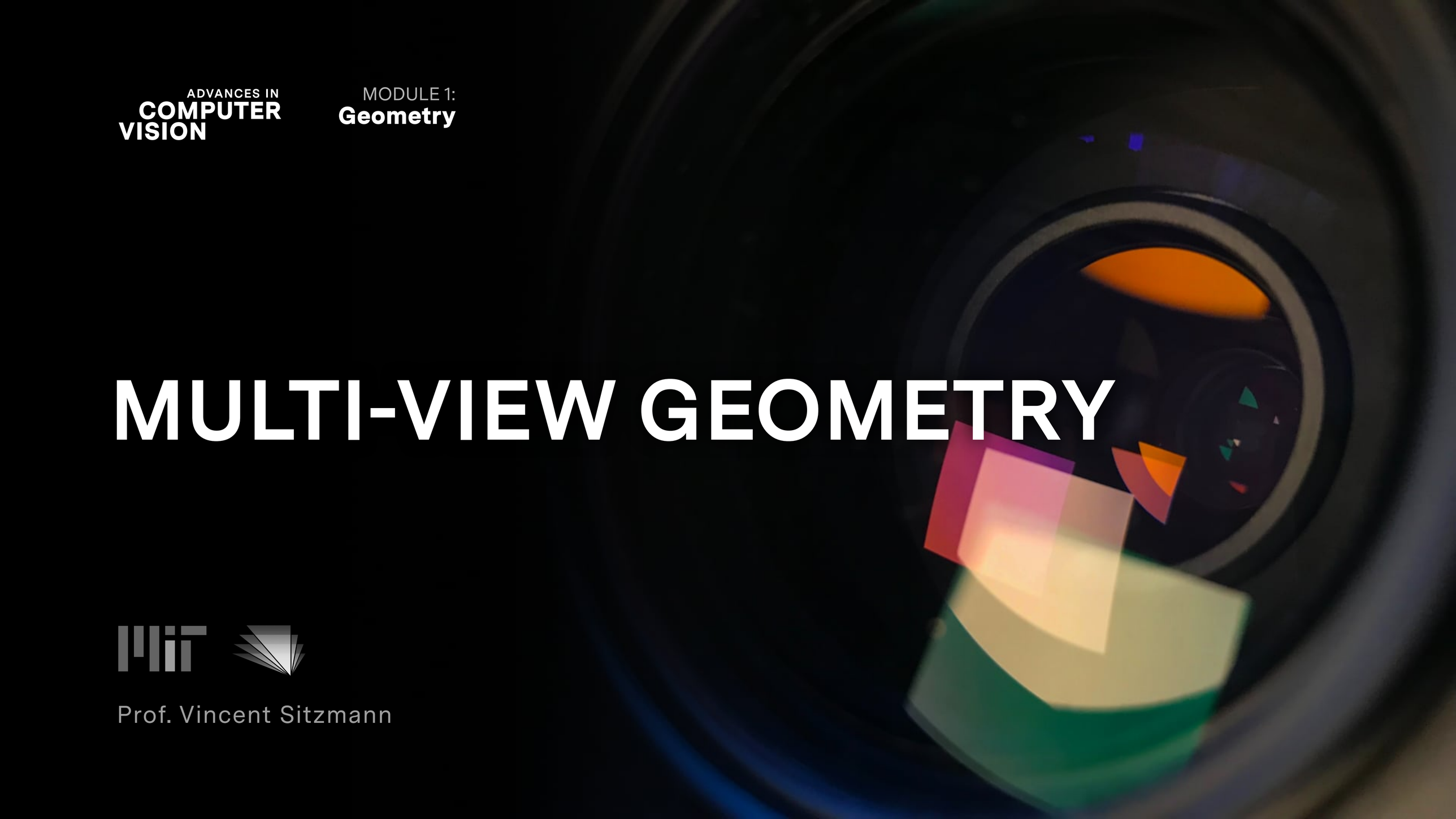
ADVANCES IN COMPUTER VISION

MODULE 1: Geometry

MULTI-VIEW GEOMETRY



Prof. Vincent Sitzmann

New Office Hour Schedule!

New Office Hour Schedule!

Going forward, the two TAs responsible for making the week's problem set will host multiple office hours throughout the week plus a few extra office hours during the late days.

New Office Hour Schedule!

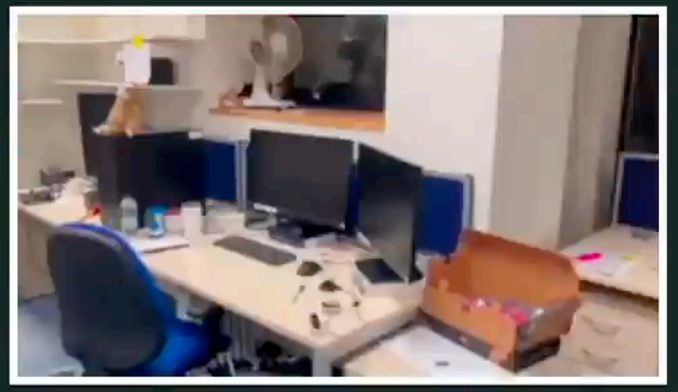
Motivation: TAs who wrote the problem set can give the best feedback to students struggling with nuanced aspects of the homework. There are also usually two office hours per day, so hopefully there are times that work for everyone!

New Office Hour Schedule!

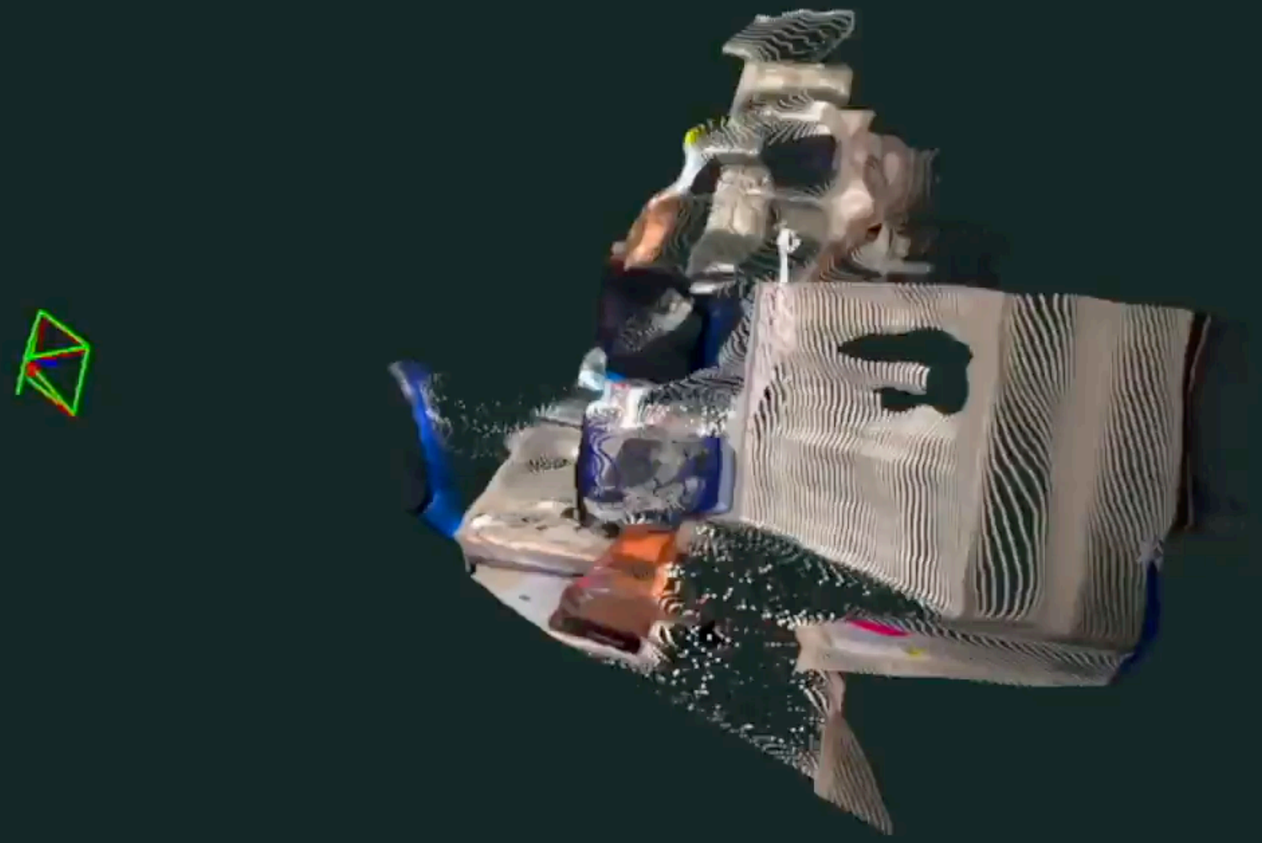
New schedule: https://piazza.com/class/m5fhhrt0jzc6hx/post/30

Uncalibrated RGB
8x Playback

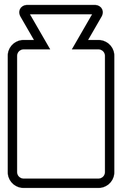
Uncalibrated RGB
8x Playback



Uncalibrated RGB
8x Playback

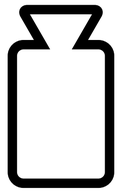


Murai, Dexheimer et al., MASt3R-SLAM: Real-Time Dense SLAM with 3D Reconstruction Priors, CVPR 2025

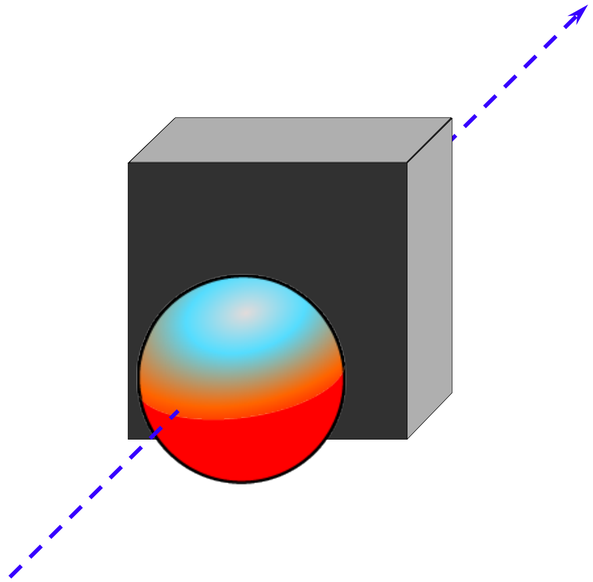


Known $\mathbf{P}_1, \mathbf{P}_2$!

What's 3D location of point?



Triangulation - the infinitesimal perspective

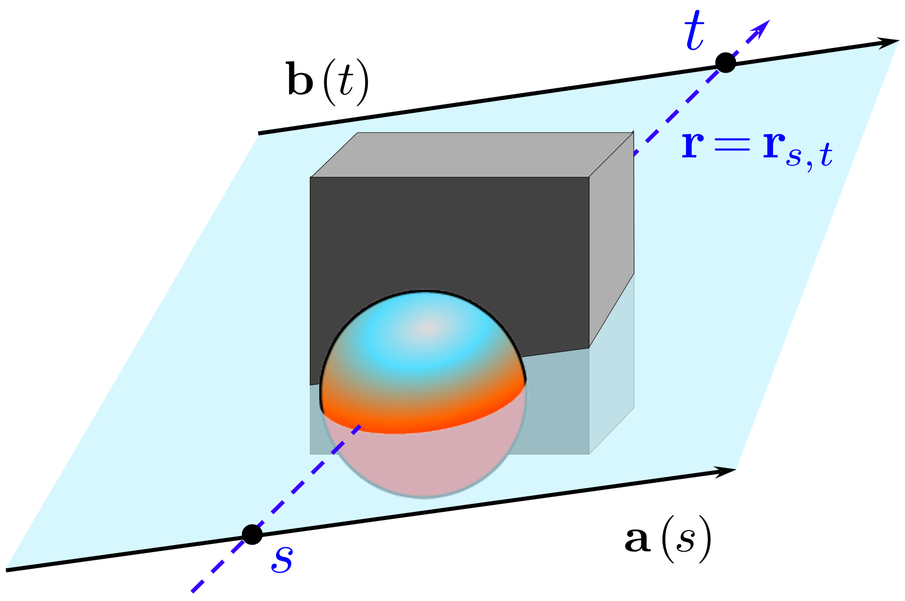


Light Field model of a 3D Scene:
Function that assigns rays to colors

It’s a five-dimensional function…
We can’t visualize that.

We will look at a 2D subset of the rays.

Triangulation - the infinitesimal perspective

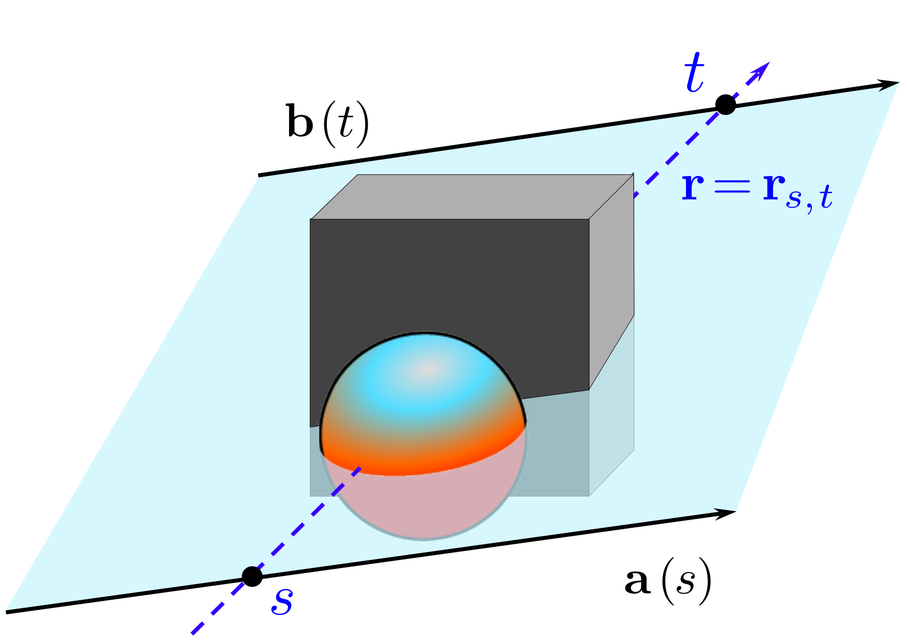


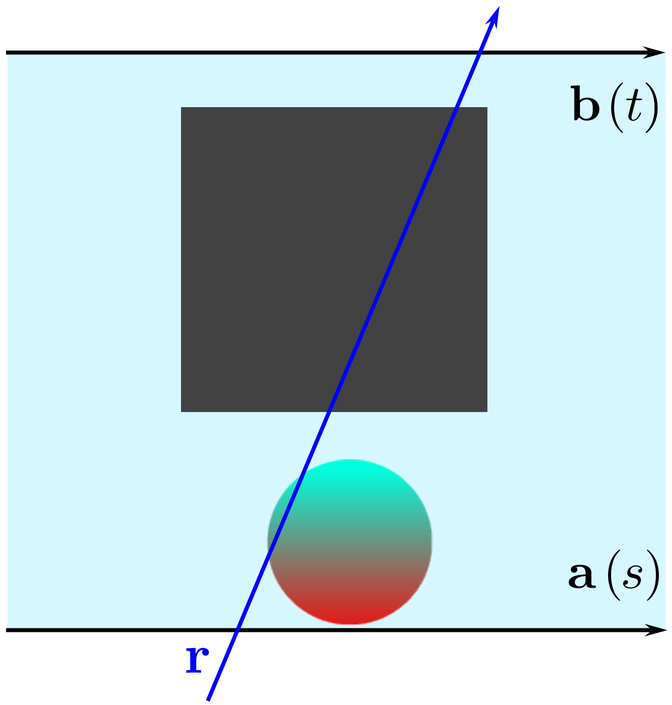
Pick a random 3D plane that slices 3D scene.

We will think about all the rays that lie in this plane.

We will identify them by their intersection with two parallel lines that lie in this plane.

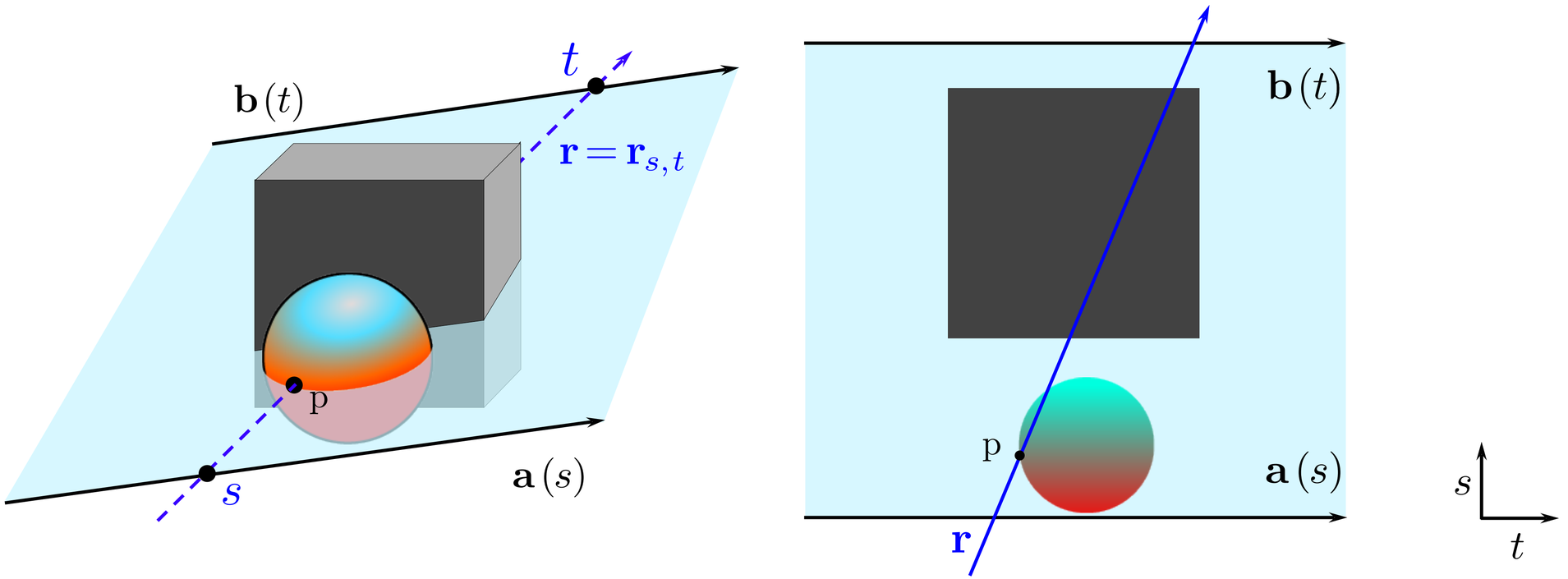
Triangulation - the infinitesimal perspective





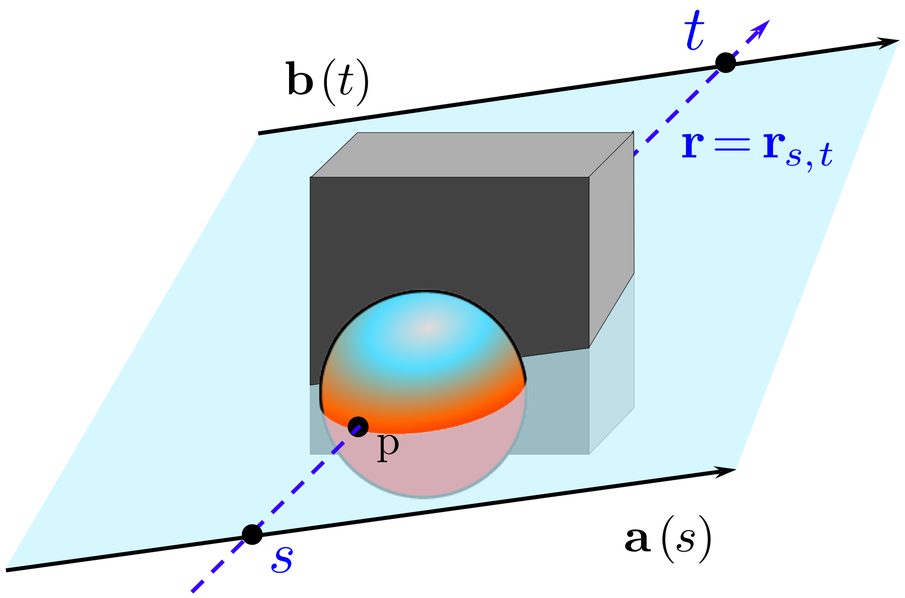
That is, each ray has coordinates a and b which are scalars that say where on the line the ray intersects with the line.

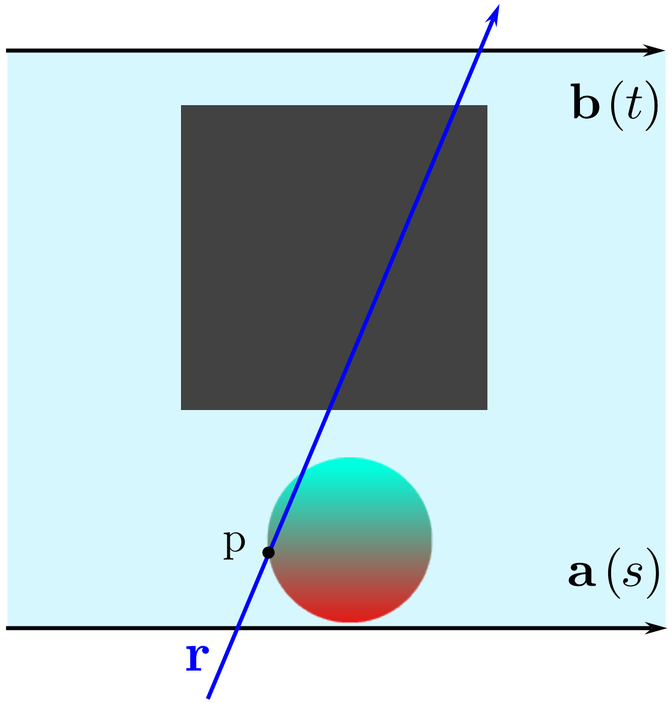
Triangulation - the infinitesimal perspective

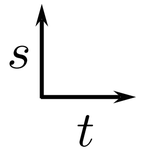


The color of each ray will be the color of the 3D point where it first intersects the 3D scene.

Triangulation - the infinitesimal perspective

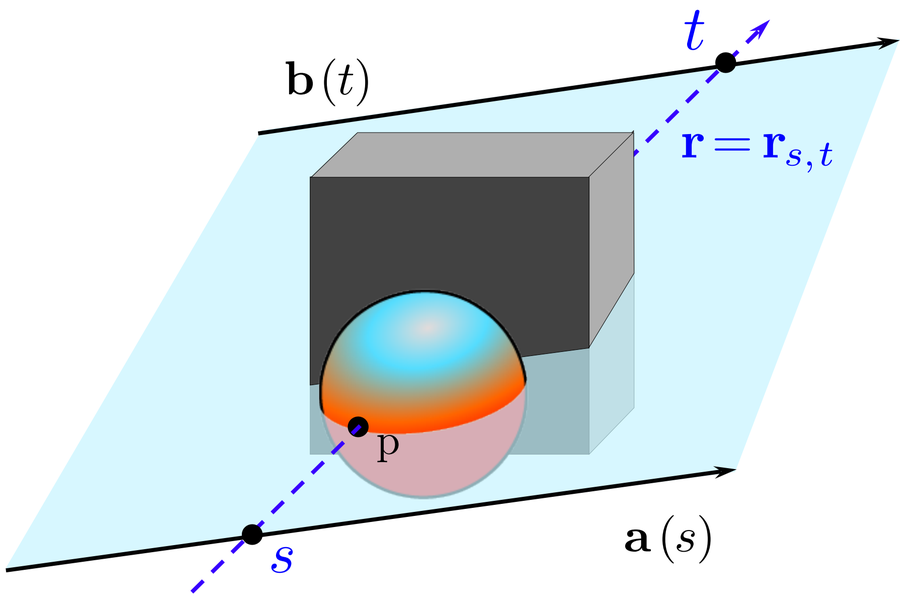


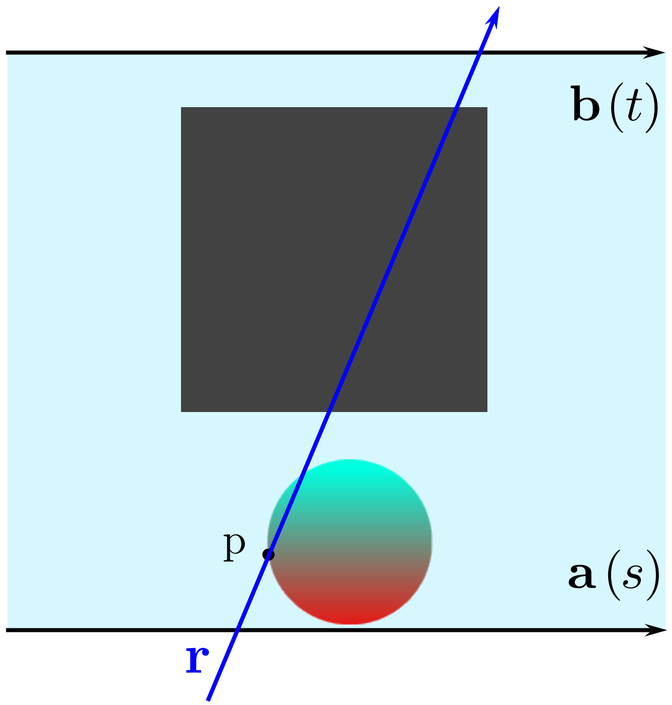


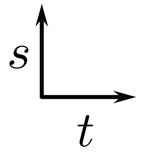


This ray has a teal color.

