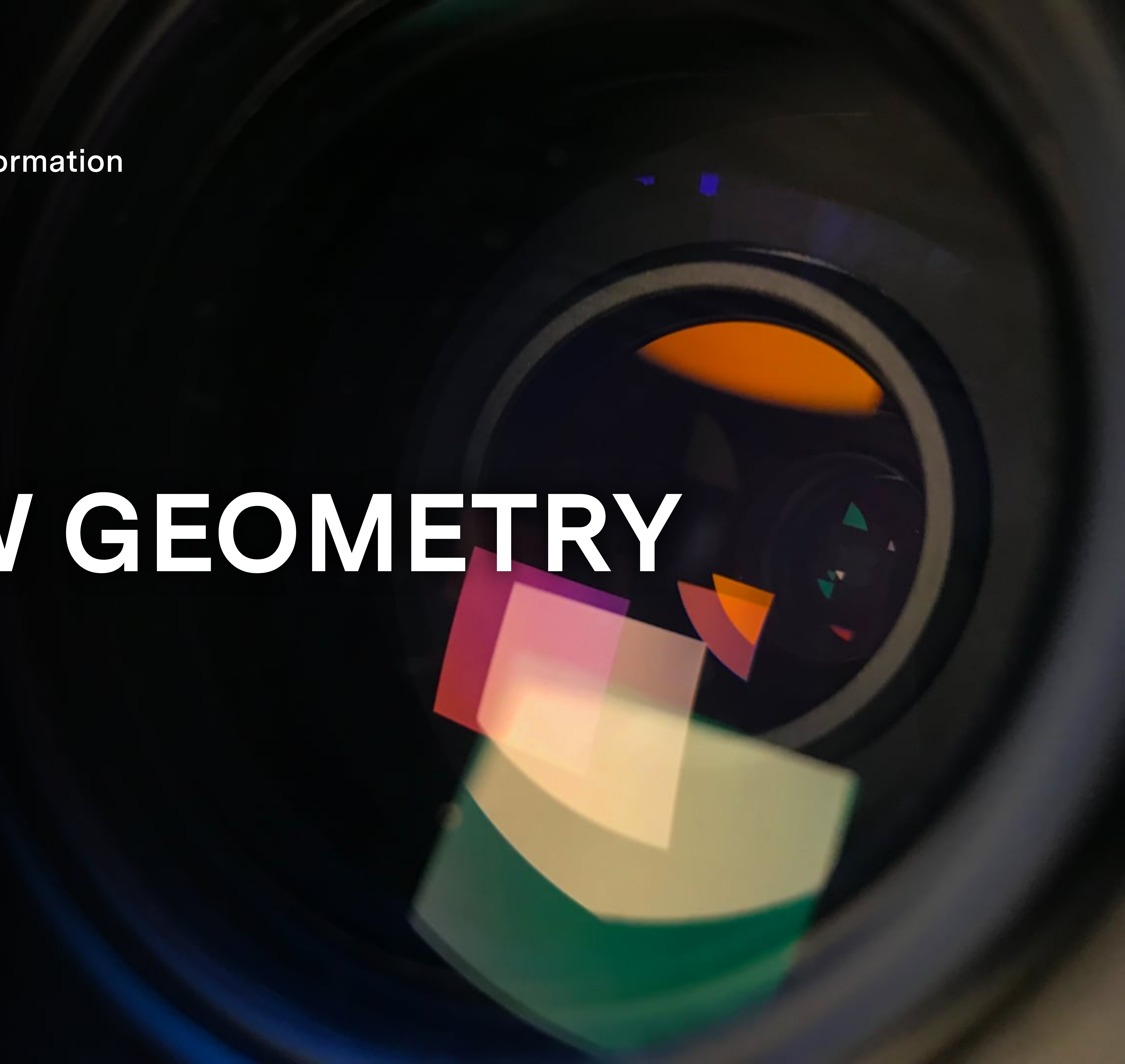
INVERSE GRAPHICS

MODULE 1:
Fundamentals of Image Formation

MULTI-VIEW GEOMETRY



6.S980

Admin Things

- Notifications to for-credit students and listeners are out.
- If you want to drop the class, please do so now, and let us know :)
- First Assignment is out!
- Signup for paper sessions online soon.

Recap: Hom. Coordinates

2D

$\mathbb{R}^2$ $\mathbb{P}^2$

$$\mathbf{x} = (x, y) \rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \tilde{\mathbf{x}}$$

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \rightarrow \left(x/w, y/w\right)$$

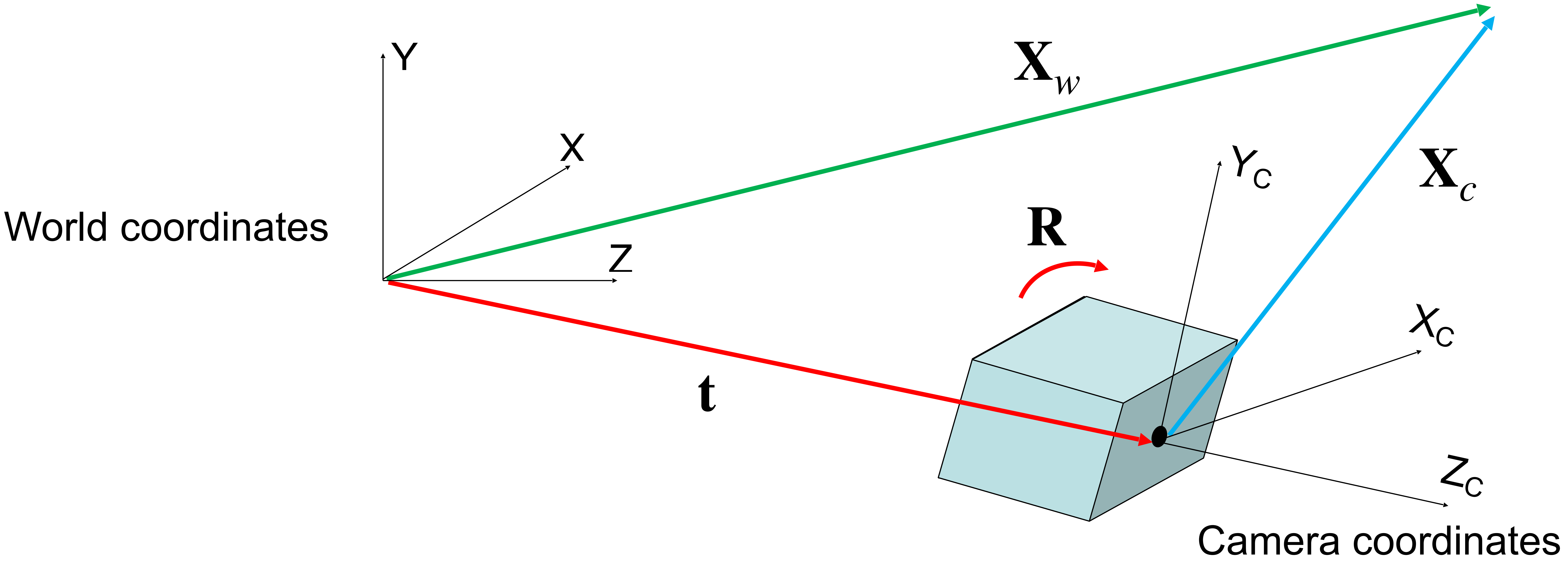
3D

$\mathbb{R}^3$ $\mathbb{P}^3$

$$\mathbf{X} = (X, Y, Z) \rightarrow \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \tilde{\mathbf{X}}$$

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

Recap: Camera parameters



World coordinates to camera coordinates

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

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

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

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

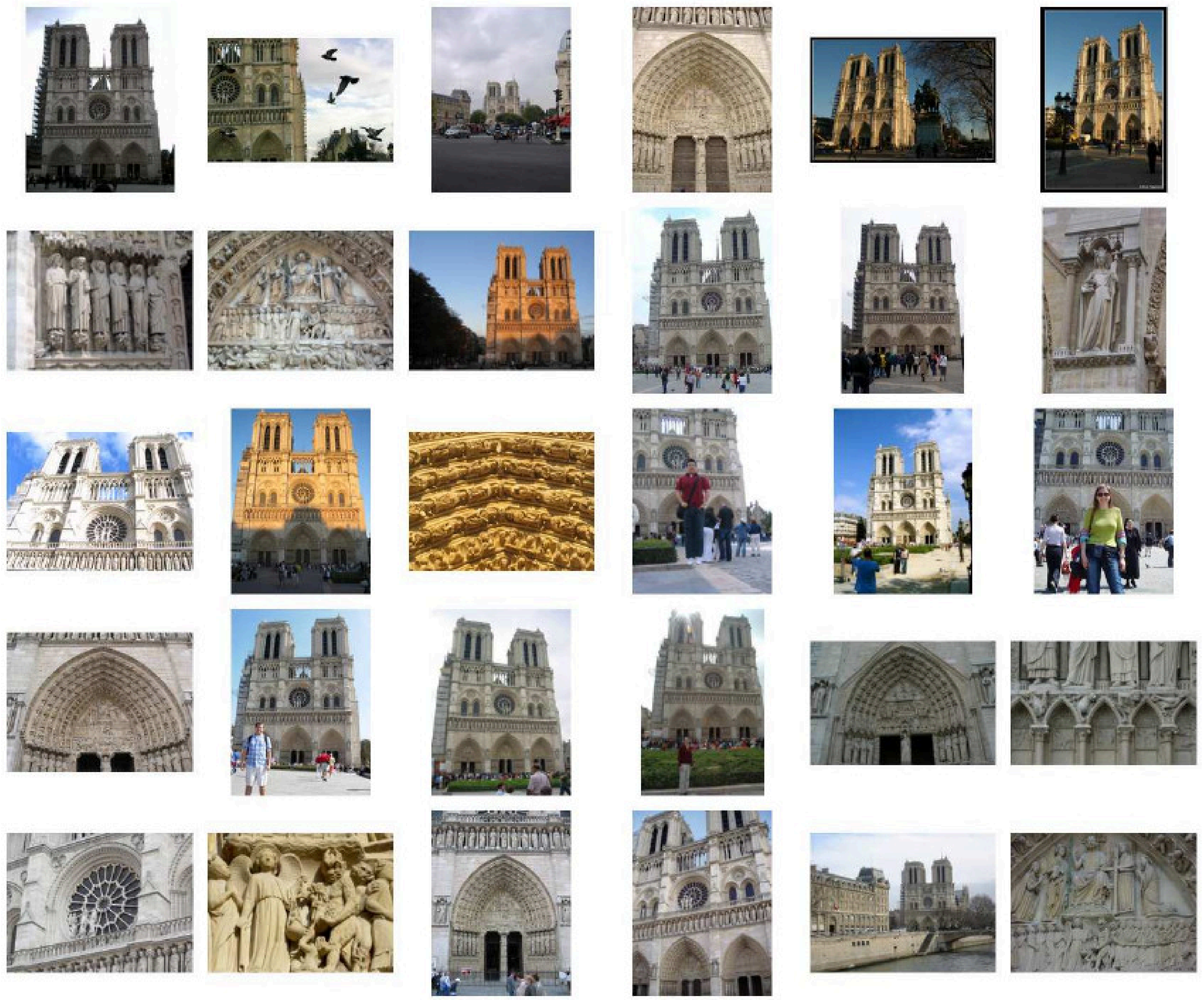
Camera coordinates to image coordinates

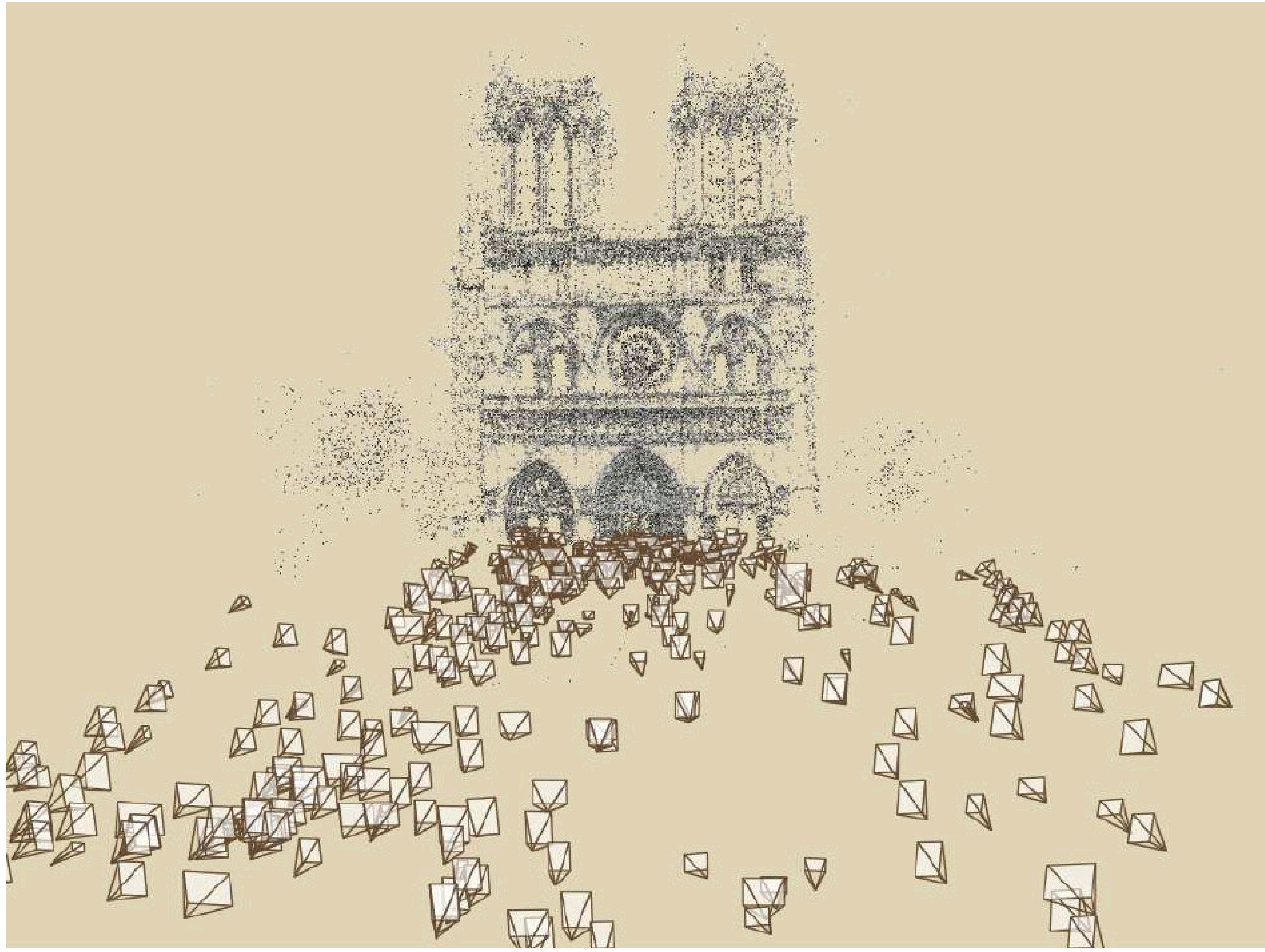
$$\tilde{\mathbf{x}} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tilde{\mathbf{X}}_c$$

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

$$\boxed{\begin{gathered} \tilde{\mathbf{x}} = \mathbf{K}[\mathbf{I} \,|\, 0] \mathbf{C}^{W2C} \tilde{\mathbf{X}}_w \\ \tilde{\mathbf{x}} = \mathbf{P} \tilde{\mathbf{X}}_w \end{gathered}}$$

Now: Multi-View Geometry

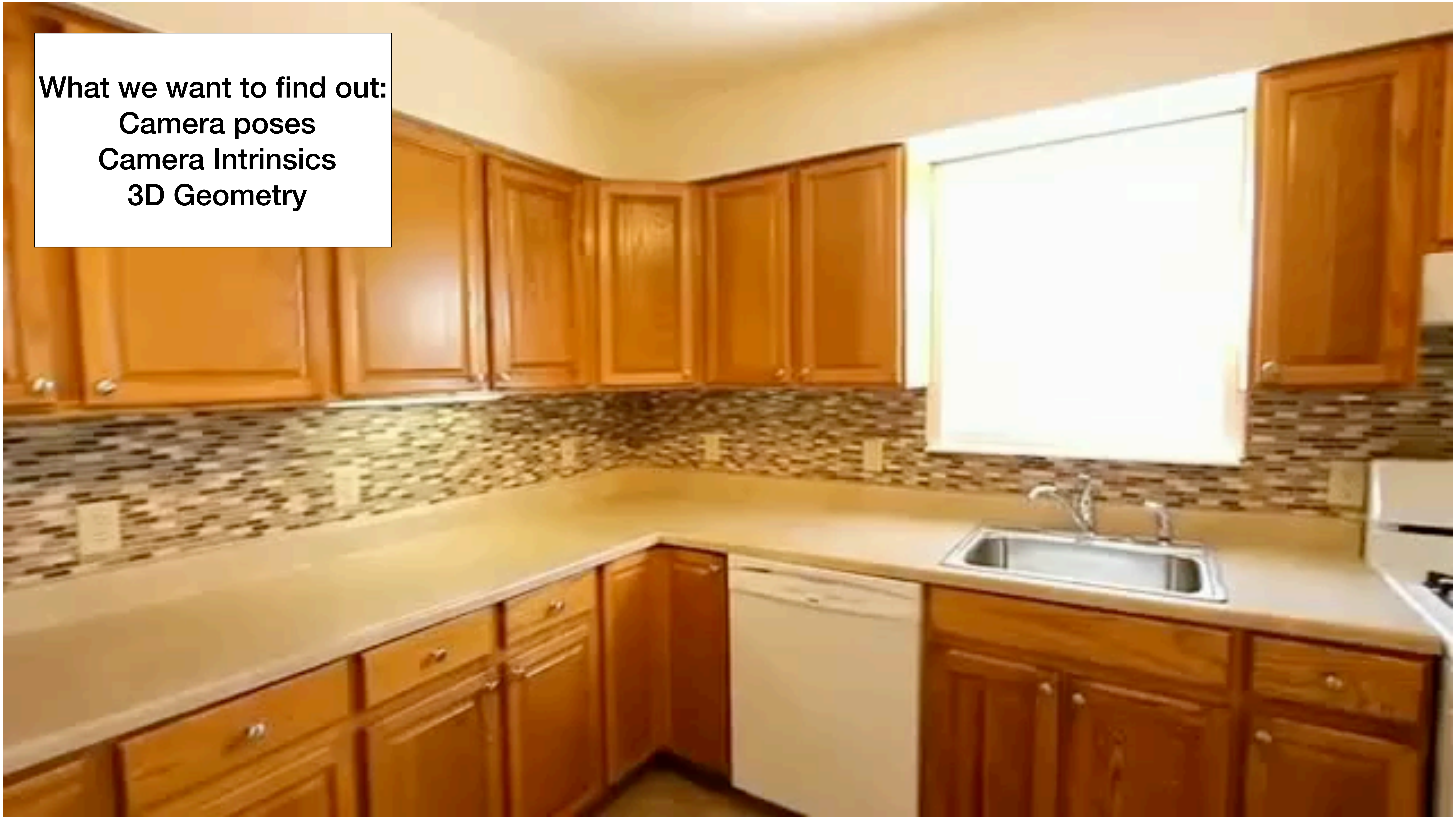




Why?	We want to understand 3D world only from 2D observations (images). For that, we need to have a mathematical understanding of how they are connected.
What you'll learn.	Mathematical model of cameras. Reconstruct camera poses, approximate geometry, and camera parameters from 2D images of a scene.

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?

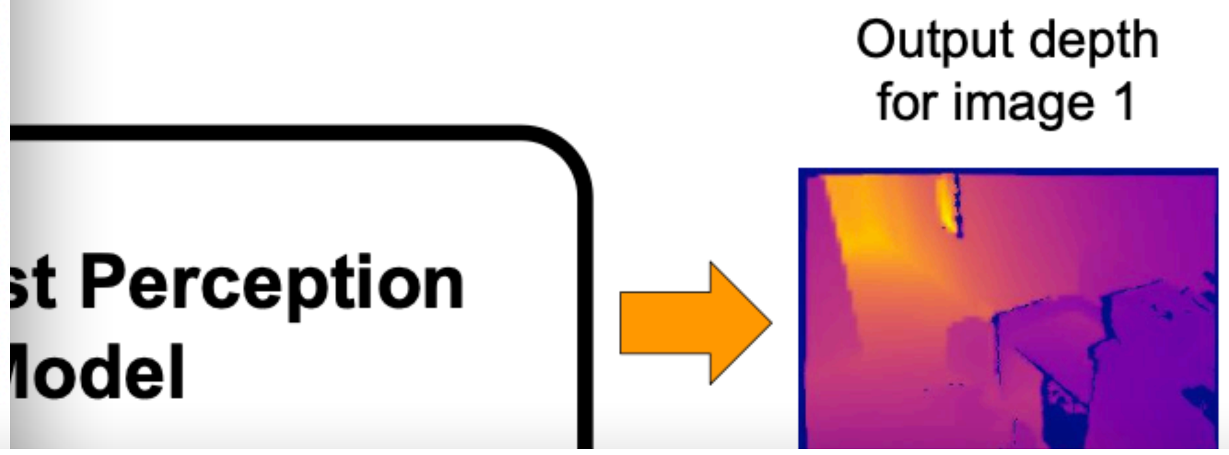
The 8-Point Algorithm as an Inductive Bias for Relative Pose Prediction by ViTs

22

Input-level Inductive Biases for 3D Reconstruction

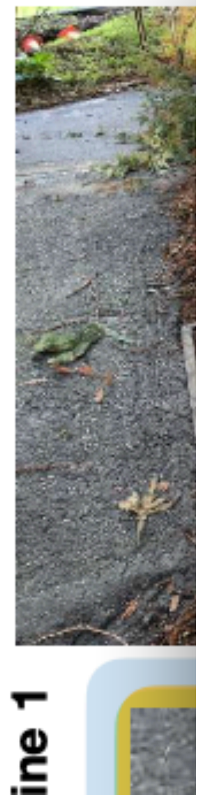
Wang Yifan[1*] Carl Doersch[2] Relja Arandjelović[2] João Carreira[2] Andrew Zisserman[2,3]

Science, University of Oxford



Generalizable Patch-Based Neural Rendering

Mohammed Suhail[1], Carlos Esteves[4], Leonid Sigal[1,2,3], and Ameesh Makadia[4]



BARF 🤮: Bundle-Adjusting Neural Radiance Fields

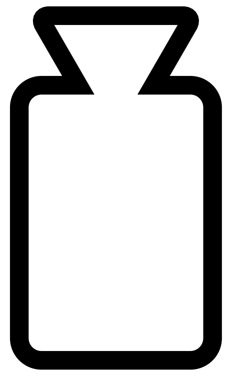
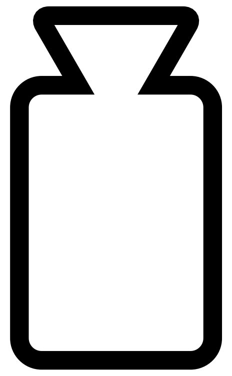
Chen-Hsuan Lin[1] Wei-Chiu Ma[2] Antonio Torralba[2] Simon Lucey[1,3]

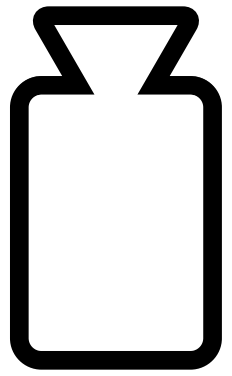
[1]Carnegie Mellon University [2]Massachusetts Institute of Technology [3]The University of Adelaide

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

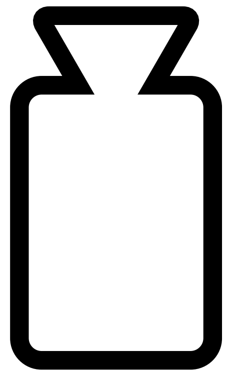
Antonio’s old office!

 Known $\mathbf{P}_1, \mathbf{P}_2$! 



Known $\mathbf{P}_1, \mathbf{P}_2$!

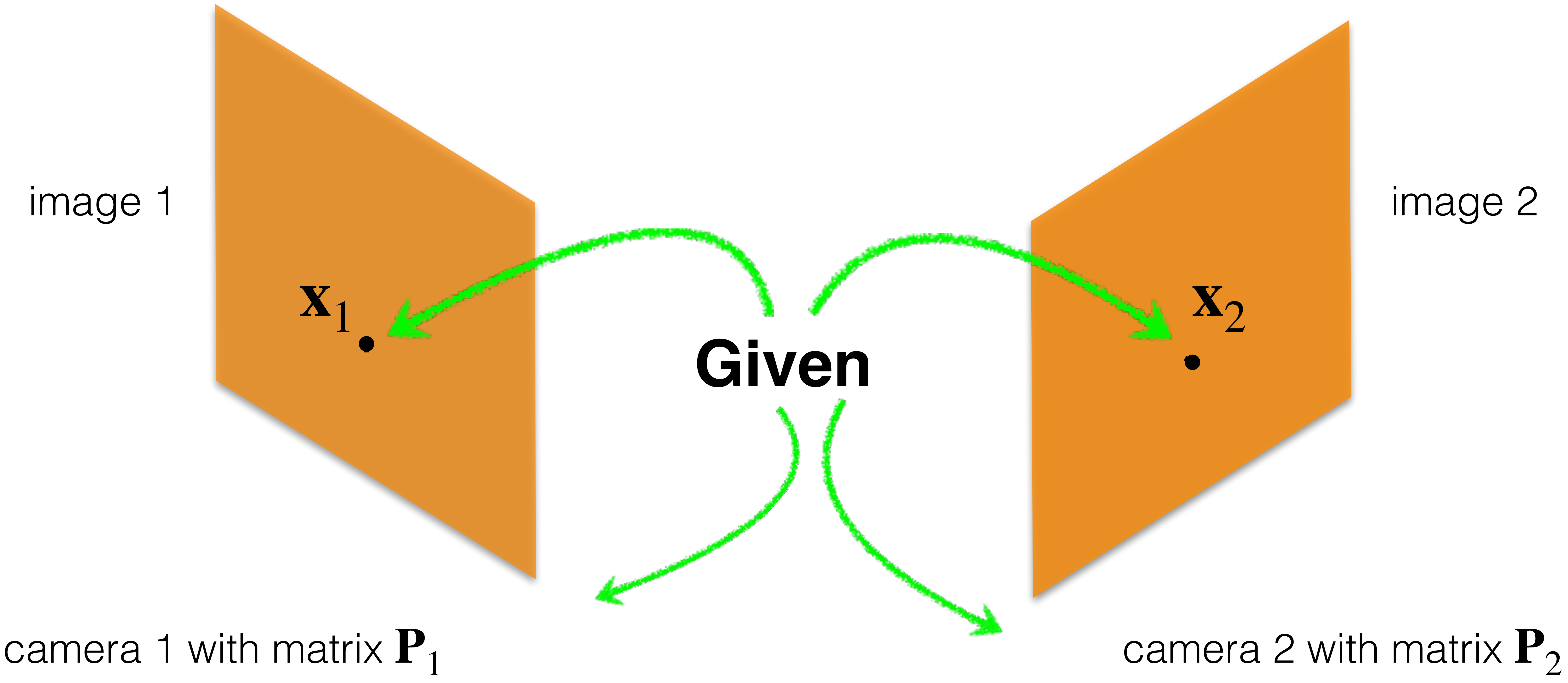


Bundle Adjustment

Triangulation	Epipolar Lines	Fundamental & Essential Matrices	No Time
How to compute 3D locations of point correspondences if cameras are known.	Which pixels in two cameras observe same 3D point? Where to look for multi-view correspondences?	Elegant formulation of Epipolar Lines A way of *estimating camera poses, intrinsics, and extrinsic from correspondences.*	• • •

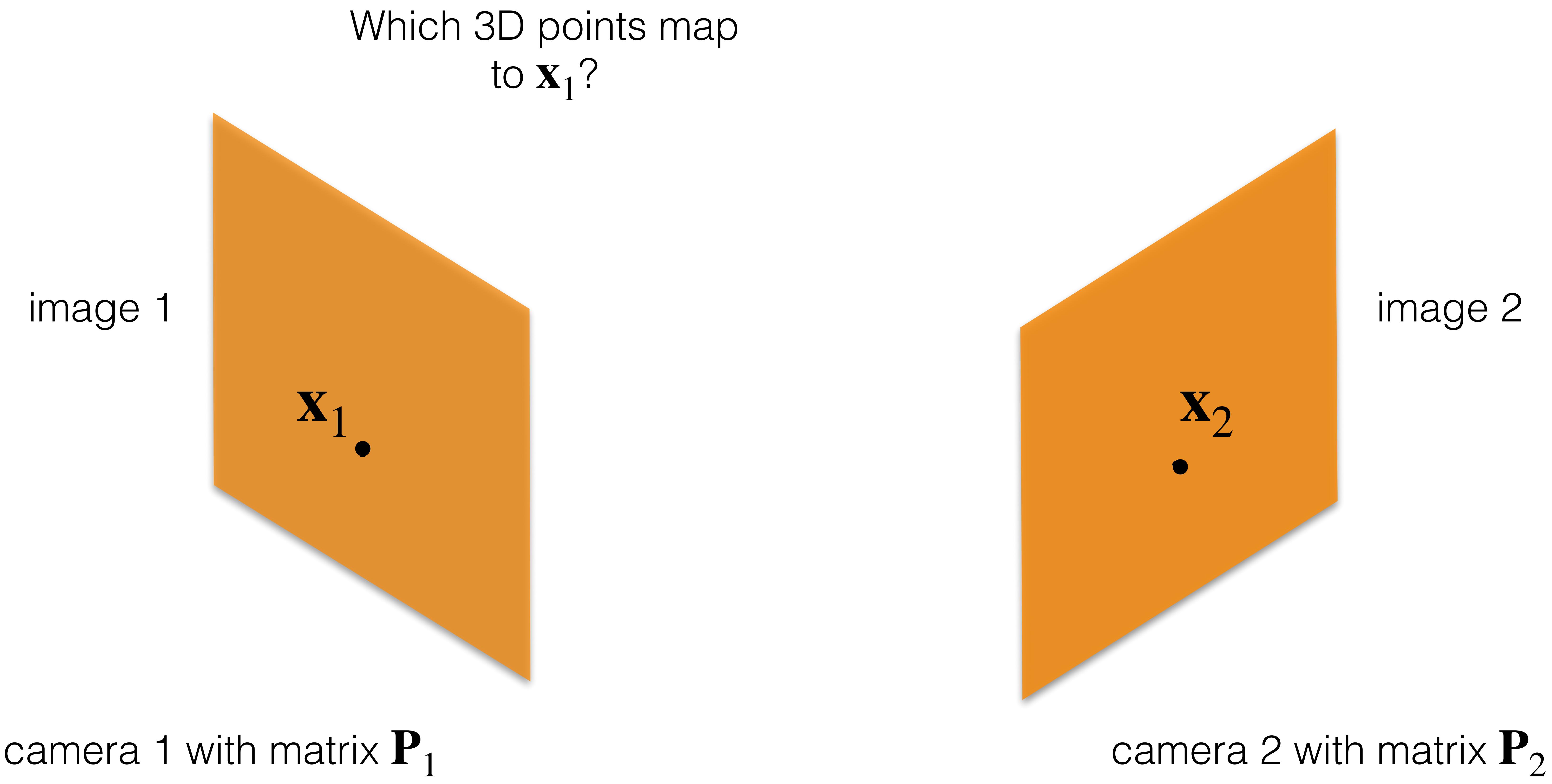
What has changed since Deep Learning?

