacity is a function of density

$$\alpha_i = 1 - e^{-\sigma_i \Delta t}$$



NeRF. B. Mildenhall et al. ECCV 2020.

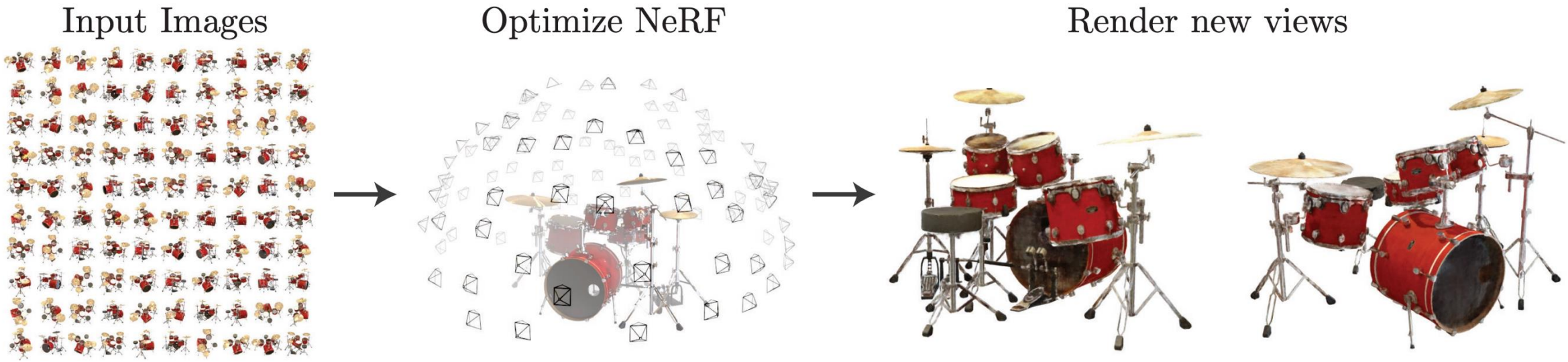
Optimized from Multi-view Images



NeRF. B. Mildenhall et al. ECCV 2020.
Slide Source: C. Rupprecht

Optimized from Multi-view Images

- Optimized on ~100 training views
- Once optimized, we can render any novel views (Novel View Synthesis)



