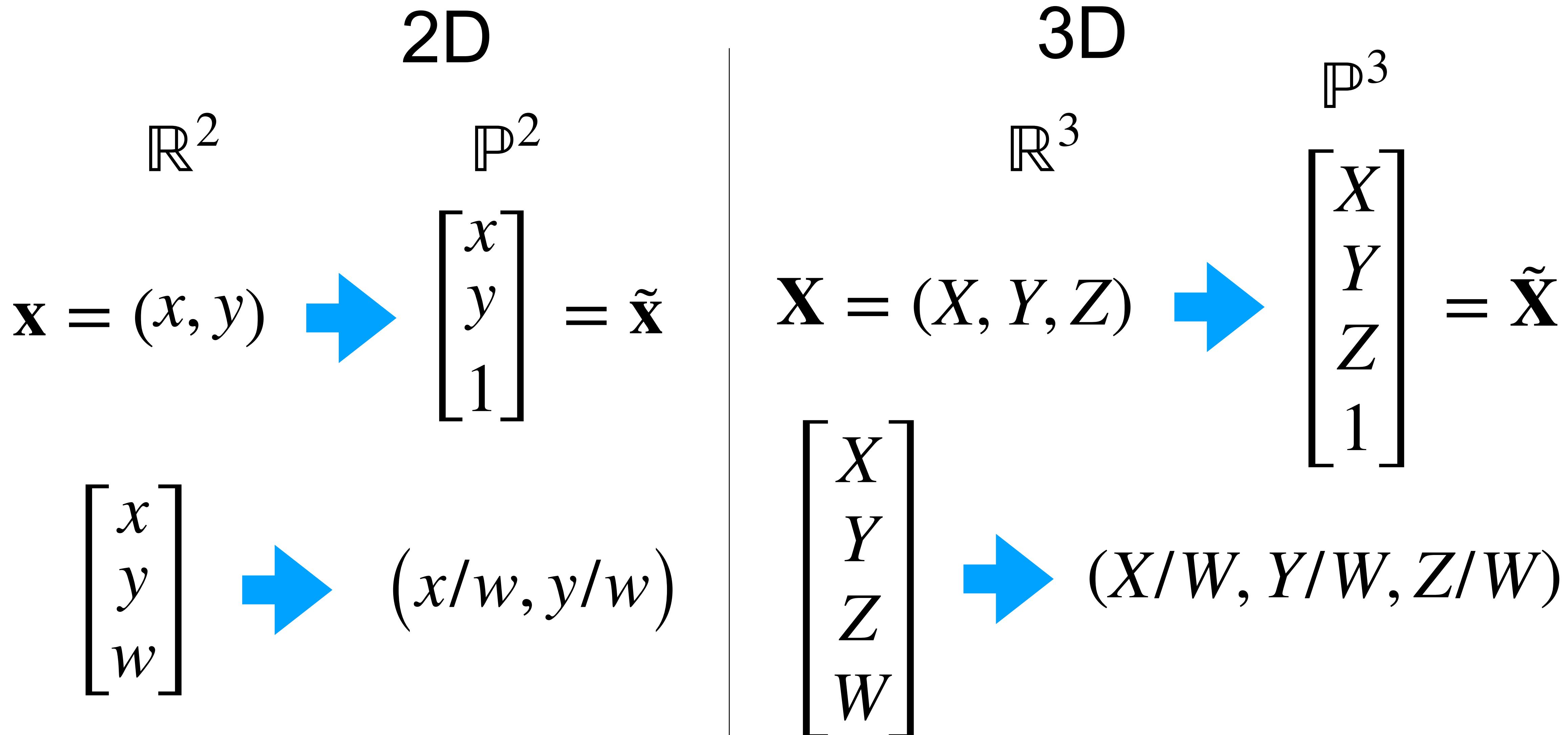


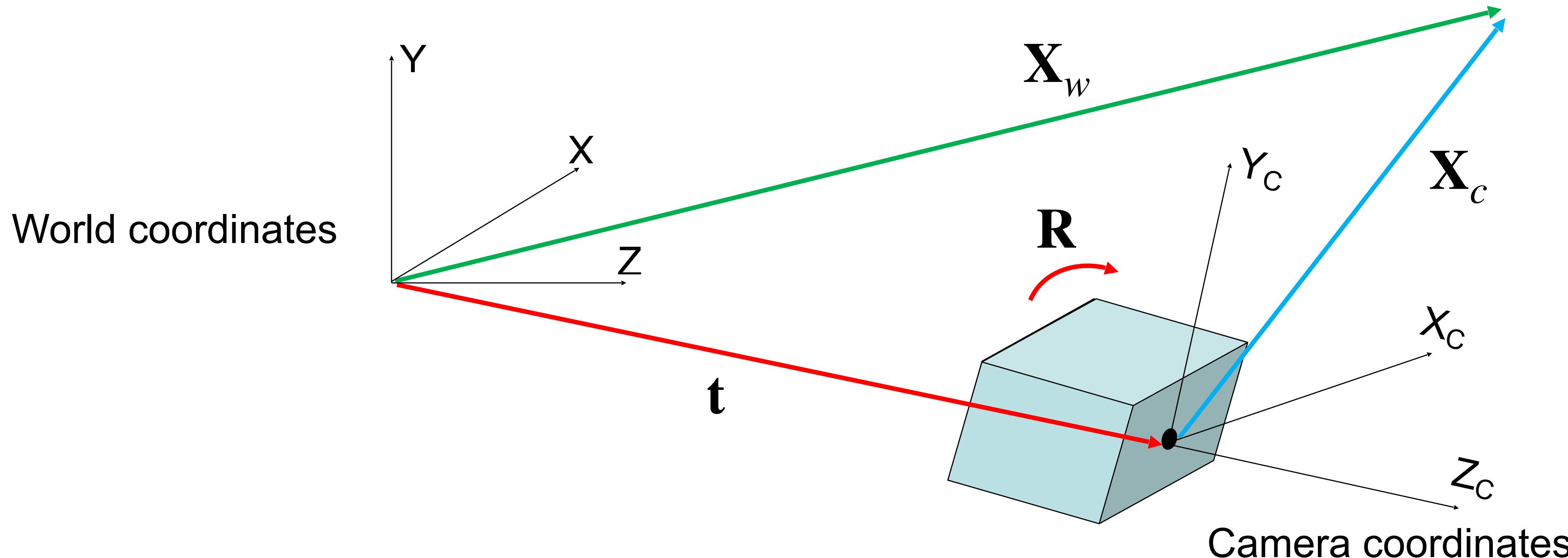
MULTI-VIEW GEOMETRY

6.S980

Recap: Hom. Coordinates



Recap: Camera parameters



World coordinates to camera coordinates

$$\tilde{\mathbf{X}}_c = \begin{bmatrix} \mathbf{R} & -\mathbf{R}\mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \tilde{\mathbf{X}}_w$$

Camera coordinates to image coordinates

$$\tilde{\mathbf{x}} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tilde{\mathbf{X}}_c$$

Camera parameters

World coordinates to camera coordinates

$$\tilde{\mathbf{X}}_c = \begin{bmatrix} \mathbf{R} & -\mathbf{R}\mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \tilde{\mathbf{X}}_w$$

$$\tilde{\mathbf{X}}_c = \mathbf{C}^{W2C} \tilde{\mathbf{X}}_w$$

Camera coordinates to image coordinates

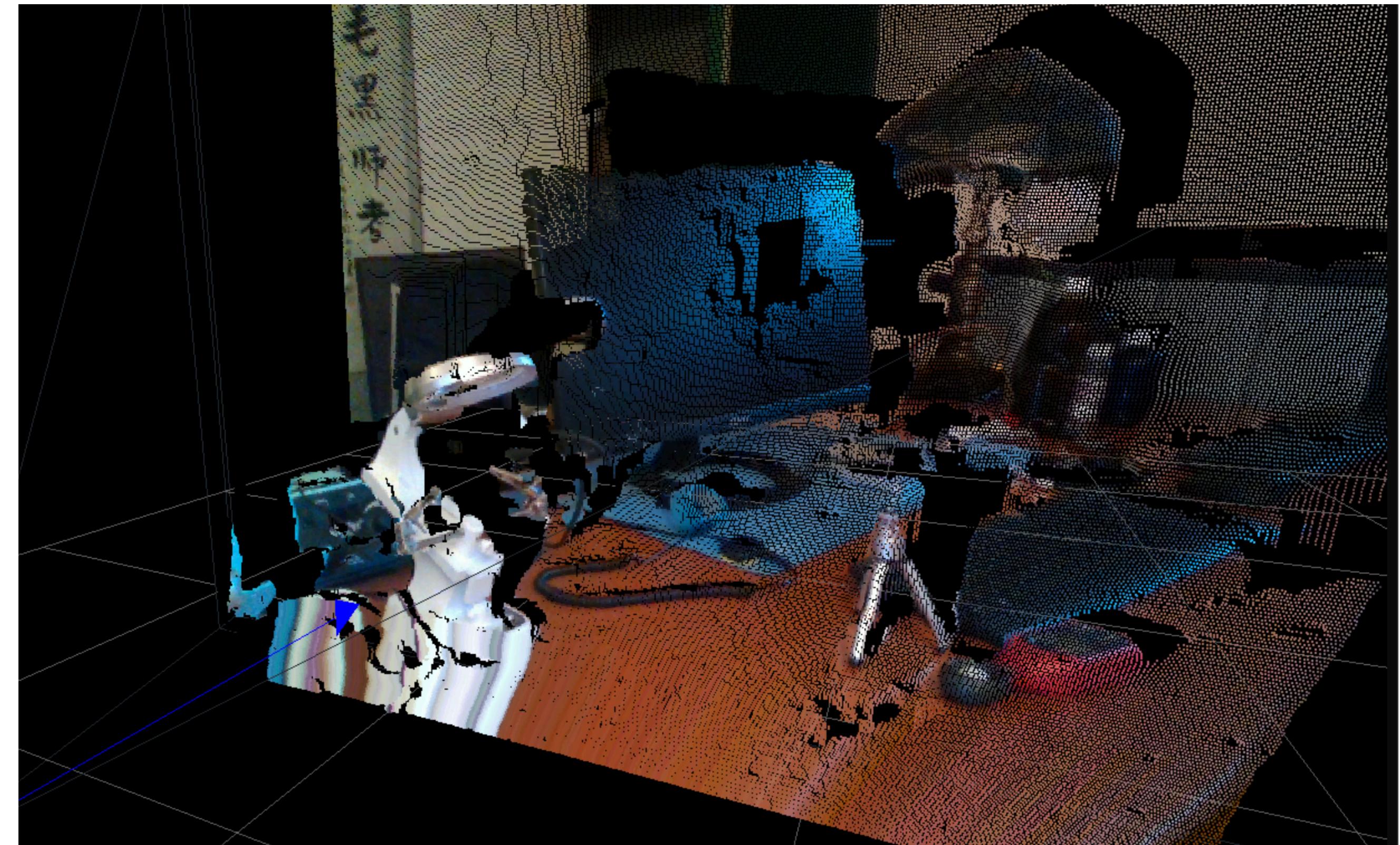
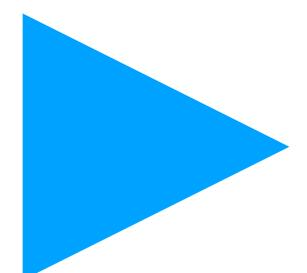
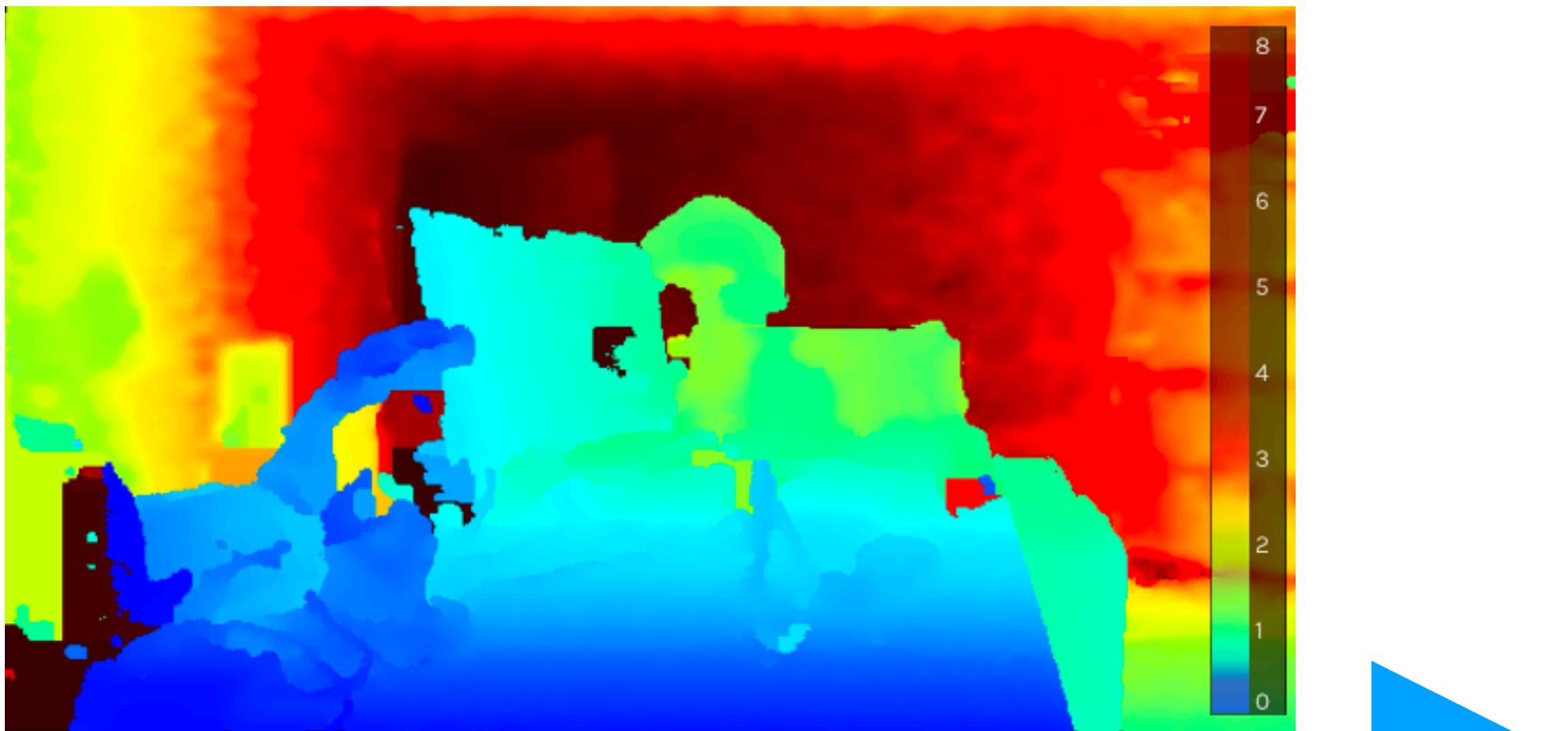
$$\tilde{\mathbf{x}} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tilde{\mathbf{X}}_c$$

$$\tilde{\mathbf{x}} = \mathbf{K}[\mathbf{I} | 0] \tilde{\mathbf{X}}_c$$

$$\tilde{\mathbf{x}} = \mathbf{K}[\mathbf{I} | 0] \mathbf{C}^{W2C} \tilde{\mathbf{X}}_w$$

$$\tilde{\mathbf{x}} = \mathbf{P} \tilde{\mathbf{X}}_w$$

“Unproject”: Go from pixel and depth to 3D point



“Unproject”: Go from pixel and depth to 3D point

$$\begin{pmatrix} f \cdot X + Z \cdot p_x \\ f \cdot Y + Z \cdot p_y \\ Z \end{pmatrix} = \mathbf{K} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

“Unproject”: Go from pixel and depth to 3D point

$$\begin{pmatrix} f \cdot X + Z \cdot p_x \\ f \cdot Y + Z \cdot p_y \\ Z \end{pmatrix} = \mathbf{K} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \mathbf{K}^{-1} \begin{pmatrix} f \cdot X + Z \cdot p_x \\ f \cdot Y + Z \cdot p_y \\ Z \end{pmatrix}$$

“Unproject”: Go from pixel and depth to 3D point

$$\begin{pmatrix} f \cdot X + Z \cdot p_x \\ f \cdot Y + Z \cdot p_y \\ Z \end{pmatrix} = \mathbf{K} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \mathbf{K}^{-1} \begin{pmatrix} f \cdot X + Z \cdot p_x \\ f \cdot Y + Z \cdot p_y \\ Z \end{pmatrix}$$

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = Z \cdot \mathbf{K}^{-1} \begin{pmatrix} \frac{f \cdot X}{Z} + p_x \\ \frac{f \cdot Y}{Z} + p_y \\ 1 \end{pmatrix}$$

“Unproject”: Go from pixel and depth to 3D point

$$\begin{pmatrix} f \cdot X + Z \cdot p_x \\ f \cdot Y + Z \cdot p_y \\ Z \end{pmatrix} = \mathbf{K} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \mathbf{K}^{-1} \begin{pmatrix} f \cdot X + Z \cdot p_x \\ f \cdot Y + Z \cdot p_y \\ Z \end{pmatrix}$$

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = Z \cdot \mathbf{K}^{-1} \begin{pmatrix} \frac{f \cdot X}{Z} + p_x \\ \frac{f \cdot Y}{Z} + p_y \\ 1 \end{pmatrix}$$

$$\mathbf{x} = Z \cdot \mathbf{K}^{-1} \begin{pmatrix} x_{pix} \\ y_{pix} \\ 1 \end{pmatrix}$$

“Unproject”: Go from pixel and depth to 3D point

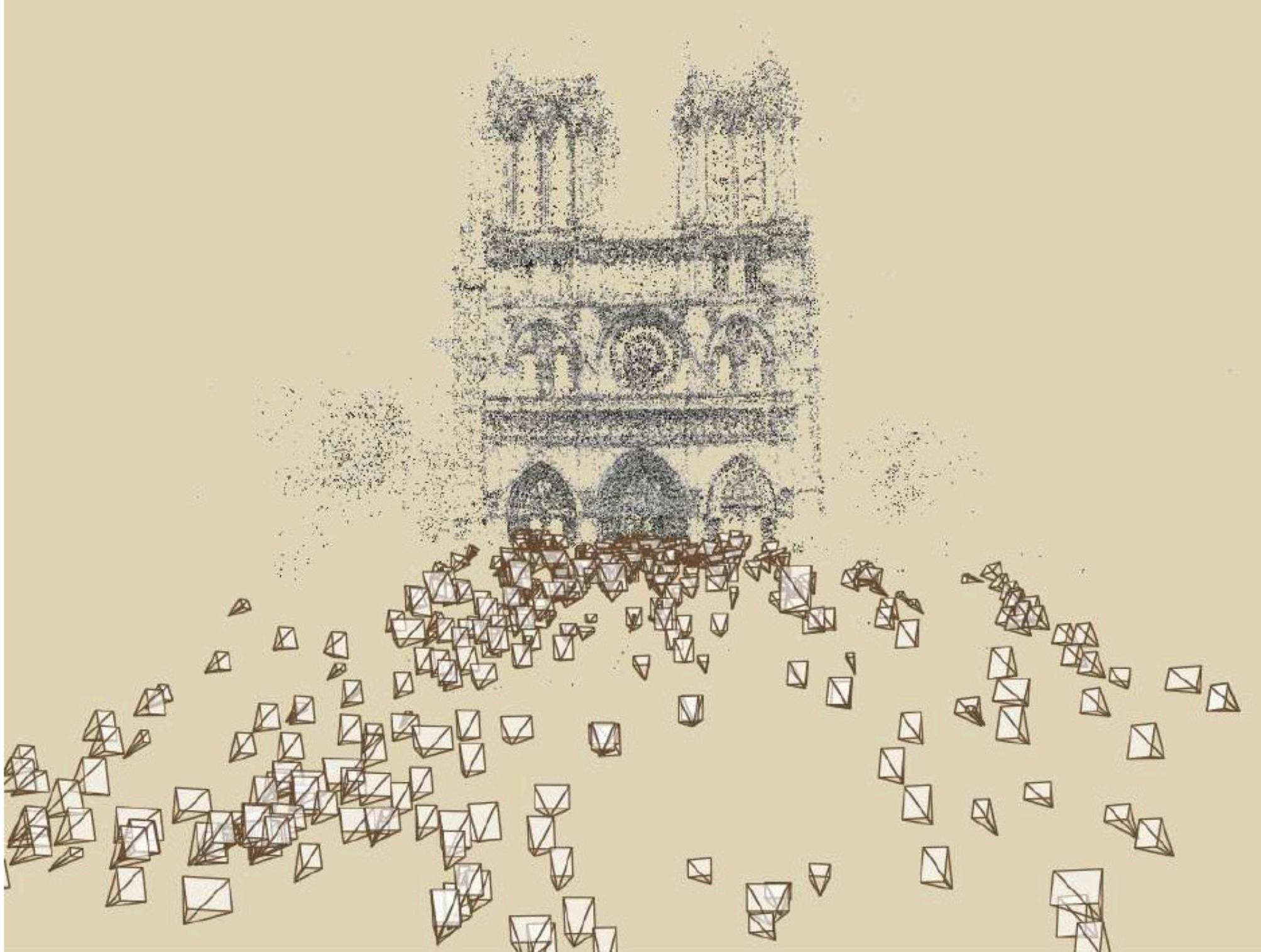
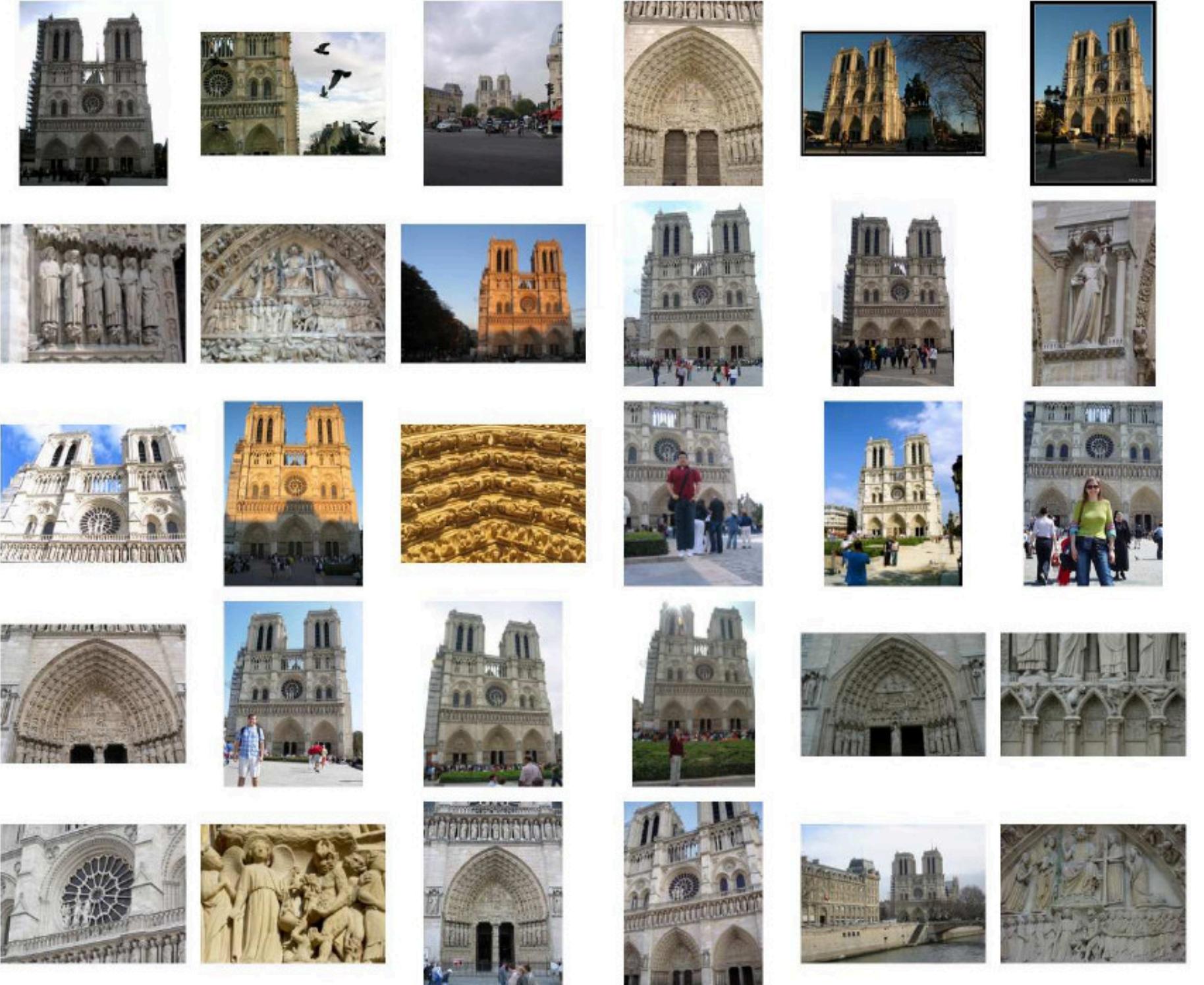
$$\mathbf{X} = Z \cdot \mathbf{K}^{-1} \begin{pmatrix} x_{pix} \\ y_{pix} \\ 1 \end{pmatrix}$$

“Unproject”: Go from pixel and depth to 3D point

$$\mathbf{X} = Z \cdot \mathbf{K}^{-1} \begin{pmatrix} x_{pix} \\ y_{pix} \\ 1 \end{pmatrix}$$

$$\text{unproject}(\mathbf{x}, Z) = Z \cdot \mathbf{K}^{-1} \begin{pmatrix} x_{pix} \\ y_{pix} \\ 1 \end{pmatrix}$$

Now: Multi-View Geometry



Why?

We want to understand 3D world only from 2D observations (images). For that, we need to have a mathematical understanding of how they are connected.

What you'll learn.

Mathematical model of cameras. Reconstruct camera poses, approximate geometry, and camera parameters from 2D images of a scene.

Some Slides adapted from...

- CMU 16-385: Computer Vision

Prof. Kris Kitani

- MIT 6.819/6.869: Advances in Computer Vision,

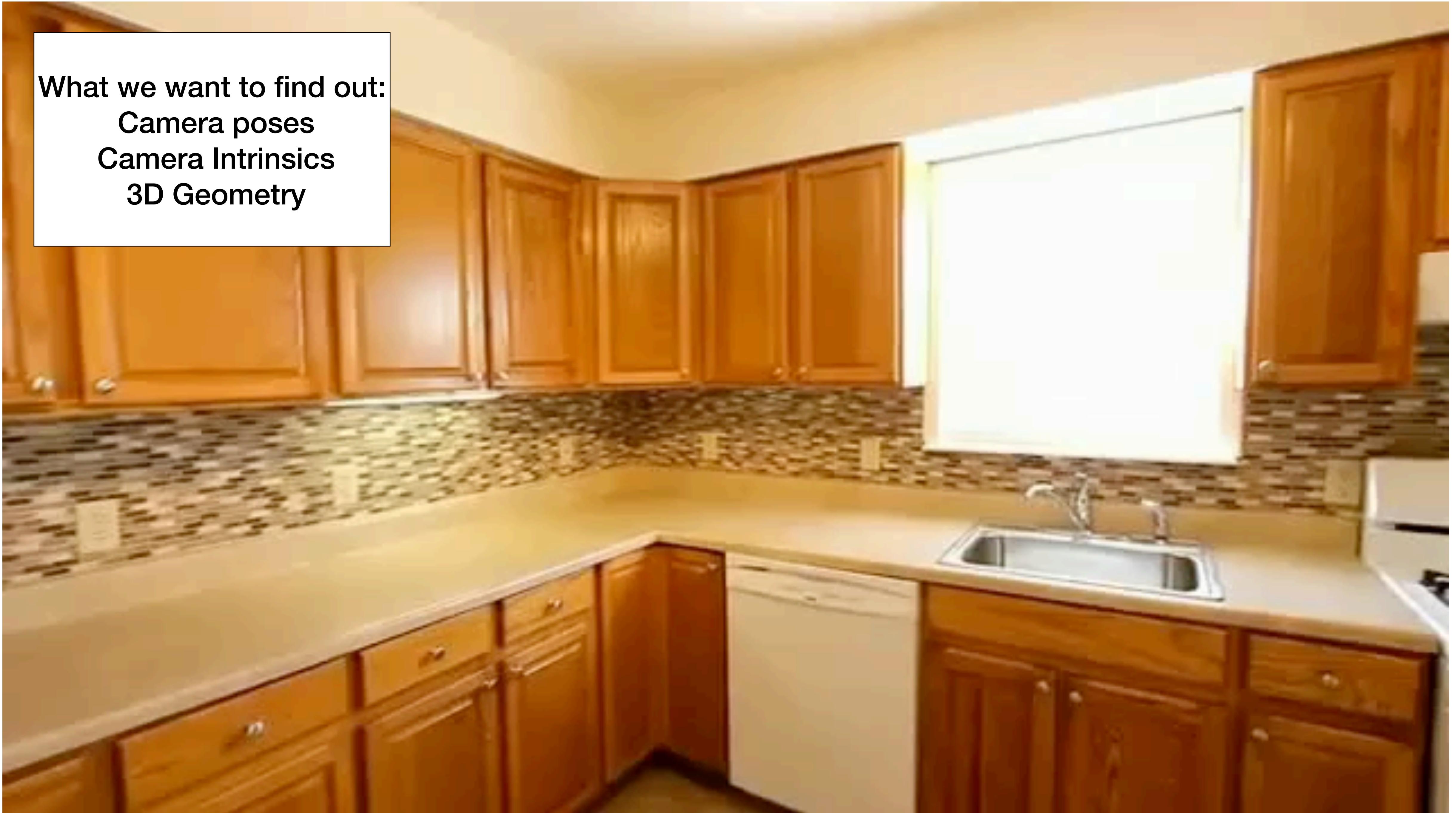
Profs. Bill Freeman, Phillip Isola, Antonio Torralba

- University of Tübingen: Computer Vision

Prof. Andreas Geiger

What we want to find out:

- Camera poses
- Camera Intrinsics
- 3D Geometry



Bundle Adjustment

Triangulation

How to compute 3D locations of point correspondences if cameras are known.

Epipolar Lines

Which pixels in two cameras observe same 3D point?

Where to look for multi-view correspondences?

Fundamental & Essential Matrices

Elegant formulation of Epipolar Lines

A way of *estimating camera poses, intrinsics, and extrinsic from correspondences.*

No Time

Correspondences
RANSAC
Incremental
Bundle
Adjustment
Practically solving
for \mathbf{F} and \mathbf{K}

Read:
Computer Vision:
Algorithms and
Applications, 2nd
ed.

What has changed since Deep Learning?