Triangulation



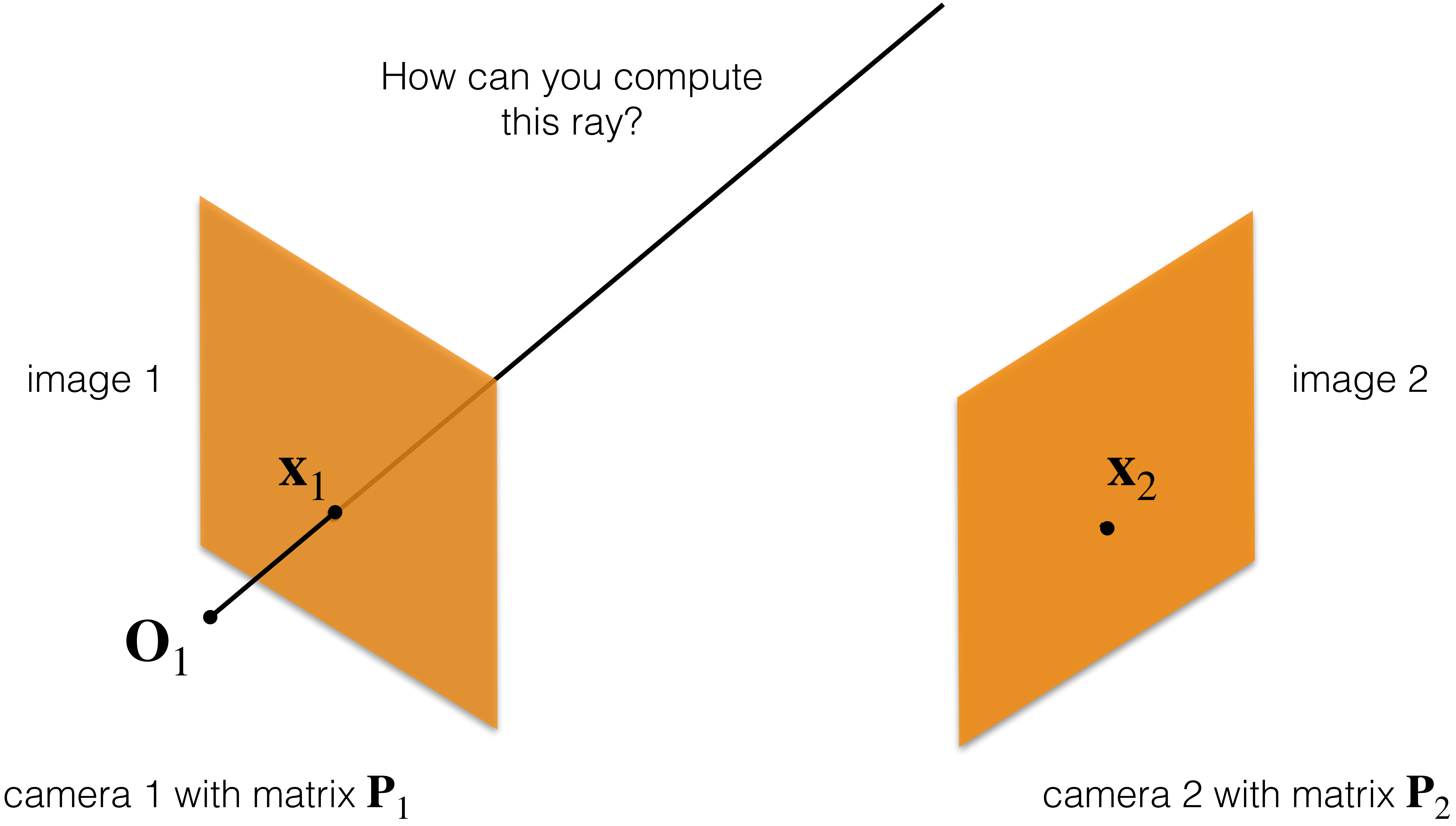
Slide credit: CMU 16-385 (Yannis, Kris)

Triangulation



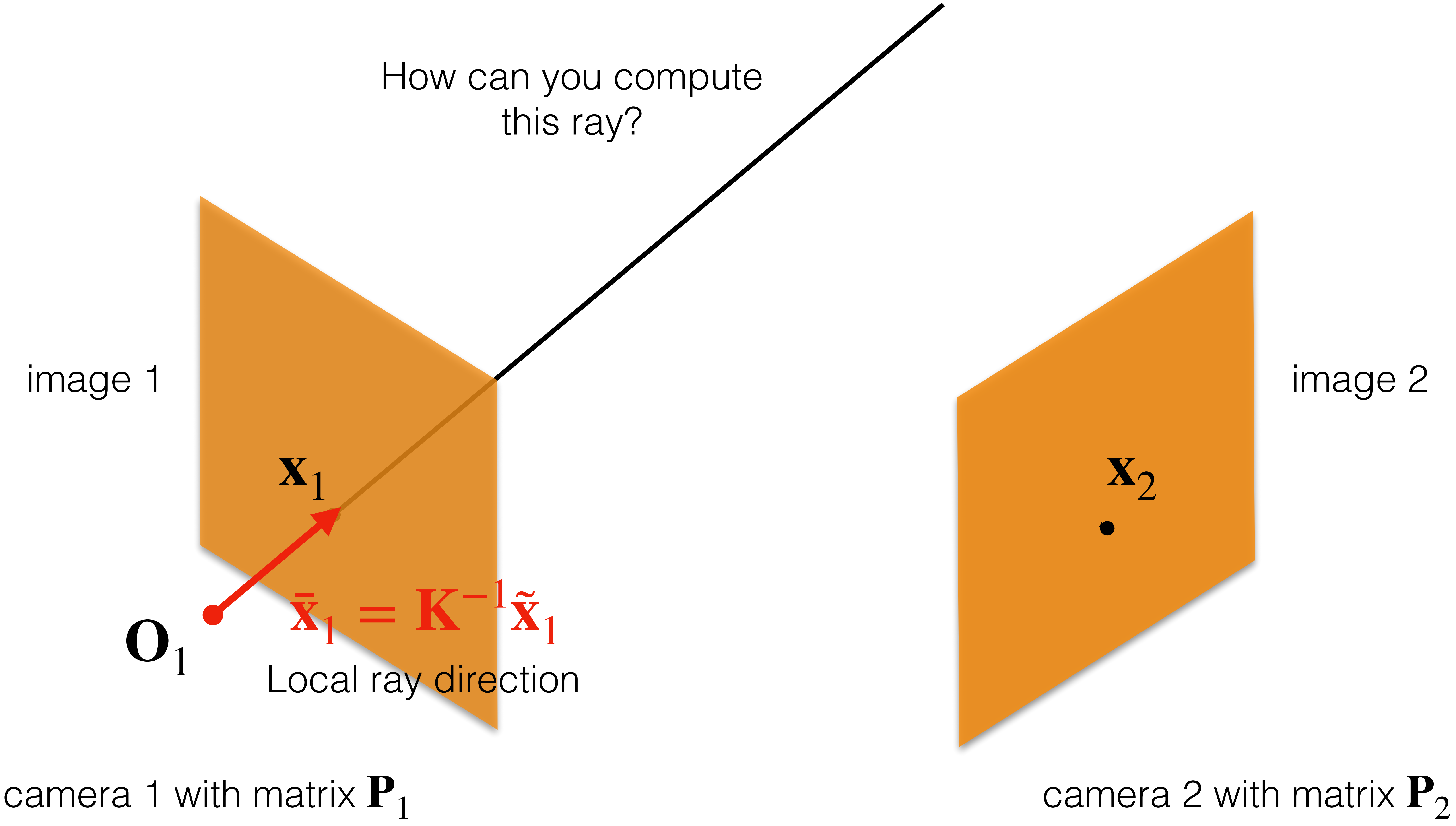
Slide credit: CMU 16-385 (Yannis, Kris)

Triangulation



Slide credit: CMU 16-385 (Yannis, Kris)

Triangulation

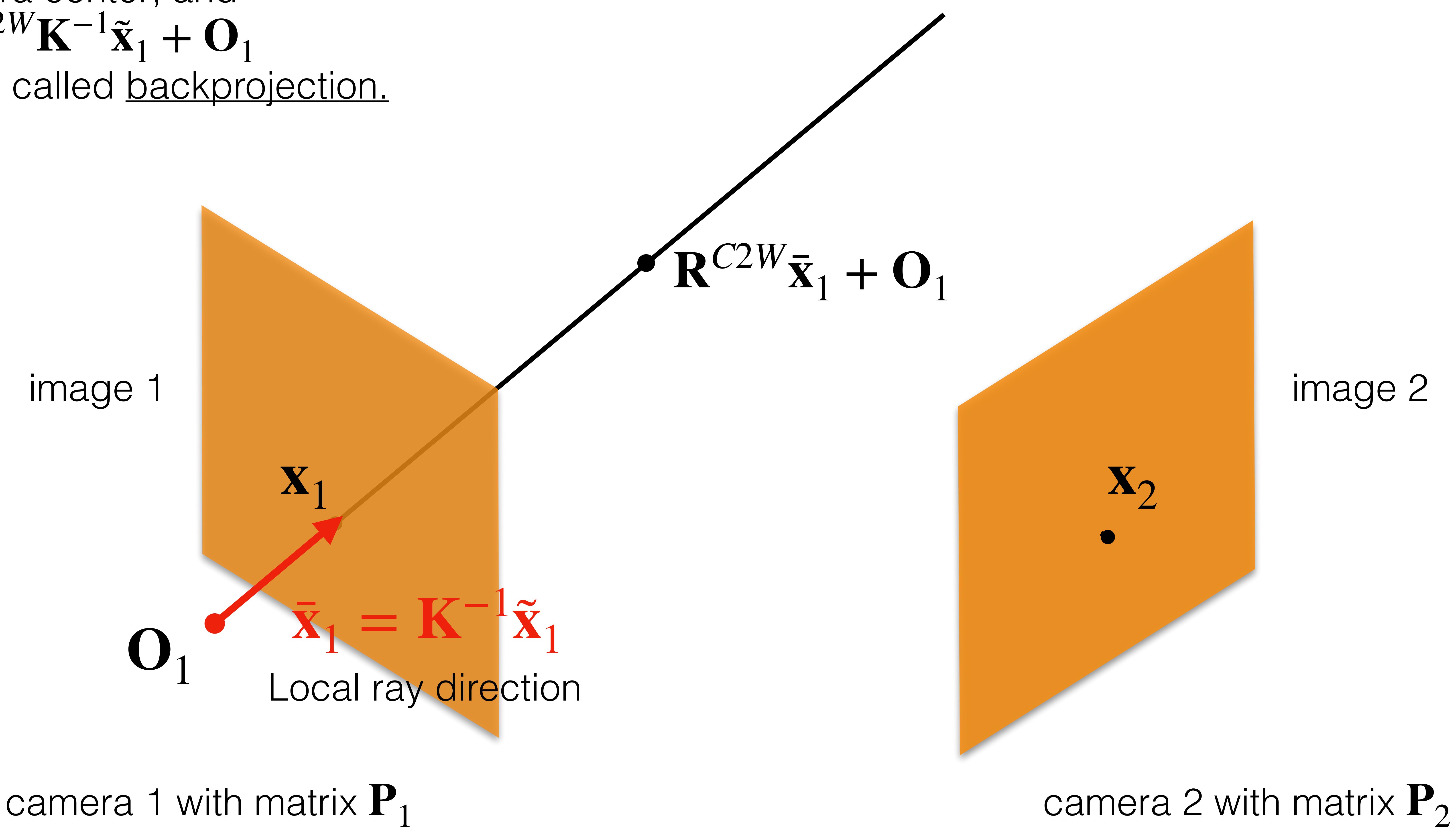


Triangulation

Create two points on the ray:

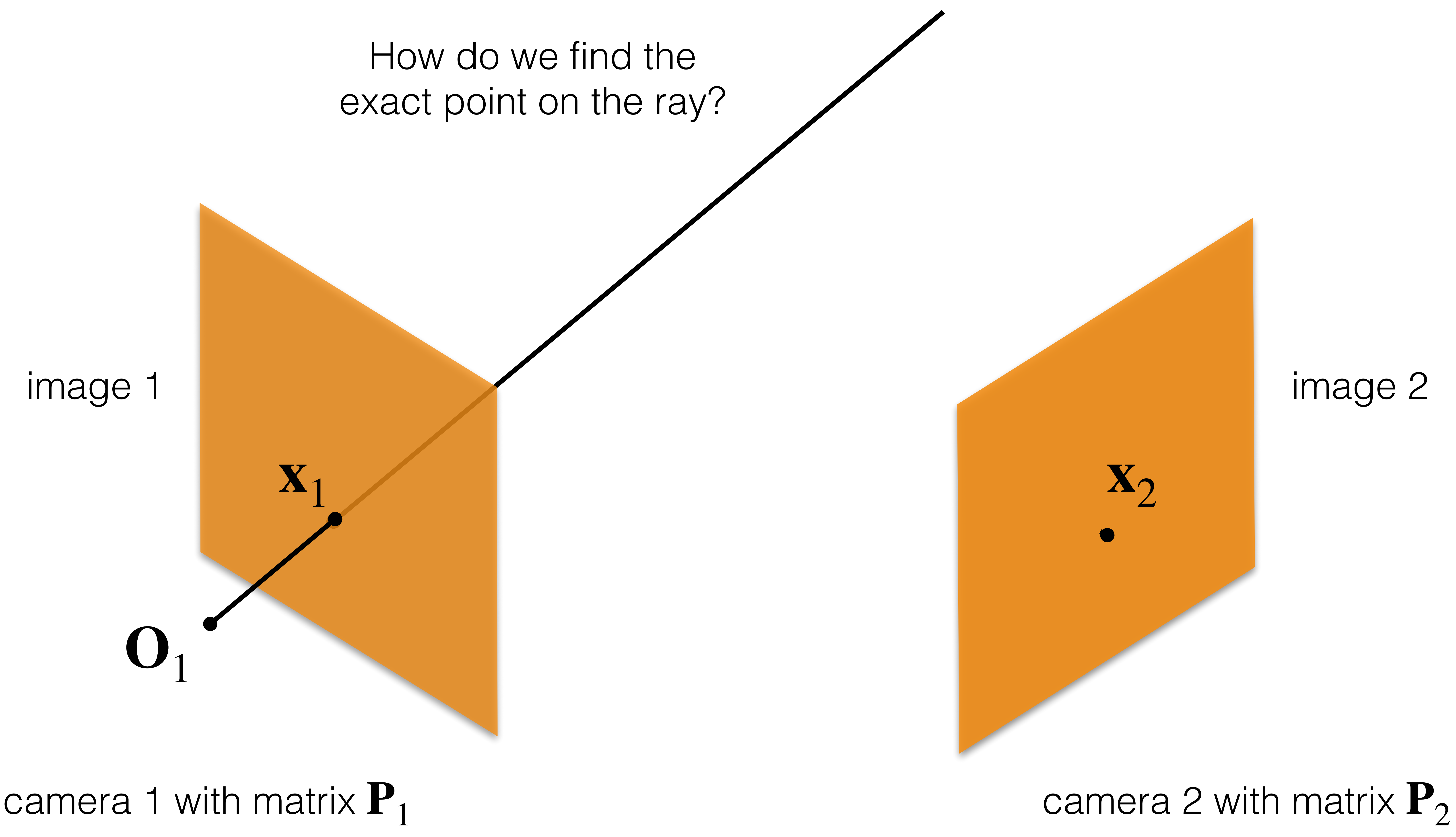
1) find the camera center; and
2) Compute $\mathbf{R}^{C2W}\mathbf{K}^{-1}\tilde{\mathbf{x}}_1 + \mathbf{O}_1$

This procedure is called <u>backprojection.</u>



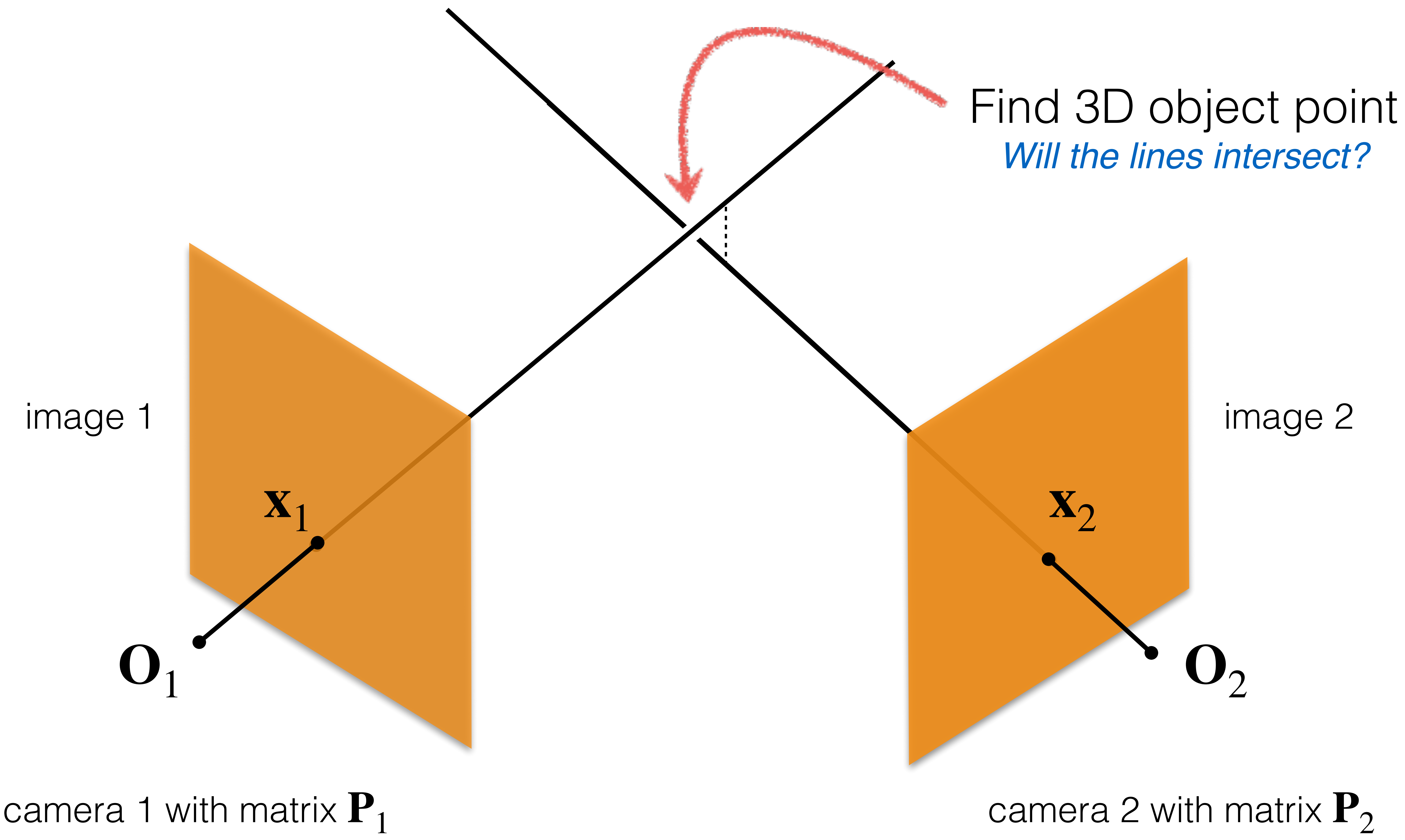
Slide credit: CMU 16-385 (Yannis, Kris)

Triangulation



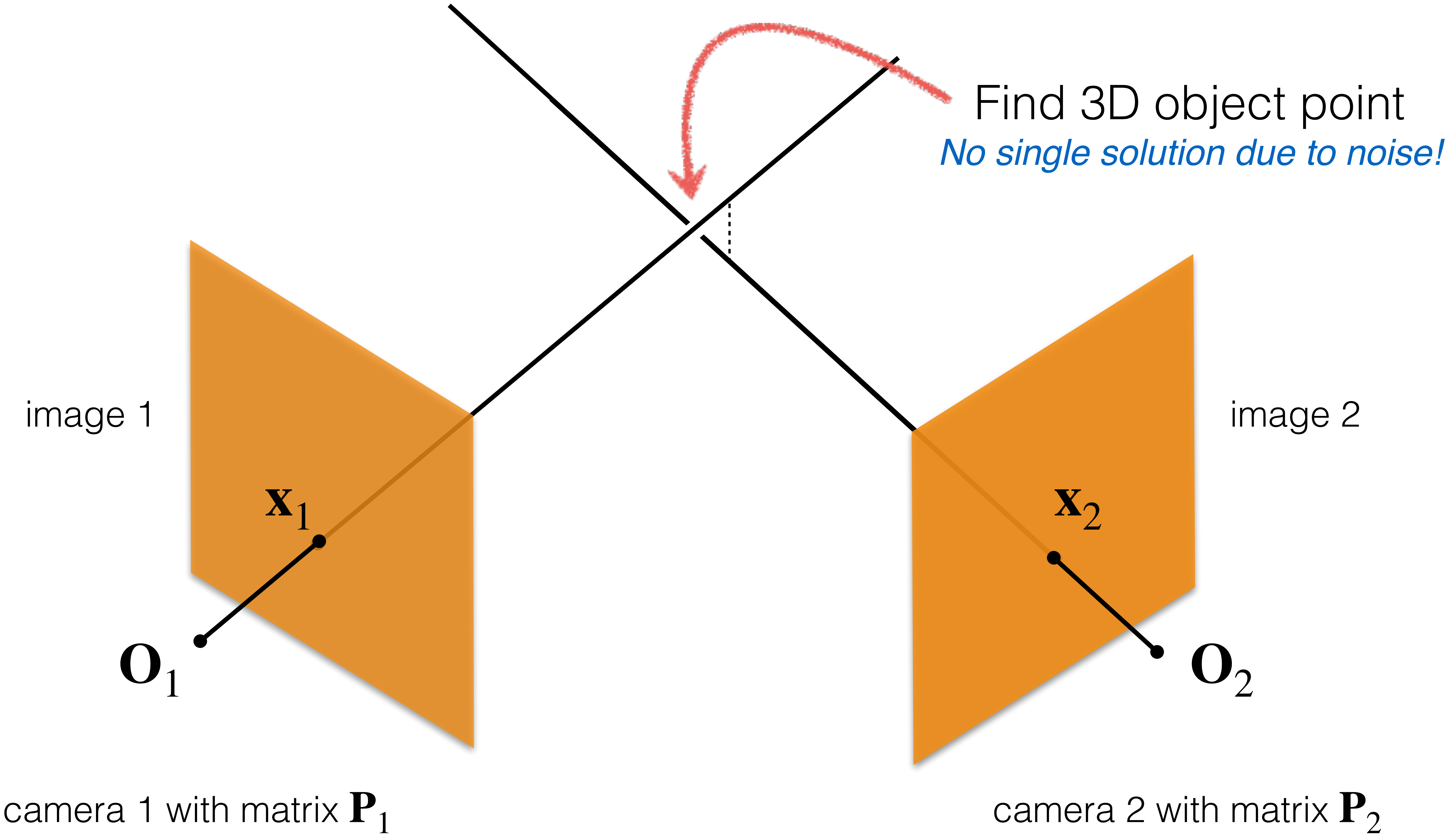
Slide credit: CMU 16-385 (Yannis, Kris)

Triangulation



Slide credit: CMU 16-385 (Yannis, Kris)

Triangulation



Slide credit: CMU 16-385 (Yannis, Kris)

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

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

Estimate the 3D point

$$\mathbf{X}$$

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

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

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:

$$\mathbf{X}^* = argmin_{\mathbf{X}} \sum_{i}^{N} \|\pi_i(\mathbf{X}) - \mathbf{x}_i\|_2^2$$

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

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

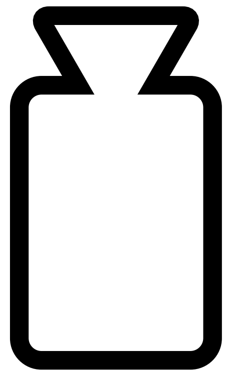
Estimate the 3D point

$$\mathbf{X}$$

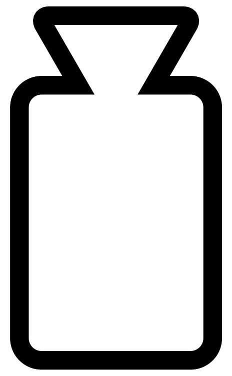
Then we can solve a little least squares problem:

$$\mathbf{X}^* = argmin_{\mathbf{X}} \sum_{i}^{N} \|\pi_i(\mathbf{X}) - \mathbf{x}_i\|_2^2$$

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



Known $\mathbf{P}_1, \mathbf{P}_2$!



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

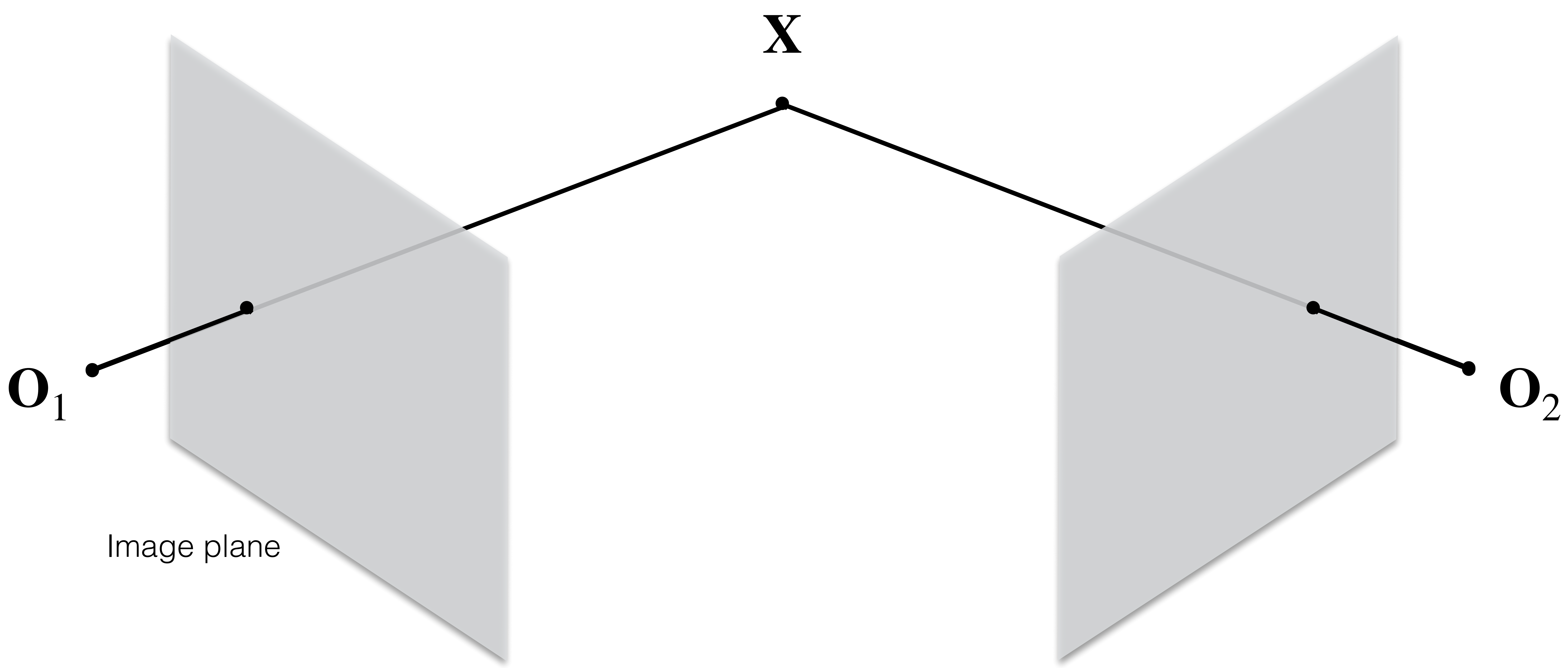
•

•

•

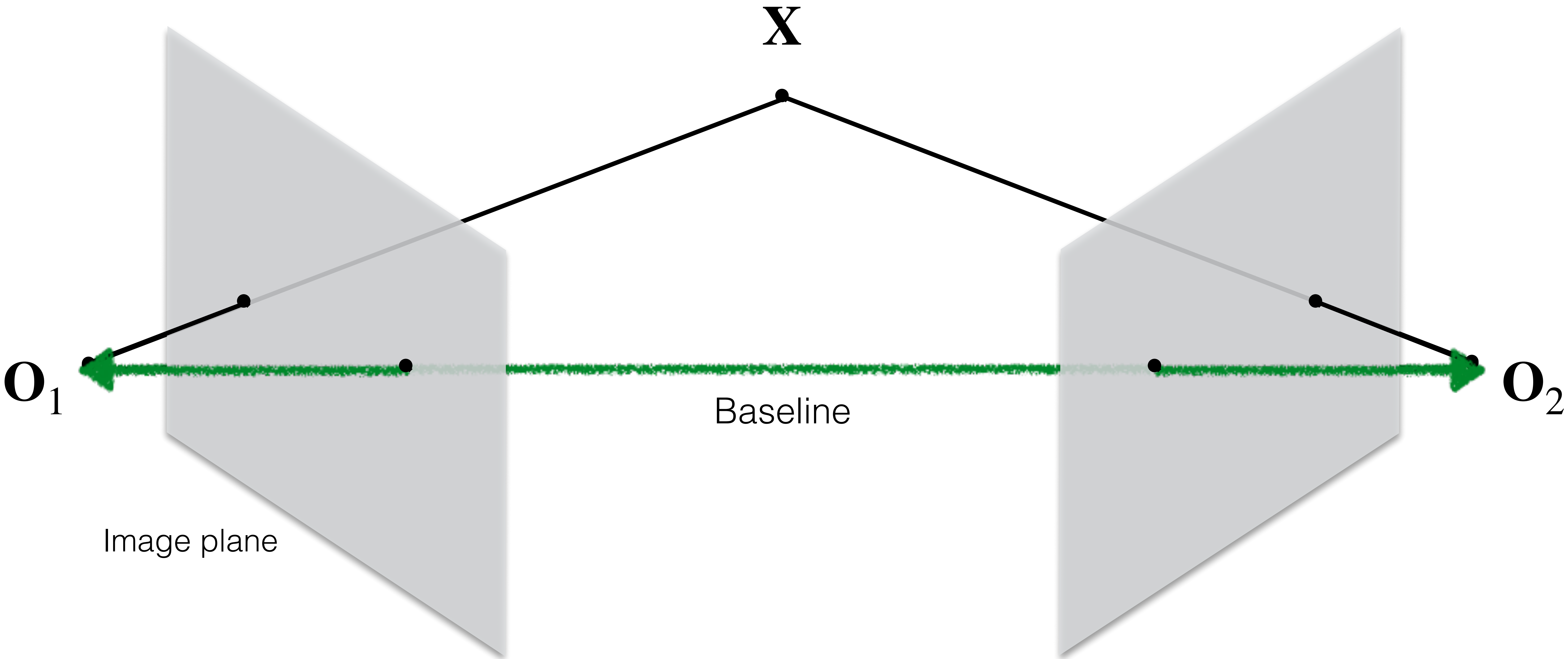
What has changed since Deep Learning?