Triangulation - the infinitesimal perspective

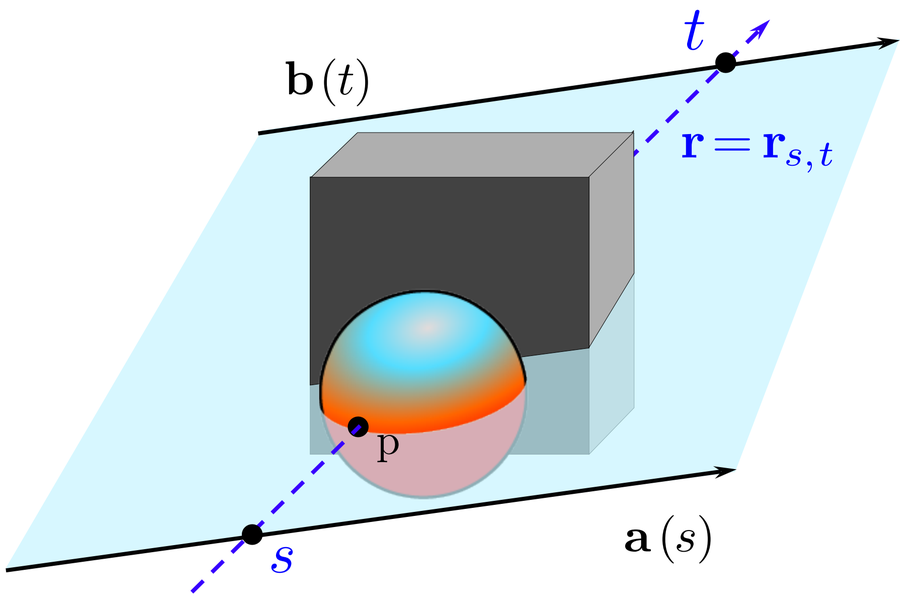


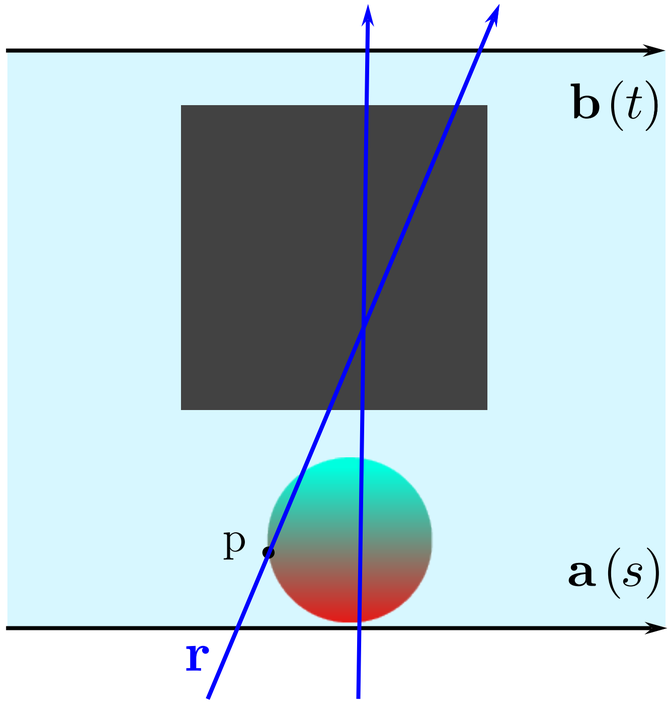


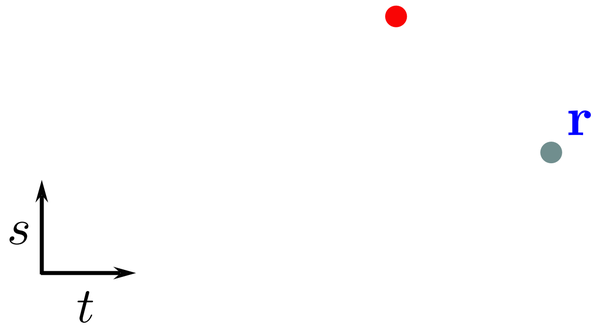


Some other rays

Triangulation - the infinitesimal perspective

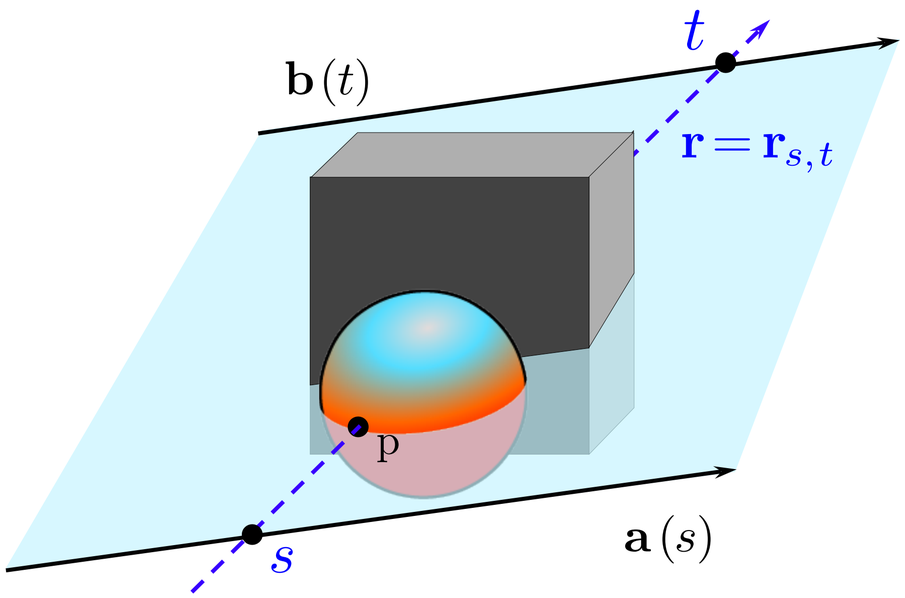


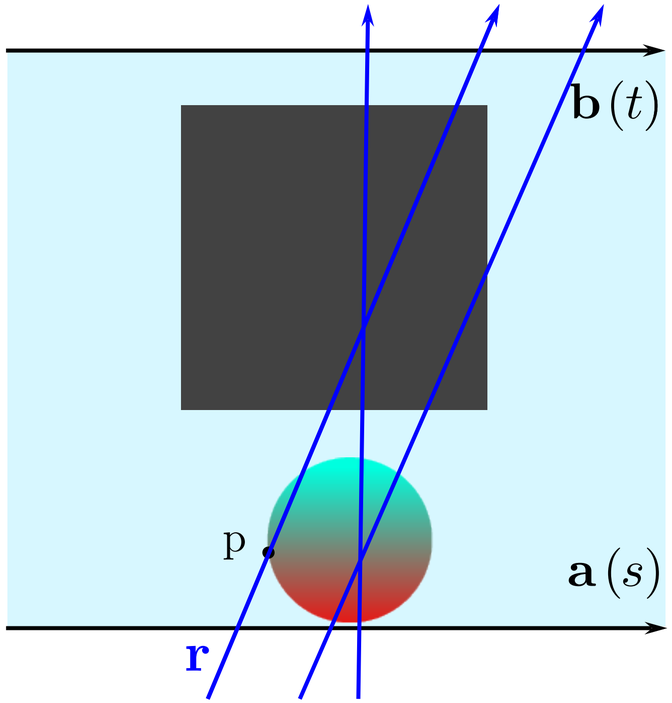


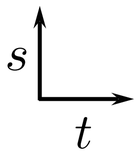


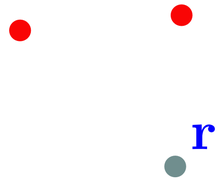
Some other rays

Triangulation - the infinitesimal perspective



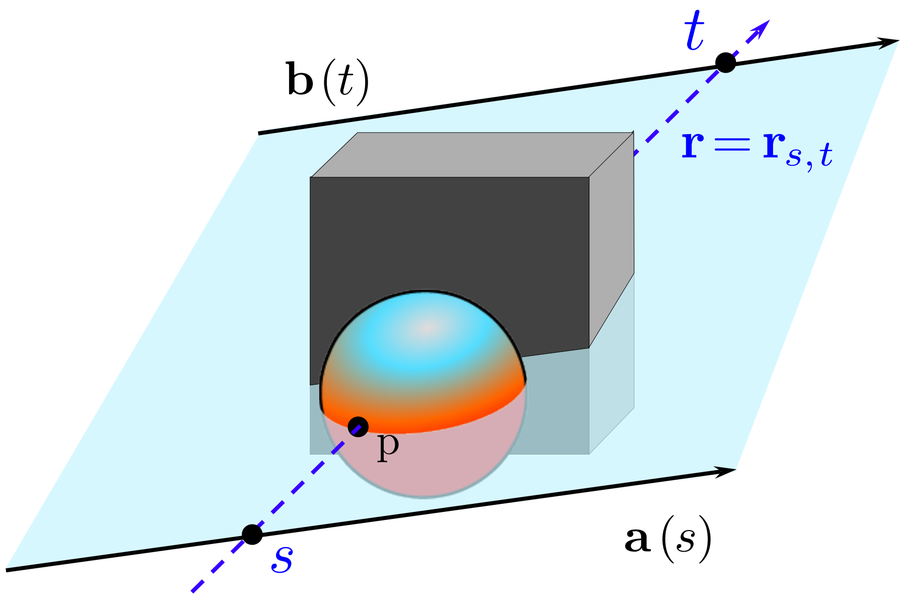


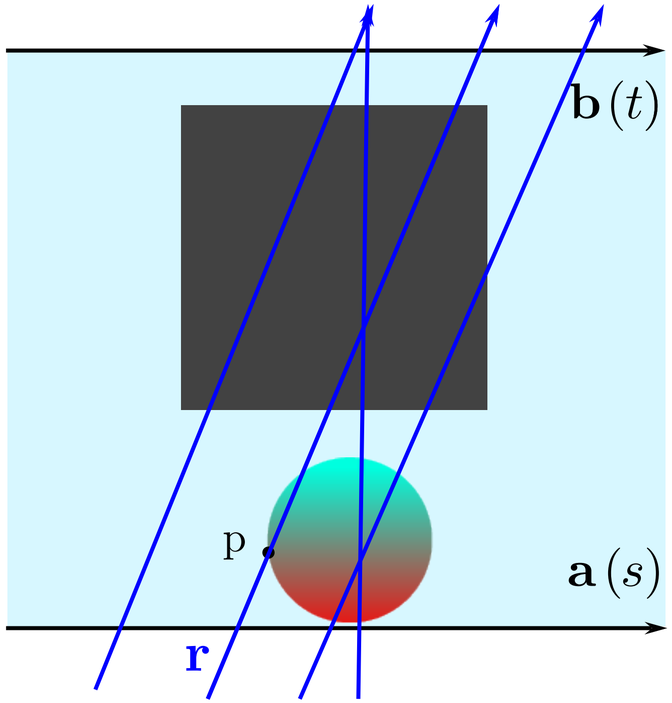


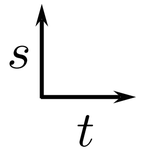


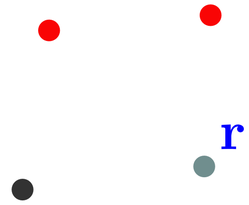
Some other rays

Triangulation - the infinitesimal perspective



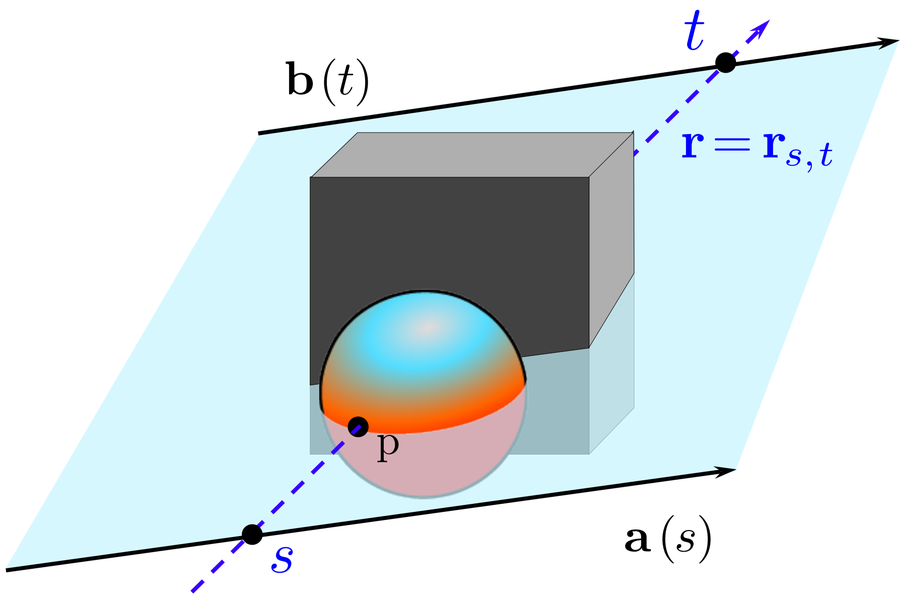


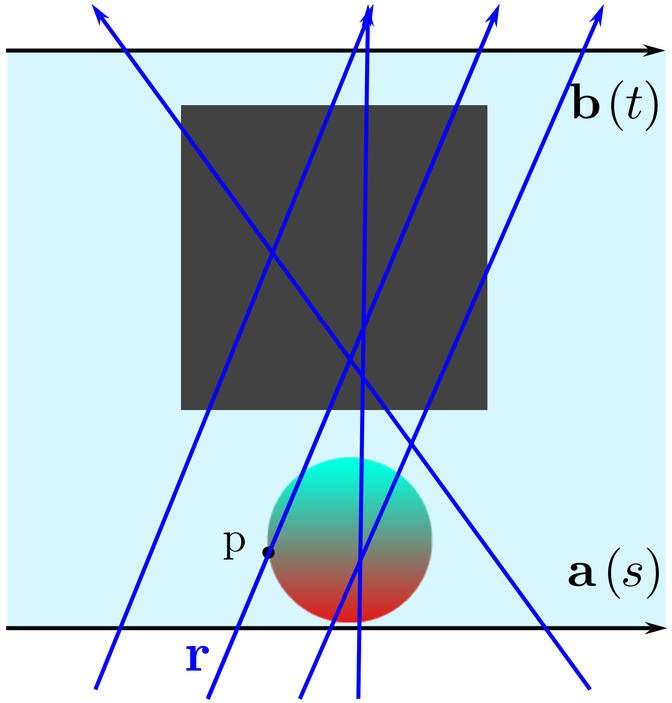


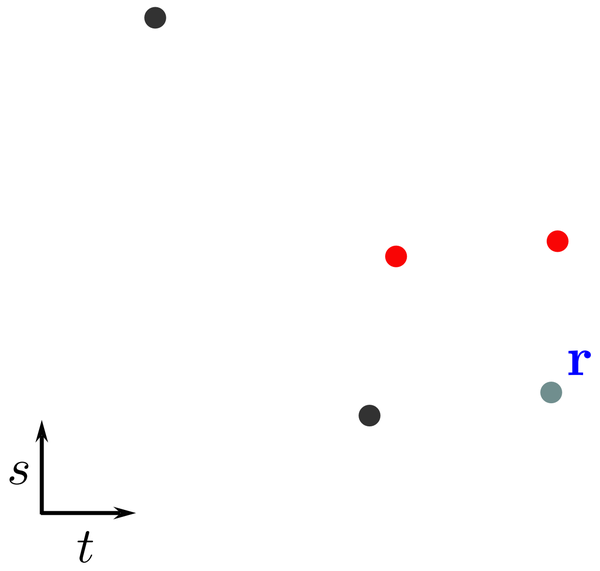


Some other rays

Triangulation - the infinitesimal perspective

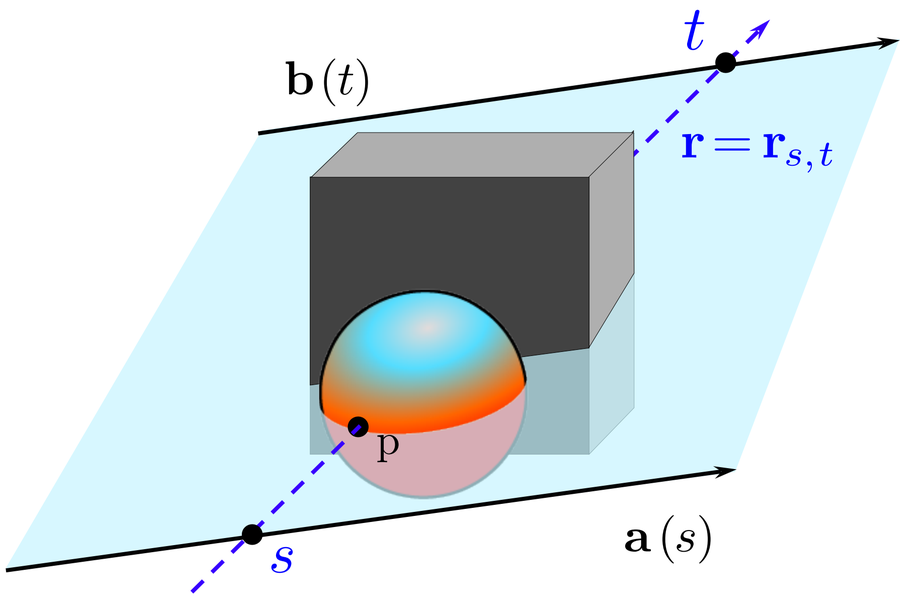


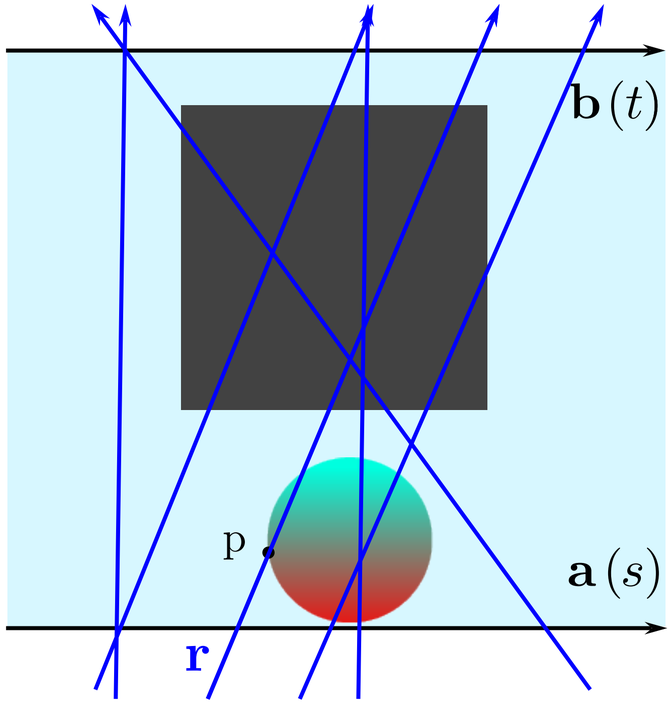


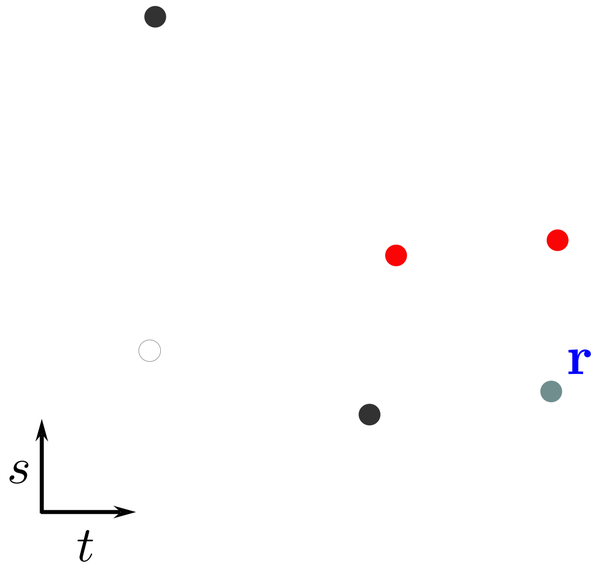


Some other rays

Triangulation - the infinitesimal perspective

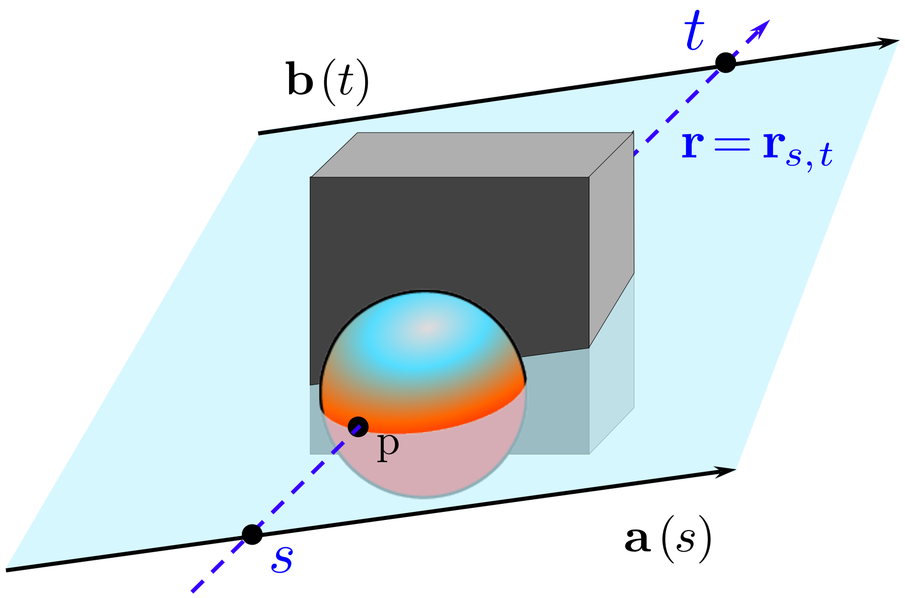


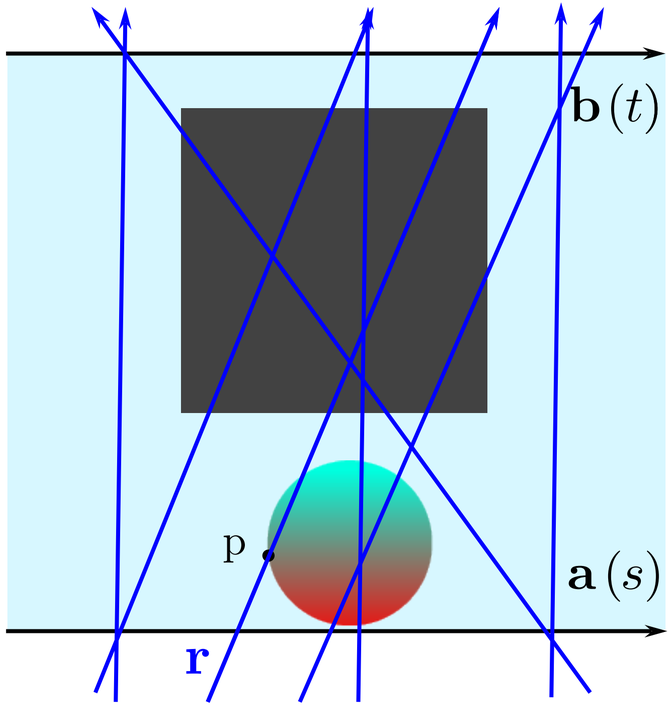


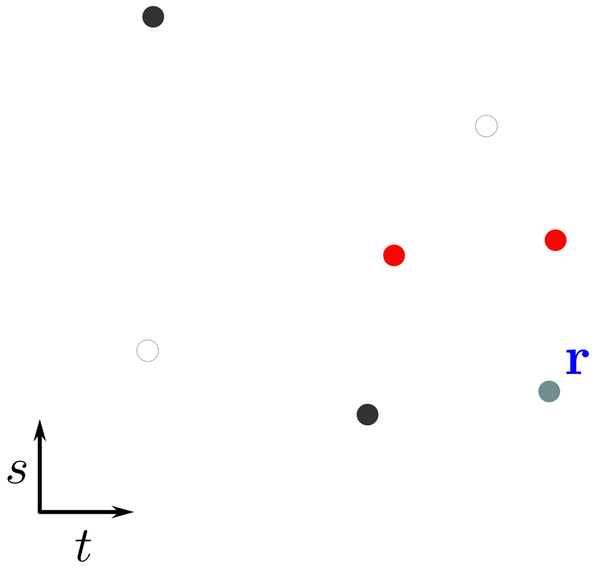


Some other rays

Triangulation - the infinitesimal perspective

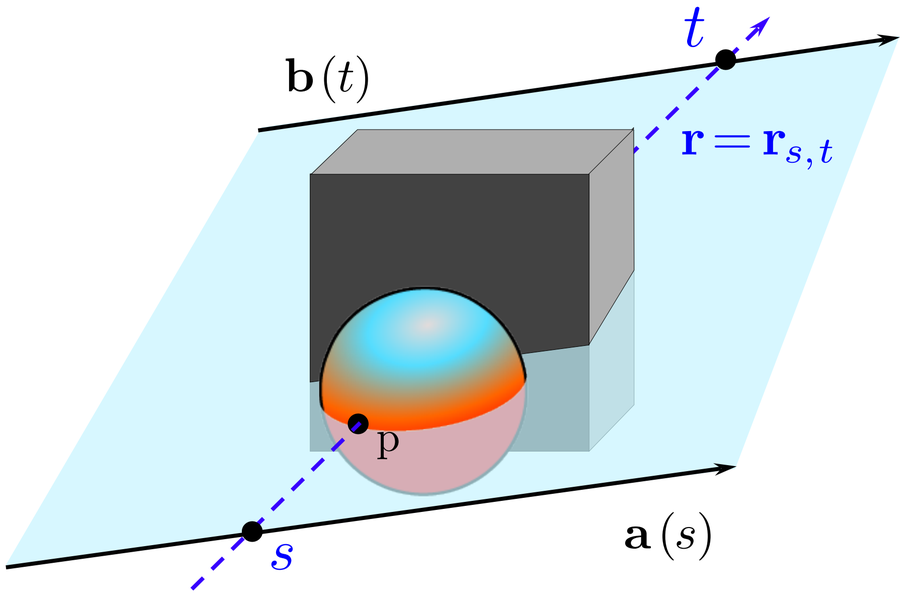


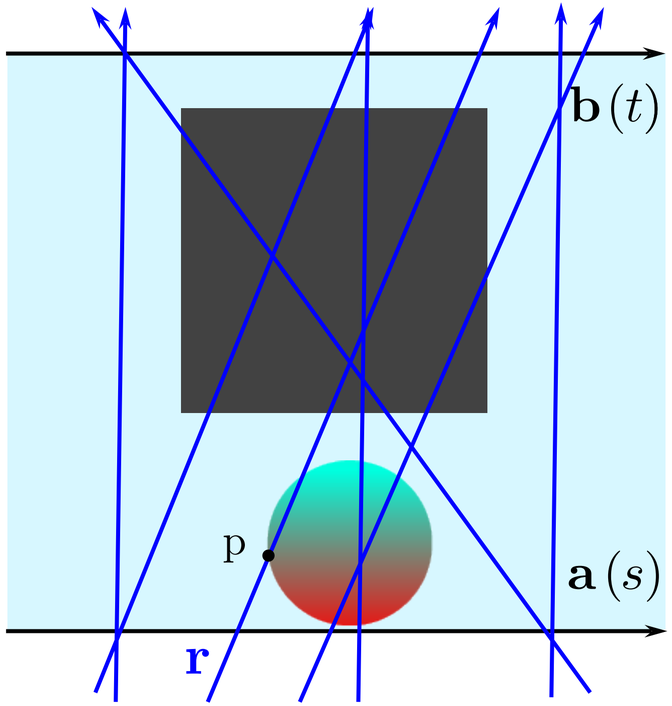




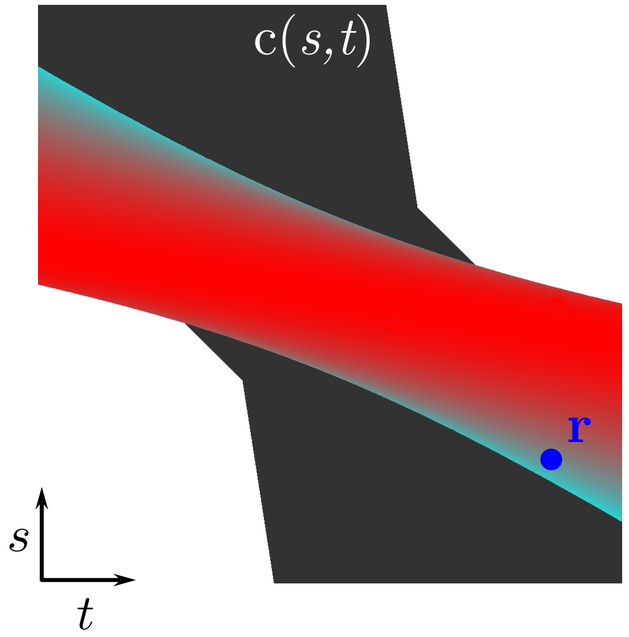
Some other rays

Triangulation - the infinitesimal perspective

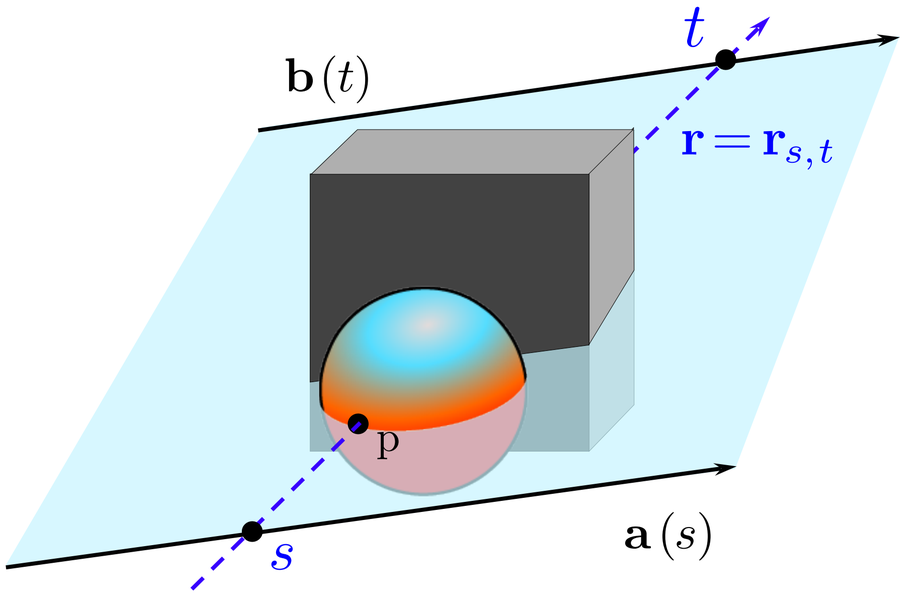


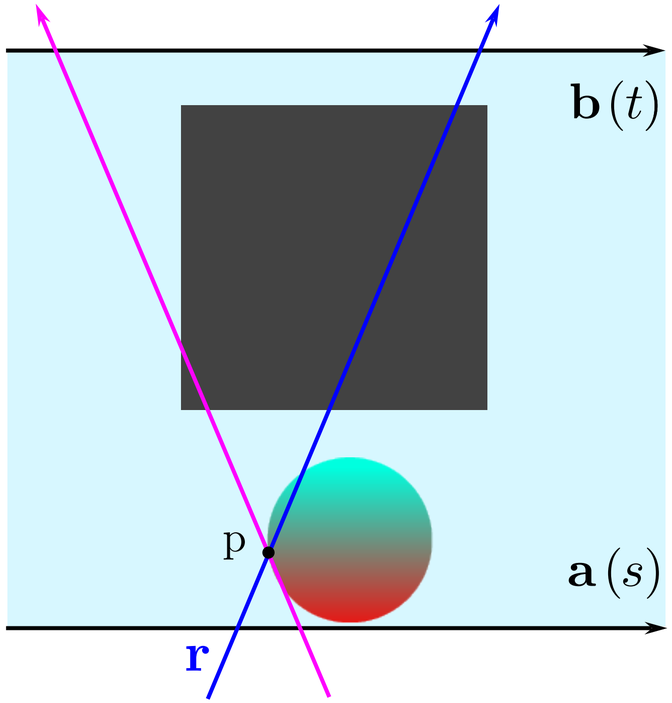


Epipolar Plane Image

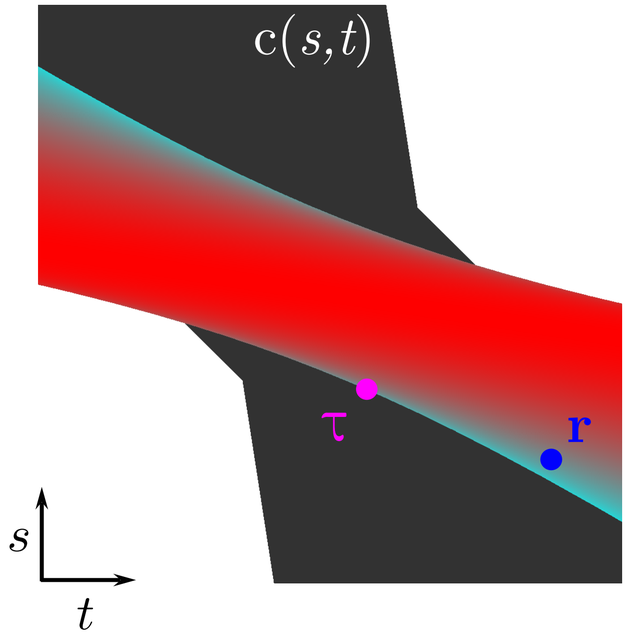


Triangulation - the infinitesimal perspective

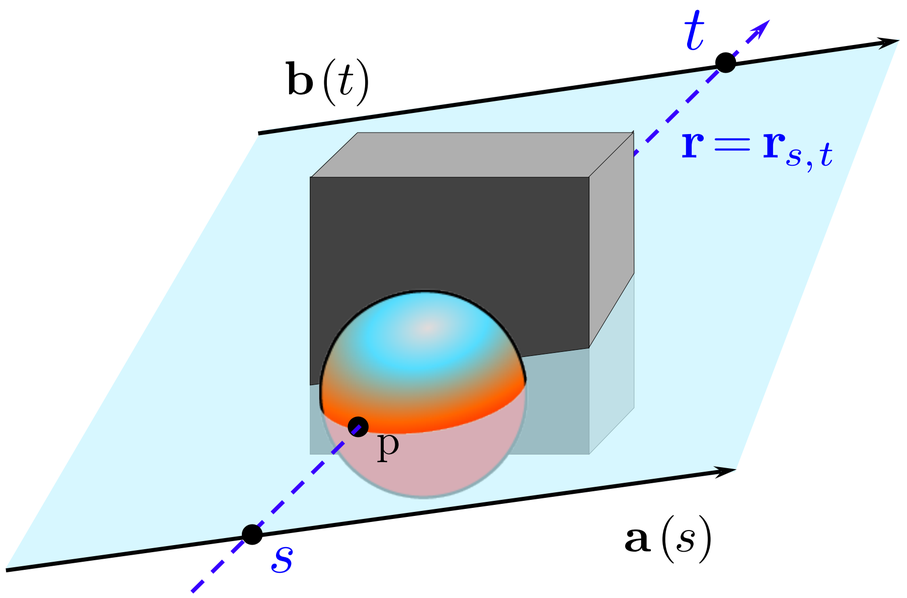


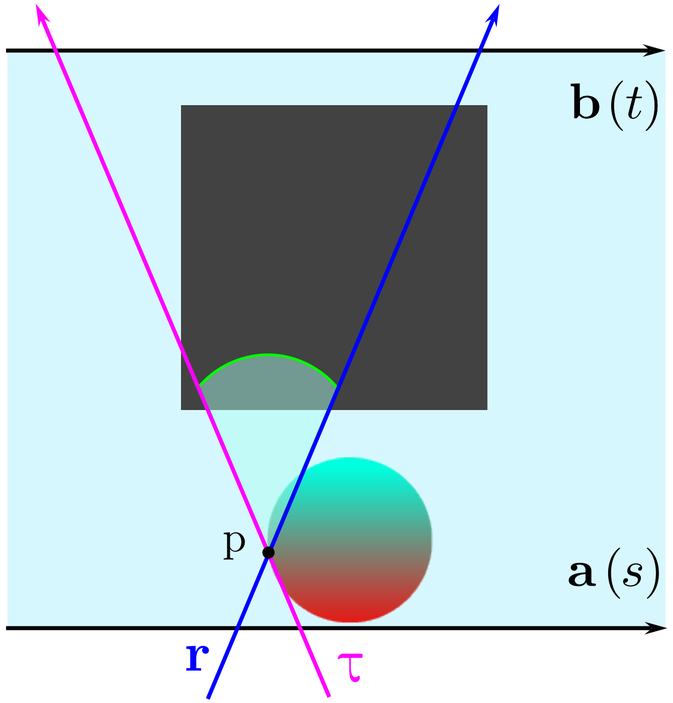


Epipolar Plane Image

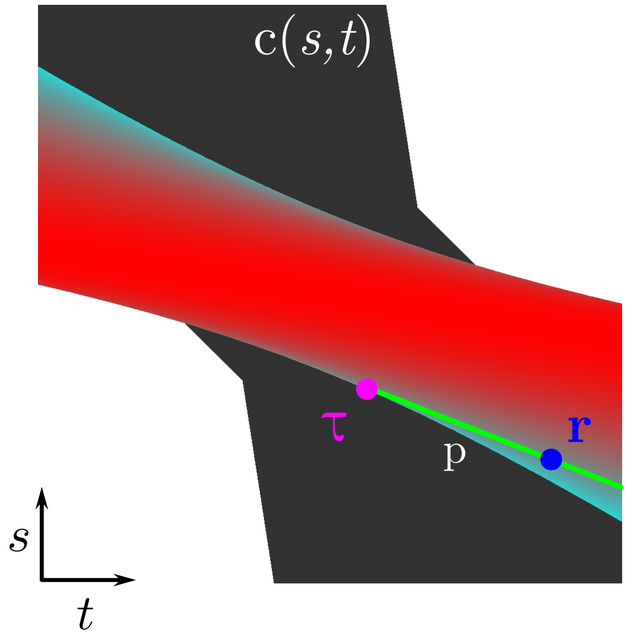


Triangulation - the infinitesimal perspective

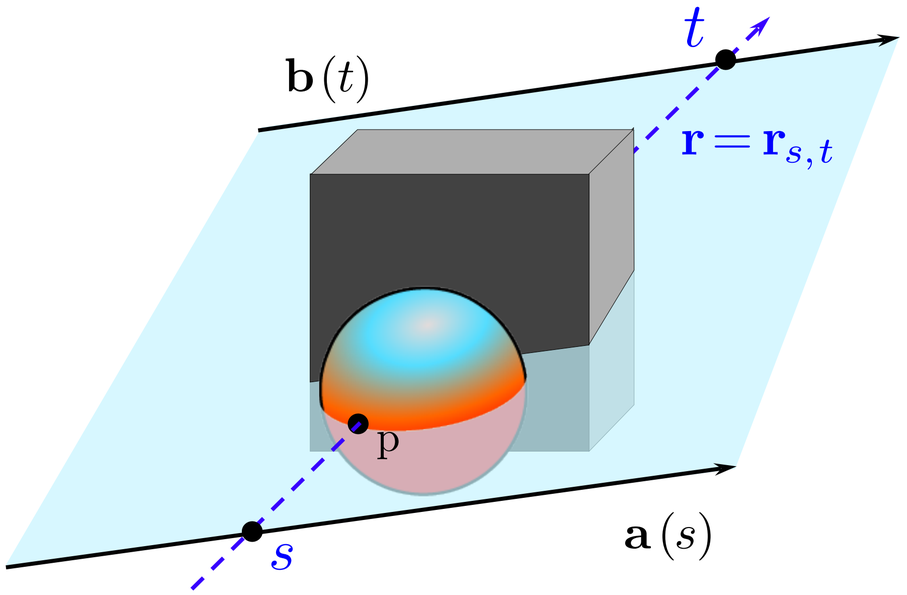


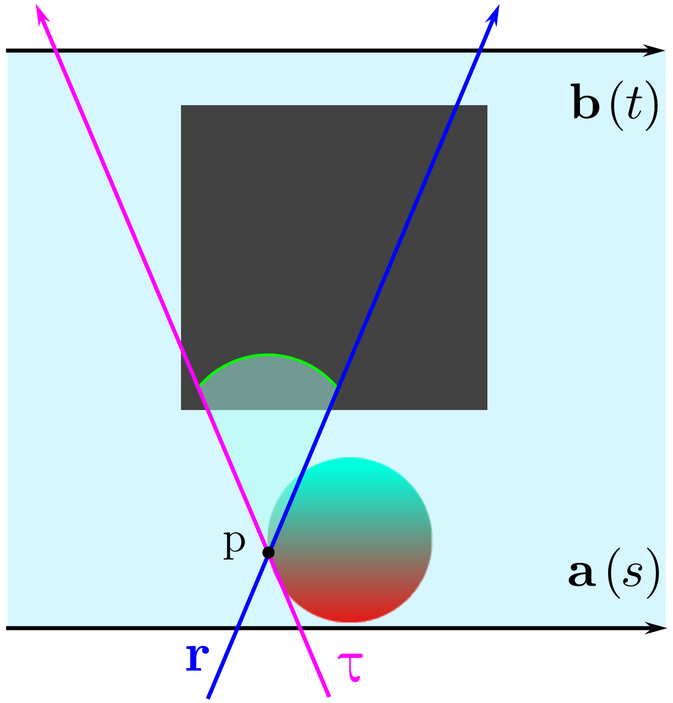


Epipolar Plane Image

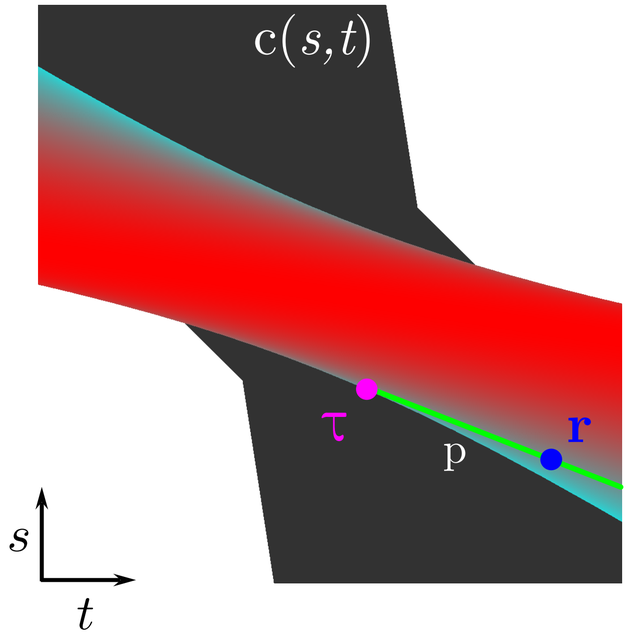


Triangulation - the infinitesimal perspective



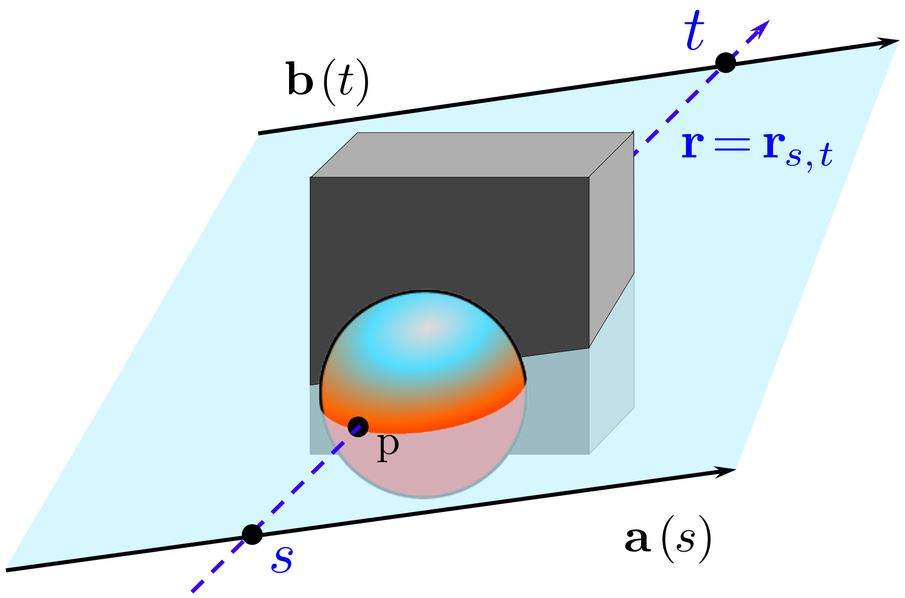


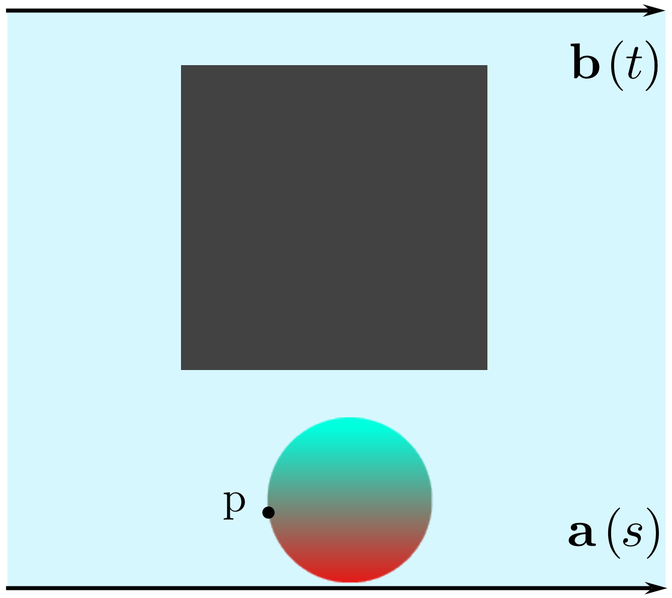
Epipolar Plane Image



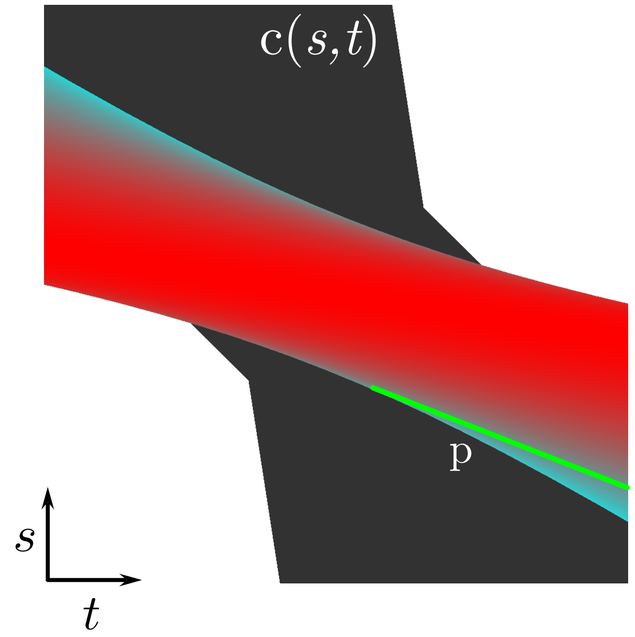
Points give lines of constant color in EPI $\mathbf{c}(s,t)$ – line is a levelset of the EPI.

Triangulation - the infinitesimal perspective





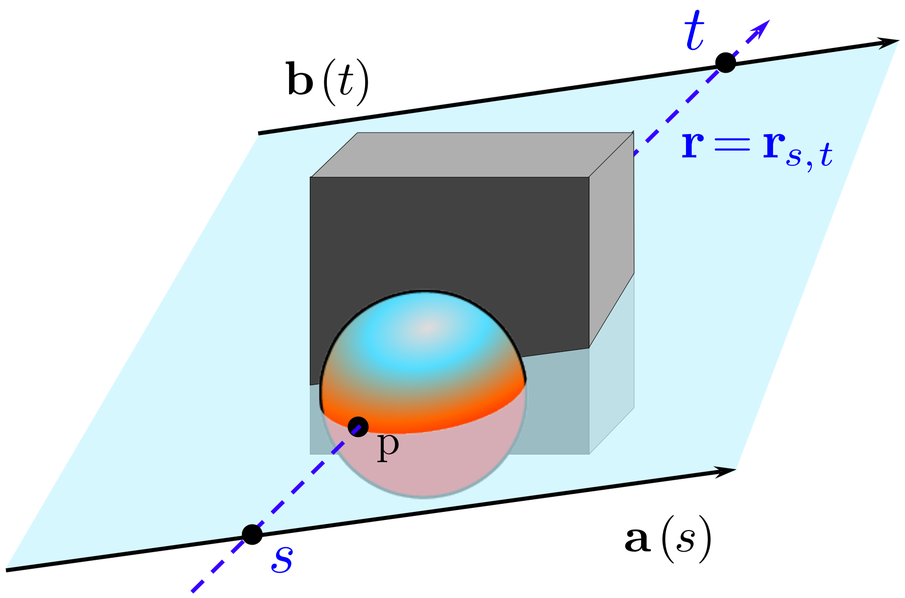
Epipolar Plane Image

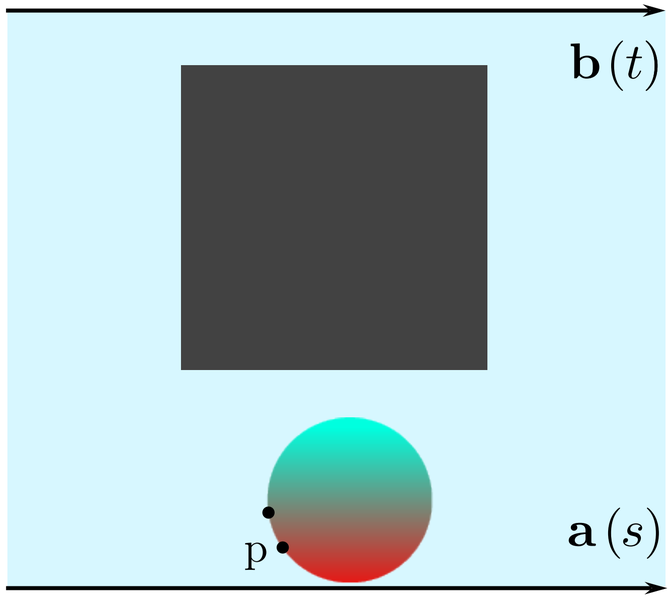


Points give lines of constant color in EPI $\mathbf{c}(s,t)$ – line is a levelset of the EPI.

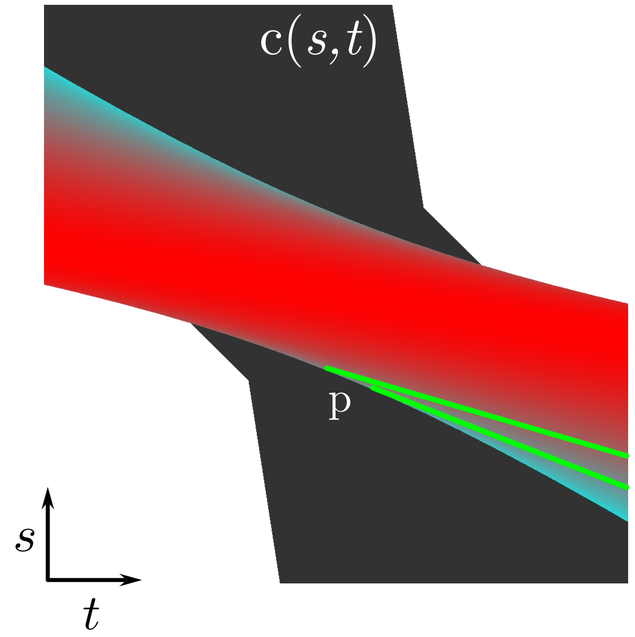
Slope of line decreases as point moves closer.