The 8-Point Algorithm as an Inductive Bias for Relative Pose Prediction by ViTs

22

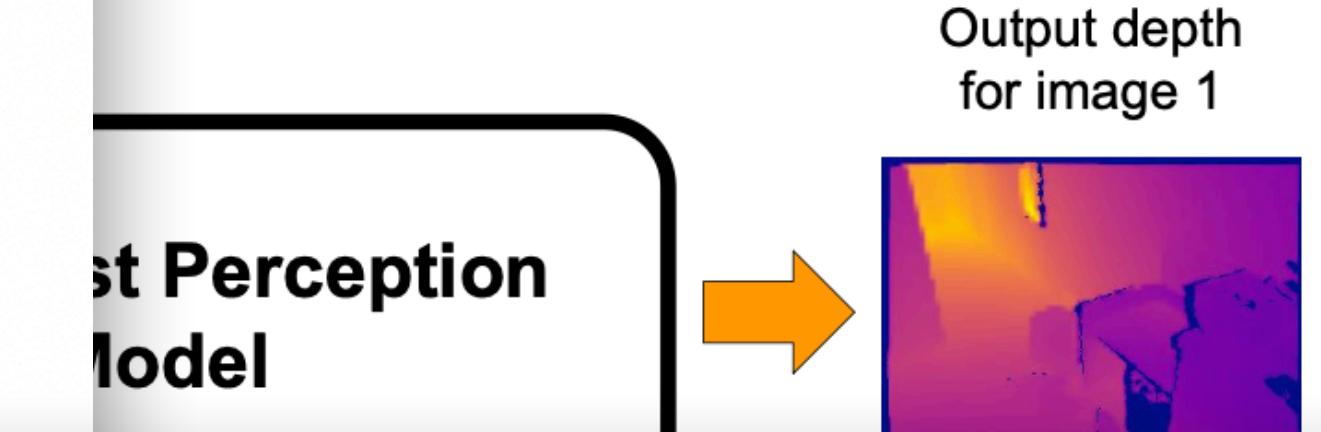
Input-level Inductive Biases for 3D Reconstruction

Wang Yifan^{1*} Carl Doersch² Relja Arandjelović² João Carreira² Andrew Zisserman^{2,3}

Science, University of Oxford

Generalizable Patch-Based Neural Rendering

Mohammed Suhail¹, Carlos Esteves⁴, Leonid Sigal^{1,2,3}, and Ameesh Makadia⁴



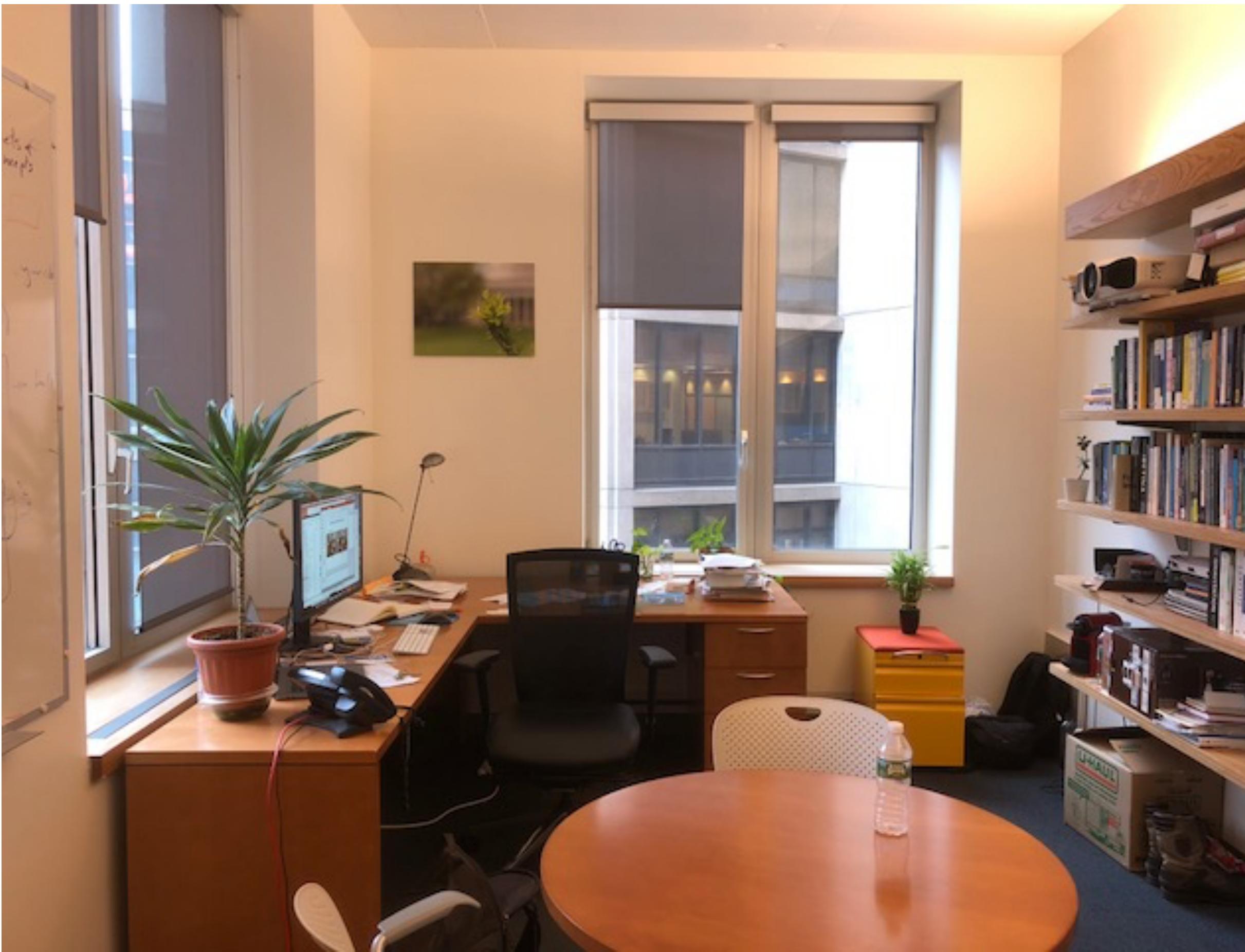
BARF 😱: Bundle-Adjusting Neural Radiance Fields

Chen-Hsuan Lin¹ Wei-Chiu Ma² Antonio Torralba² Simon Lucey^{1,3}

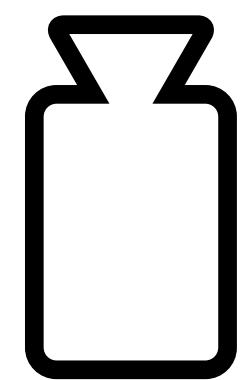
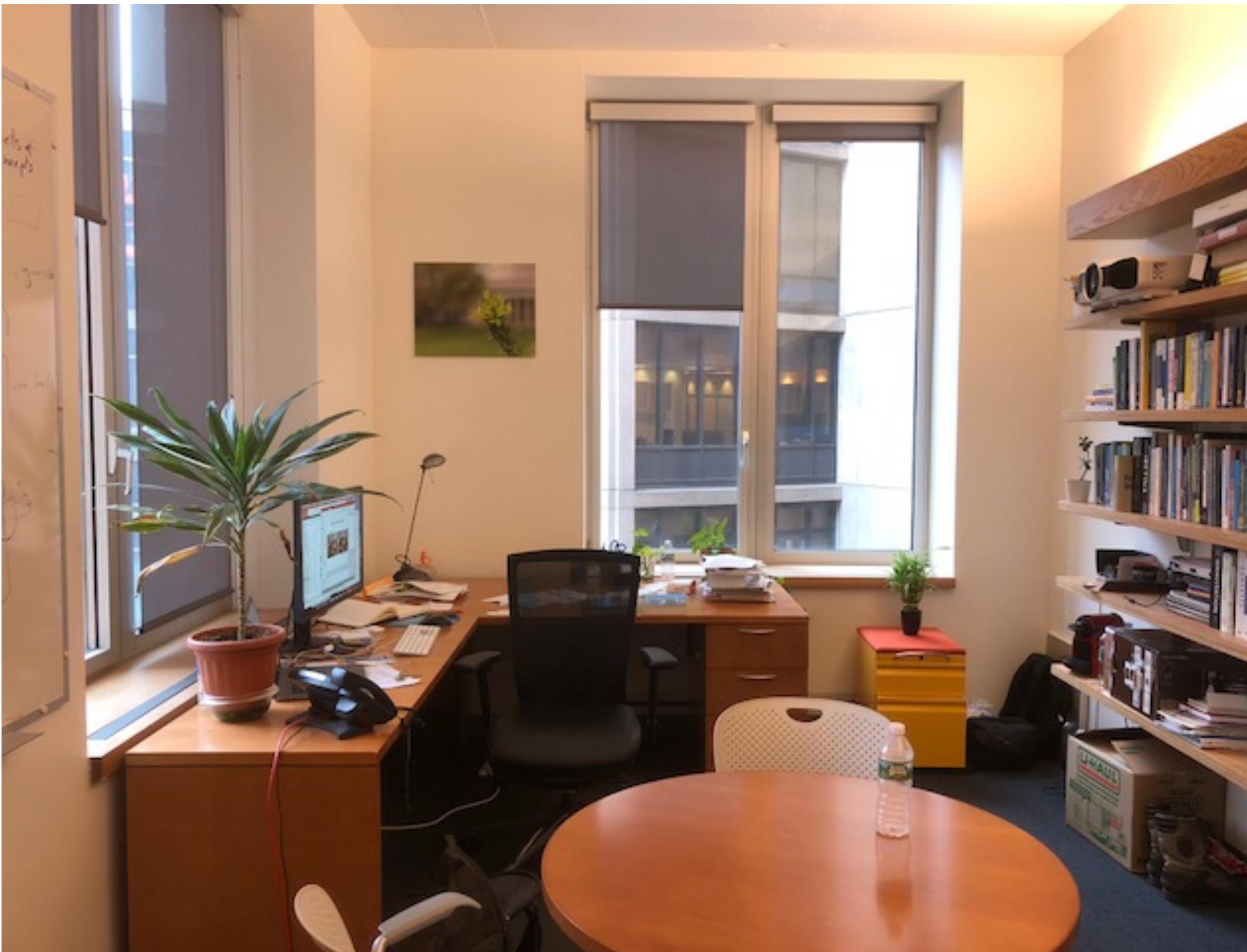
¹Carnegie Mellon University ²Massachusetts Institute of Technology ³The University of Adelaide

<https://chenhsuanlin.bitbucket.io/bundle-adjusting-NeRF>

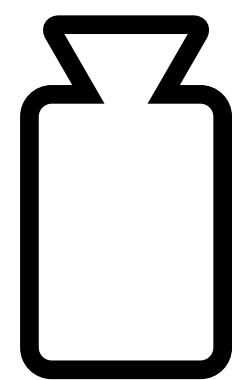


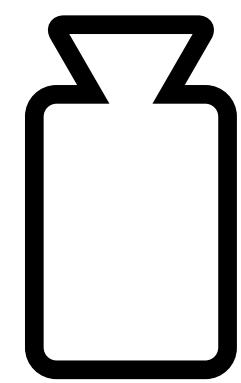
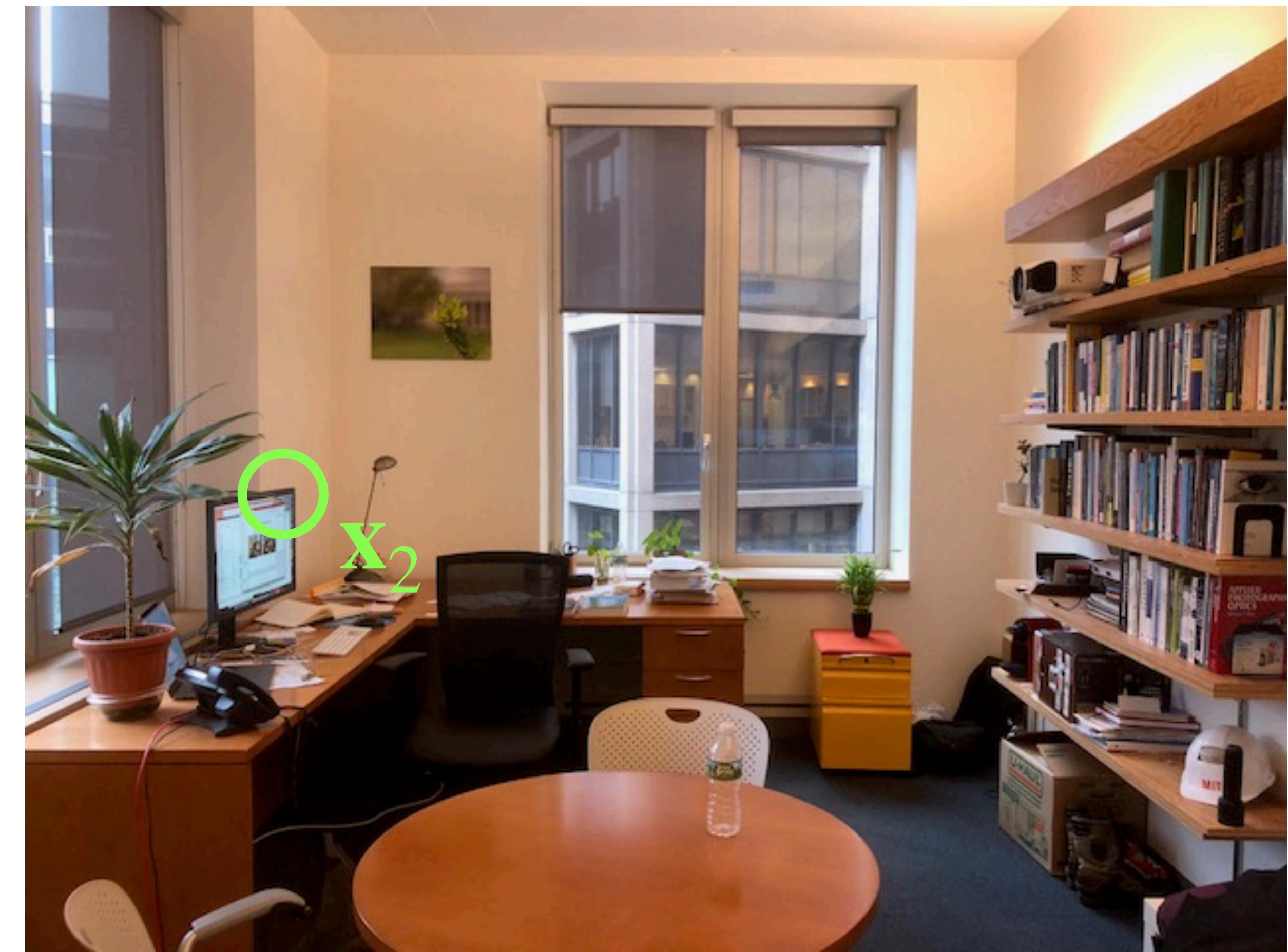
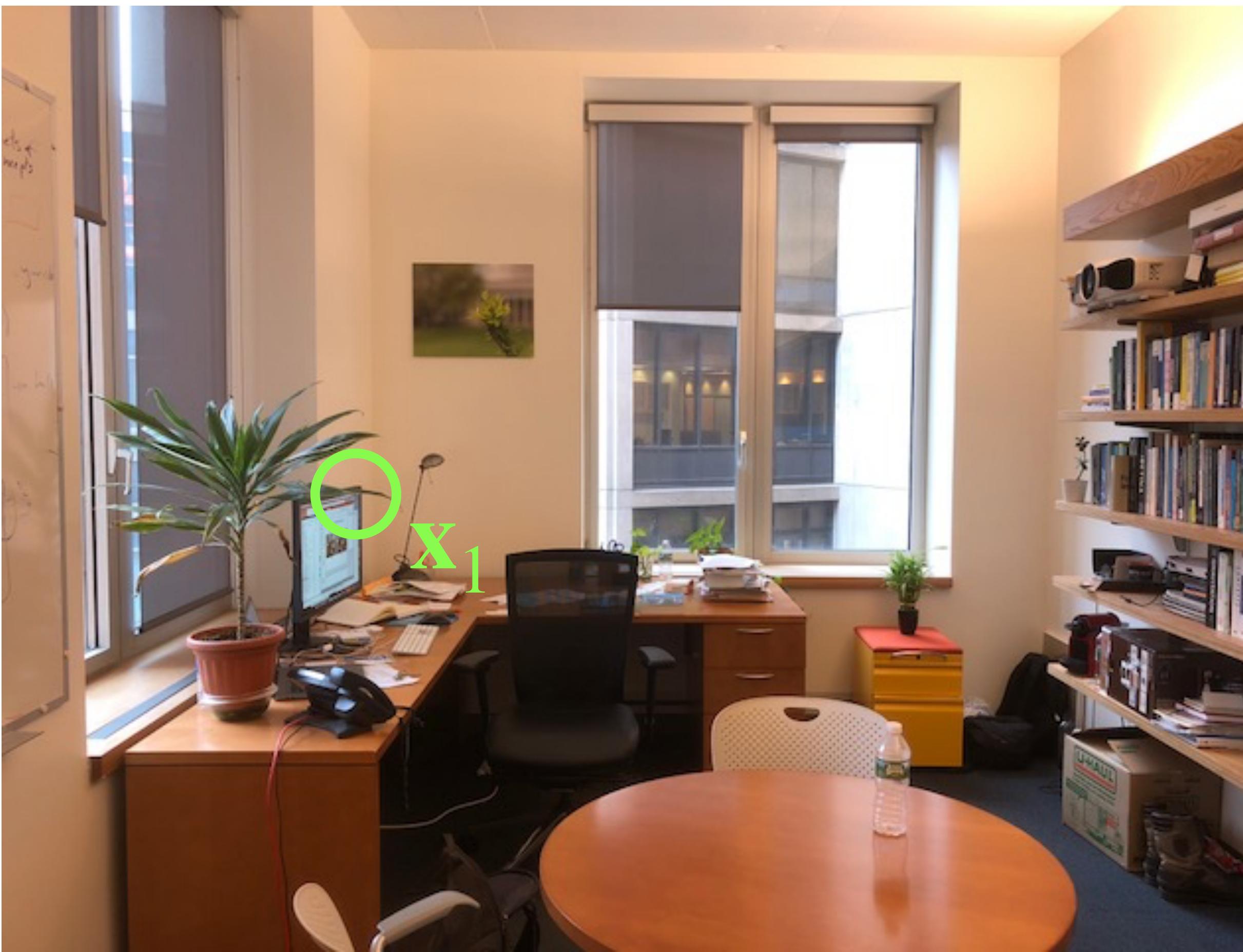


Antonio's old office!

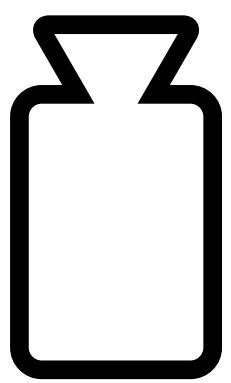


Known $P_1, P_2!$





Known $P_1, P_2!$



Bundle Adjustment

Triangulation

How to compute 3D locations of point correspondences if cameras are known.

Epipolar Lines

Which pixels in two cameras observe same 3D point?
Where to look for multi-view correspondences?

Fundamental & Essential Matrices

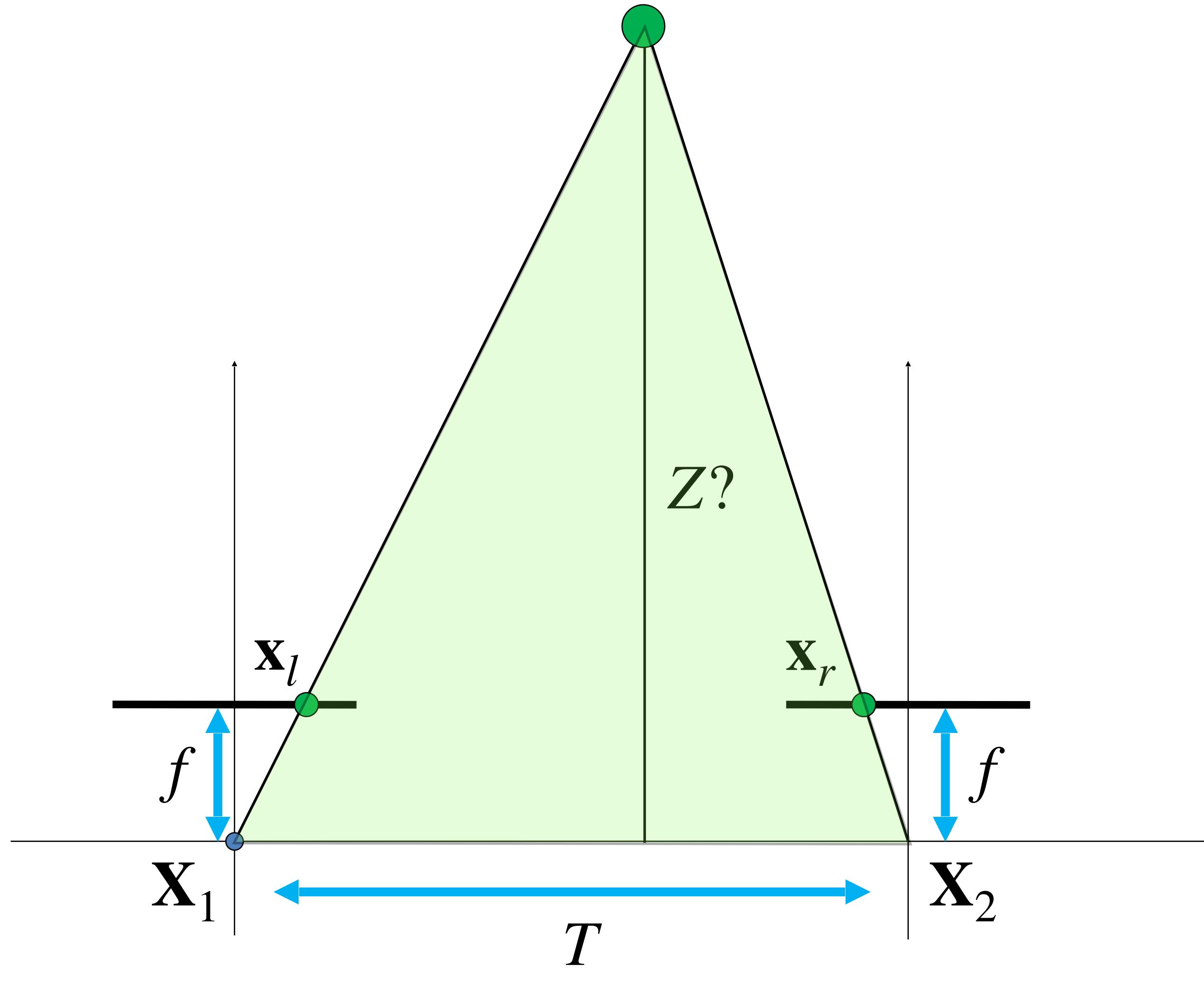
Elegant formulation of Epipolar Lines
A way of estimating camera poses, intrinsics, and extrinsic from correspondences.

No Time

-
-
-

What has changed since Deep Learning?

Last time: Simple Stereo System



Similar Triangles:

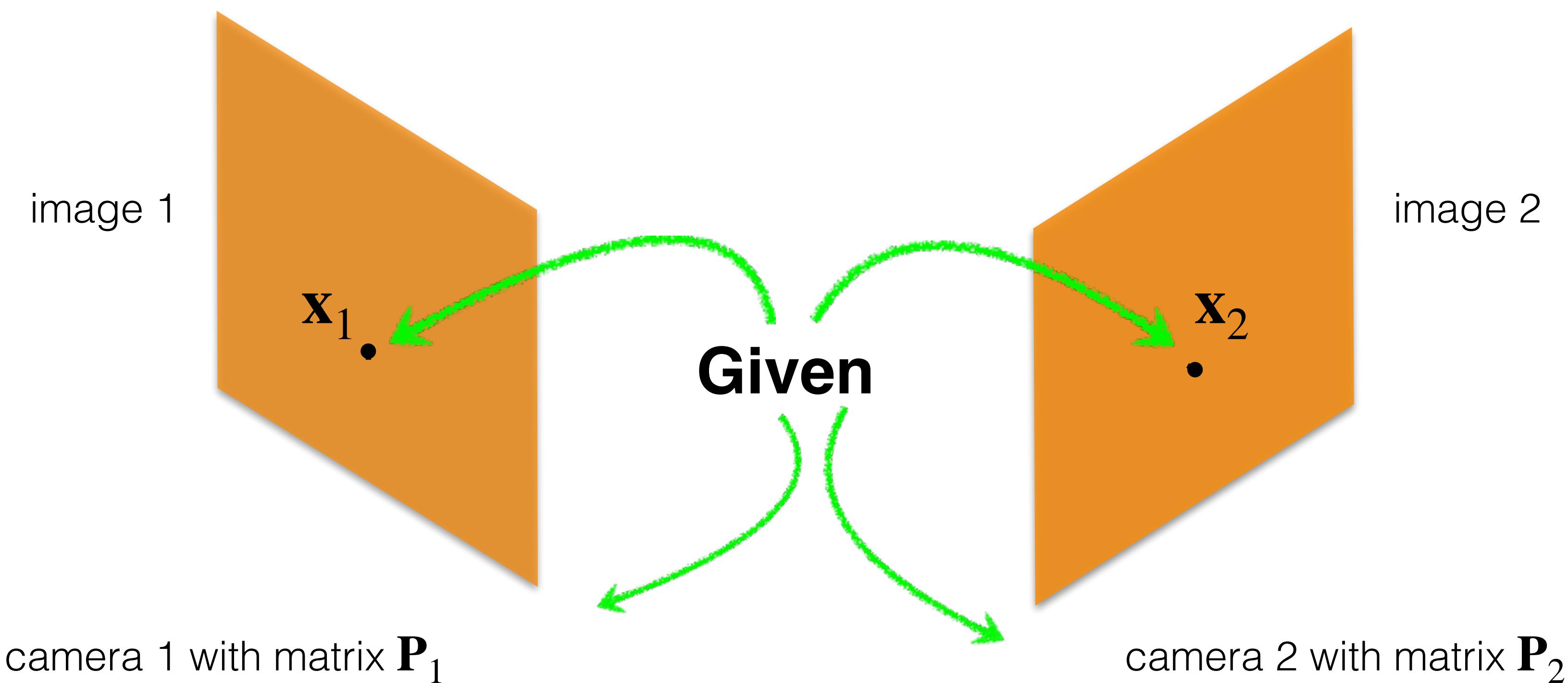
$$\frac{T + \mathbf{x}_r - \mathbf{x}_l}{Z - f} = \frac{T}{Z}$$

Solve for Z :

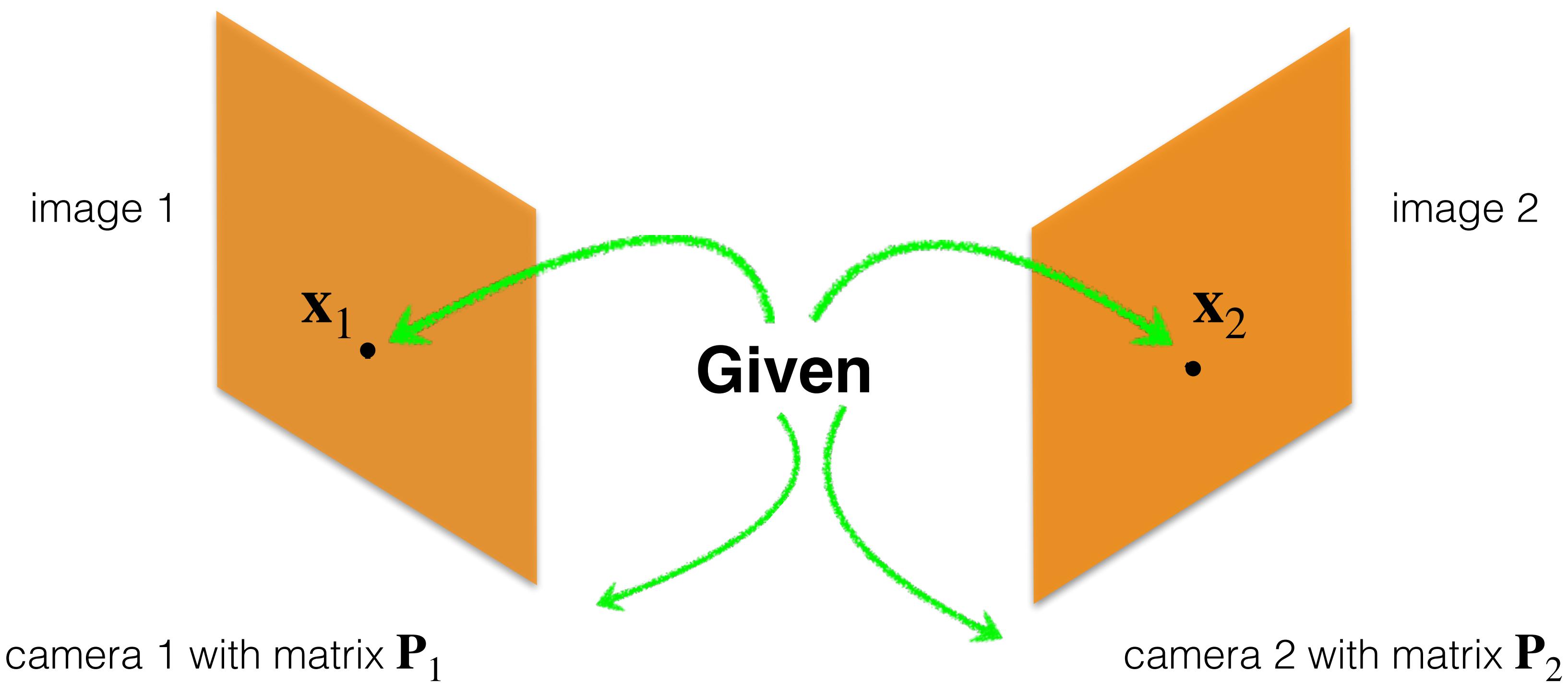
$$Z = f \frac{T}{\mathbf{x}_l - \mathbf{x}_r}$$