Epipolar geometry



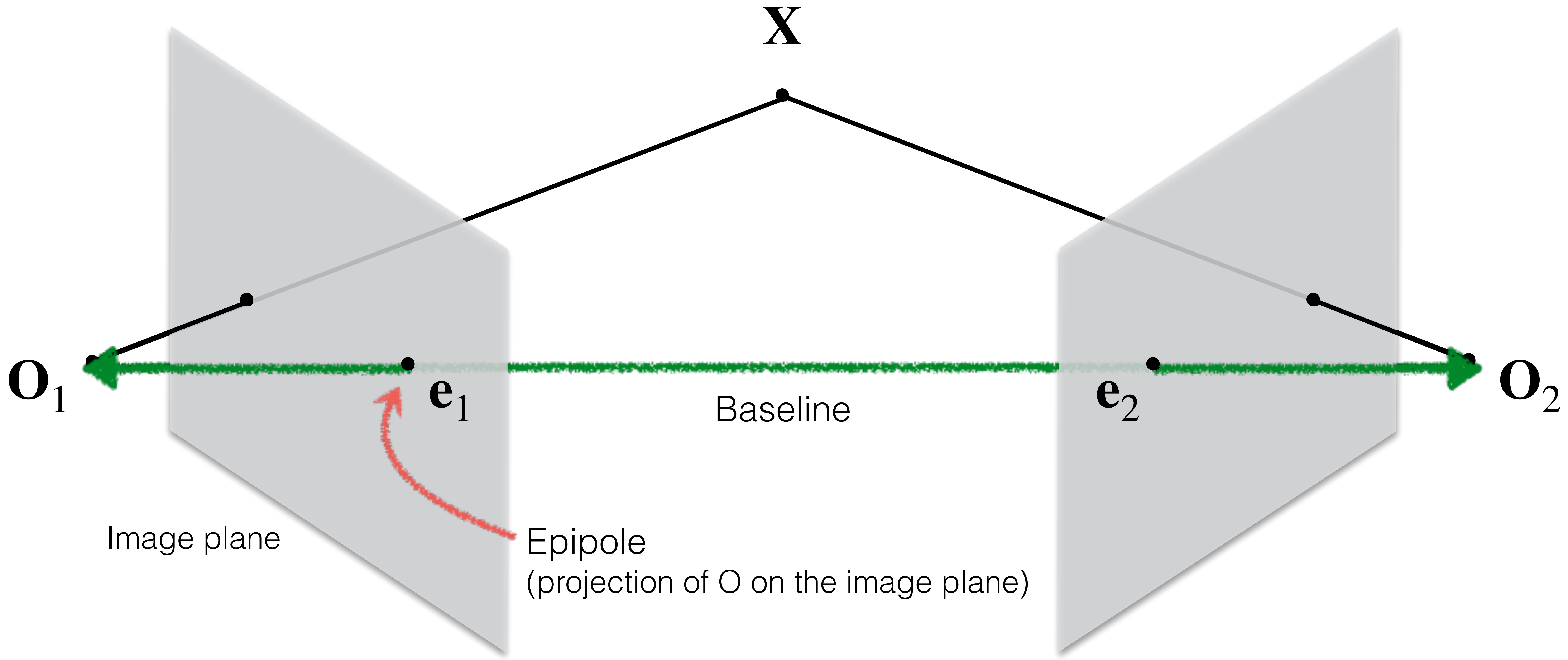
Adapted from: CMU 16-385 (Yannis, Kris)

Epipolar geometry



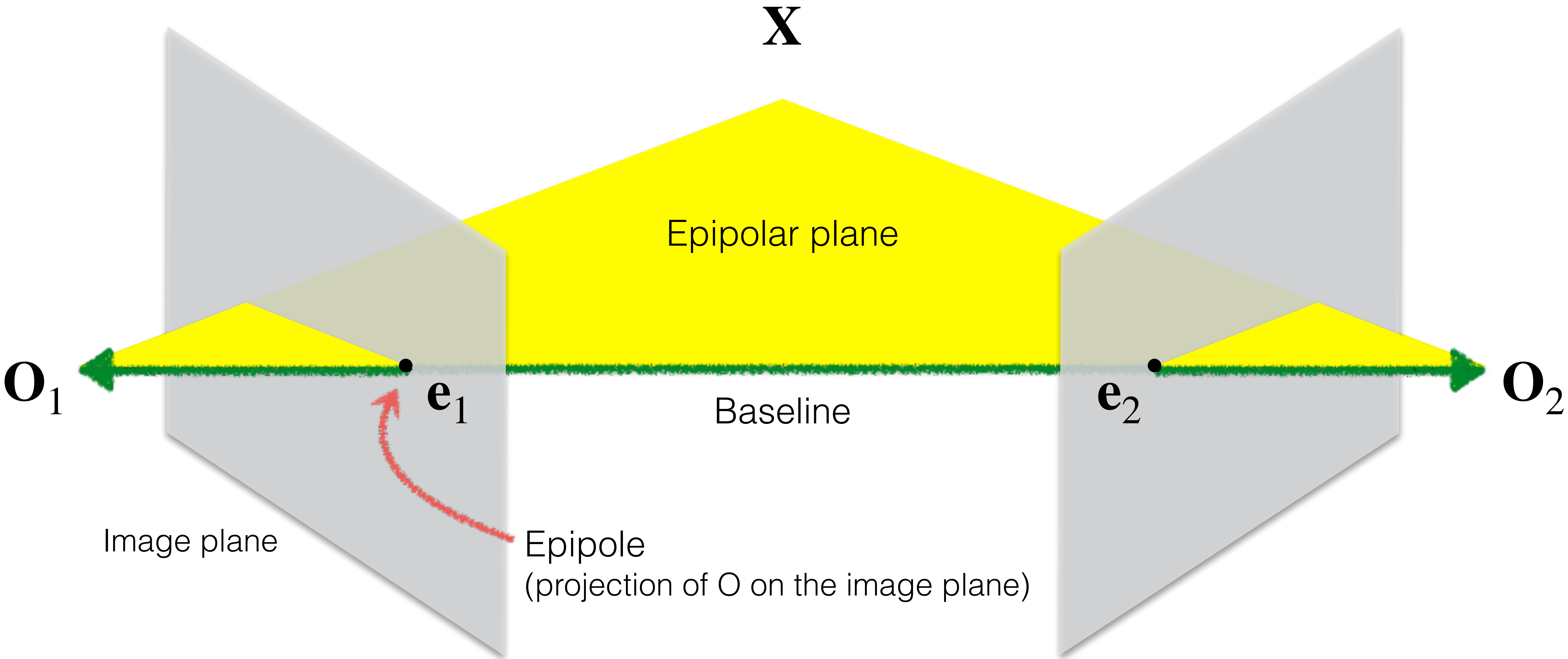
Adapted from: CMU 16-385 (Yannis, Kris)

Epipolar geometry



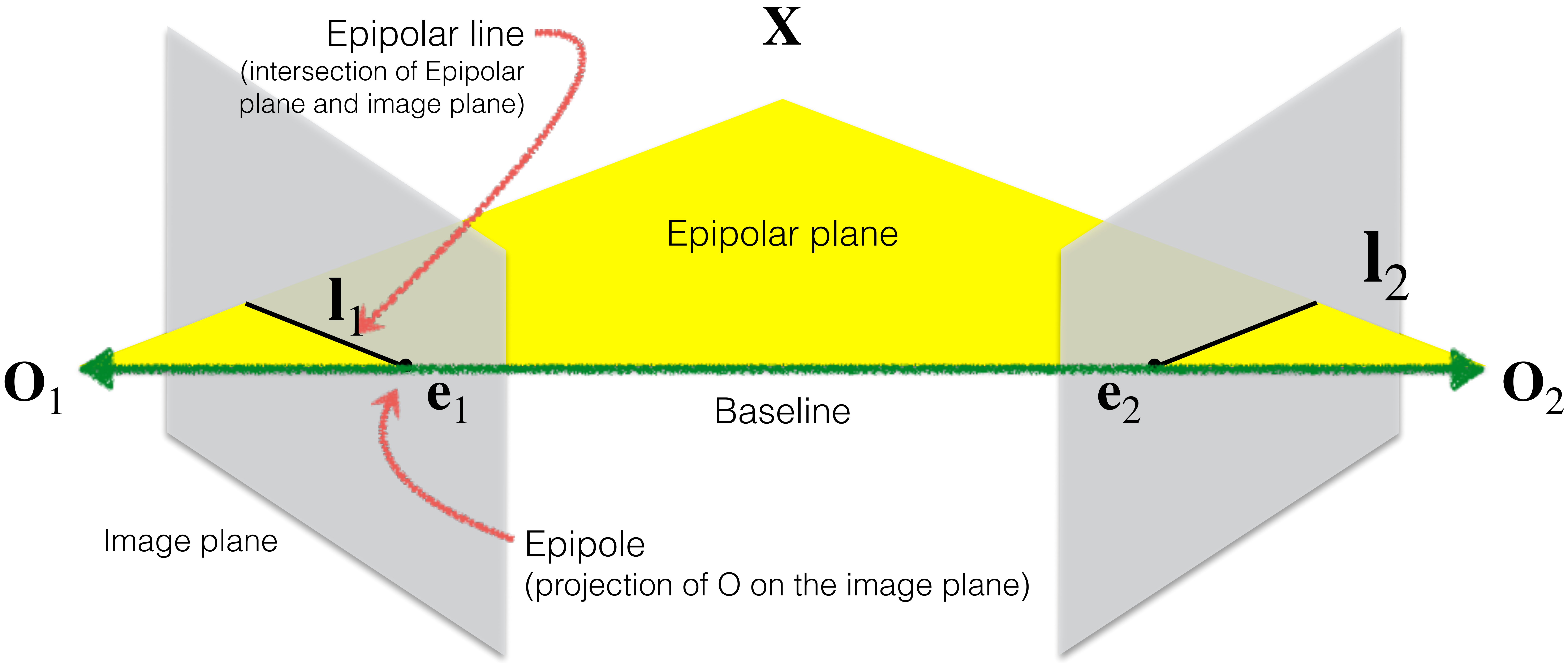
Adapted from: CMU 16-385 (Yannis, Kris)

Epipolar geometry



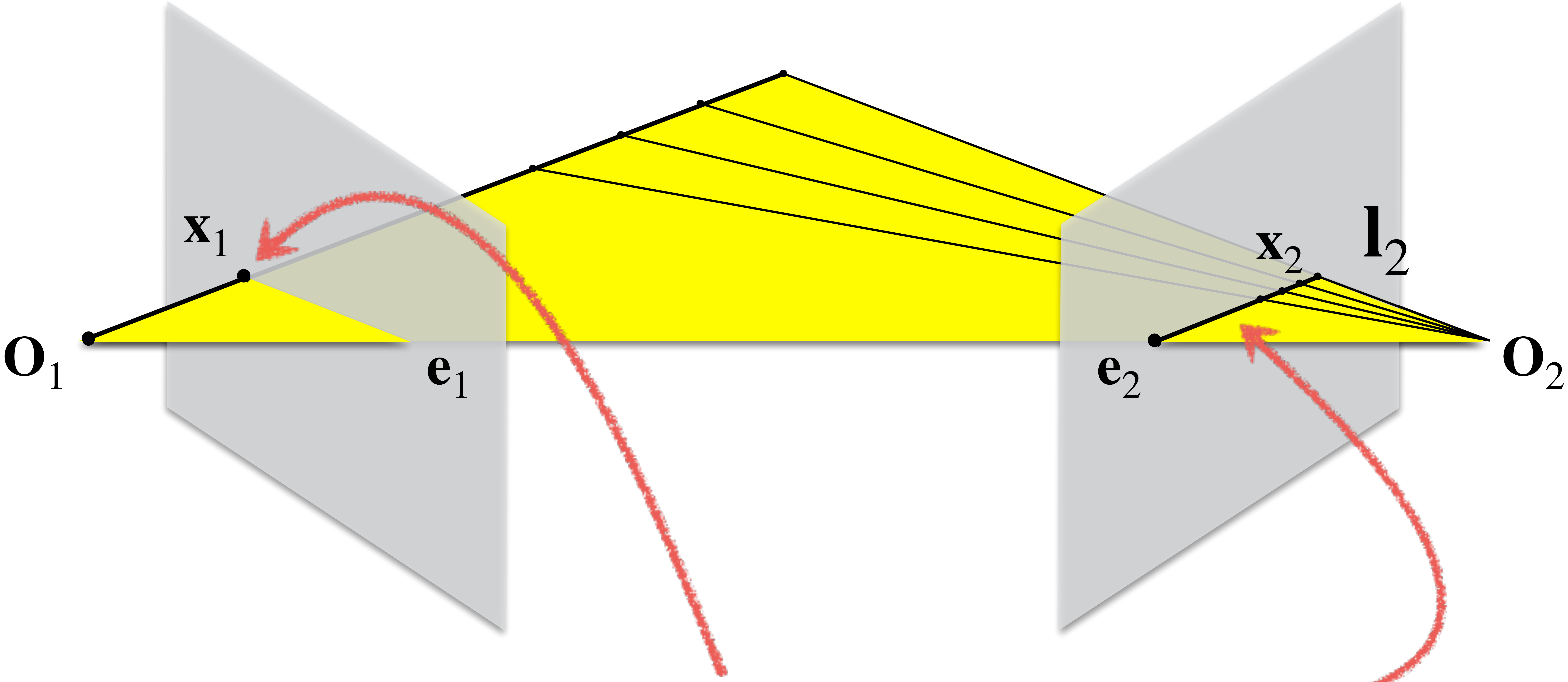
Adapted from: CMU 16-385 (Yannis, Kris)

Epipolar geometry



Adapted from: CMU 16-385 (Yannis, Kris)

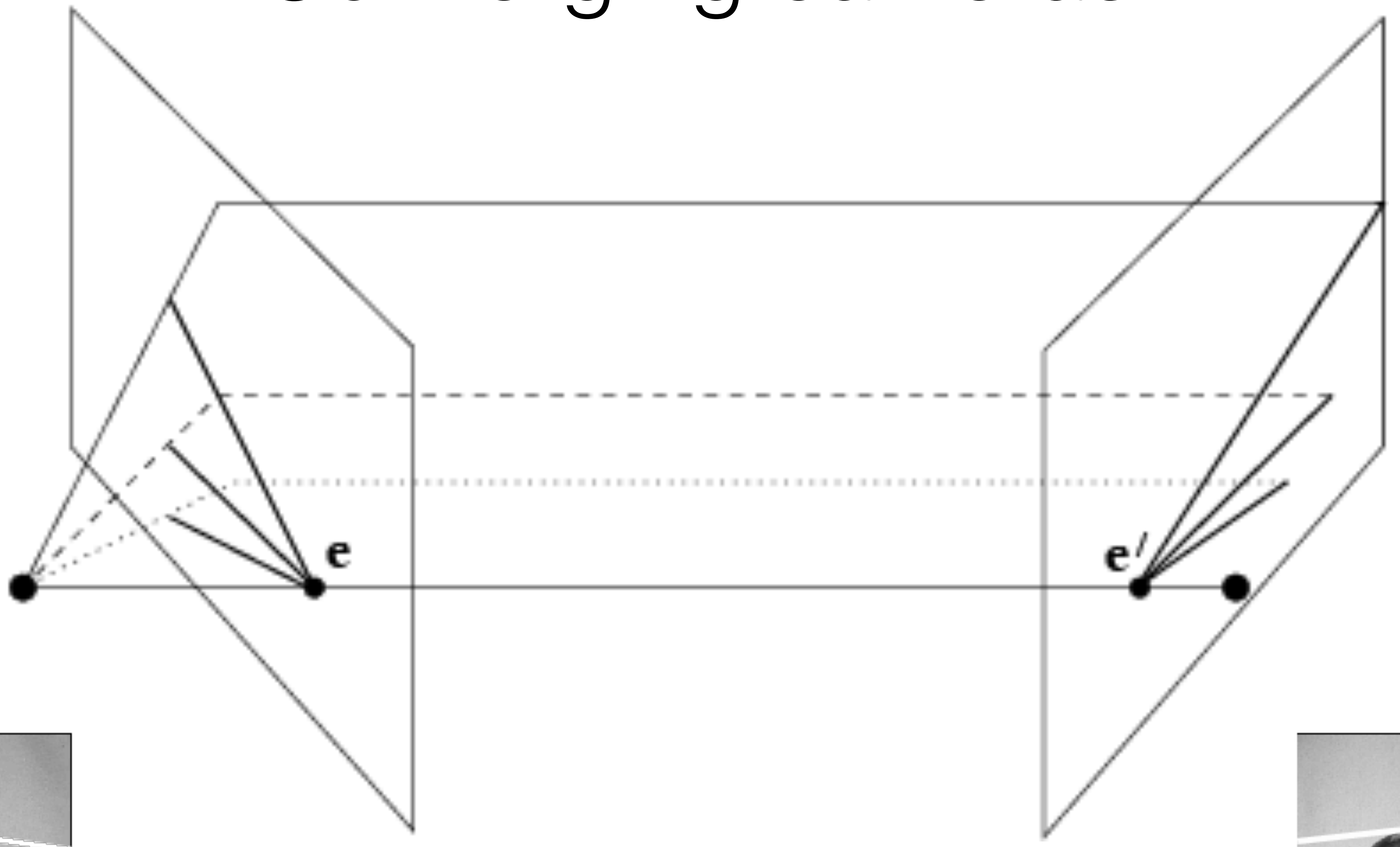
Epipolar constraint

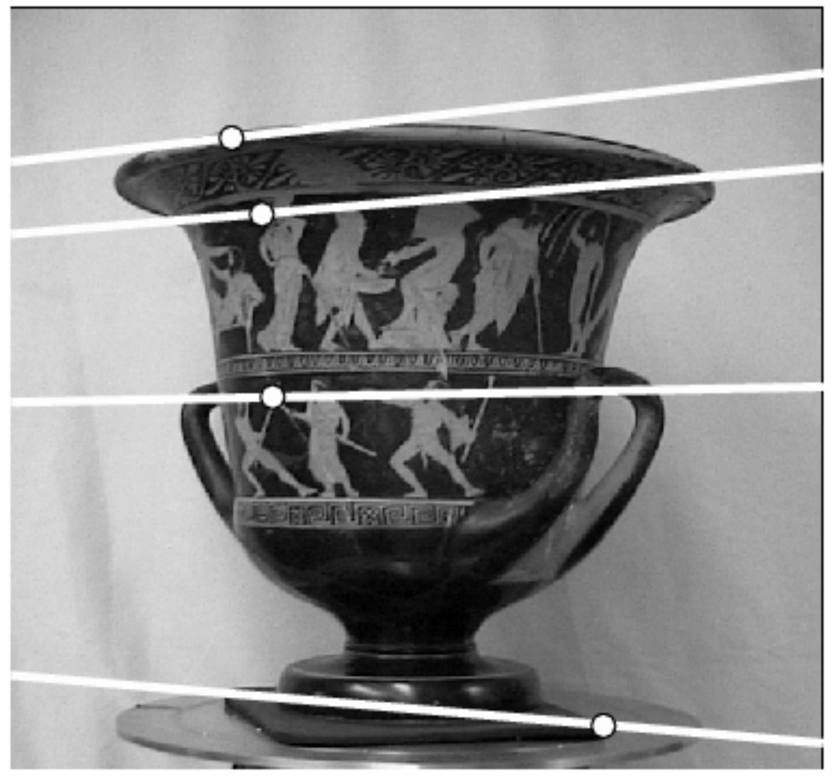


Potential matches for $\mathbf{x}_1$ lie on the epipolar line $\mathbf{l}_2$

Adapted from: CMU 16-385 (Yannis, Kris)

Converging cameras

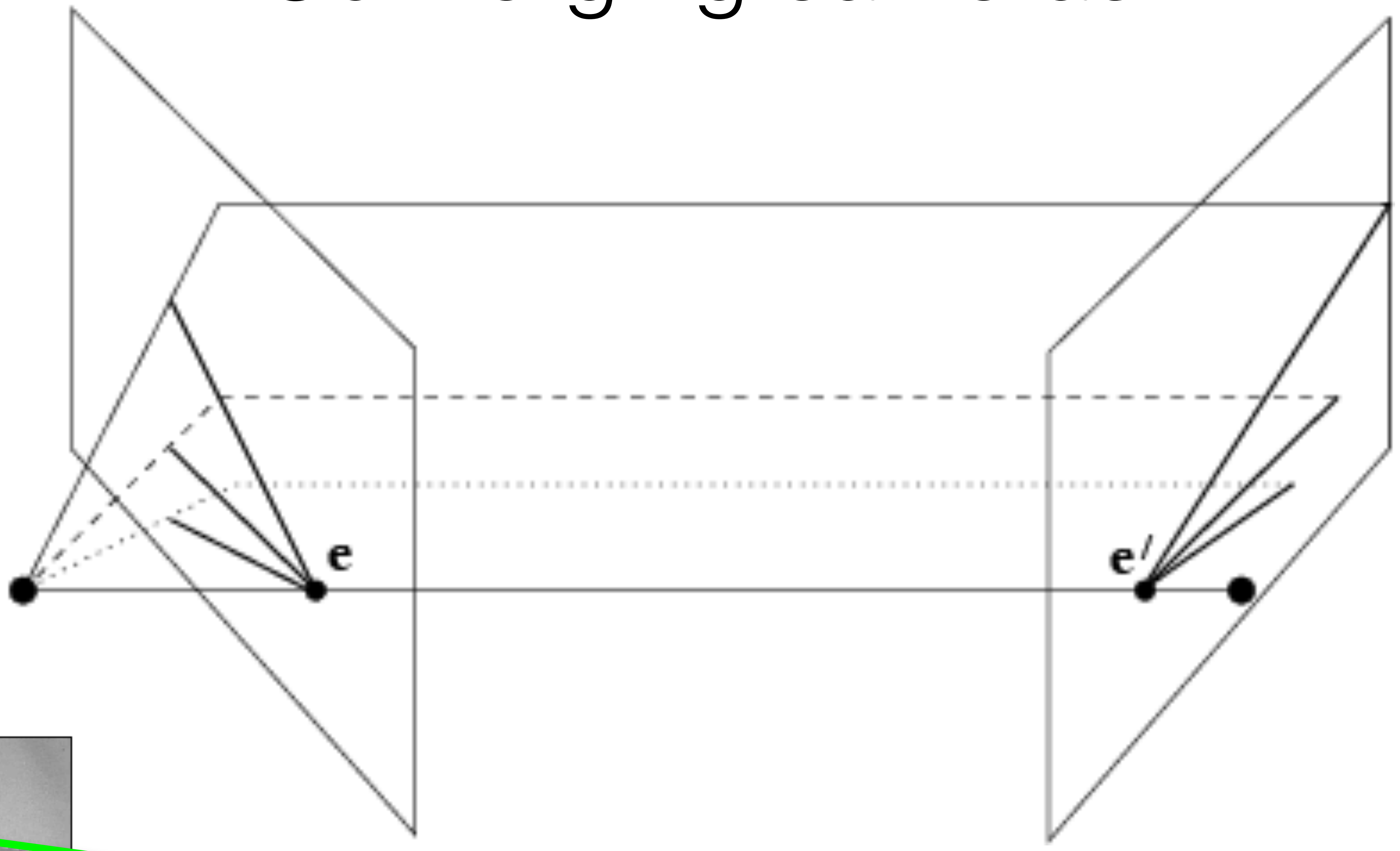


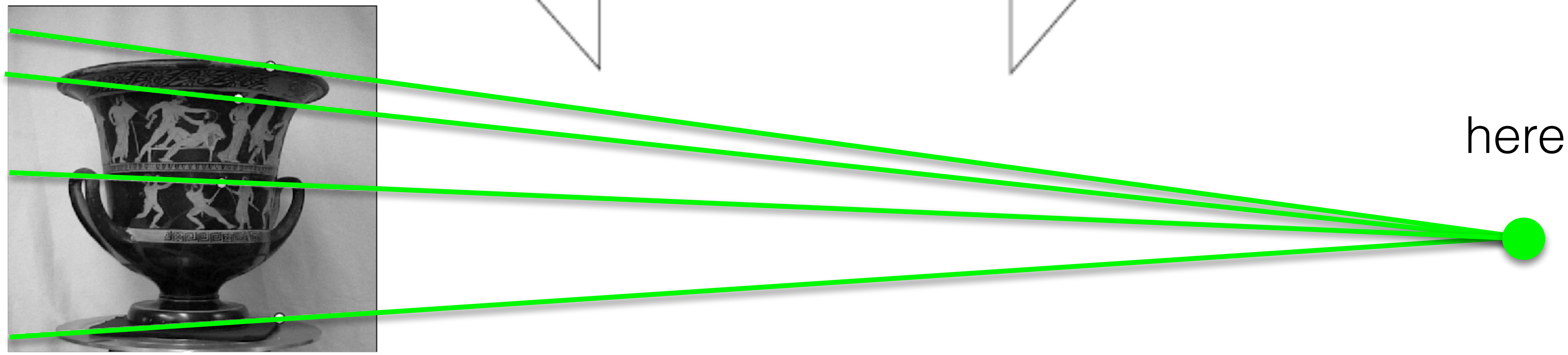


Where is the epipole in this image?

Adapted from: CMU 16-385 (Yannis, Kris)

Converging cameras





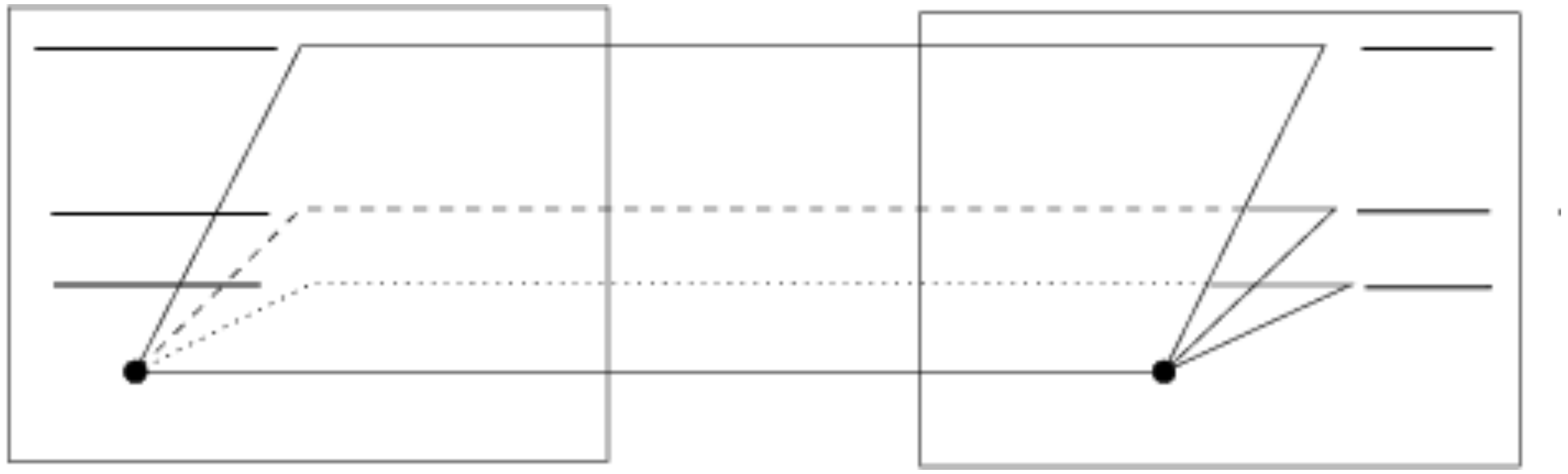
here!

Where is the epipole in this image?