Triangulation - the infinitesimal perspective





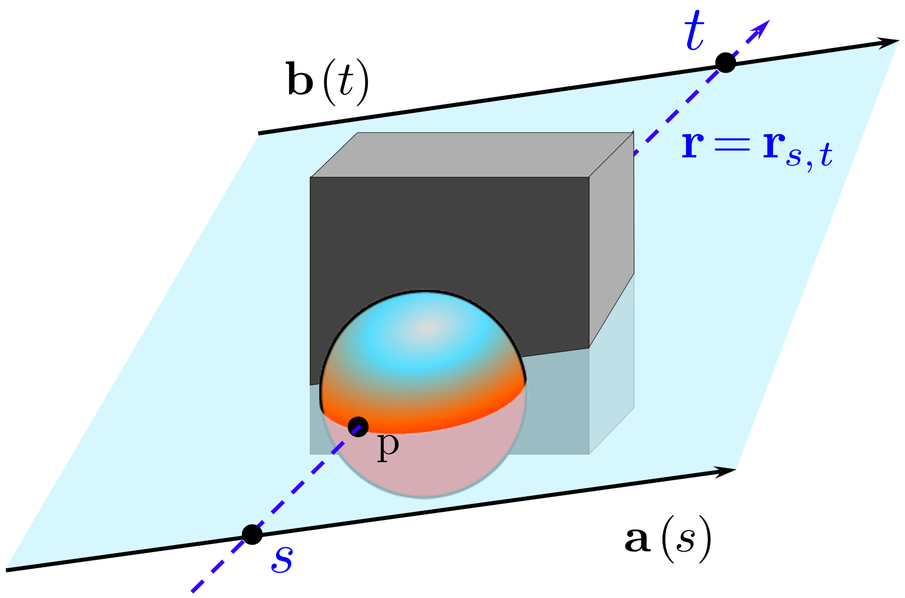
Epipolar Plane Image

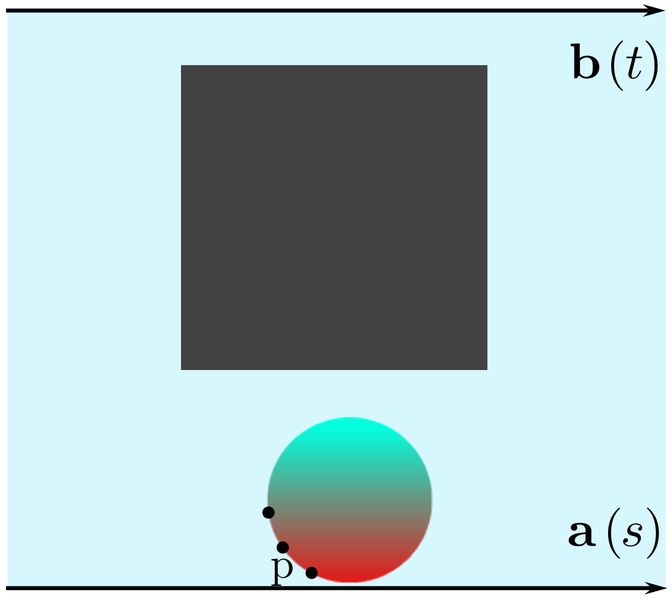


Points give lines of constant color in EPI $\mathbf{c}(s,t)$ – line is a levelset of the EPI.

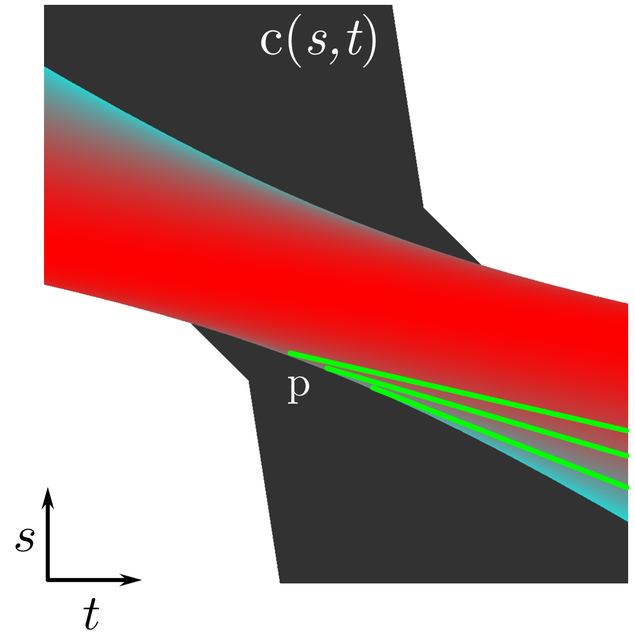
Slope of line decreases as point moves closer.

Triangulation - the infinitesimal perspective





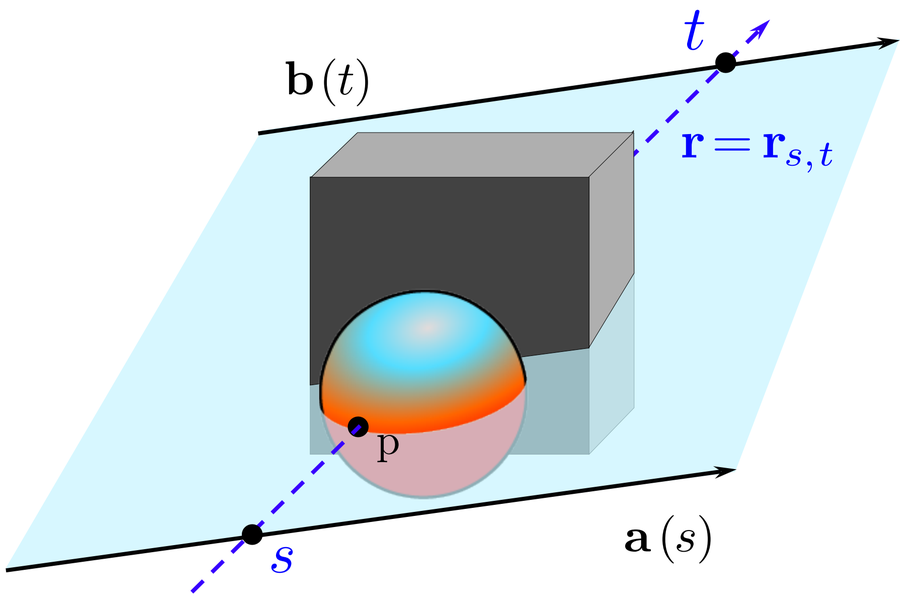
Epipolar Plane Image

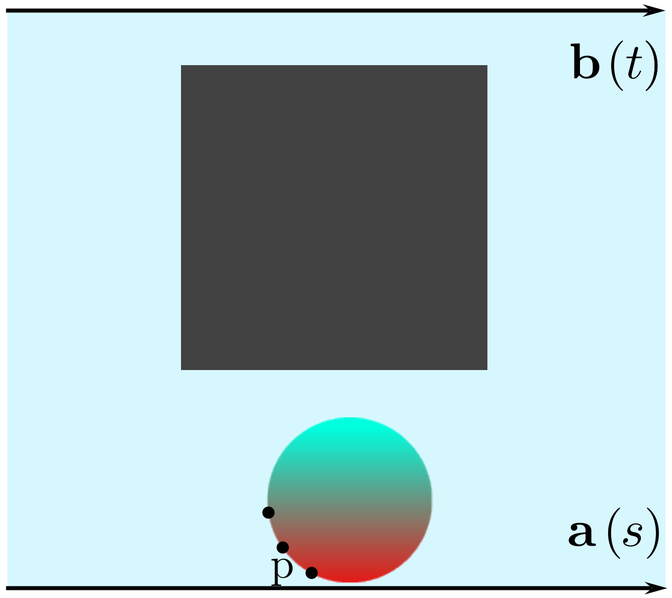


Points give lines of constant color in EPI $\mathbf{c}(s,t)$ – line is a levelset of the EPI.

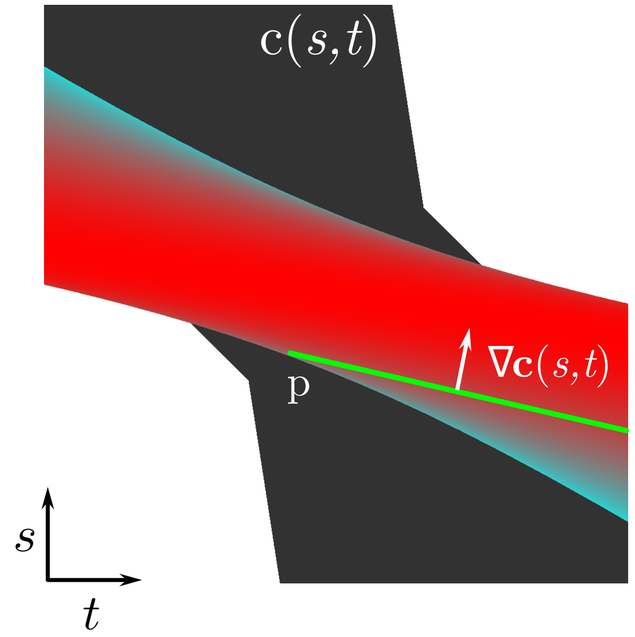
Slope of line decreases as point moves closer.

Triangulation - the infinitesimal perspective





Epipolar Plane Image



Points give lines of constant color in EPI $\mathbf{c}(s,t)$ – line is a levelset of the EPI.

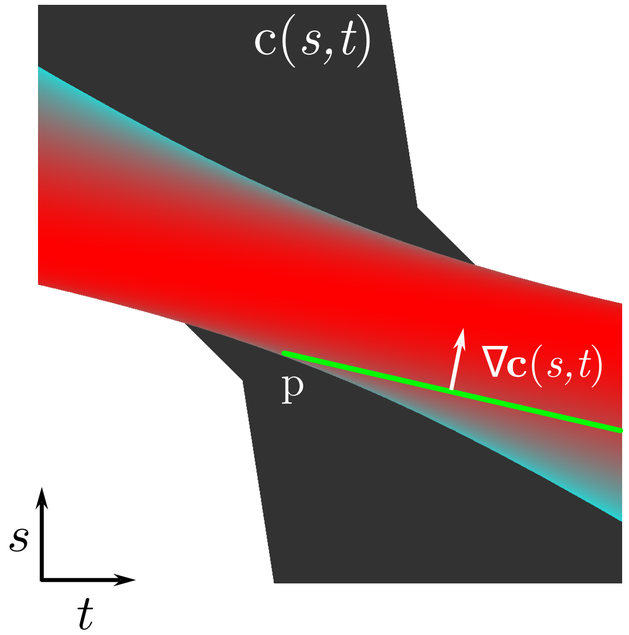
Slope of line decreases as point moves closer.

Gradient of $\mathbf{c}(s,t)$ is orthogonal to levelset -

Triangulation - the infinitesimal perspective

$$d(s,t) = D\frac{\partial_t \mathbf{c}(s,t)}{\partial_s \mathbf{c}(s,t) + \partial_t \mathbf{c}(s,t)}$$

Epipolar Plane Image

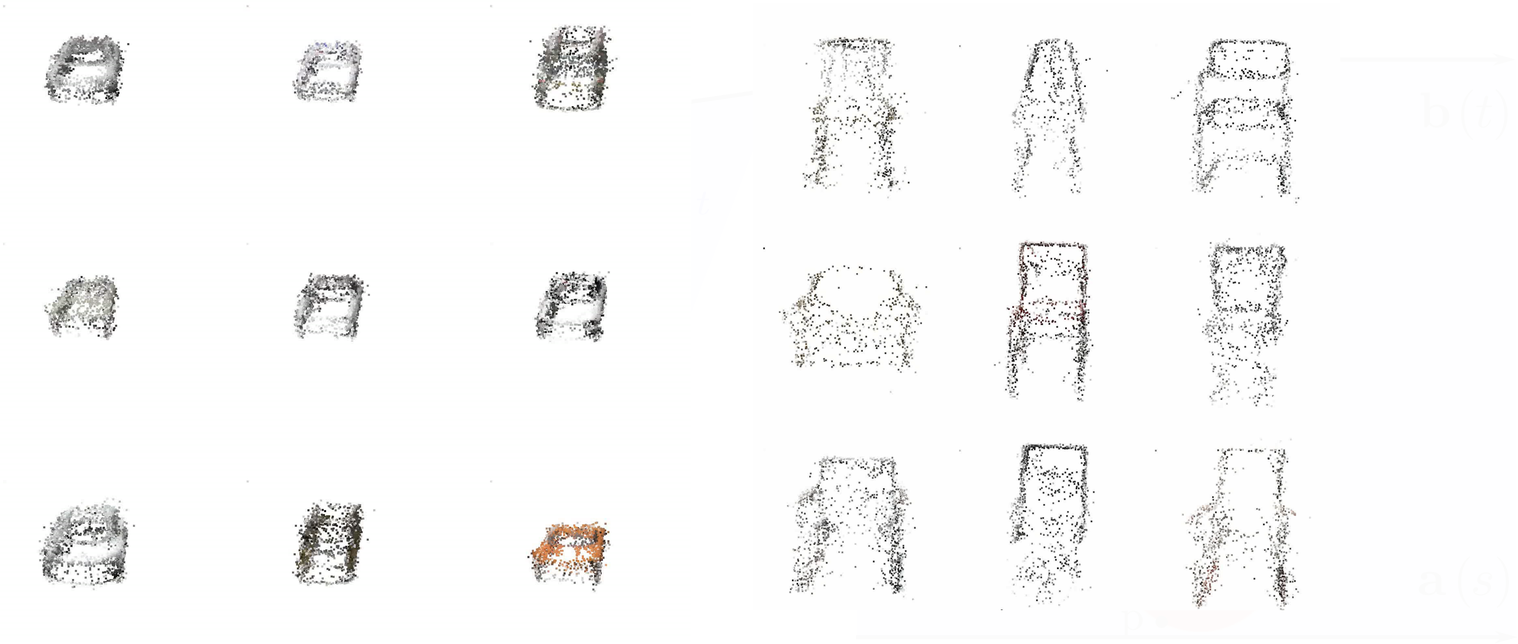


Points give lines of constant color in EPI $\mathbf{c}(s,t)$ – line is a levelset of the EPI.

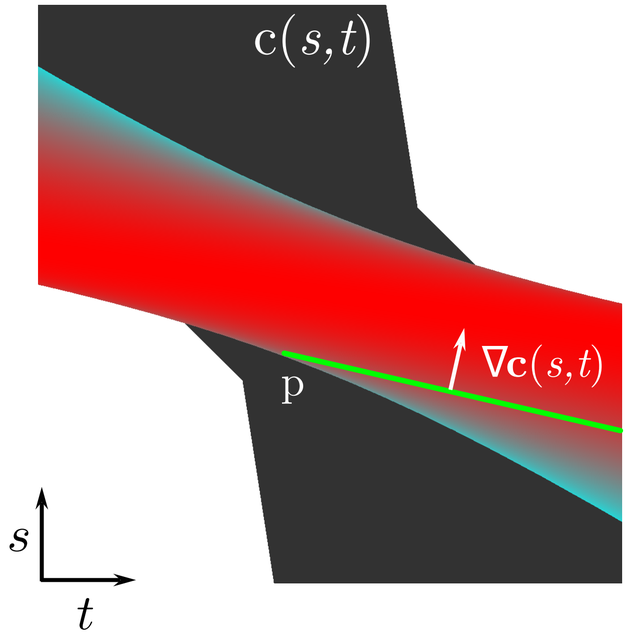
Slope of line decreases as point moves closer.

Gradient of $\mathbf{c}(s,t)$ is orthogonal to levelset - can extract depth from gradients of light field.

Triangulation - the infinitesimal perspective



Epipolar Plane Image

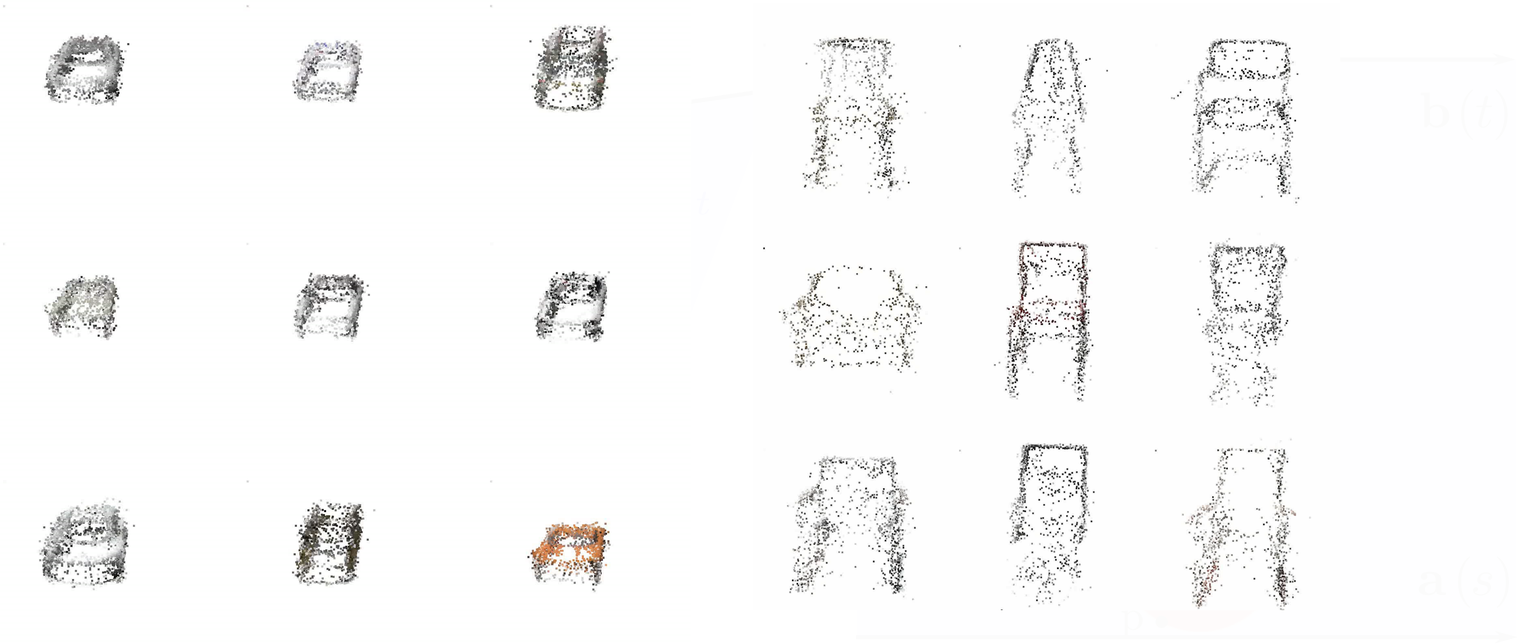


Points give lines of constant color in EPI $\mathbf{c}(s,t)$ – line is a levelset of the EPI.

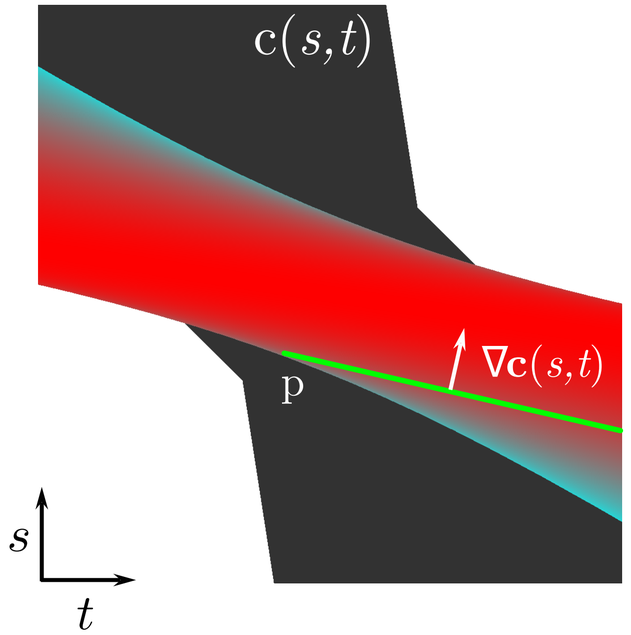
Slope of line decreases as point moves closer.

Gradient of $\mathbf{c}(s,t)$ is orthogonal to levelset - can extract depth from gradients of light field.

Triangulation - the infinitesimal perspective



Epipolar Plane Image

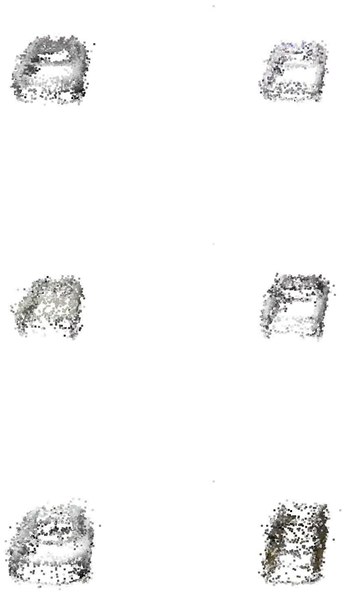


Points give lines of constant color in EPI $\mathbf{c}(s,t)$ – line is a levelset of the EPI.

Slope of line decreases as point moves closer.

Gradient of $\mathbf{c}(s,t)$ is orthogonal to levelset - can extract depth from gradients of light field.

Triangulation - the infinitesimal perspective



Light Field Networks: Neural Scene Representations with Single-Evaluation Rendering

Vincent Sitzmann[1,*] sitzmann@mit.edu

Semon Rezchikov[2,*] skr@math.columbia.edu

William T. Freeman[1,3] billf@mit.edu

Joshua B. Tenenbaum[1,4,5] jbt@mit.edu

Frédo Durand[1] fredo@mit.edu

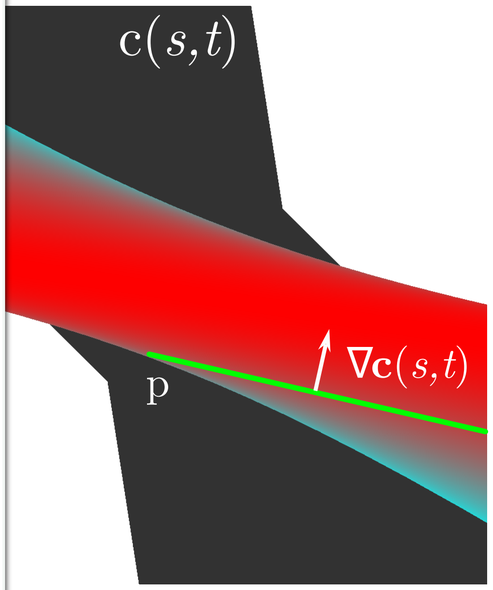
[1]MIT CSAIL [2]Columbia University [3] IAFI [4]MIT BCS [5] CBMM

vsitzmann.github.io/lfns/

Abstract

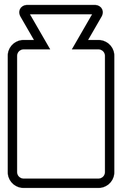
Inferring representations of 3D scenes from 2D observations is a fundamental problem of computer graphics, computer vision, and artificial intelligence. Emerging 3D-structured neural scene representations are a promising approach to 3D scene understanding. In this work, we propose a novel neural scene representation, Light Field Networks or LFNs, which represent both geometry and appearance of the underlying 3D scene in a 360-degree, four-dimensional light field parameterized via a neural network. Rendering a ray from an LFN requires only a *single* network evaluation, as opposed to hundreds of evaluations per ray for ray-marching or volumetric based renderers in 3D-structured neural scene representations. In the setting of simple scenes, we leverage meta-learning to learn a prior over LFNs that enables multi-view consistent light field reconstruction from as little as a single image observation. This results in dramatic reductions in time and memory complexity, and enables real-time rendering. The cost of storing a 360-degree light field via an LFN is two orders of magnitude lower than conventional methods such as the Lumigraph. Utilizing the analytical differentiability of neural implicit representations and a novel parameterization of light space, we further demonstrate the extraction of sparse depth maps from LFNs.

Epipolar Plane Image



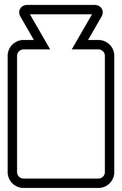
Points g of the EPI.

Gradient of $\mathbf{c}(s,t)$ is orthogonal to levelset - can extract depth from gradients of light field.

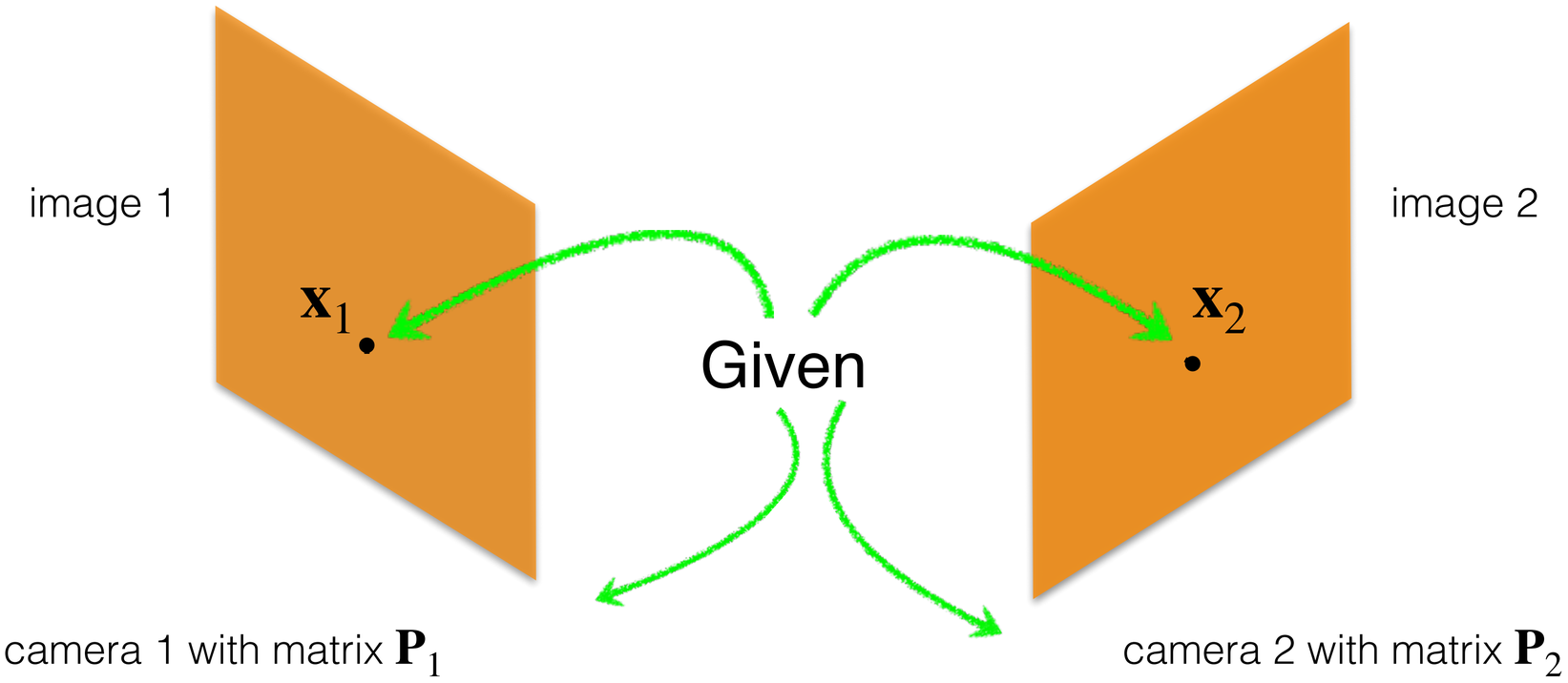


Known $\mathbf{P}_1, \mathbf{P}_2$!

What's 3D location of point?

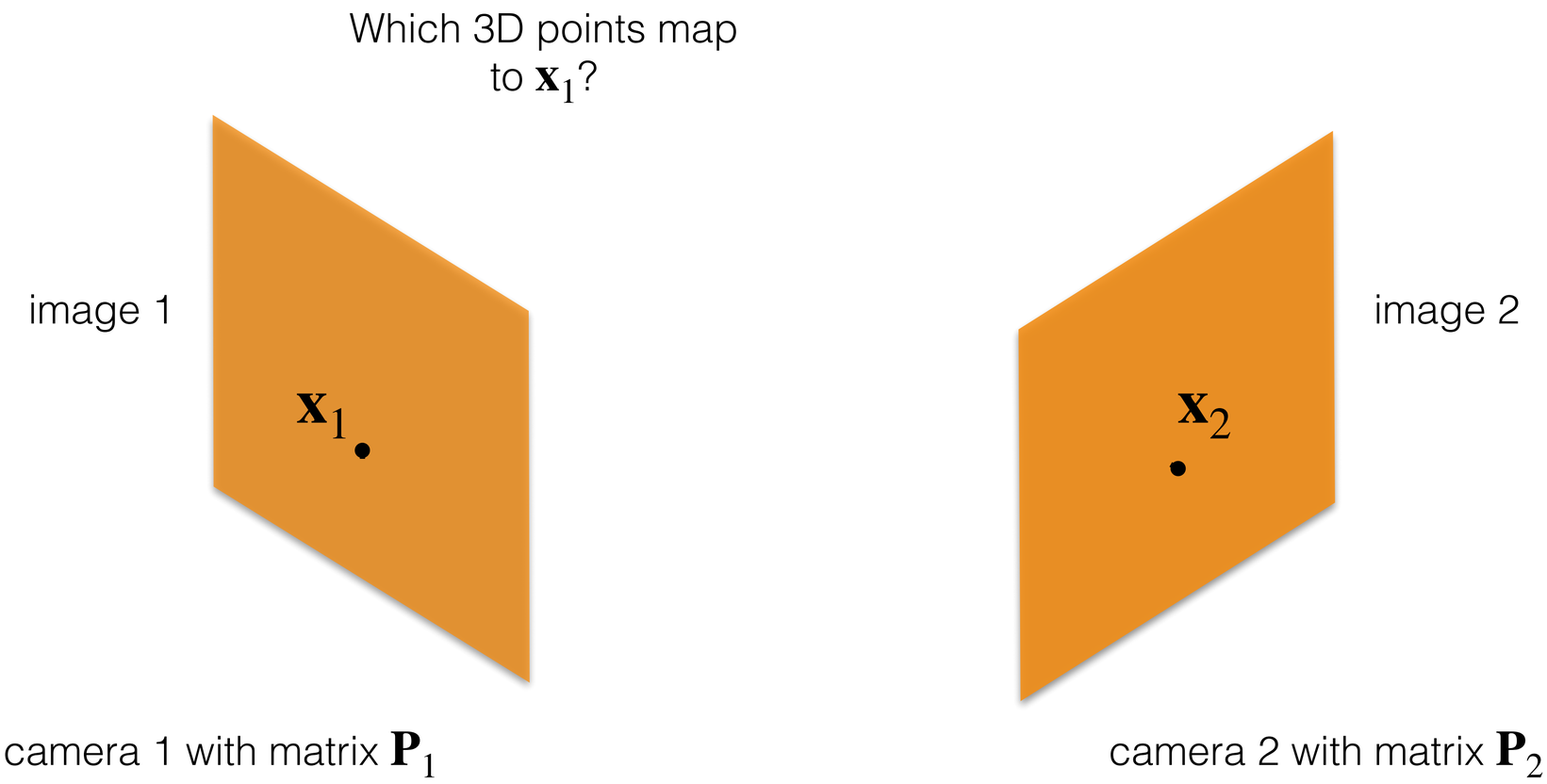


Triangulation - the finite perspective



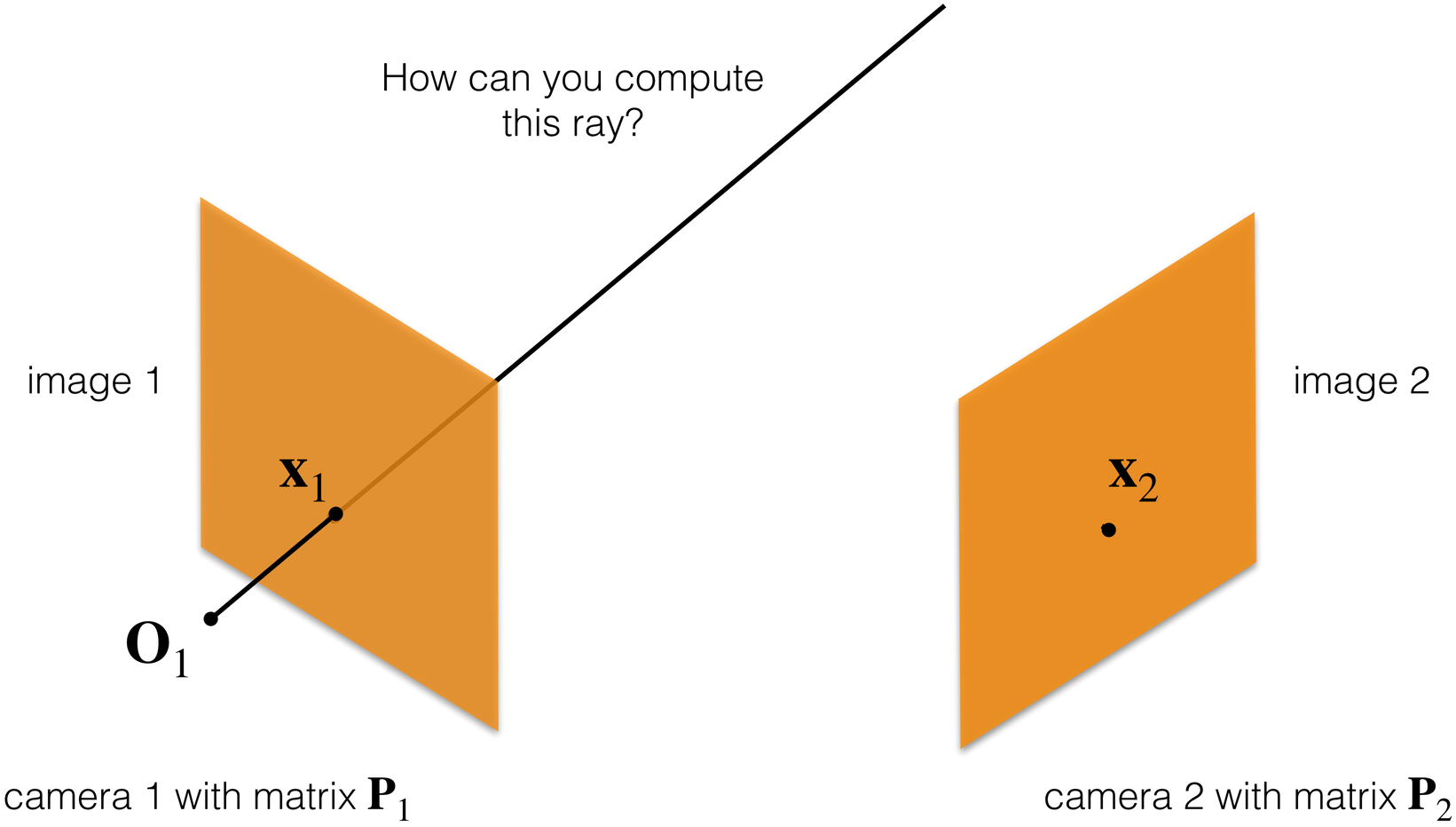
Adapted from: CMU 16-385 (Yannis, Kris)

Triangulation - the finite perspective



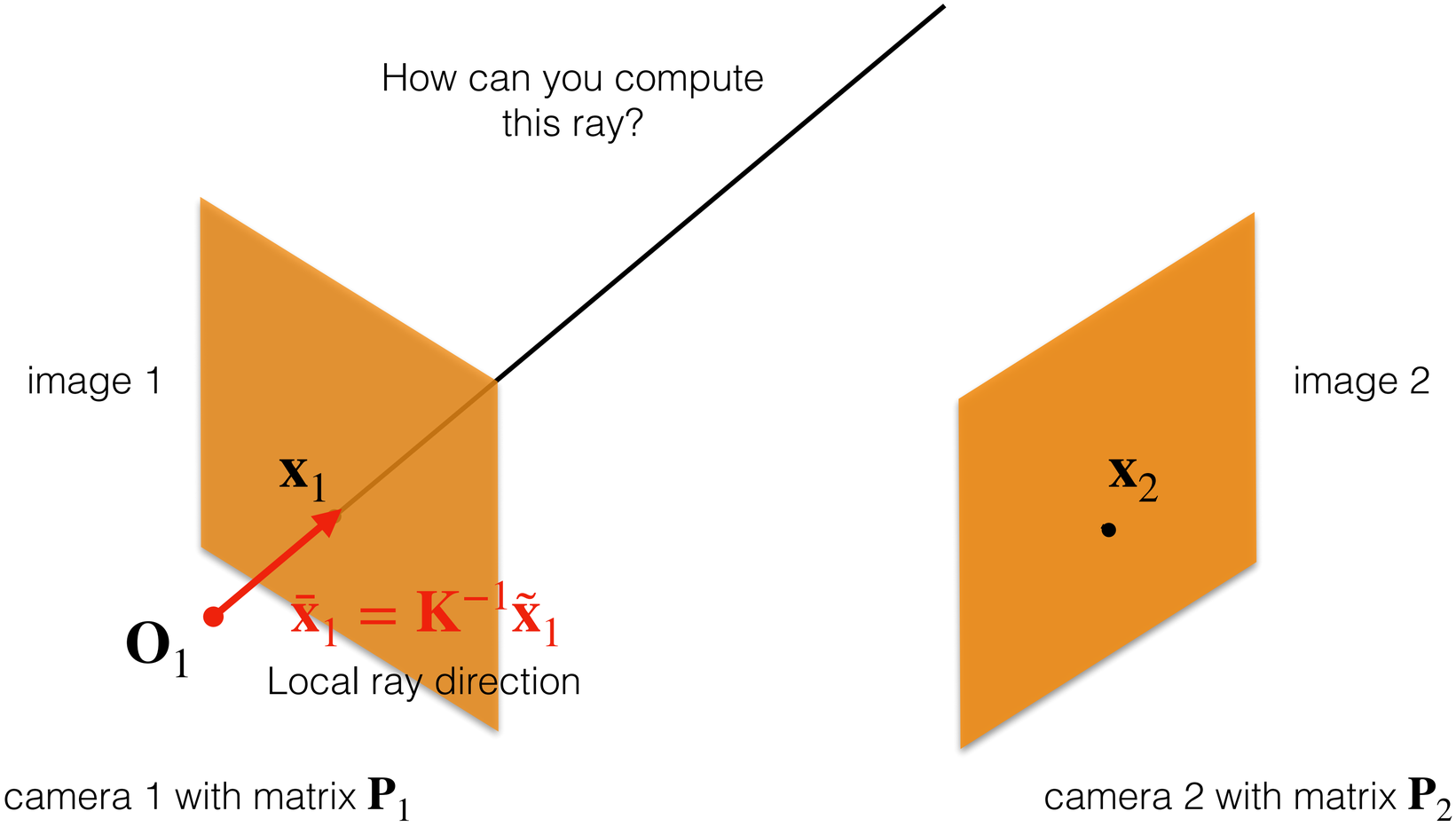
Adapted from: CMU 16-385 (Yannis, Kris)