Disparity

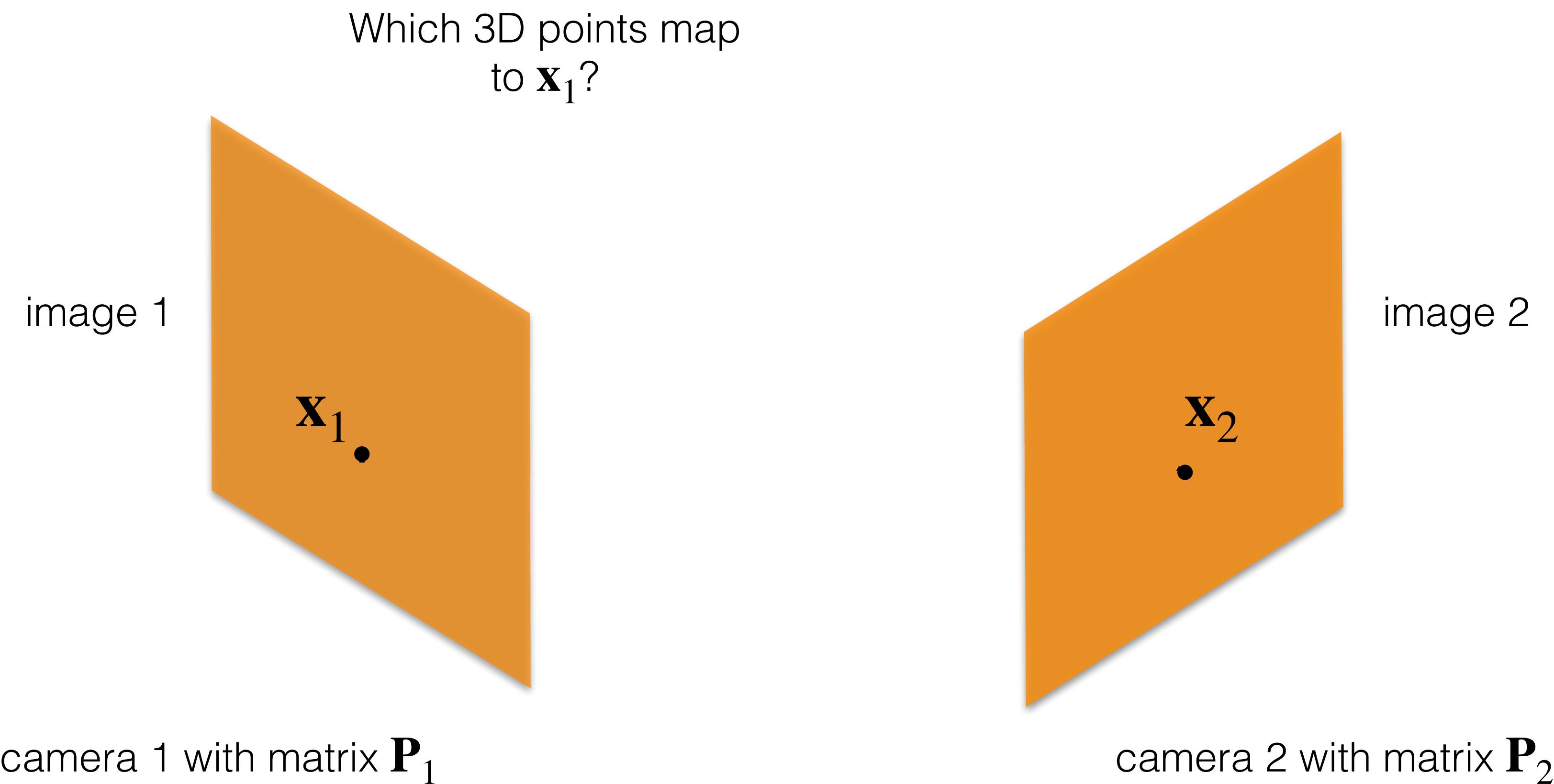
Now: General Stereo system



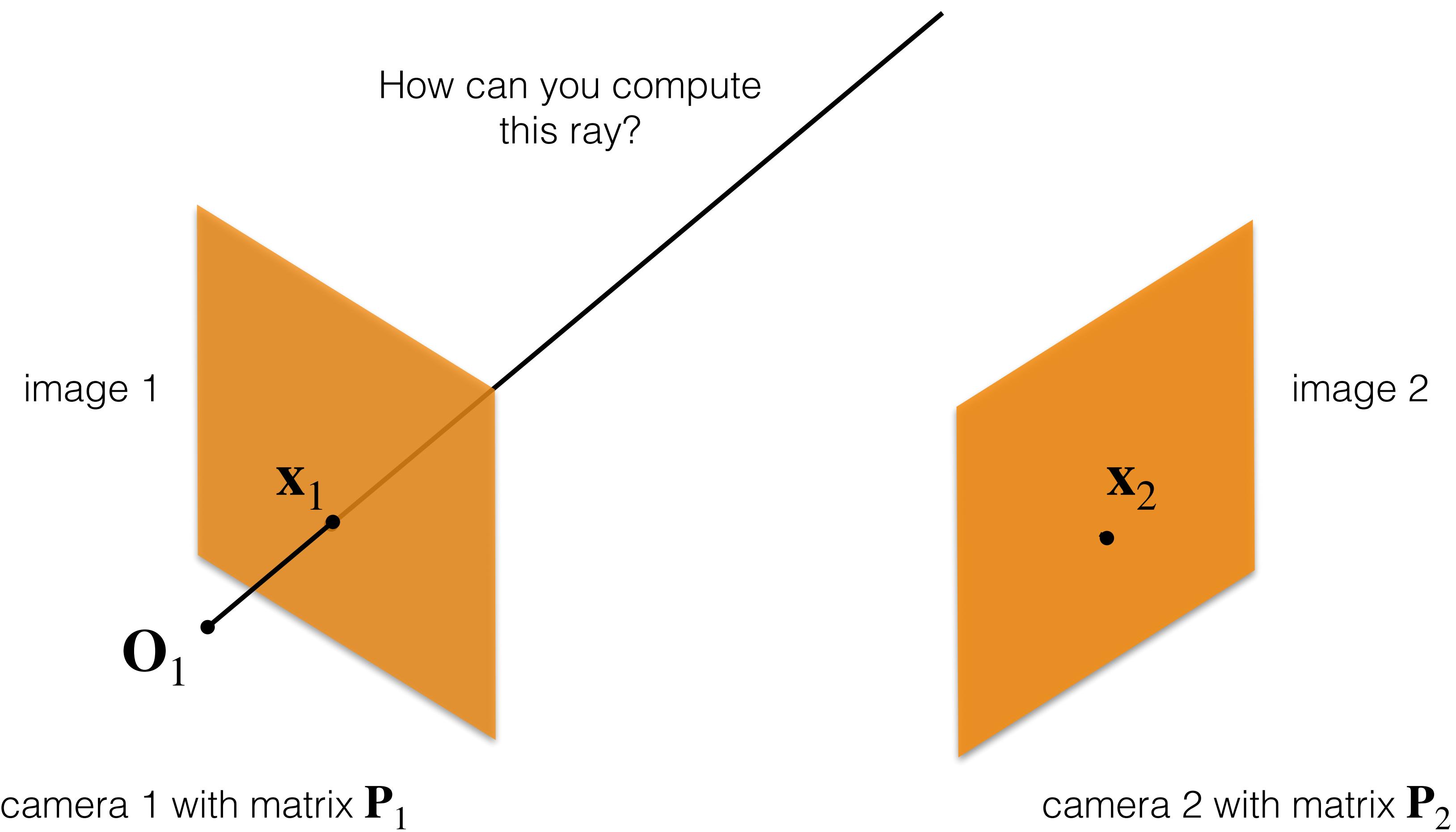
Triangulation



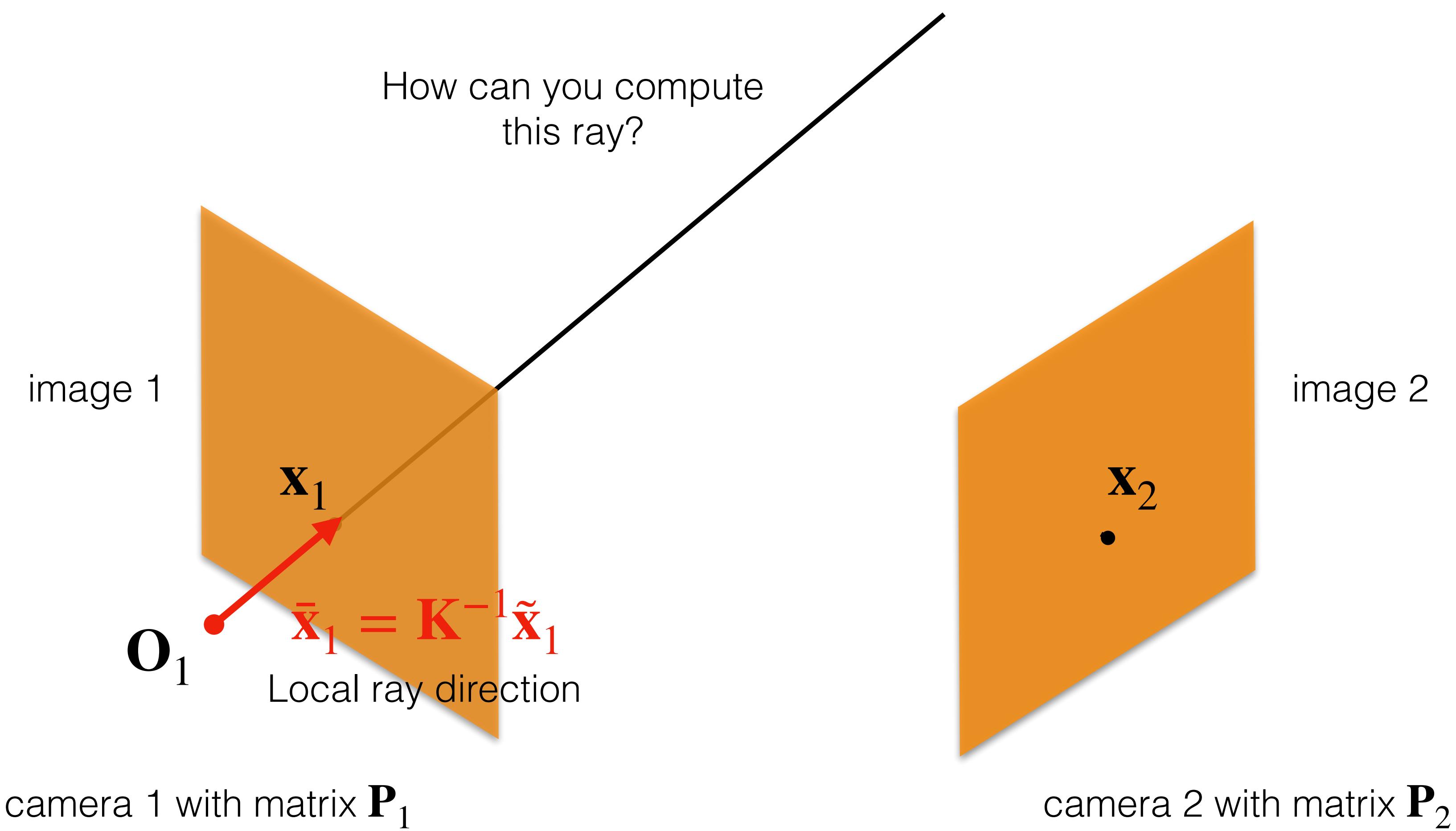
Triangulation



Triangulation



Triangulation

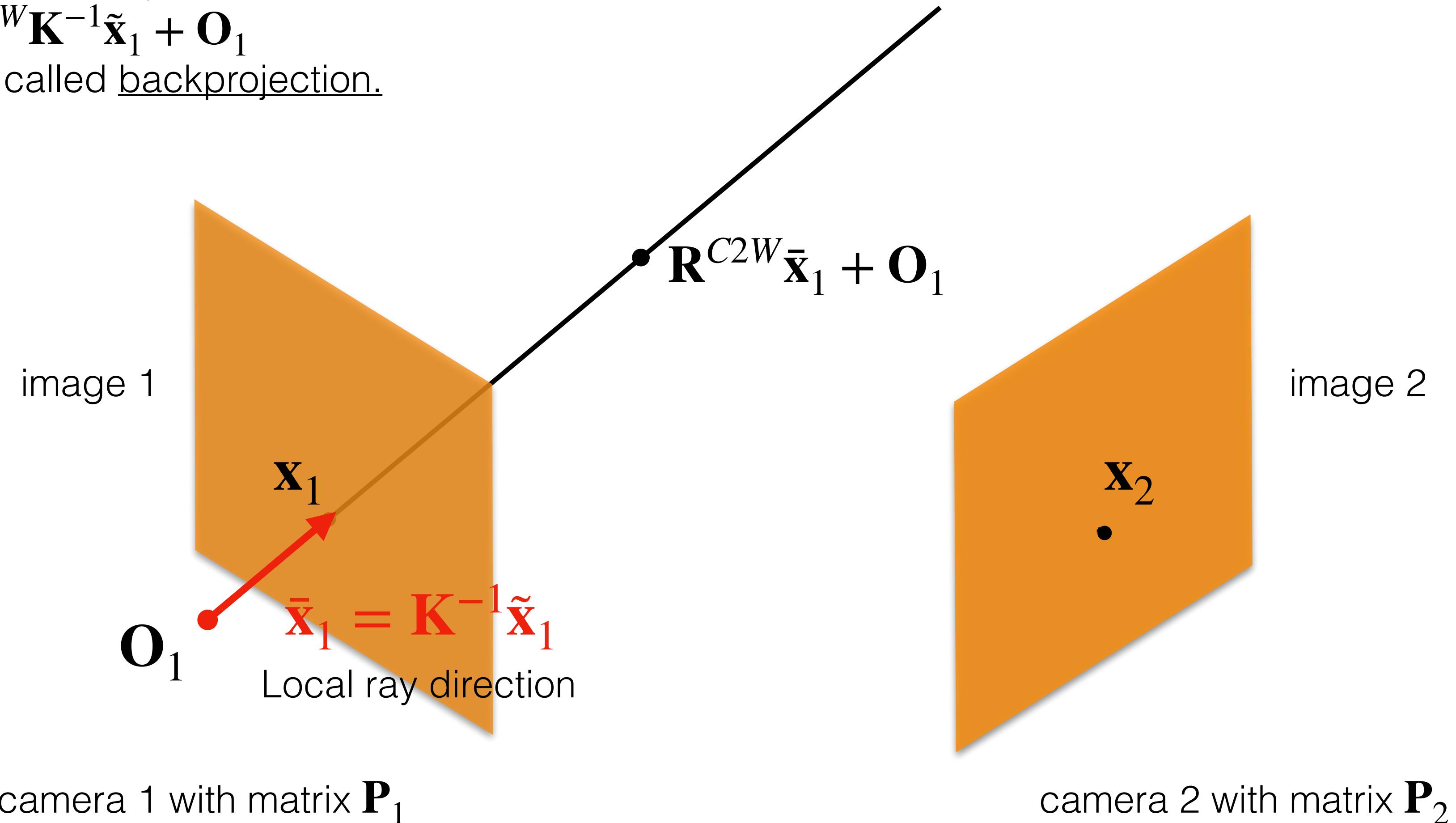


Triangulation

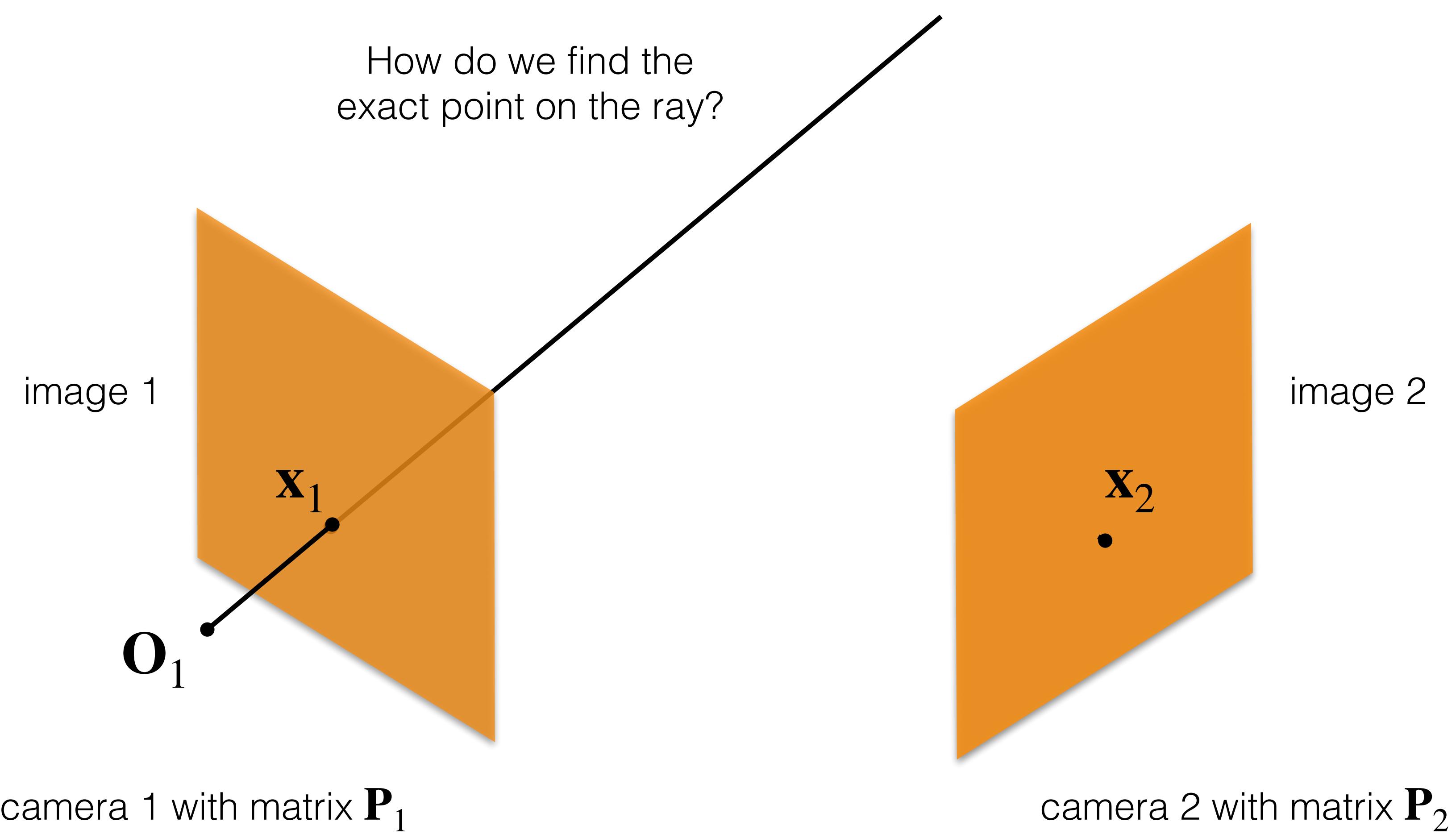
Create two points on the ray:

- 1) find the camera center; and
- 2) Compute $\mathbf{R}^{C2W}\mathbf{K}^{-1}\tilde{\mathbf{x}}_1 + \mathbf{O}_1$

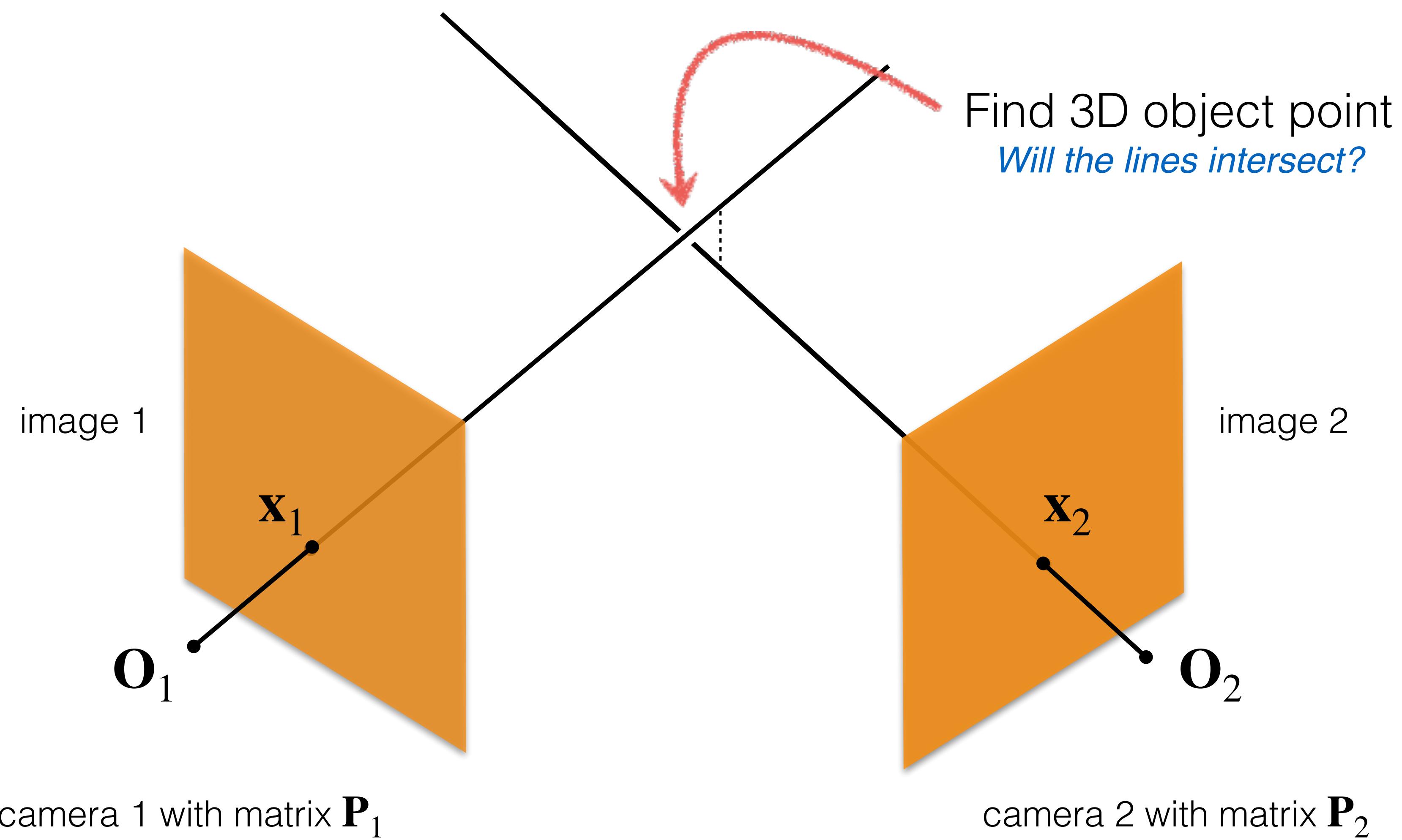
This procedure is called backprojection.



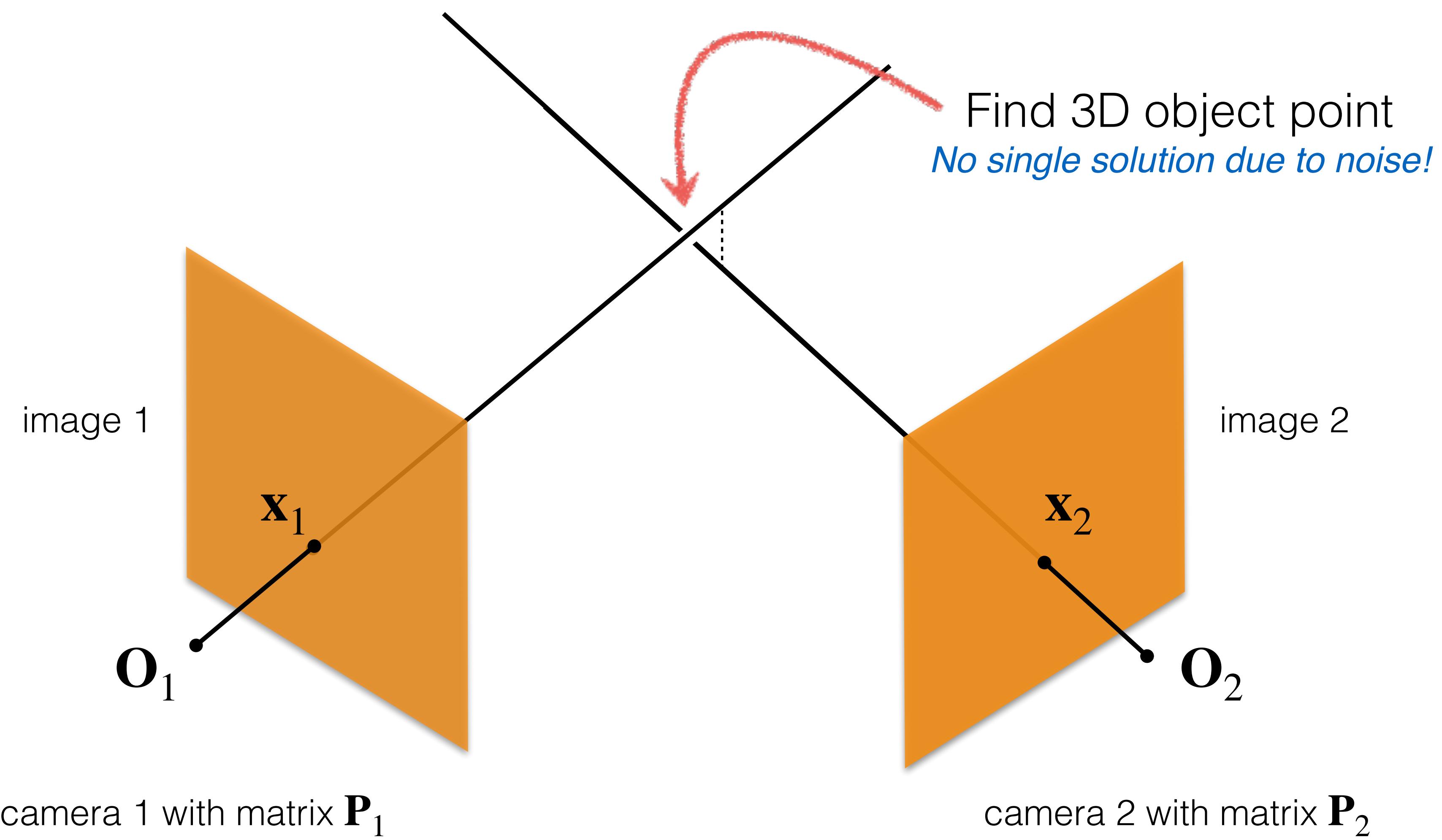
Triangulation



Triangulation



Triangulation



Triangulation

Given a set of (noisy) matched
pixel coordinates

$$\{\mathbf{x}_i\}_{i=1}^N$$

Estimate the 3D point

$$\mathbf{X}$$

Triangulation

Given a set of (noisy) matched
pixel coordinates

$$\{\mathbf{x}_i\}_{i=1}^N$$

Denote projection of \mathbf{X} into i-th camera as

$$\tilde{\pi}_i(\mathbf{X}) = \mathbf{K}_i[\mathbf{I} | 0] \mathbf{C}_i^{W2C} \tilde{\mathbf{X}}$$

Estimate the 3D point

$$\mathbf{X}$$

Triangulation

Given a set of (noisy) matched
pixel coordinates

$$\{\mathbf{x}_i\}_{i=1}^N$$

Denote projection of \mathbf{X} into i-th camera as

$$\tilde{\pi}_i(\mathbf{X}) = \mathbf{K}_i[\mathbf{I} | 0] \mathbf{C}_i^{W2C} \tilde{\mathbf{X}}$$

Estimate the 3D point

$$\mathbf{X}$$

Then we can solve a little least squares problem:

$$\mathbf{X}^* = \operatorname{argmin}_{\mathbf{X}} \sum_i^N \|\pi_i(\mathbf{X}) - \mathbf{x}_i\|_2^2$$

Triangulation

Given a set of (noisy) matched
pixel coordinates

$$\{\mathbf{x}_i\}_{i=1}^N$$

Denote projection of \mathbf{X} into i-th camera as

$$\tilde{\pi}_i(\mathbf{X}) = \mathbf{K}_i[\mathbf{I} | 0] \mathbf{C}_i^{W2C} \tilde{\mathbf{X}}$$

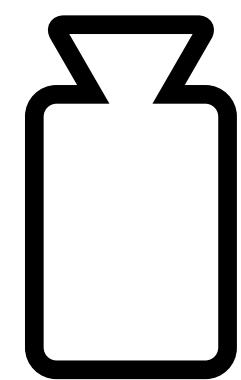
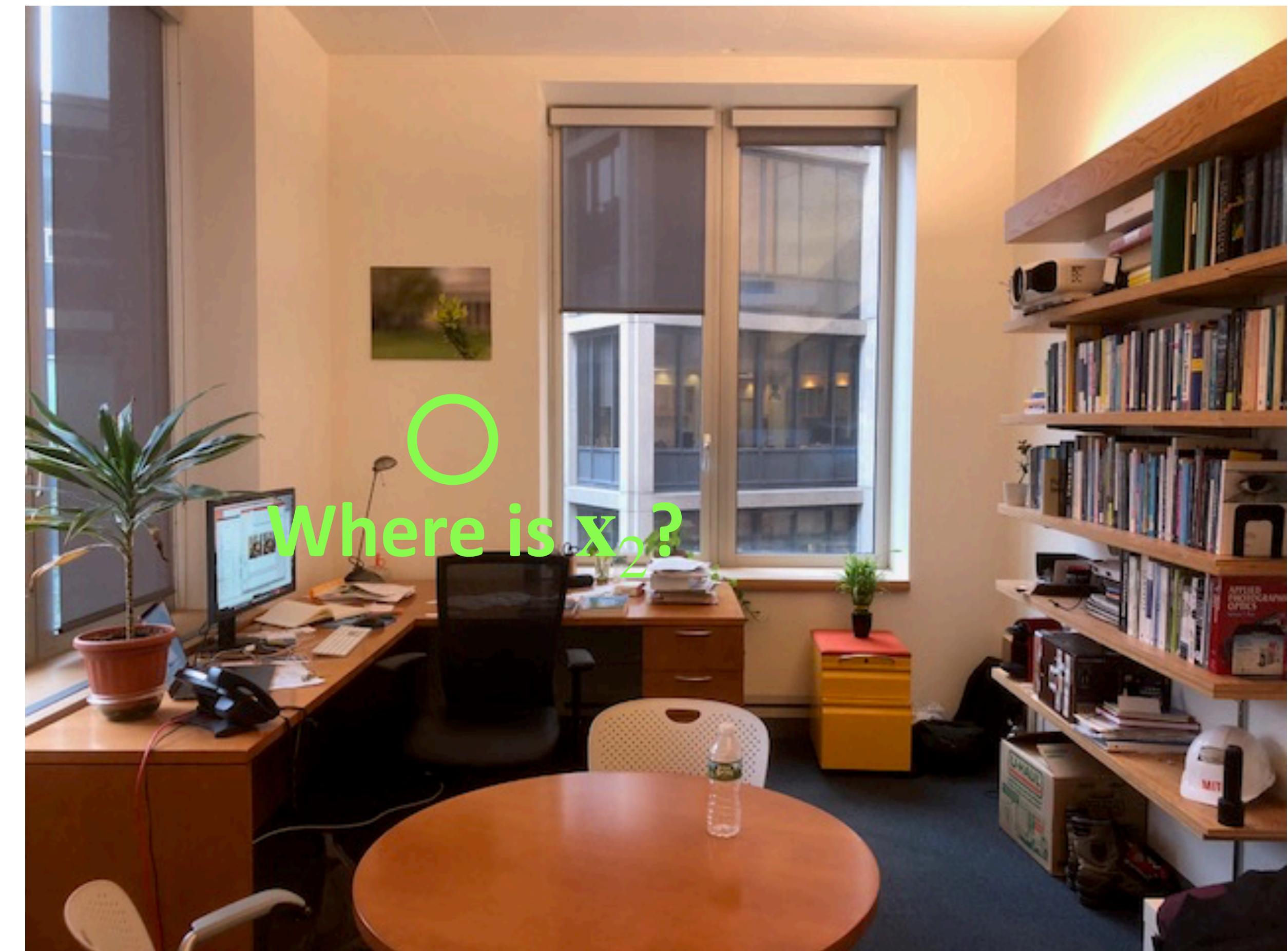
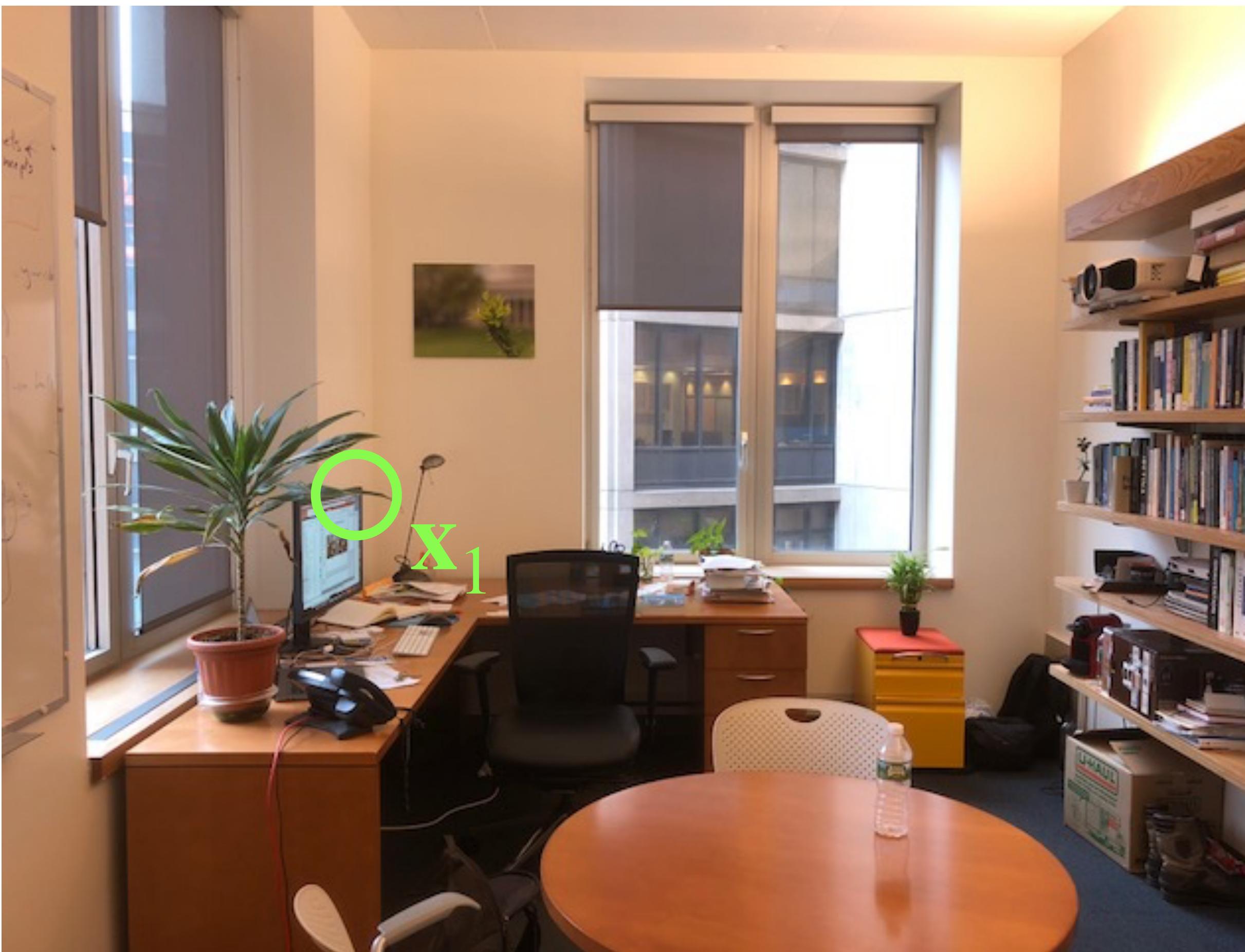
Estimate the 3D point

$$\mathbf{X}$$

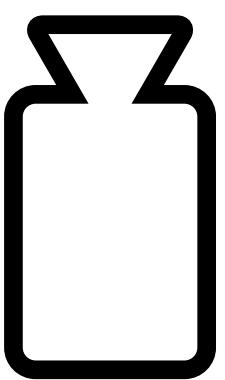
Then we can solve a little least squares problem:

$$\mathbf{X}^* = \operatorname{argmin}_{\mathbf{X}} \sum_i^N \|\pi_i(\mathbf{X}) - \mathbf{x}_i\|_2^2$$

*Can be solved via numerical optimization
(Gradient Descent, or smarter, Levenberg-Marquardt)*



Known $P_1, P_2!$



Bundle Adjustment

Triangulation

How to compute 3D locations of point correspondences if cameras are known.

Epipolar Lines

Which pixels in two cameras observe same 3D point?
Where to look for multi-view correspondences?

Fundamental & Essential Matrices

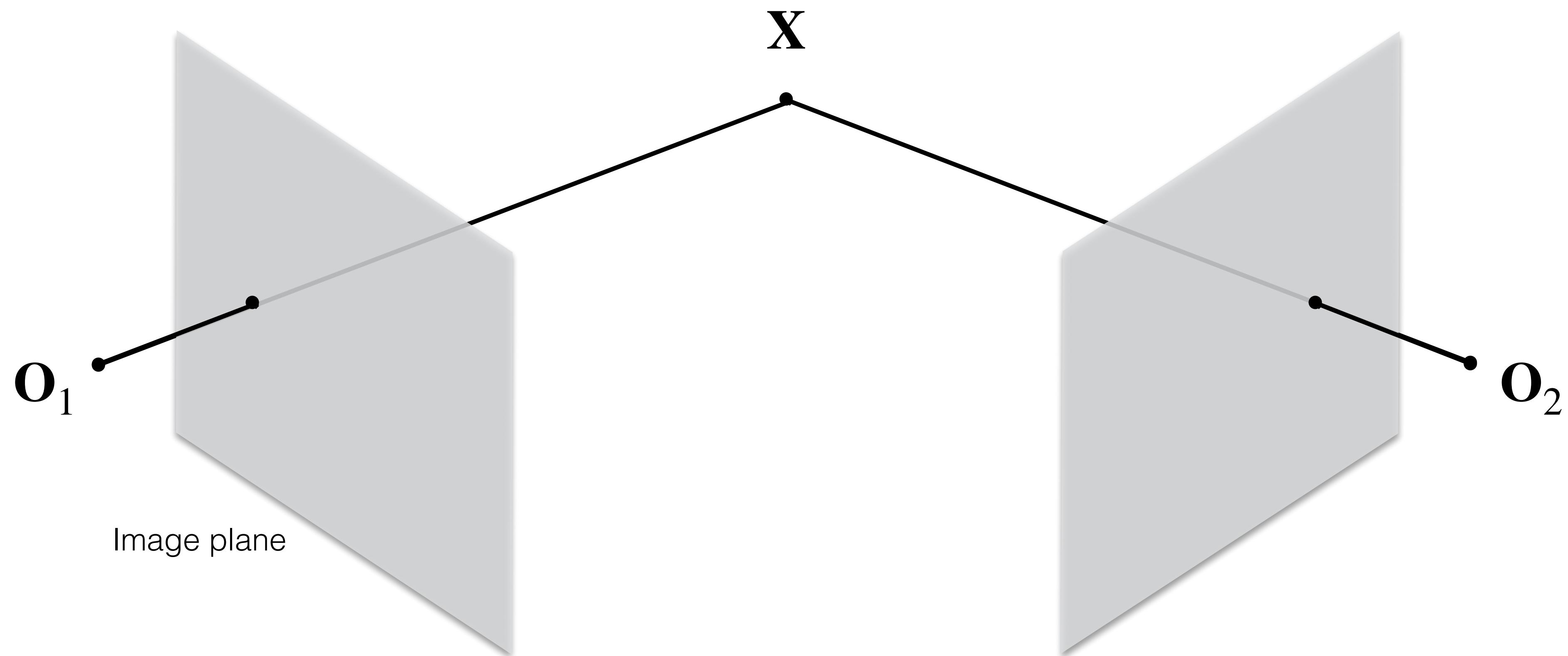
Elegant formulation of Epipolar Lines
A way of estimating camera poses, intrinsics, and extrinsic from correspondences.

No Time

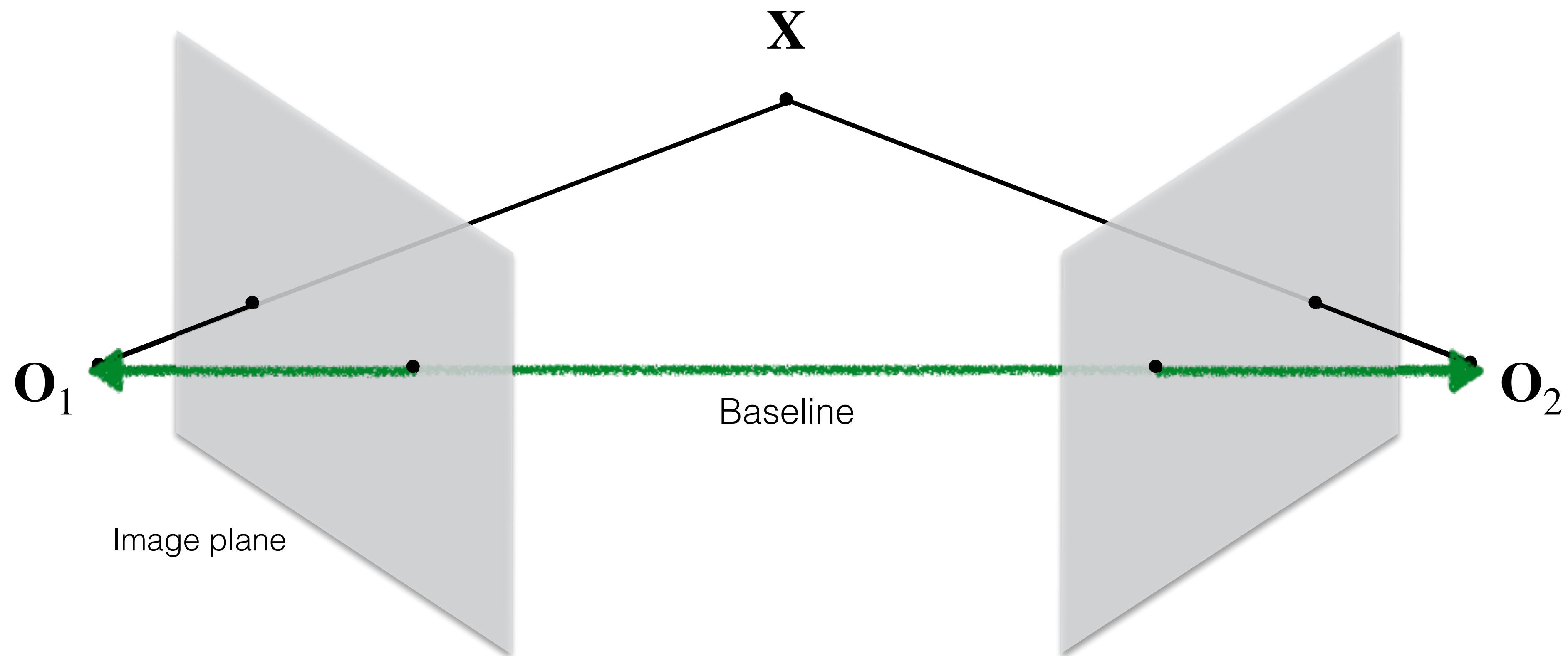
-
-
-

What has changed since Deep Learning?