It’s not always in the image

Adapted from: CMU 16-385 (Yannis, Kris)

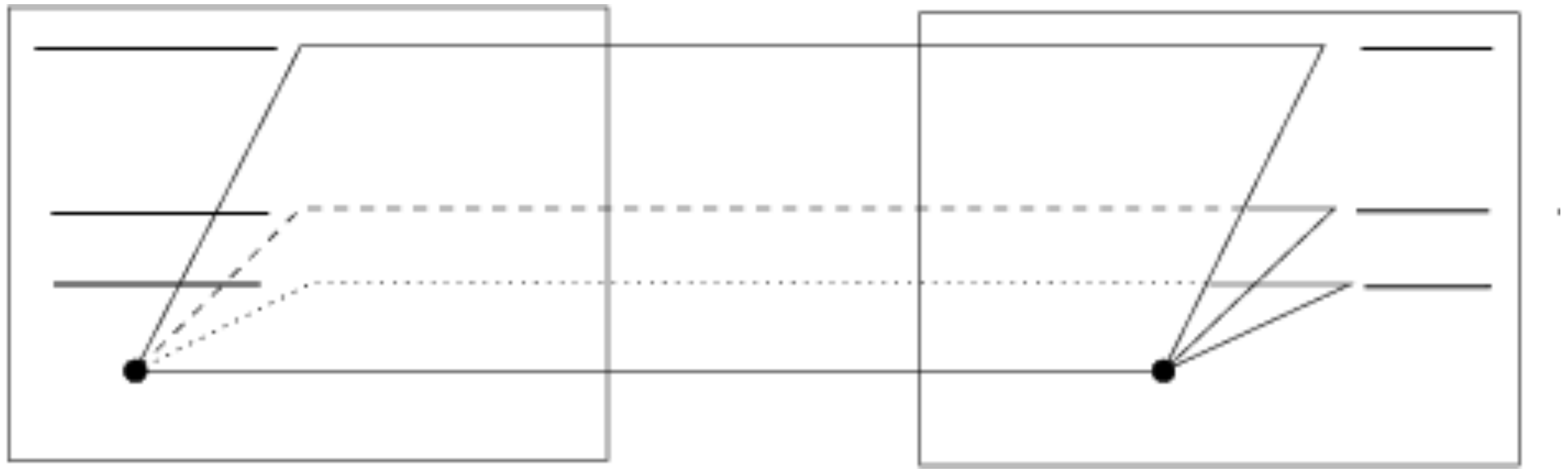
Parallel cameras

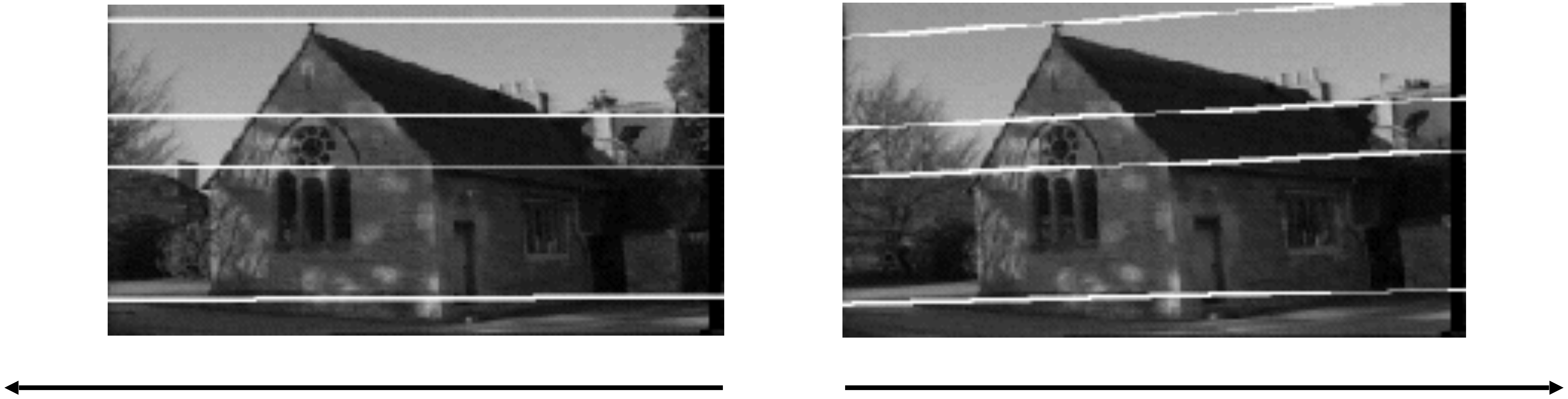


Where is the epipole?

Adapted from: CMU 16-385 (Yannis, Kris)

Parallel cameras

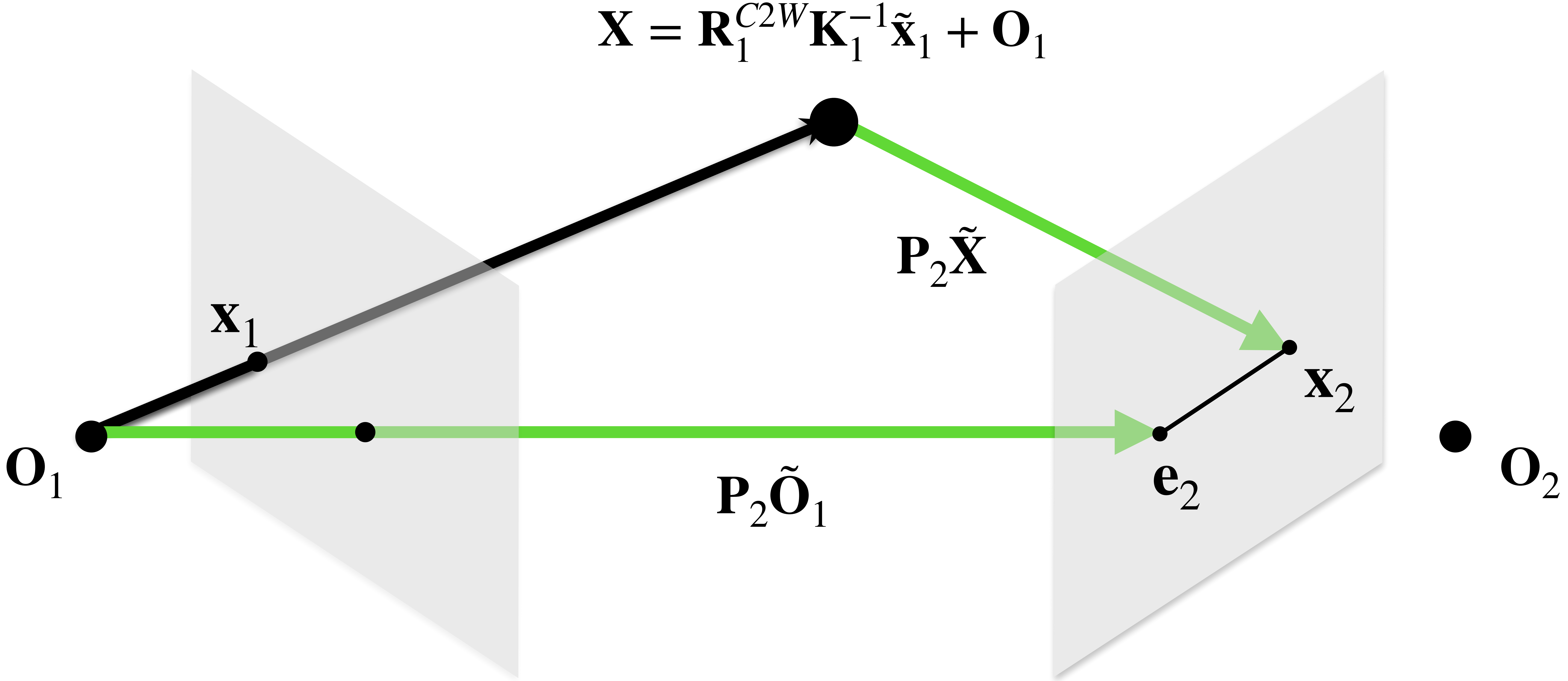


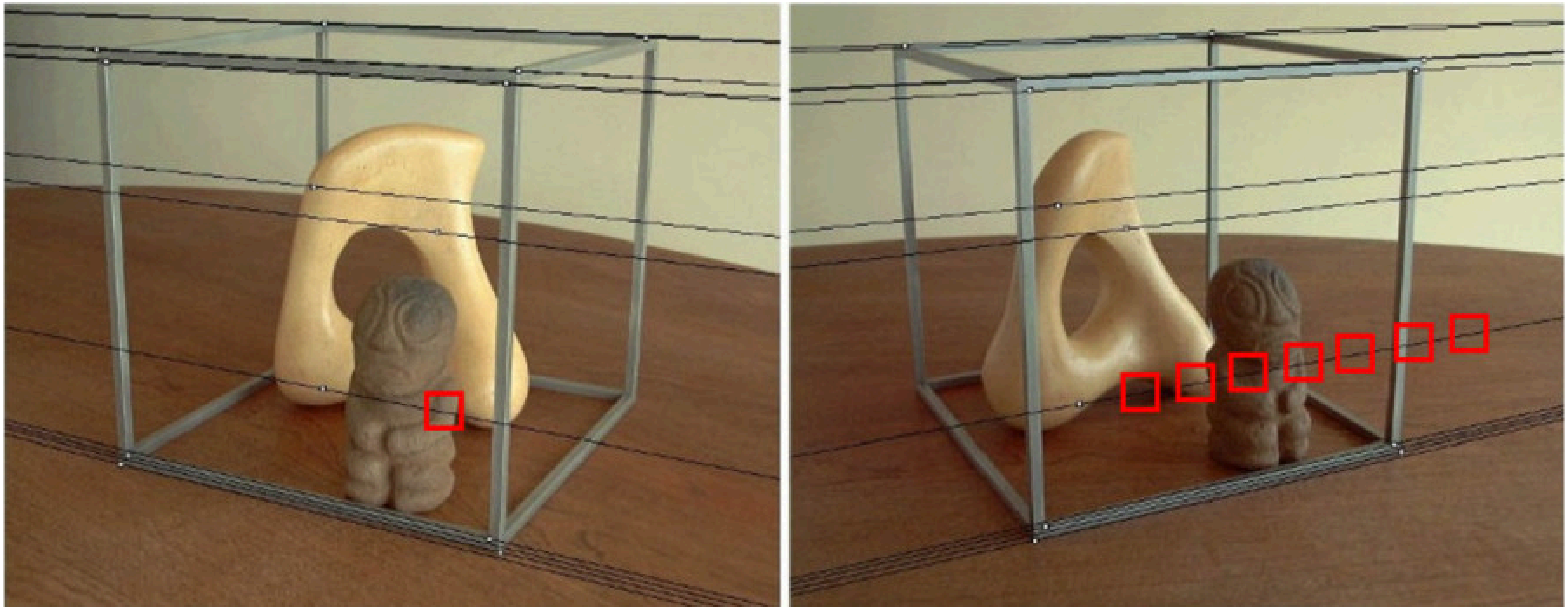


epipole at infinity

Adapted from: CMU 16-385 (Yannis, Kris)

Epipolar Lines: The Hacky Way





Adapted from: CMU 16-385 (Yannis, Kris)

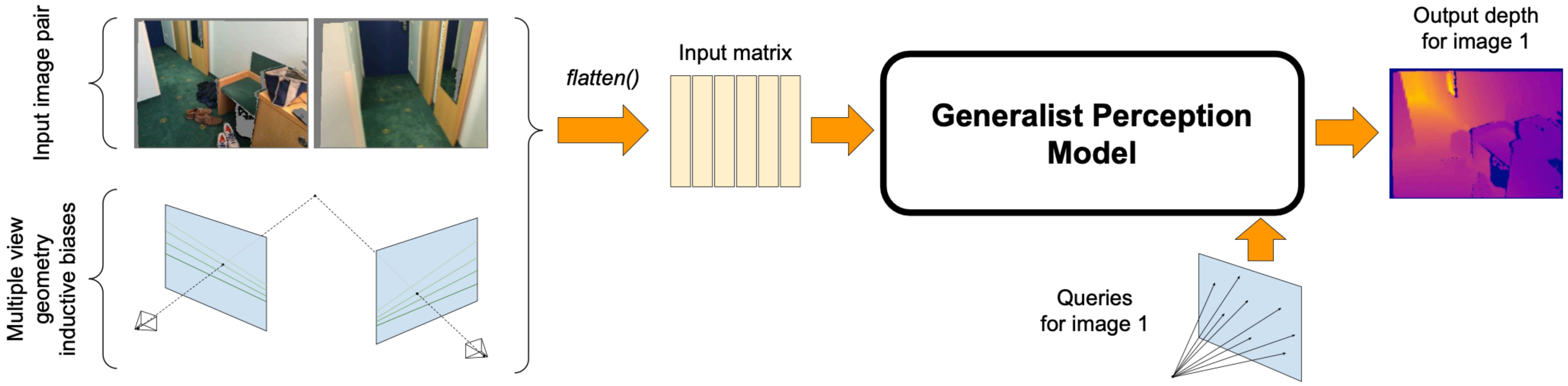
Generalizable Patch-Based Neural Rendering

Mohammed Suhail[1], Carlos Esteves[4], Leonid Sigal[1,2,3], and Ameesh Makadia[4]

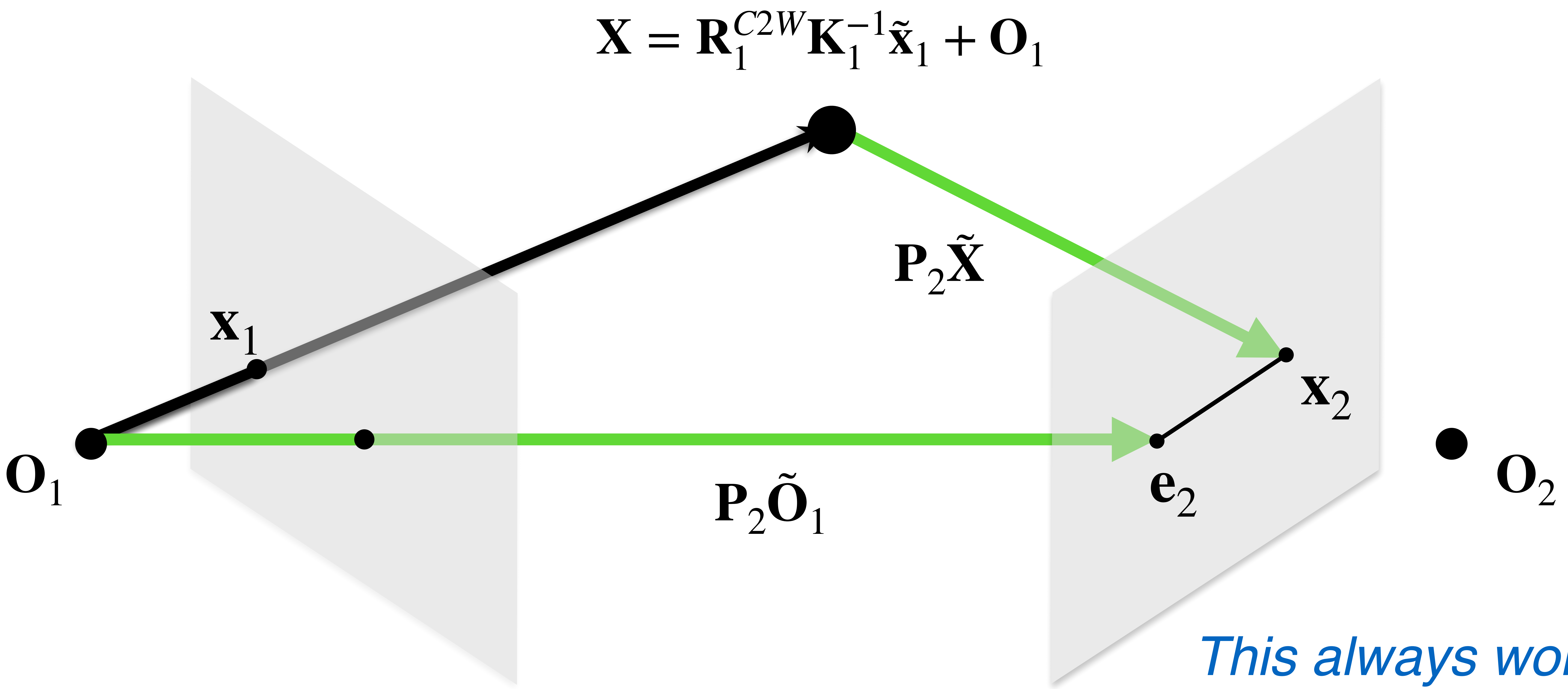
Input-level Inductive Biases for 3D Reconstruction

Wang Yifan[1*] Carl Doersch[2] Relja Arandjelović[2] João Carreira[2] Andrew Zisserman[2,3]

[1]ETH Zurich [2]DeepMind [3]VGG, 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

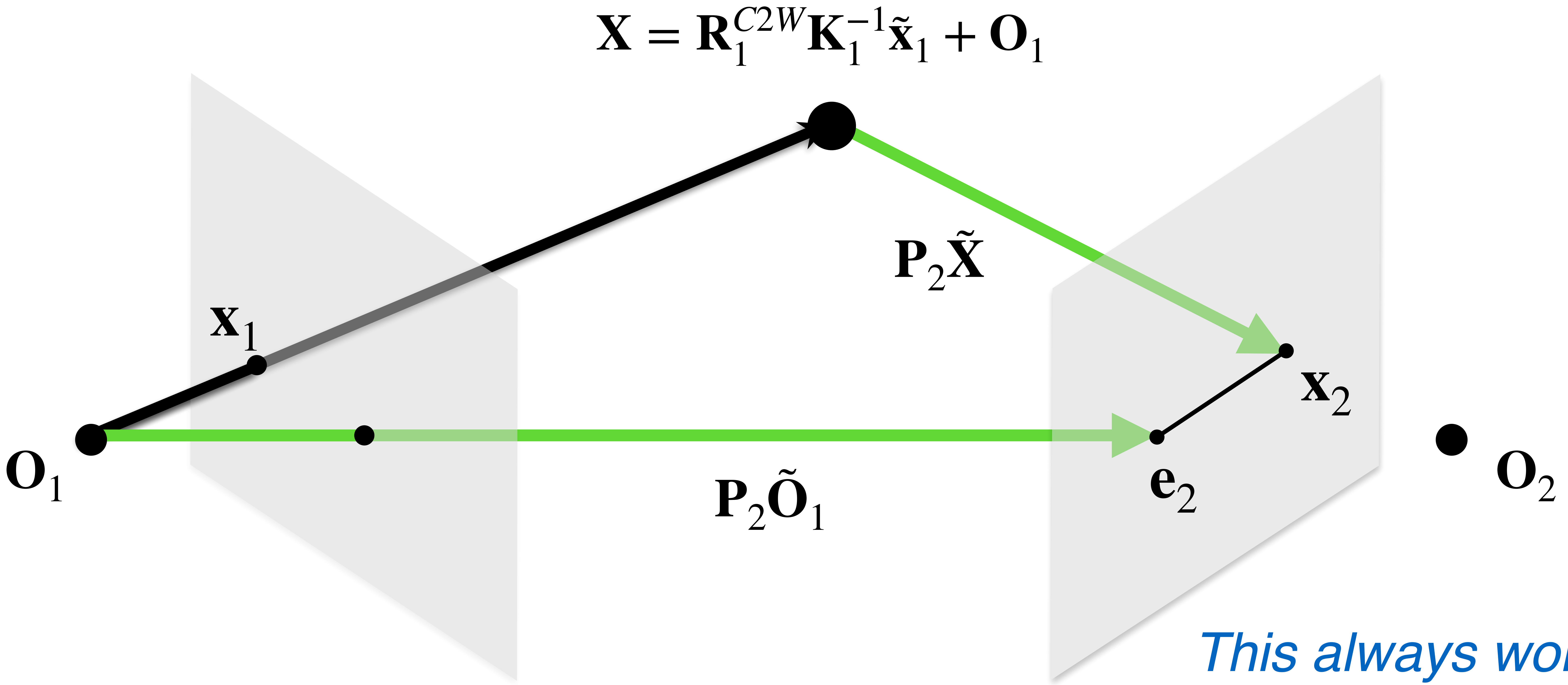
A way of *estimating camera poses, intrinsics, and extrinsic from correspondences.*

No Time

-
-
-

What has changed since Deep Learning?

Epipolar Lines: The Hacky Way



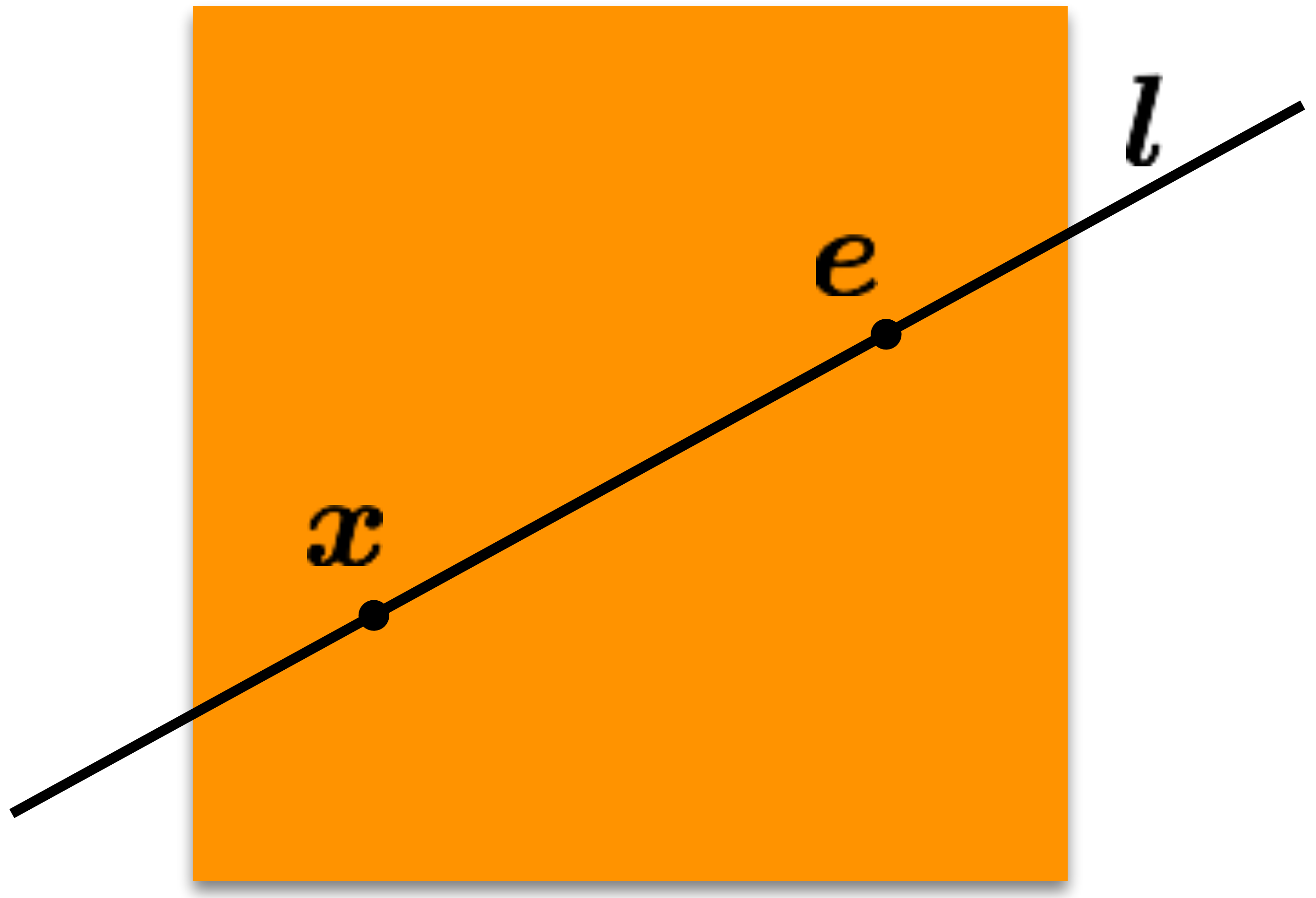
This always works ;)
But: Not clean, many steps.
Is there a better way?

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

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

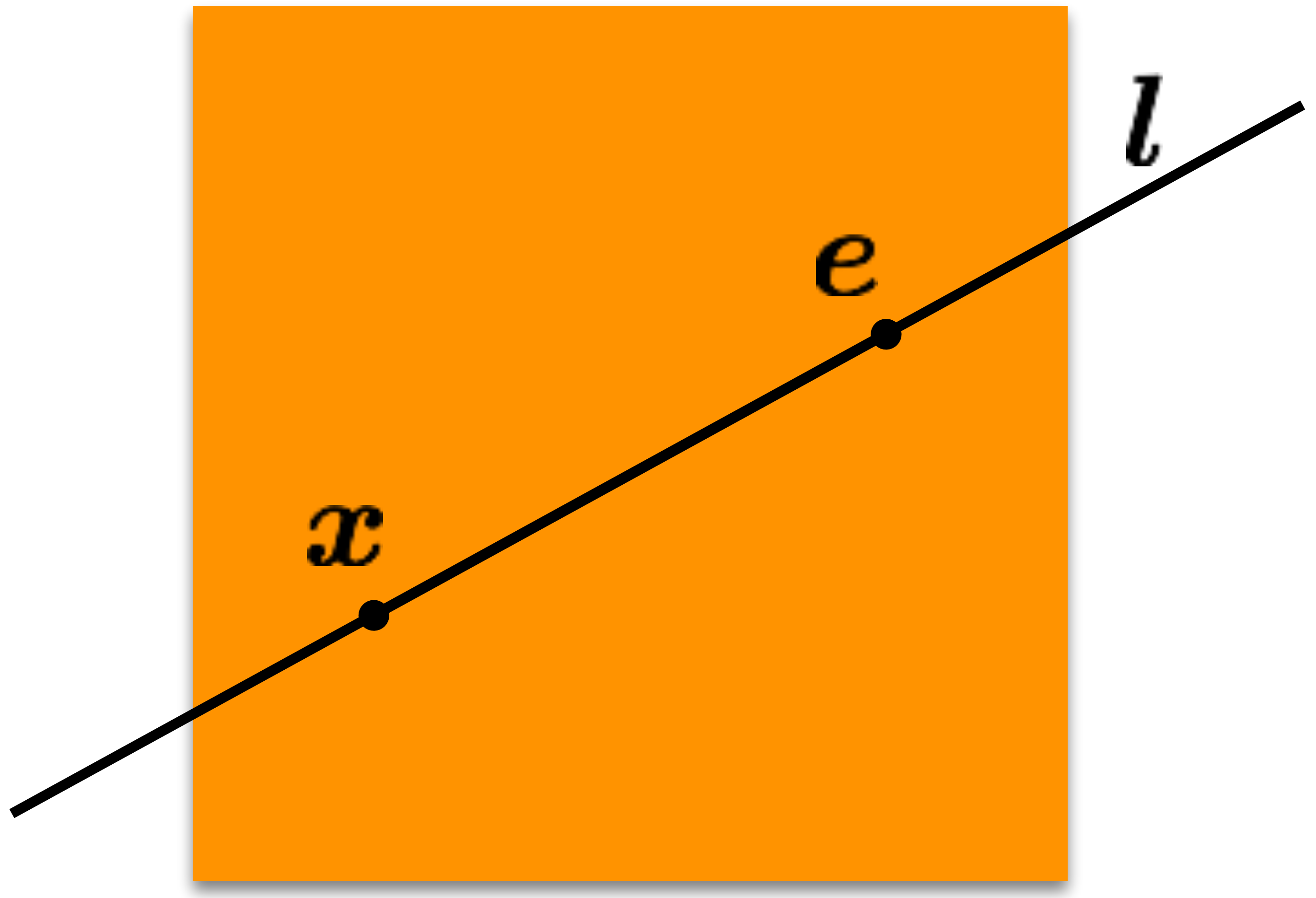
Adapted from: CMU 16-385 (Yannis, Kris)

Lines in Homogeneous Coordinates

$$ax + by + c = 0$$

in vector form

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



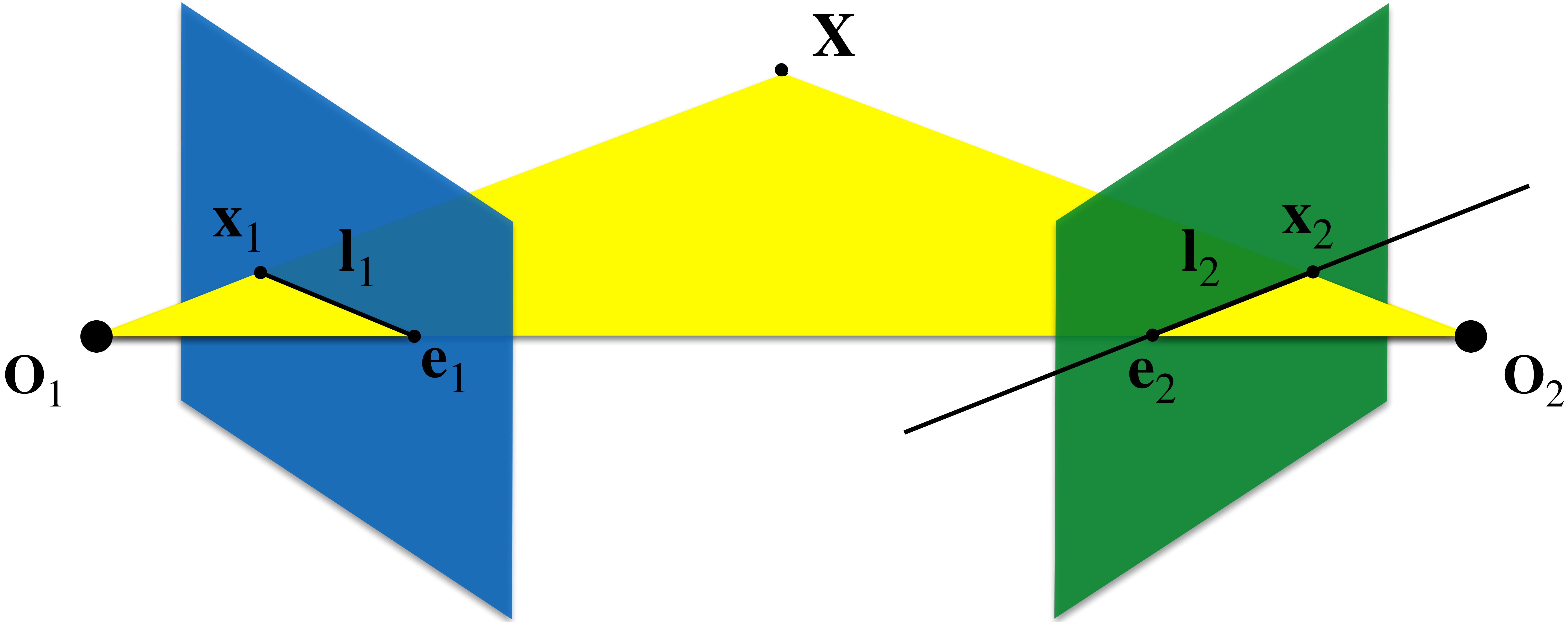
If the point $\mathbf{x}$ is on the epipolar line $\mathbf{l}$ then