Triangulation - the finite perspective



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

Triangulation - the finite perspective

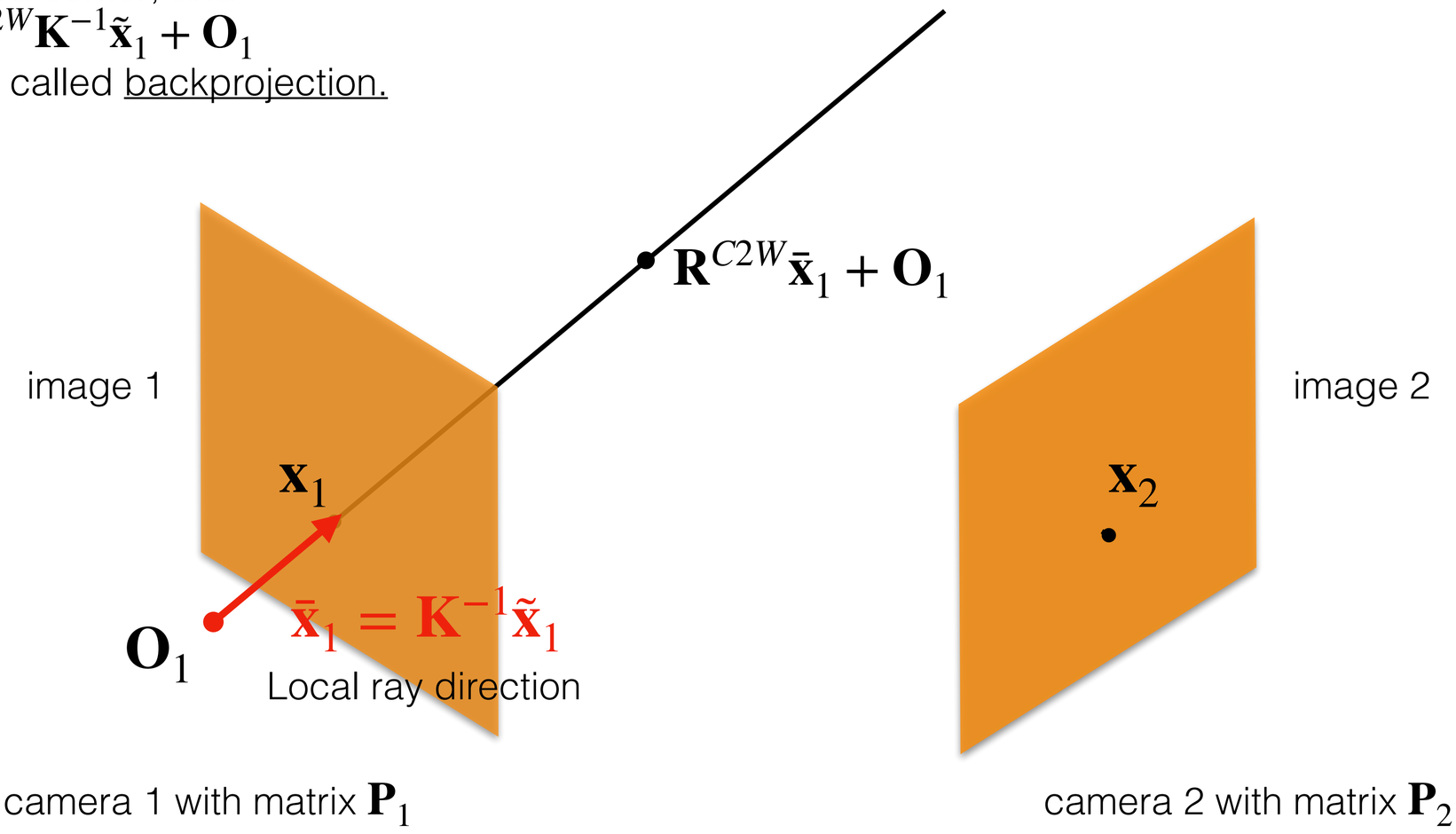


Adapted from: CMU 16-385 (Yannis, Kris)

Triangulation - the finite perspective

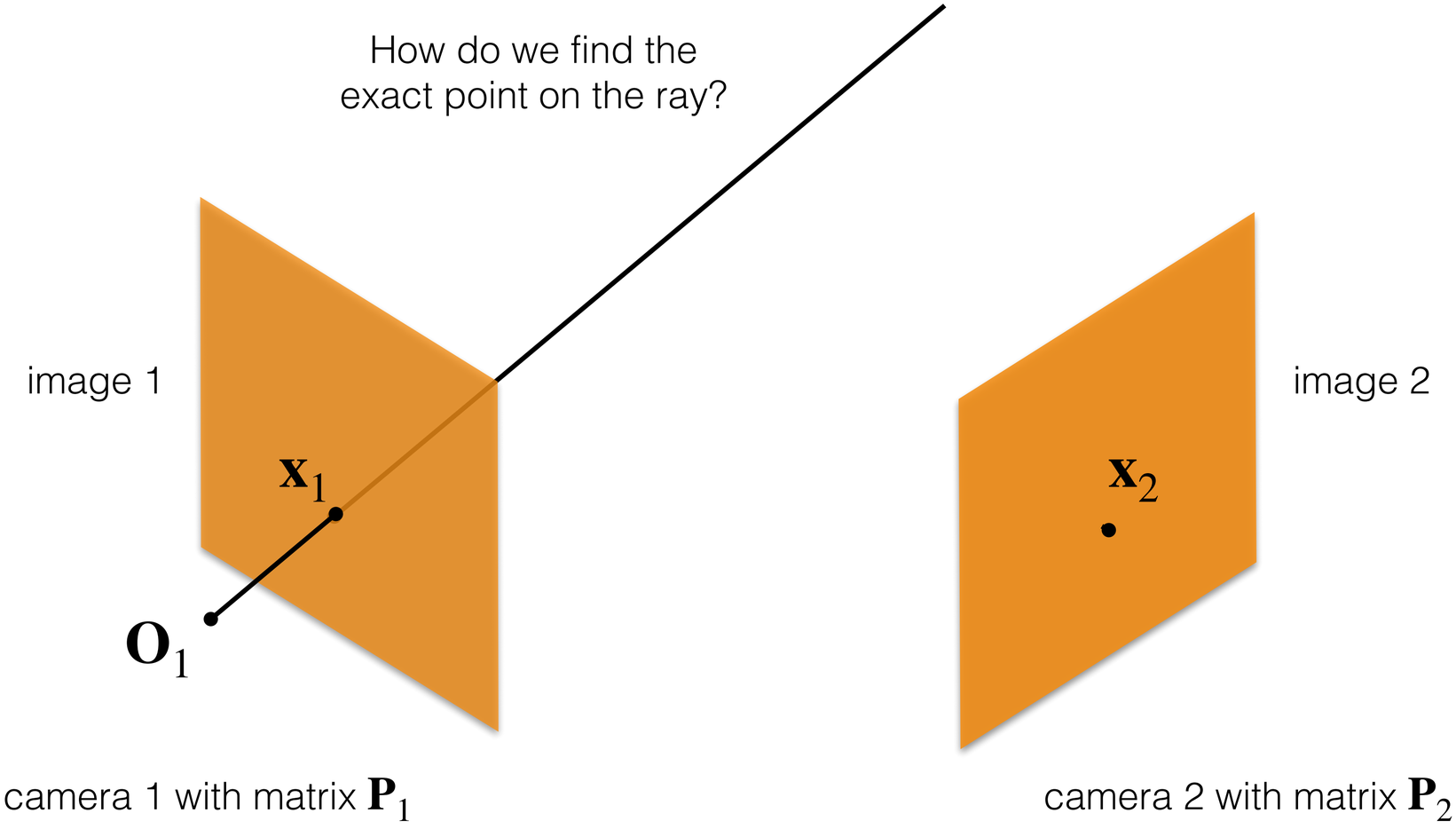
Create two points on the ray:
1) find the camera center; and
2) Compute $\mathbf{R}^{C2W}\mathbf{K}^{-1}\tilde{\mathbf{x}}_1 + \mathbf{O}_1$

This procedure is called backprojection.



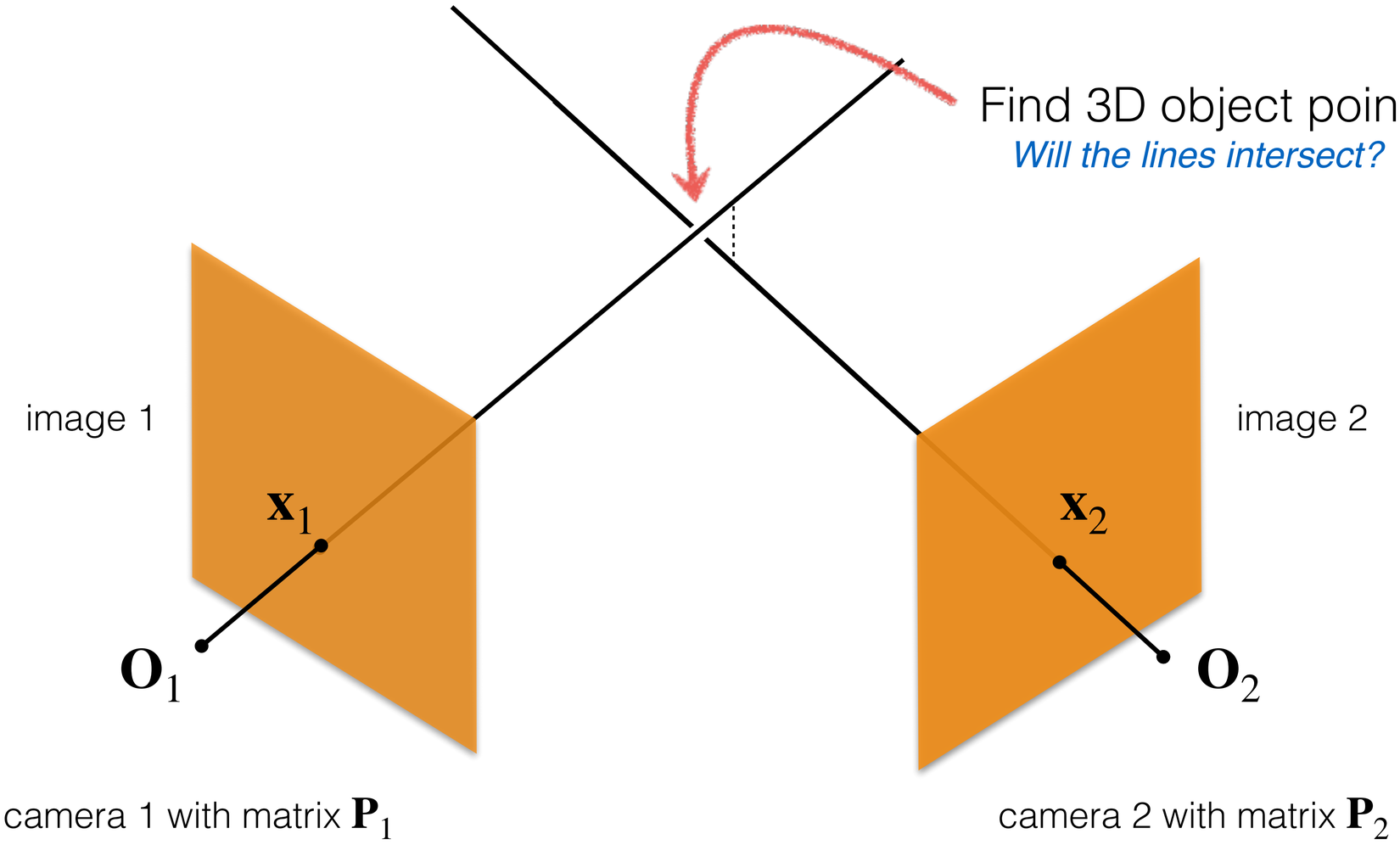
Adapted from: CMU 16-385 (Yannis, Kris)

Triangulation - the finite perspective



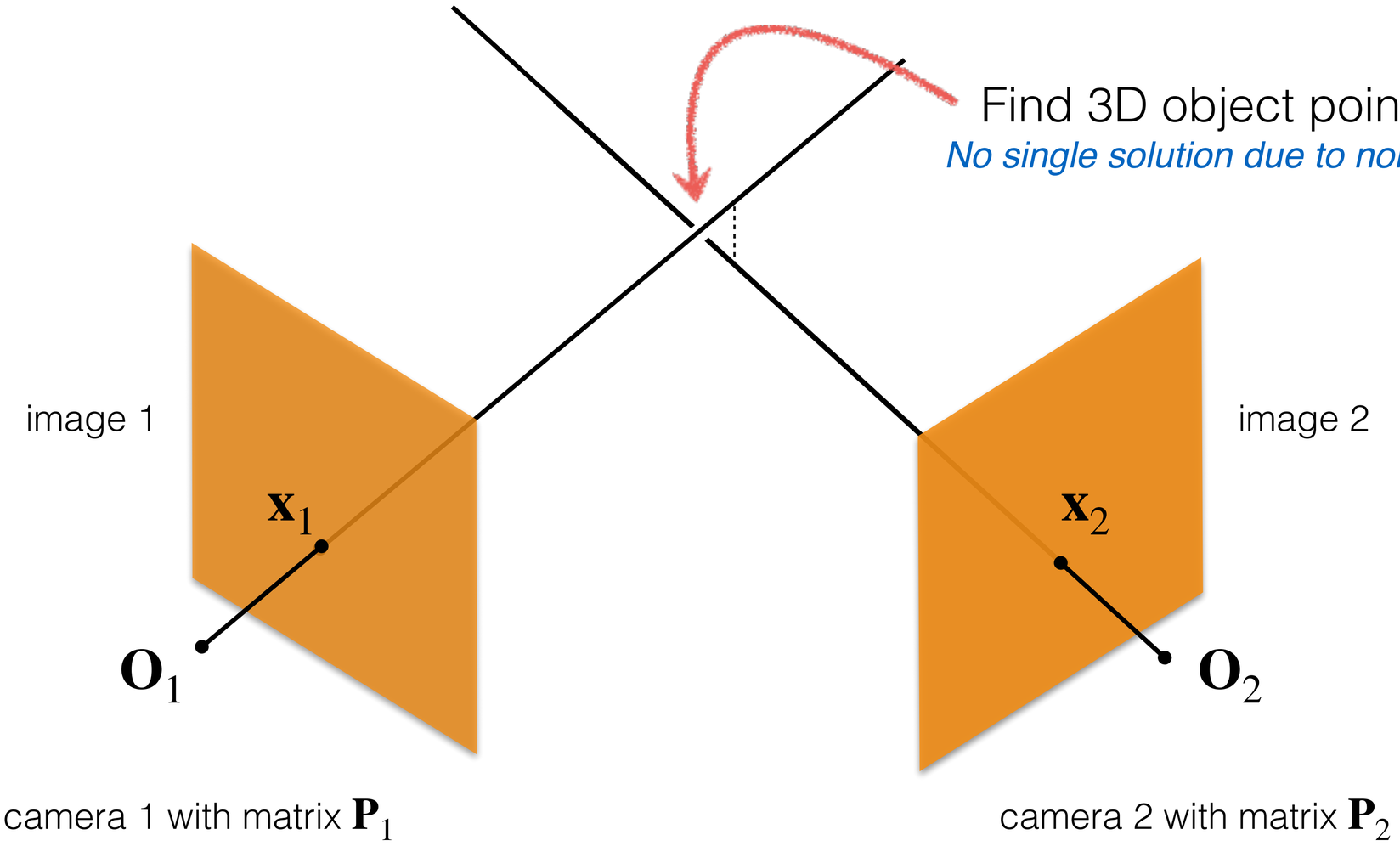
Adapted from: CMU 16-385 (Yannis, Kris)

Triangulation - the finite perspective



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

Triangulation - the finite perspective



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

Triangulation - the finite perspective

Given a set of (noisy) matched pixel coordinates

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

Estimate the 3D point

$$\mathbf{X}$$

Triangulation - the finite perspective

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 - the finite perspective

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

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

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 - the finite perspective

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

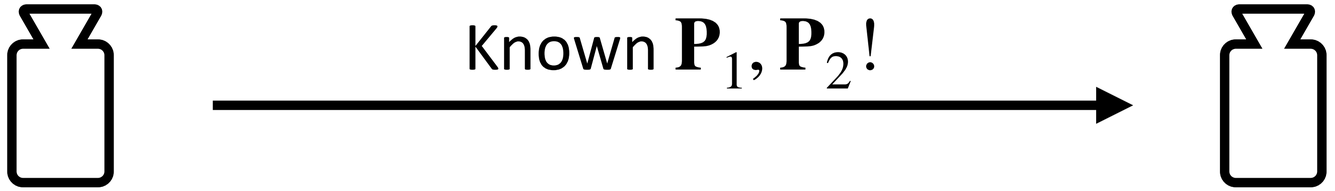
$$\tilde{\pi}_i(\mathbf{X}) = \mathbf{K}_i[\mathbf{I}\,|\,0]\mathbf{C}_i^{W2C}\tilde{\mathbf{X}}$$

Then we can solve a little least squares problem:

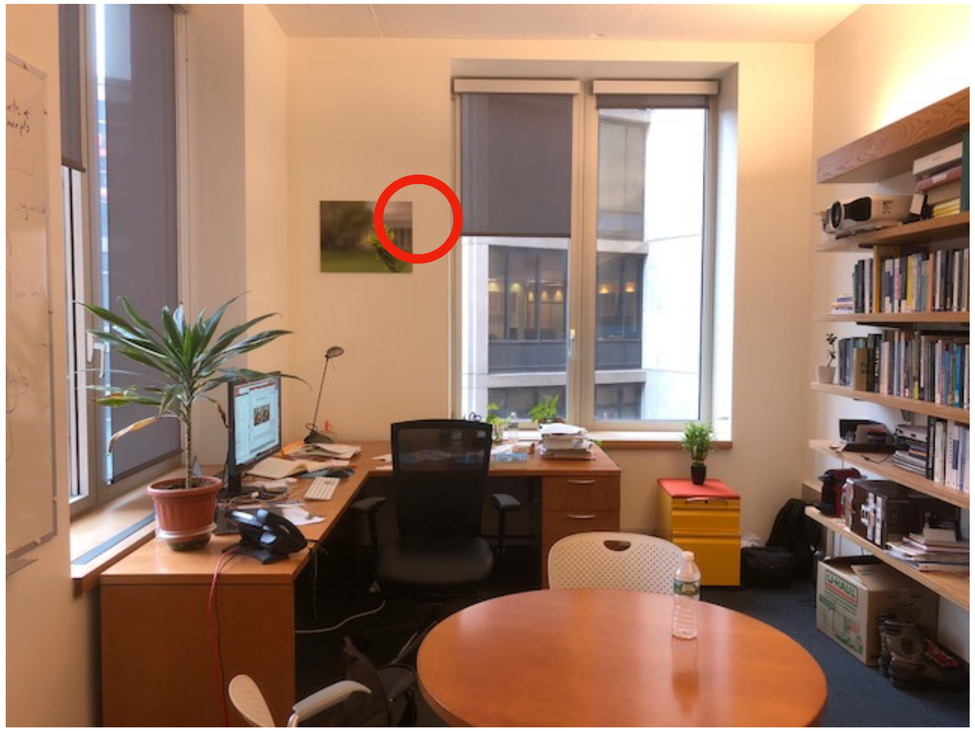
$$\mathbf{X}^* = argmin_{\mathbf{X}} \sum_{i}^{N} \|\pi_i(\mathbf{X}) - \mathbf{x}_i\|_2^2$$

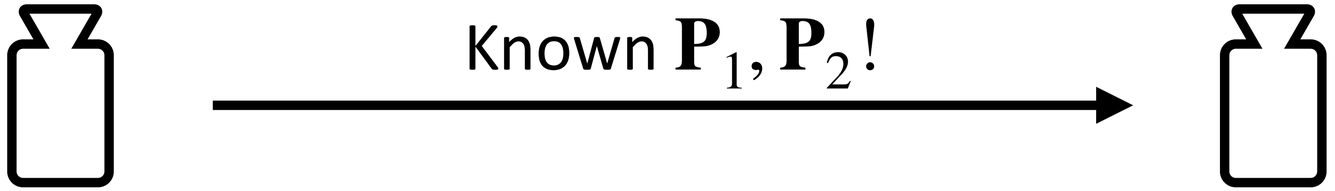
Can be solved via numerical optimization
(Gradient Descent, or smarter, Levenberg-Marquardt)

Motivation: Epipolar Lines



Motivation: Epipolar Lines

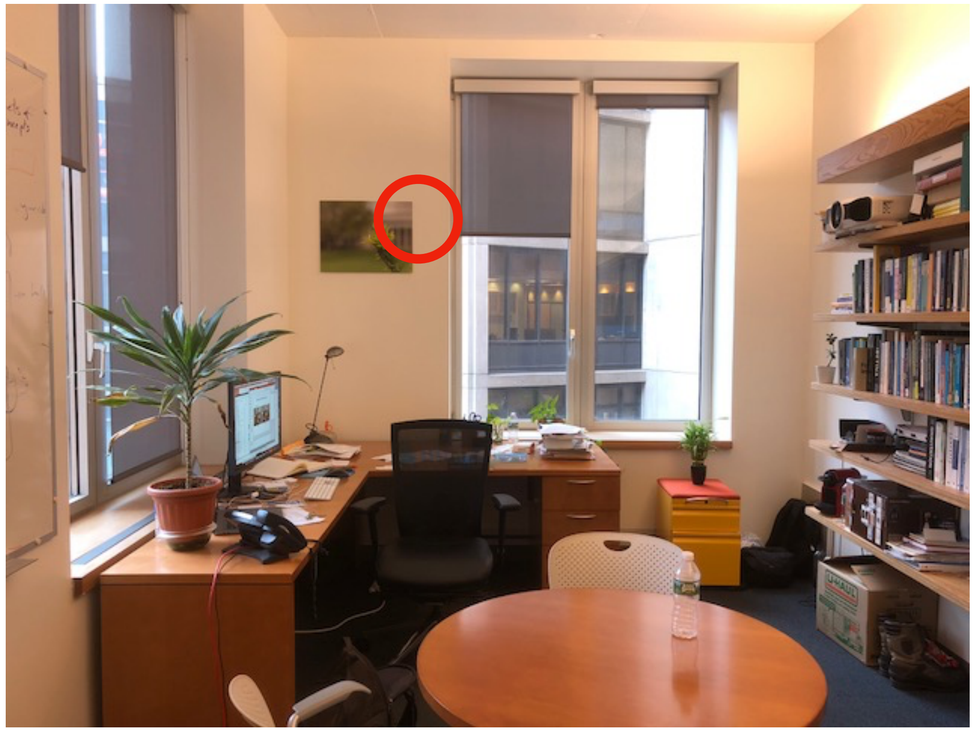


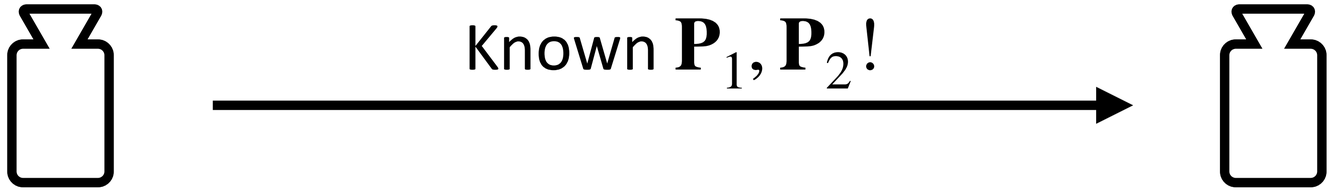


Motivation: Epipolar Lines

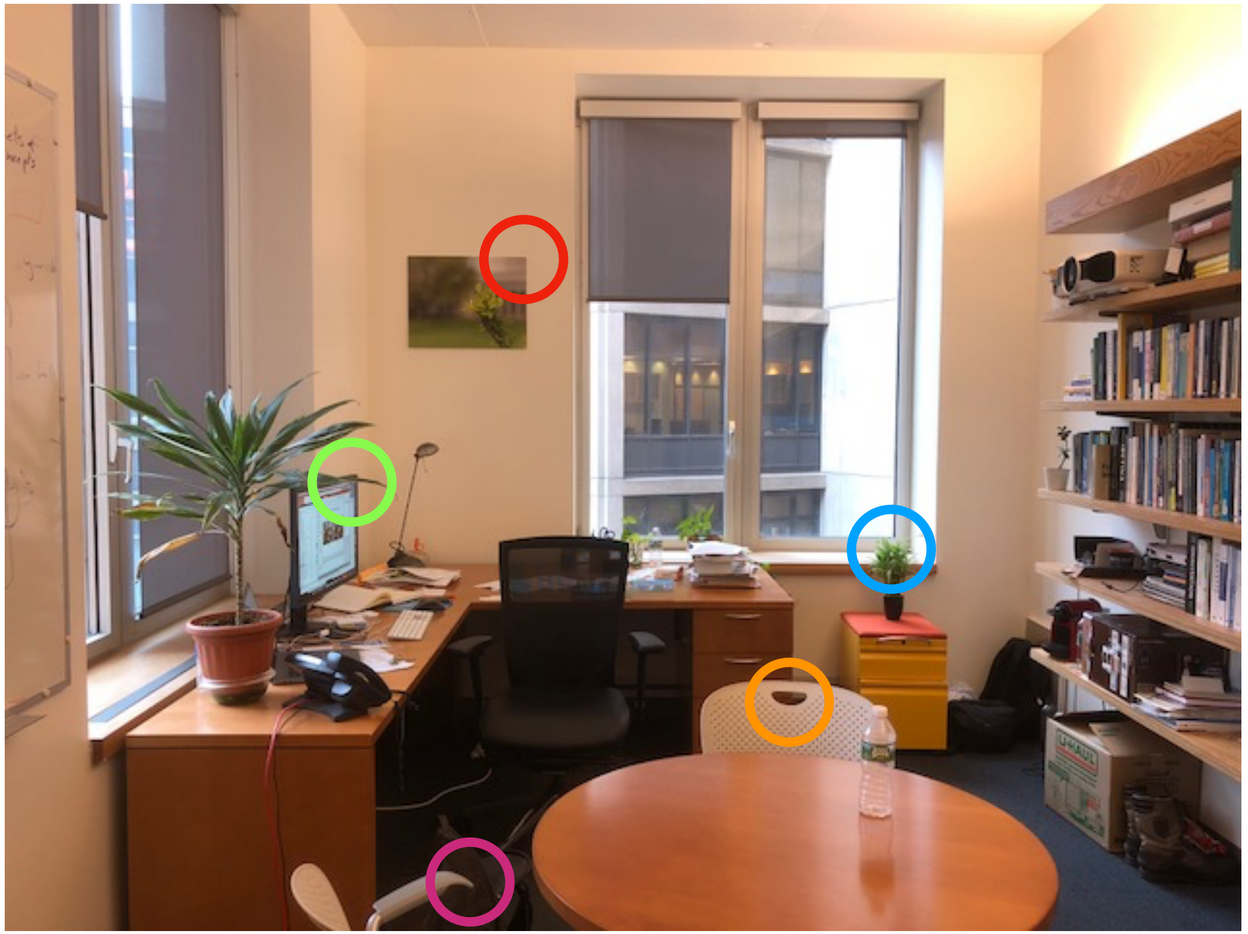
Where to look for the match?

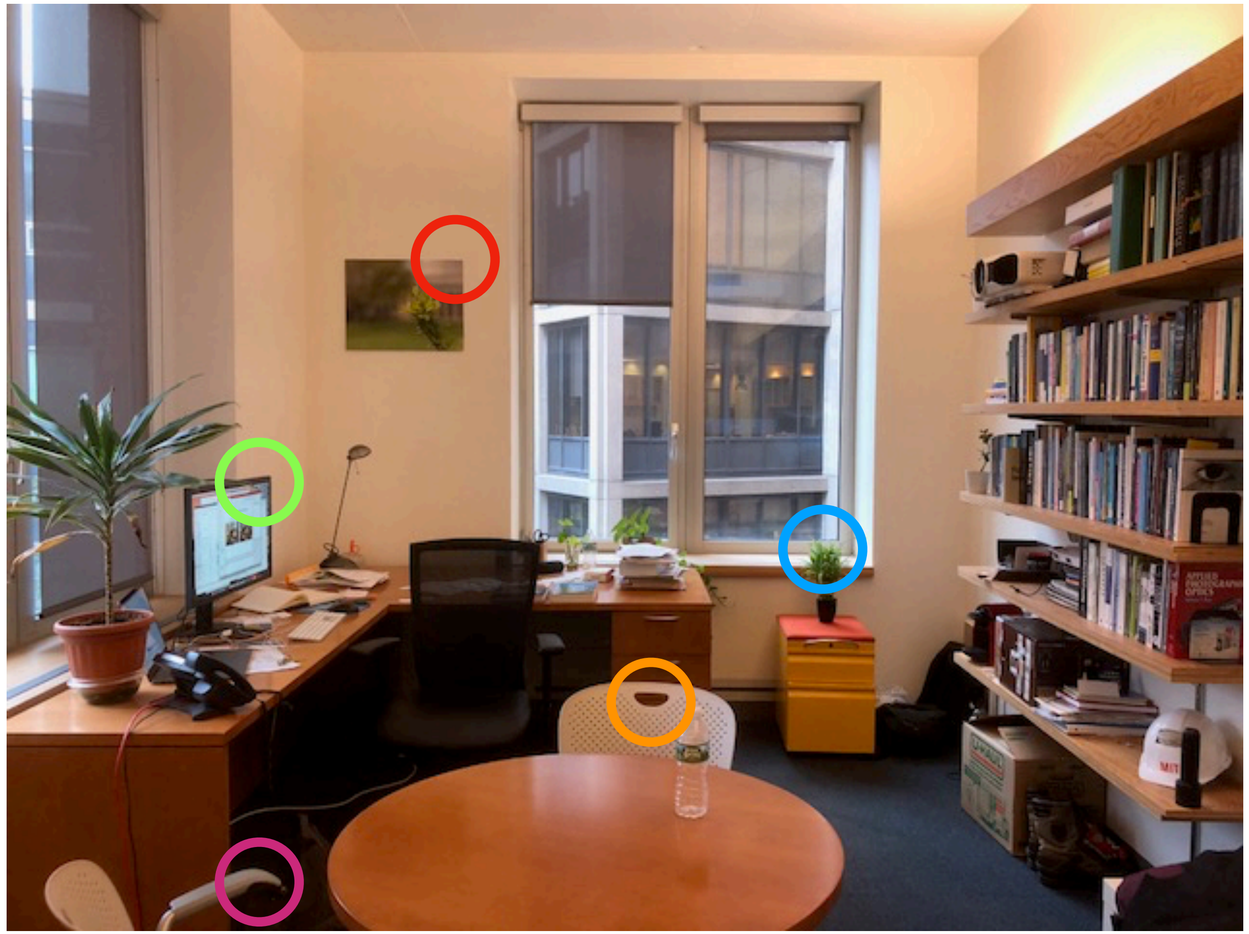
$O(n^2)$ search problem…

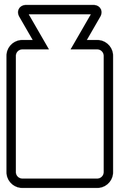




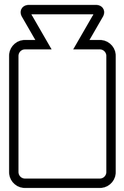
Motivation: Epipolar Lines



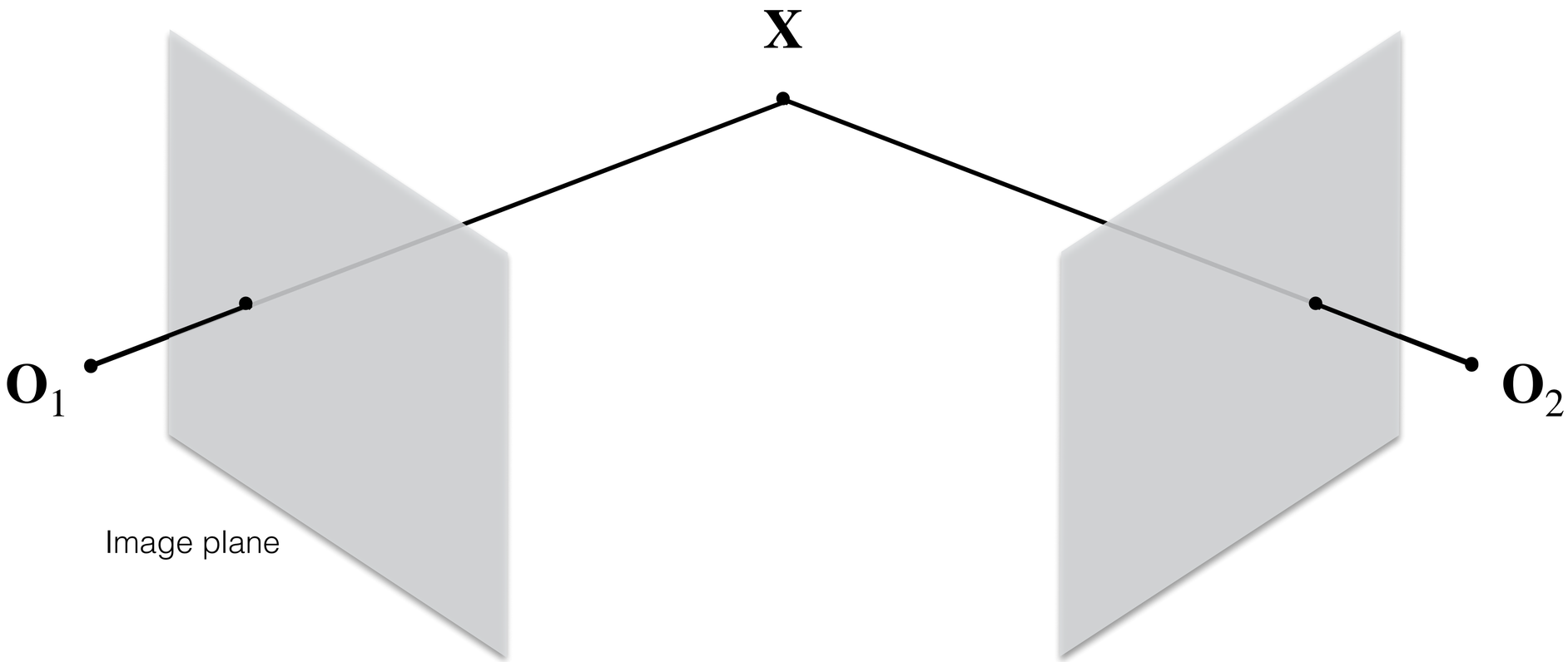




What if we know correspondences, but $\mathbf{P}_1, \mathbf{P}_2$ *aren't* known?