NeRF. B. Mildenhall et al. ECCV 2020. 125







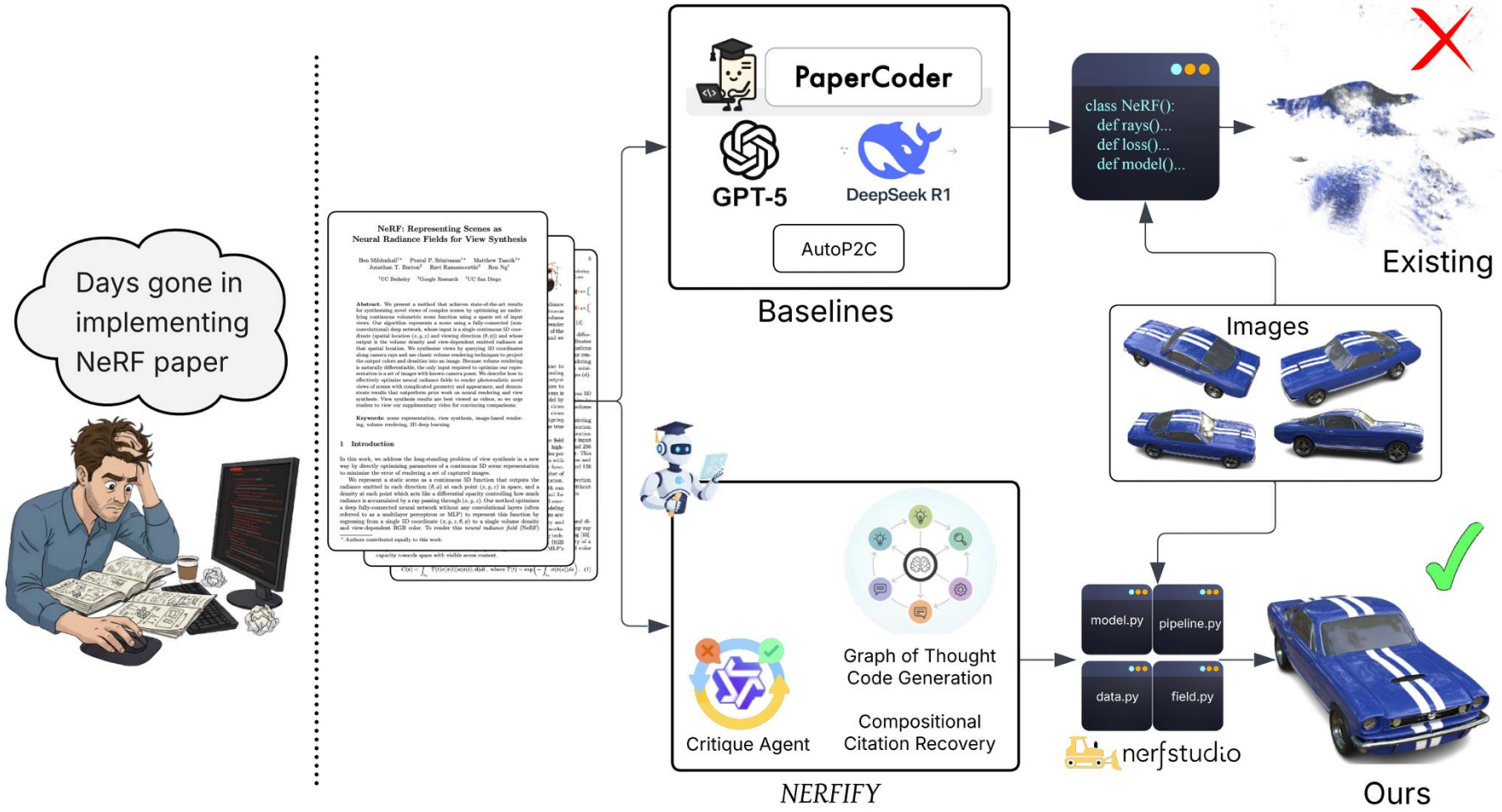

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. B. Mildenhall et al. ECCV 2020 (Best Paper Honorable Mention).

The NeRF Song

Daniel Wedge feat. Yale Song (2024): https://danielwedge.com/nerfs/

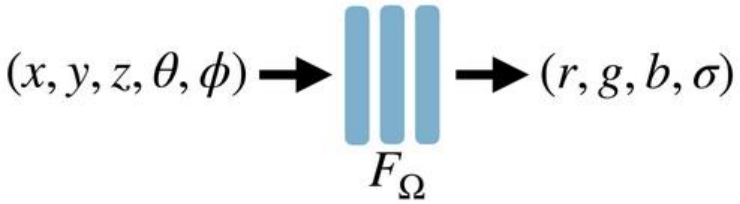
NERFIFY:

A Multi-Agent Framework for Turning NeRF Papers into Code

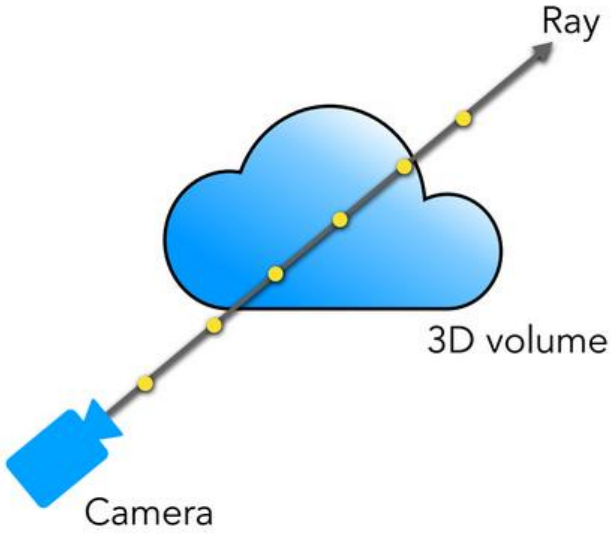


Neural Radiance Fields (NeRF)