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

Adapted from: CMU 16-385 (Yannis, Kris)

Introducing: The Fundamental Matrix $\mathbf{F}$

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



Adapted from: CMU 16-385 (Yannis, Kris)

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

The Fundamental Matrix is a 3 x 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

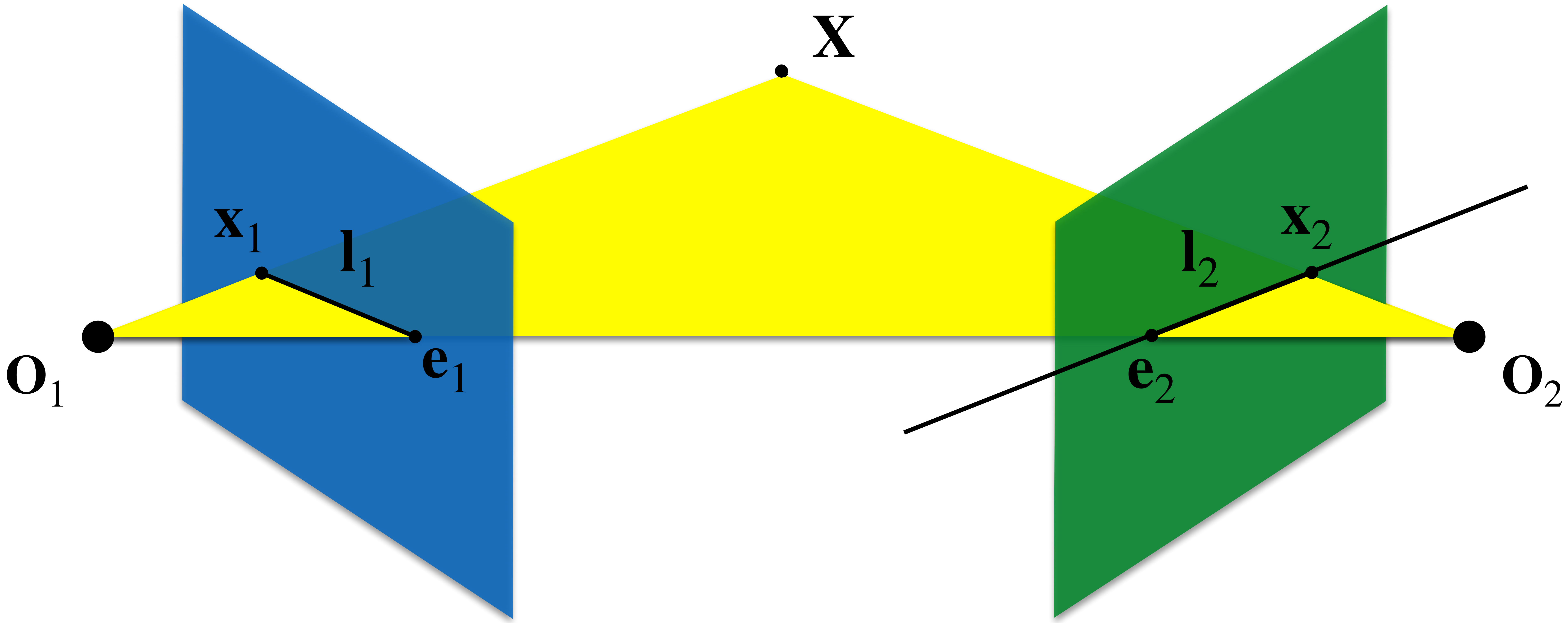
$$\boxed{\mathbf{F}\tilde{\mathbf{x}}_1 = \mathbf{l}_2}$$

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

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

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

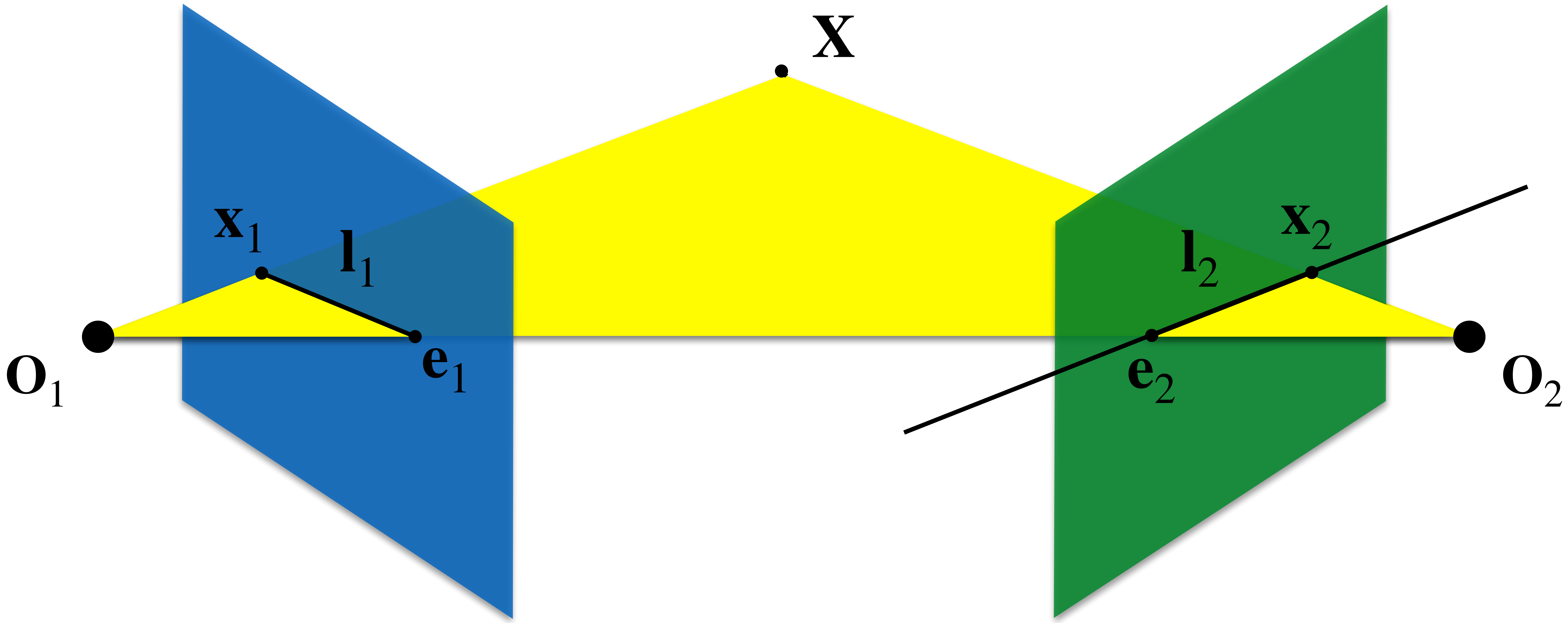
$$\tilde{\mathbf{x}}_2^T \mathbf{F} \tilde{\mathbf{x}}_1 = ?$$



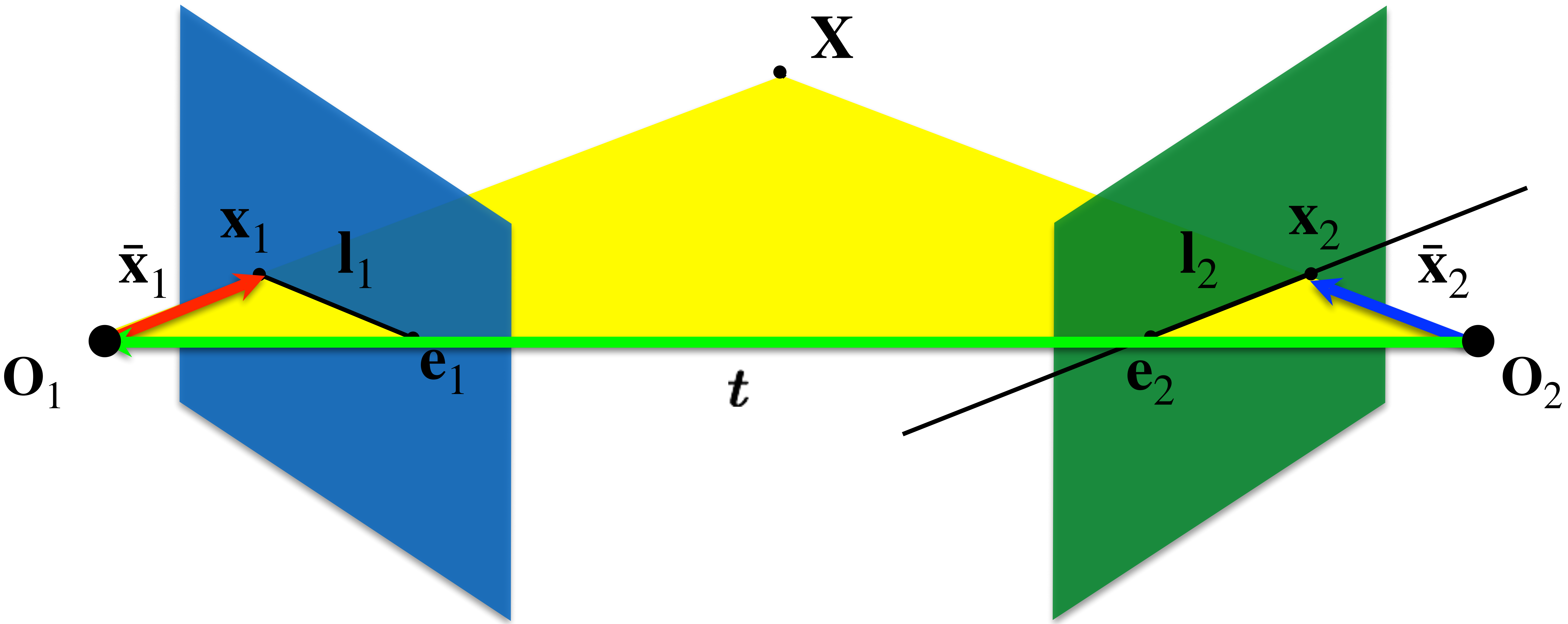
Adapted from: CMU 16-385 (Yannis, Kris)

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

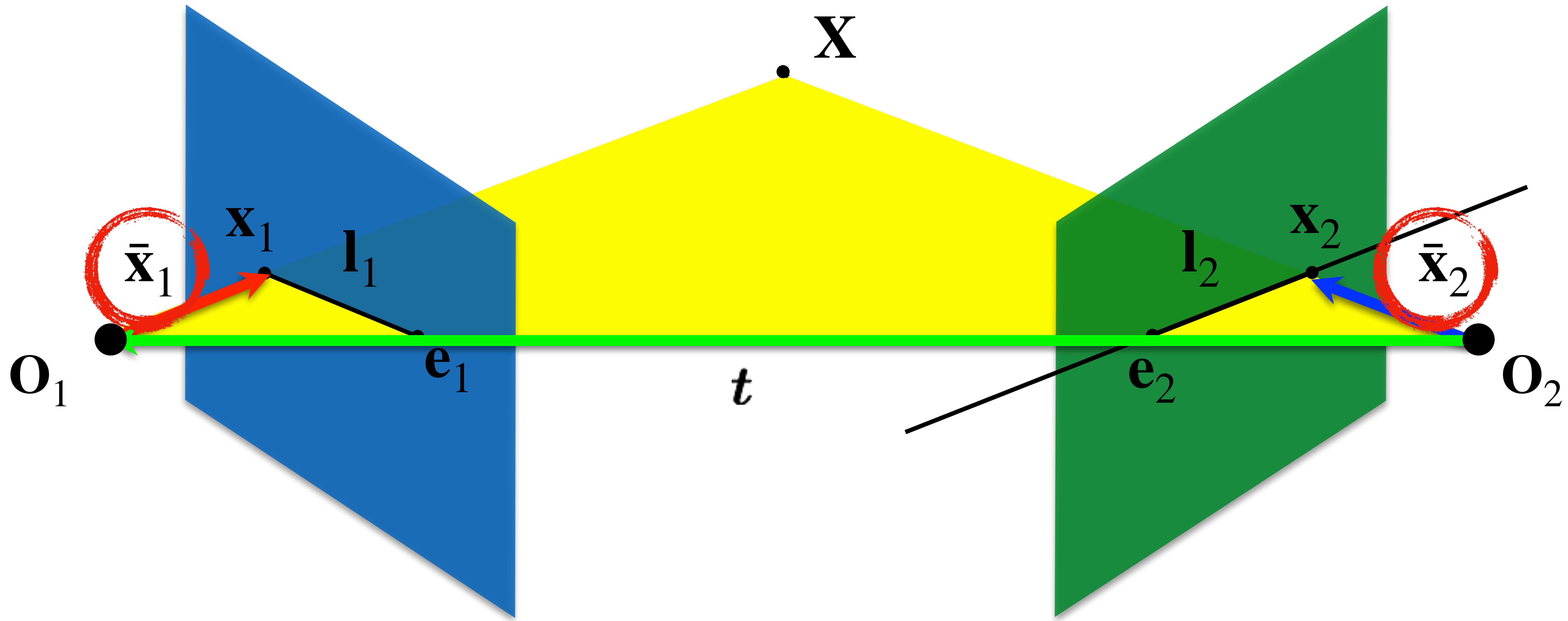
$$\boxed{\tilde{\mathbf{x}}_2^T \mathbf{F} \tilde{\mathbf{x}}_1 = 0}$$



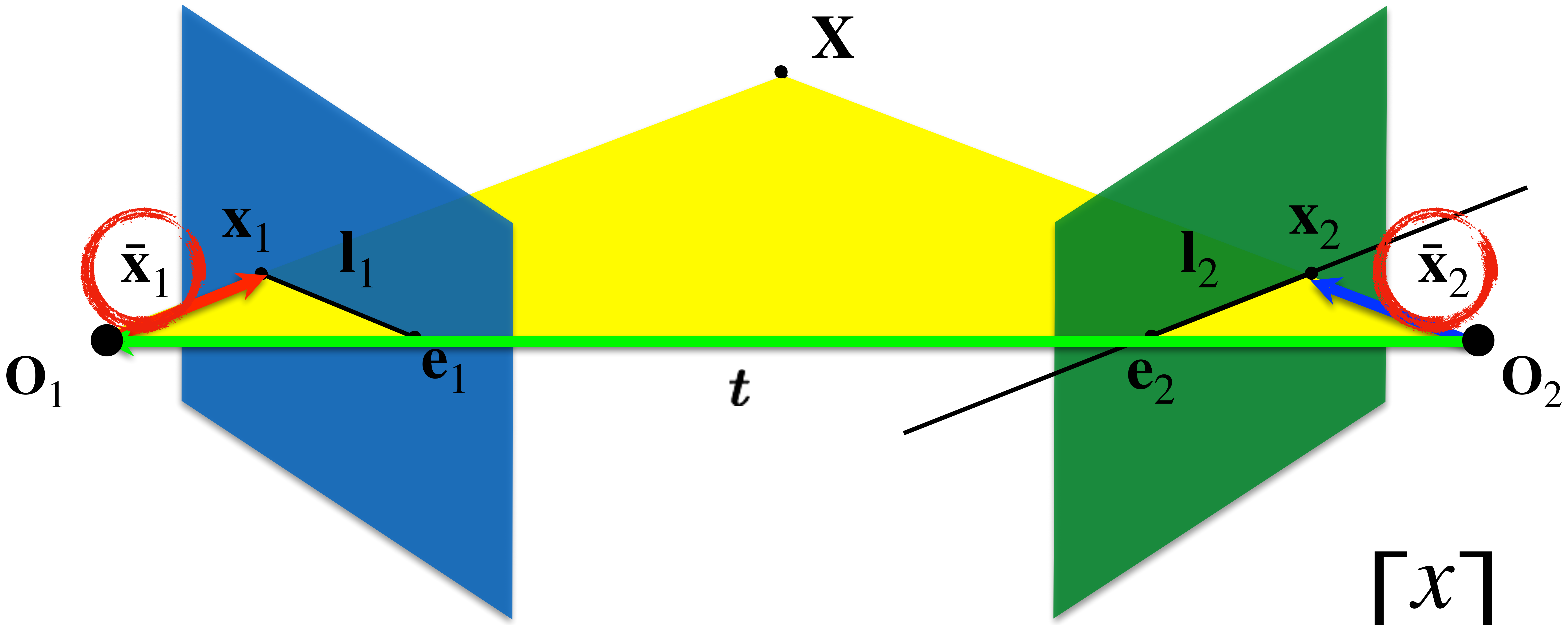
Adapted from: CMU 16-385 (Yannis, Kris)



Adapted from: CMU 16-385 (Yannis, Kris)



Adapted from: CMU 16-385 (Yannis, Kris)



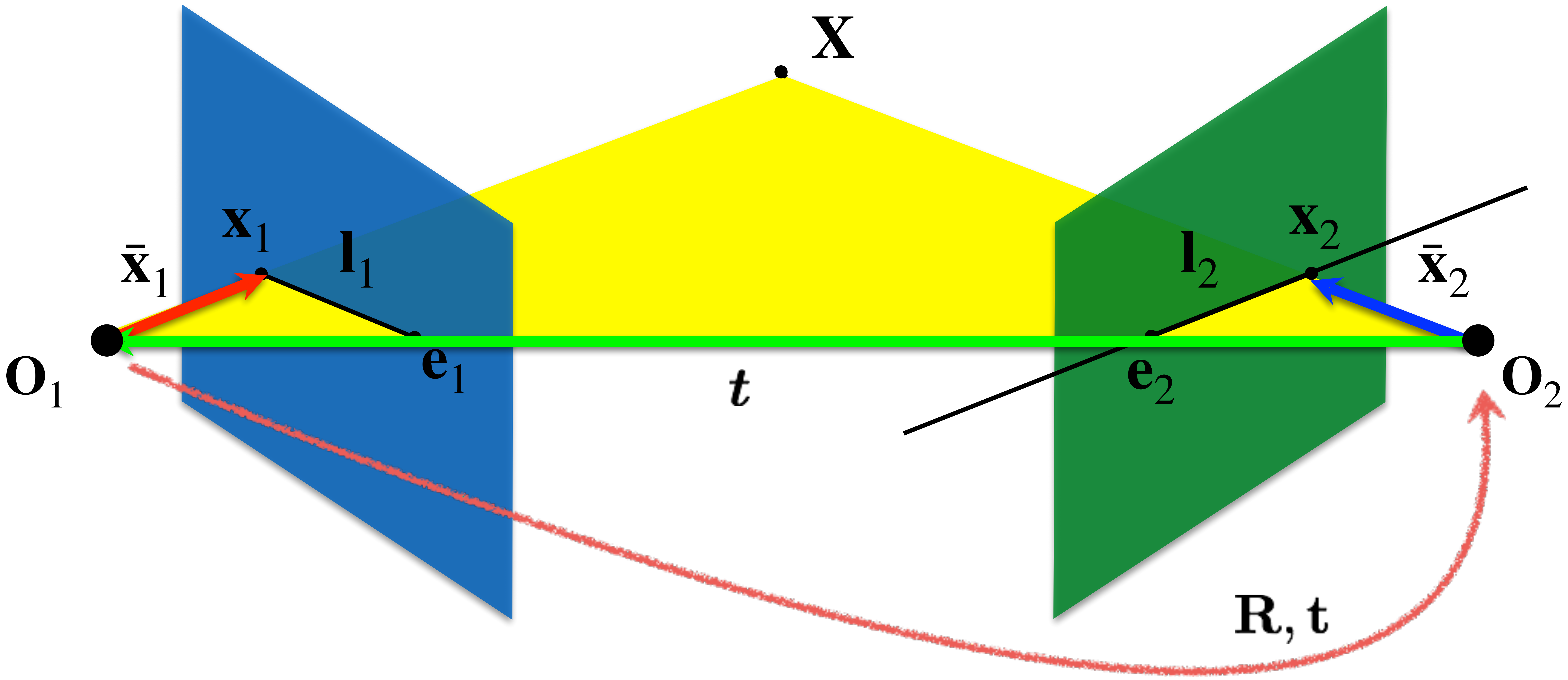
$$\bar{\mathbf{x}} = \mathbf{K}^{-1}\tilde{\mathbf{x}}$$

The local ray direction!

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

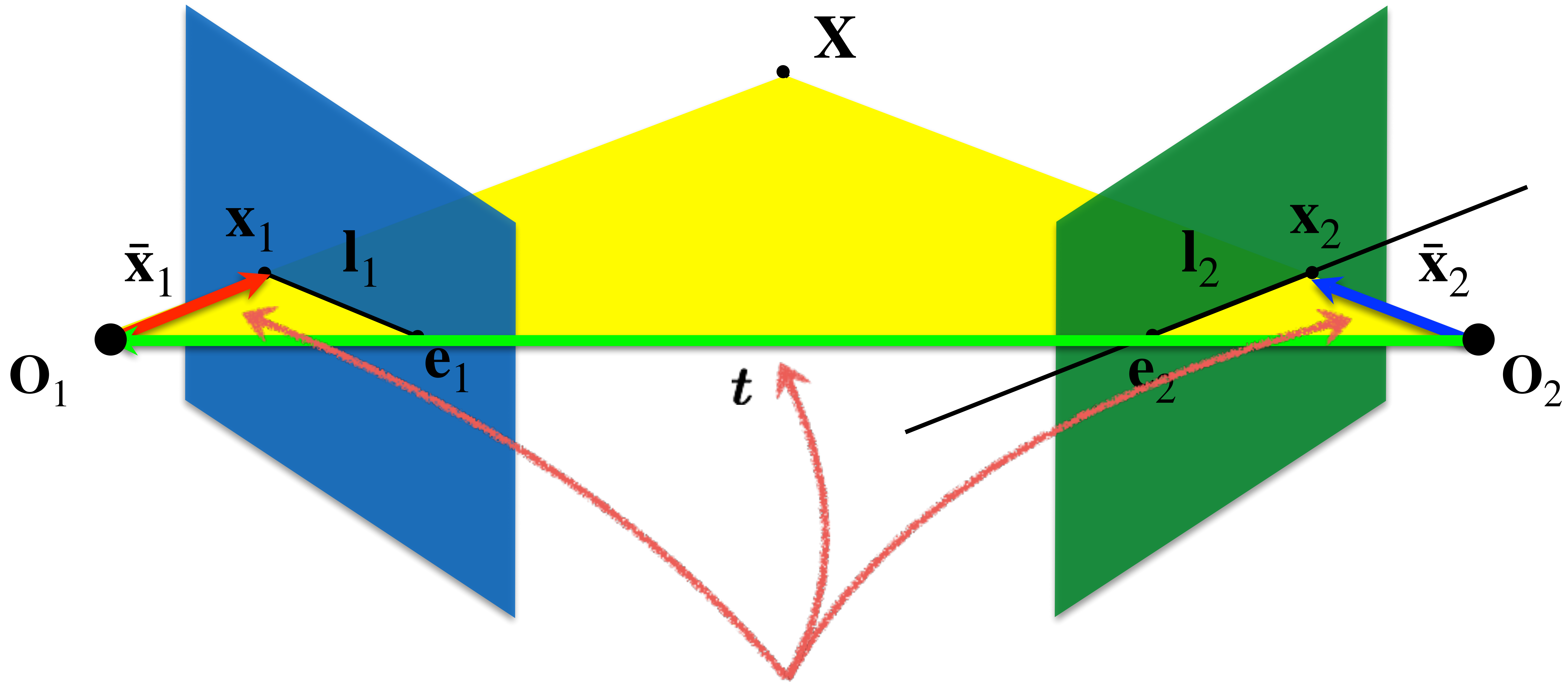
Homogenized pixel coordinate

Adapted from: CMU 16-385 (Yannis, Kris)



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

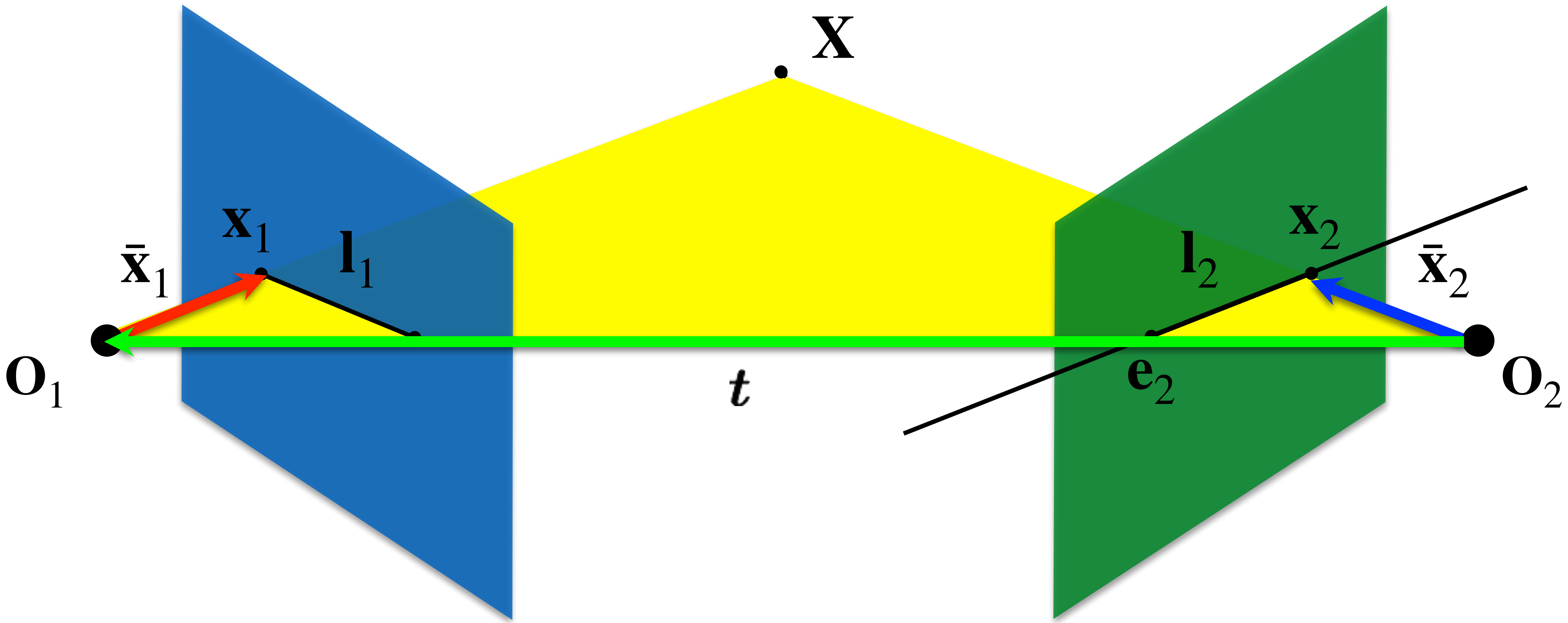
Adapted from: CMU 16-385 (Yannis, Kris)



These three vectors are coplanar

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

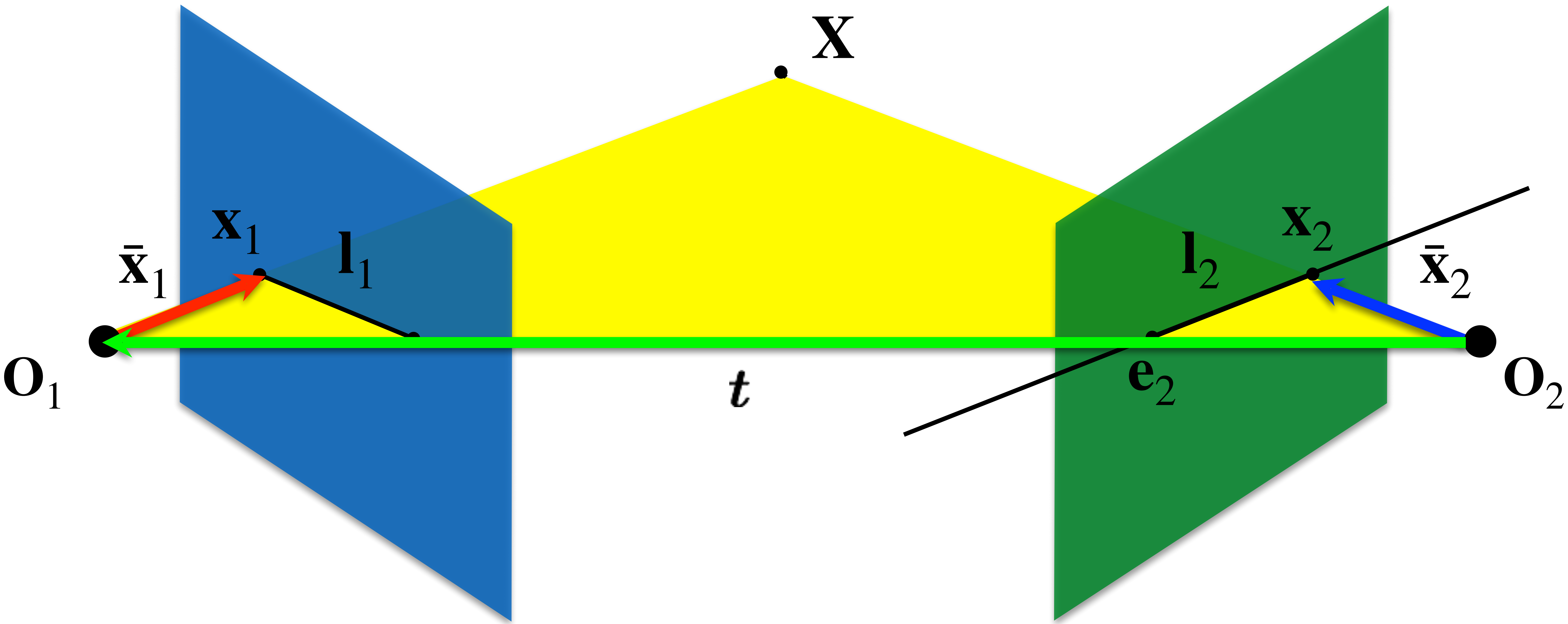
Adapted from: CMU 16-385 (Yannis, Kris)



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

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

Adapted from: CMU 16-385 (Yannis, Kris)