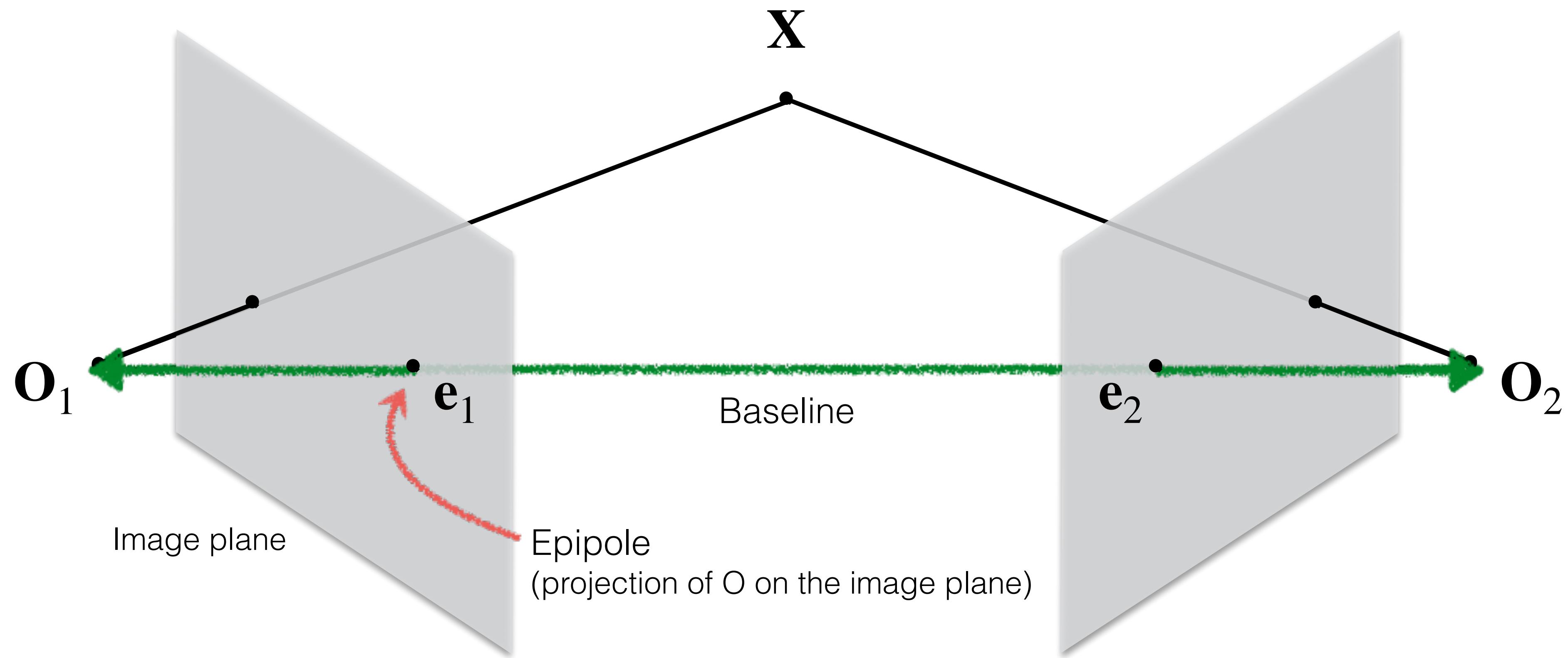
Epipolar geometry



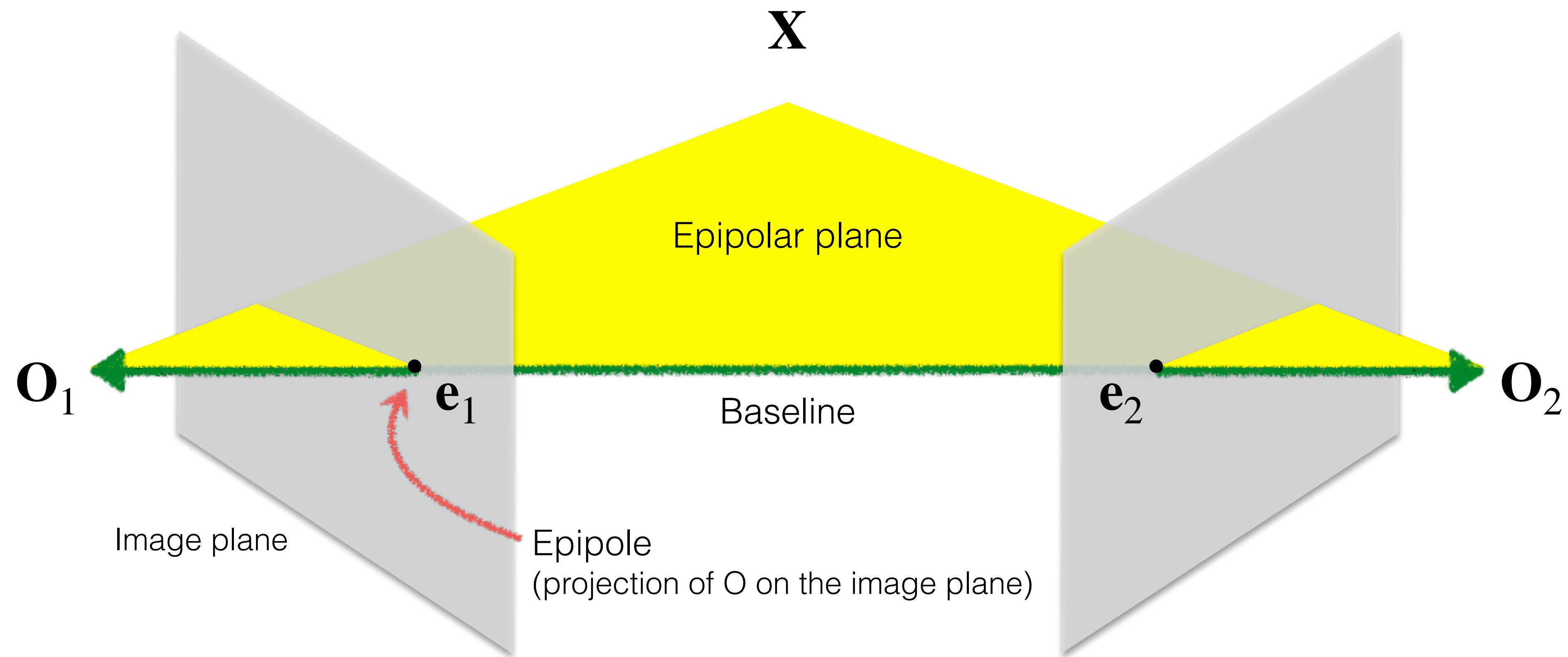
Epipolar geometry



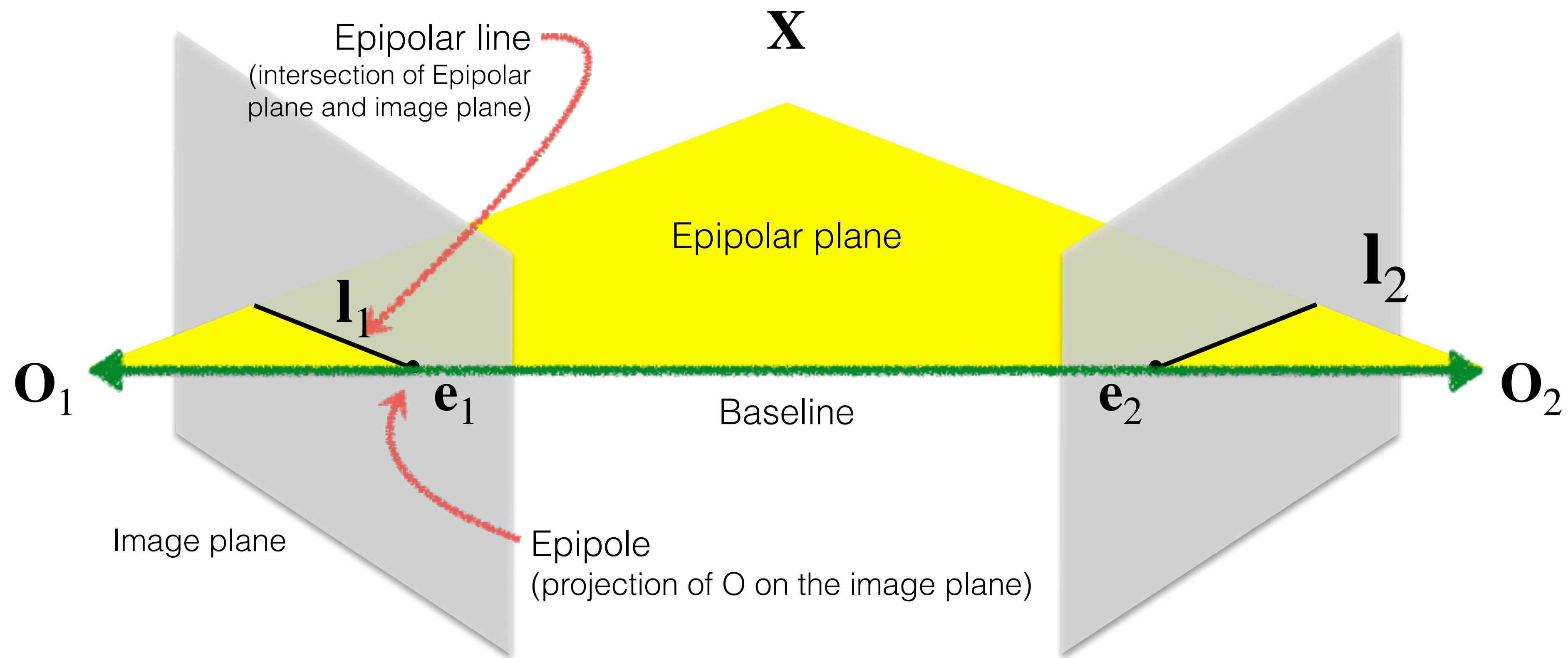
Epipolar geometry



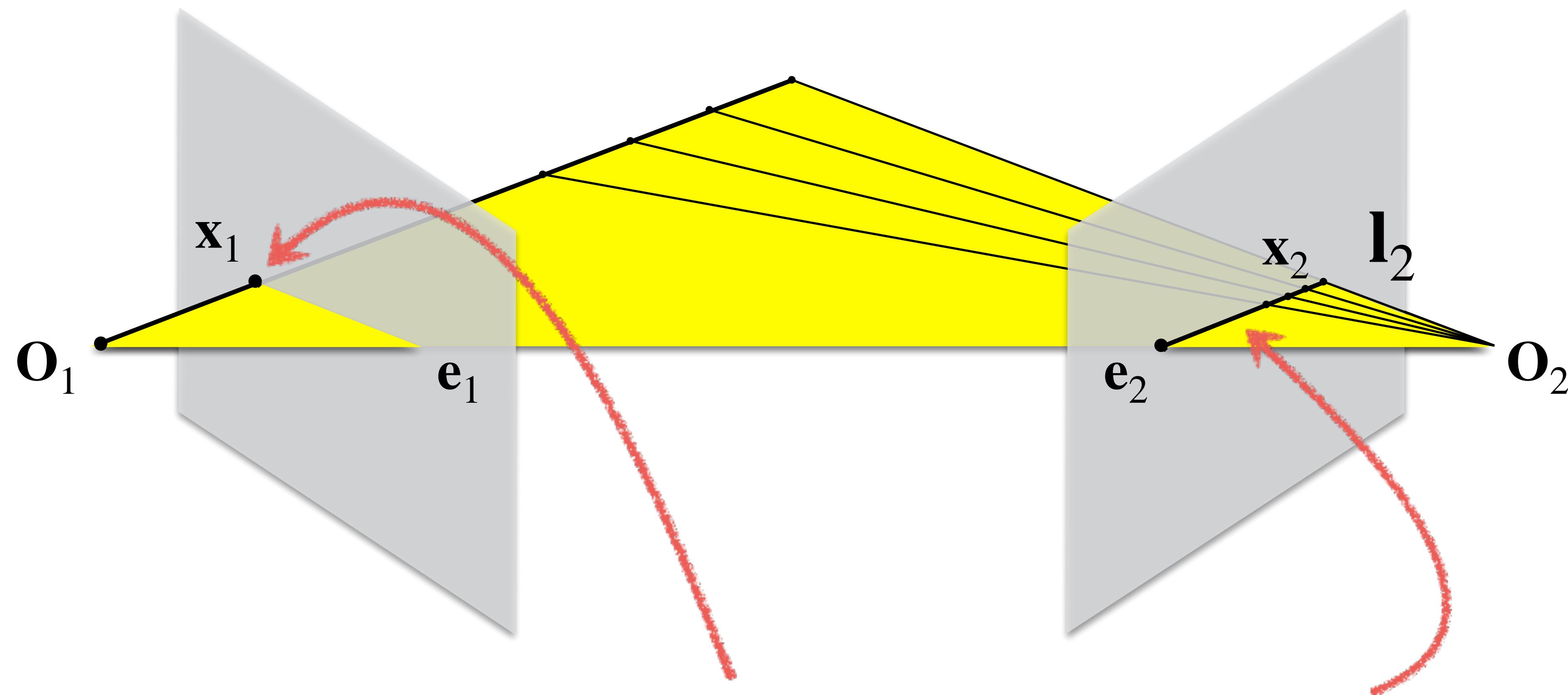
Epipolar geometry



Epipolar geometry

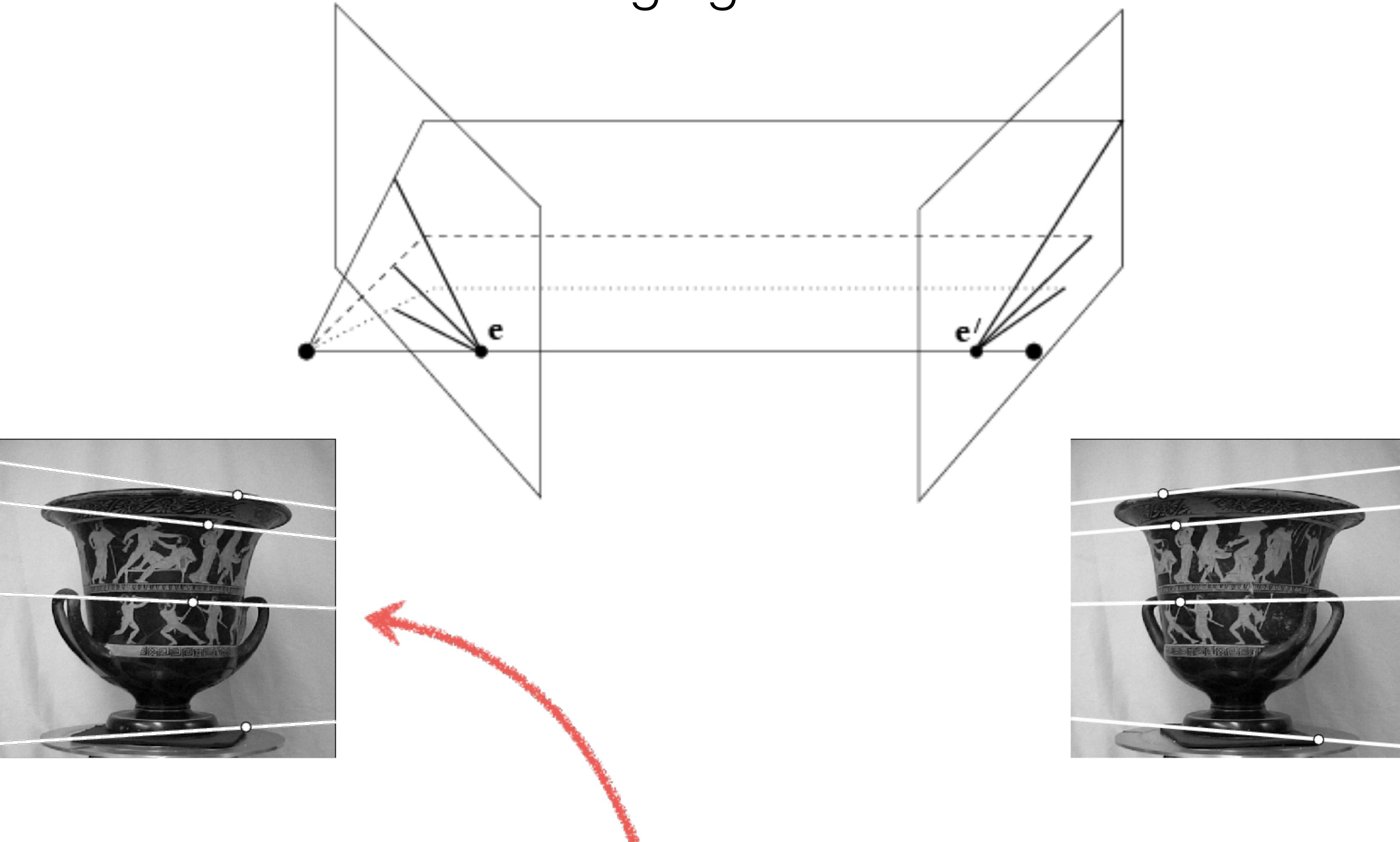


Epipolar constraint



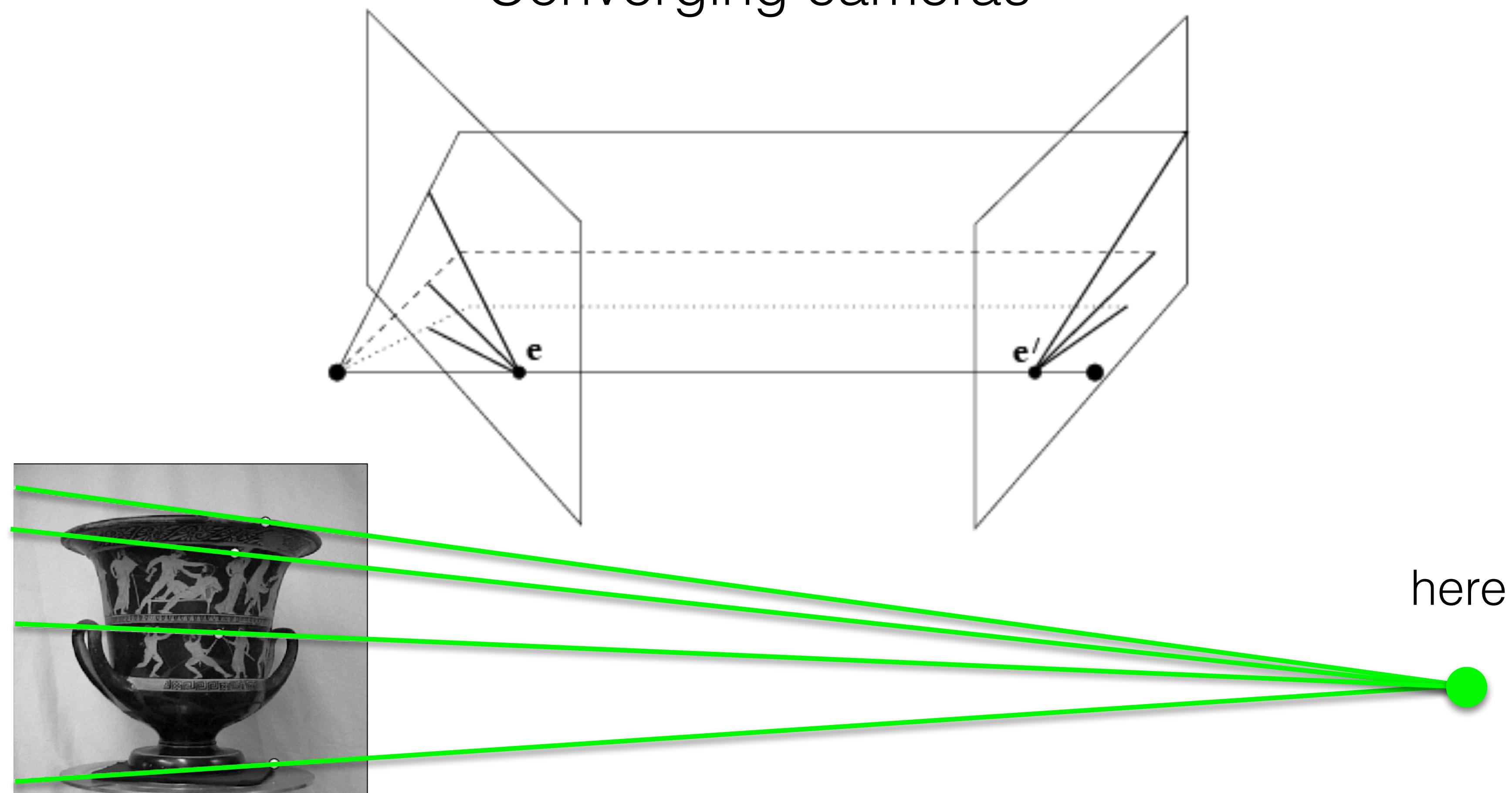
Potential matches for \mathbf{x}_1 lie on the epipolar line \mathbf{l}_2

Converging cameras



Where is the epipole in this image?

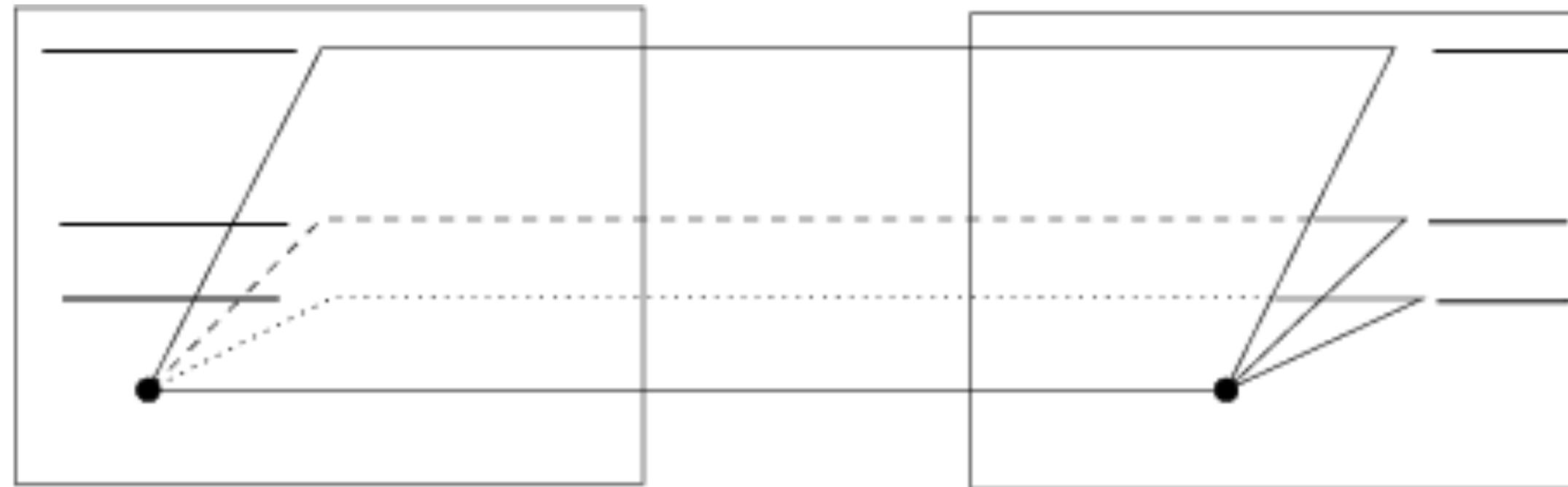
Converging cameras



Where is the epipole in this image?

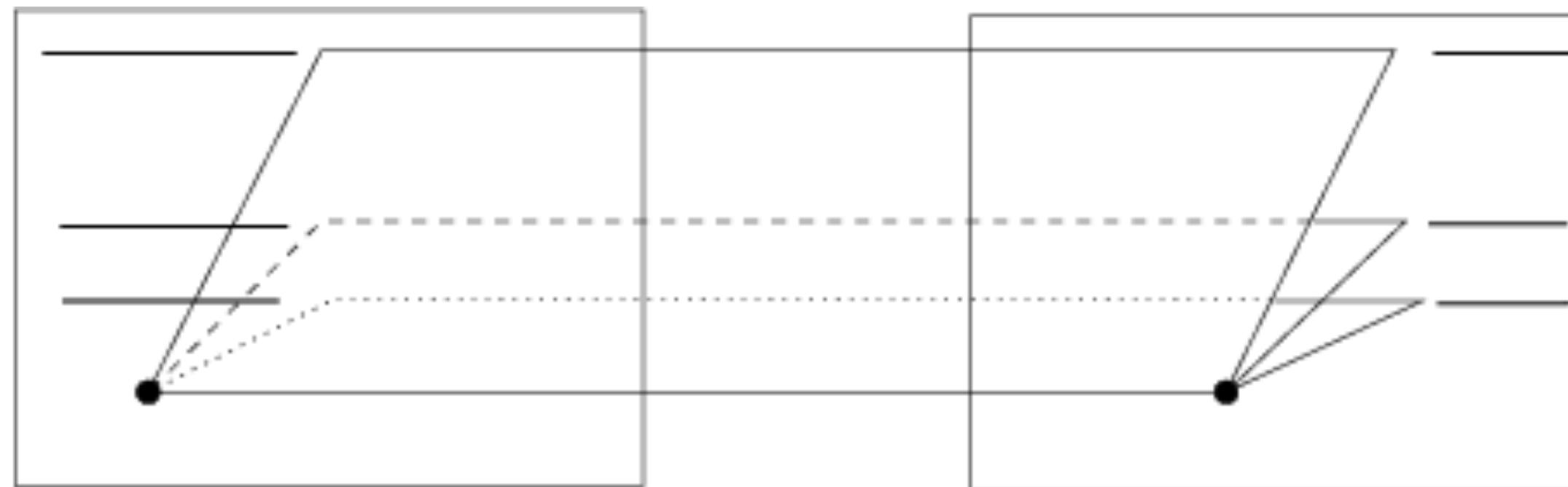
It's not always in the image

Parallel cameras



Where is the epipole?

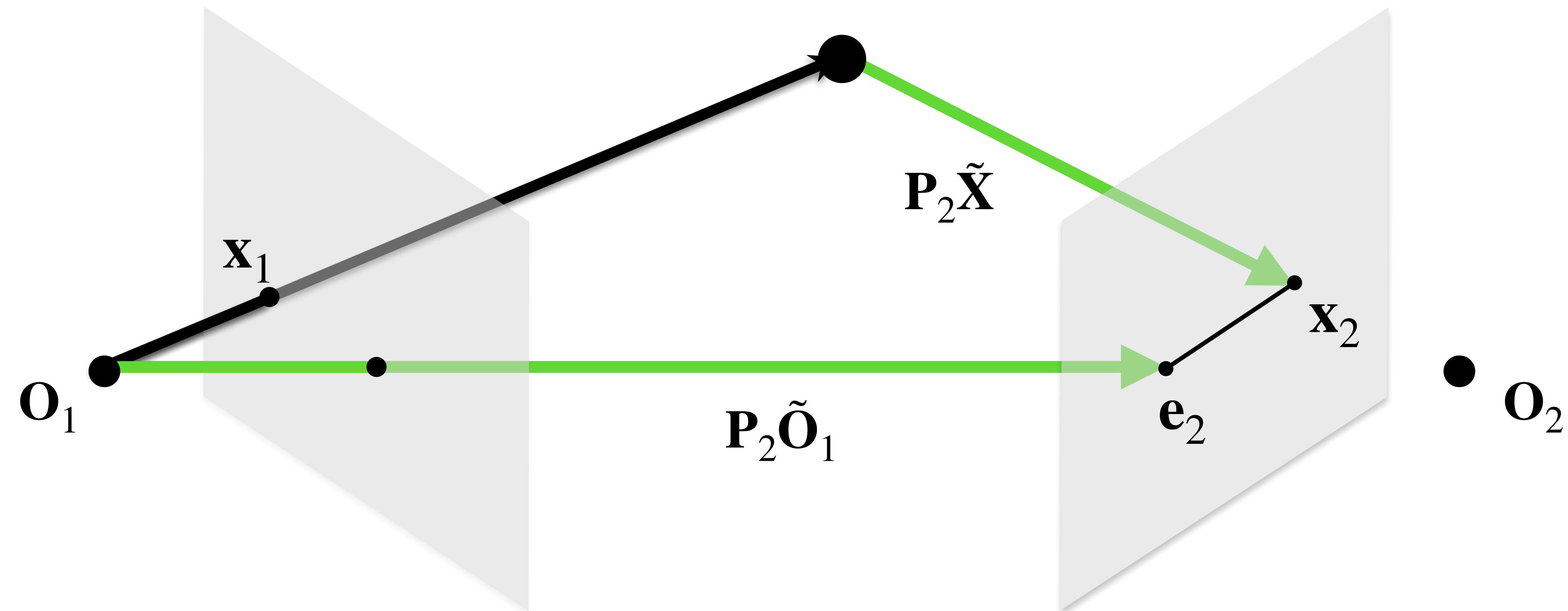
Parallel cameras

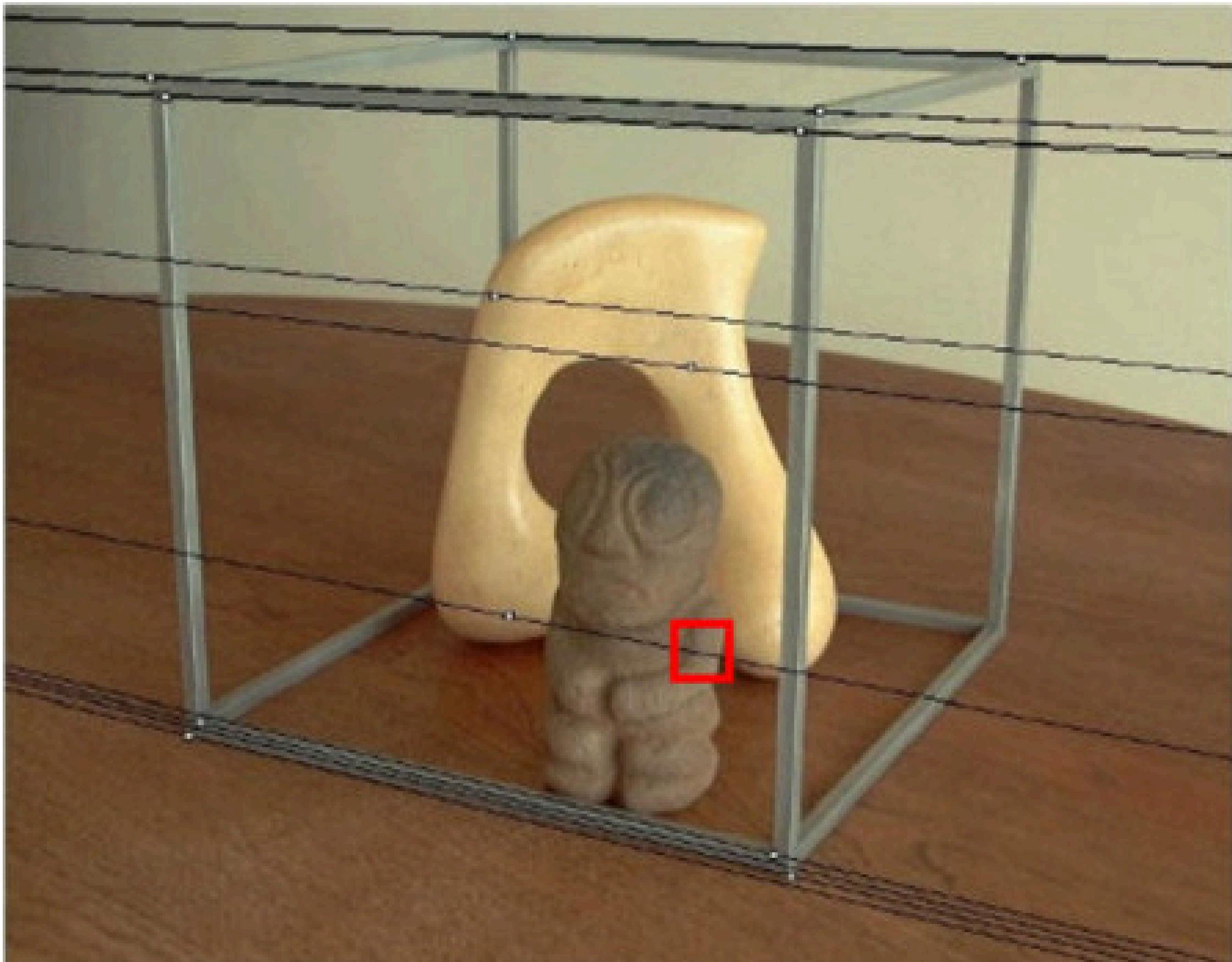


epipole at infinity

Epipolar Lines: The Hacky Way

$$\mathbf{X} = \mathbf{R}_1^{C2W} \mathbf{K}_1^{-1} \tilde{\mathbf{x}}_1 + \mathbf{O}_1$$





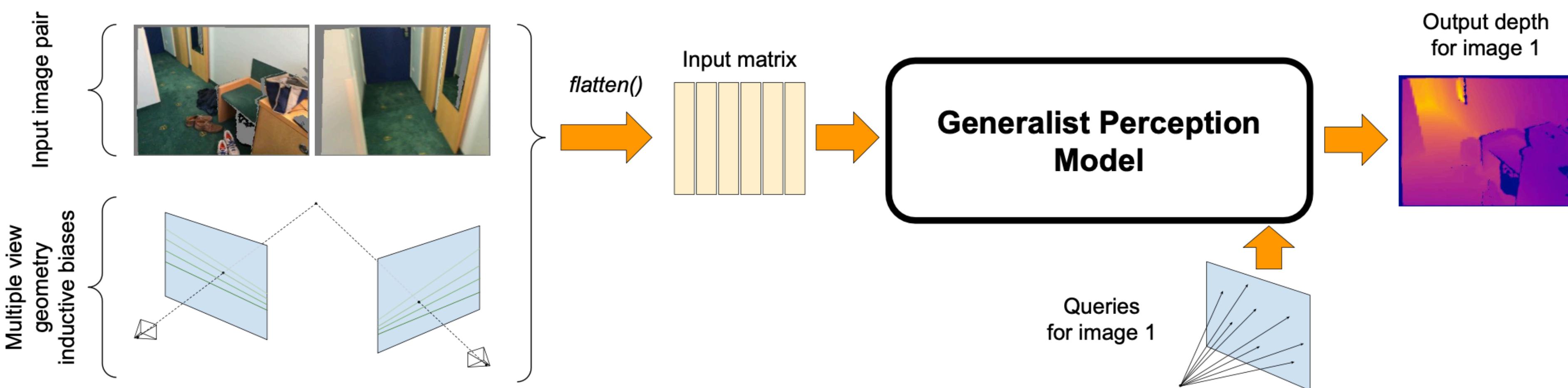
Generalizable Patch-Based Neural Rendering

Mohammed Suhail¹, Carlos Esteves⁴, Leonid Sigal^{1,2,3}, and Ameesh Makadia⁴

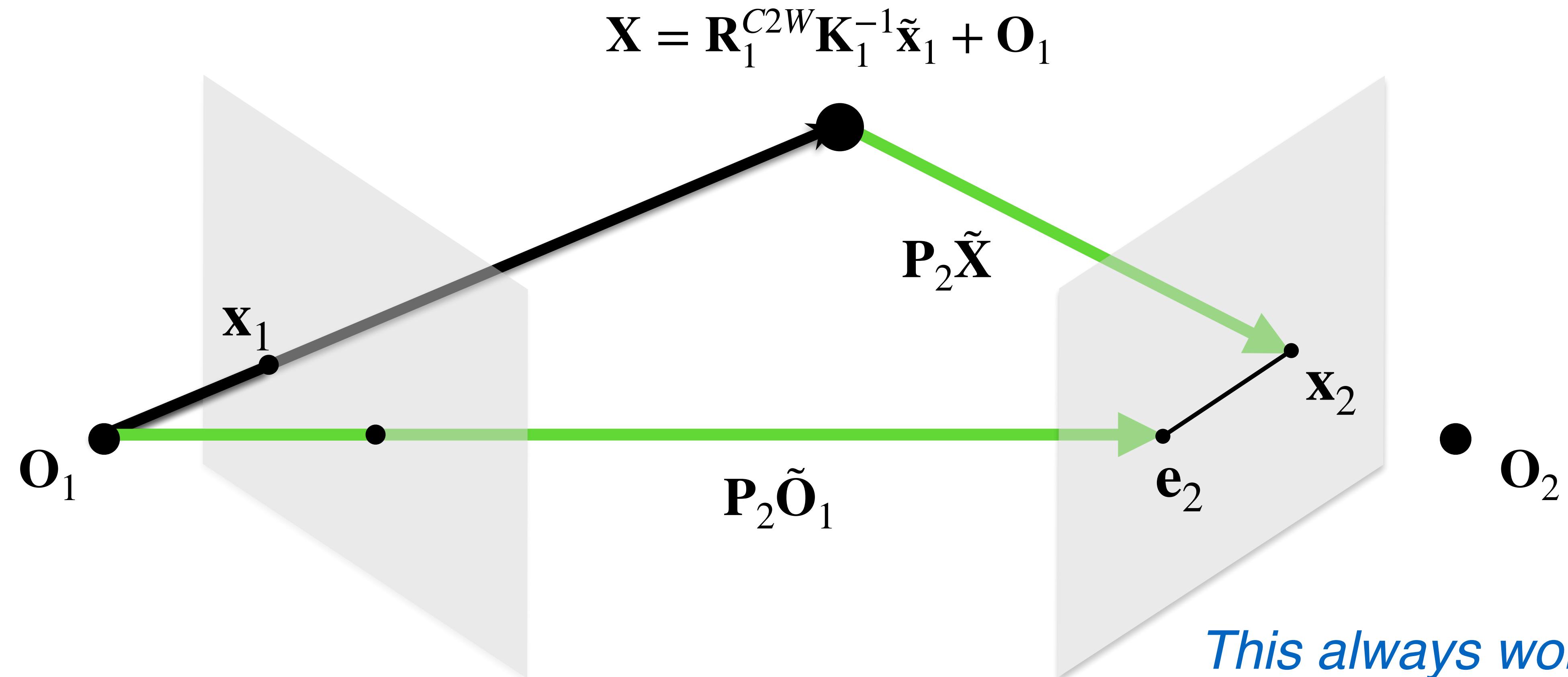
Input-level Inductive Biases for 3D Reconstruction

Wang Yifan^{1*} Carl Doersch² Relja Arandjelović² João Carreira² Andrew Zisserman^{2,3}

¹ETH Zurich ²DeepMind ³VGG, Department of Engineering Science, University of Oxford



Epipolar Lines: The Hacky Way



*This always works ;)
But: Not clean, many steps.
Is there a better way?*

Bundle Adjustment

Triangulation

How to compute 3D locations of point correspondences if cameras are known.

Epipolar Lines

Which pixels in two cameras observe same 3D point?

Where to look for multi-view correspondences?