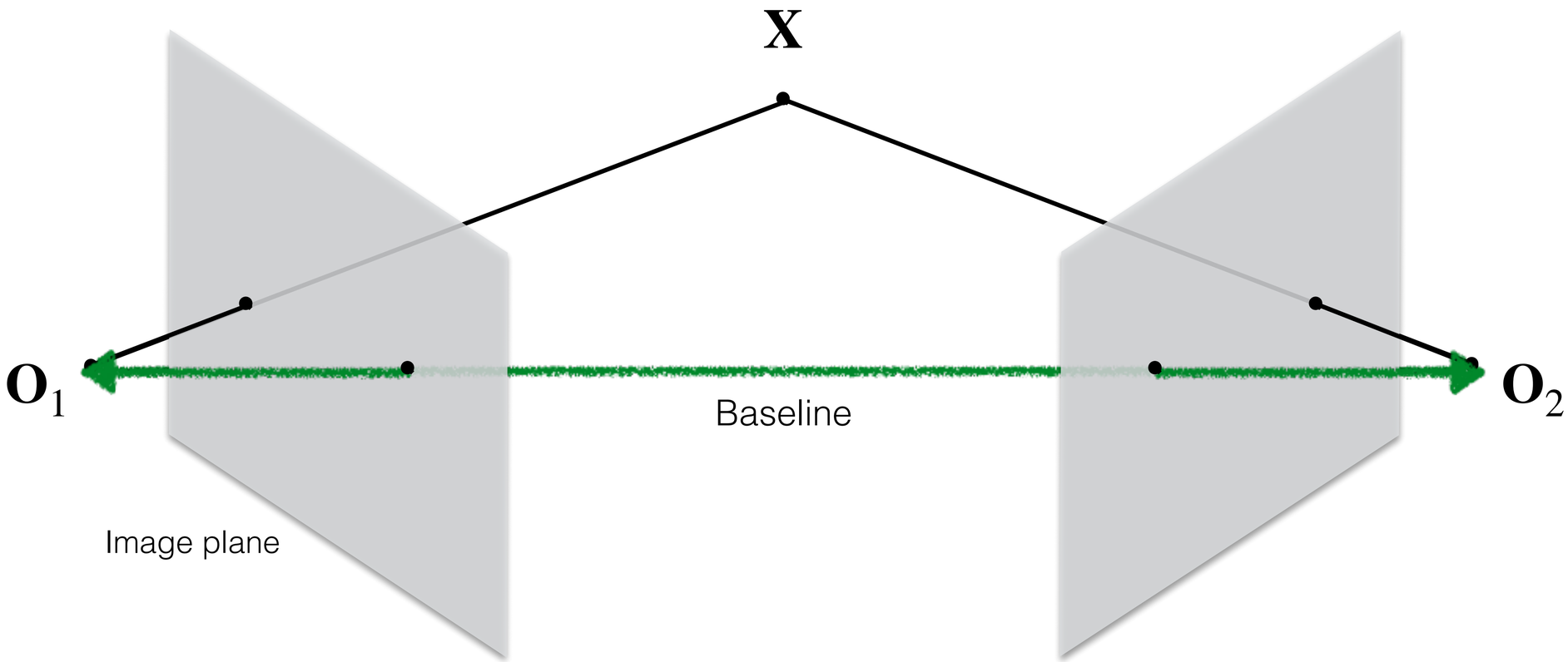


Epipolar geometry



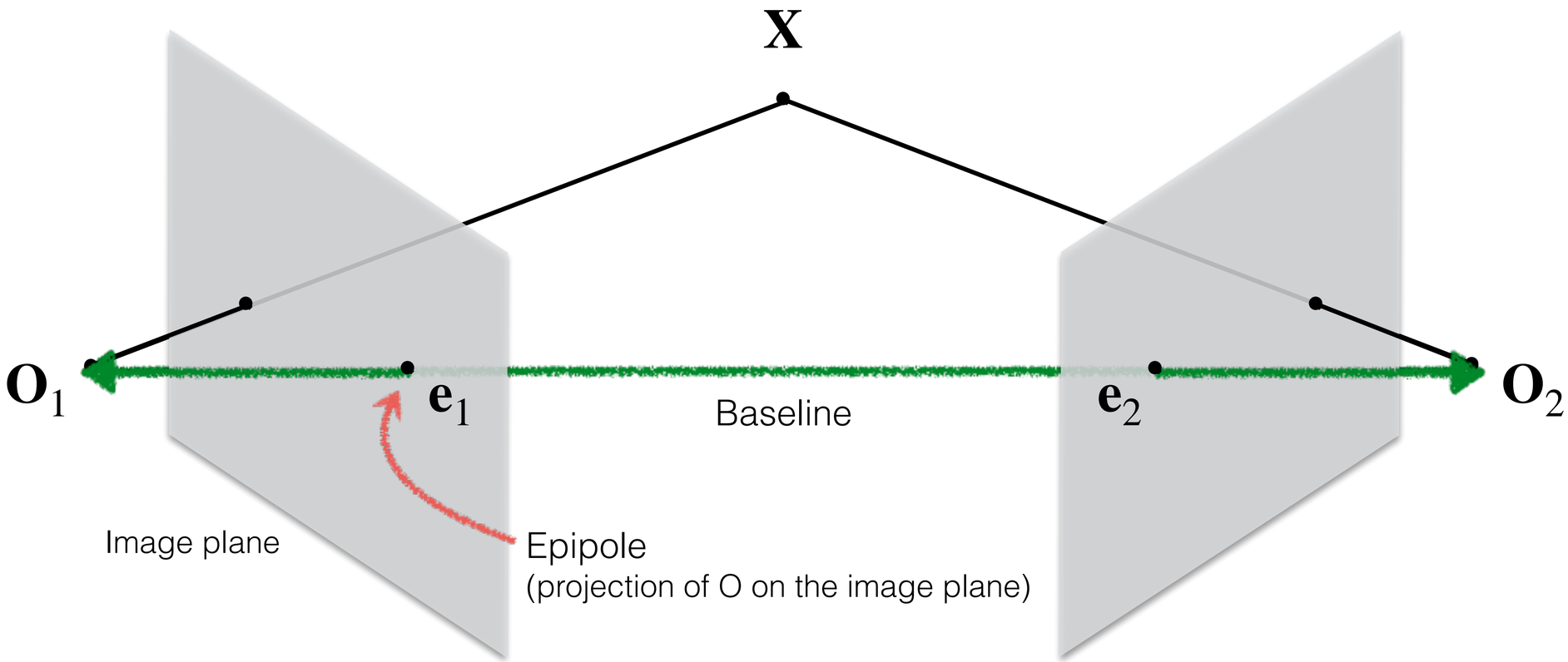
Adapted from: CMU 16-385 (Yannis, Kris)

Epipolar geometry



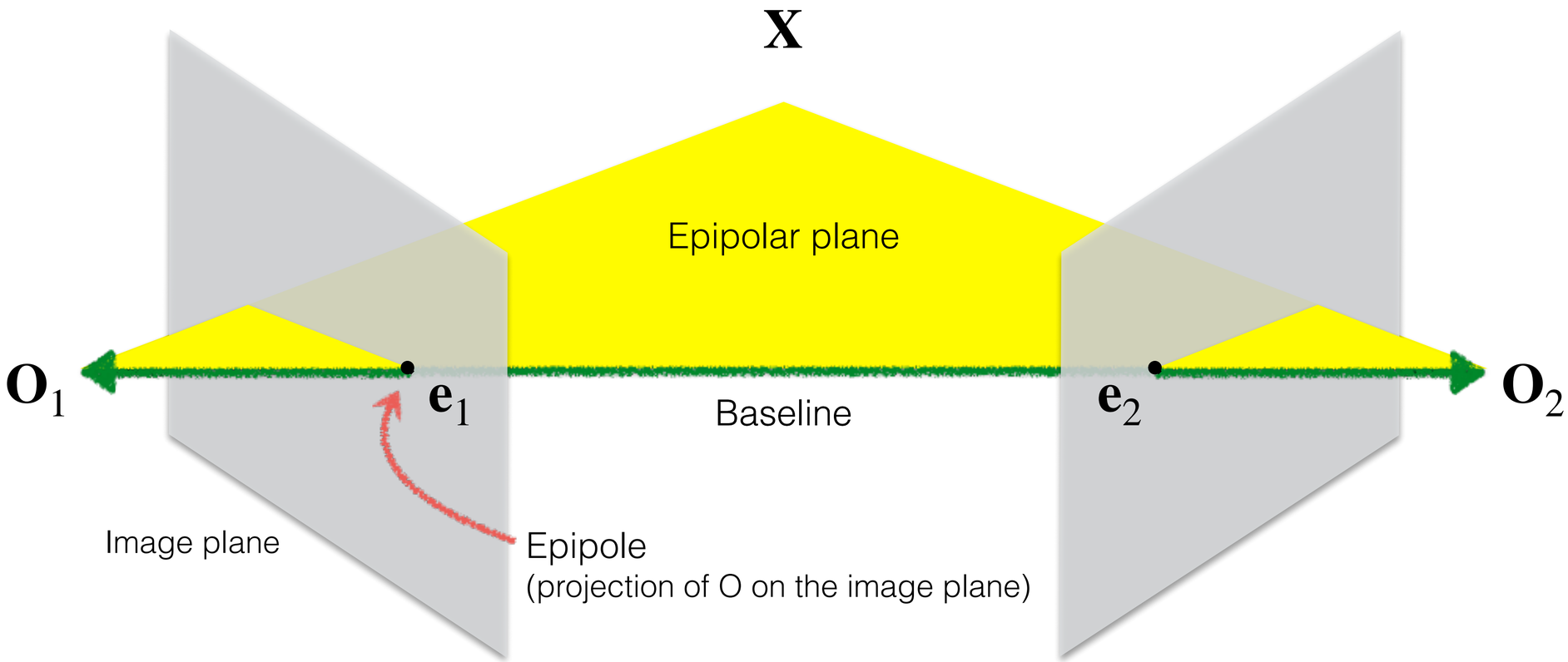
Adapted from: CMU 16-385 (Yannis, Kris)

Epipolar geometry



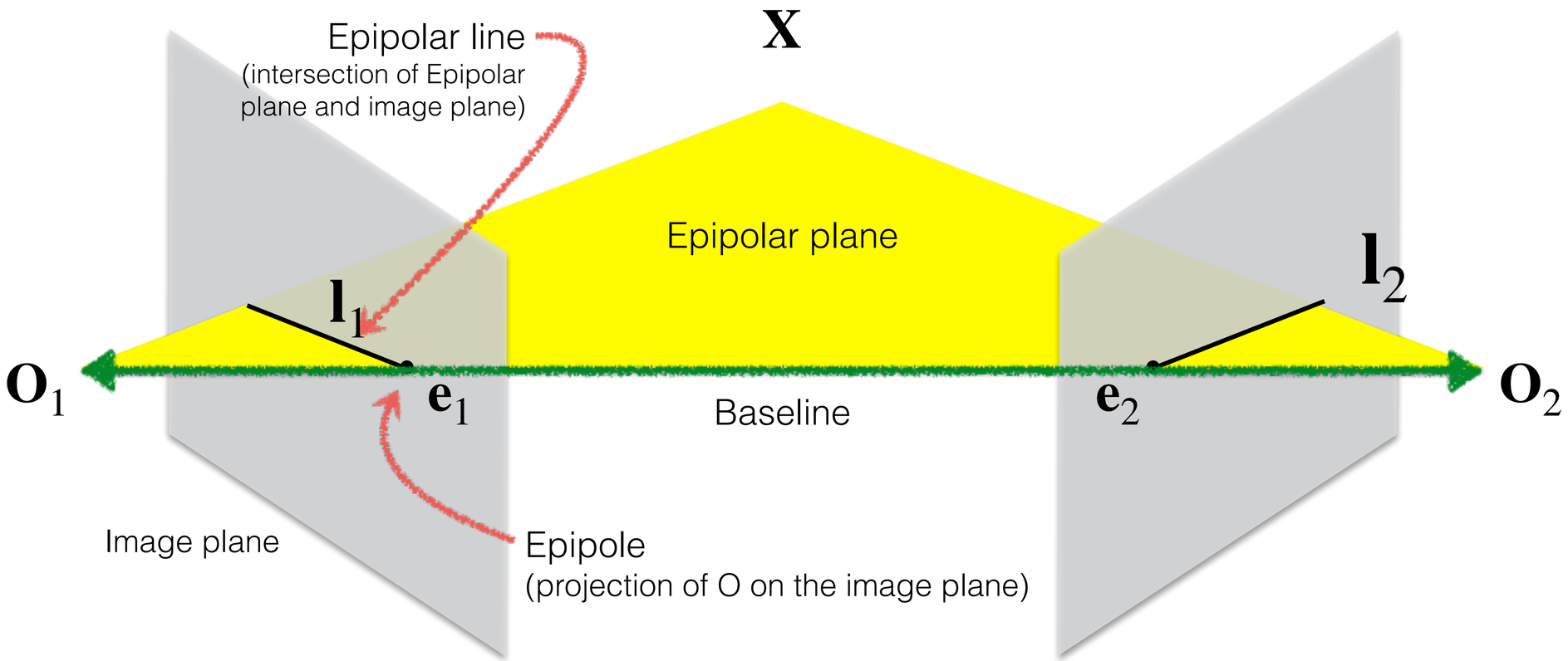
Adapted from: CMU 16-385 (Yannis, Kris)

Epipolar geometry



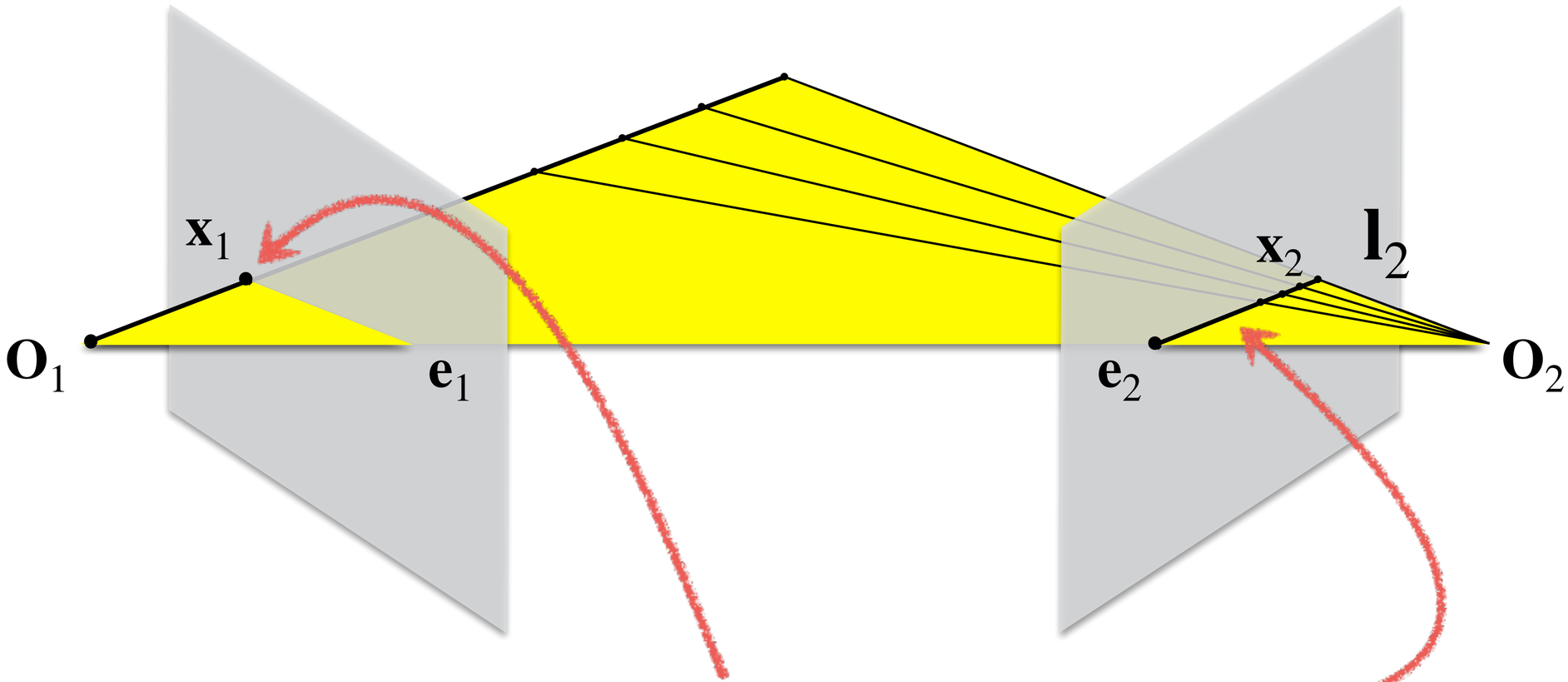
Adapted from: CMU 16-385 (Yannis, Kris)

Epipolar geometry



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

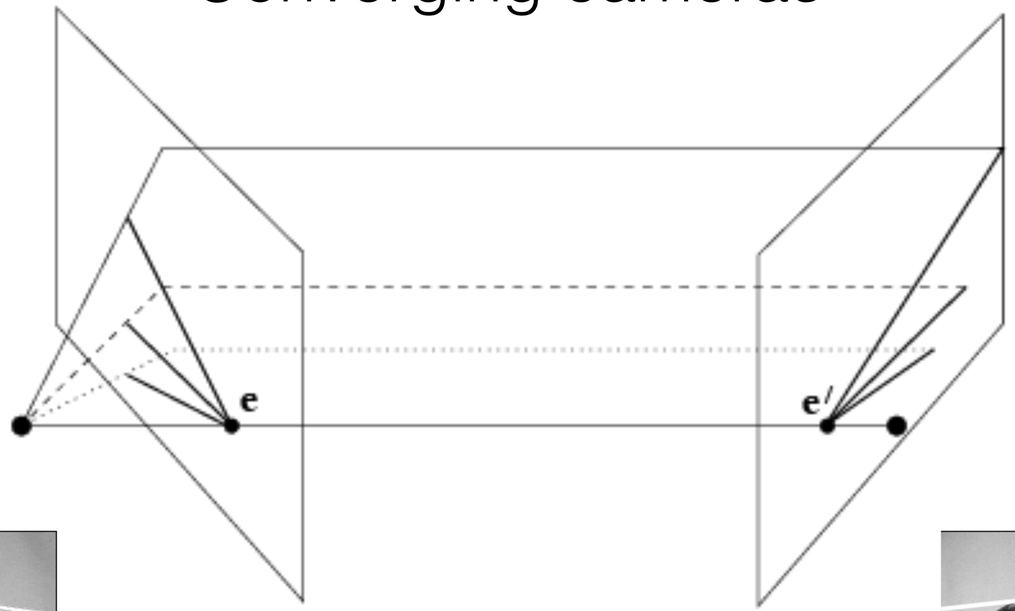
Epipolar constraint

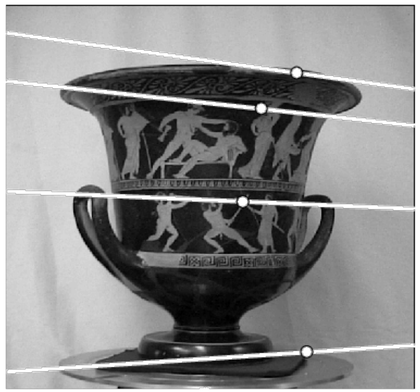


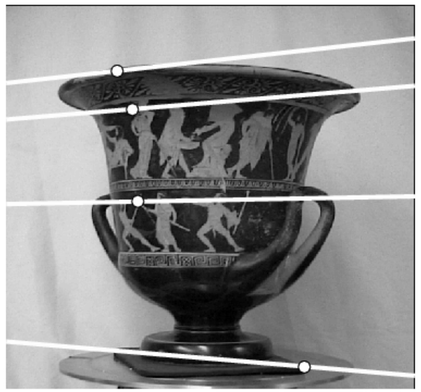
Potential matches for $\mathbf{x}_1$ lie on the epipolar line $\mathbf{l}_2$

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

Converging cameras



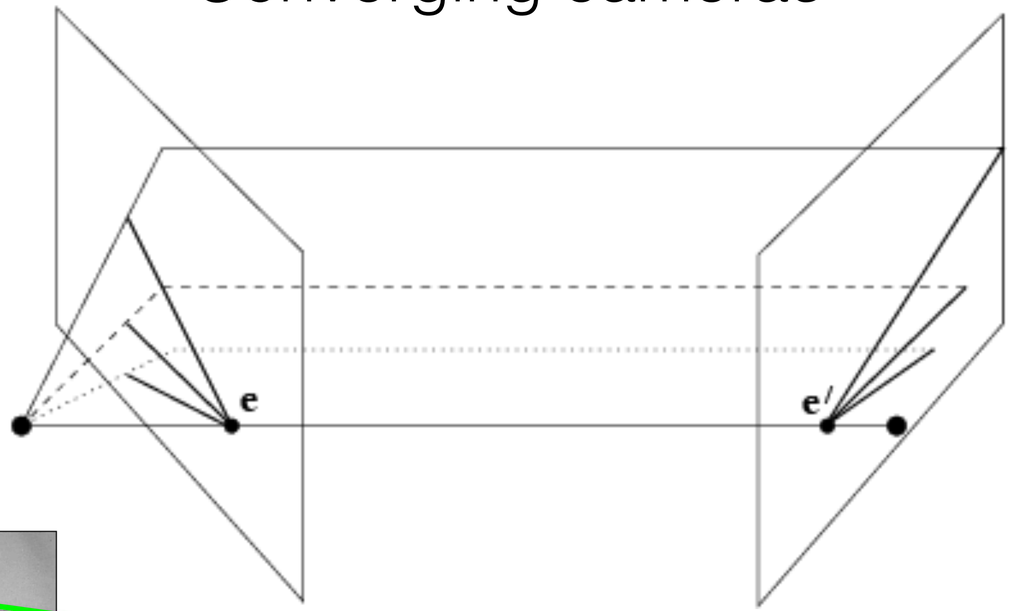


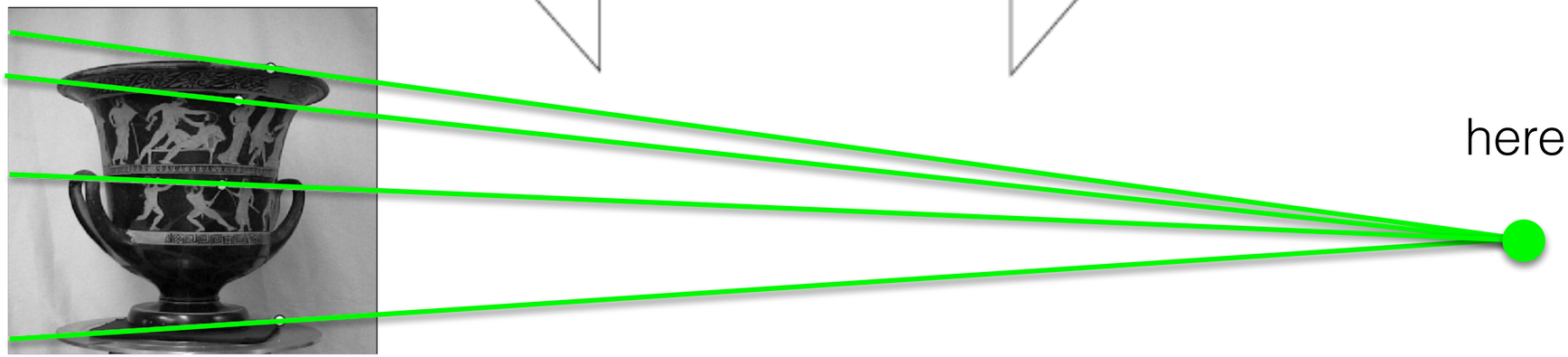


Where is the epipole in this image?

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

Converging cameras





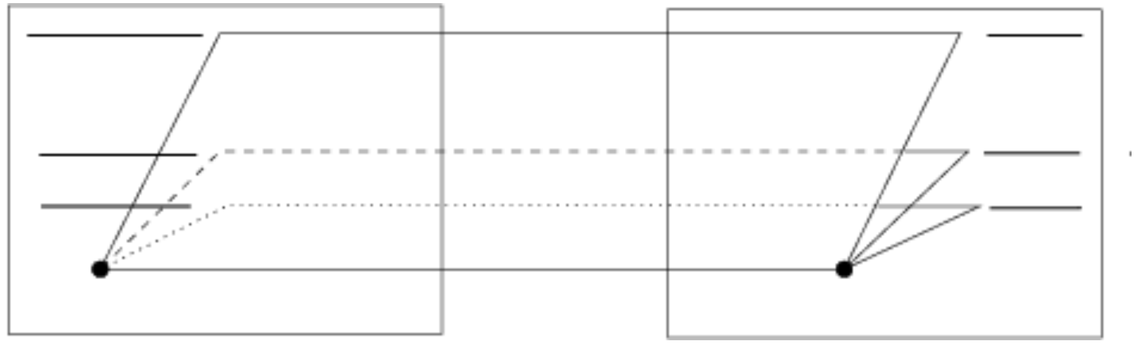
here!

Where is the epipole in this image?

It's not always in the image

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

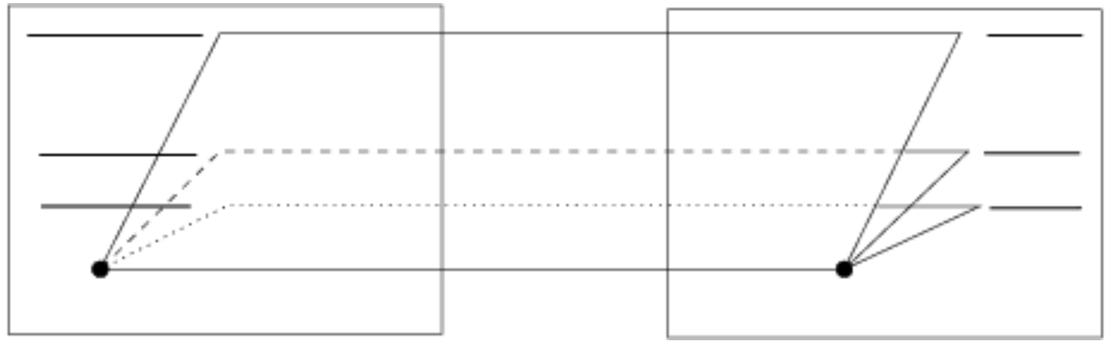
Parallel cameras

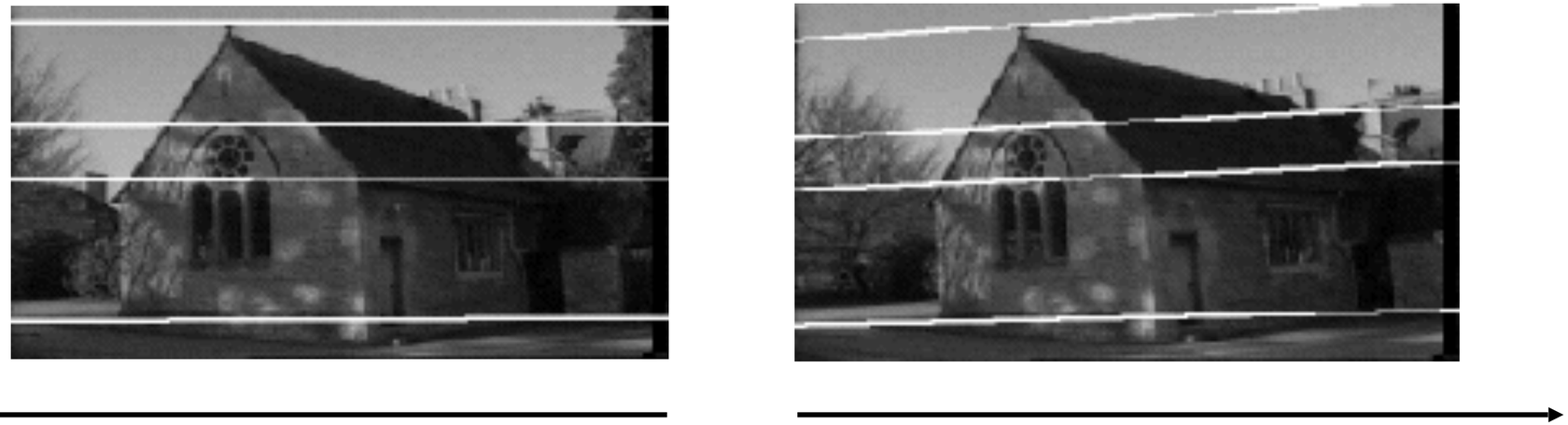


Where is the epipole?

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

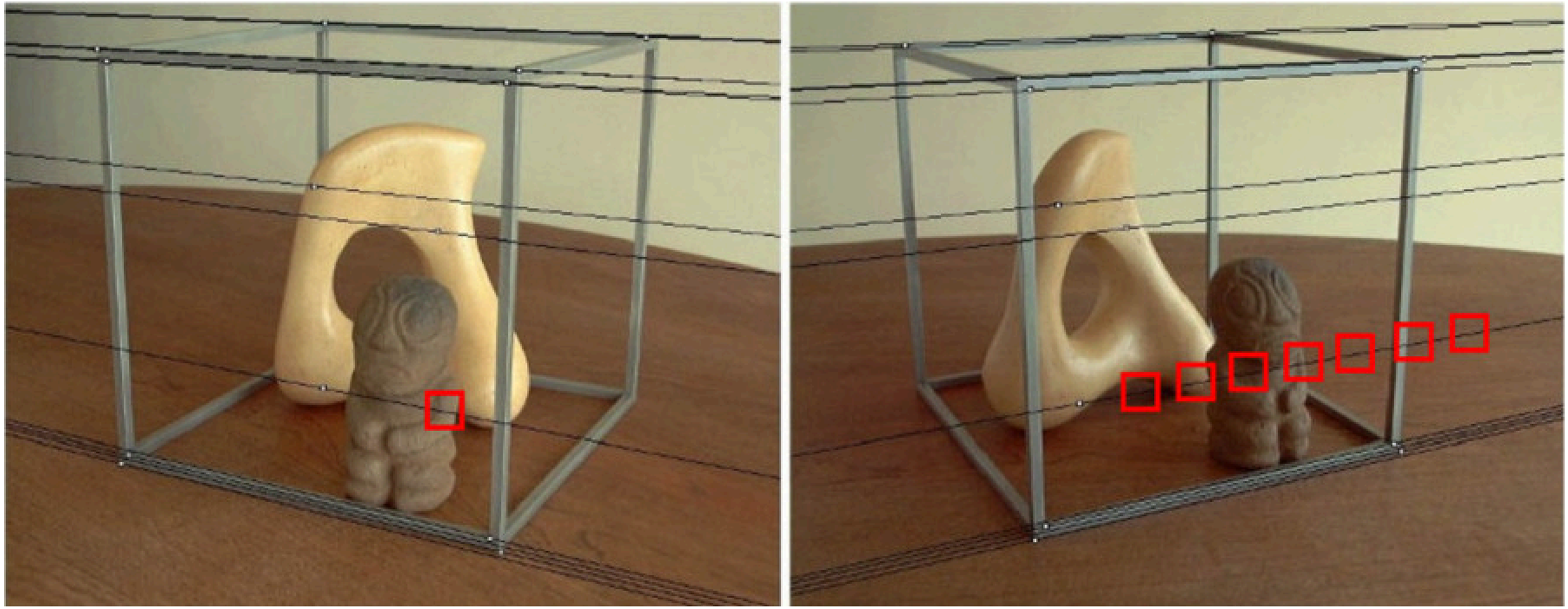
Parallel cameras





epipole at infinity

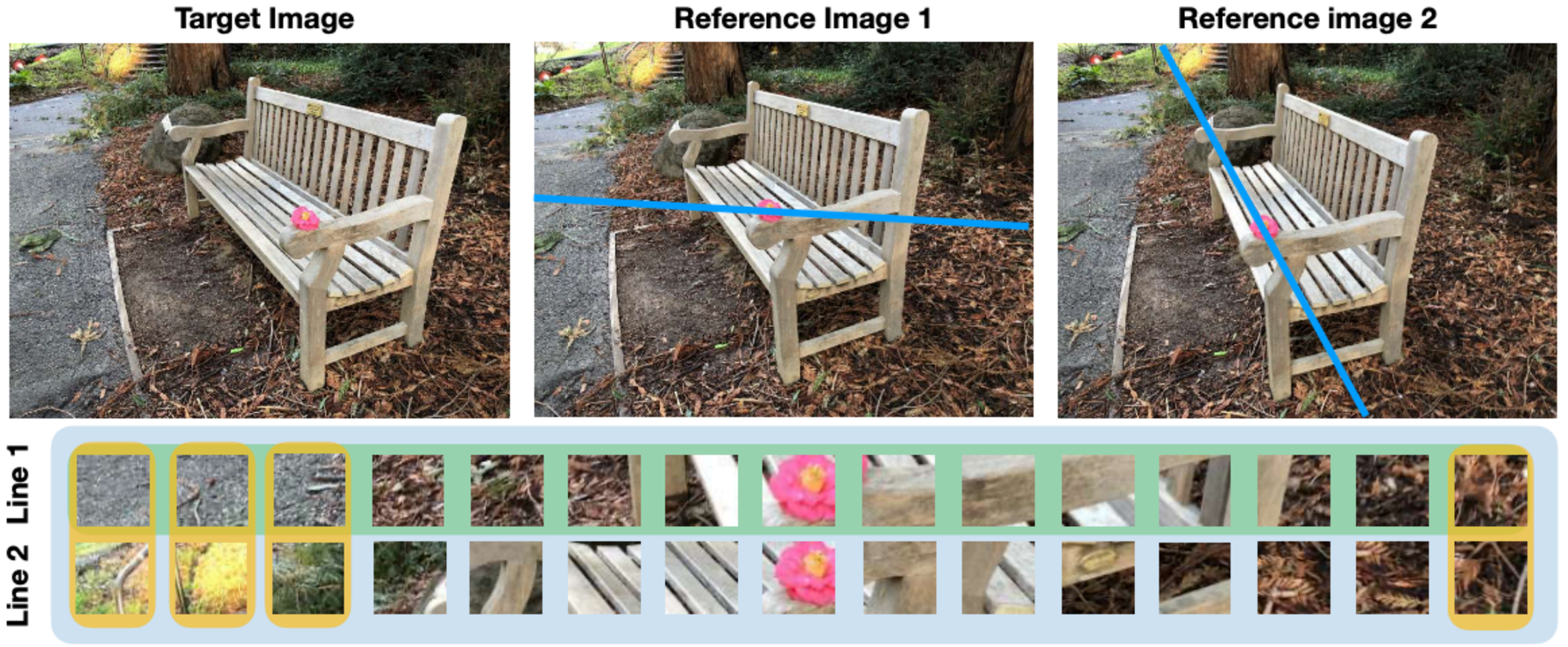
Adapted from: CMU 16-385 (Yannis, Kris)



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]



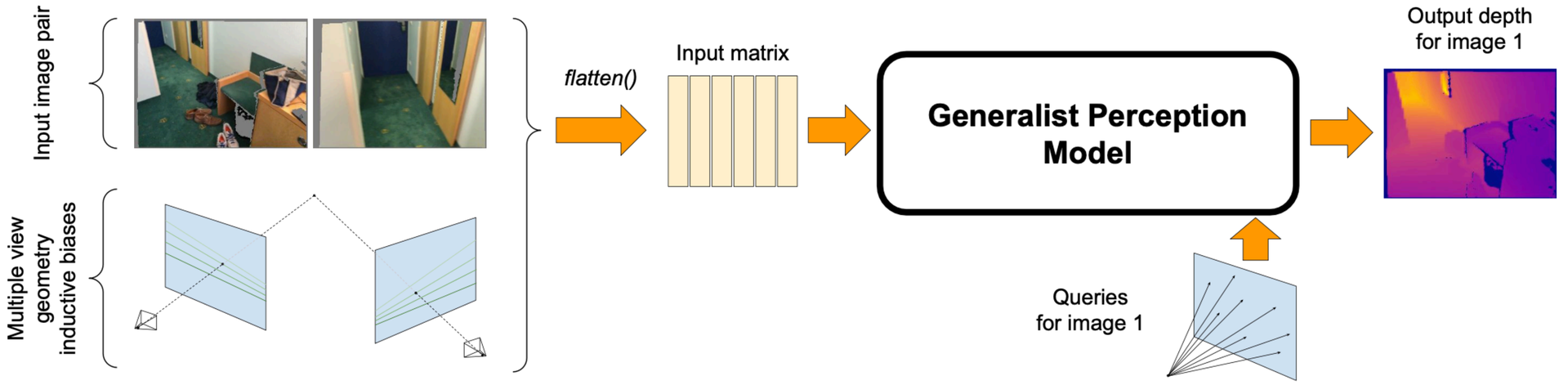
Generalizable Patch-Based Neural Rendering

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

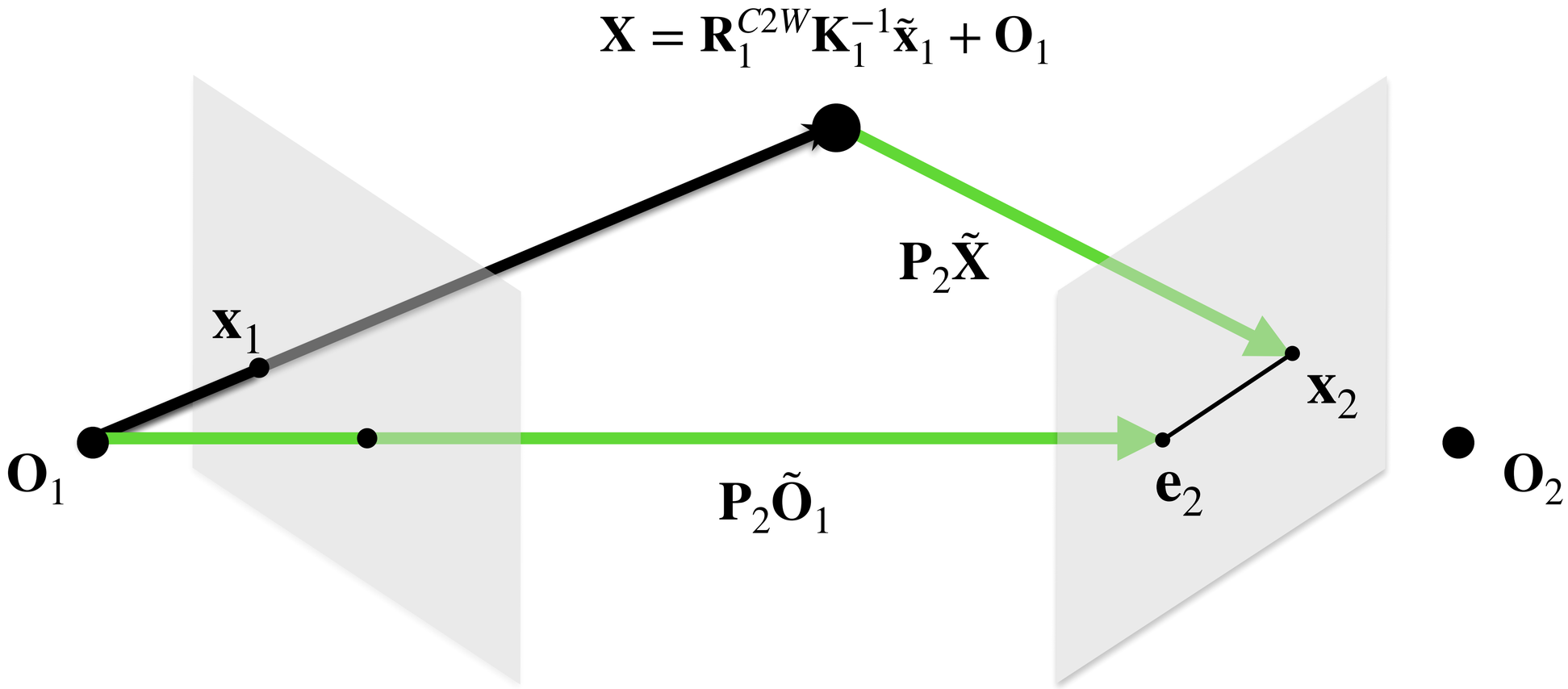
Input-level Inductive Biases for 3D Reconstruction

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

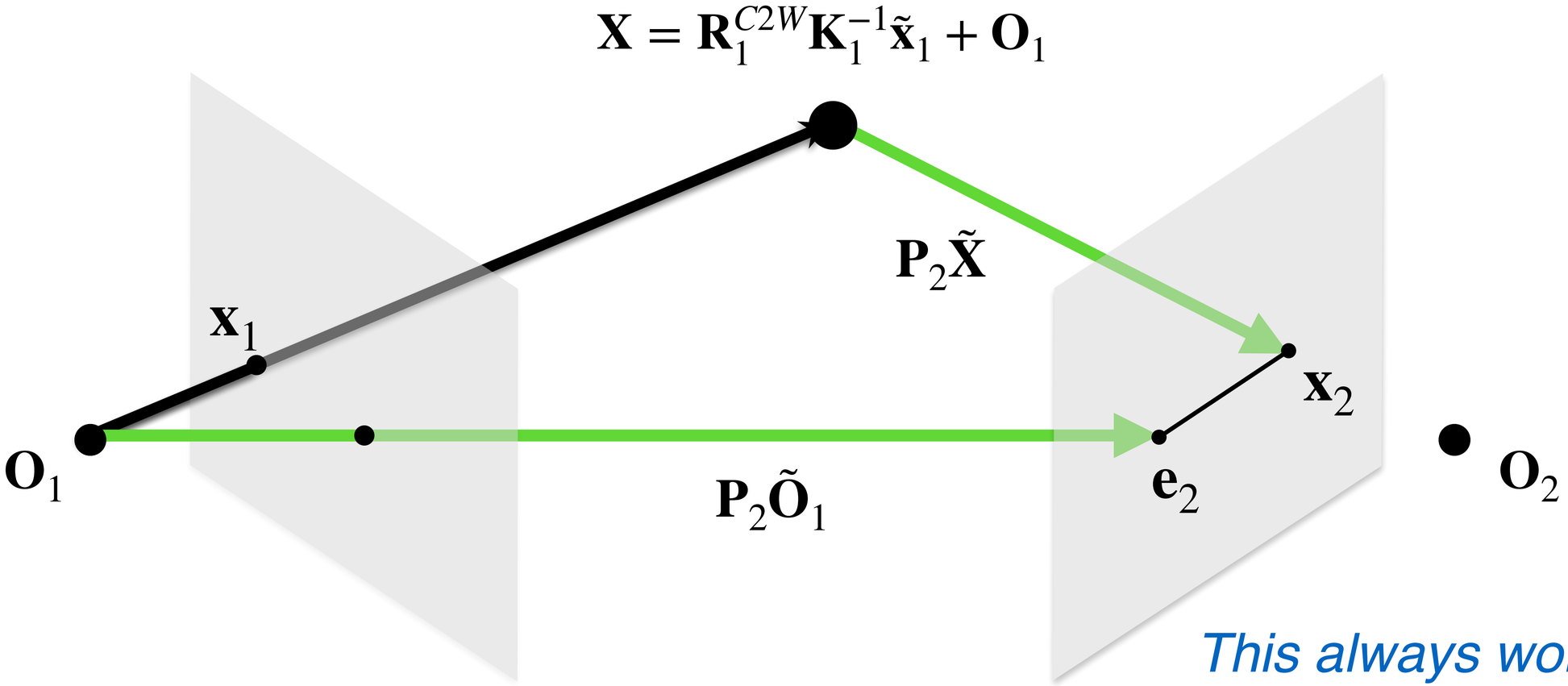
[1]ETH Zurich [2]DeepMind [3]VGG, Department of Engineering Science, University of Oxford