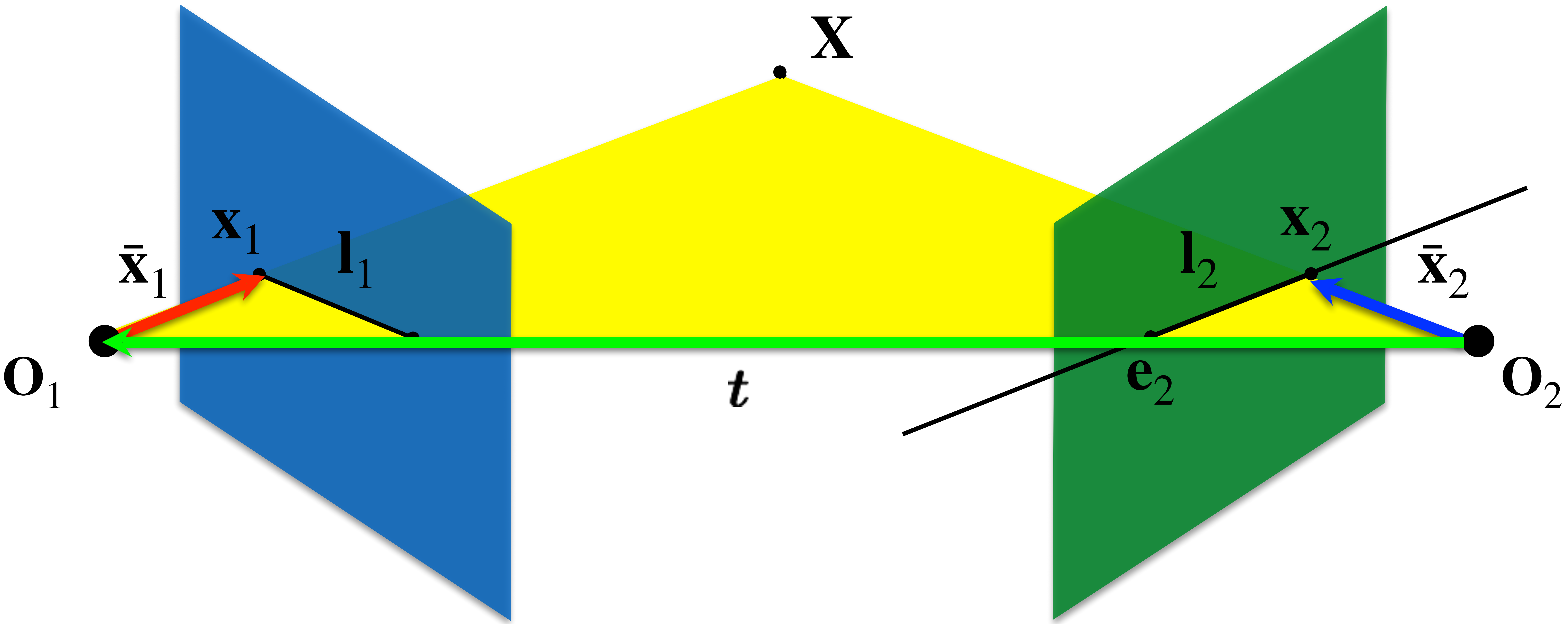


If $\bar{\mathbf{x}}_1, \mathbf{t}, \bar{\mathbf{x}}_2$ are coplanar then

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

dot product of orthogonal vectors

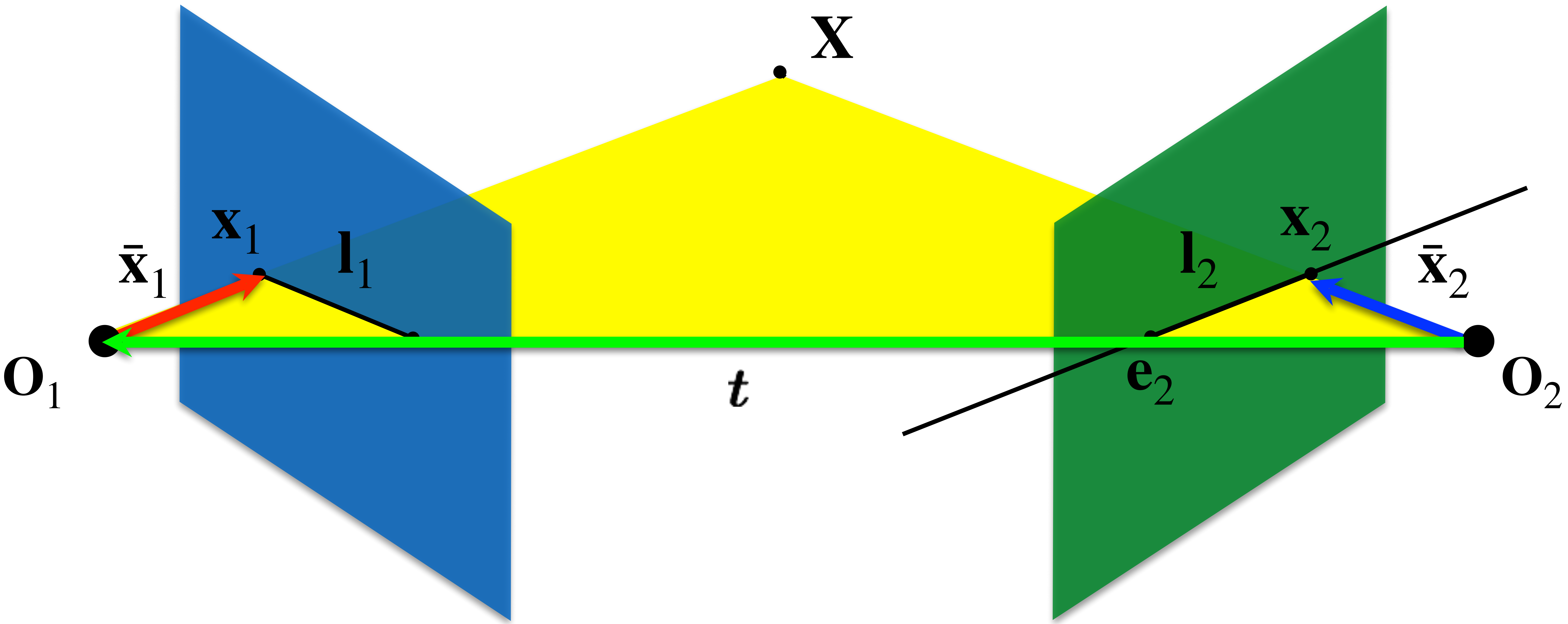
cross-product: vector orthogonal to plane



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

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

Adapted from: CMU 16-385 (Yannis, Kris)



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

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

Adapted from: CMU 16-385 (Yannis, Kris)

putting it together

rigid motion

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

coplanarity

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

Adapted from: CMU 16-385 (Yannis, Kris)

putting it together

rigid motion

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

coplanarity

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

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

Adapted from: CMU 16-385 (Yannis, Kris)

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

Adapted from: CMU 16-385 (Yannis, Kris)

putting it together

rigid motion

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

coplanarity

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

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

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

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

Adapted from: CMU 16-385 (Yannis, Kris)

putting it together

rigid motion

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

coplanarity

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

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

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

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

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

Adapted from: CMU 16-385 (Yannis, Kris)

putting it together

rigid motion

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

coplanarity

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

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

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

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

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

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

Adapted from: CMU 16-385 (Yannis, Kris)

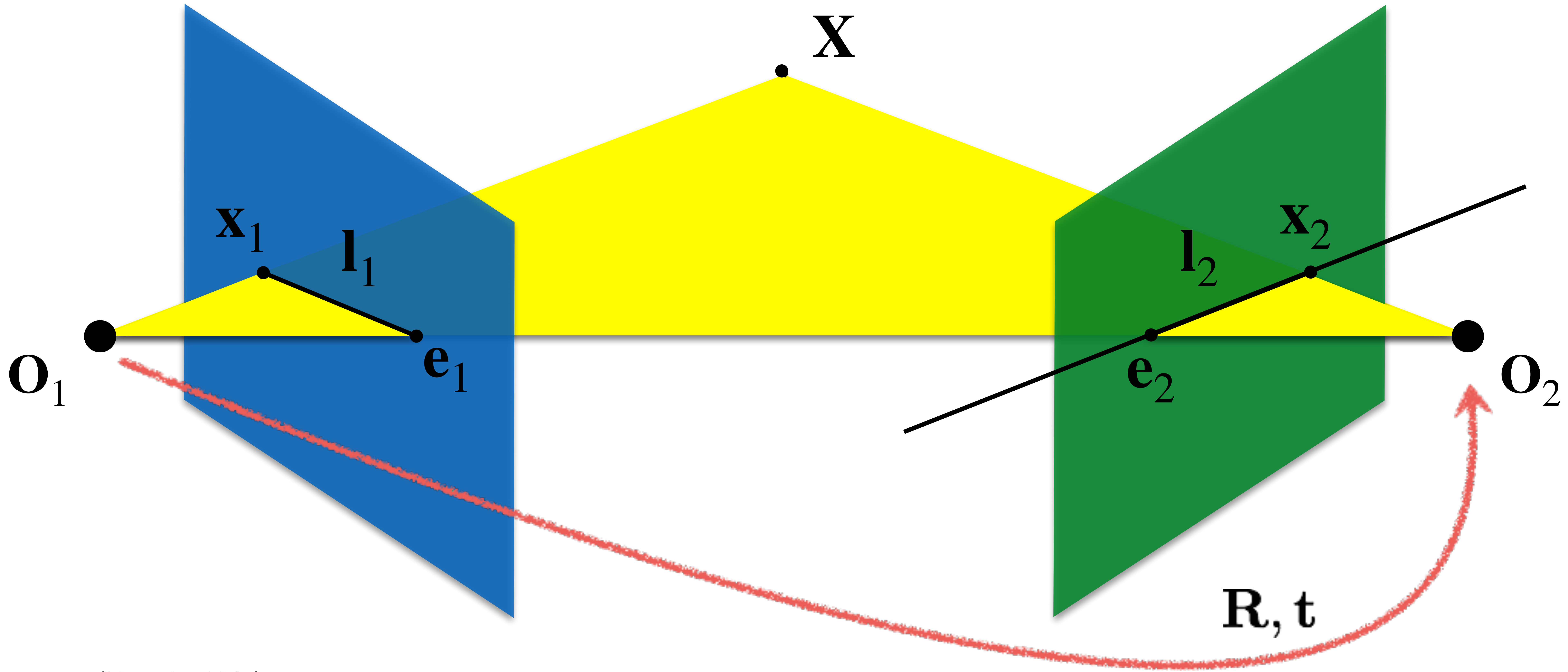
putting it together

$$\boxed{\begin{gathered}\mathbf{F} = \mathbf{K}_2^{-1}(\mathbf{R}[\mathbf{t}]_{\times})\mathbf{K}_1^{-1} \\ \mathbf{F}\tilde{\mathbf{x}}_1 = \mathbf{l}_2\end{gathered}}$$

Adapted from: CMU 16-385 (Yannis, Kris)

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

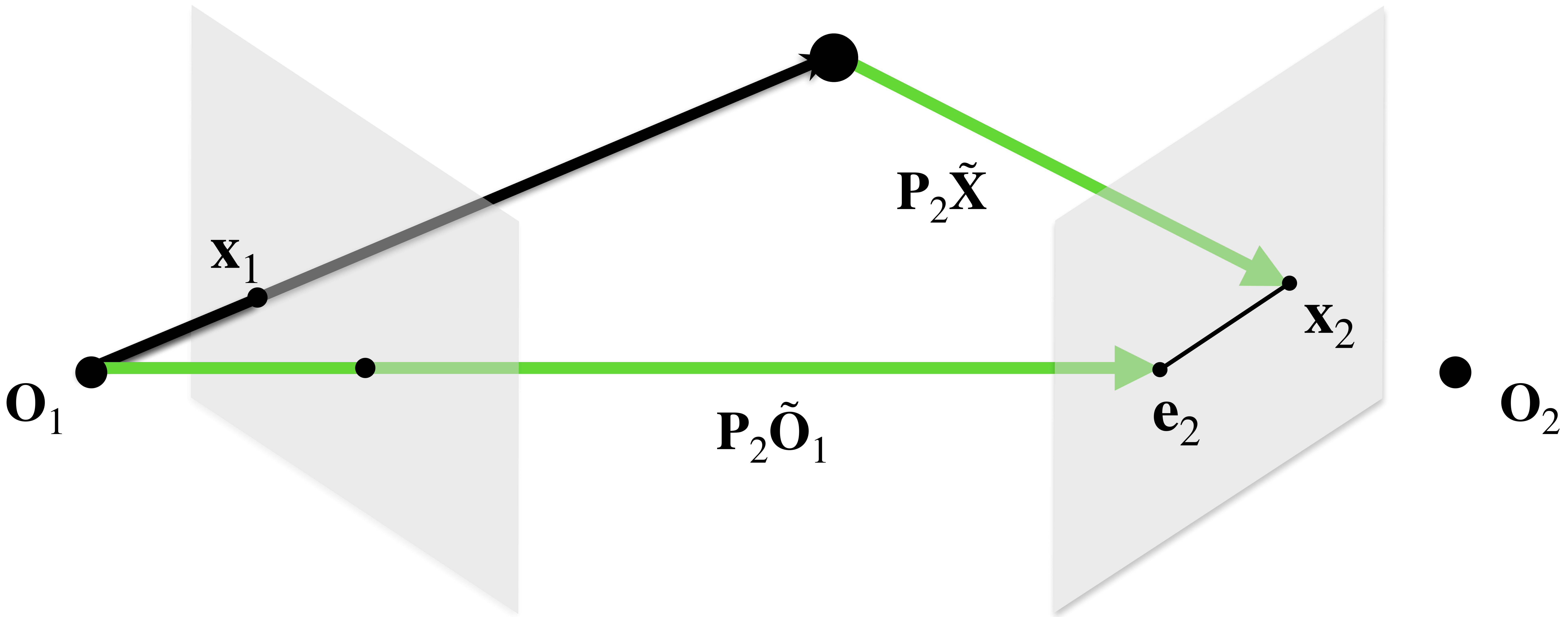
$$\mathbf{F}\tilde{\mathbf{x}}_1 = \mathbf{l}_2$$



Adapted from: CMU 16-385 (Yannis, Kris)

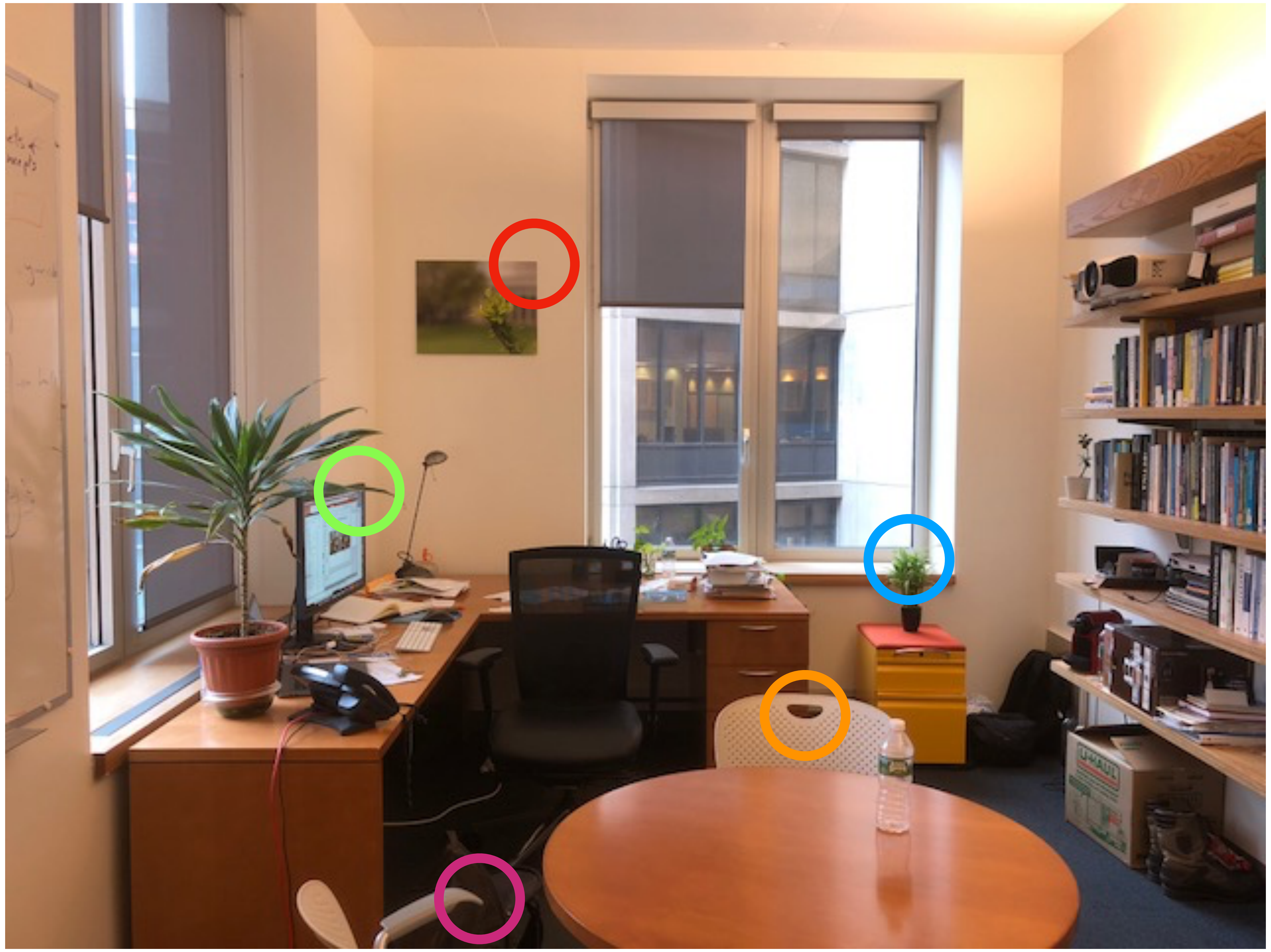
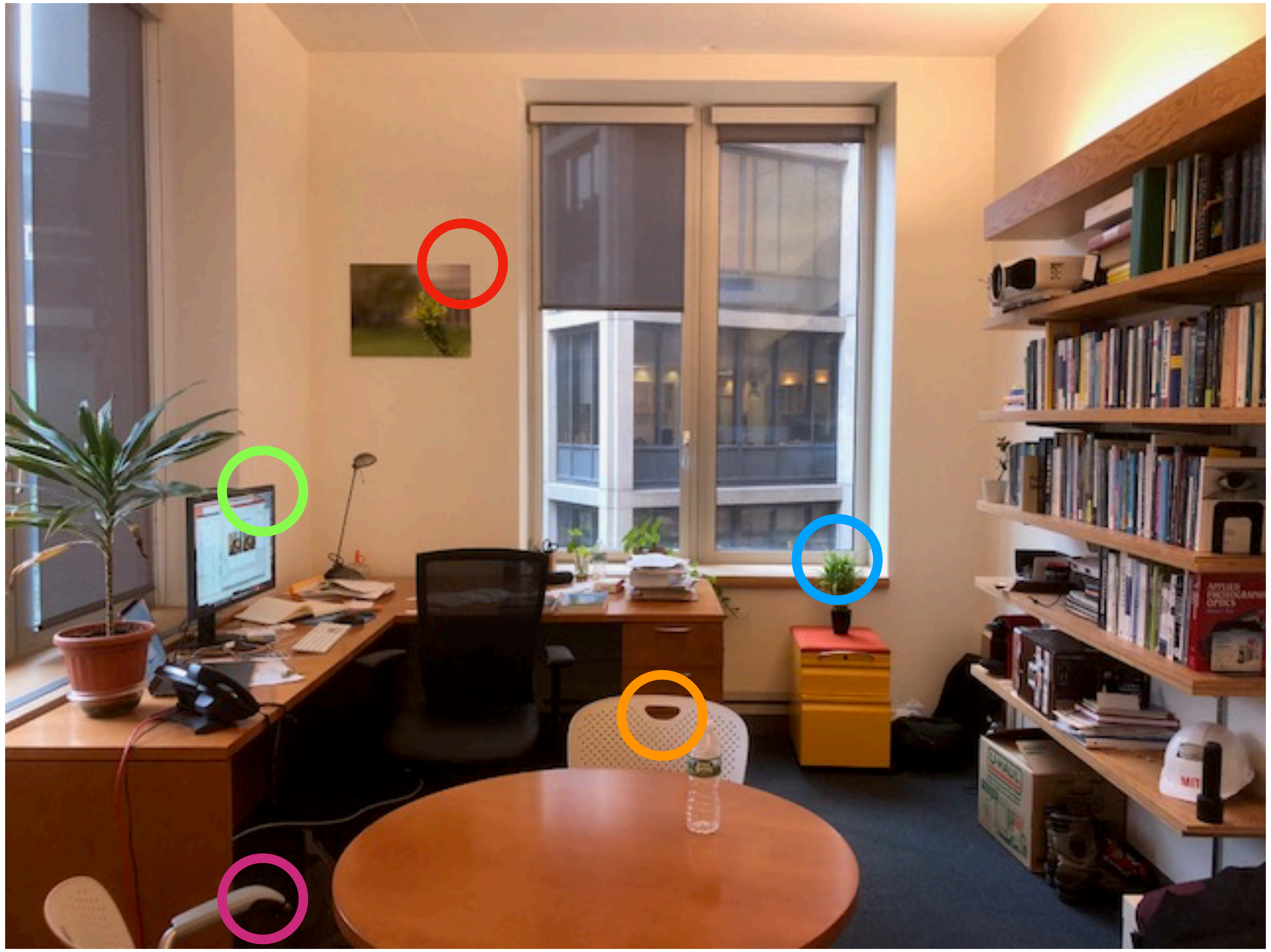
Epipolar Lines: The Hacky Way

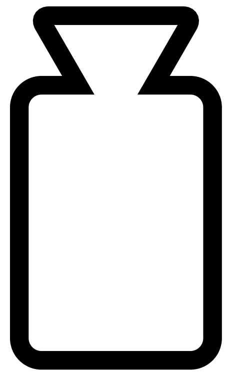
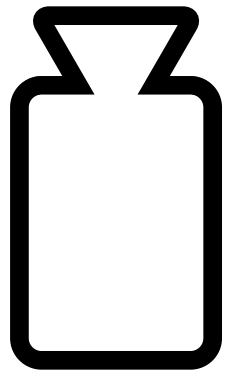
$$\mathbf{X} = \mathbf{R}_1^{C2W}\mathbf{K}_1^{-1}\tilde{\mathbf{x}}_1 + \mathbf{O}_1$$



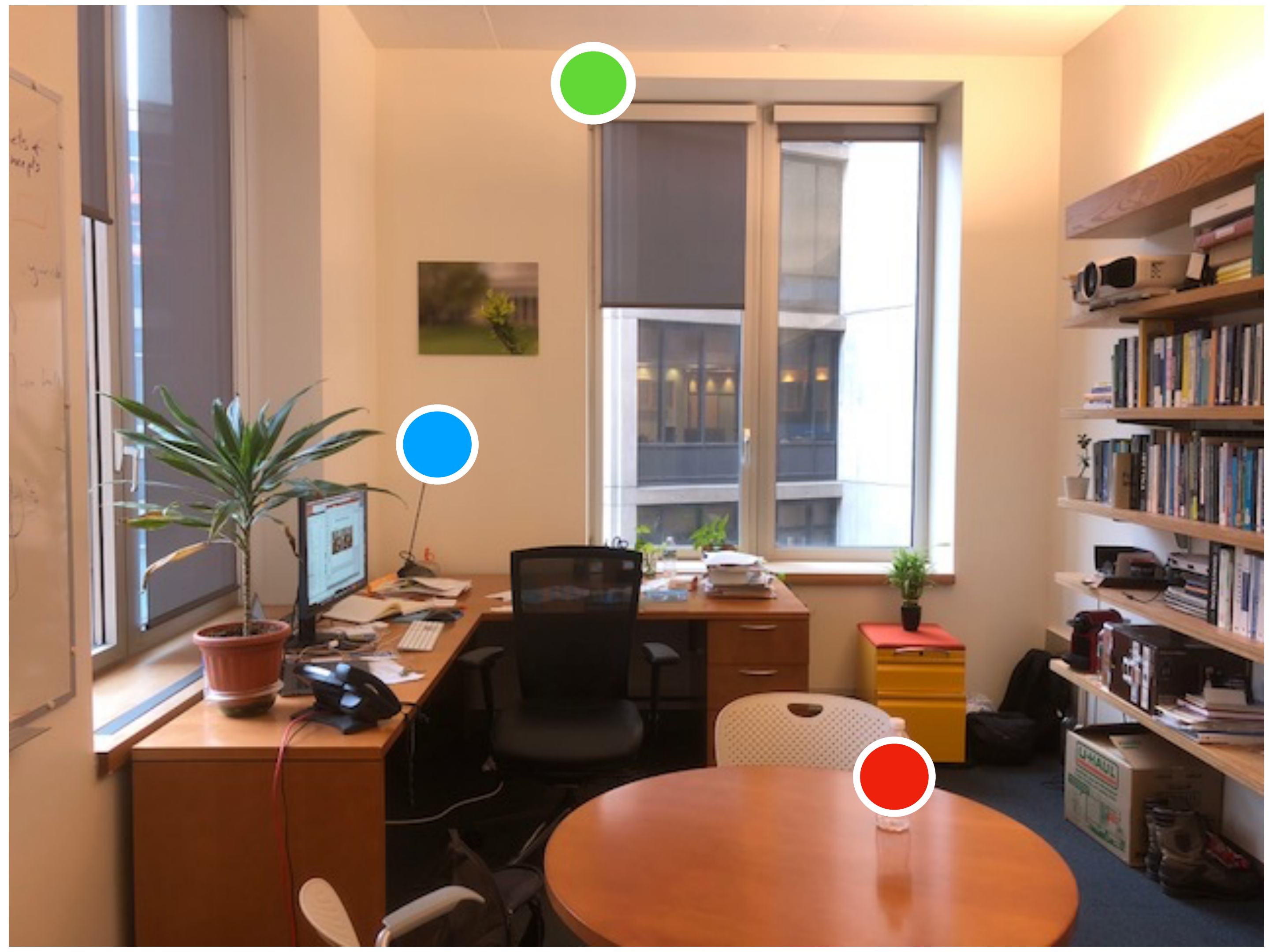
This always works ;)

Adapted from: CMU 16-385 (Yannis, Kris)

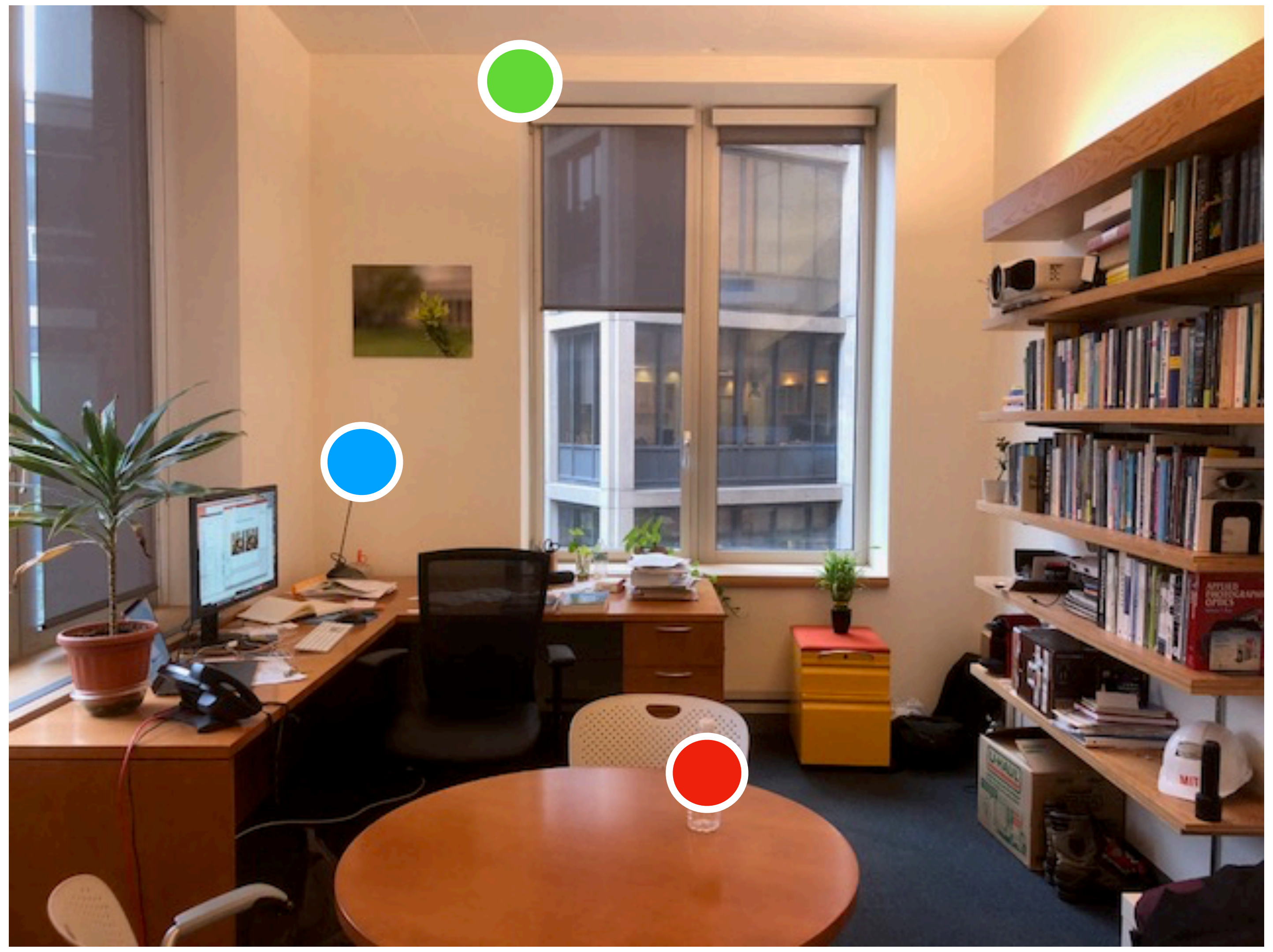
 

 What if $\mathbf{P}_1, \mathbf{P}_2$ *aren't* known? 