Fundamental & Essential Matrices

Elegant formulation of Epipolar Lines

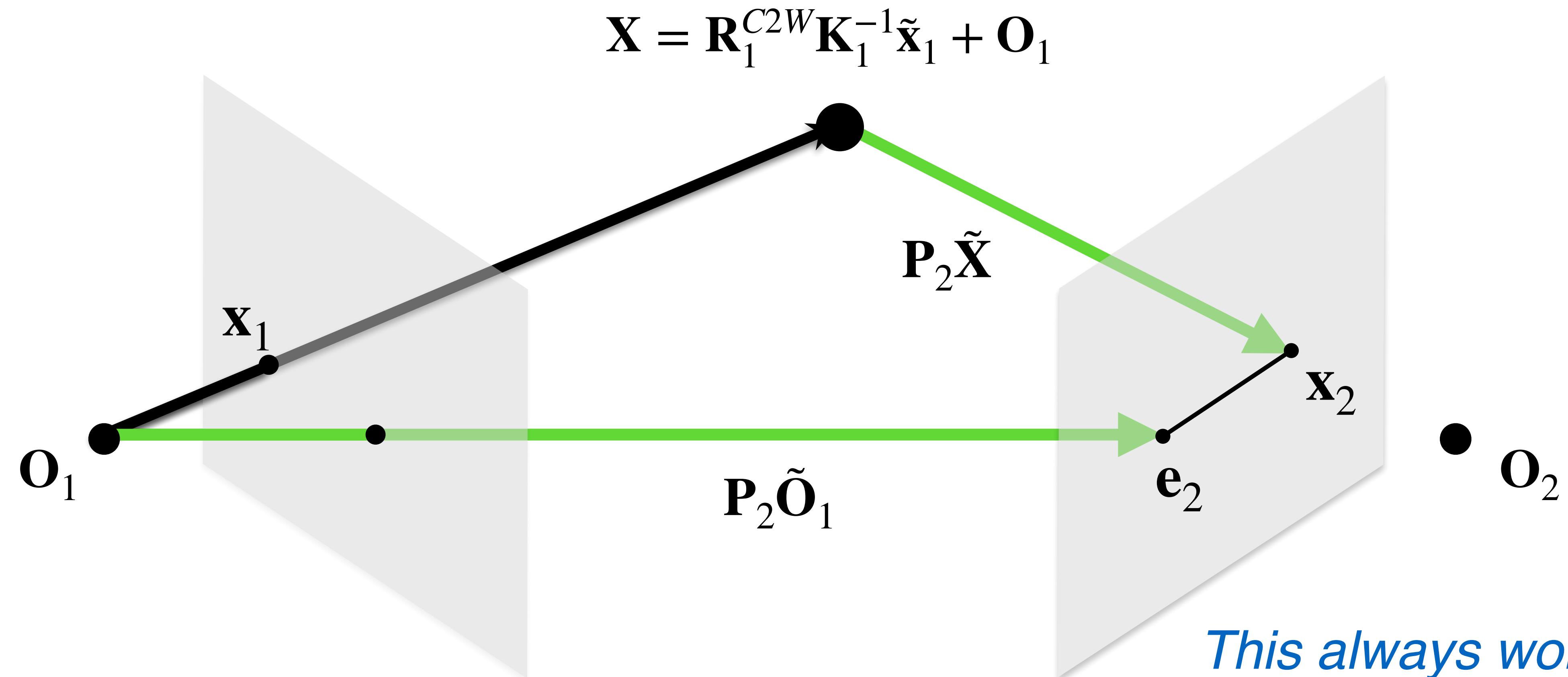
A way of *estimating camera poses, intrinsics, and extrinsic from correspondences.*

No Time

-
-
-

What has changed since Deep Learning?

Epipolar Lines: The Hacky Way



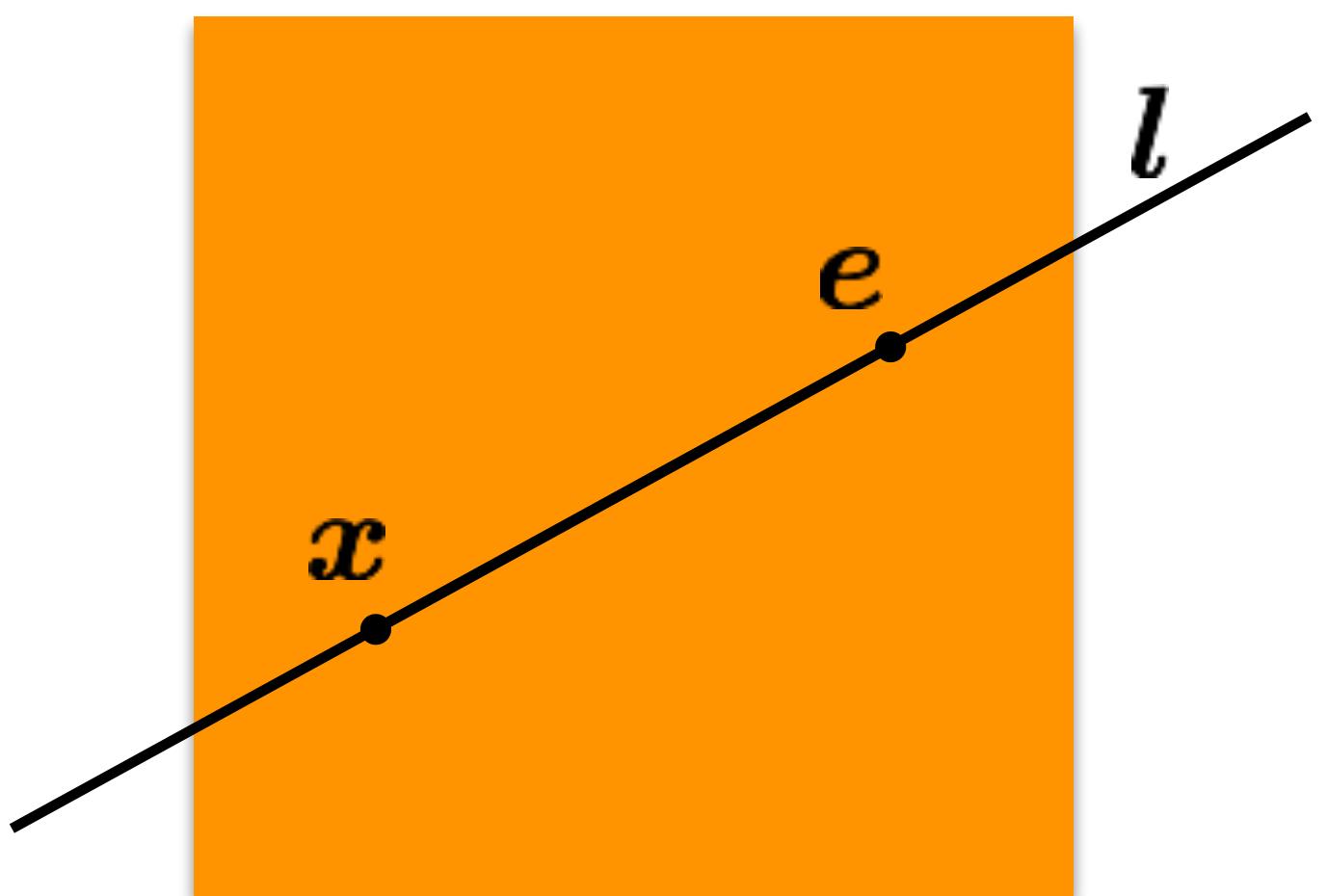
*This always works ;)
But: Not clean, many steps.
Is there a better way?*

Lines in Homogeneous Coordinates

$$ax + by + c = 0$$

in vector form

$$\mathbf{l} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$



If the point \mathbf{x} is on the epipolar line \mathbf{l} then

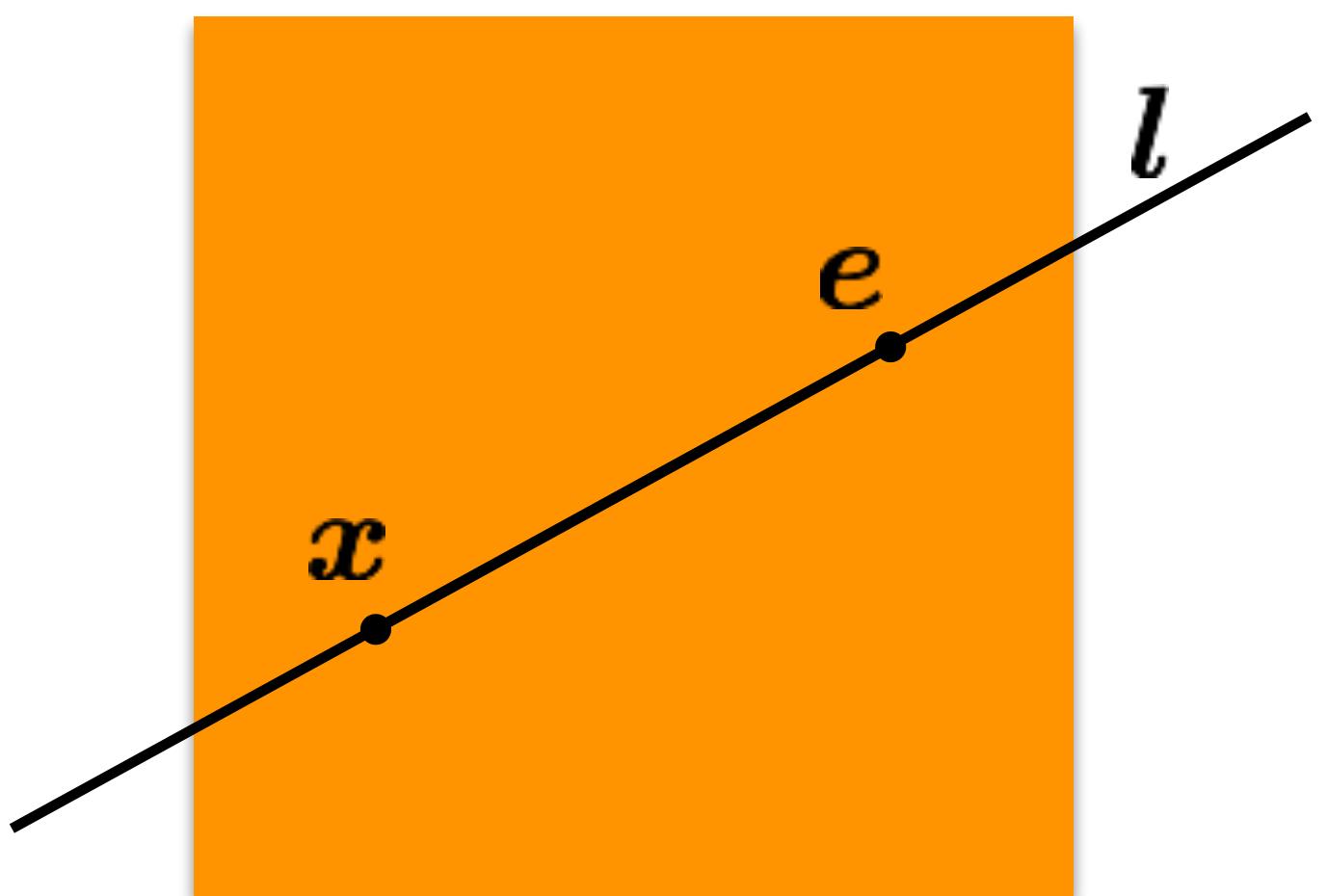
$$\tilde{\mathbf{x}}^T \mathbf{l} = ?$$

Lines in Homogeneous Coordinates

$$ax + by + c = 0$$

in vector form

$$\mathbf{l} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

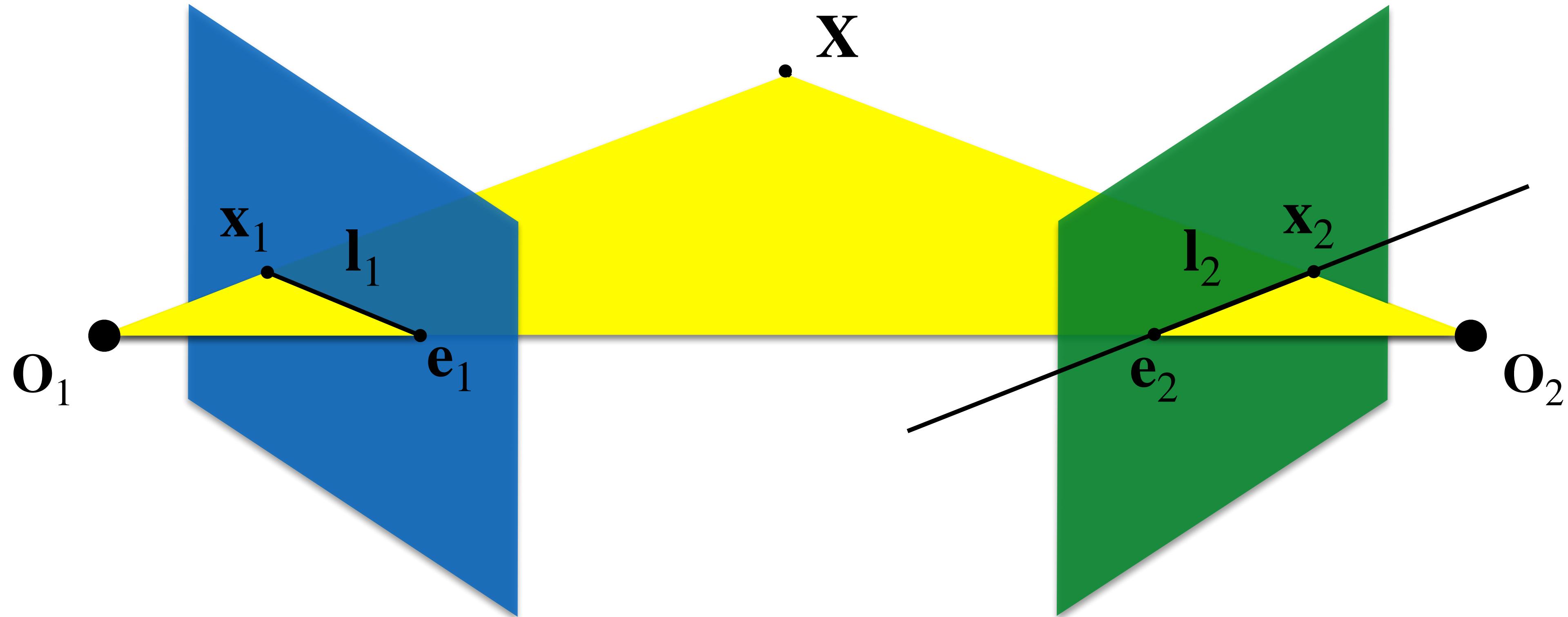


If the point \mathbf{x} is on the epipolar line \mathbf{l} then

$$\tilde{\mathbf{x}}^T \mathbf{l} = 0$$

Introducing: The Fundamental Matrix \mathbf{F}

$$\mathbf{F}\tilde{\mathbf{x}}_1 = \mathbf{l}_2$$



$$\mathbf{F}\tilde{\mathbf{x}}_1 = \mathbf{l}_2$$

The Fundamental Matrix is a 3×3 matrix
that encodes **epipolar geometry**

Given a point in one image,
multiplying by the **fundamental matrix** will tell us
the **epipolar line** in the second image.

*We'll first derive an analytical formula for \mathbf{F} , and then discuss
a numerical algorithm to estimate it from point correspondences.*

Definition of
Fundamental Matrix

$$\mathbf{F}\tilde{\mathbf{x}}_1 = \mathbf{l}_2$$

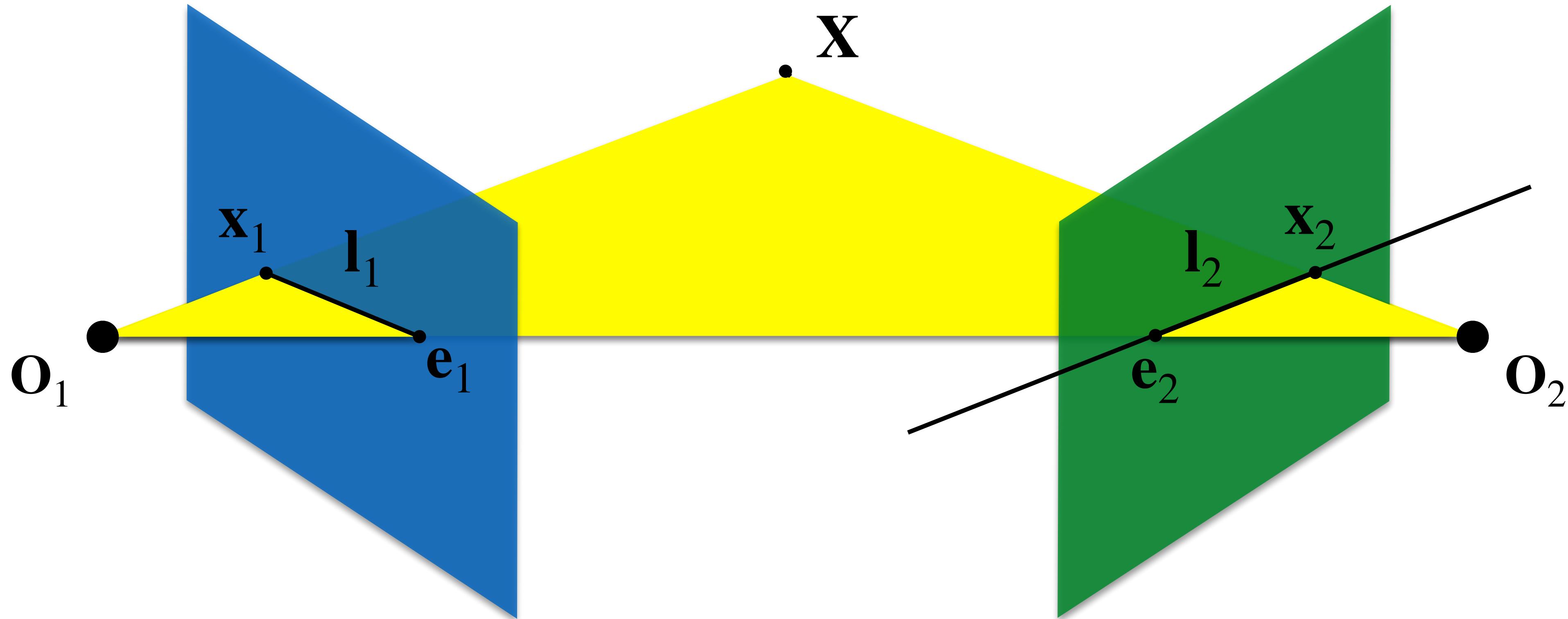
Point $\tilde{\mathbf{x}}_2$ lies on epipolar line \mathbf{l}_2

$$\tilde{\mathbf{x}}_2^T \mathbf{l}_2 = 0$$

$$\tilde{\mathbf{x}}_2^T \mathbf{l}_2 = 0$$

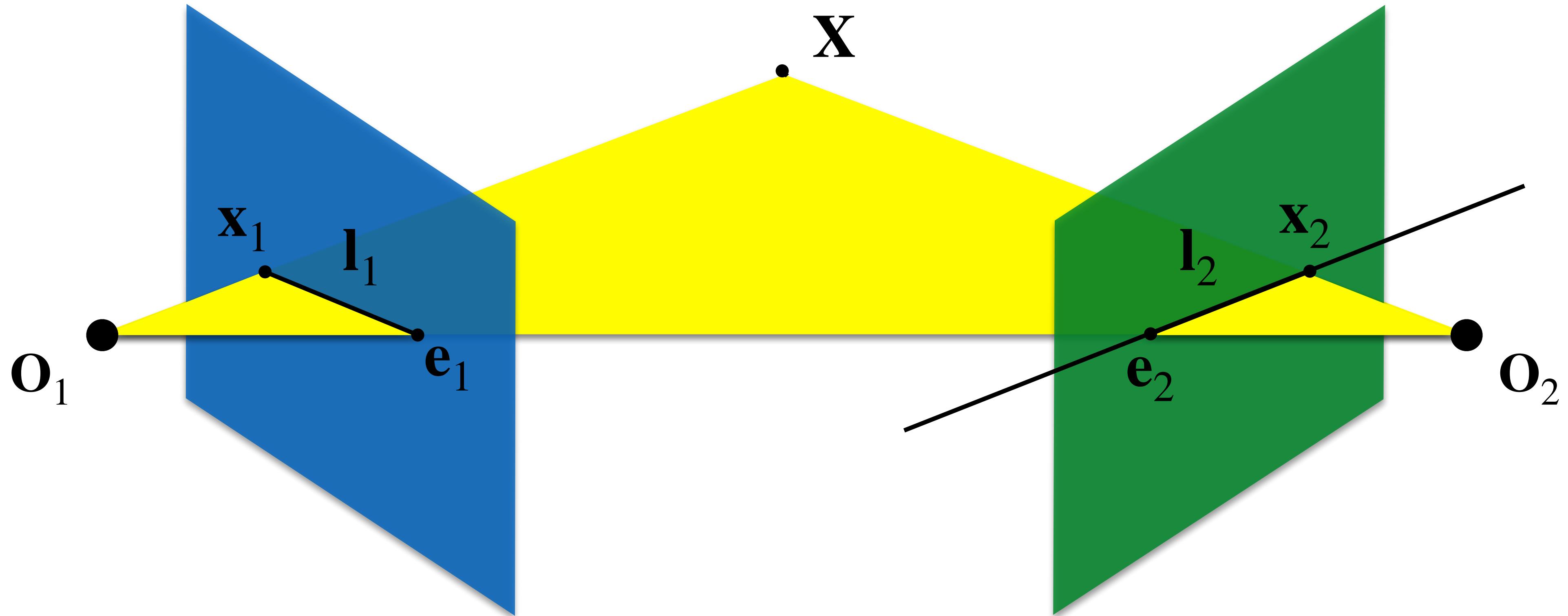
$$\mathbf{F}\tilde{\mathbf{x}}_1 = \mathbf{l}_2$$

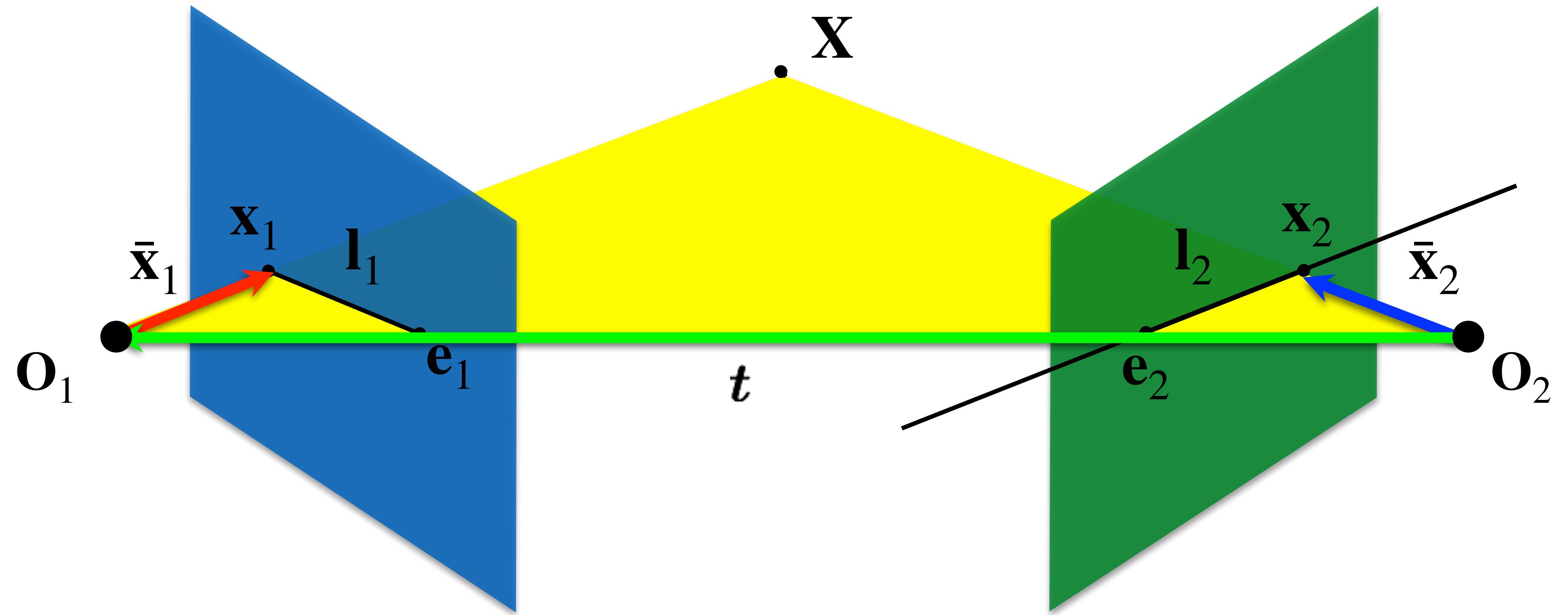
$$\tilde{\mathbf{x}}_2^T \mathbf{F} \tilde{\mathbf{x}}_1 = ?$$

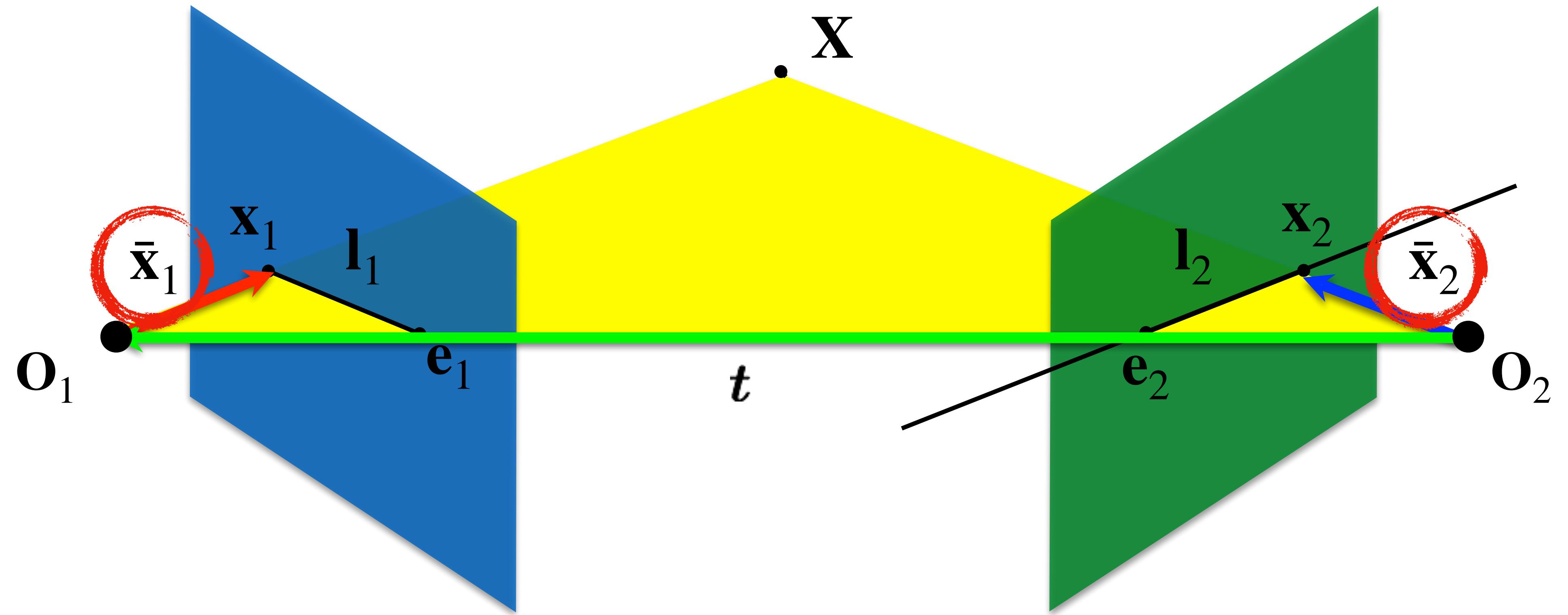


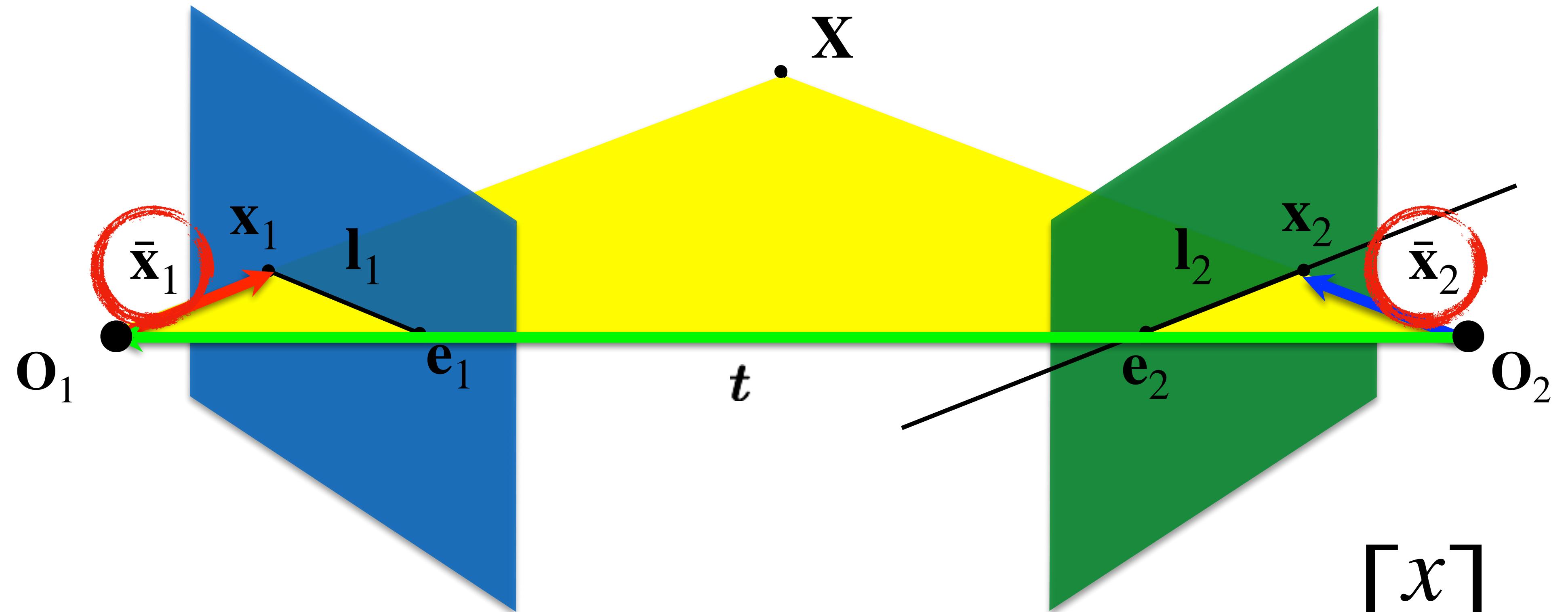
We'll now work off of this constraint to derive an analytical formula for \mathbf{F} .