Epipolar Lines: The Hacky Way

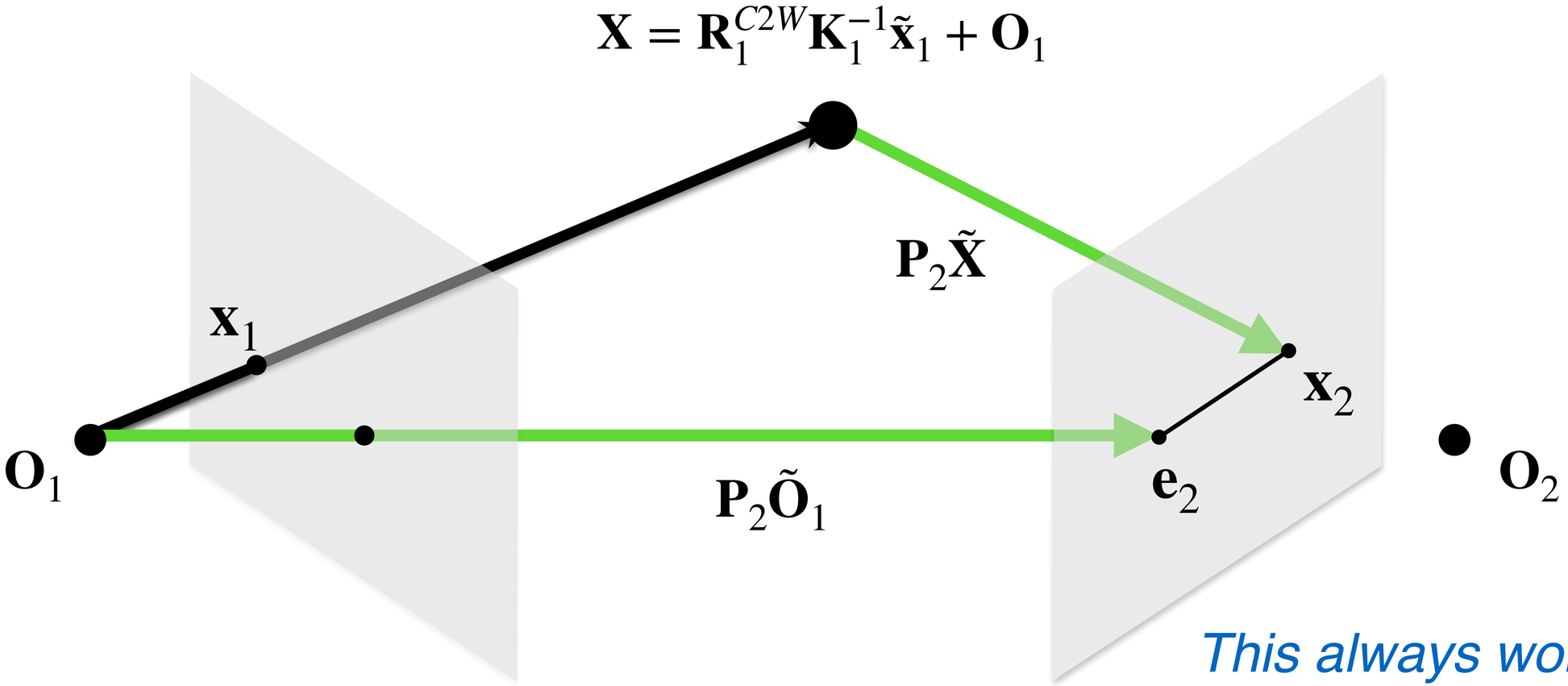


Epipolar Lines: The Hacky Way



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

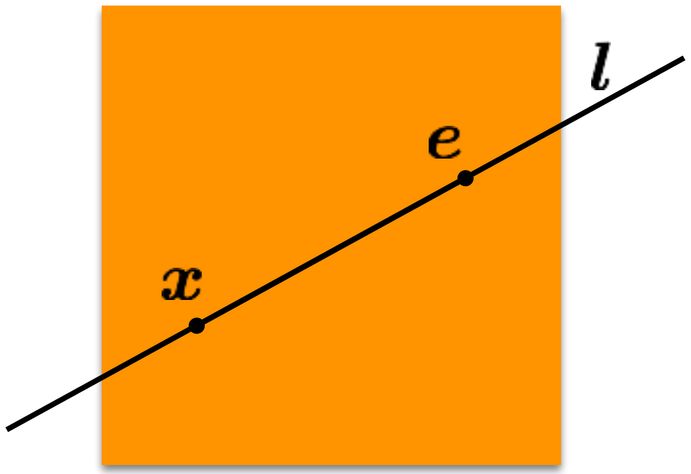
Epipolar Lines: The Hacky Way



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

Lines in Homogeneous Coordinates

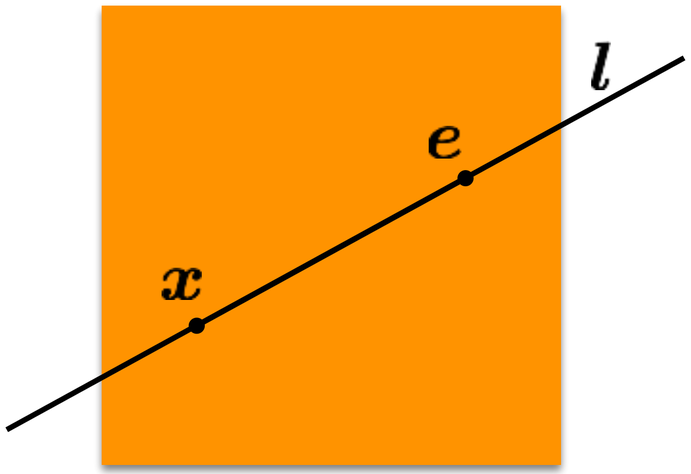
Lines in Homogeneous Coordinates



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

Lines in Homogeneous Coordinates

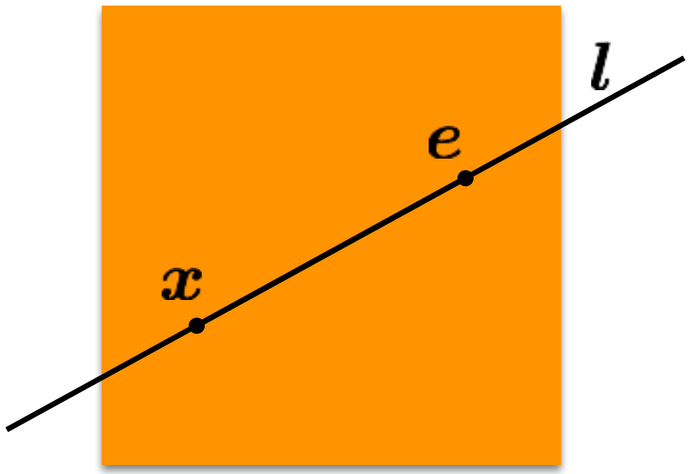
$$ax + by + c = 0$$



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

Lines in Homogeneous Coordinates

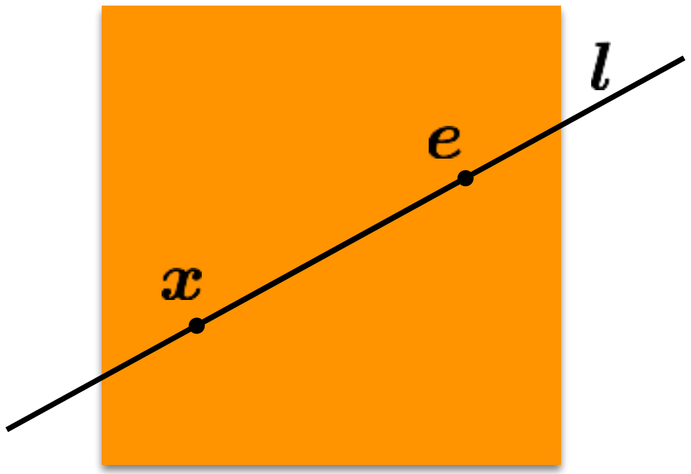
$$ax + by + c = 0 \qquad \text{in vector form} \qquad \mathbf{l} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$



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

Lines in Homogeneous Coordinates

$$ax + by + c = 0$$ in vector form $$\mathbf{l} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$



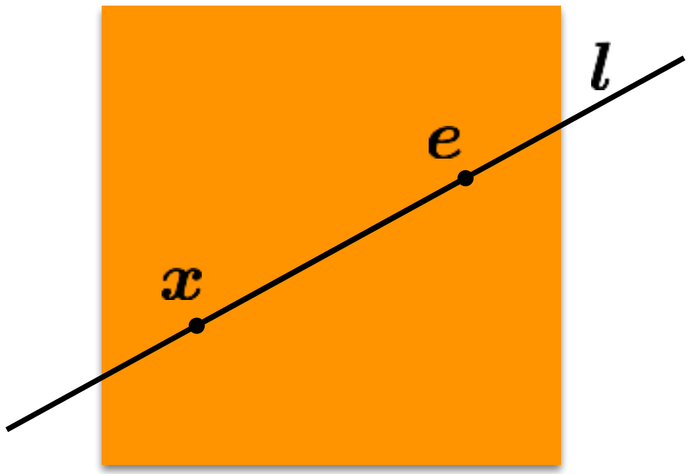
If the point $\mathbf{x}$ is on the epipolar line $\mathbf{l}$ then

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

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

Lines in Homogeneous Coordinates

$$ax + by + c = 0$$ in vector form $$\mathbf{l} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$



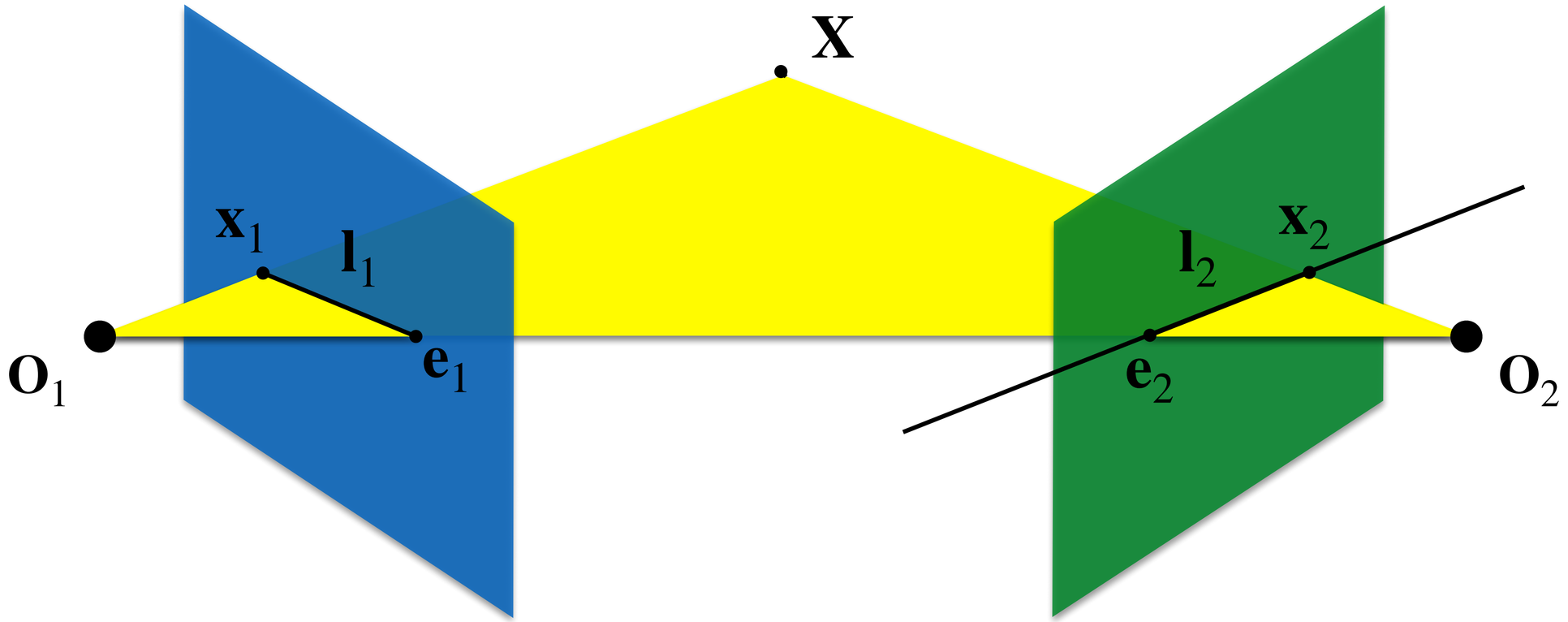
If the point $\mathbf{x}$ is on the epipolar line $\mathbf{l}$ then

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

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

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

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



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

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

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

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

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

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.

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

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.*

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

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

Definition of
Fundamental Matrix

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

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

Definition of
Fundamental Matrix

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

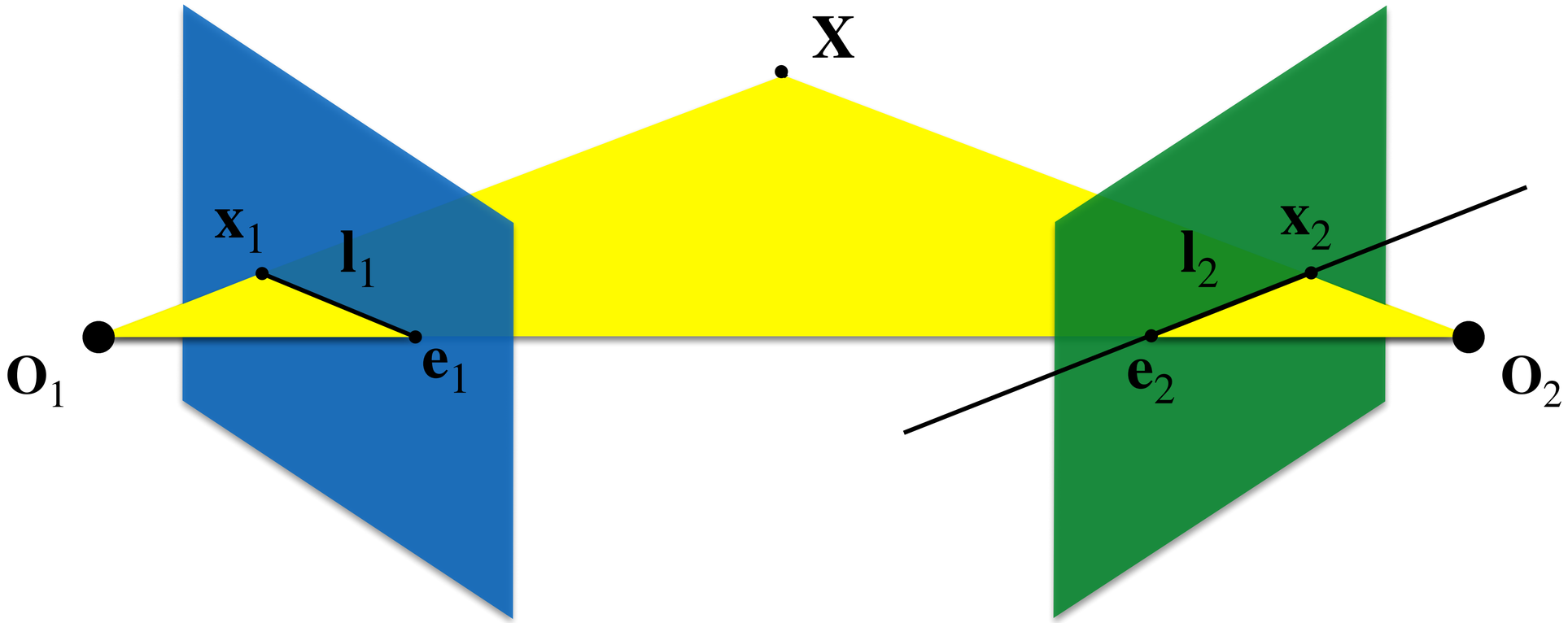
Point $\tilde{\mathbf{x}}_2$ lies on epipolar line $\mathbf{l}_2$

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

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

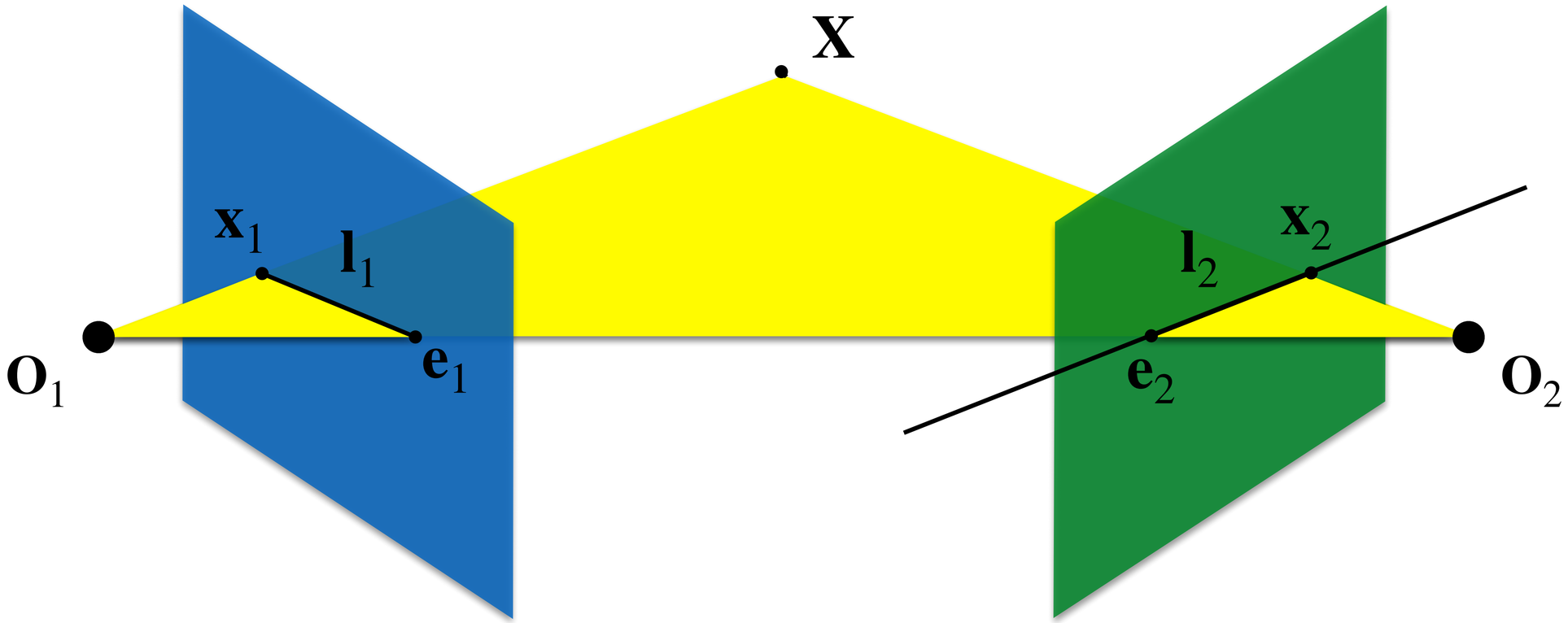
$$\mathbf{F}\tilde{\mathbf{x}}_1 = \mathbf{l}_2$$

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



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

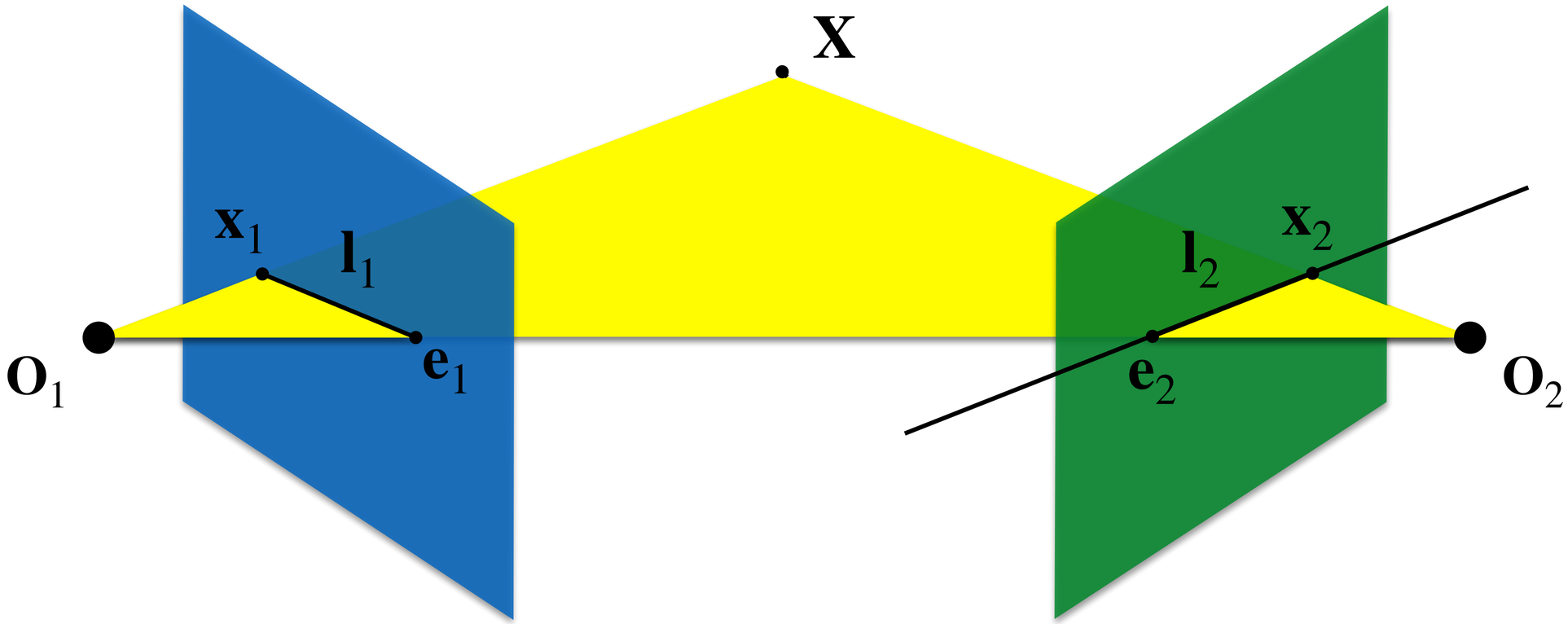
$$\tilde{\mathbf{x}}_2^T \mathbf{F} \tilde{\mathbf{x}}_1 = 0$$



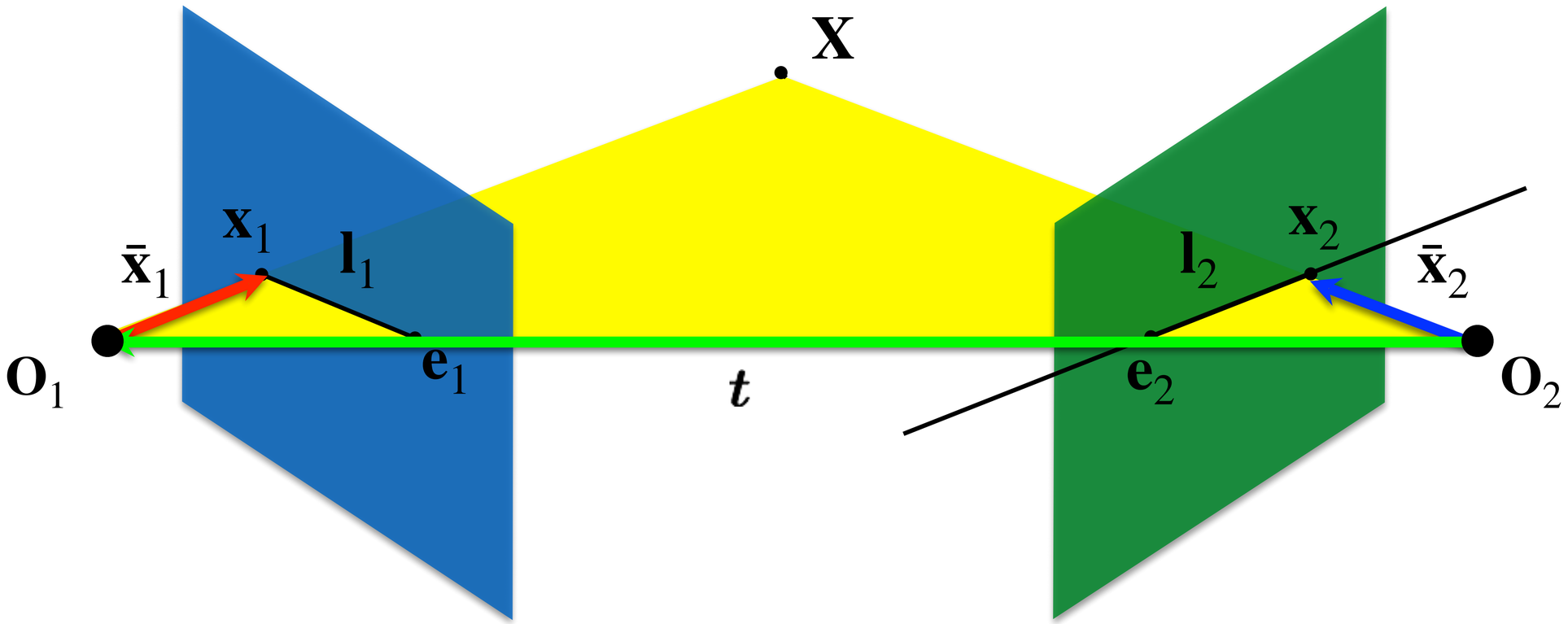
Adapted from: CMU 16-385 (Yannis, Kris)

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

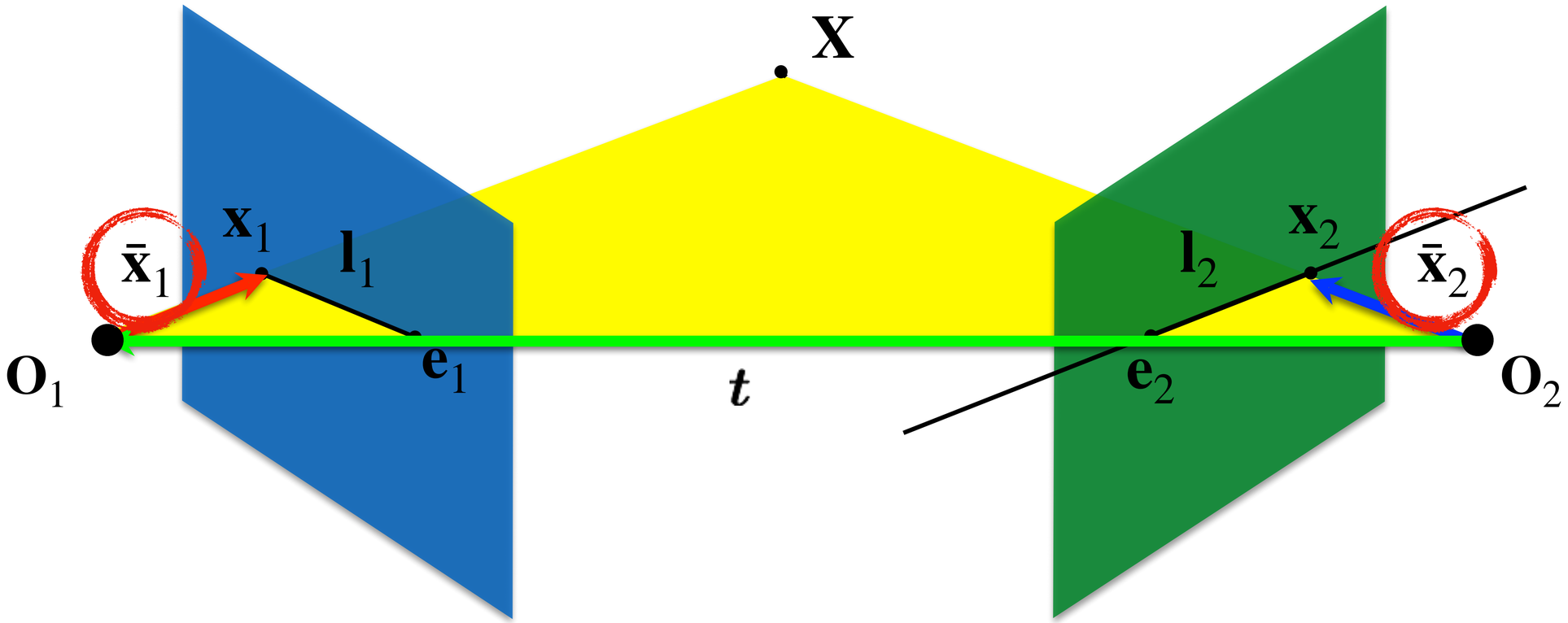
$$\tilde{\mathbf{x}}_2^T \mathbf{F} \tilde{\mathbf{x}}_1 = 0$$



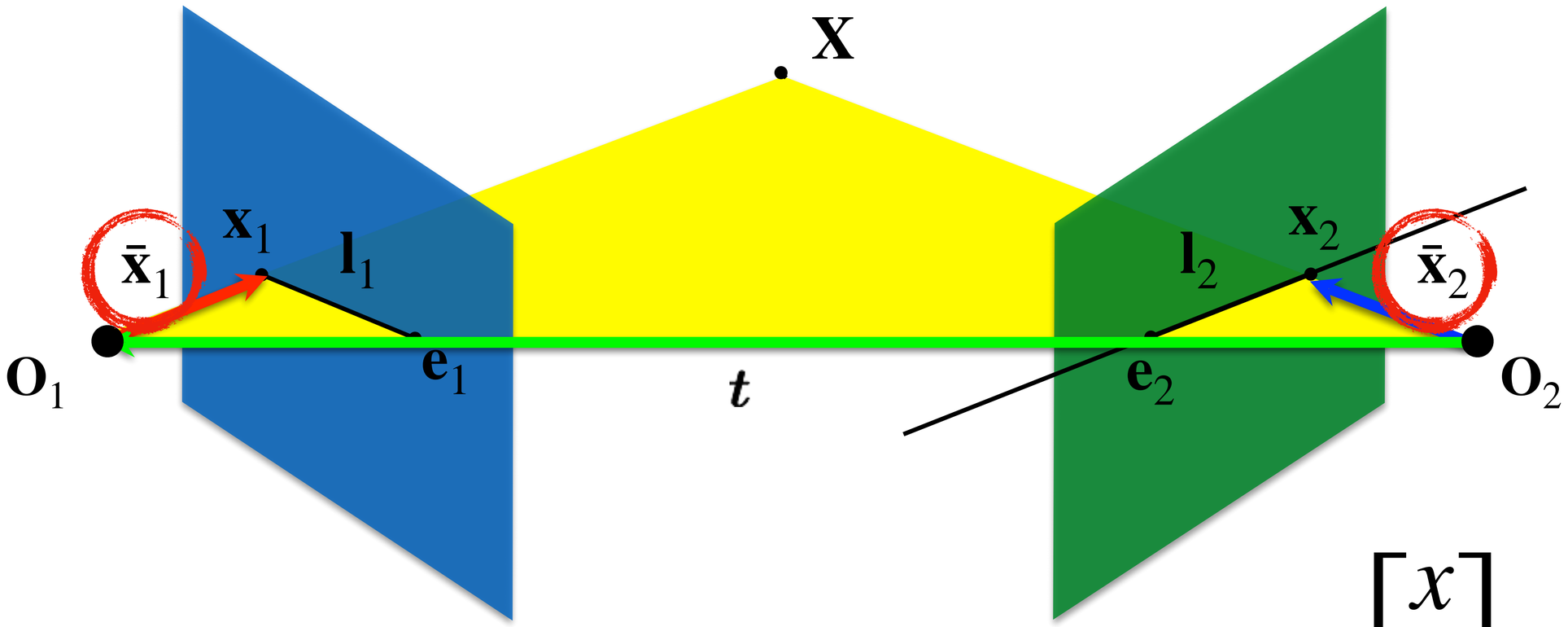
Adapted from: CMU 16-385 (Yannis, Kris)



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



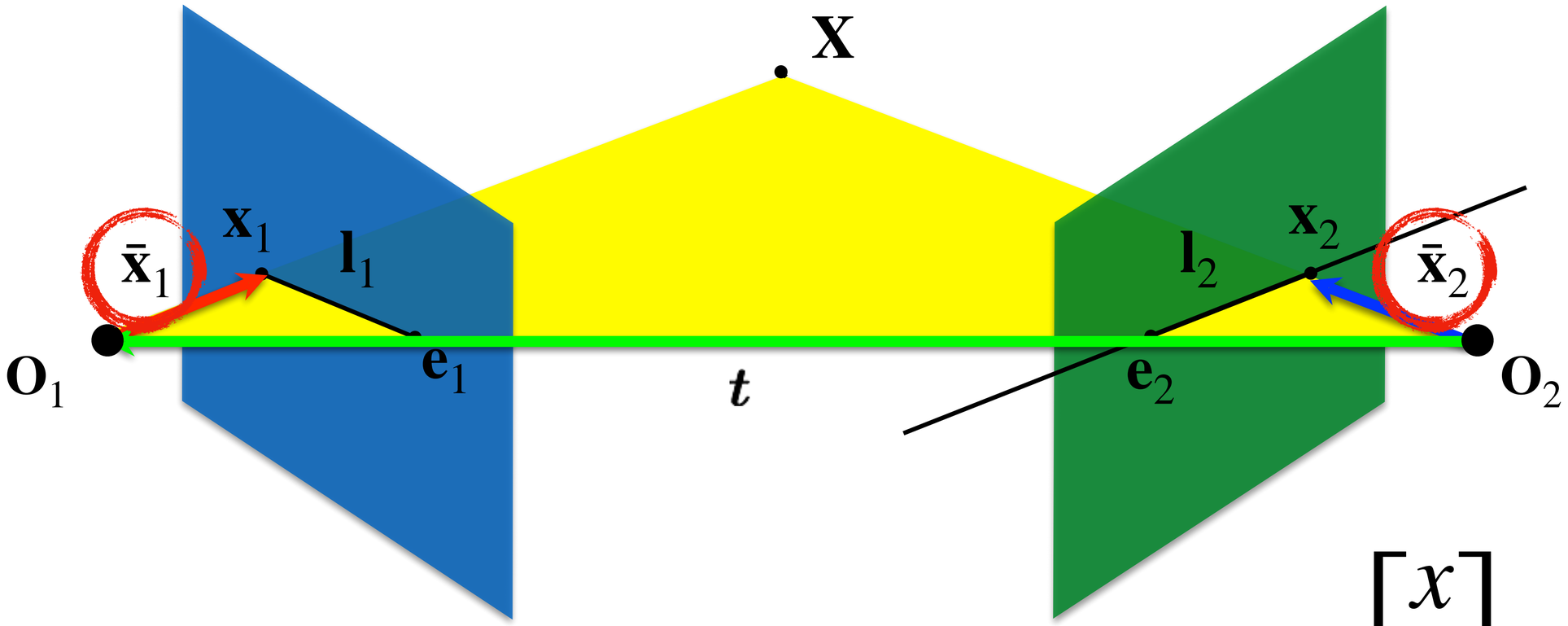
Adapted from: CMU 16-385 (Yannis, Kris)



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

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

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

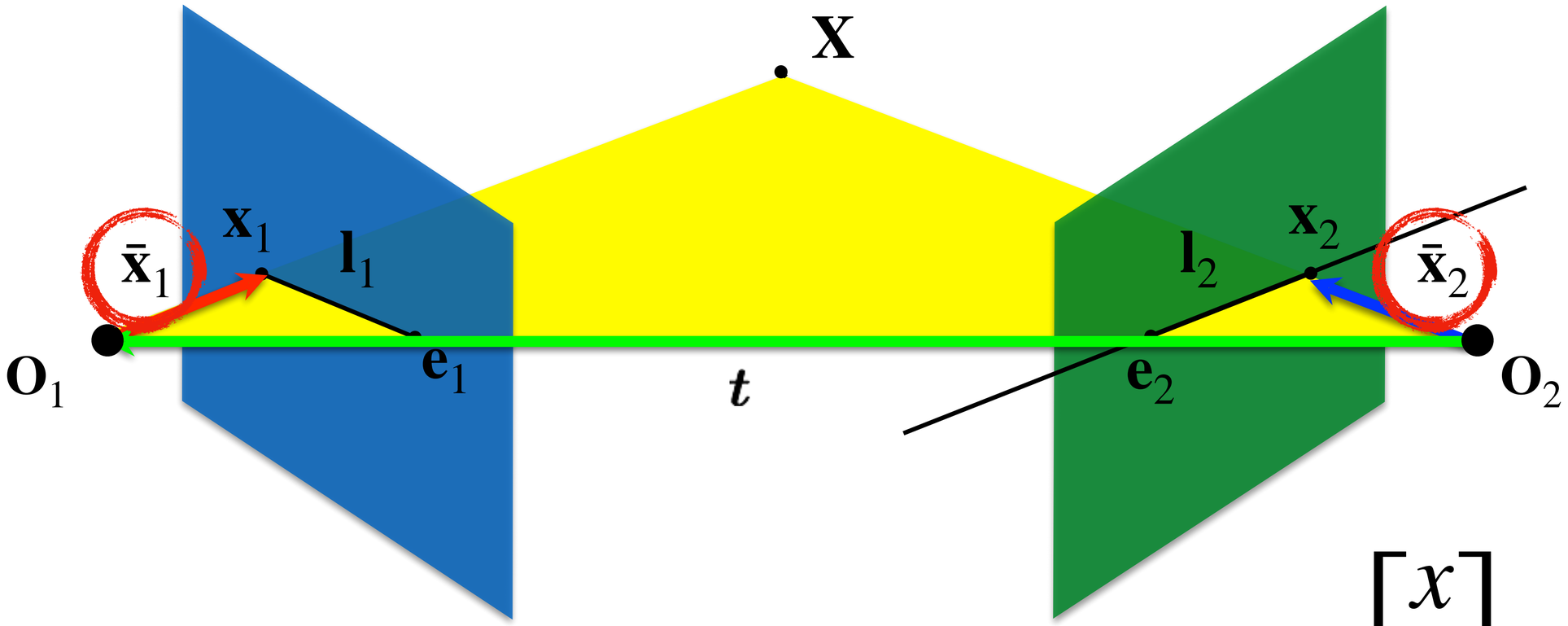


$$\bar{\mathbf{x}} = \mathbf{K}^{-1}\tilde{\mathbf{x}}$$

The local ray direction!