- Three key components

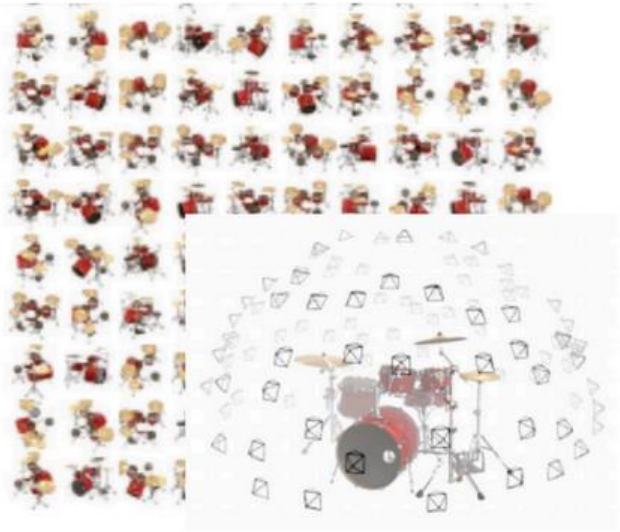


Neural Volumetric 3D Scene Representation



Differentiable Volumetric Rendering Function

Objective: Synthesize all training views

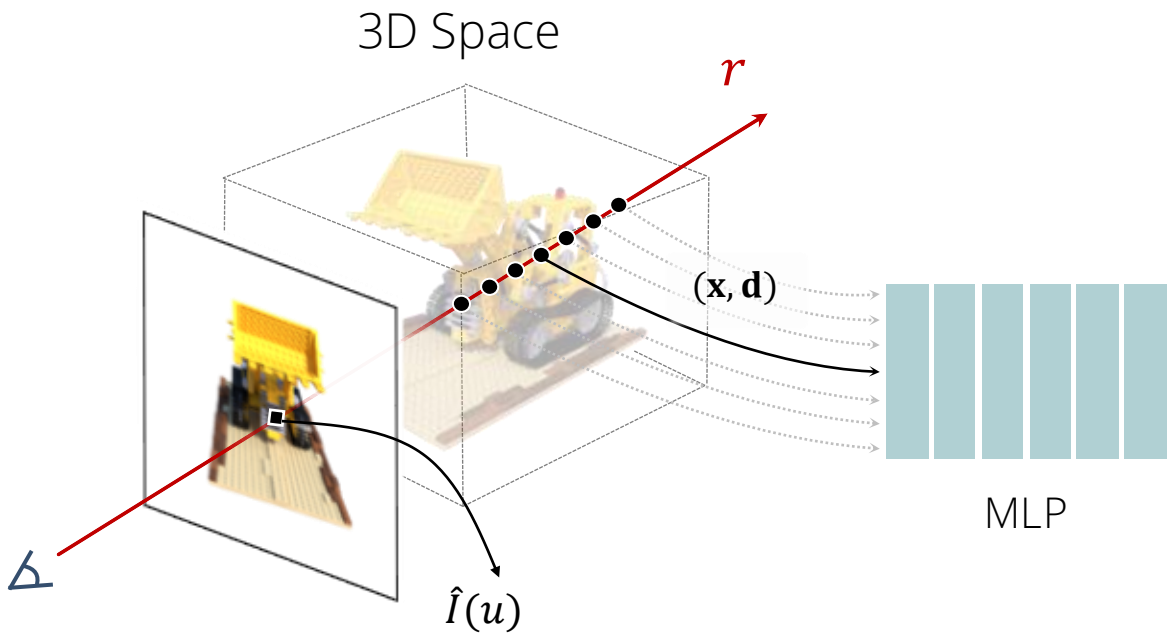


Optimization via Analysis-by-Synthesis

NeRF. B. Mildenhall et al. ECCV 2020.
Slide Source: A. Kanazawa

Volume Rendering Is Slow

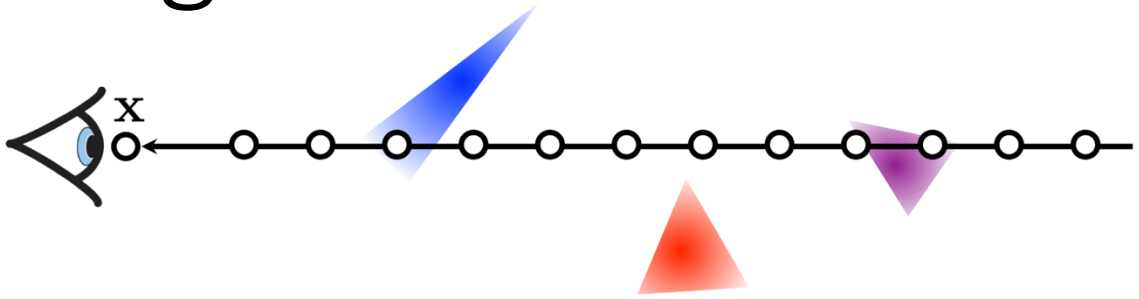
- 64+128 coarse-to-fine point samples per pixel
- ~50M forward passes through the MLP to render a 512x512 image



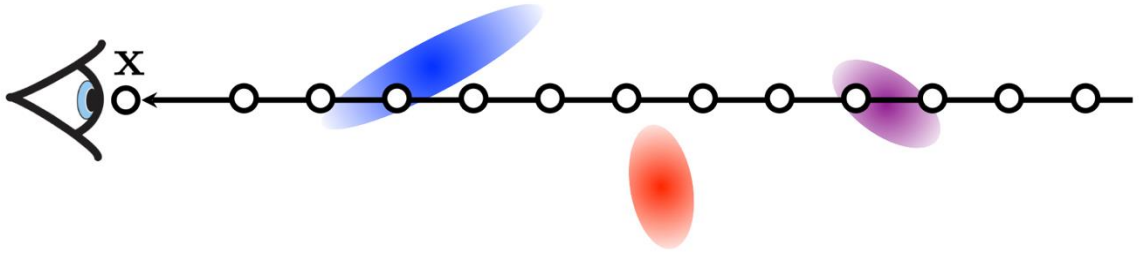
- Very slow!