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







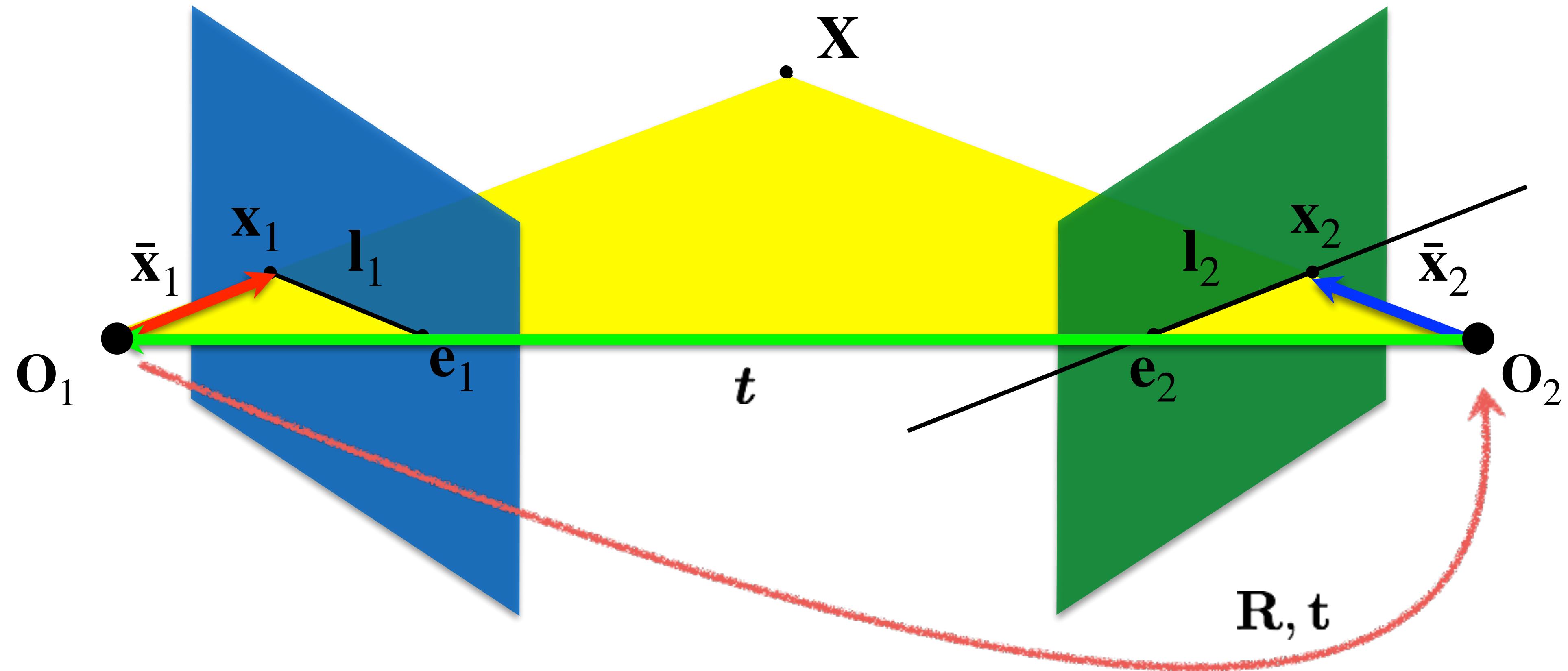


$$\bar{\mathbf{x}} = \mathbf{K}^{-1} \tilde{\mathbf{x}}$$

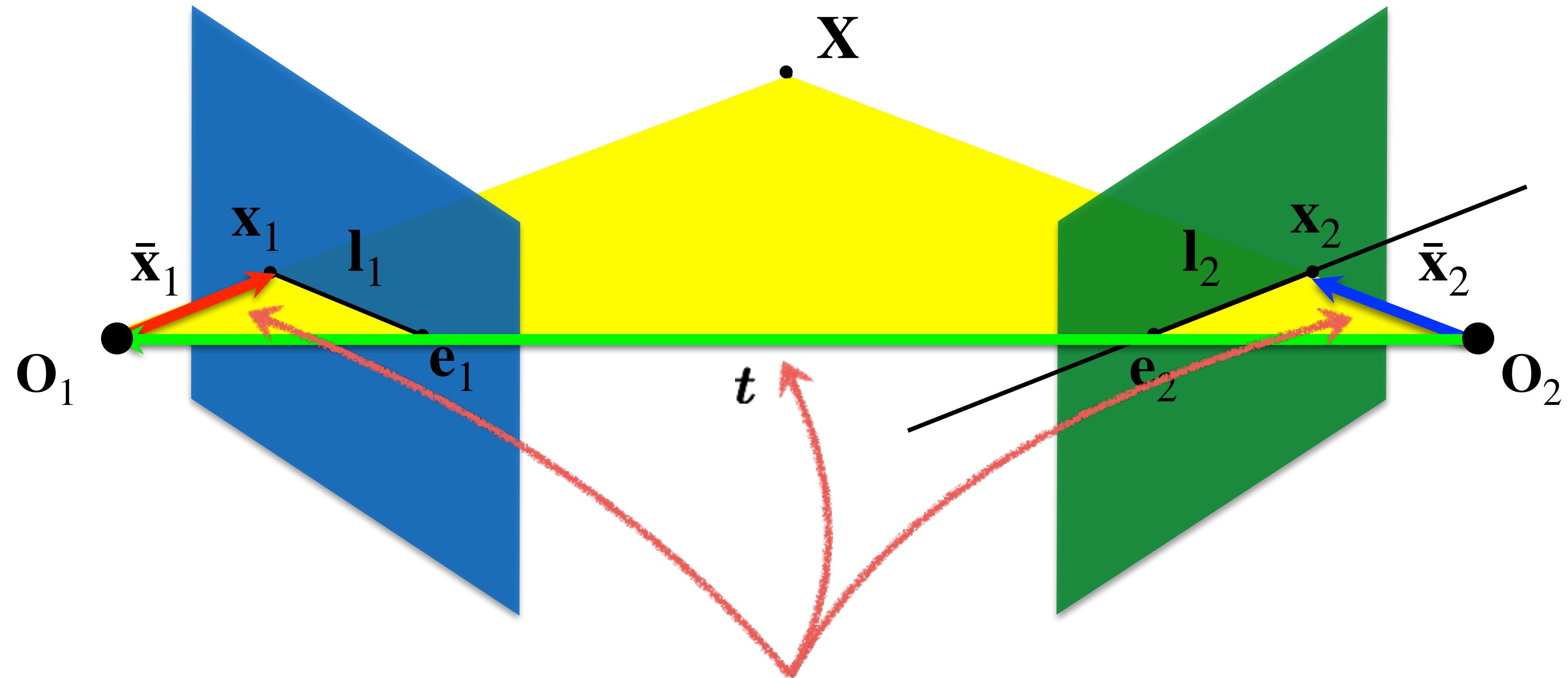
The local ray direction!

$$\tilde{\mathbf{x}} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Homogenized pixel coordinate

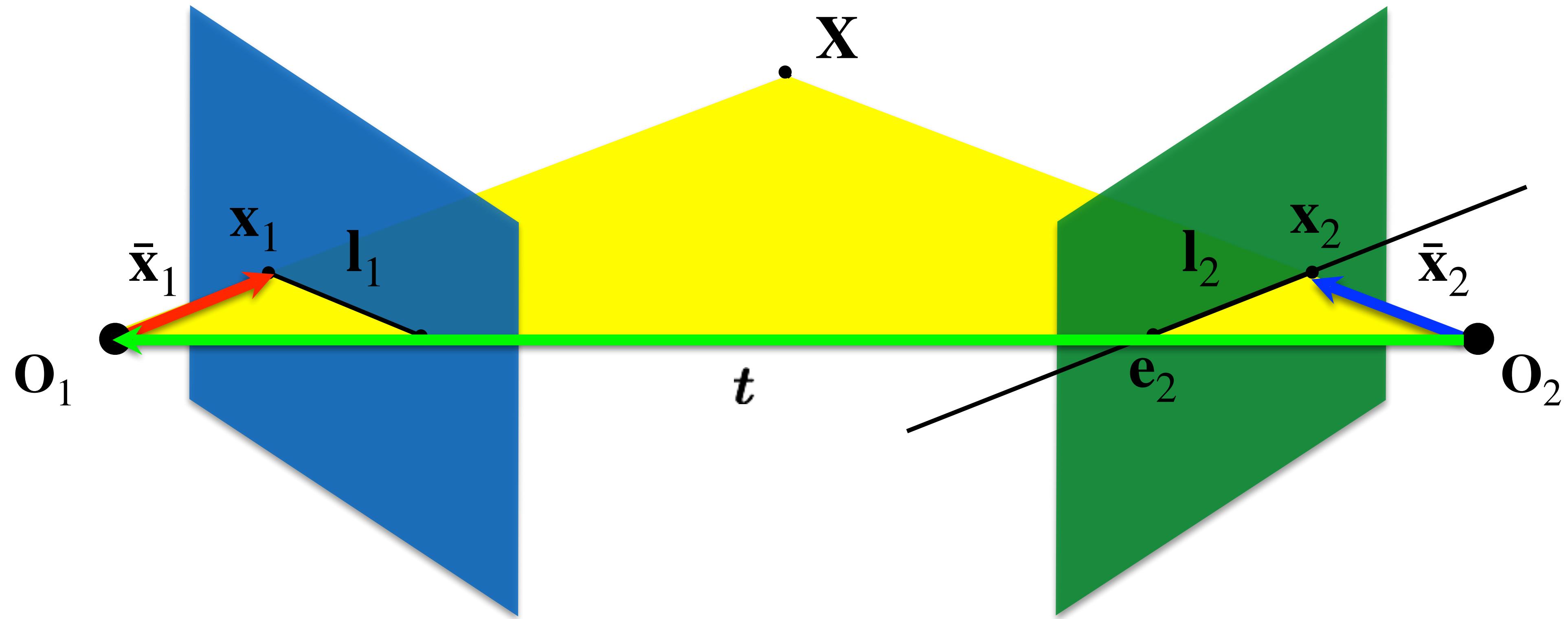


$$\bar{x}_2 = R(\bar{x}_1 - \mathbf{t})$$



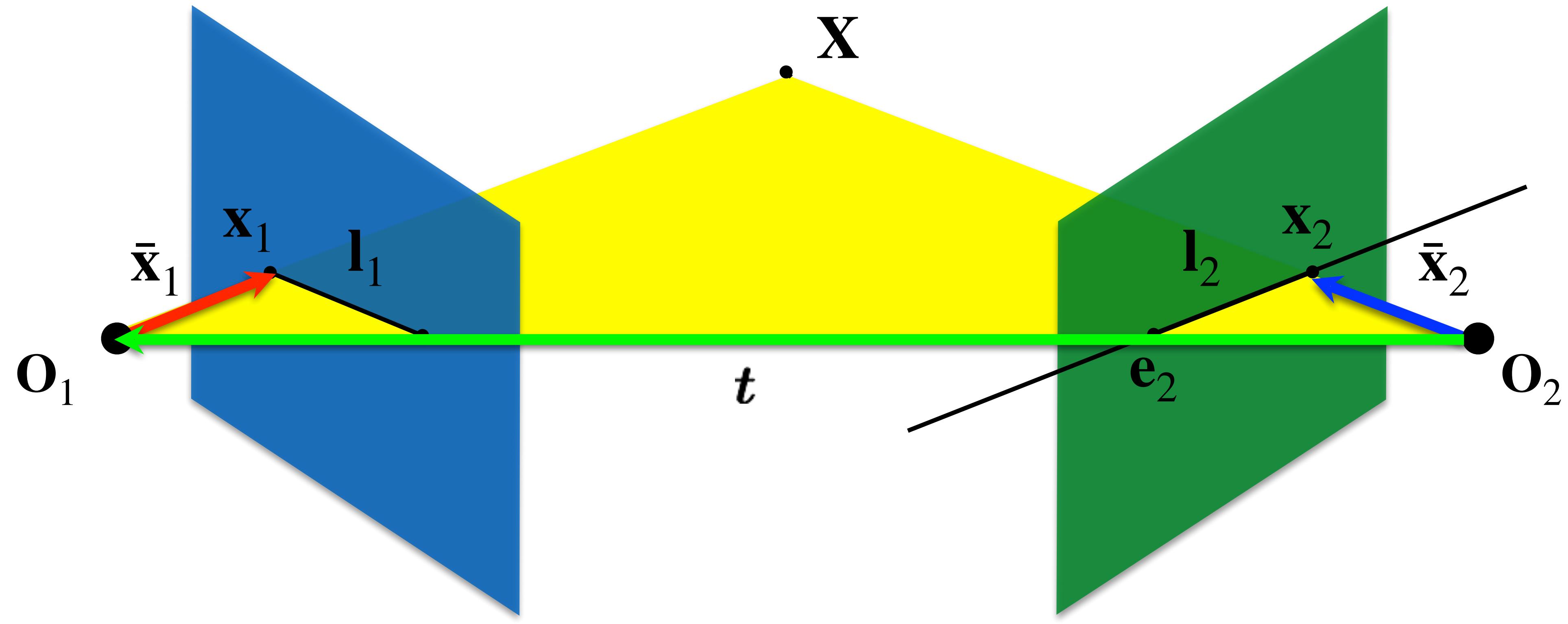
These three vectors are coplanar

$$\bar{\mathbf{x}}_1, \mathbf{t}, \bar{\mathbf{x}}_2$$



If \bar{x}_1, t, \bar{x}_2 are coplanar then

$$\bar{x}_1^T(t \times \bar{x}_1) = 0$$



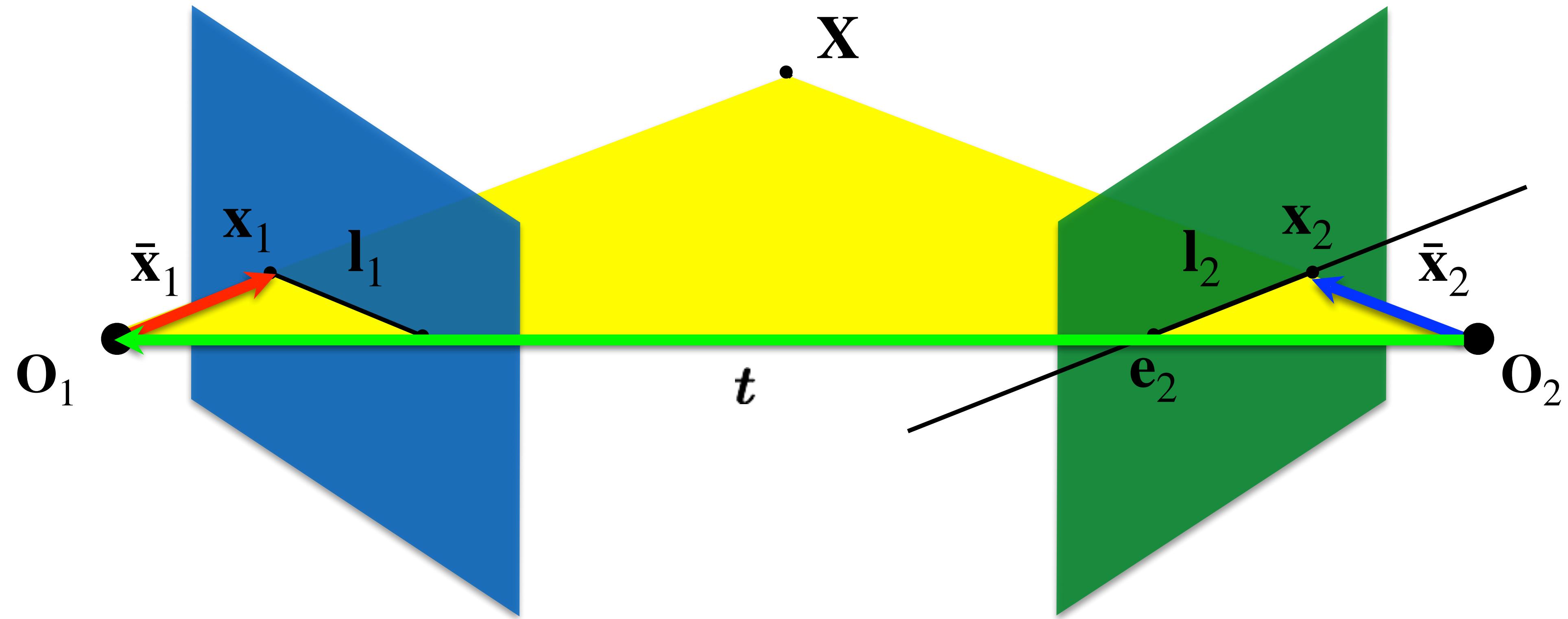
If \bar{x}_1, t, \bar{x}_2 are coplanar then

$$\bar{x}_1^T(t \times \bar{x}_1) = 0$$

dot product of orthogonal vectors

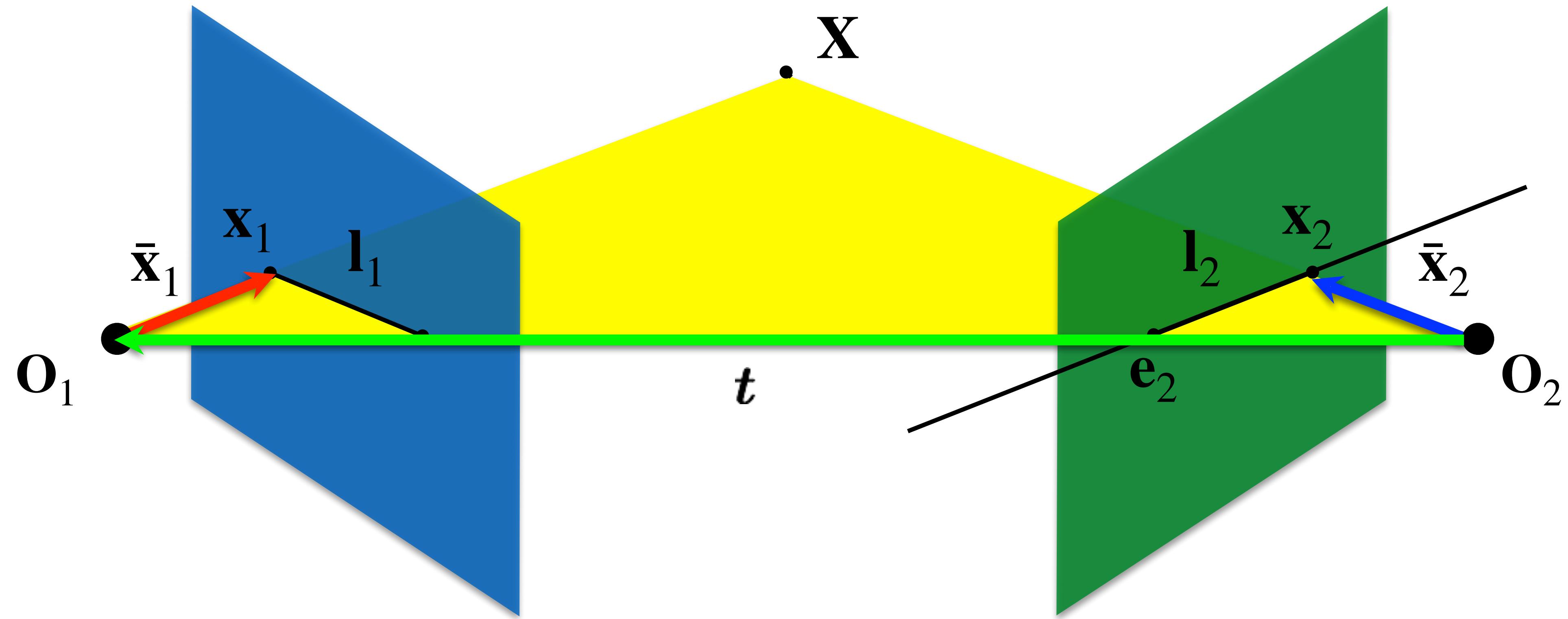


cross-product: vector orthogonal to plane



If \bar{x}_1, t, \bar{x}_2 are coplanar then

$$(\bar{x}_1 - t)^T(t \times \bar{x}_1) = ?$$



If \bar{x}_1, t, \bar{x}_2 are coplanar then

$$(\bar{x}_1 - t)^T(t \times \bar{x}_1) = 0$$

putting it together

rigid motion

$$\bar{\mathbf{x}}_2 = \mathbf{R}(\bar{\mathbf{x}}_1 - \mathbf{t})$$

coplanarity

$$(\bar{\mathbf{x}}_1 - \mathbf{t})^T(\mathbf{t} \times \bar{\mathbf{x}}_1) = 0$$

putting it together

rigid motion

$$\bar{\mathbf{x}}_2 = \mathbf{R}(\bar{\mathbf{x}}_1 - \mathbf{t})$$

coplanarity

$$(\bar{\mathbf{x}}_1 - \mathbf{t})^T(\mathbf{t} \times \bar{\mathbf{x}}_1) = 0$$

$$(\bar{\mathbf{x}}_2^T \mathbf{R})(\mathbf{t} \times \bar{\mathbf{x}}_1) = 0$$

putting it together

rigid motion

$$\bar{\mathbf{x}}_2 = \mathbf{R}(\bar{\mathbf{x}}_1 - \mathbf{t})$$

coplanarity

$$(\bar{\mathbf{x}}_1 - \mathbf{t})^T(\mathbf{t} \times \bar{\mathbf{x}}_1) = 0$$

$$(\bar{\mathbf{x}}_2^T \mathbf{R})(\mathbf{t} \times \bar{\mathbf{x}}_1) = 0$$

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

with cross product matrix $[\mathbf{t}]_\times = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix}$

putting it together

rigid motion

$$\bar{\mathbf{x}}_2 = \mathbf{R}(\bar{\mathbf{x}}_1 - \mathbf{t})$$

coplanarity

$$(\bar{\mathbf{x}}_1 - \mathbf{t})^T (\mathbf{t} \times \bar{\mathbf{x}}_1) = 0$$

$$(\bar{\mathbf{x}}_2^T \mathbf{R})(\mathbf{t} \times \bar{\mathbf{x}}_1) = 0$$

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

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

putting it together

rigid motion

$$\bar{\mathbf{x}}_2 = \mathbf{R}(\bar{\mathbf{x}}_1 - \mathbf{t})$$

coplanarity

$$(\bar{\mathbf{x}}_1 - \mathbf{t})^T (\mathbf{t} \times \bar{\mathbf{x}}_1) = 0$$

$$(\bar{\mathbf{x}}_2^T \mathbf{R})(\mathbf{t} \times \bar{\mathbf{x}}_1) = 0$$

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

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

$$\tilde{\mathbf{x}}_2^T \mathbf{K}_2^{-T} (\mathbf{R}[\mathbf{t}]_\times) \mathbf{K}_1^{-1} \tilde{\mathbf{x}}_1 = 0$$

putting it together

rigid motion

$$\bar{\mathbf{x}}_2 = \mathbf{R}(\bar{\mathbf{x}}_1 - \mathbf{t})$$

coplanarity

$$(\bar{\mathbf{x}}_1 - \mathbf{t})^T (\mathbf{t} \times \bar{\mathbf{x}}_1) = 0$$

$$(\bar{\mathbf{x}}_2^T \mathbf{R})(\mathbf{t} \times \bar{\mathbf{x}}_1) = 0$$

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

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

$$\tilde{\mathbf{x}}_2^T \mathbf{K}_2^{-T} (\mathbf{R}[\mathbf{t}]_\times) \mathbf{K}_1^{-1} \tilde{\mathbf{x}}_1 = 0$$

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

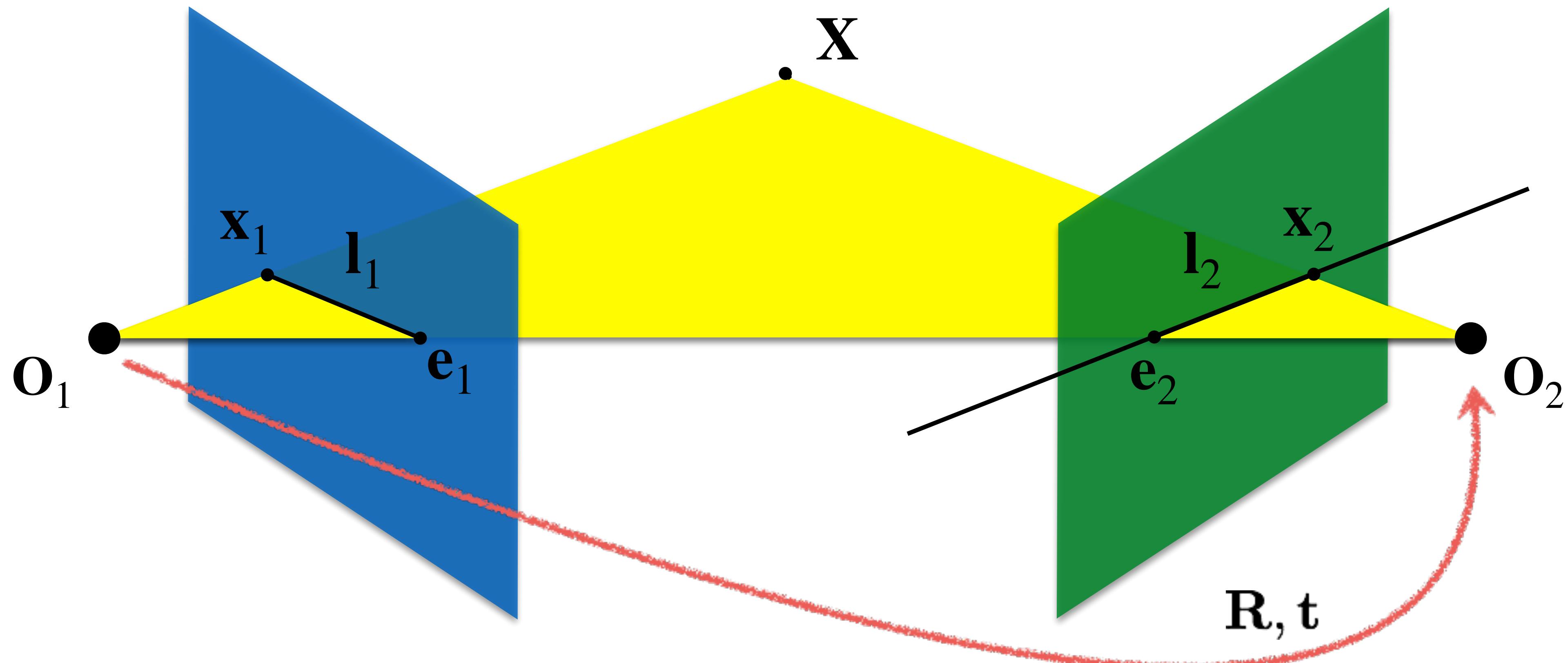
putting it together

$$\mathbf{F} = \mathbf{K}_2^{-T}(\mathbf{R[t]_x})\mathbf{K}_1^{-1}$$

$$\mathbf{F}\tilde{\mathbf{x}}_1 = \mathbf{l}_2$$

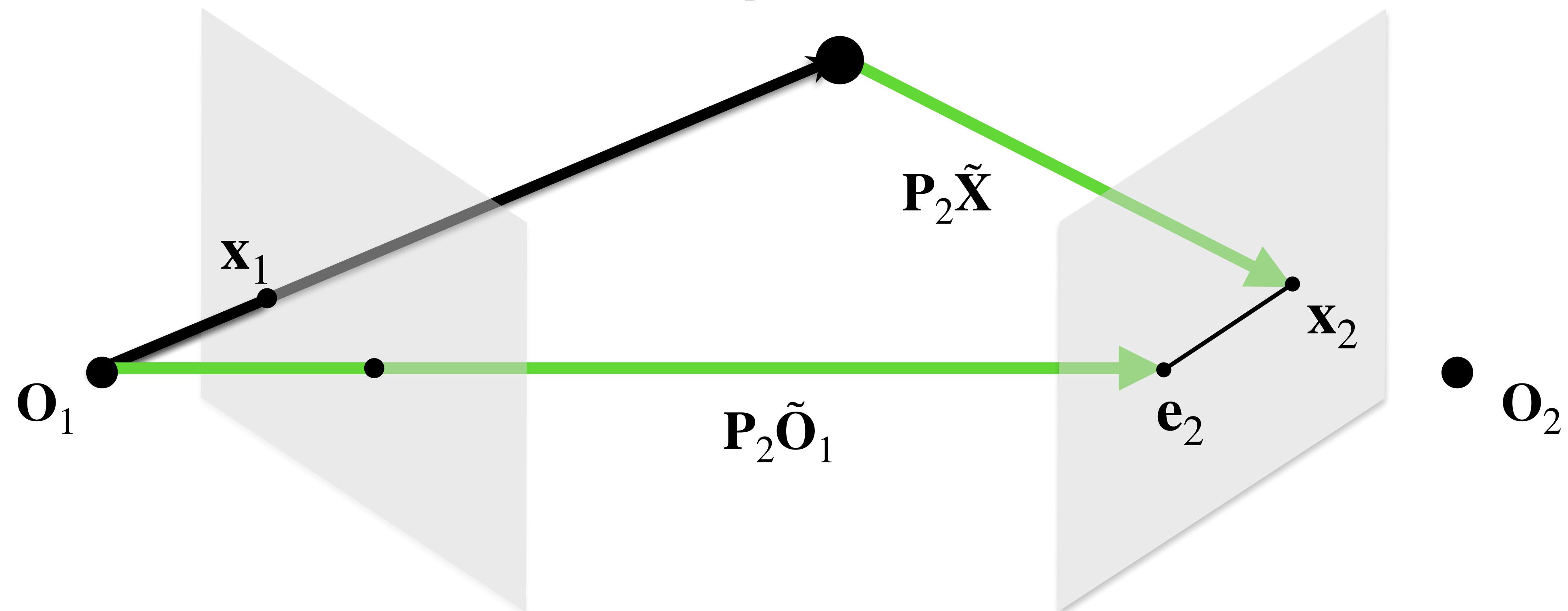
$$\mathbf{F} = \mathbf{K}_2^{-T}(\mathbf{R}[t]_{\times})\mathbf{K}_1^{-1}$$

$$\mathbf{F}\tilde{\mathbf{x}}_1 = \mathbf{l}_2$$

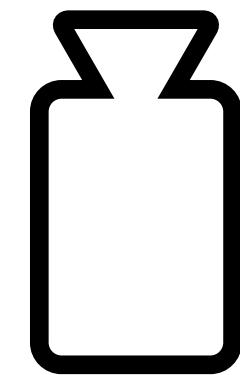
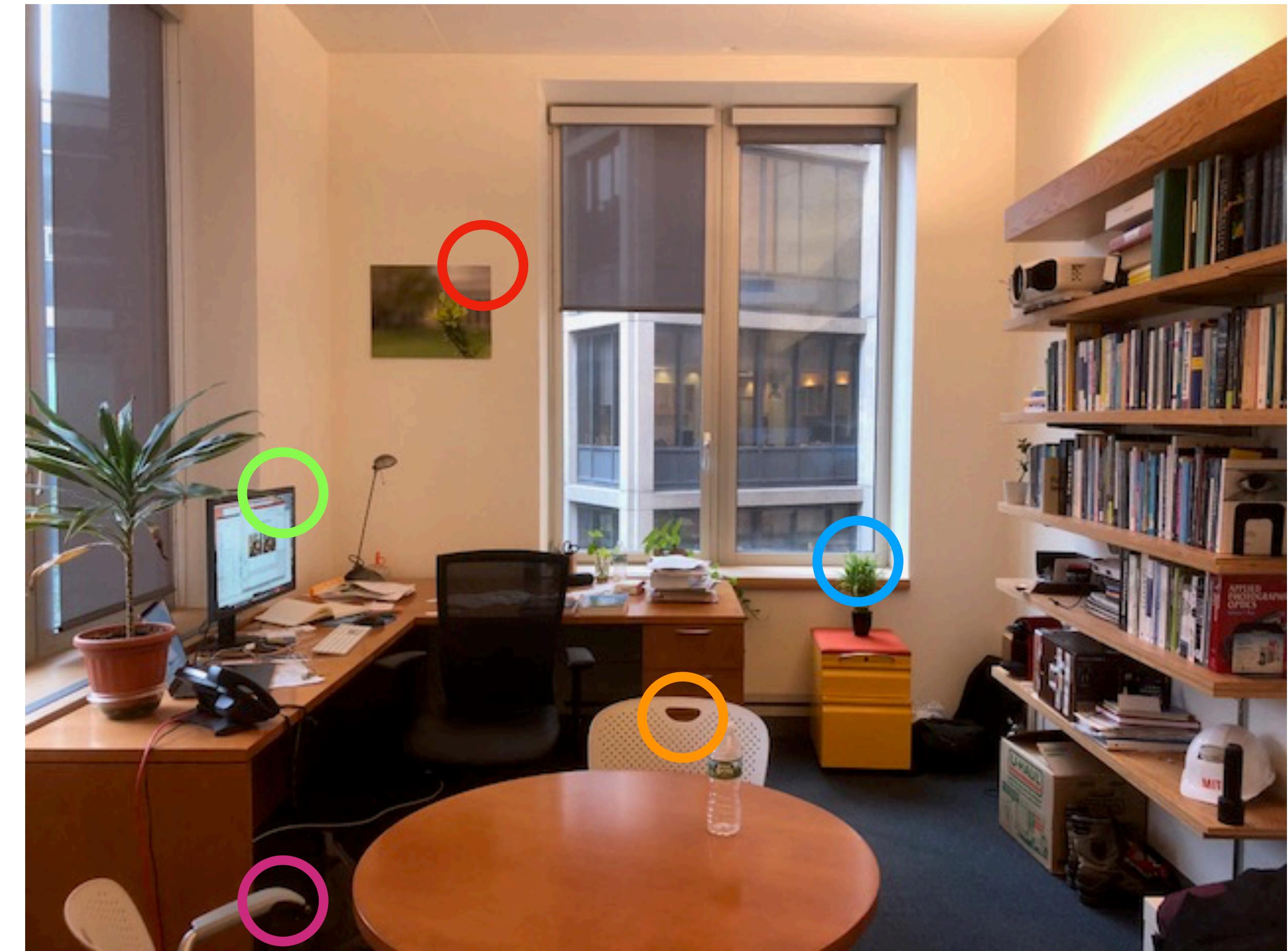
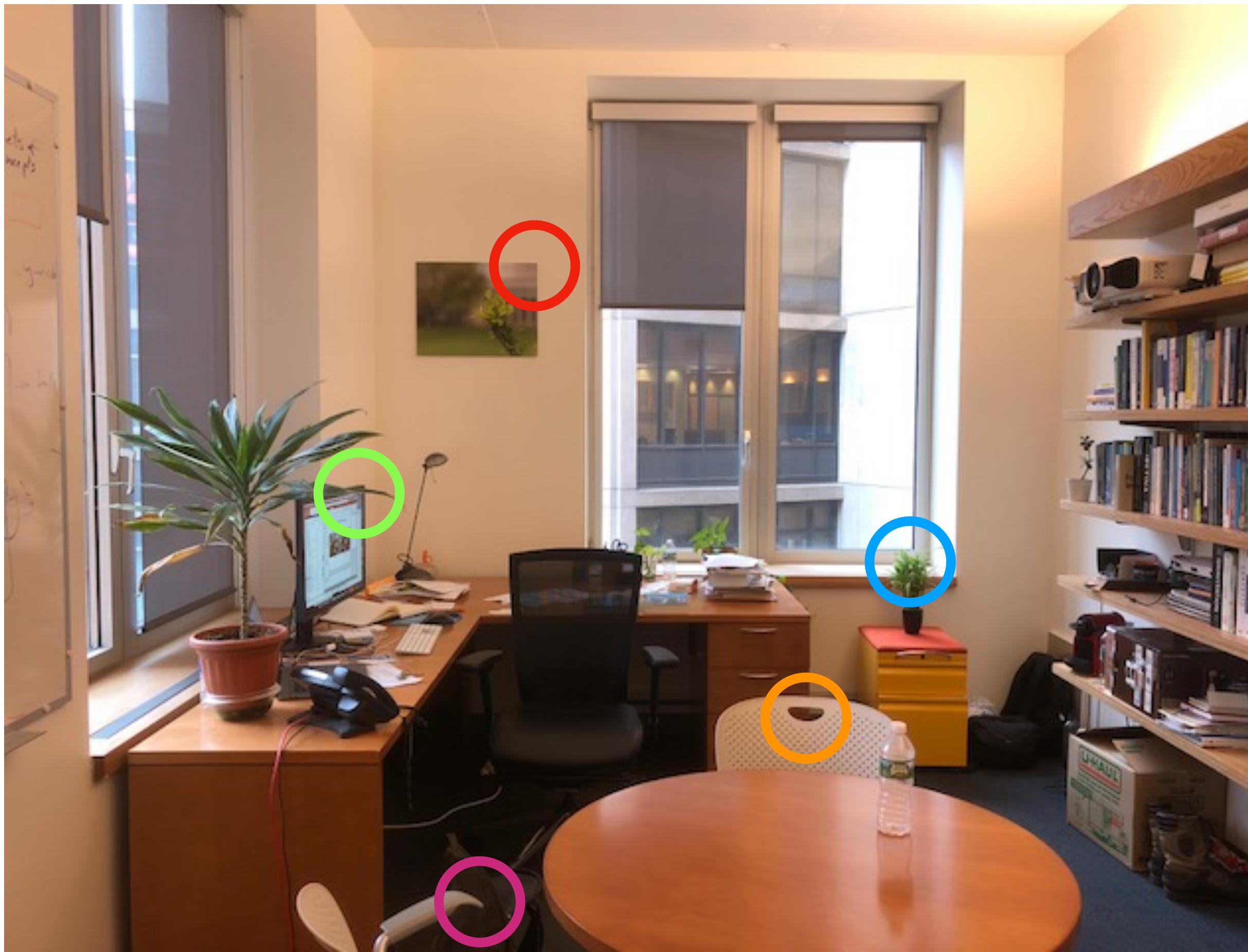


Epipolar Lines: The Hacky Way

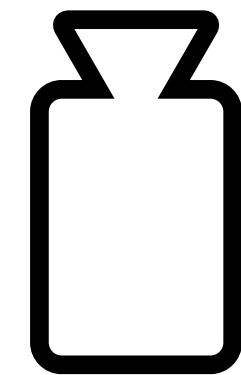
$$\mathbf{X} = \mathbf{R}_1^{C2W} \mathbf{K}_1^{-1} \tilde{\mathbf{x}}_1 + \mathbf{O}_1$$



This always works ;)

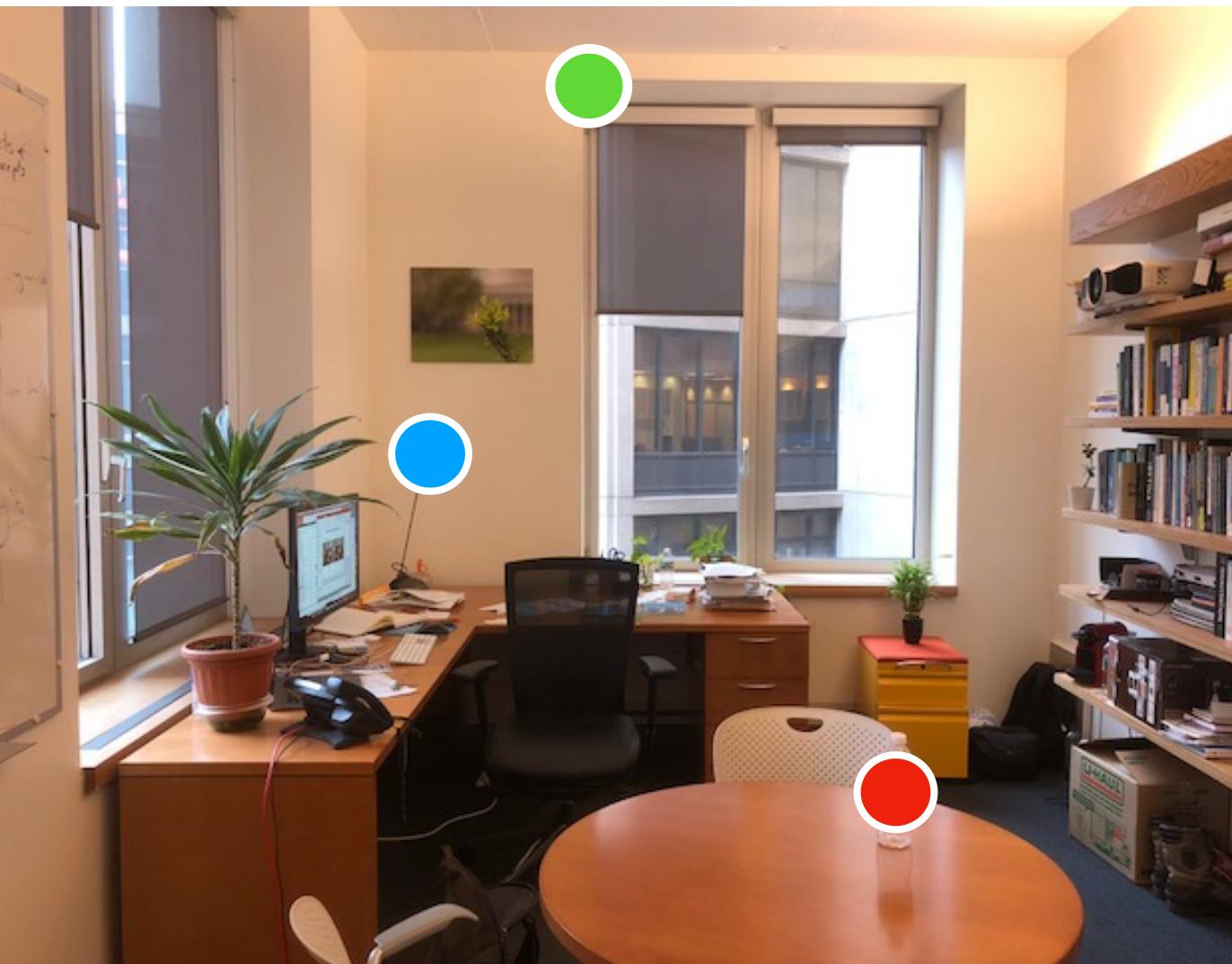
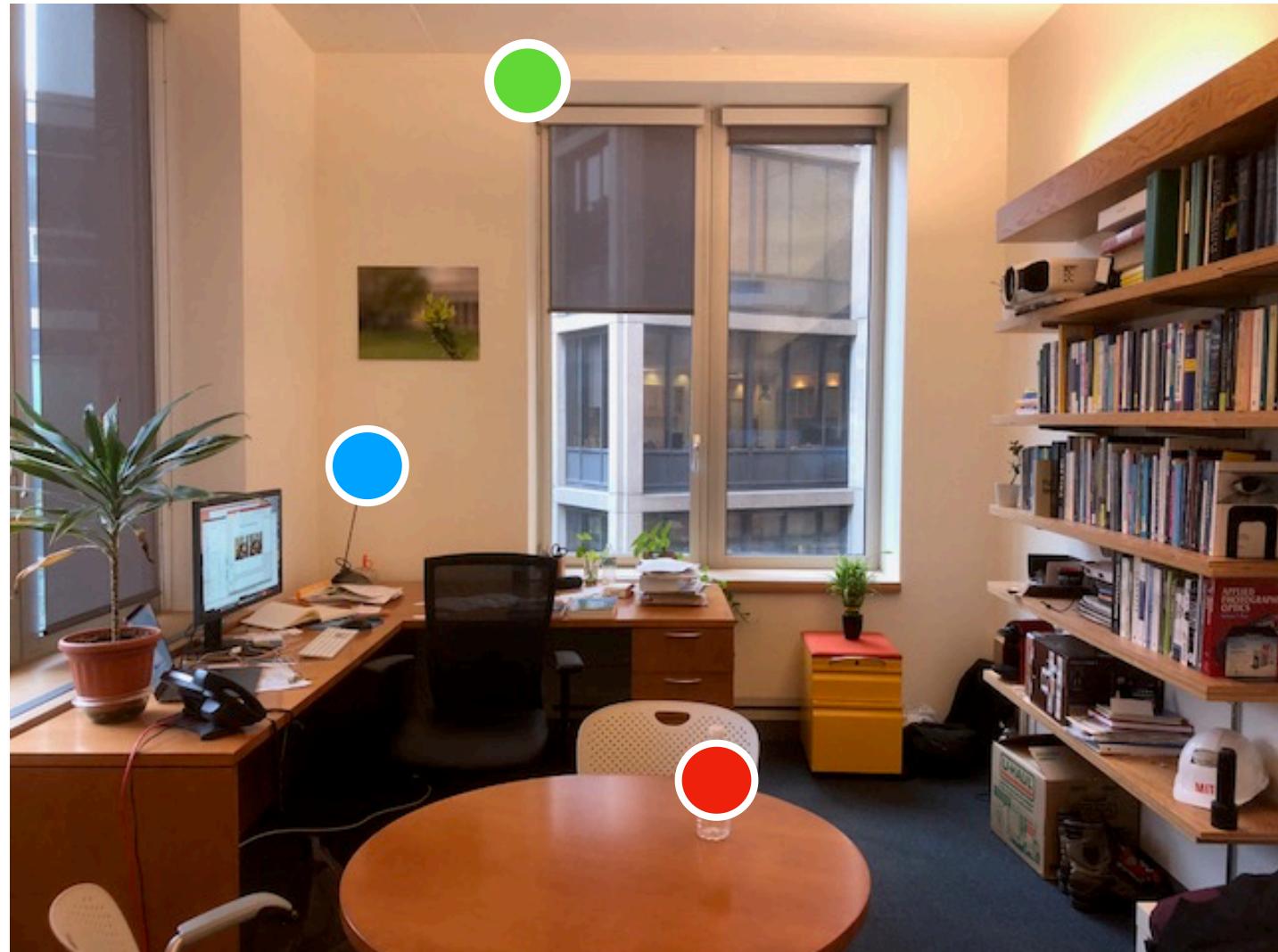


What if P_1, P_2 aren't known?