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

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



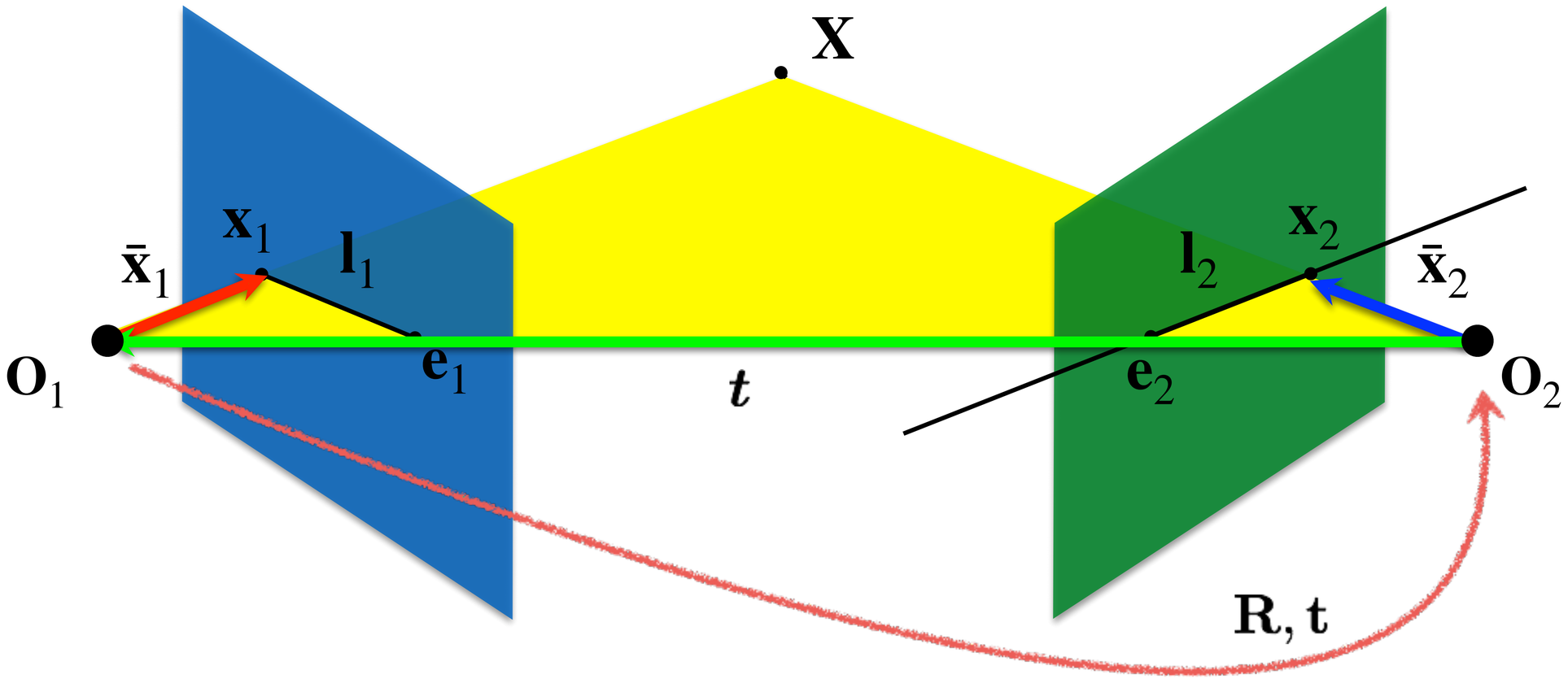
$$\bar{\mathbf{x}} = \mathbf{K}^{-1}\tilde{\mathbf{x}}$$

The local ray direction!

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

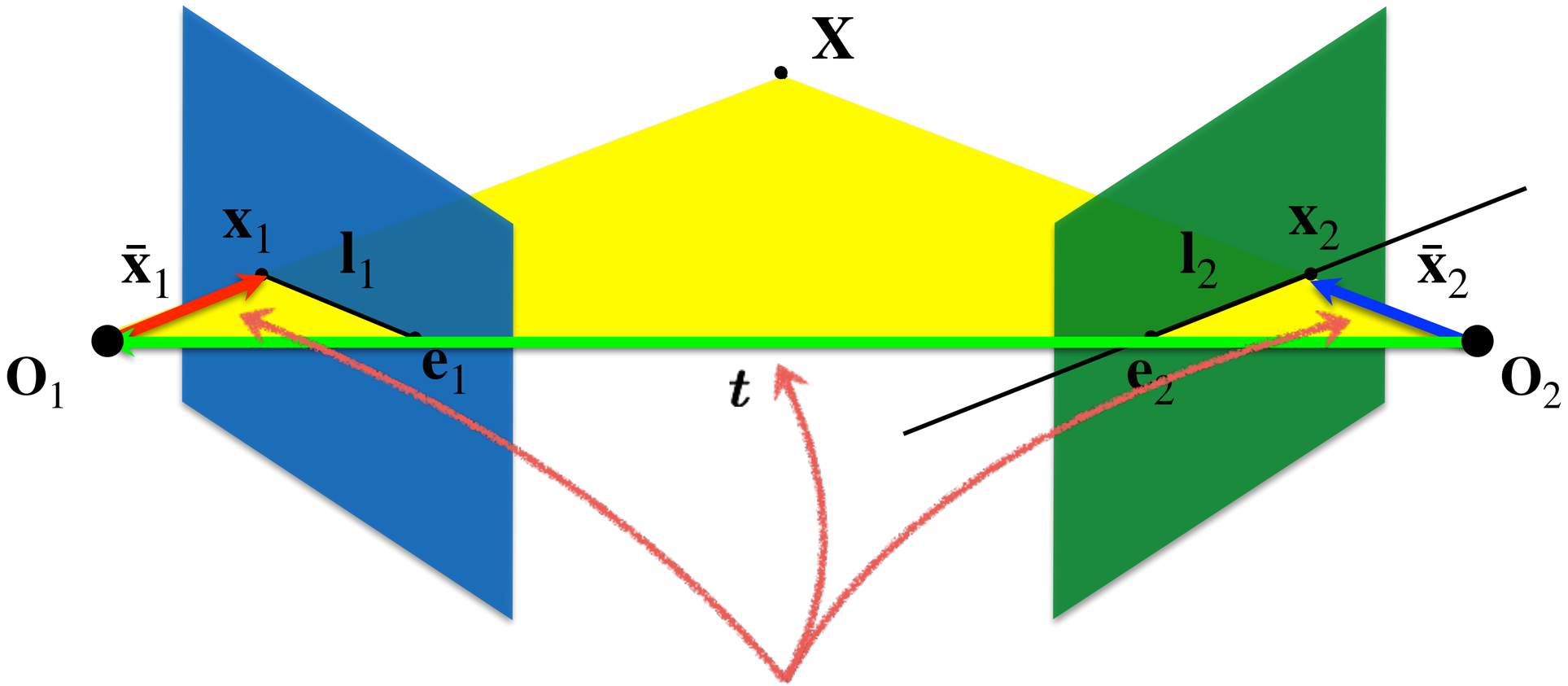
Homogenized pixel coordinate

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



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

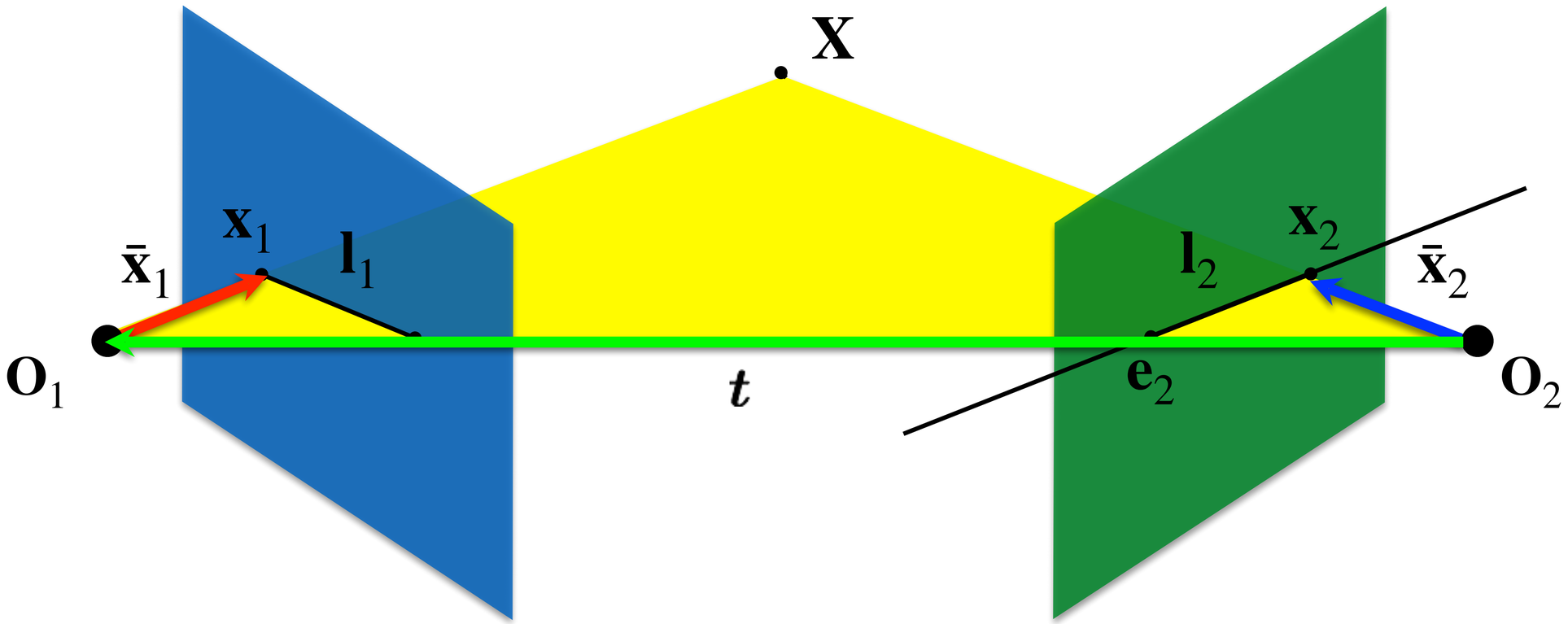
Adapted from: CMU 16-385 (Yannis, Kris)



These three vectors are coplanar

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

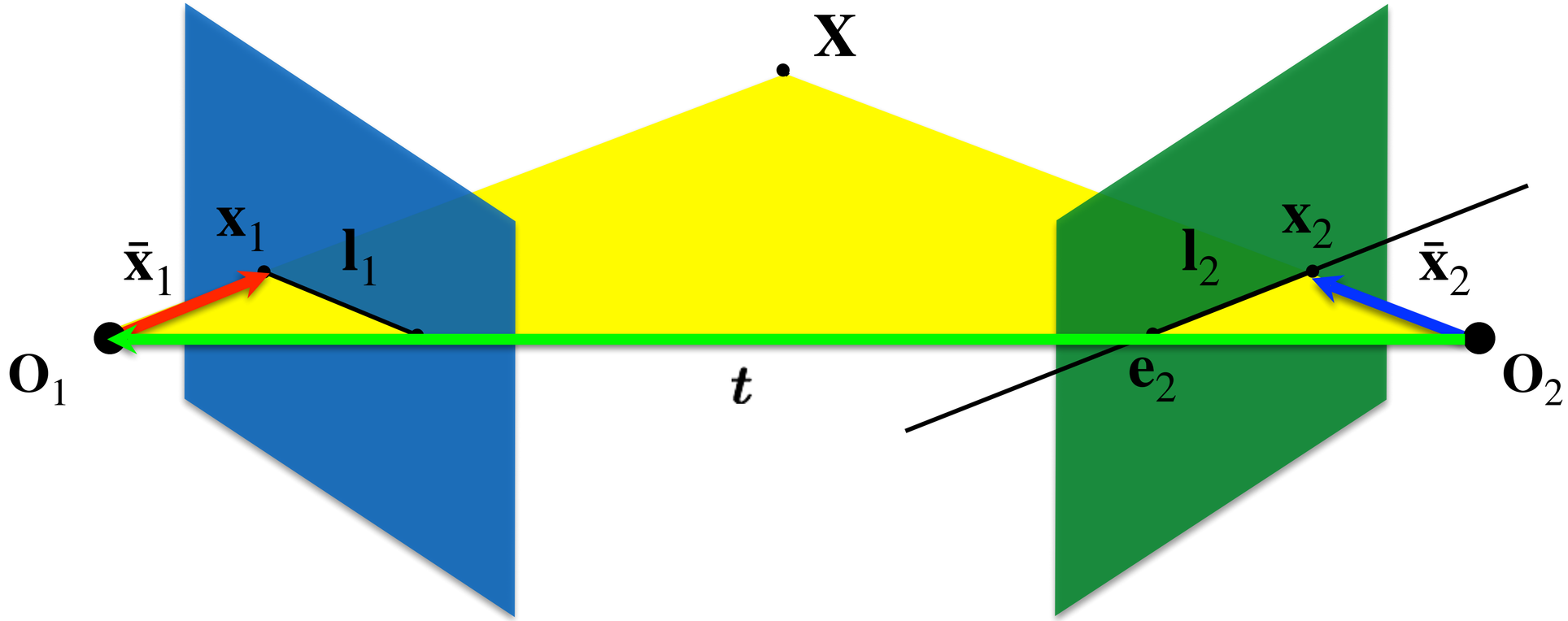
Adapted from: CMU 16-385 (Yannis, Kris)



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

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

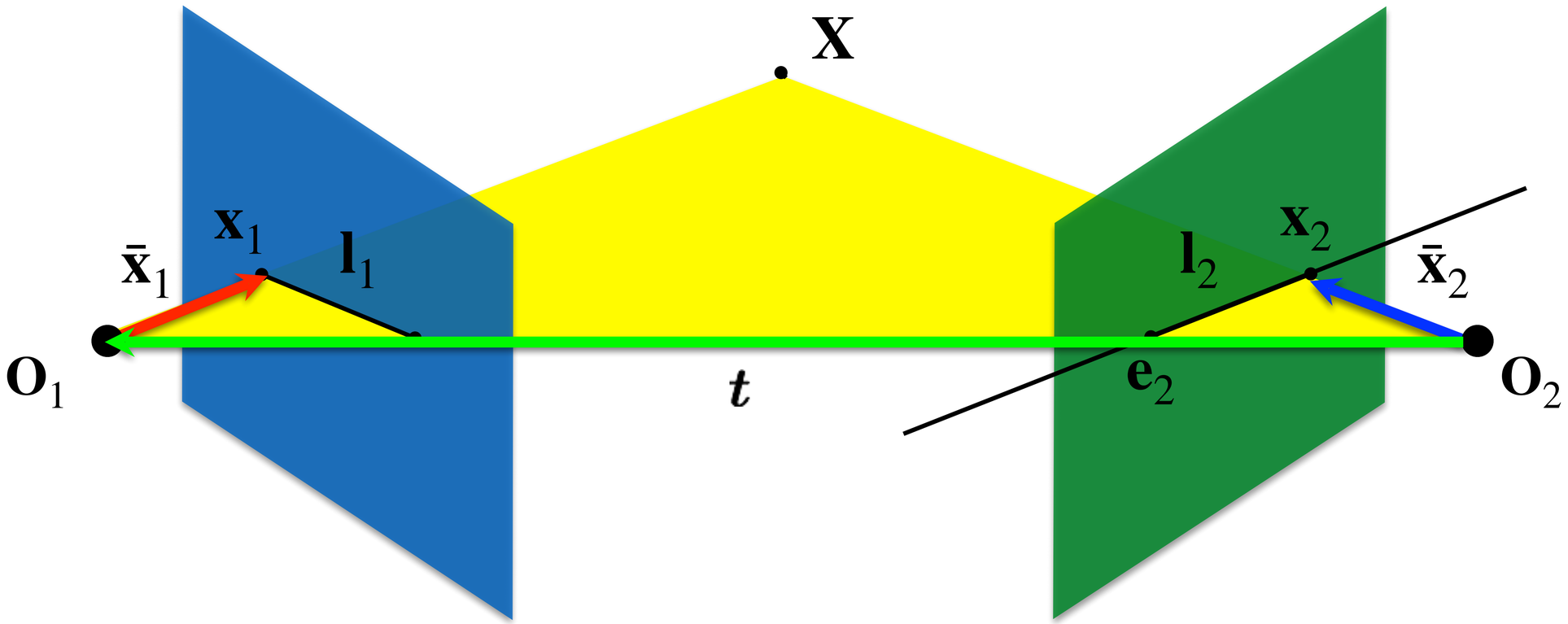
Adapted from: CMU 16-385 (Yannis, Kris)



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

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

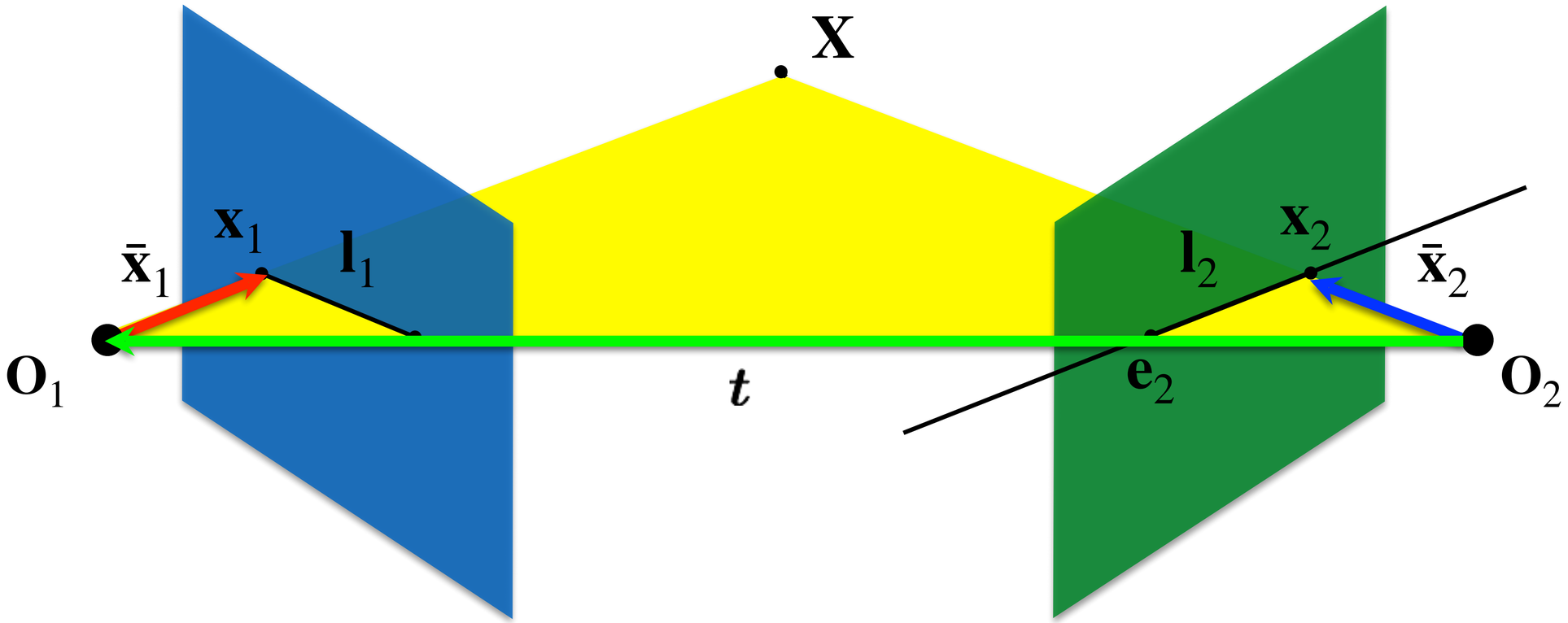
dot product of orthogonal vectors ↗ ↖ cross-product: vector orthogonal to plane



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

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

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

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

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

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

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

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

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

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

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

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

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

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

putting it together

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

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

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

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

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

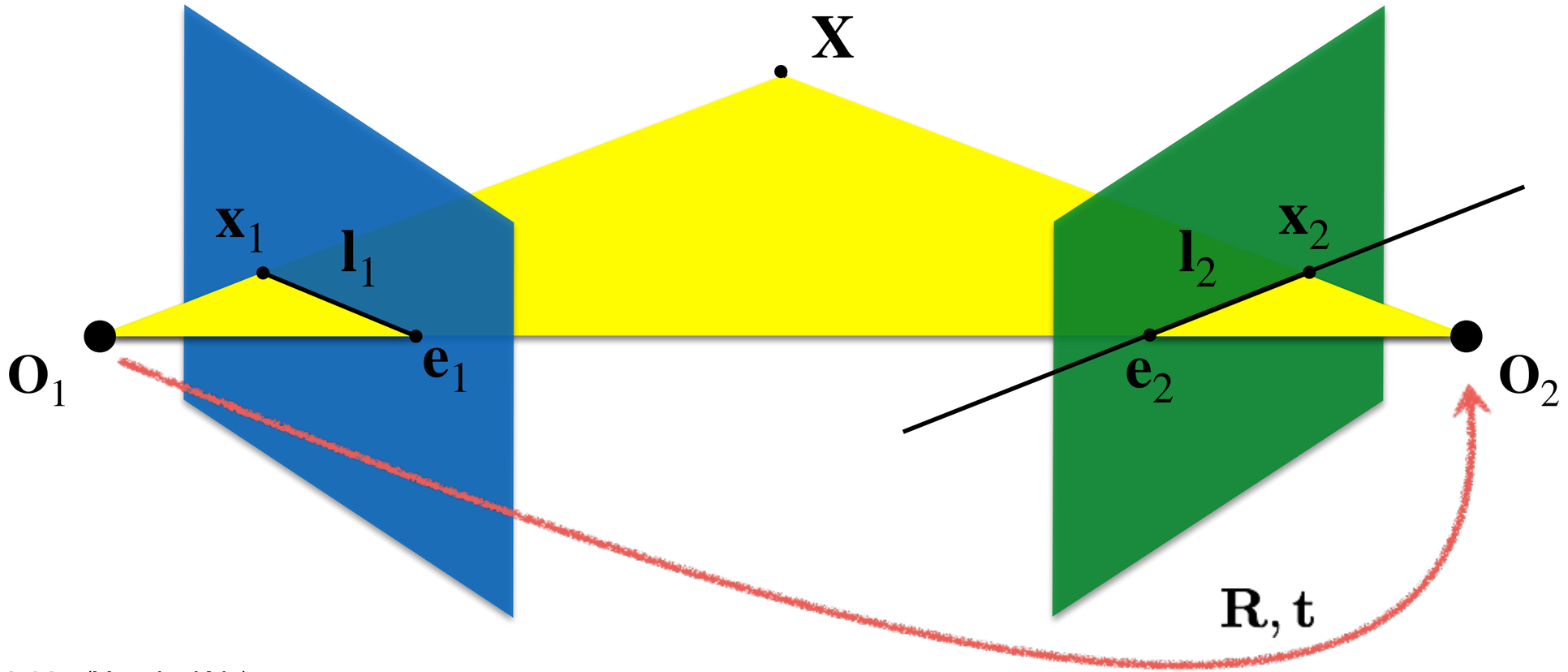
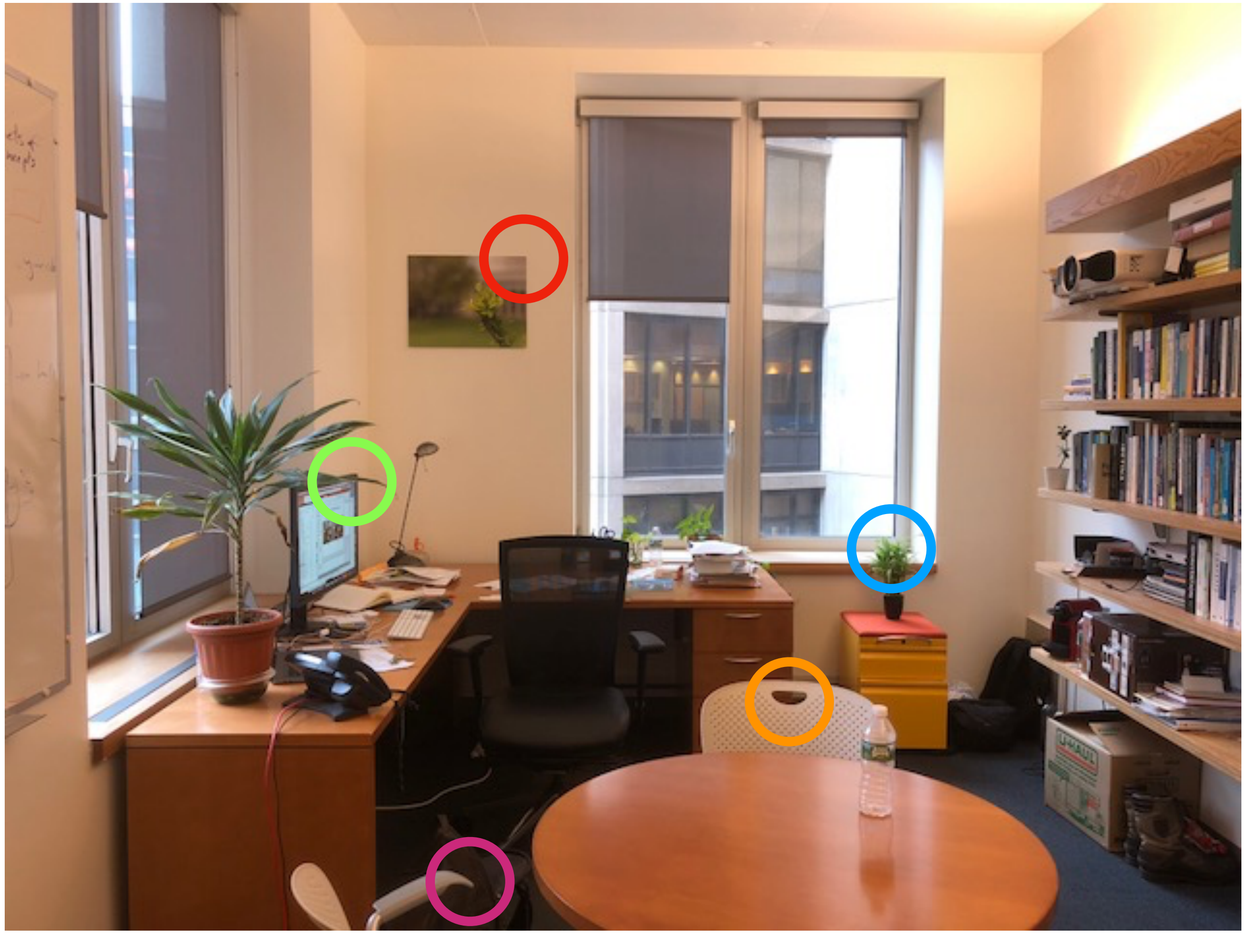
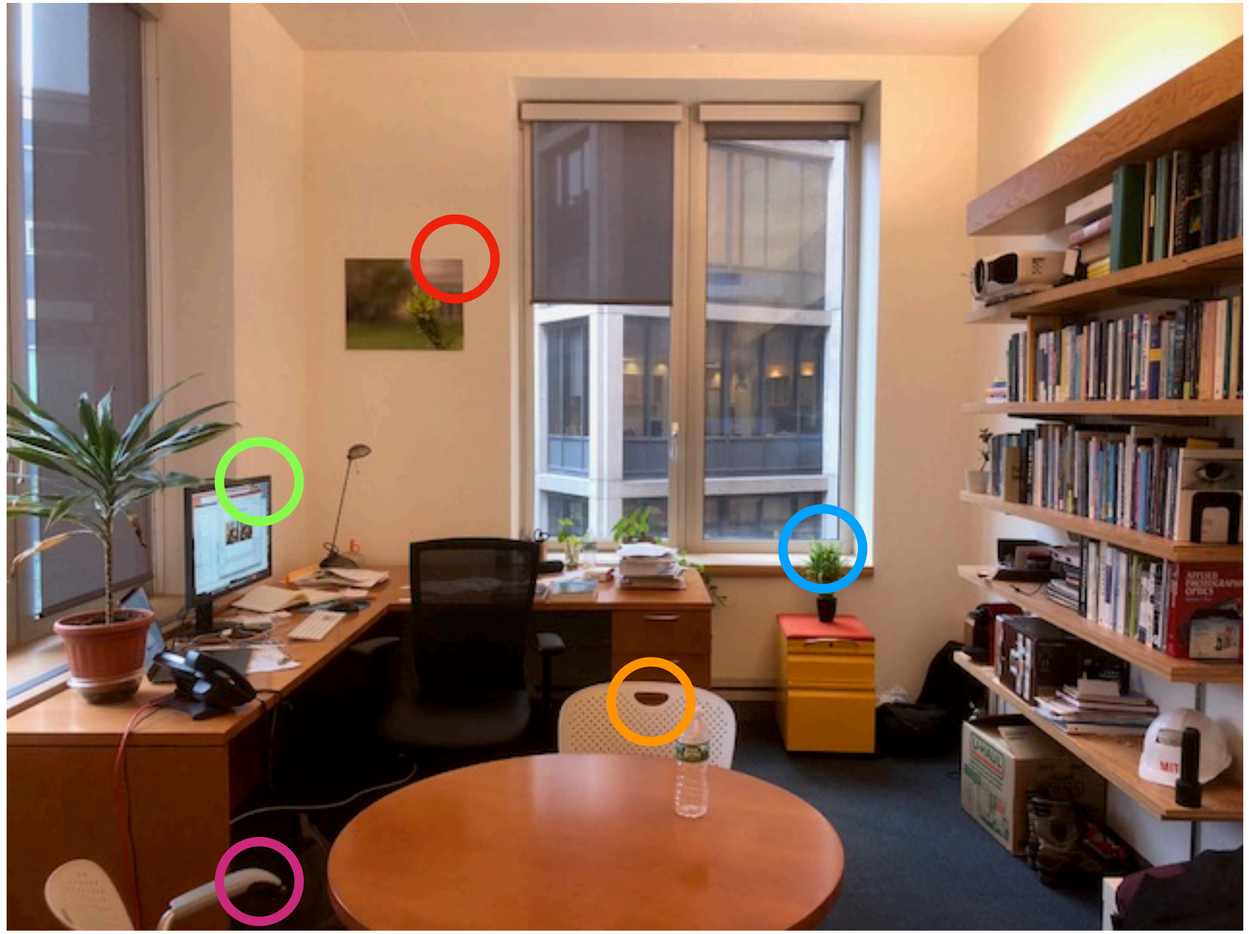
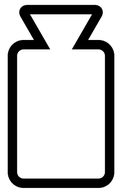


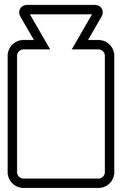
Figure credit: CMU 16-385 (Yannis, Kris)







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



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

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

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

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

$$\begin{bmatrix} x_1x_2, x_1y_2, x_1, yx_2, yy_2, y, x_2, y_2, 1 \end{bmatrix} \begin{bmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{bmatrix} = 0$$

The Eight-Point Algorithm

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

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

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

We can determine $\mathbf{f}$ *up to scale via a "homogeneous least squares"*

solve ($\arg\min_{\|\mathbf{f}\|=1} \|\mathbf{W}\mathbf{f}\|_2^2$ *)*

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

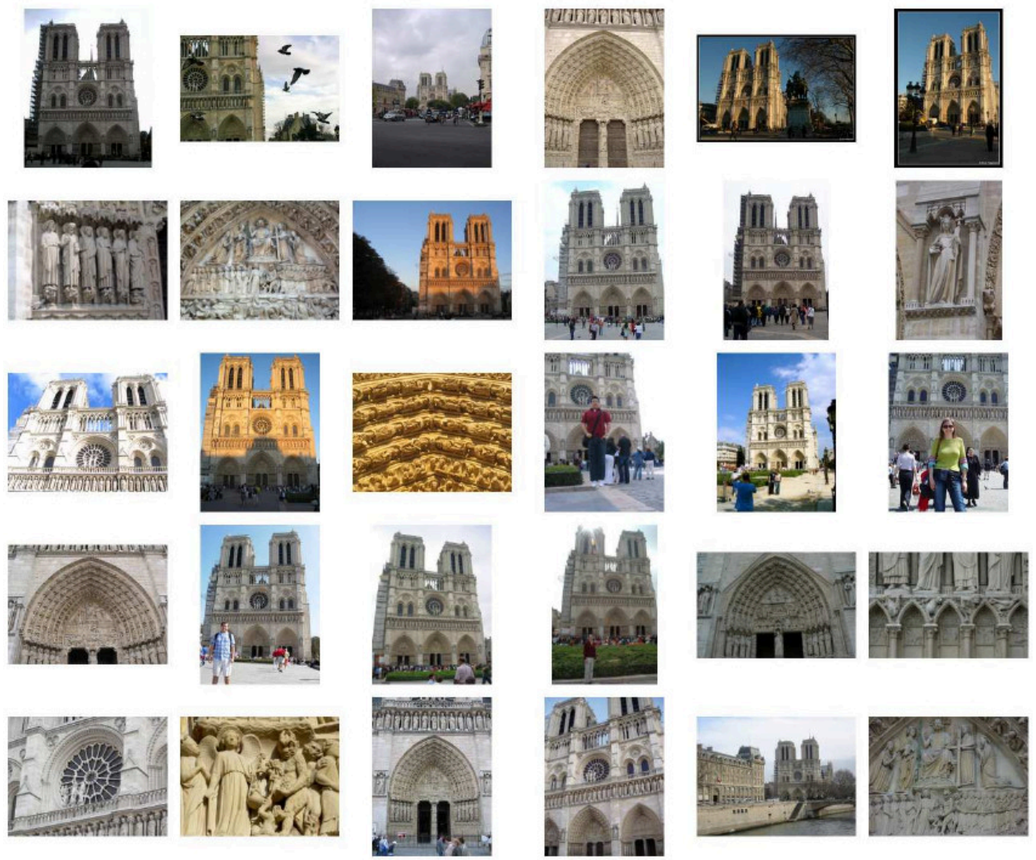
We can determine $\mathbf{f}$ *up to scale via a "homogeneous least squares"*
solve ($\arg\min_{\|\mathbf{f}\|=1} \|\mathbf{W}\mathbf{f}\|_2^2$ *)*

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

If $\mathbf{K}_i$ *are known, we can then back out* $\mathbf{R}$ *and* $\mathbf{t}$*.*
If not, can exploit that $\mathbf{F}$ *has rank 2 via SVD & throwing out smallest EV.*

What if we have **many** views?