Finding correspondences



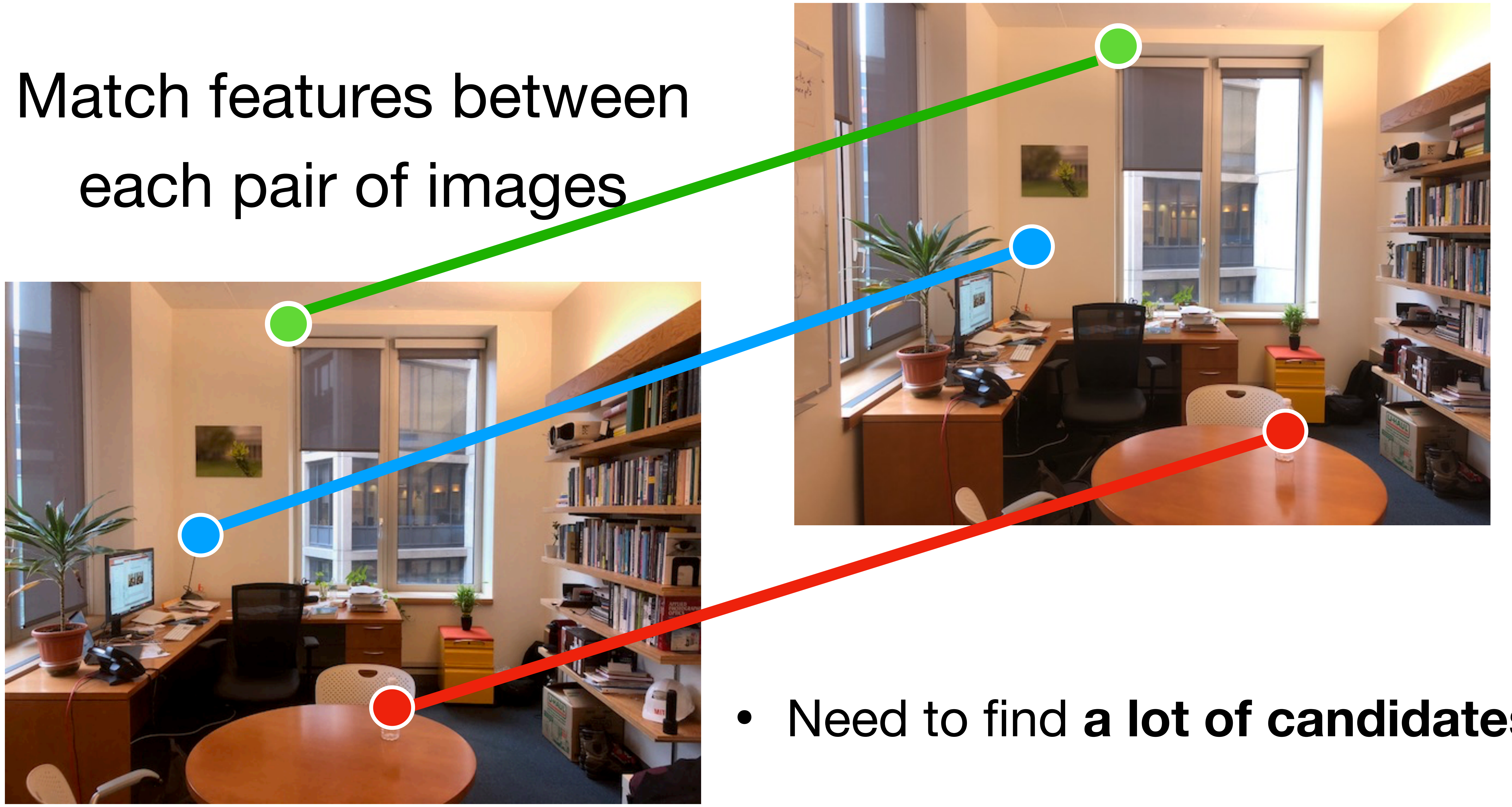
Match features between each pair of images



- 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

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

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

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

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

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

The Eight-Point Algorithm

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

$$\mathbf{W}\mathbf{f} = 0$$

with $\mathbf{W} \in \mathbb{R}^{8\times9}$, i.e., 8 correspondences stacked on top of each other

The Eight-Point Algorithm

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

$$\mathbf{W}\mathbf{f} = 0$$

with $\mathbf{W} \in \mathbb{R}^{8 \times 9}$, 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

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

$$\mathbf{Wf} = 0$$

with $\mathbf{W} \in \mathbb{R}^{8\times9}$, 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.

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

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.

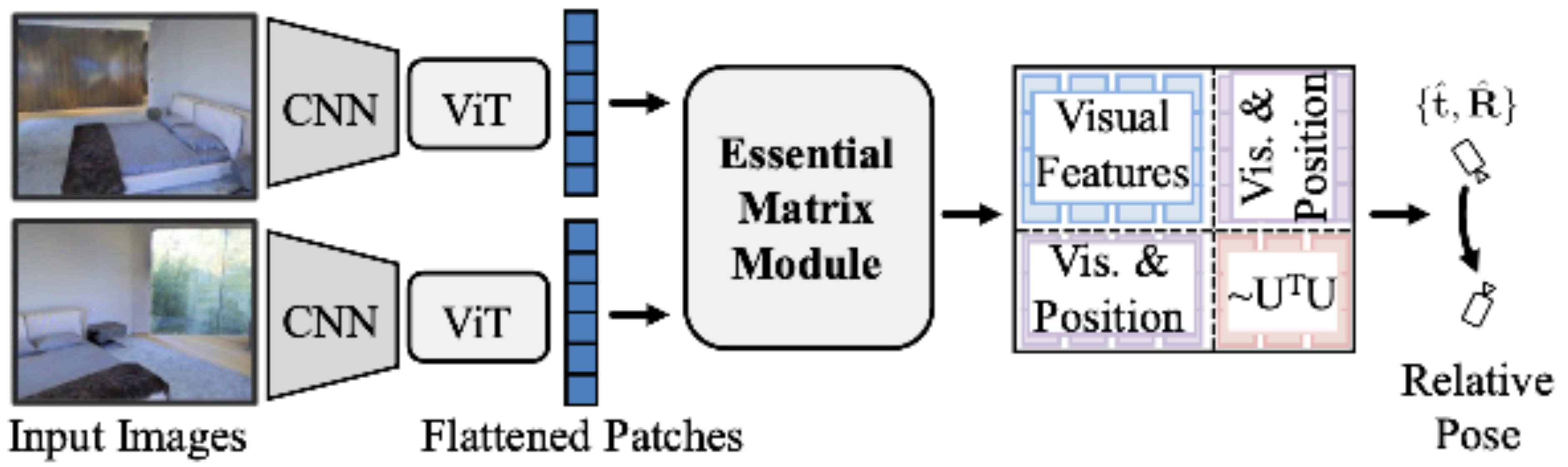


Figure 1. We propose three small modifications to a ViT via the Essential Matrix Module, enabling computations similar to the Eight-Point algorithm. The resulting mix of visual and positional features is a good inductive bias for pose estimation.

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.*

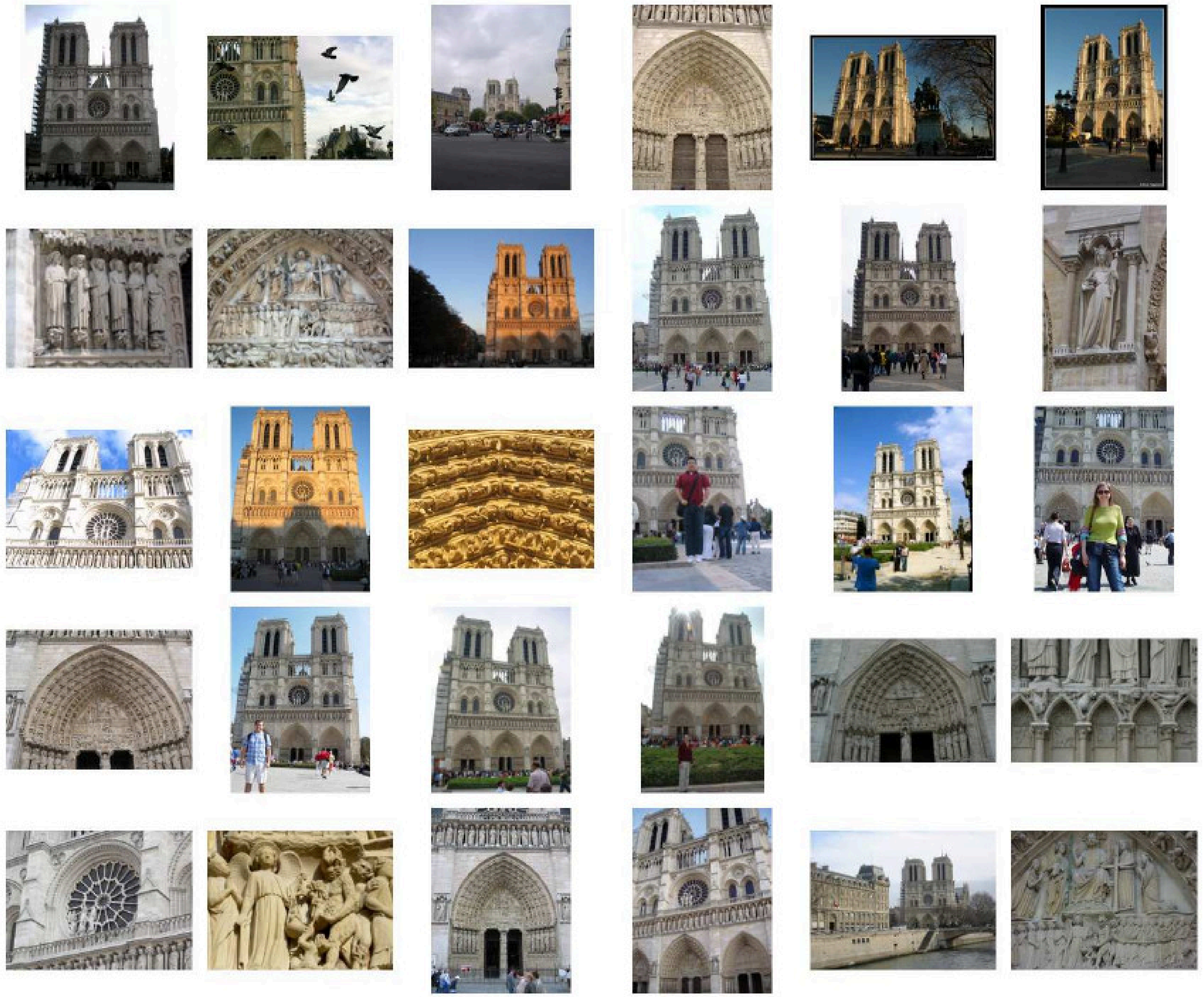
No Time

-
-
-

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**

Snavely, Seitz and Szeliski: Photo tourism: exploring photo collections in 3D. SIGGRAPH, 2006.

Adapted from: University of Tübingen: Computer Vision, Prof. Andreas Geiger

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.

Adapted from: University of Tübingen: Computer Vision, Prof. Andreas Geiger

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:

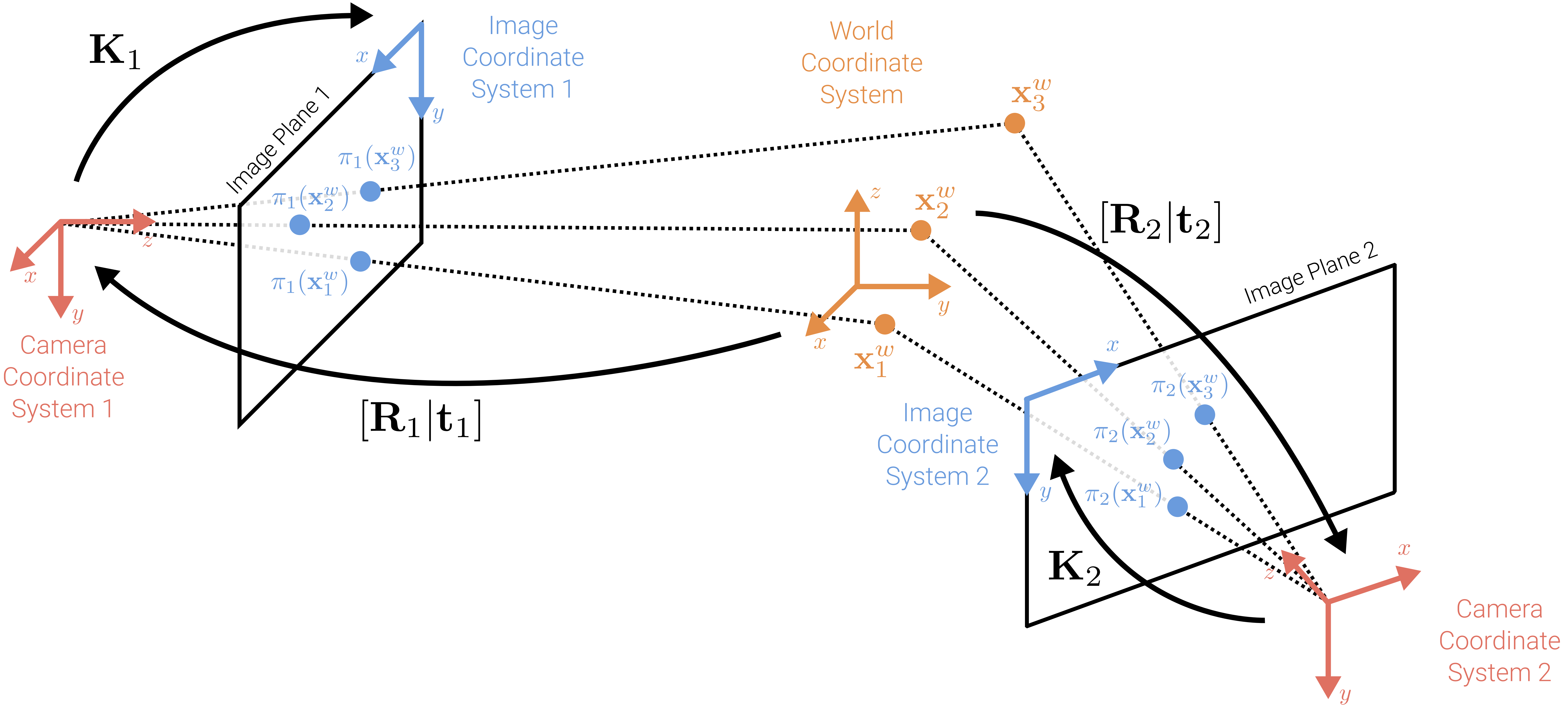
$$\Pi^*, \mathcal{X}_w^* = \underset{\Pi, \mathcal{X}_w}{\operatorname{argmin}} \sum_{i=1}^{N} \sum_{p=1}^{P} w_{ip} \left\| \mathbf{x}_{ip}^s - \pi_i(\mathbf{x}_p^w) \right\|_2^2$$

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.:

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

Adapted from: University of Tübingen: Computer Vision, Prof. Andreas Geiger

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.

Adapted from: University of Tübingen: Computer Vision, Prof. Andreas Geiger

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

Adapted from: University of Tübingen: Computer Vision, Prof. Andreas Geiger

Results and Applications

COLMAP SfM

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

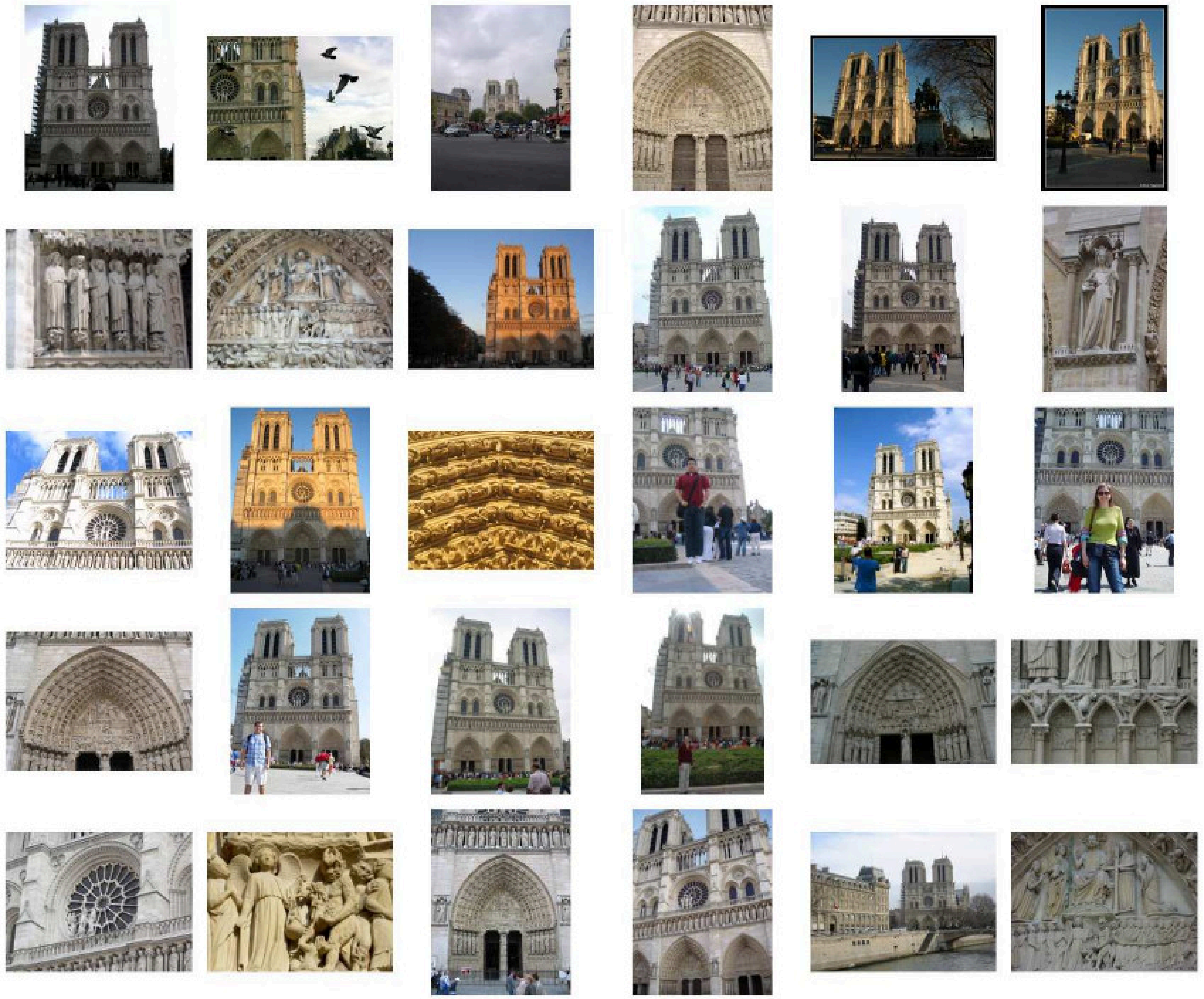
Schönberger and Frahm: Structure-from-Motion Revisited. CVPR, 2016.

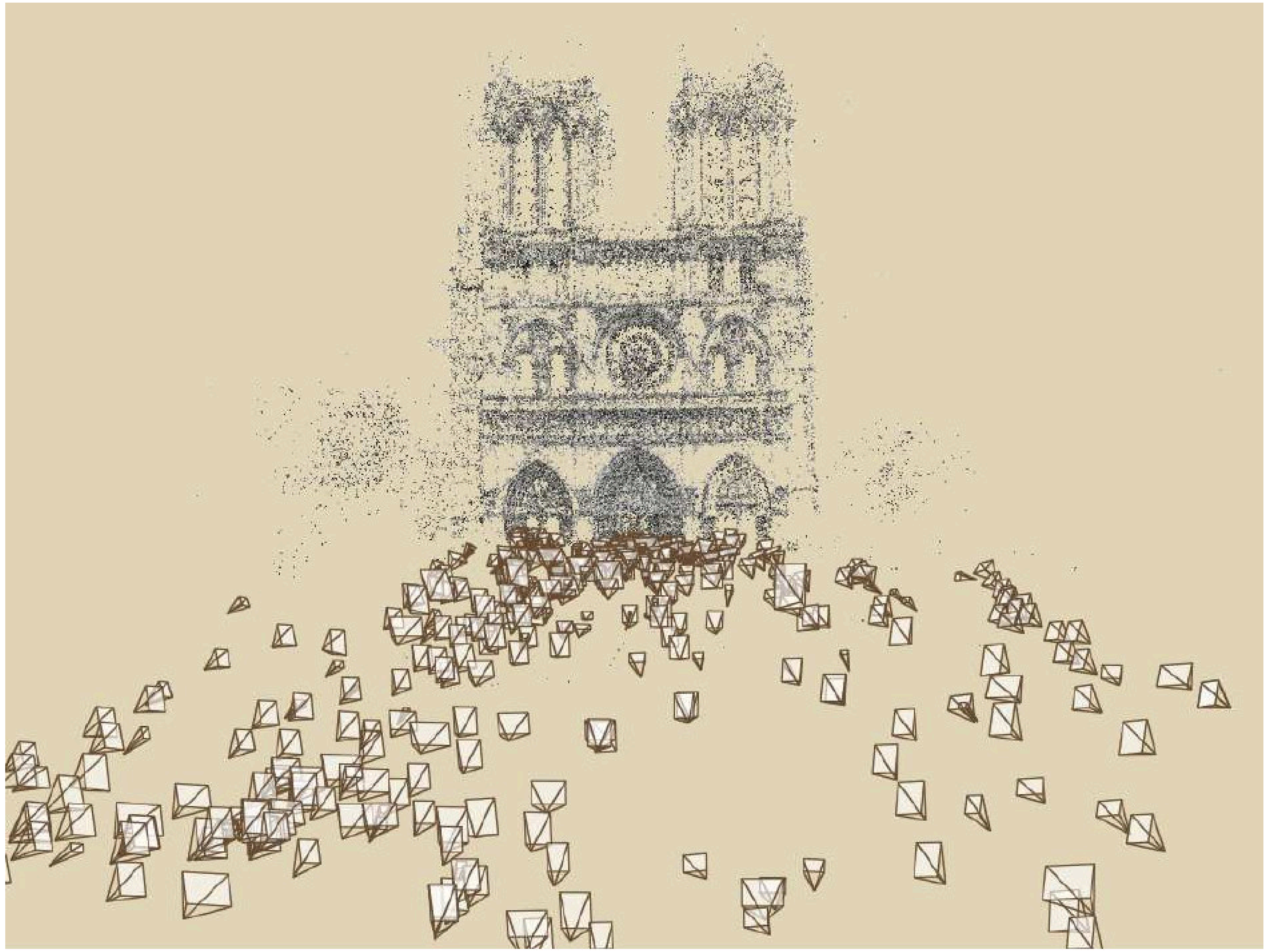
COLMAP MVS

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

Schönberger, Zheng, Frahm and Marc Pollefeys: Pixelwise View Selection for Unstructured Multi-View Stereo. ECCV, 2016.

Photo Tourism



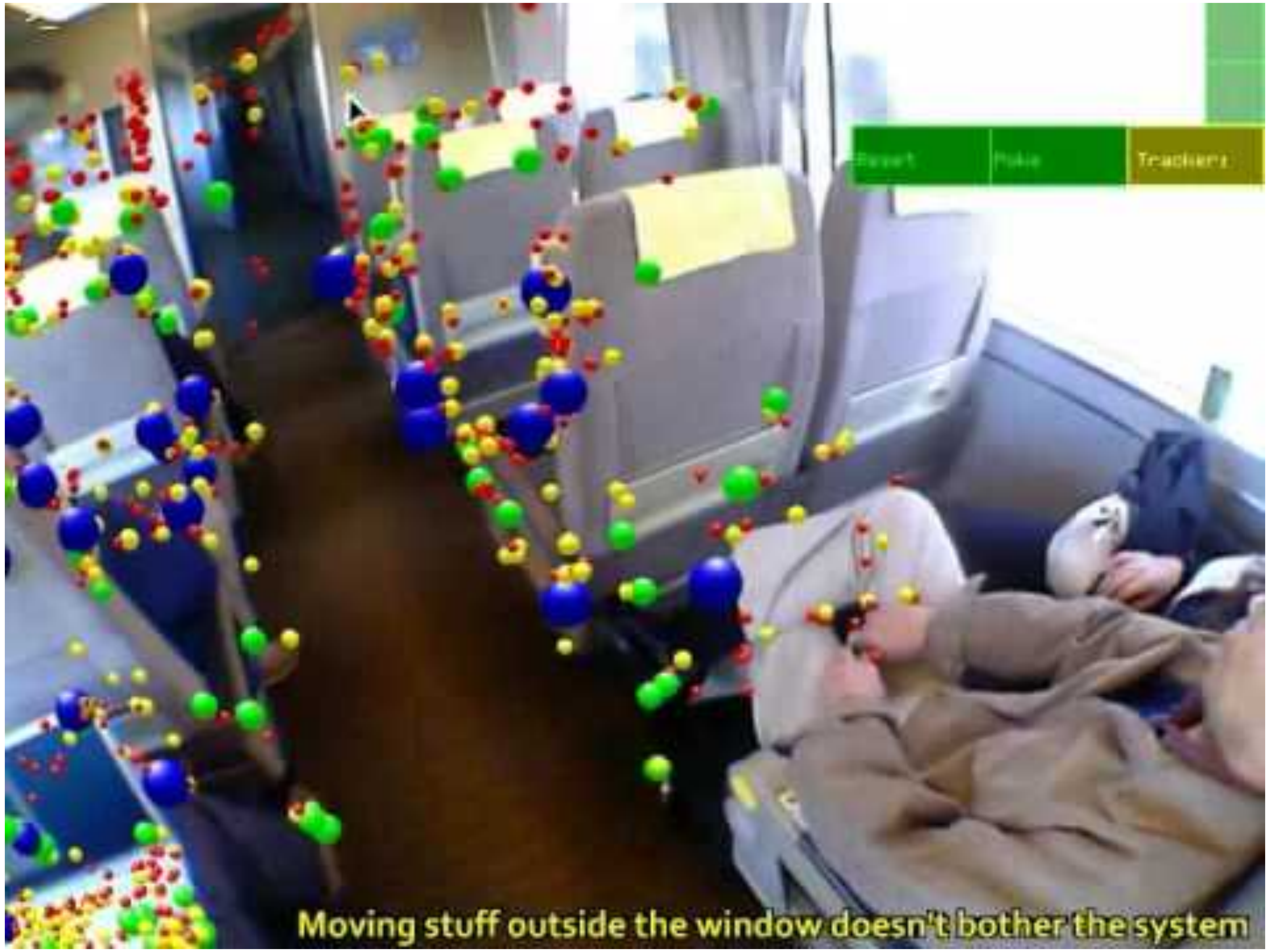


- **Photo Tourism / PhotoSynth** allows for exploring photo collections in 3D

Snavely, Seitz and Szeliski: Photo tourism: exploring photo collections in 3D. SIGGRAPH, 2006.

Adapted from: University of Tübingen: Computer Vision, Prof. Andreas Geiger

Parallel Tracking and Mapping (PTAM)



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

Klein and Murray: Parallel Tracking and Mapping for Small AR Workspaces. ISMAR, 2007.

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

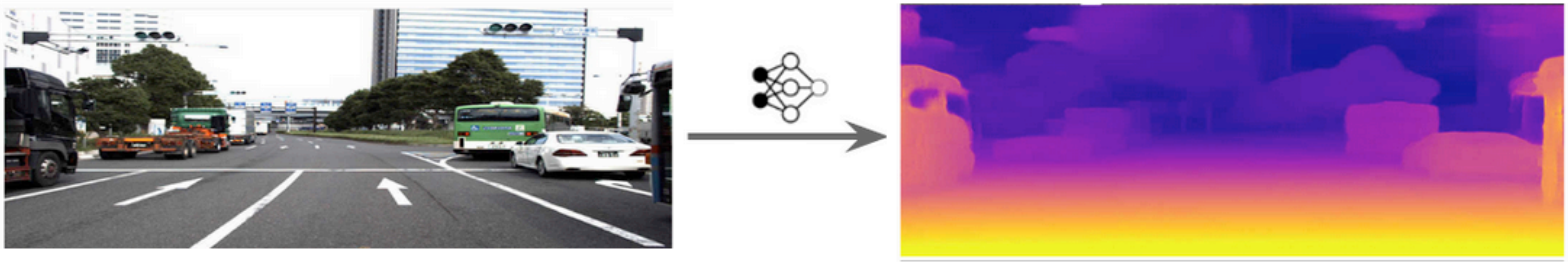
A way of *estimating camera poses, intrinsics, and extrinsic from correspondences.*

No Time

-
-
-

What has changed since Deep Learning?

Supervised **Monocular** Depth Estimation



https://medium.com/toyotaresearch/self-supervised-learning-in-depth-part-1-of-2-74825baaaa04

Unsupervised Depth and Ego-Motion from Video (Zhou et al. 2017)

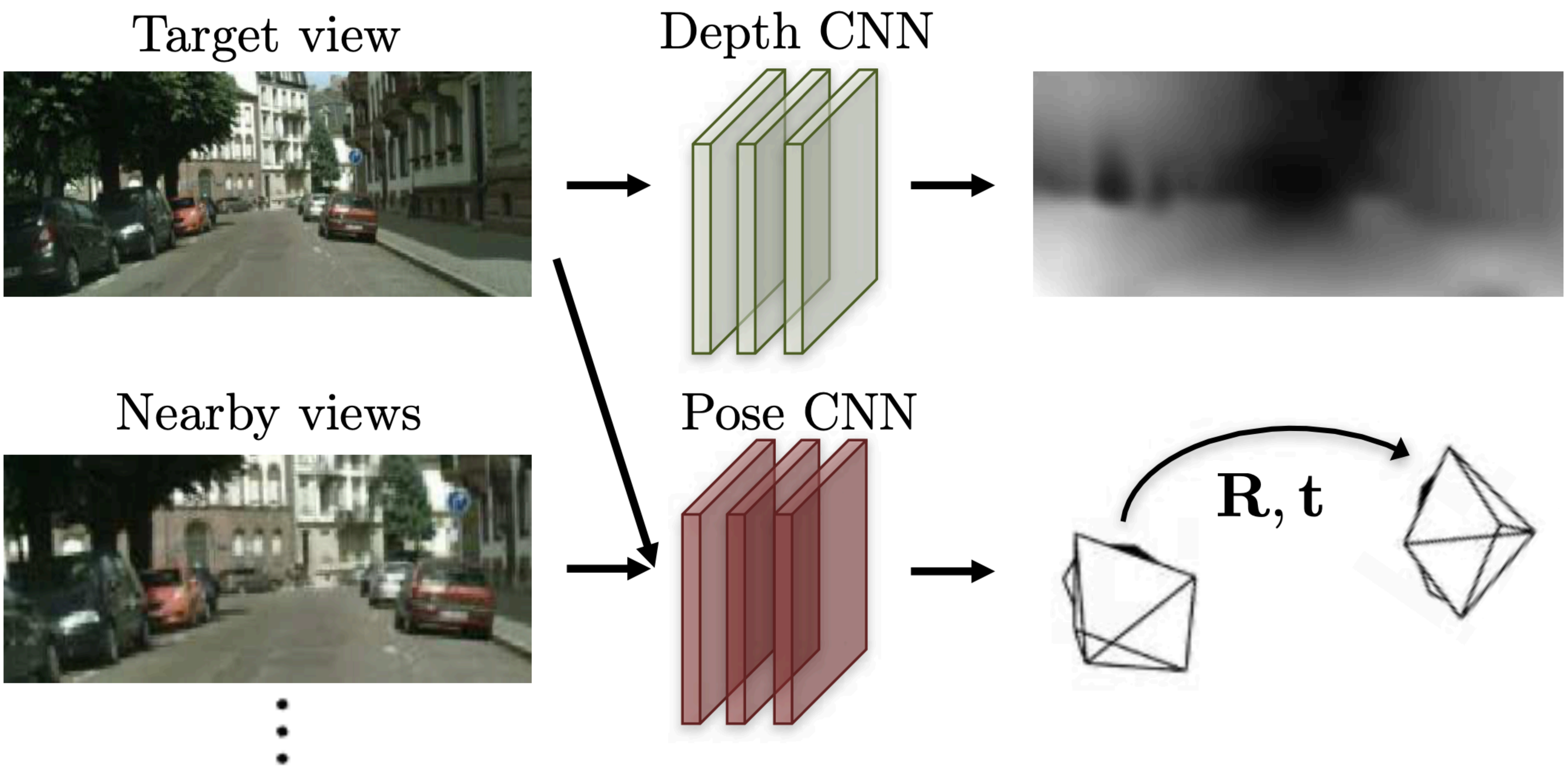
Frame at time t_1

. . .