Finding correspondences

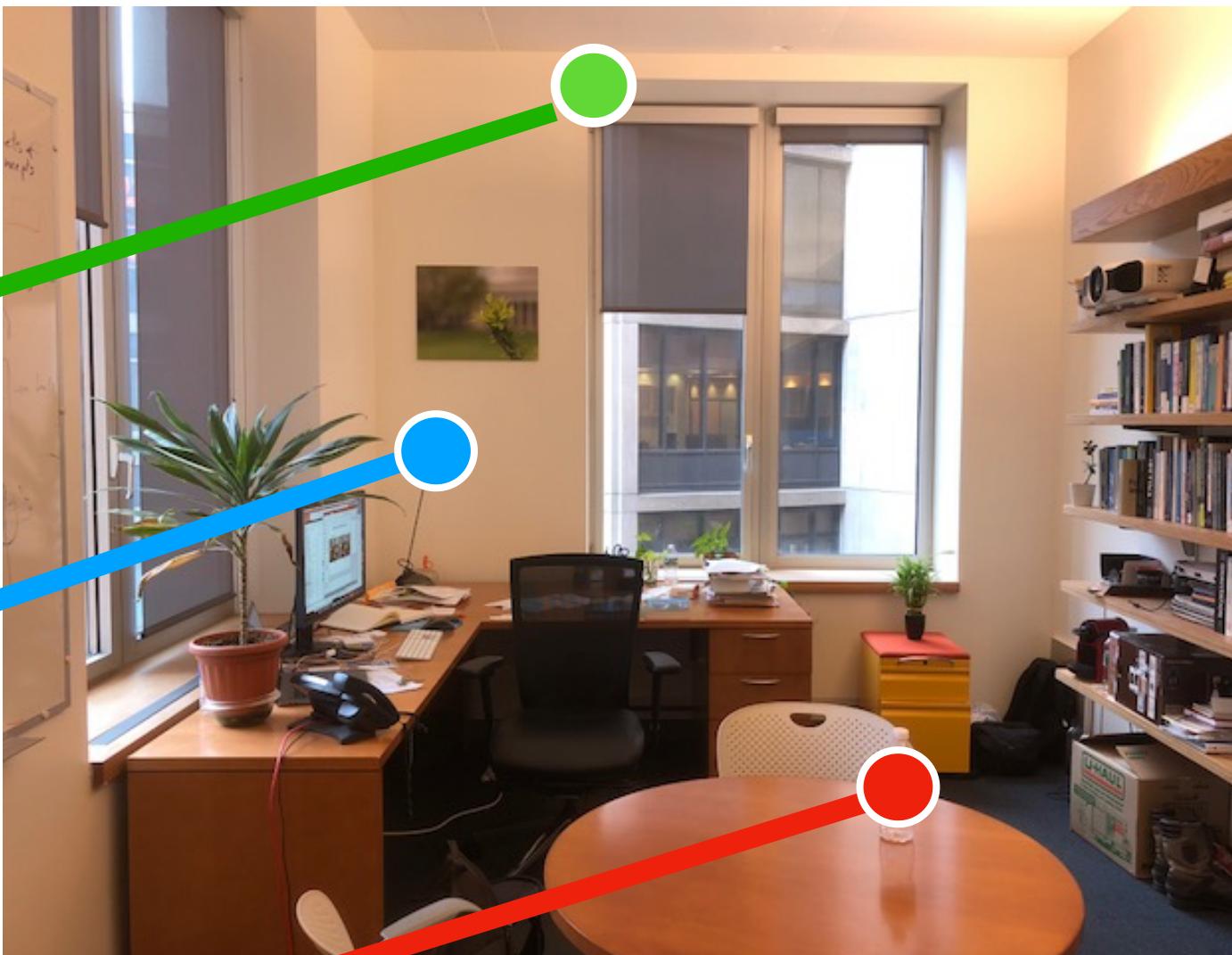
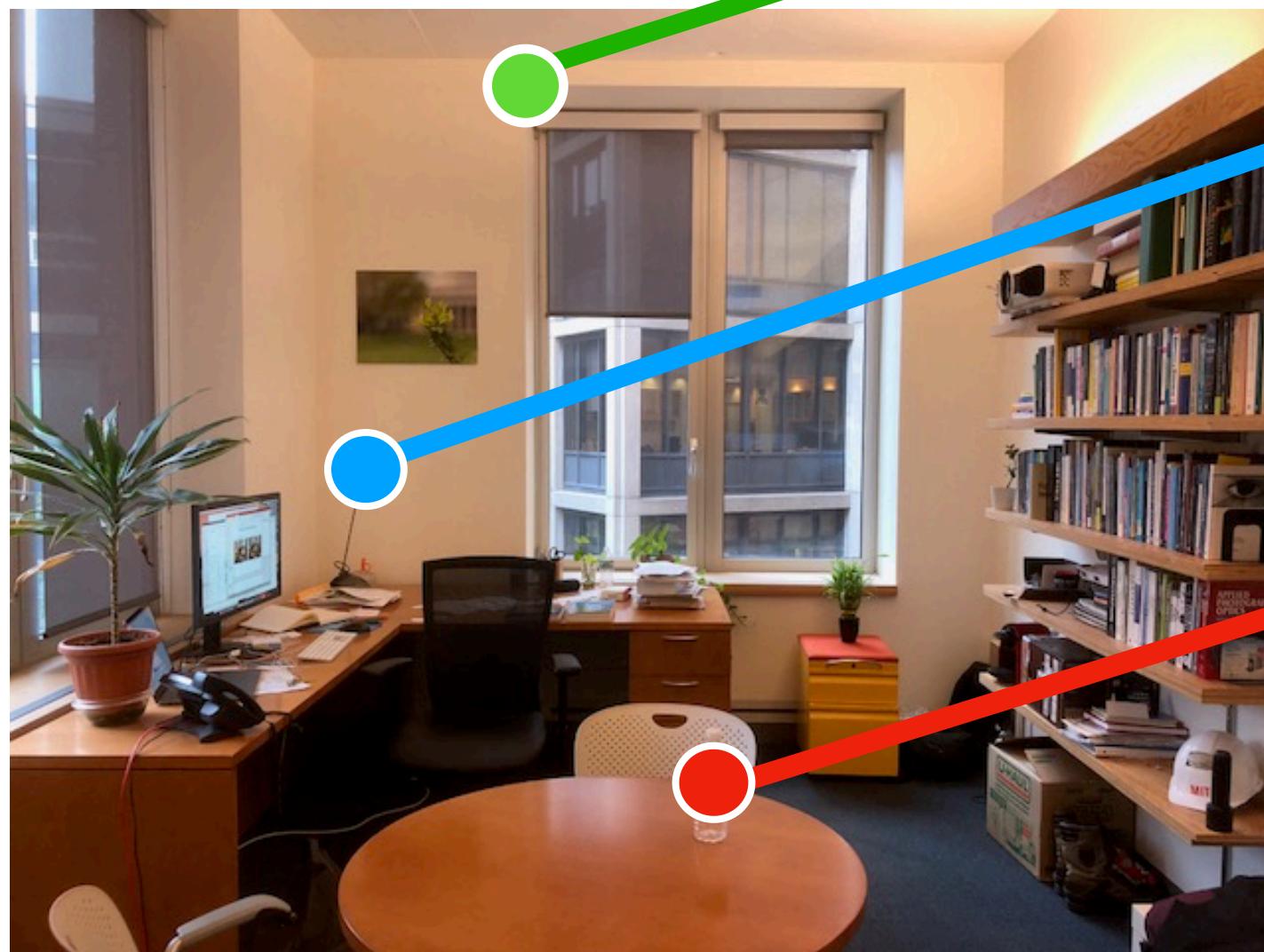
Match features between each pair of images



- Need to find **a lot of candidates.**
- Typical algorithm for keypoint detection & descriptor computation: SIFT
- Outlier rejection with RANSAC (no time to talk about that, but very cool :)

Finding correspondences

Match features between
each pair of images



- Need to find **a lot of candidates.**
- Typical algorithm for keypoint detection & descriptor computation: SIFT
- Outlier rejection with RANSAC (no time to talk about that, but very cool :)

The Eight-Point Algorithm

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

$$\begin{bmatrix} x_1, y_1, 1 \end{bmatrix} \begin{bmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = 0$$

The Eight-Point Algorithm

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

Idea: Leverage epipolar constraint to estimate \mathbf{F} from correspondences!

The Eight-Point Algorithm

$$\tilde{\mathbf{x}}_2^T \mathbf{F} \tilde{\mathbf{x}}_1 = 0 \quad \begin{bmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{bmatrix} = 0$$
$$[x_1x_2, x_1y_2, x_1, yx_2, yy_2, y, x_2, y_2, 1]$$

The Eight-Point Algorithm

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

$$\mathbf{W}\mathbf{f} = 0 \quad \text{with } \mathbf{W} \in \mathbb{R}^{8 \times 9}, \text{ i.e., 8 correspondences}$$

stacked on top of each other

The Eight-Point Algorithm

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

$$\mathbf{W}\mathbf{f} = 0 \quad \text{with } \mathbf{W} \in \mathbb{R}^{8 \times 9}, \text{ i.e., 8 correspondences stacked on top of each other}$$

By determining the null-space of \mathbf{W} , we can determine \mathbf{f} up to scale.

The Eight-Point Algorithm

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

$$\mathbf{W}\mathbf{f} = 0 \quad \text{with } \mathbf{W} \in \mathbb{R}^{8 \times 9}, \text{ i.e., 8 correspondences stacked on top of each other}$$

By determining the null-space of \mathbf{W} , we can determine \mathbf{f} up to scale.

$$\mathbf{F} = \mathbf{K}_2^{-T} (\mathbf{R}[\mathbf{t}]_{\times}) \mathbf{K}_1^{-1}$$

If \mathbf{K}_i are known, we can then back out \mathbf{R} and \mathbf{t} .

If not, need additional constraints.

None of this is straightforward.

The 8-Point Algorithm as an Inductive Bias for Relative Pose Prediction by ViTs

Chris Rockwell, Justin Johnson, David F. Fouhey
University of Michigan

Abstract

We present a simple baseline for directly estimating the relative pose (rotation and translation, including scale) between two images. Deep methods have recently shown strong progress but often require complex or multi-stage architectures. We show that a handful of modifications can be applied to a Vision Transformer (ViT) to bring its computations close to the Eight-Point Algorithm. This inductive bias enables a simple method to be competitive in multiple settings, often substantially improving over the state of the art with strong performance gains in limited data regimes.

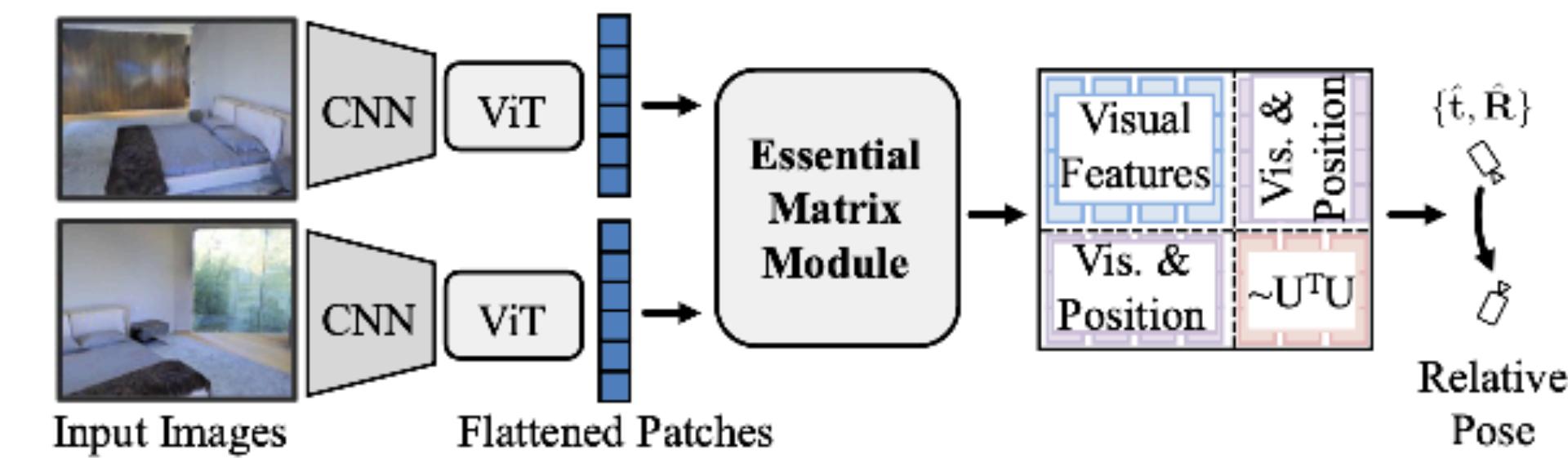


Figure 1. We propose three small modifications to a ViT via the Essential Matrix Module, enabling computations similar to the Eight-Point algorithm. The resulting mix of visual and positional features is a good inductive bias for pose estimation.

challenge in the wide-baseline setting, and the conversion

Bundle Adjustment

Triangulation

How to compute 3D locations of point correspondences if cameras are known.

Epipolar Lines

Which pixels in two cameras observe same 3D point?
Where to look for multi-view correspondences?

Fundamental & Essential Matrices

Elegant formulation of Epipolar Lines

A way of estimating camera poses, intrinsics, and extrinsic from correspondences.

No Time

Correspondences
RANSAC
Incremental
Bundle
Adjustment
Practically solving
for \mathbf{F} and \mathbf{K}

Read:
Computer Vision:
Algorithms and
Applications, 2nd
ed.

What has changed since Deep Learning?

Bundle Adjustment

Triangulation

How to compute 3D locations of point correspondences if cameras are known.

Epipolar Lines

Which pixels in two cameras observe same 3D point?
Where to look for multi-view correspondences?

Fundamental & Essential Matrices

Elegant formulation of Epipolar Lines
A way of estimating camera poses, intrinsics, and extrinsic from correspondences.

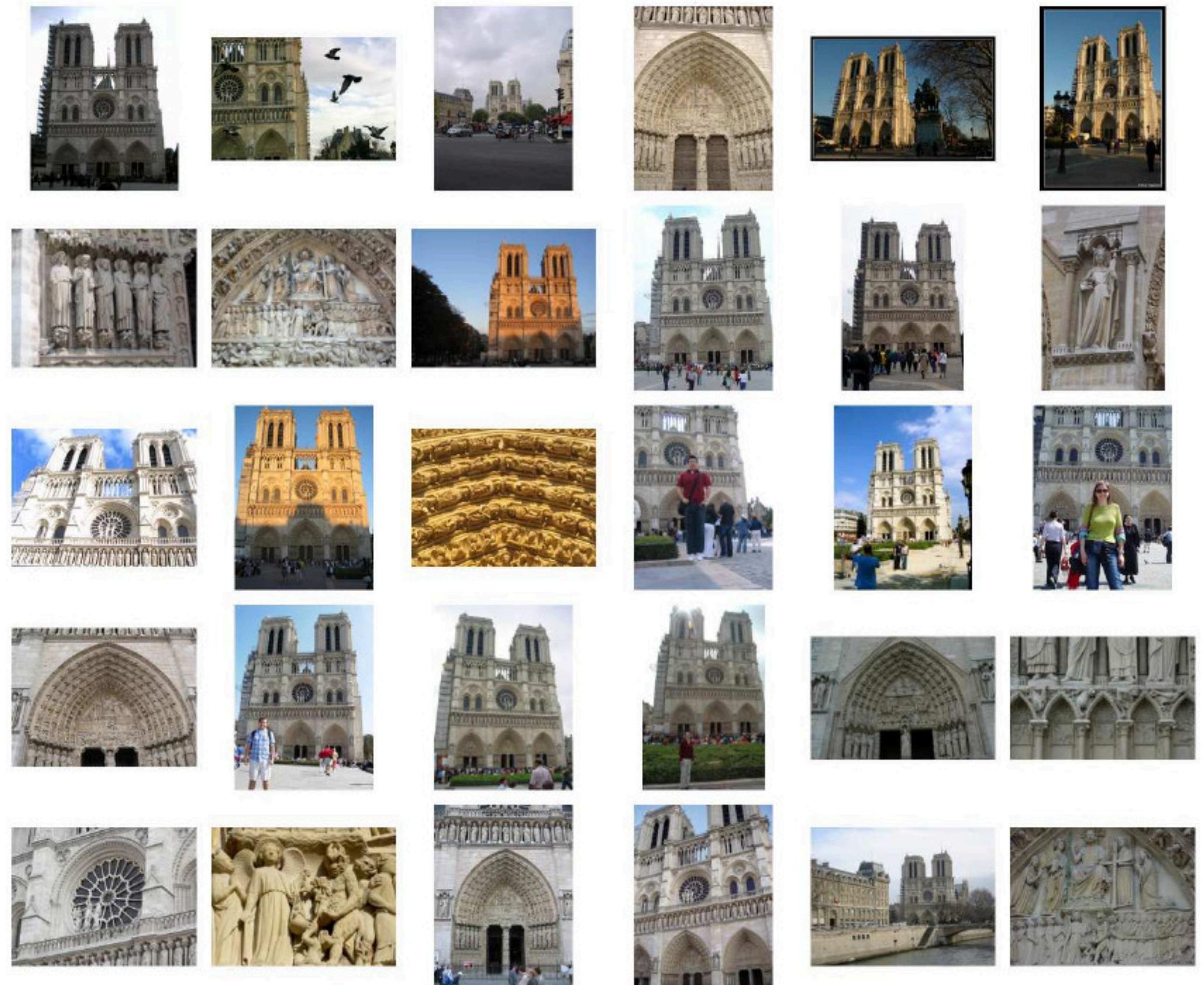
No Time

-
-
-

What has changed since Deep Learning?

What if we have **many** views?

Bundle Adjustment



► **Goal: Optimize reprojection errors** (distance between observed feature and projected 3D point in image plane) **wrt. camera parameters and 3D point cloud**

Bundle Adjustment

Let $\Pi = \{\pi_i\}$ denote the N cameras including their intrinsic and extrinsic parameters.

Let $\mathcal{X}_w = \{\mathbf{x}_p^w\}$ with $\mathbf{x}_p^w \in \mathbb{R}^3$ denote the set of P 3D points in world coordinates.

Let $\mathcal{X}_s = \{\mathbf{x}_{ip}^s\}$ with $\mathbf{x}_{ip}^s \in \mathbb{R}^2$ denote the image (screen) observations in all i cameras.

Bundle Adjustment

Let $\Pi = \{\pi_i\}$ denote the N cameras including their intrinsic and extrinsic parameters.

Let $\mathcal{X}_w = \{\mathbf{x}_p^w\}$ with $\mathbf{x}_p^w \in \mathbb{R}^3$ denote the set of P 3D points in world coordinates.

Let $\mathcal{X}_s = \{\mathbf{x}_{ip}^s\}$ with $\mathbf{x}_{ip}^s \in \mathbb{R}^2$ denote the image (screen) observations in all i cameras.

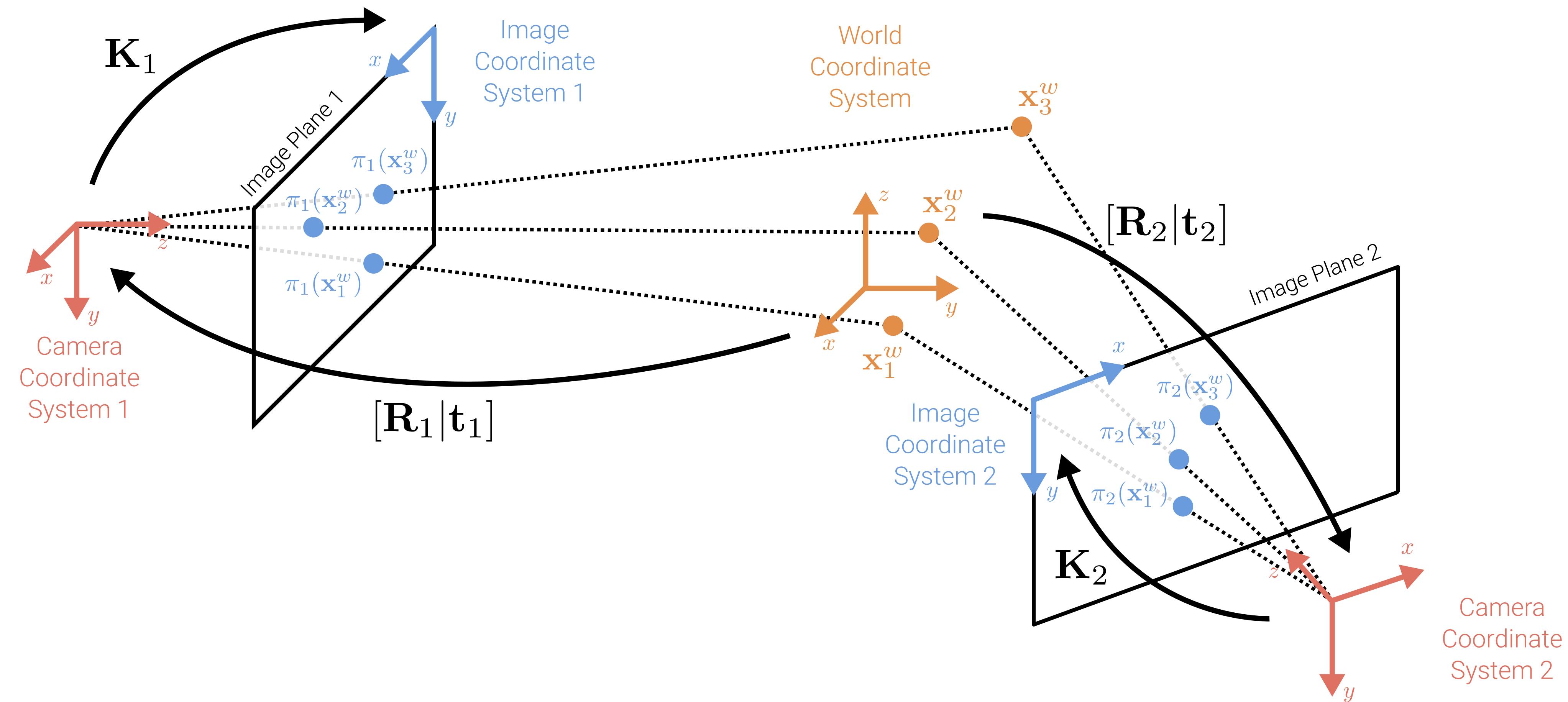
Bundle adjustment minimizes the reprojection error of all observations:

$$\Pi^*, \mathcal{X}_w^* = \underset{\Pi, \mathcal{X}_w}{\operatorname{argmin}} \sum_{i=1}^N \sum_{p=1}^P w_{ip} \|\mathbf{x}_{ip}^s - \pi_i(\mathbf{x}_p^w)\|_2^2$$

Here, w_{ip} indicates if point p is observed in image i and $\pi_i(\mathbf{x}_p^w)$ is the 3D-to-2D projection of 3D world point \mathbf{x}_p^w onto the 2D image plane of the i 'th camera, i.e.:

$$\pi_i(\mathbf{x}_p^w) = \begin{pmatrix} \tilde{x}_p^s / \tilde{w}_p^s \\ \tilde{y}_p^s / \tilde{w}_p^s \end{pmatrix} \quad \text{with} \quad \tilde{\mathbf{x}}_p^s = \mathbf{K}_i (\mathbf{R}_i \mathbf{x}_p^w + \mathbf{t}_i)$$

Bundle Adjustment



\mathbf{K}_i and $[\mathbf{R}_i | \mathbf{t}_i]$ are the intrinsic and extrinsic parameters of π_i , respectively.

During bundle adjustment, we optimize $\{(\mathbf{K}_i, \mathbf{R}_i, \mathbf{t}_i)\}$ and $\{\mathbf{x}_p^w\}$ jointly.

Challenges of Bundle Adjustment

Initialization:

- ▶ The energy landscape of the bundle adjustment problem is highly **non-convex**
- ▶ A good **initialization** is crucial to avoid getting trapped in bad local minima
- ▶ As initializing all 3D points and cameras jointly is difficult (occlusion, viewpoint, matching outliers), **incremental bundle adjustment** initializes with a carefully selected two-view reconstruction and iteratively adds new images/cameras

Challenges of Bundle Adjustment

Initialization:

- ▶ The energy landscape of the bundle adjustment problem is highly **non-convex**
- ▶ A good **initialization** is crucial to avoid getting trapped in bad local minima
- ▶ As initializing all 3D points and cameras jointly is difficult (occlusion, viewpoint, matching outliers), **incremental bundle adjustment** initializes with a carefully selected two-view reconstruction and iteratively adds new images/cameras

Optimization:

- ▶ Given millions of features and thousands of cameras, large-scale bundle adjustment is **computationally demanding** (cubic complexity in #unknowns)
- ▶ Luckily, the problem is **sparse** (not all 3D points are observed in every camera), and efficient sparse implementations (e.g., Ceres) can be exploited in practice

Results and Applications

COLMAP SfM



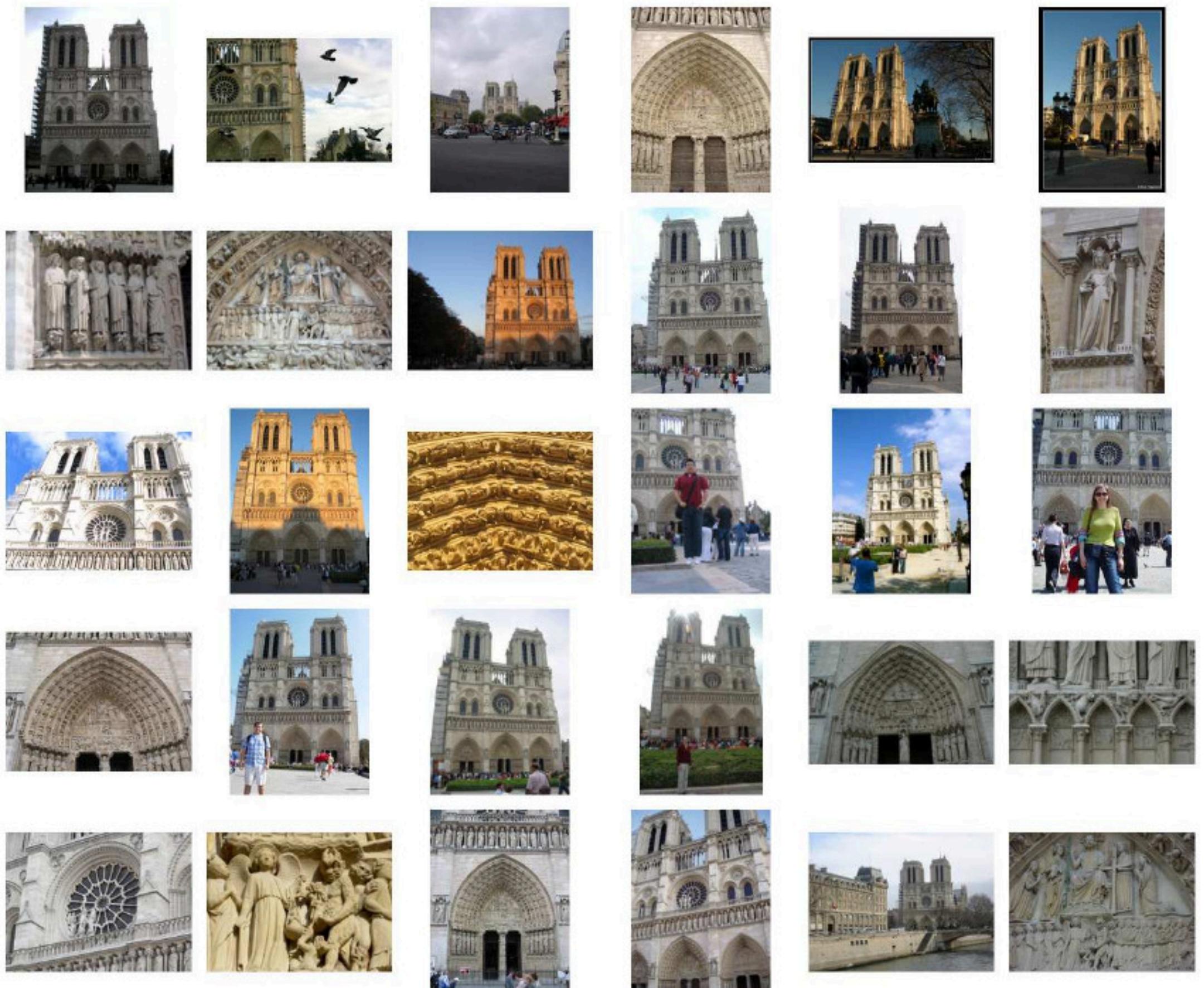
- **COLMAP** significantly improves accuracy and robustness compared to prior work

COLMAP MVS



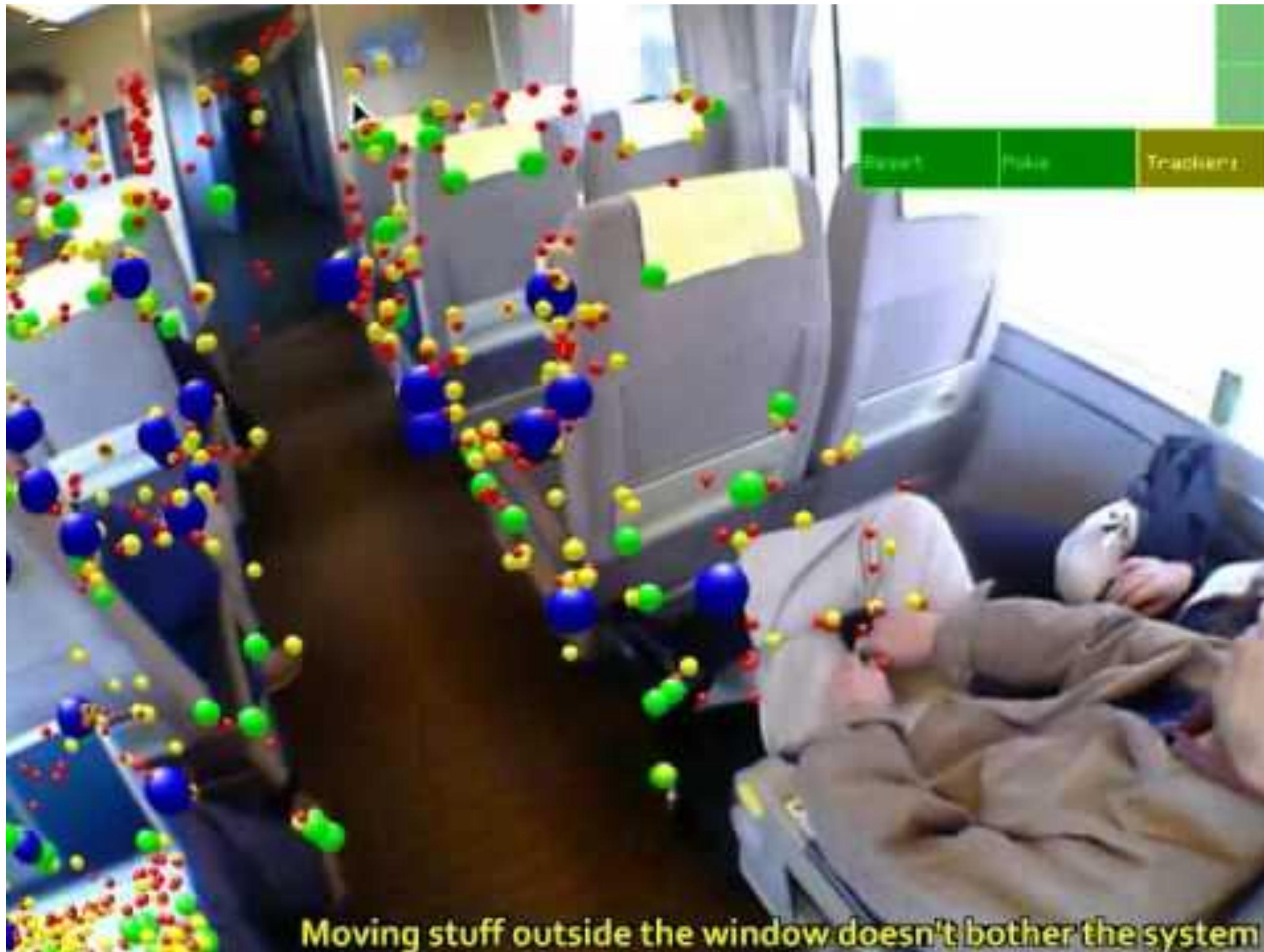
- COLMAP features a second **multi-view stereo stage** to obtain dense geometry

Photo Tourism



► **Photo Tourism / PhotoSynth** allows for exploring photo collections in 3D

Parallel Tracking and Mapping (PTAM)



- **PTAM** demonstrates real-time tracking and mapping of small workspaces

Bundle Adjustment

Triangulation

How to compute 3D locations of point correspondences if cameras are known.