Primitive Rendering

- Triangles / meshes



- 3D Gaussian blobs



- Rendering via *rasterization*: finding intersections (splatting the primitives onto the image)

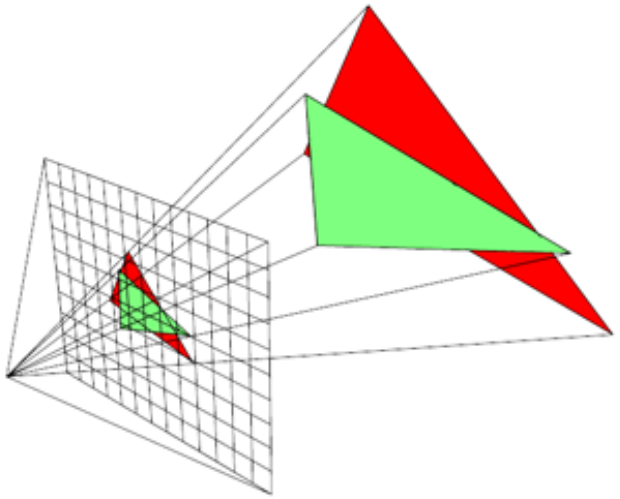
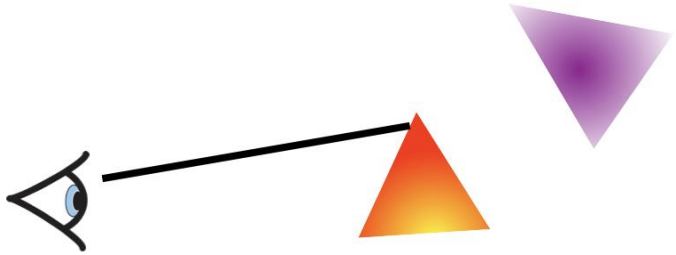


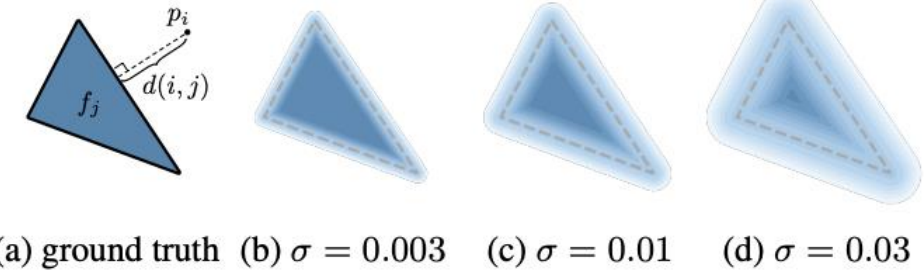
Illustration Sources: S. Tulsiani, Scratchapixel 131