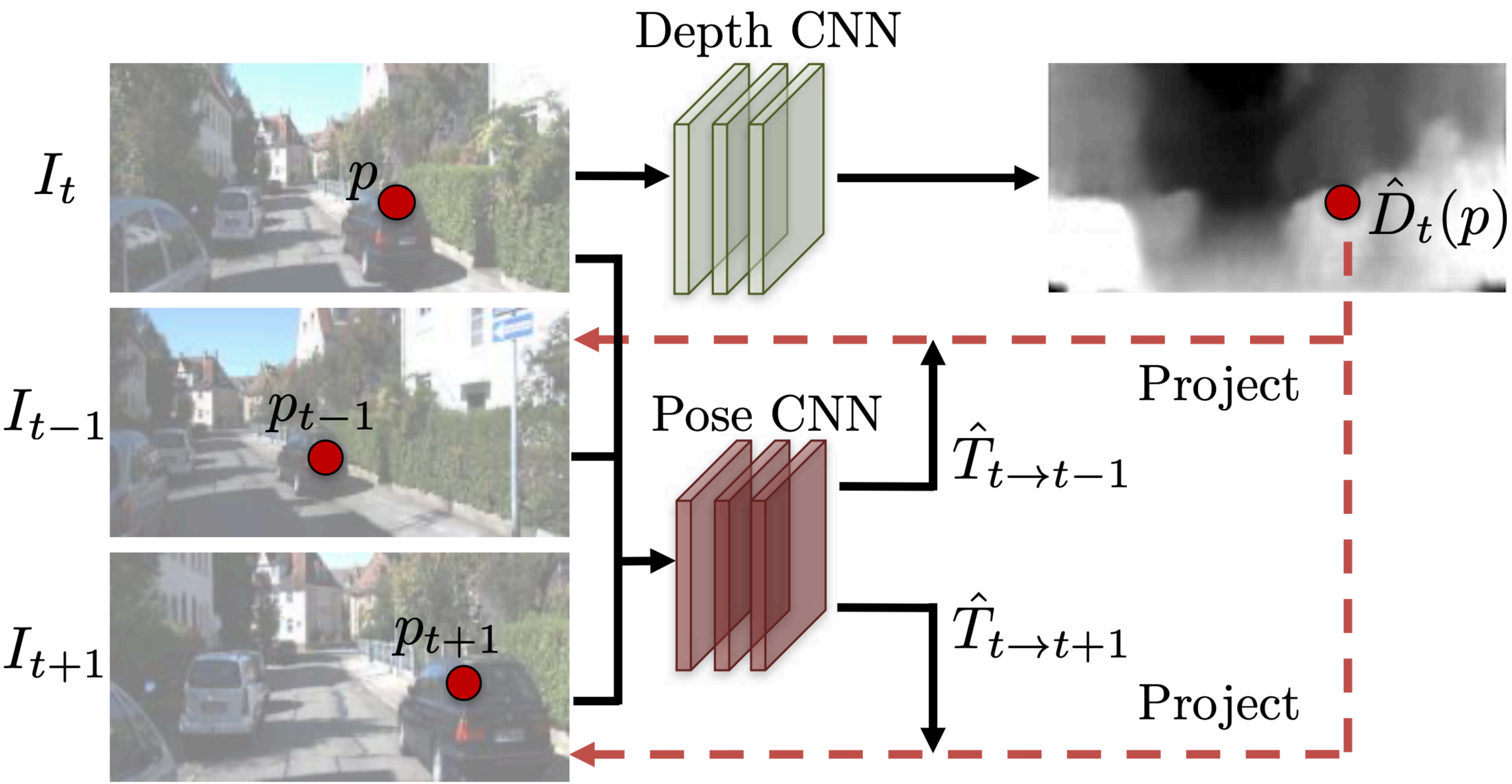
Frame at time t_2

Goal: Learn Depth and Ego-Motion (relative camera pose) just from video!

Unsupervised Depth and Ego-Motion from Video (Zhou et al. 2017)

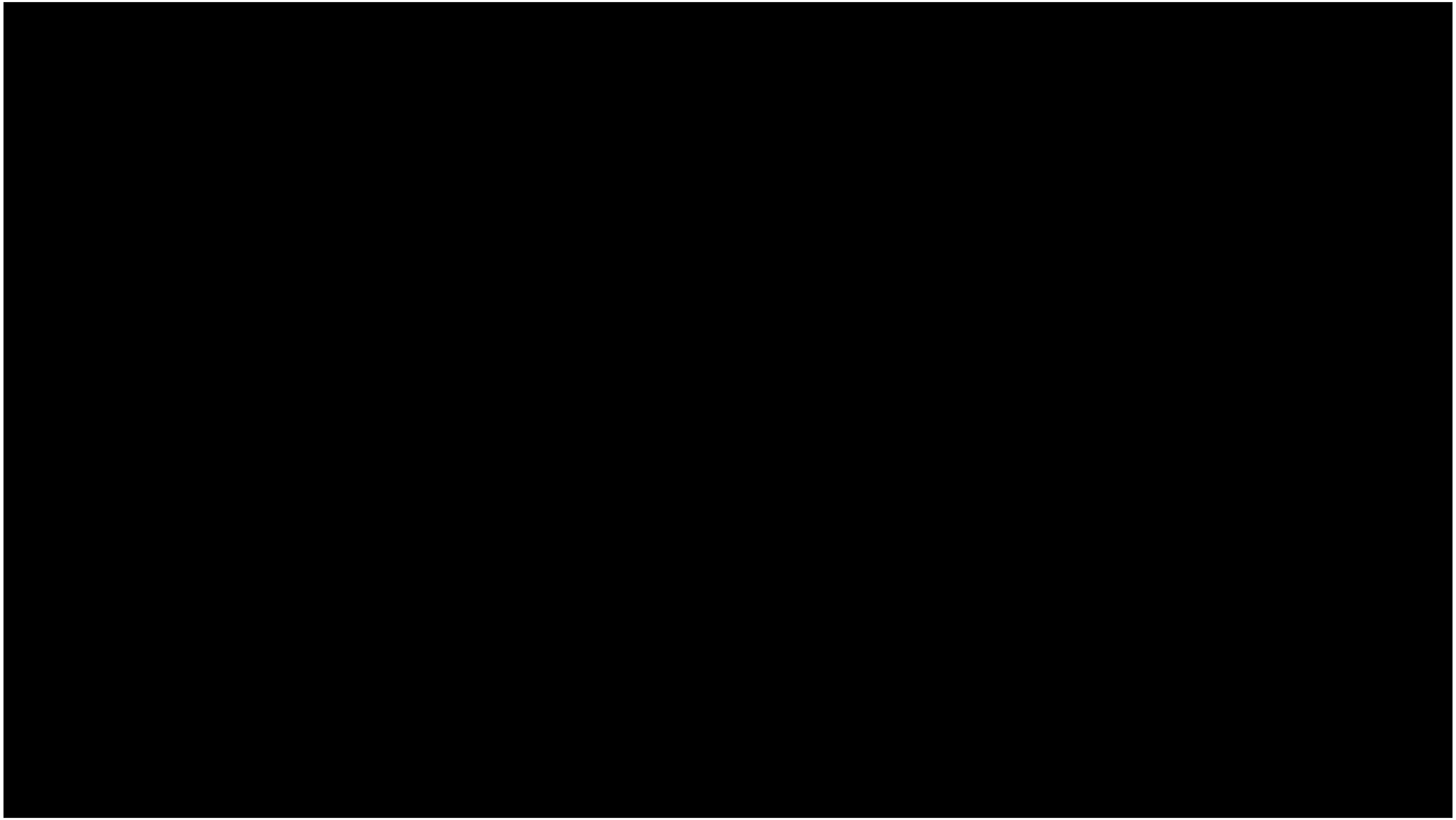


Unsupervised Depth and Ego-Motion from Video (Zhou et al. 2017)

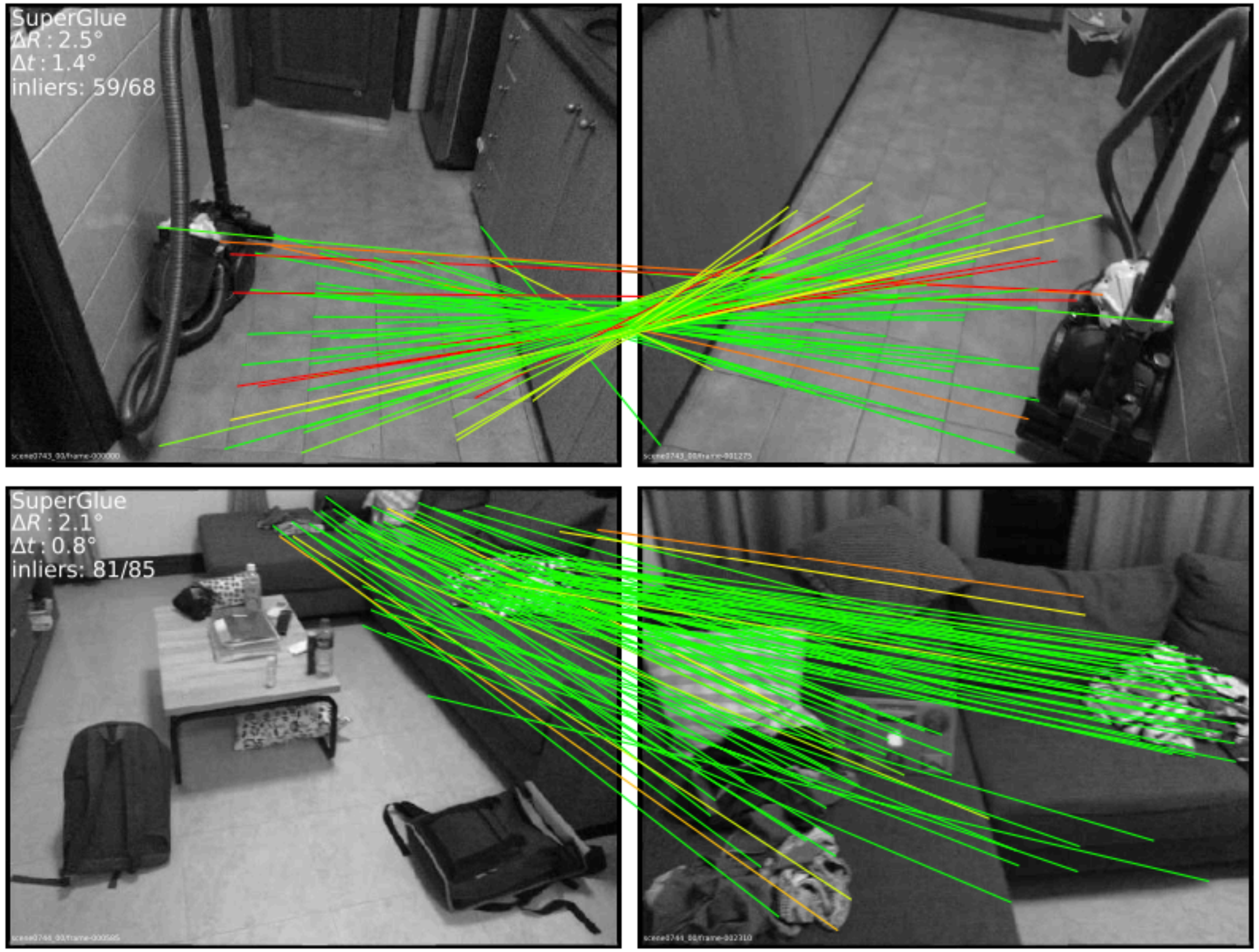


Self-supervised Learning of Depth and Pose from Video

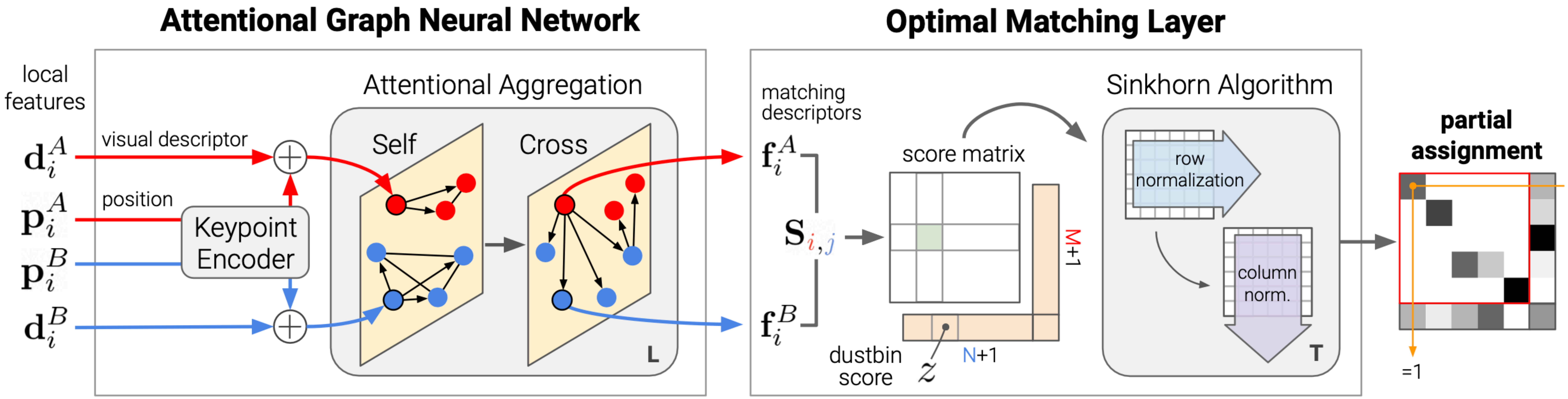
Guizilini et al. 2021



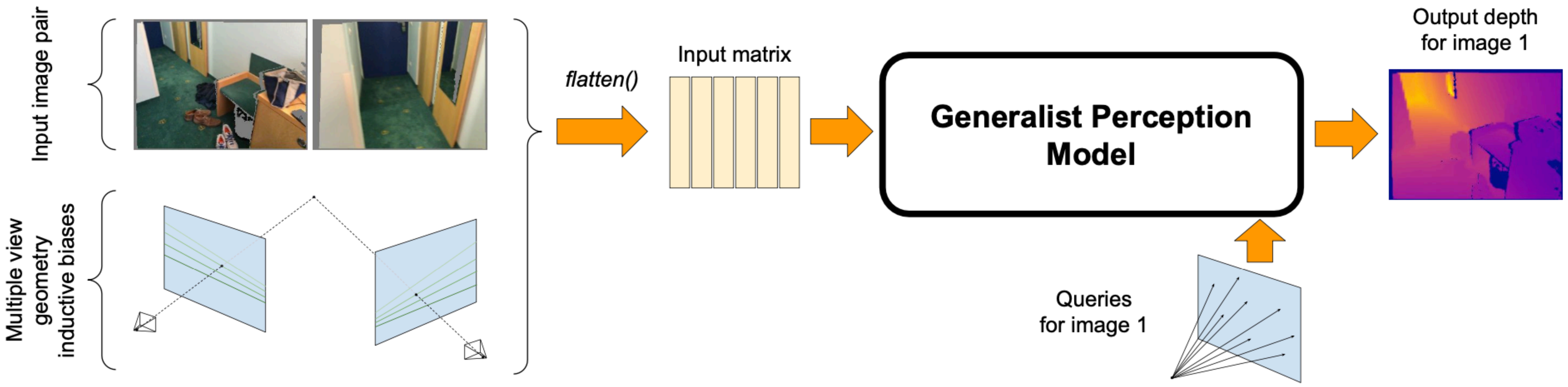
SuperGlue: Learning Feature Matching with Graph Neural Networks
Sarin et al. 2019



SuperGlue: Learning Feature Matching with Graph Neural Networks
Sarin et al. 2019

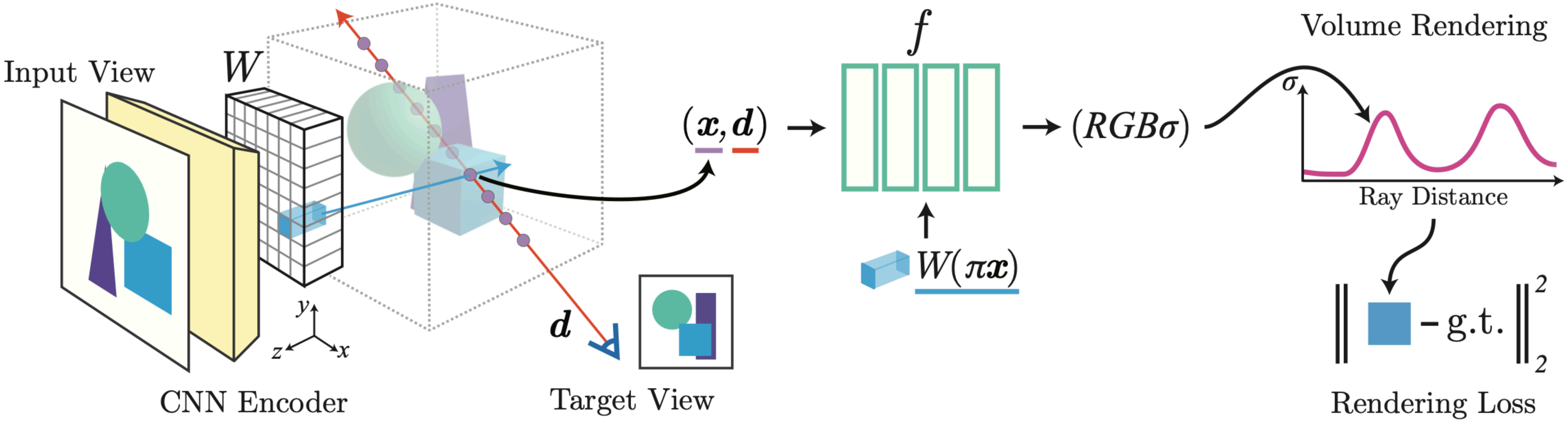


Input-Level Inductive Biases for 3D Reconstruction
Yifan et al. 2021

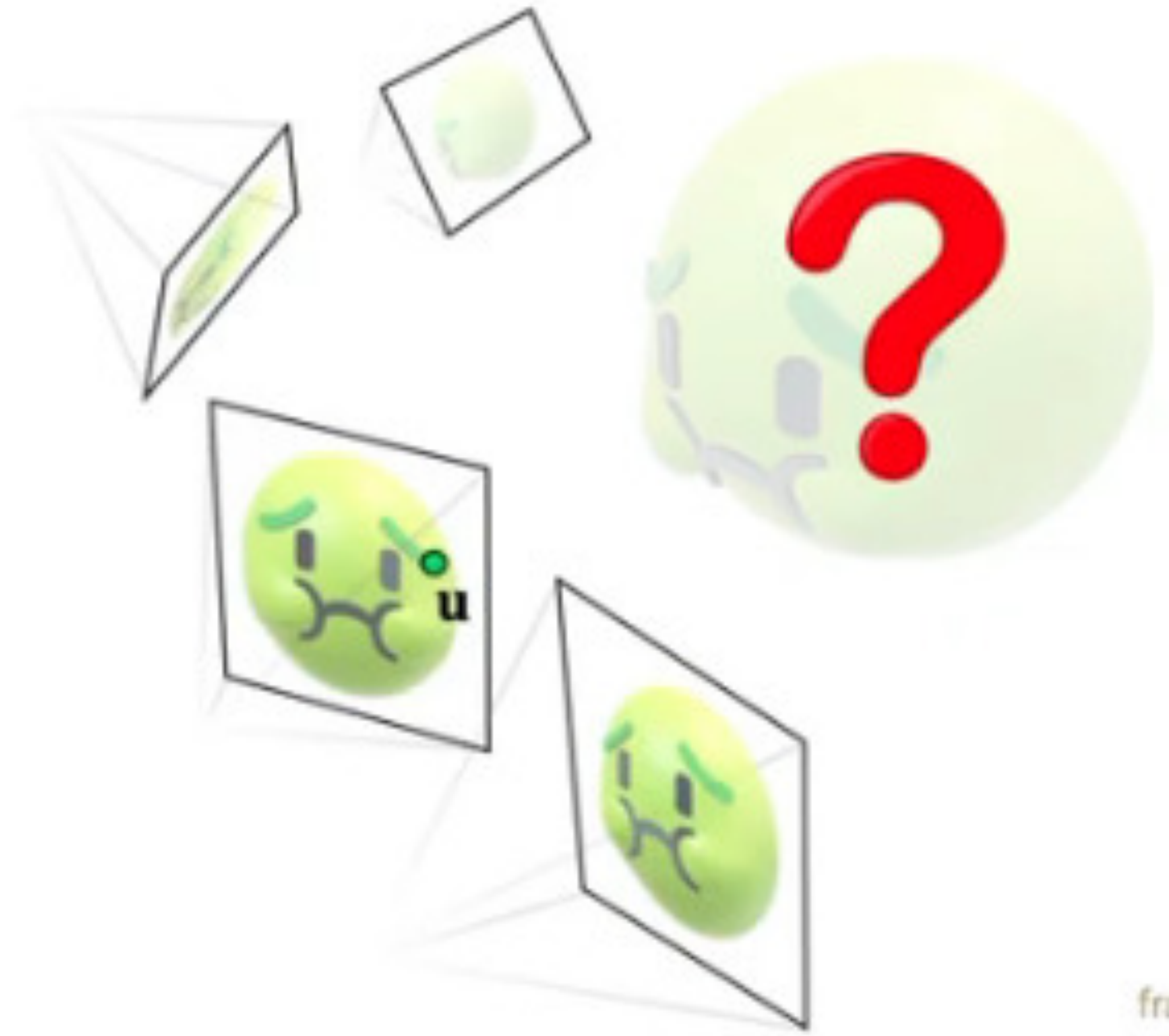


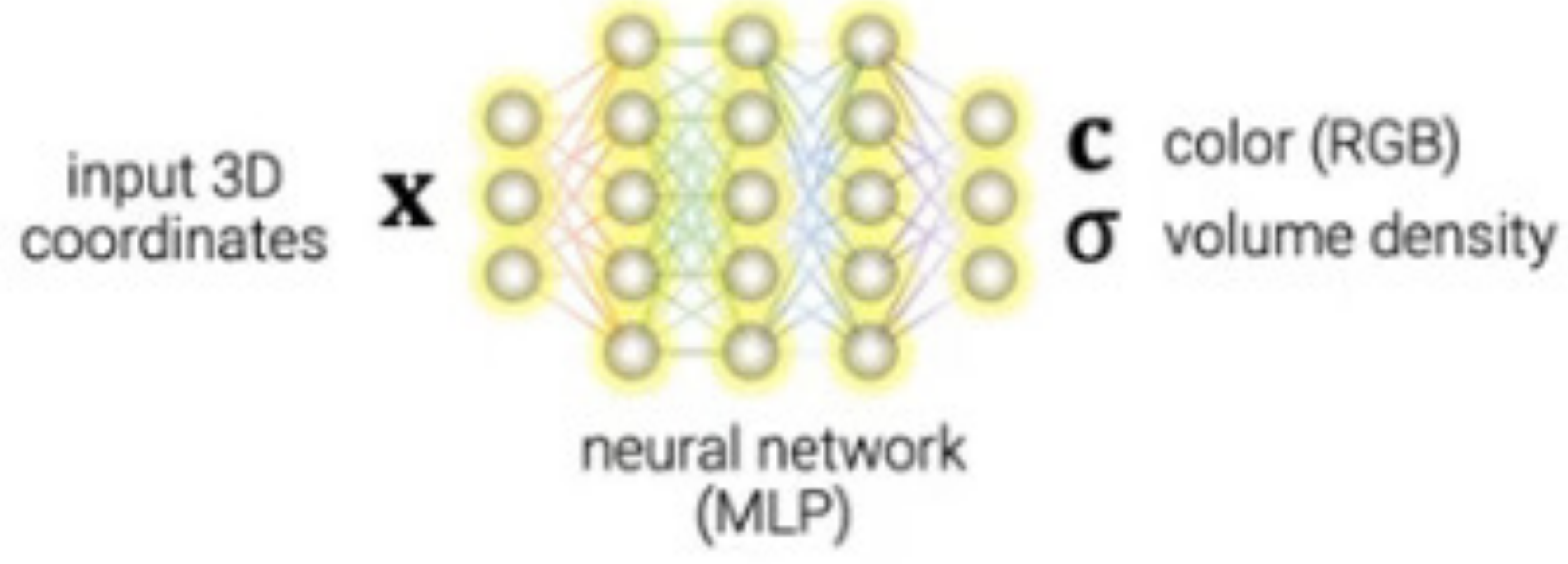
Input image pairs | Model predictions | Ground truth depth maps

PixelNeRF (Yu et al. 2020)



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





$$\min_{\mathbf{p}_1,\ldots,\mathbf{p}_M,\boldsymbol{\Theta}} \sum_{i=1}^{\overset{\text{frames}}{M}} \sum_{\mathbf{u}} \left\| \underbrace{\hat{\mathcal{I}}}_{\text{RGB (rendered)}}(\mathbf{u}; \underbrace{\mathbf{p}_i}_{\text{camera pose}}, \underbrace{\boldsymbol{\Theta}}_{\text{network params.}}) - \underbrace{\mathcal{I}_i(\mathbf{u})}_{\text{RGB}} \right\|_2^2$$

What if the **camera poses** are **imperfect** (or even **unknown**)?
Can we optimize the poses naïvely 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…

IMO we're missing the correct way to "learn" multi-view geometry in a self-supervised way. It should be possible: Build a model that watches video and learns to reconstruct both pose and a proper 3D scene representation!

Maybe one of you will get there :)

Summary

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

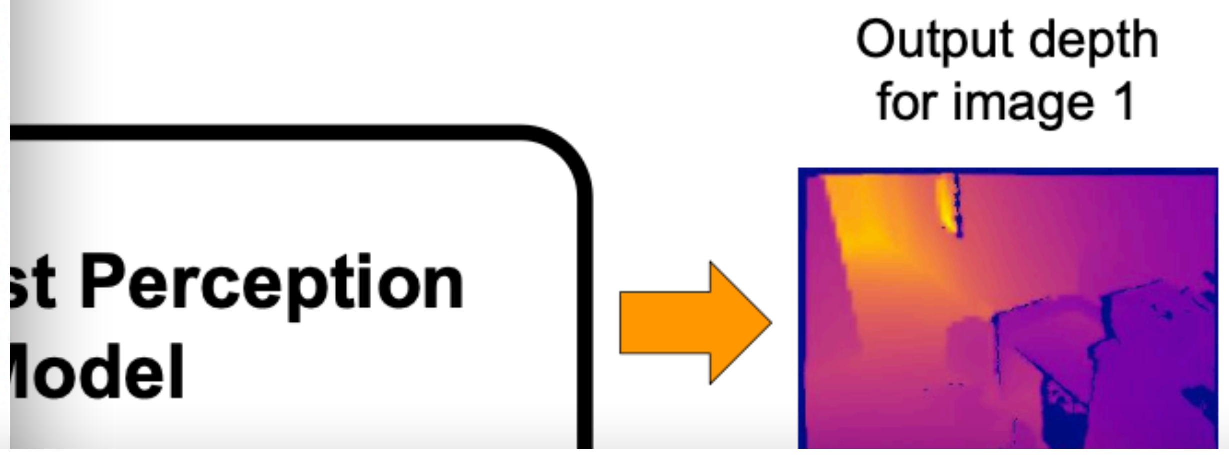
The 8-Point Algorithm as an Inductive Bias for Relative Pose Prediction by ViTs

22

Input-level Inductive Biases for 3D Reconstruction

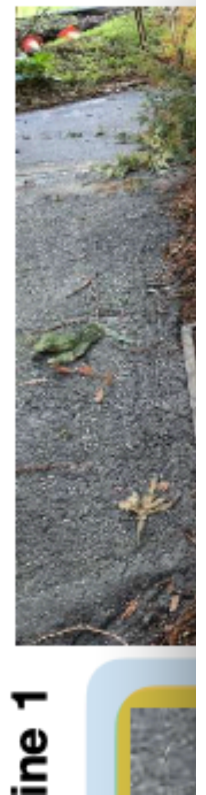
Wang Yifan[1*] Carl Doersch[2] Relja Arandjelović[2] João Carreira[2] Andrew Zisserman[2,3]

Science, University of Oxford



Generalizable Patch-Based Neural Rendering

Mohammed Suhail[1], Carlos Esteves[4], Leonid Sigal[1,2,3], and Ameesh Makadia[4]



BARF 🤮: Bundle-Adjusting Neural Radiance Fields

Chen-Hsuan Lin[1] Wei-Chiu Ma[2] Antonio Torralba[2] Simon Lucey[1,3]

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

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