Epipolar Lines

Which pixels in two cameras observe same 3D point?

Where to look for multi-view correspondences?

Fundamental & Essential Matrices

Elegant formulation of Epipolar Lines

A way of *estimating camera poses, intrinsics, and extrinsic from correspondences.*

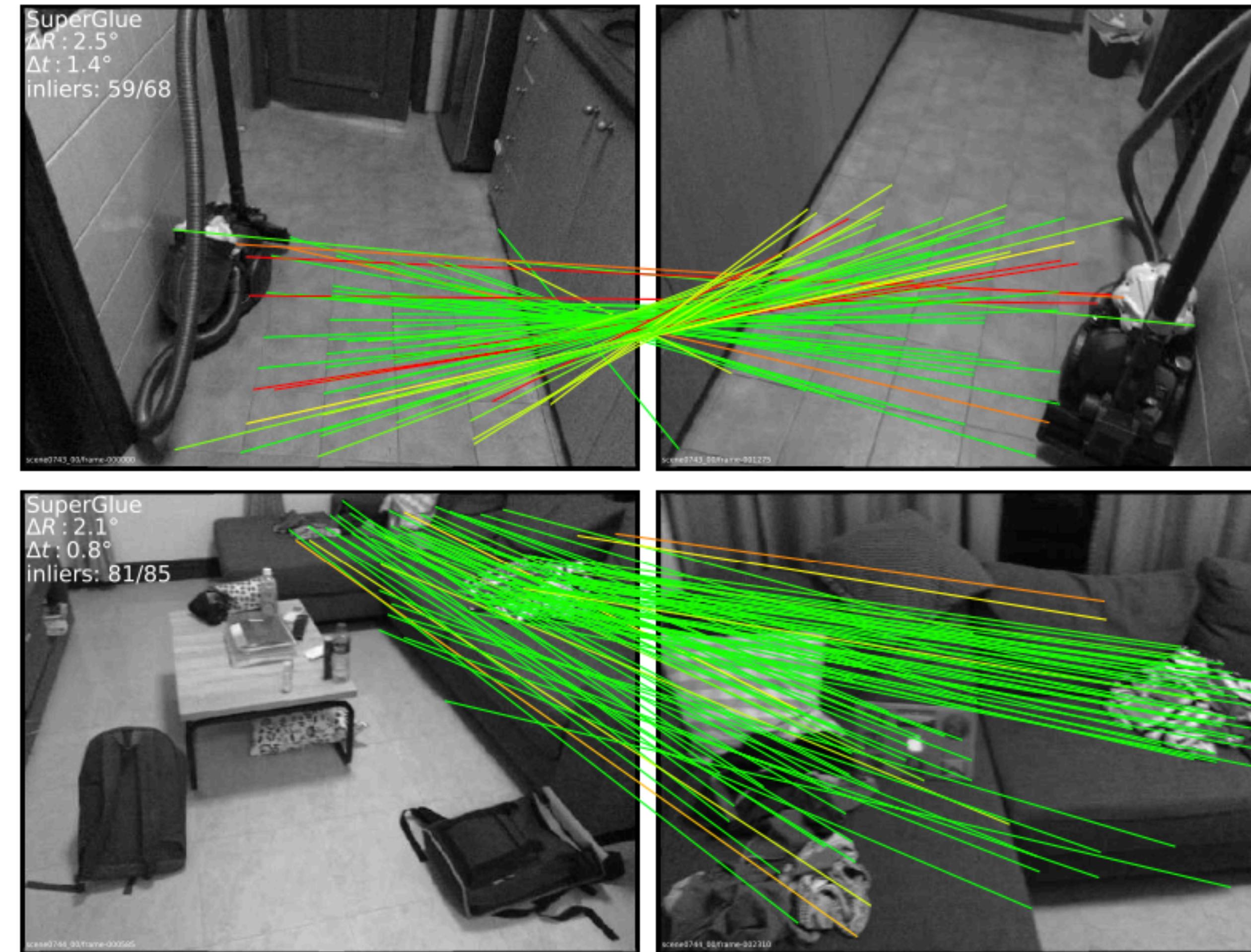
No Time

-
-
-

What has changed since Deep Learning?

SuperGlue: Learning Feature Matching with Graph Neural Networks

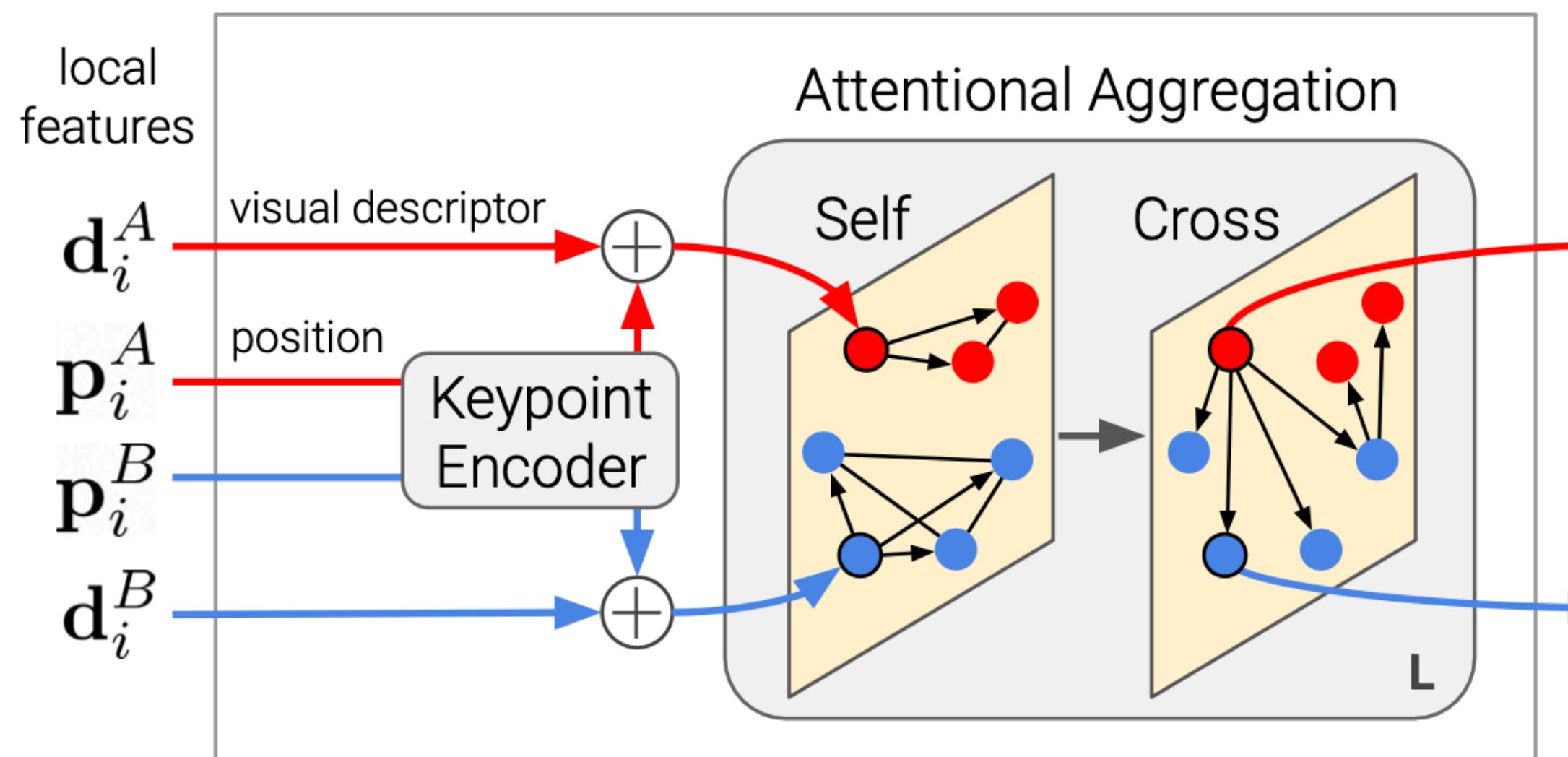
Sarin et al. 2019



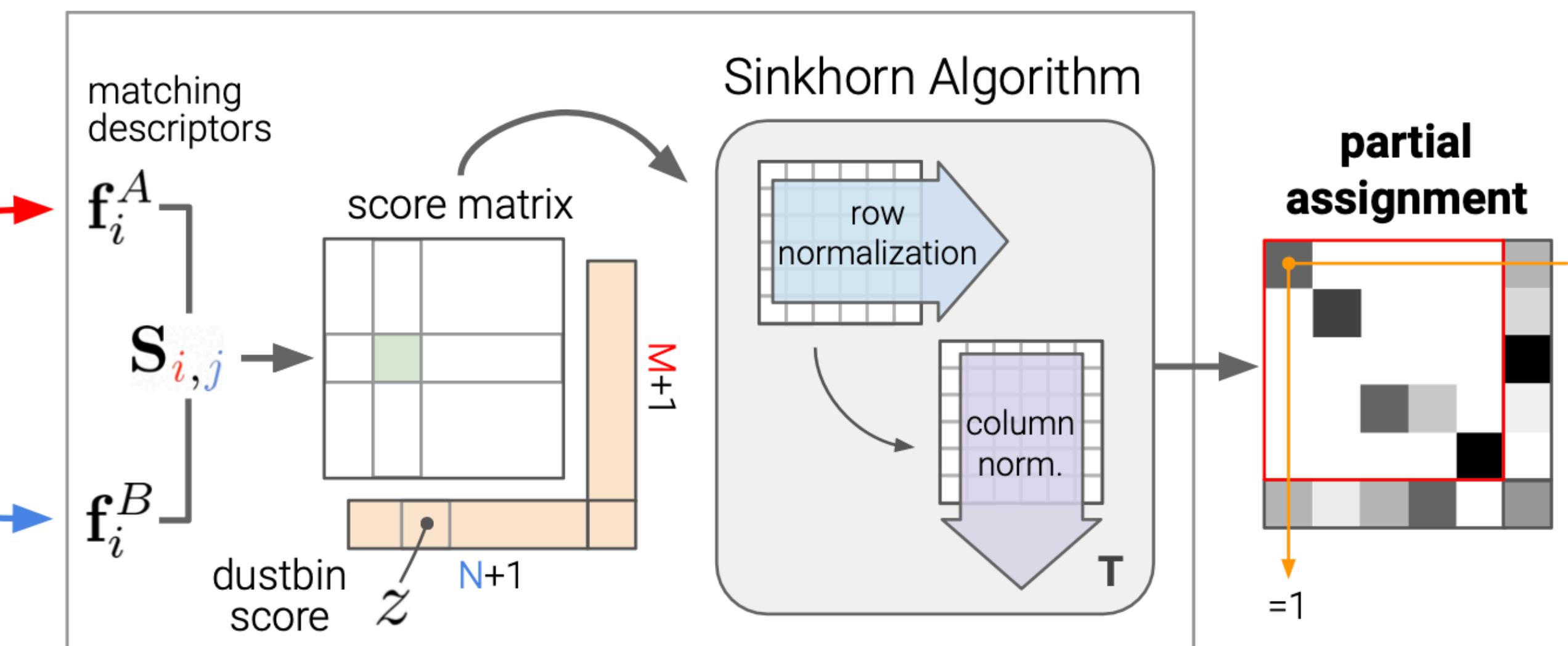
SuperGlue: Learning Feature Matching with Graph Neural Networks

Sarin et al. 2019

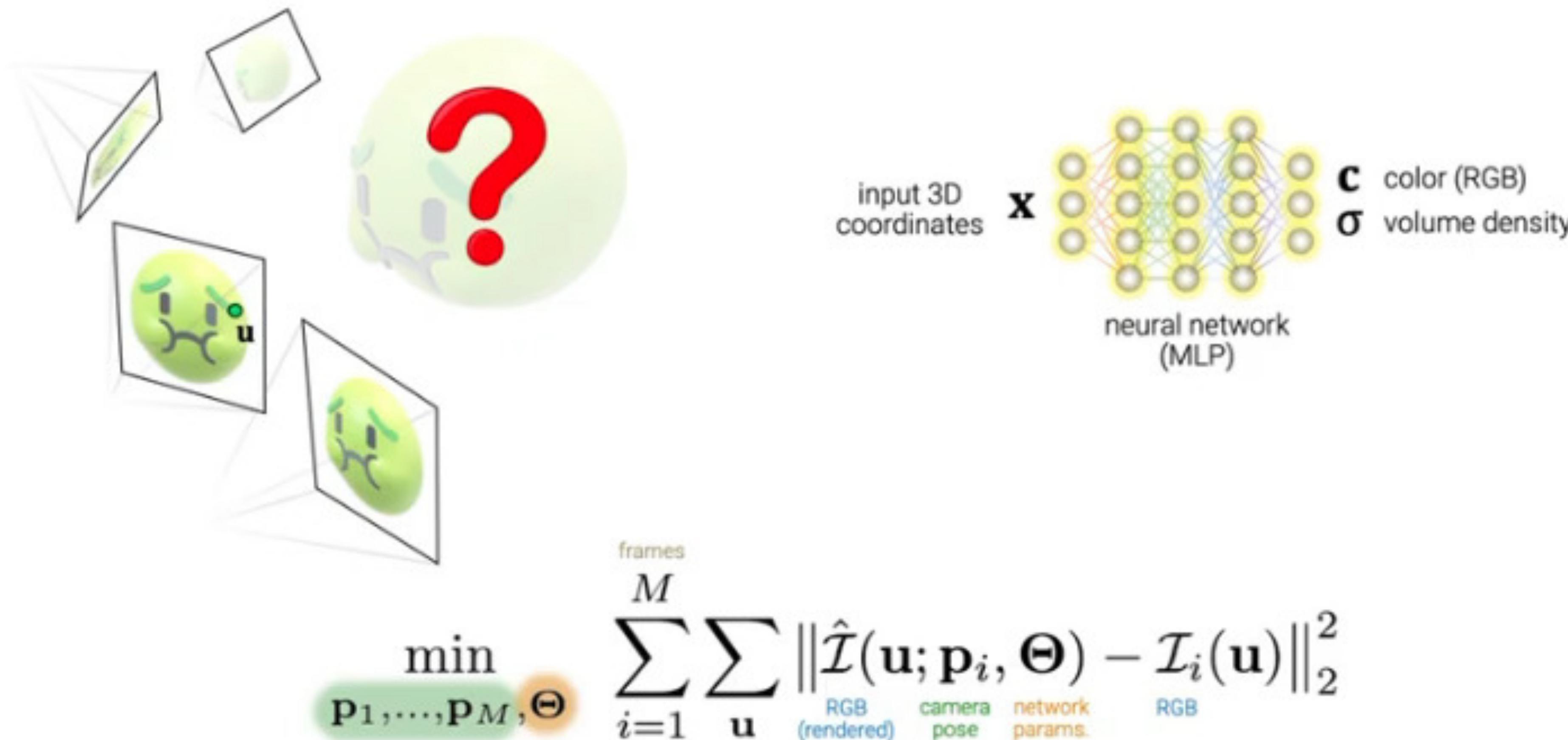
Attentional Graph Neural Network



Optimal Matching Layer



BARF: Bundle-Adjusting Neural Radiance Fields (Lin et al. 2021)



What if the **camera poses** are **imperfect** (or even **unknown**)?
Can we optimize the **poses** naively through backpropagation?

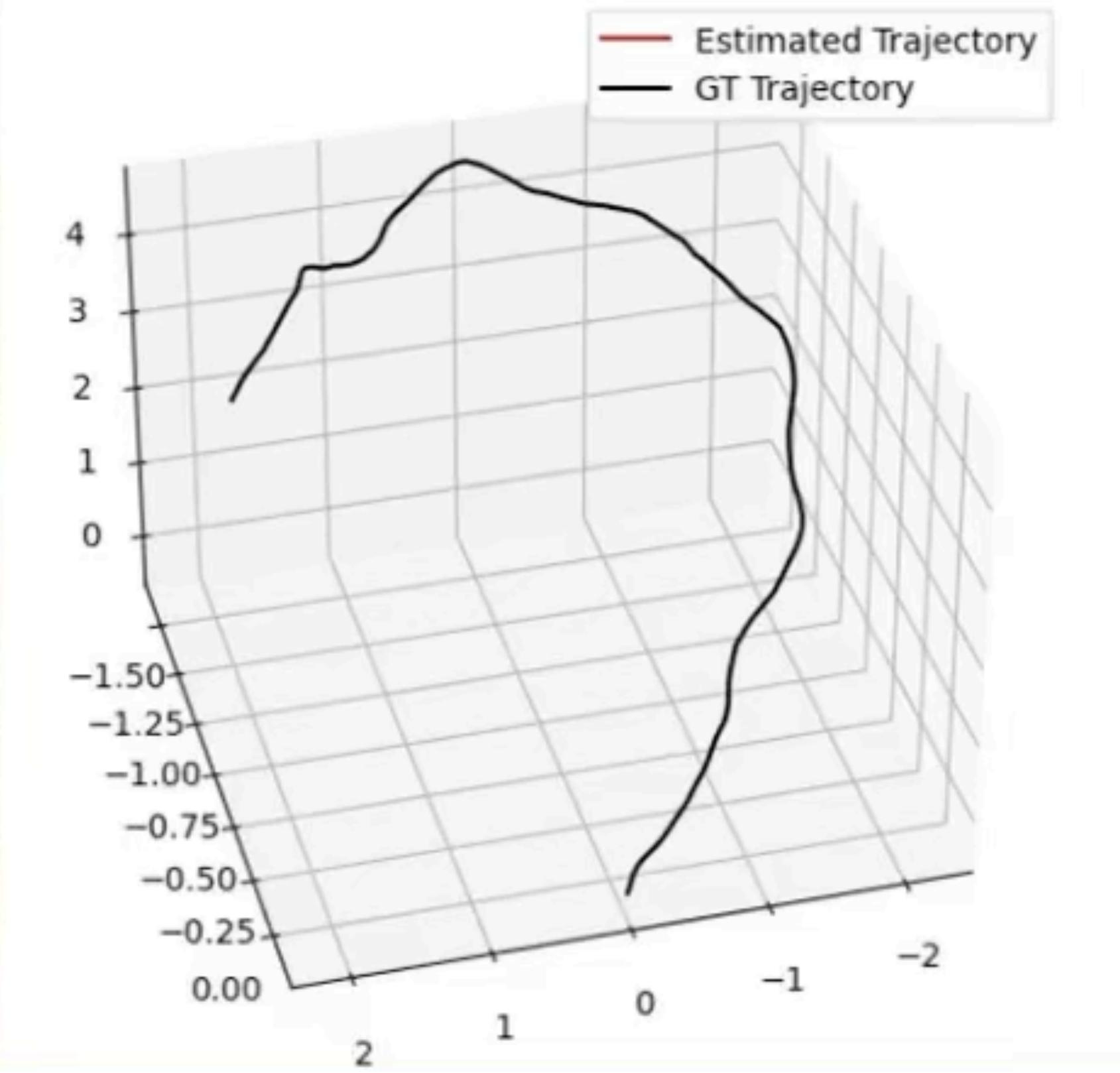
What has changed since Deep Learning?

By and large, we still rely on conventional Bundle Adjustment to solve multi-view geometry for us.

While relatively reliable, this has major downsides:
Not online, not robust to scene motion, not amenable to end-to-end learning...

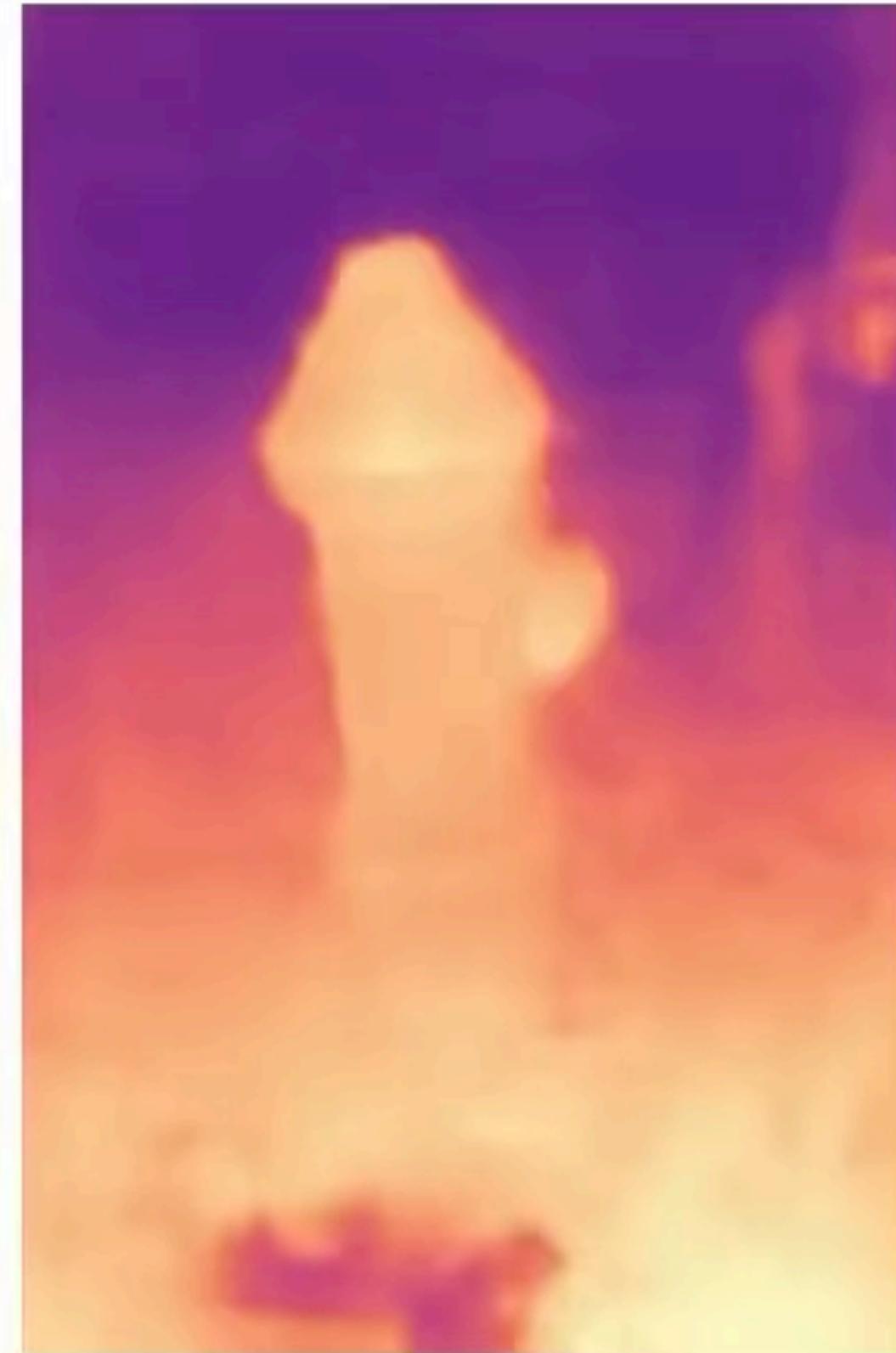
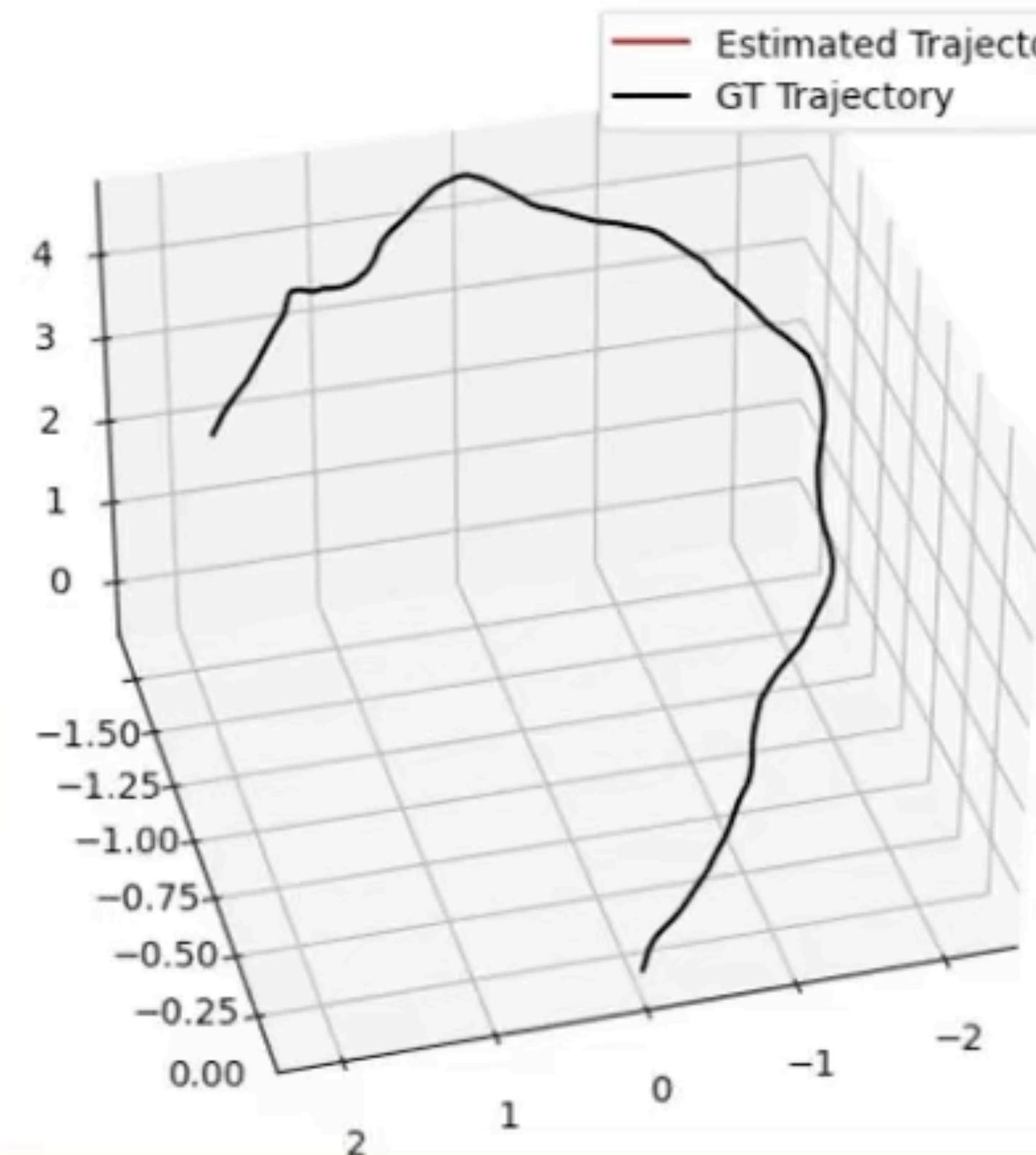
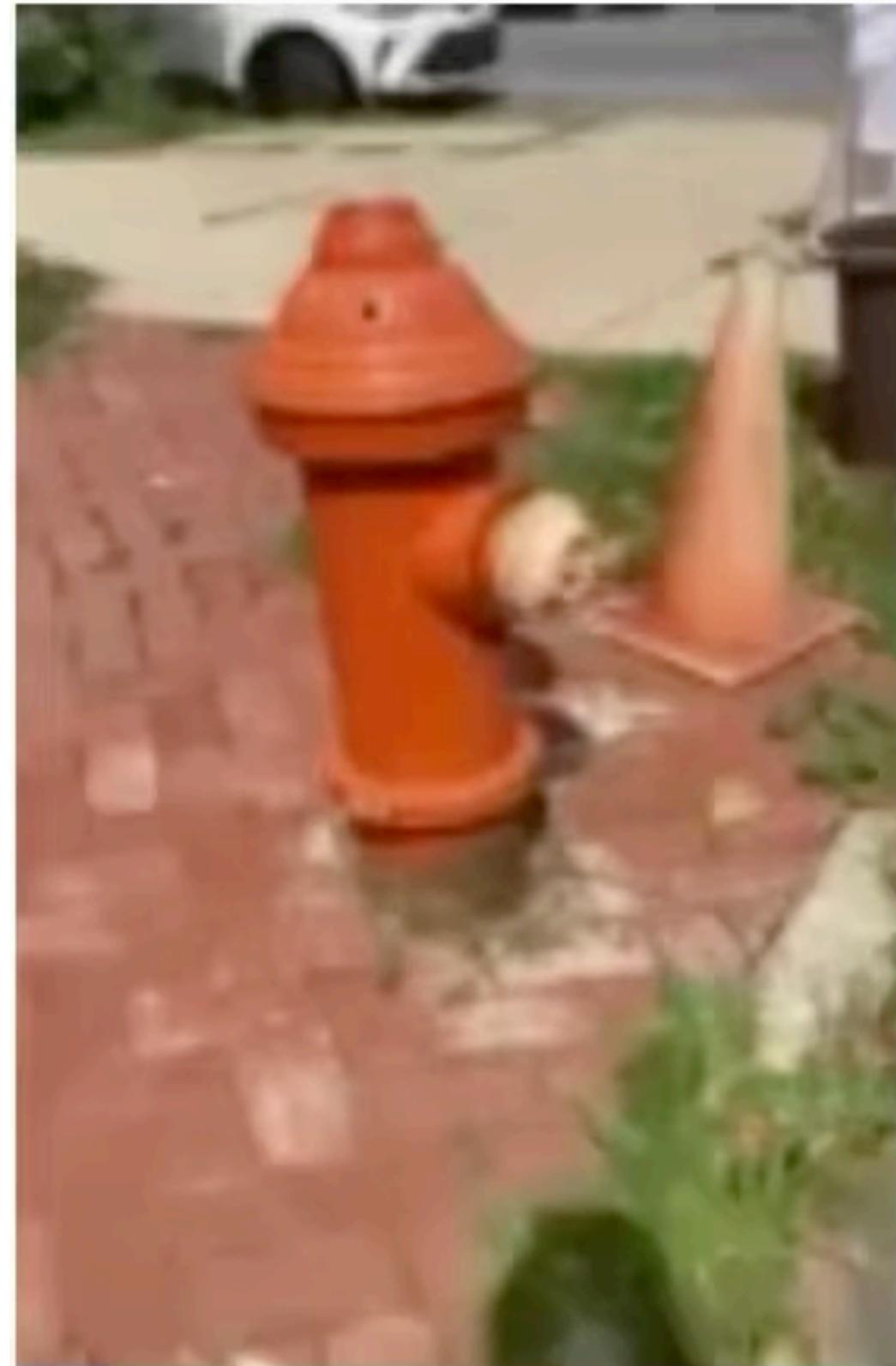
FlowCam:

Generalizable Neural Scene Representations without Camera Poses



Poses inferred feed-forward at ~20FPS. No scene-specific optimization.

FlowCam: Generalizable Neural Scene Representations without Camera Poses



Renderings from Generalizable Neural Scene Representation.
Trained end-to-end, without camera poses, only on raw video!

Summary



Given multi-view observations of *static* scene, we can solve for **camera poses, camera intrinsics, and pretty good 3D geometry**.

The 8-Point Algorithm as an Inductive Bias for Relative Pose Prediction by ViTs

22

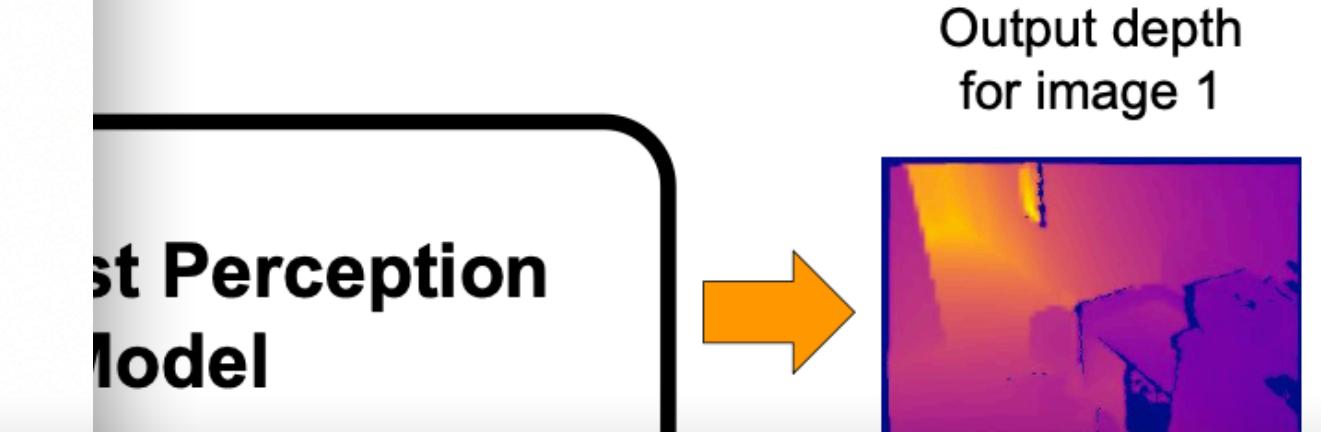
Input-level Inductive Biases for 3D Reconstruction

Wang Yifan^{1,*} Carl Doersch² Relja Arandjelović² João Carreira² Andrew Zisserman^{2,3}

Science, University of Oxford

Generalizable Patch-Based Neural Rendering

Mohammed Suhail¹, Carlos Esteves⁴, Leonid Sigal^{1,2,3}, and Ameesh Makadia⁴



BARF 😱: Bundle-Adjusting Neural Radiance Fields

Chen-Hsuan Lin¹ Wei-Chiu Ma² Antonio Torralba² Simon Lucey^{1,3}

¹Carnegie Mellon University ²Massachusetts Institute of Technology ³The University of Adelaide

<https://chenhsuanlin.bitbucket.io/bundle-adjusting-NeRF>

line 1