Differentiable Mesh Rendering

- Discontinuity w.r.t. occlusion/dis-occlusion

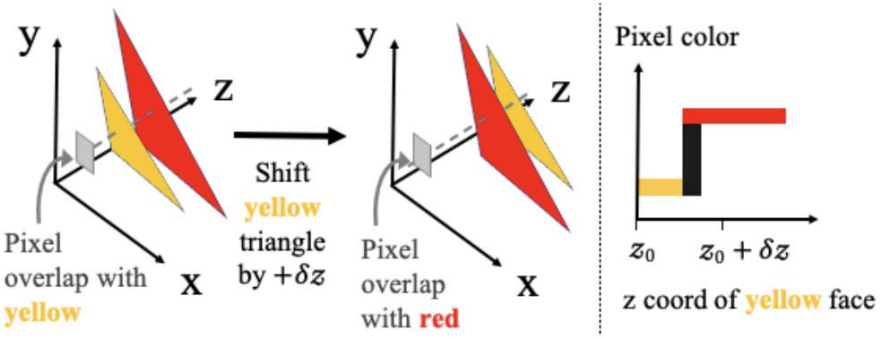


blur the edges

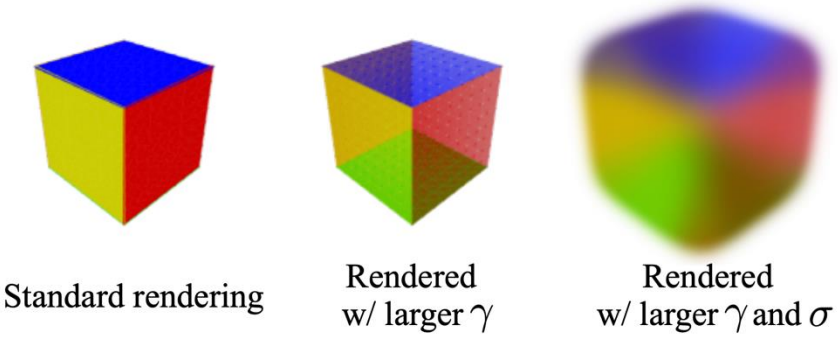


(a) ground truth (b) $\sigma = 0.003$ (c) $\sigma = 0.01$ (d) $\sigma = 0.03$

- Discontinuity w.r.t. depth ordering

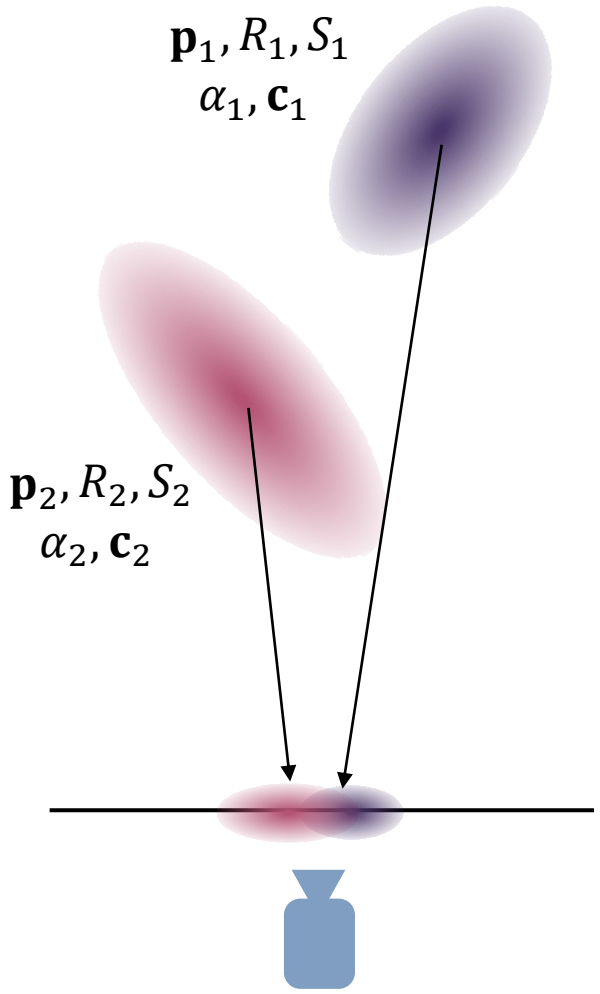


make it transparent



Standard rendering | Rendered w/ larger γ | Rendered w/ larger γ and σ

Differentiable 3D Gaussian Splatting



- Anisotropic 3D Gaussians

$$\mathcal{G}_{\mathbf{V}}(\mathbf{x}-\mathbf{p}) = \frac{1}{2\pi|\mathbf{V}|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\mathbf{p})^T \mathbf{V}^{-1}(\mathbf{x}-\mathbf{p})}$$

- Covariance matrix $\mathbf{V}$ can be factorized into scale S and rotation R

$$\mathbf{V} = RSS^TR^T \qquad S = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} \quad R \in SO(3)$$

- Each Gaussian also has an opacity α and view-dependent color $\mathbf{c}$ (using Spherical Harmonics)
- Approximate projections as 2D Gaussians -> **closed-form** volume rendering integral without sampling!
  - Need to sort the Gaussians by (centroid) depth order

Appendix: Projecting 3D Gaussians in 2D

Updating Gaussians under affine transforms: $\phi(\mathbf{x}) = M\mathbf{x} + \mathbf{t}$

$$\mathcal{G}_{\mathbf{V}}(\Phi^{-1}(\mathbf{u}) - \mathbf{p}) = \frac{1}{|\mathbf{M}^{-1}|}\mathcal{G}_{\mathbf{M}\mathbf{V}\mathbf{M}^T}(\mathbf{u} - \Phi(\mathbf{p}))$$

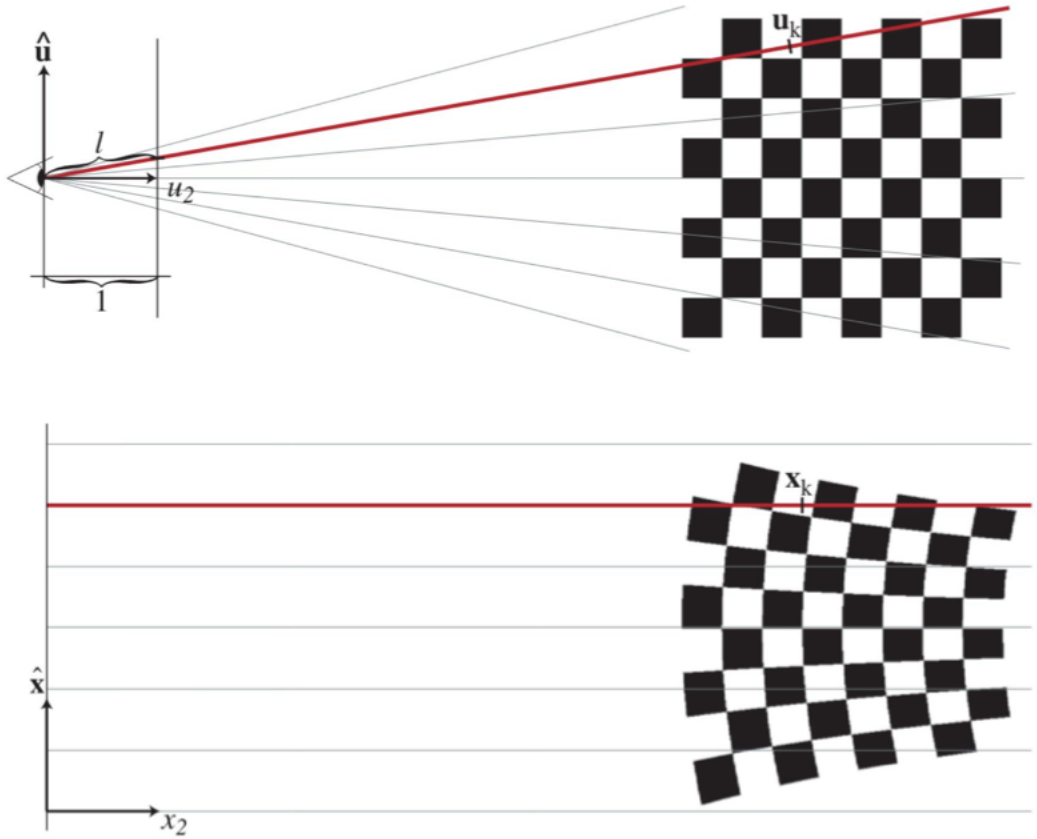
Integration along an axis:

$$\int_{\mathbb{R}} \mathcal{G}^3_{\mathbf{V}}(\mathbf{x} - \mathbf{p})\, dx_2 = \mathcal{G}^2_{\hat{\mathbf{V}}}(\hat{\mathbf{x}} - \hat{\mathbf{p}})$$

$$\mathbf{V} = \begin{pmatrix} a & b & c \\ b & d & e \\ c & e & f \end{pmatrix} \Leftrightarrow \begin{pmatrix} a & b \\ b & d \end{pmatrix} = \hat{\mathbf{V}}$$

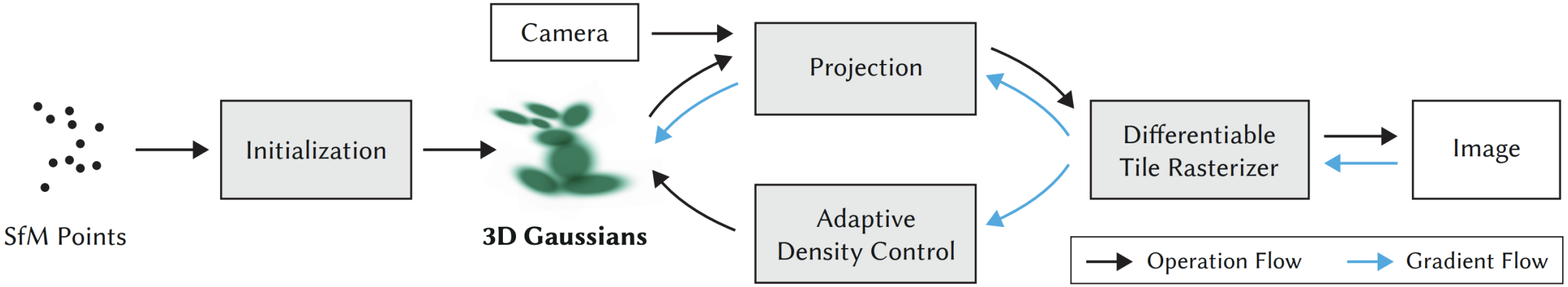
Approximate projection as an affine transform:

$$\mathbf{m}_{\mathbf{u}_k}(\mathbf{u}) = \mathbf{x}_k + \mathbf{J}_{\mathbf{u}_k} \cdot (\mathbf{u} - \mathbf{u}_k) \qquad \mathbf{J}_{\mathbf{u}_k} = \frac{\partial \mathbf{m}}{\partial \mathbf{u}}(\mathbf{u}_k)$$

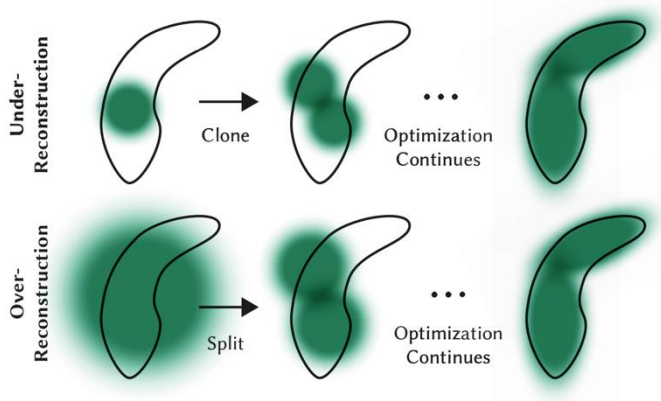


Slides thanks to Ioannis Gkioulekas & Shubham Tulsiani

3D Gaussian Splatting



- Initialize Gaussians with 3D point clouds from SfM
- Prune and densify Gaussians during optimization



3D Gaussian Splatting for Real-Time Radiance Field Rendering. B. Kerbl et al. SIGGRAPH 2023 (Best Paper).

3D Gaussian Splatting

rendered image

3D Gaussian Blobs

Source: https://blog.42yeah.is/rendering/opengl/2023/12/20/rasterizing-splats.html

NeRF & 3DGS – Limitations

- Requires dense training view coverage
- Relies on careful initialization of camera poses (and Gaussians for 3DGS)
  - Additional supervision can help, such as depth (Depth-supervised NeRF, NoPe-NeRF), flow (FlowCam)
- Ongoing research directions:
  - Feed-forward view synthesis: Large View Synthesis Model
  - Self-supervised training (no pose, using videos): RayZer, E-RayZer
  - Generative view synthesis (with diffusion models): Zero-1-to-3, Stable Virtual Camera
  - Extracting explicit geometry (surfaces, volumes): 2DGS, SuGaR, Radiance Mesh
  - Dynamic 3D scenes / editing: D-NeRF, Dynamic 3D Gaussians, Gaussian Frosting

Marble: A Multimodal World Model. WorldLabs. 2025.

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.

3D or Not 3D?

- Defending intermediate representations in computer vision, by Srinath Sridhar: https://ivl.cs.brown.edu/blogs/bittersweet_lesson_cv

Brown IVL

February 25, 2026

The flavor of the bittersweet lesson for computer vision

Srinath Sridhar

Vincent Sitzmann recently shared a blog post titled "The flavor of the bitter lesson for computer vision." In it, he argues that computer vision as we currently know it may become obsolete, with its remaining elements being absorbed into end-to-end perception-action models. His post is timely, and reflects ongoing discussions in the community.

In this article, I will examine some of Vincent's arguments, and provide an alternative perspective. Others have made overlapping points on this X post, which I highly recommend reading.

Is action the sole purpose of computer vision?

The post suggests that embodied intelligence, specifically action, is the sole purpose of computer vision. But in reality, computer vision has a wide range of applications.

Consider the following use cases, none of which require first-order action execution:

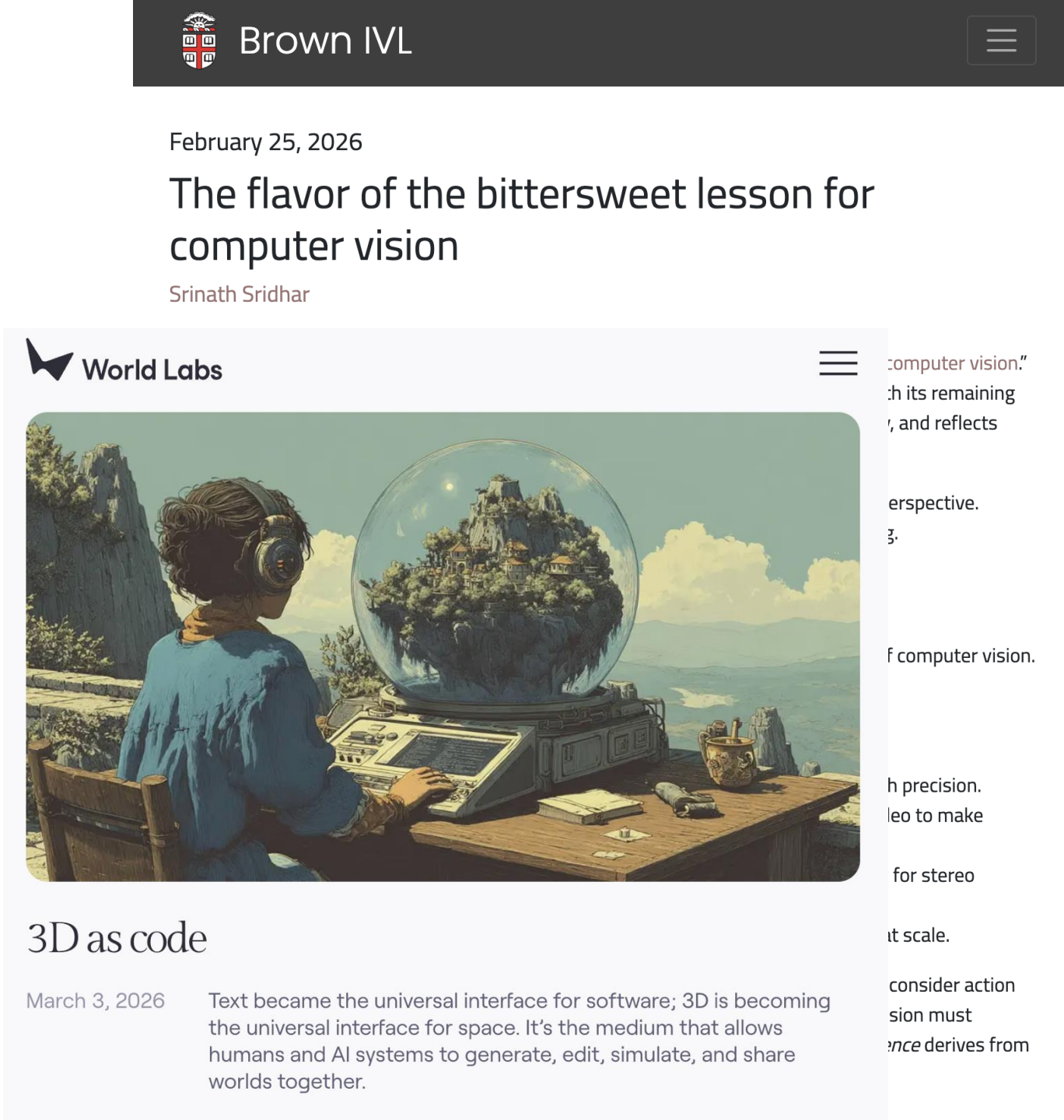
- Metrology: Engineers use computer vision to measure the physical world with high precision.
- Media & Entertainment: Creators leverage tools like face filters and generative video to make media content.
- Virtual Reality: VR artists utilize vision to create 3D content with accurate parallax for stereo headsets.
- Cognitive Science: Researchers apply vision to code and analyze animal behavior at scale.

More broadly, if computer vision is to remain a rigorous scientific discipline, we cannot consider action execution alone, or mandate action as a pre-training task for all problems. Computer vision must provide an action-agnostic understanding of the physical world. After all, the word *science* derives from the Latin *scientia*, meaning "knowledge, awareness, understanding."

3D or Not 3D?

- Defending intermediate representations in computer vision, by Srinath Sridhar: https://ivl.cs.brown.edu/blogs/bittersweet_lesson_cv

- “3D as the universal interface for space” from WorldLabs: https://www.worldlabs.ai/blog/3d-as-code



Part 3&4 Summary – Learning & Rendering

- Learning-based 3D reconstruction
  - 3D point maps: DUSt3R, VGGT
  - Object-centric: Hunyuan3D, SAM3D, TRELLIS
  - The emergence of large-scale 3D datasets is the main driver
- 3D representations and rendering
  - Classical 3D representations
  - The traditional rendering equation
  - Neural Radiance Field (NeRF)
  - Differentiable rendering
  - 3D Gaussian Splatting (3DGS)
- Ongoing research

The Era of Large Models and Expanding Horizons

– Applications and Challenges –

Elliott Wu