Bundle Adjustment

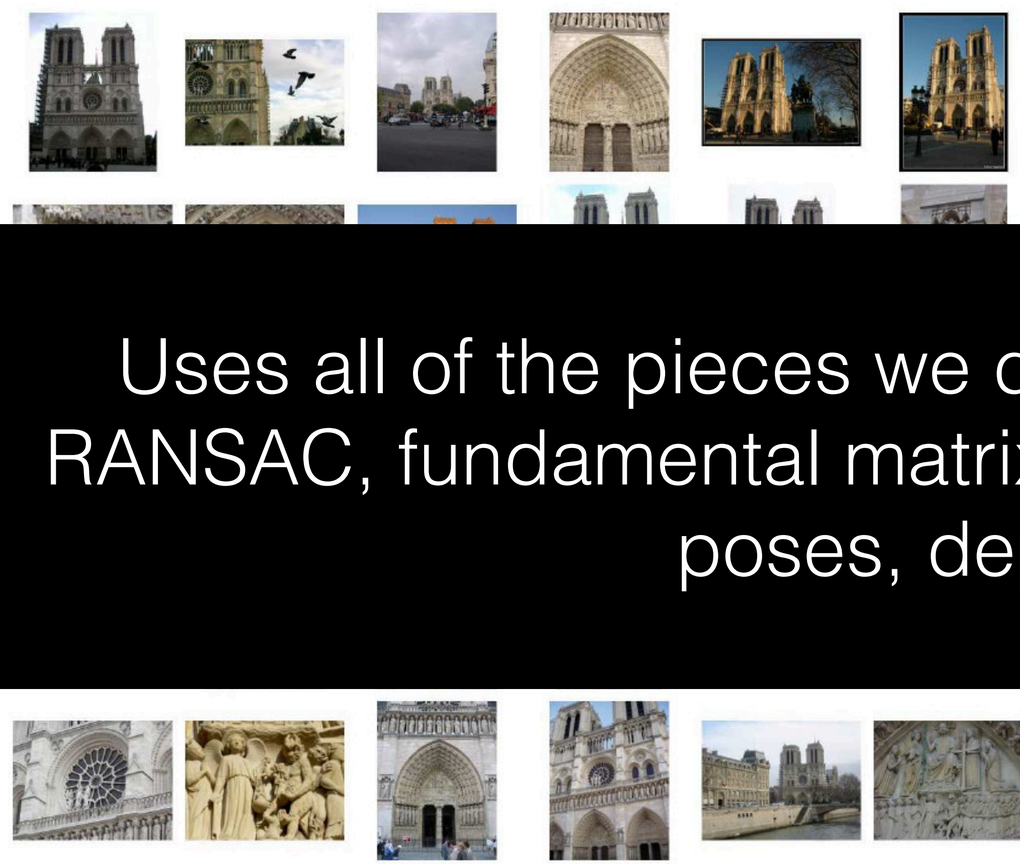


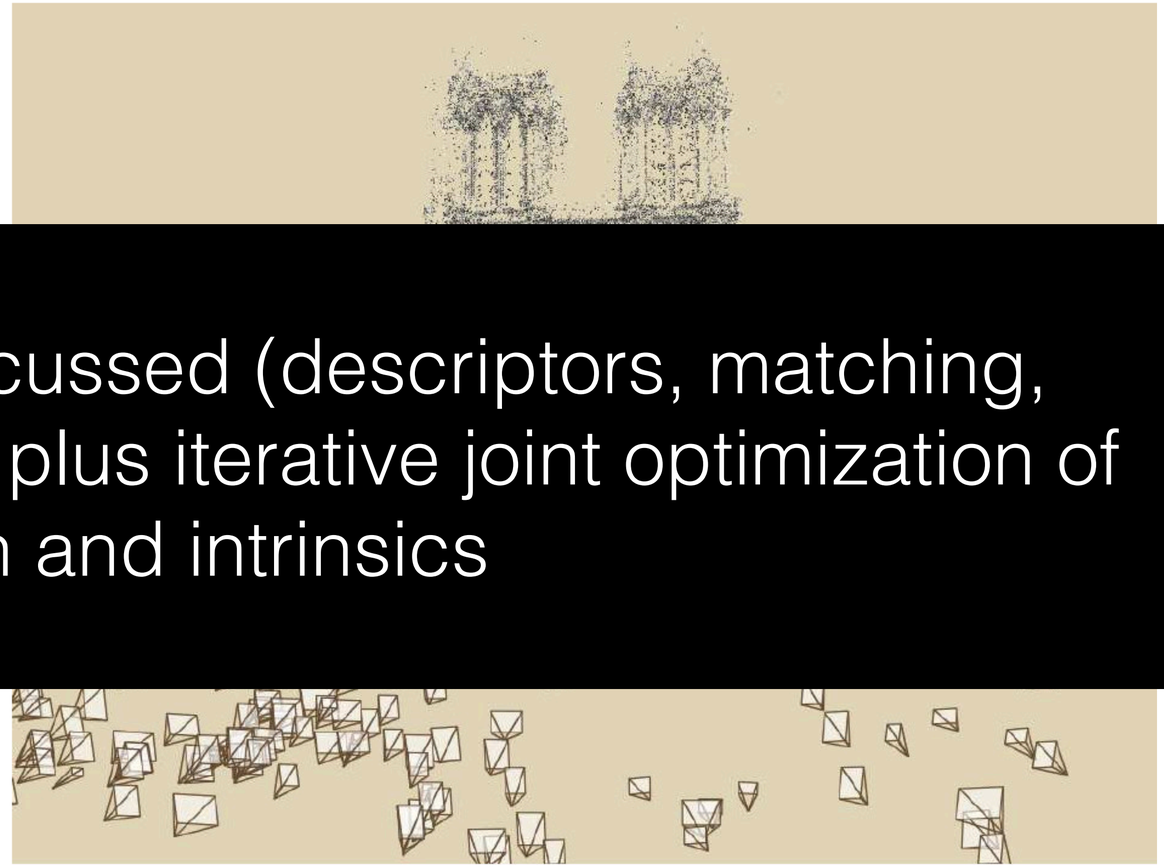
- **Goal: Optimize reprojection errors** (distance between observed feature and projected 3D point in image plane) **wrt. camera parameters and 3D point cloud**

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





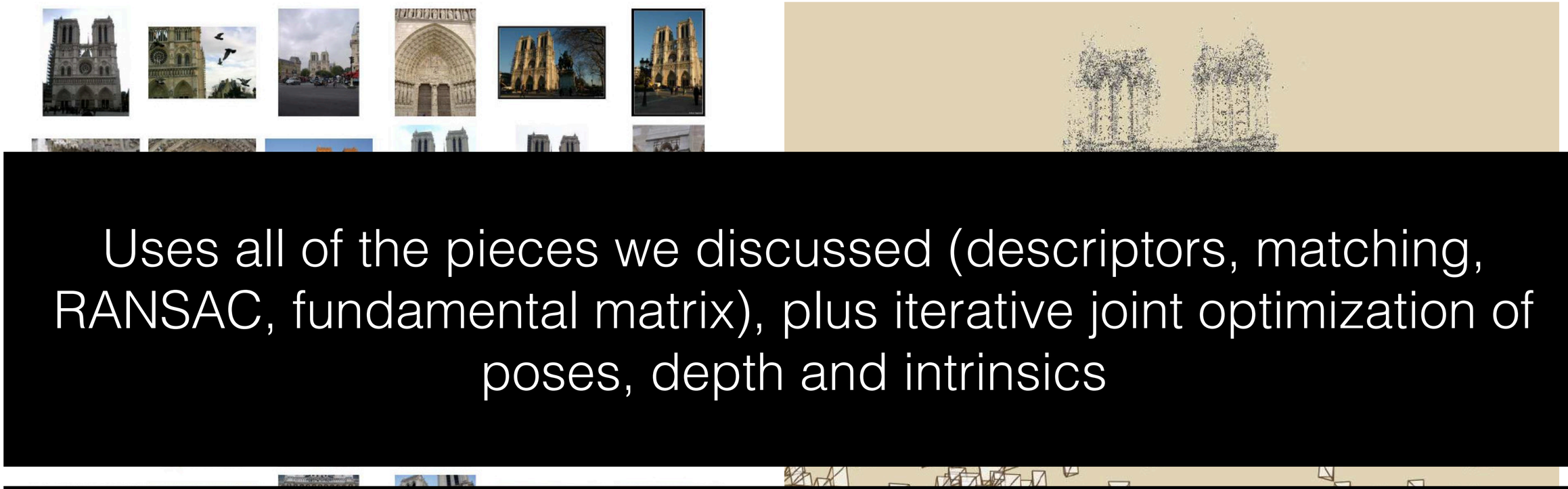
Uses all of the pieces we discussed (descriptors, matching, RANSAC, fundamental matrix), plus iterative joint optimization of poses, depth and intrinsics

- **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



Uses all of the pieces we discussed (descriptors, matching, RANSAC, fundamental matrix), plus iterative joint optimization of poses, depth and intrinsics

“COLMAP” is the de-facto standard implementation.

- **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

COLMAP SfM

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

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

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

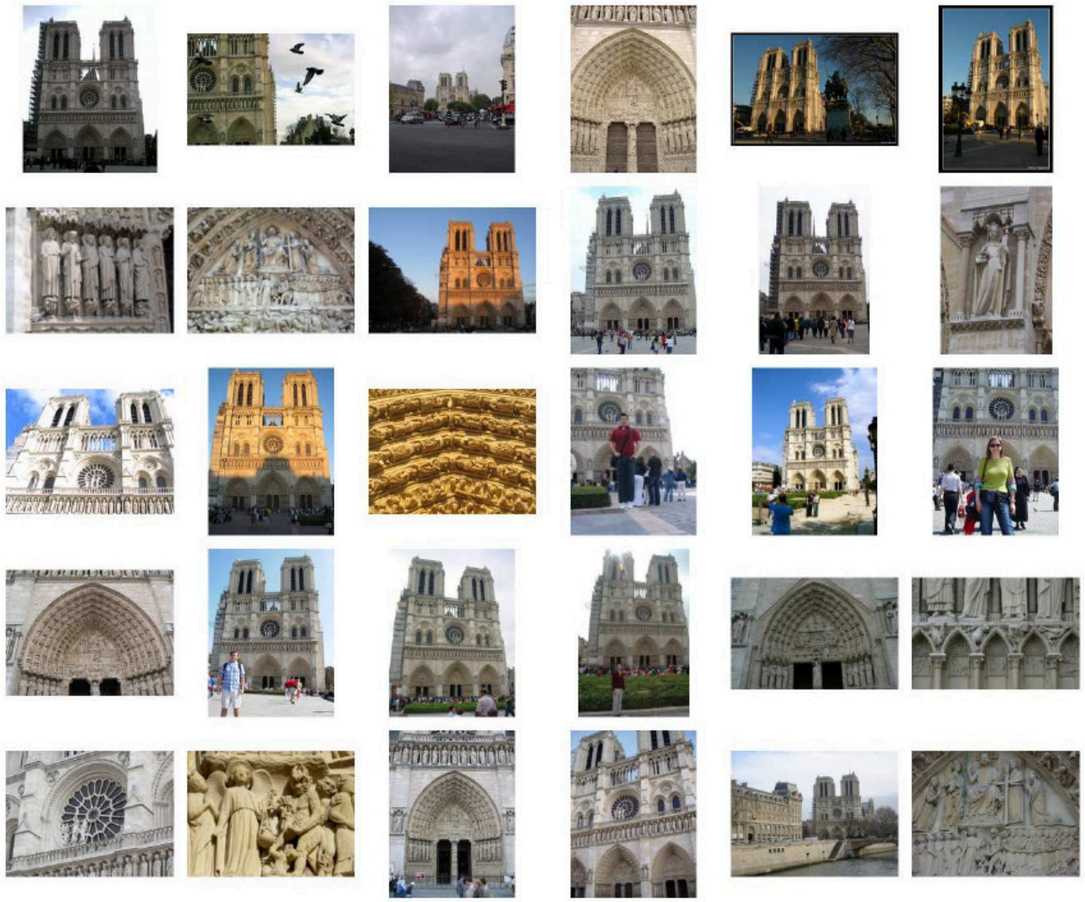
COLMAP MVS

- 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.

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

Photo Tourism

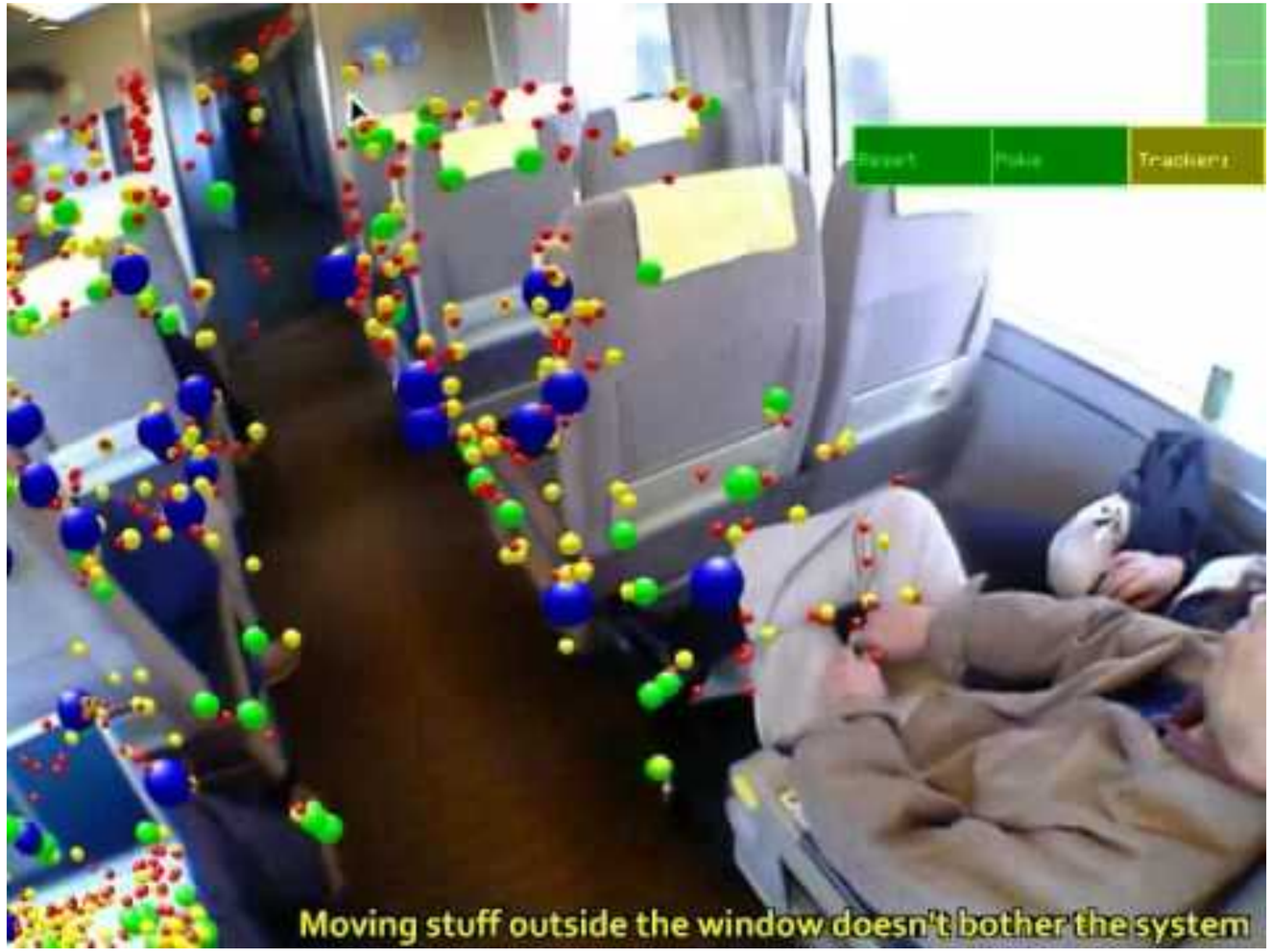


- **Photo Tourism / PhotoSynth** allows for exploring photo collections in 3D

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)

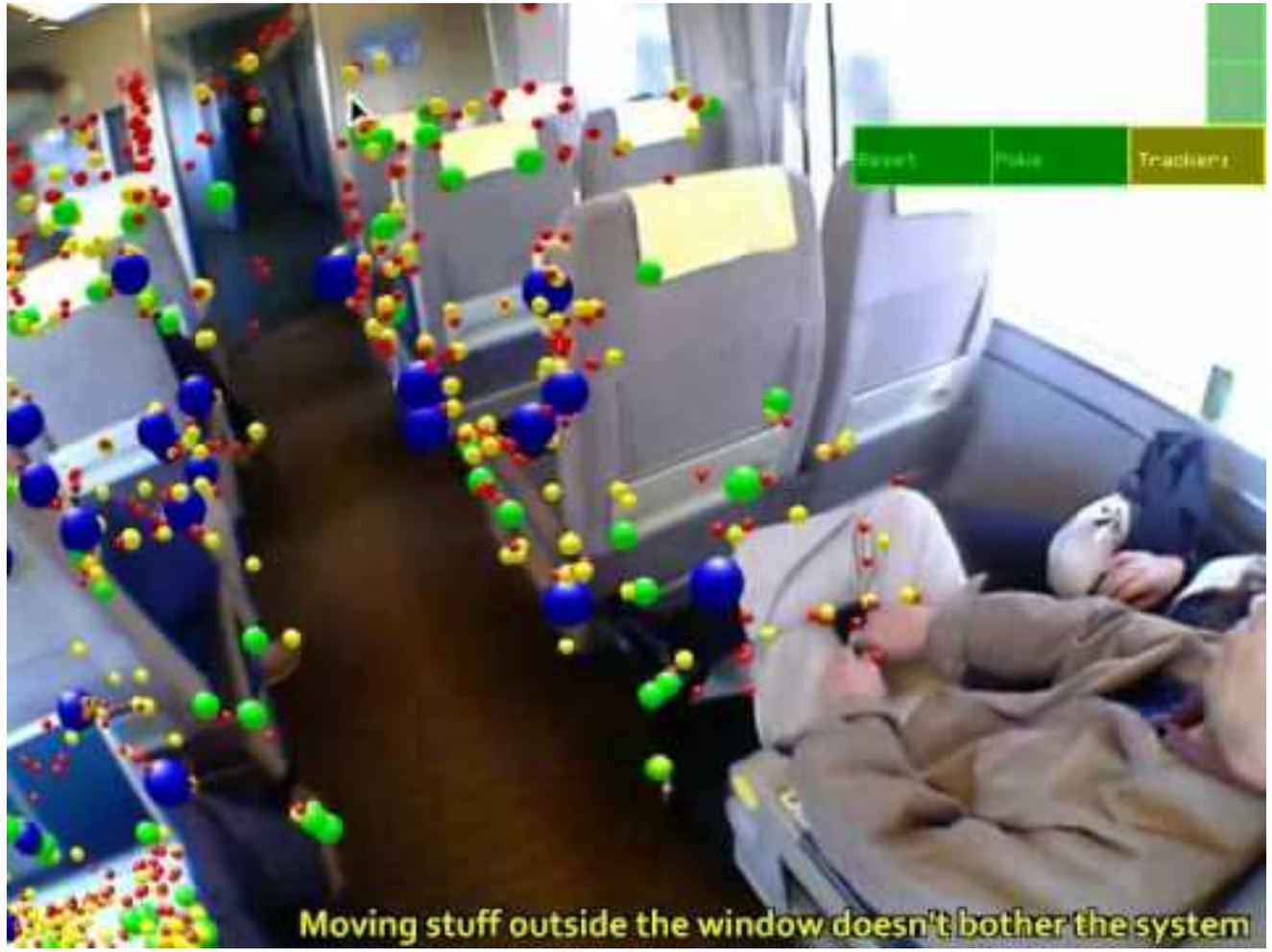


- **PTAM** demonstrates real-time tracking and mapping of small workspaces

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

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

Parallel Tracking and Mapping (PTAM)

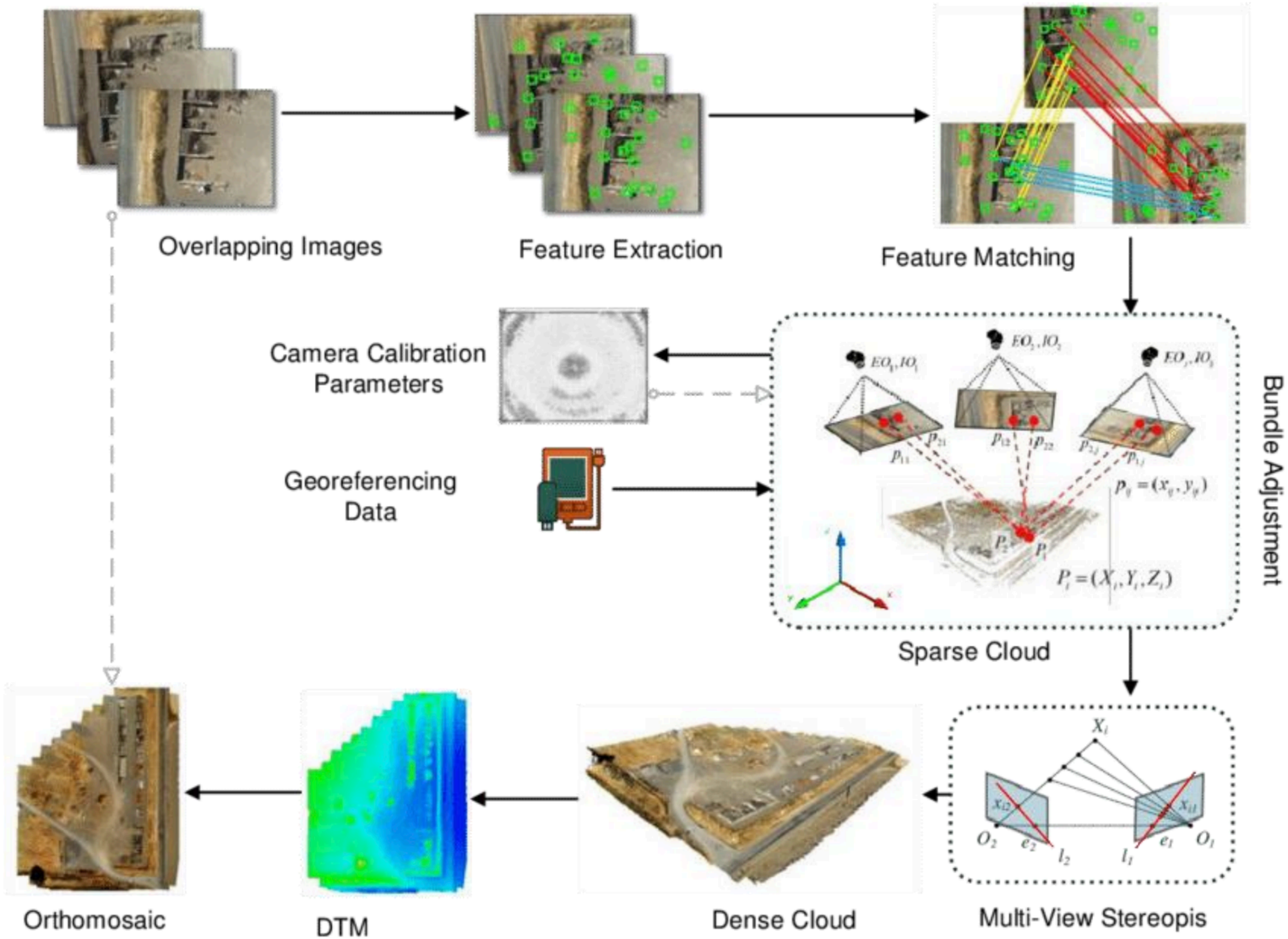


- **PTAM** demonstrates real-time tracking and mapping of small workspaces

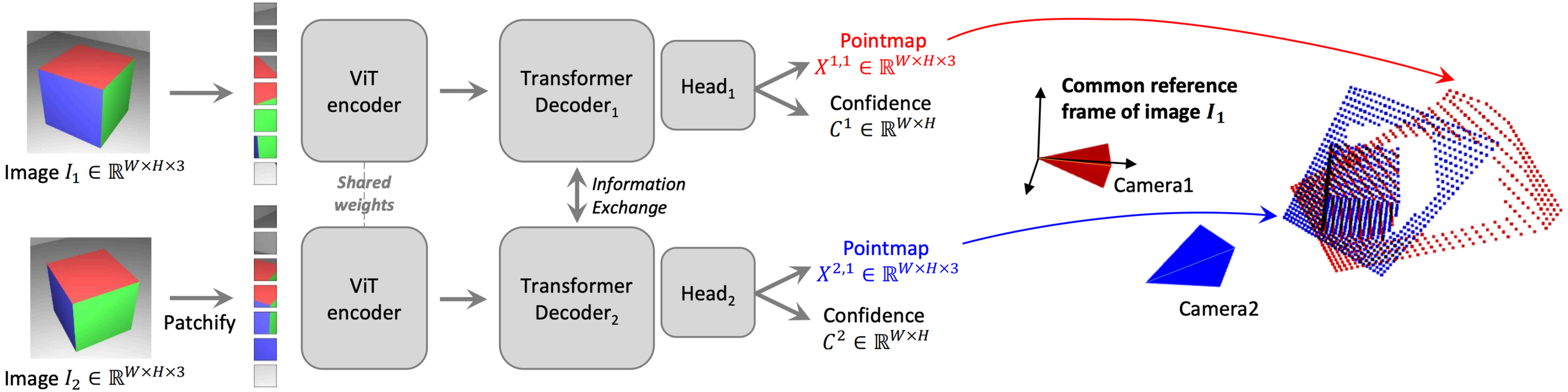
Klein and Murray: Parallel Tracking and Mapping for Small AR Workspaces. ISMAR, 2007.

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

SfM ~works, but is slow, not 100% reliable, and complicated

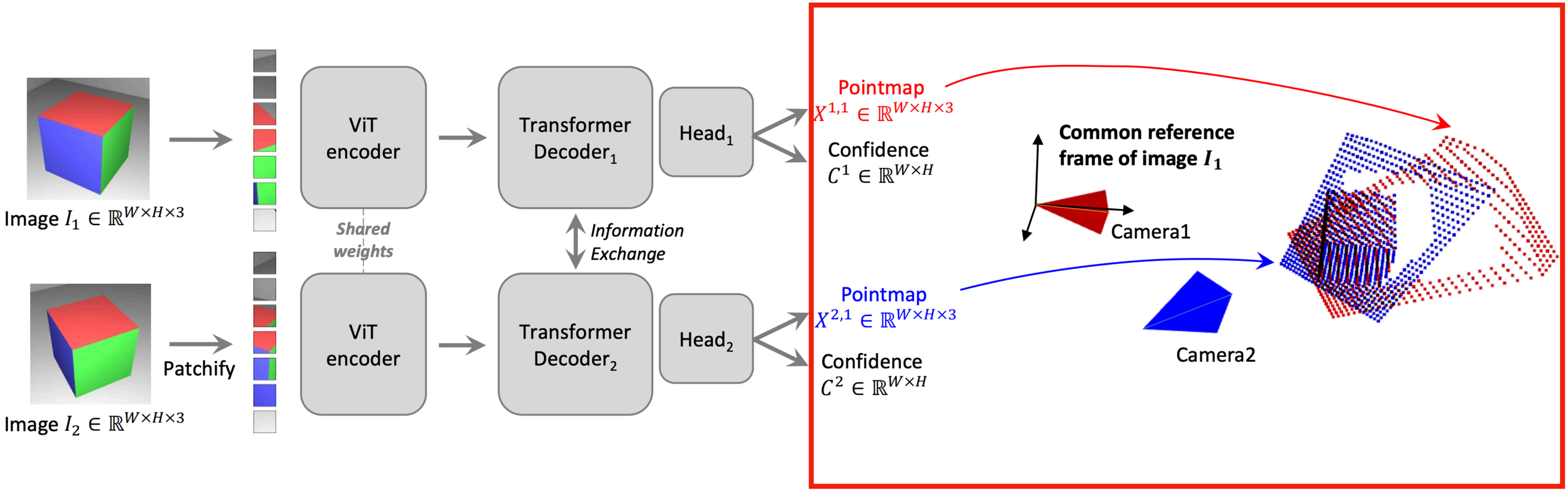


DUST3R: Pose SfM as supervised learning problem



Wang et al., DUSt3R: Geometric 3D Vision Made Easy, CVPR 2024

DUST3R: Pose SfM as supervised learning problem



What made it work:

- Curating large supervised dataset from SfM reconstructions
- PointMap parameterization

Wang et al., DUSt3R: Geometric 3D Vision Made Easy, CVPR 2024

DUST3R: Pose SfM as supervised learning problem

Input

RGB images

No calibration, No poses

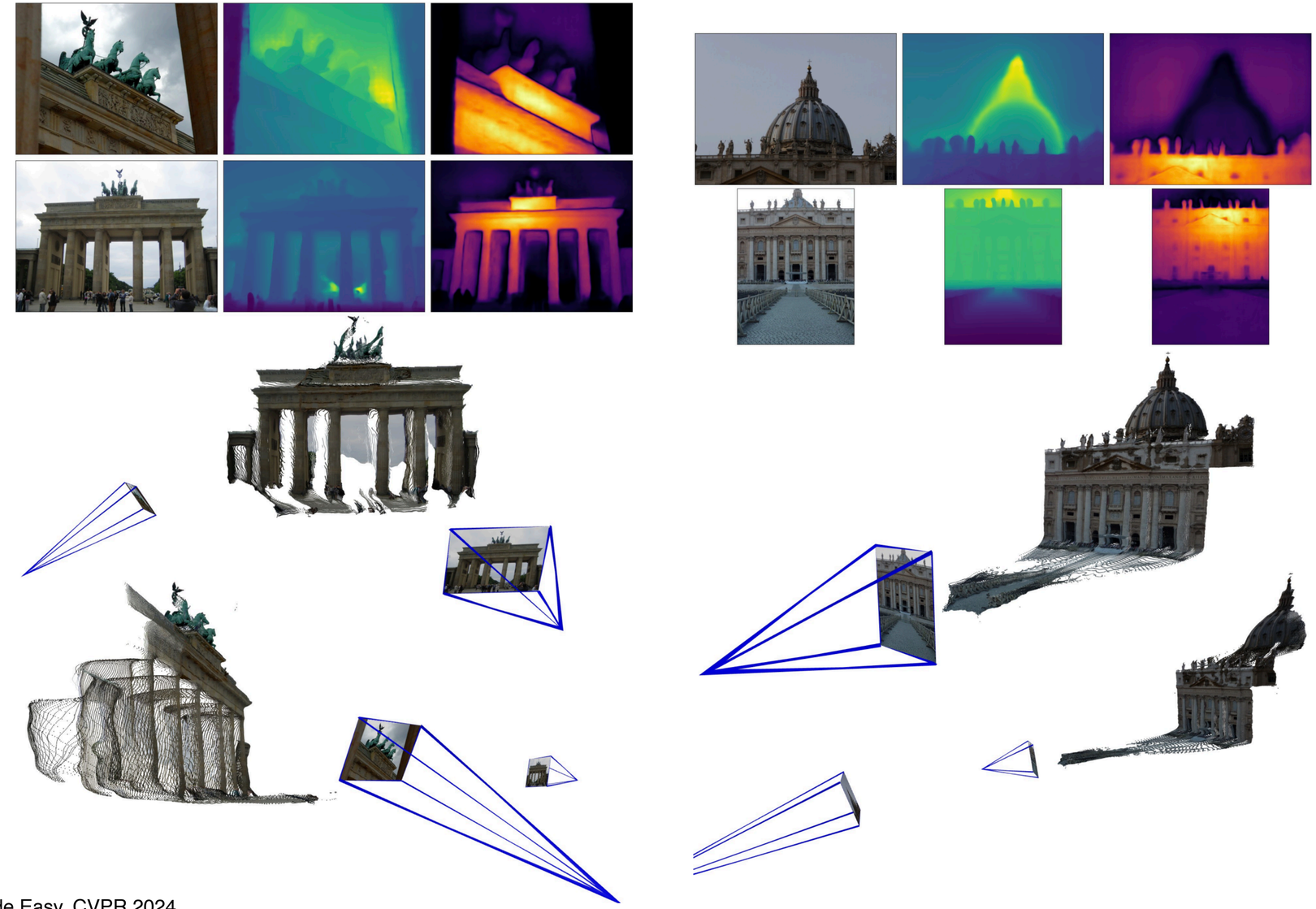
DUST3R: Pose SfM as supervised learning problem

Input

RGB images

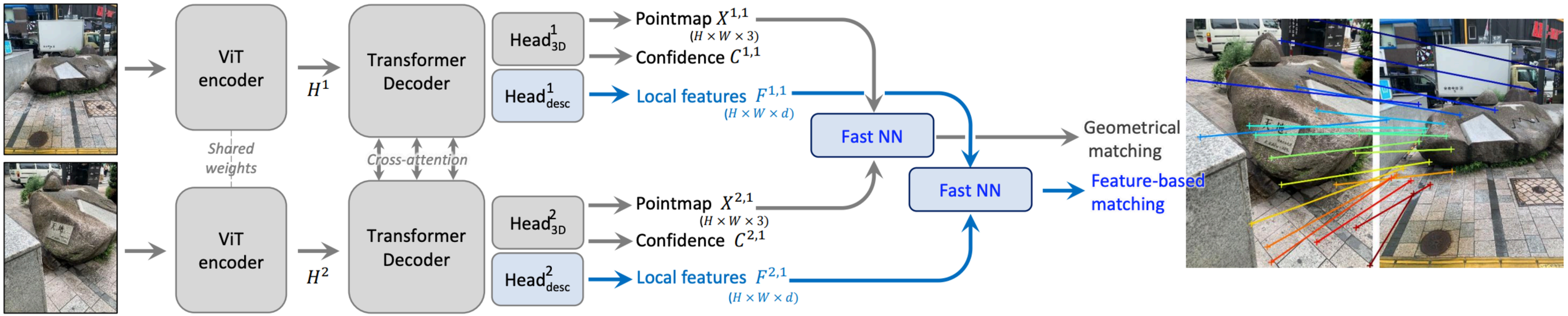
No calibration, No poses

DUST3R: Pose SfM as supervised learning problem



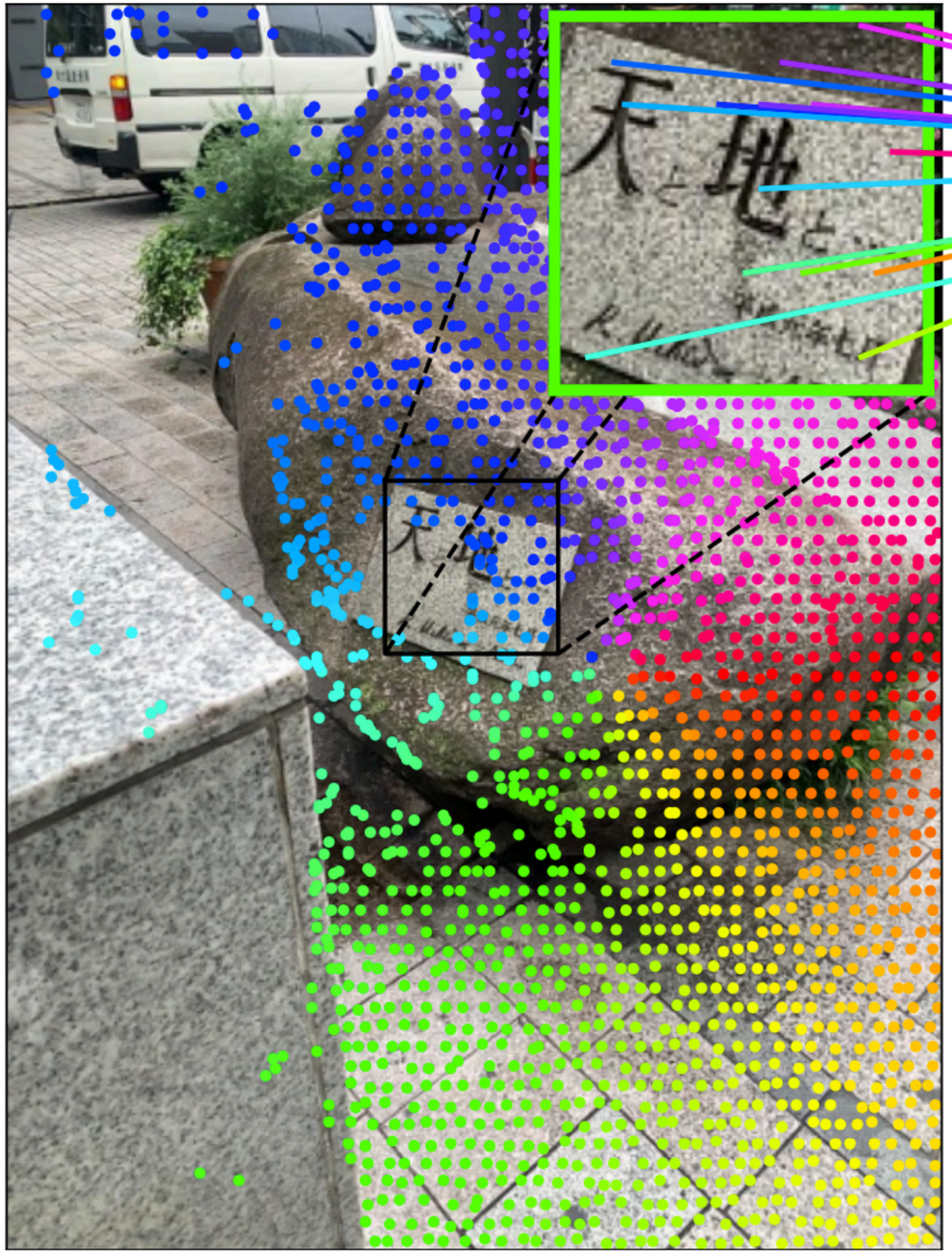
Wang et al., DUSt3R: Geometric 3D Vision Made Easy, CVPR 2024

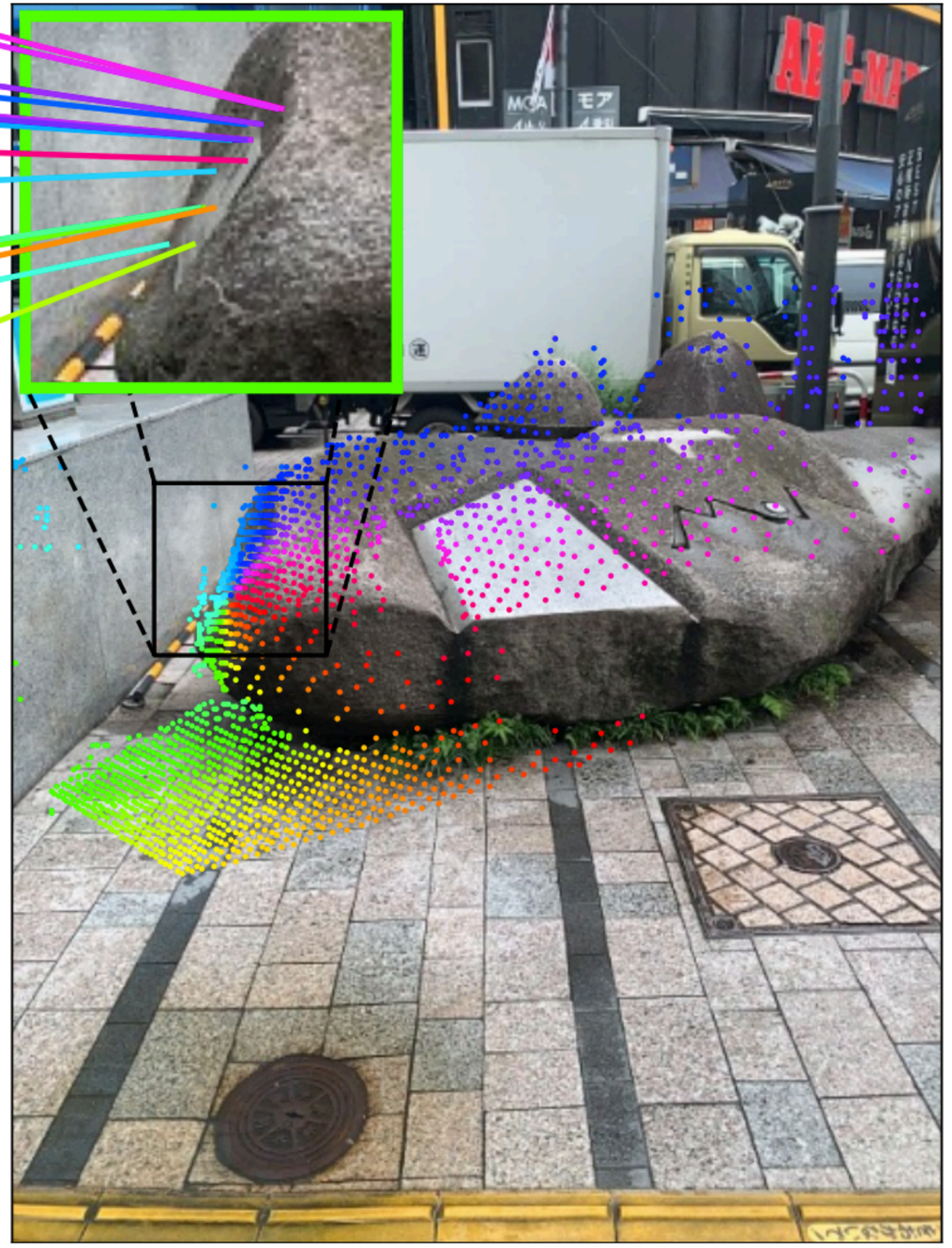
Grounding Image Matching in 3D with MASt3R



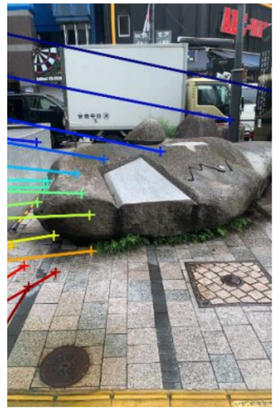
Leroy et al., Grounding Image Matching in 3D with MASt3R, CVPR 2024

G





St3R

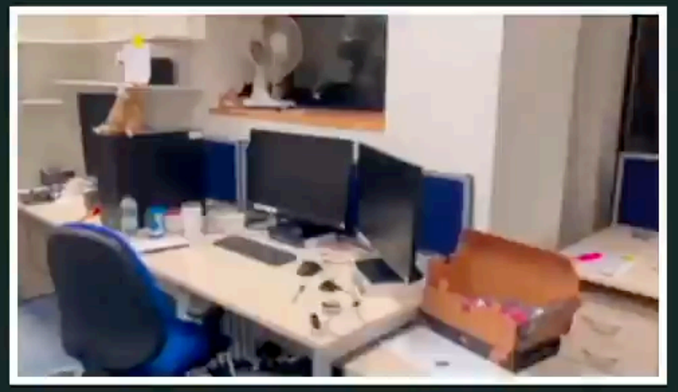


Leroy et al., Grounding Image Matching in 3D with MASt3R, CVPR 2024

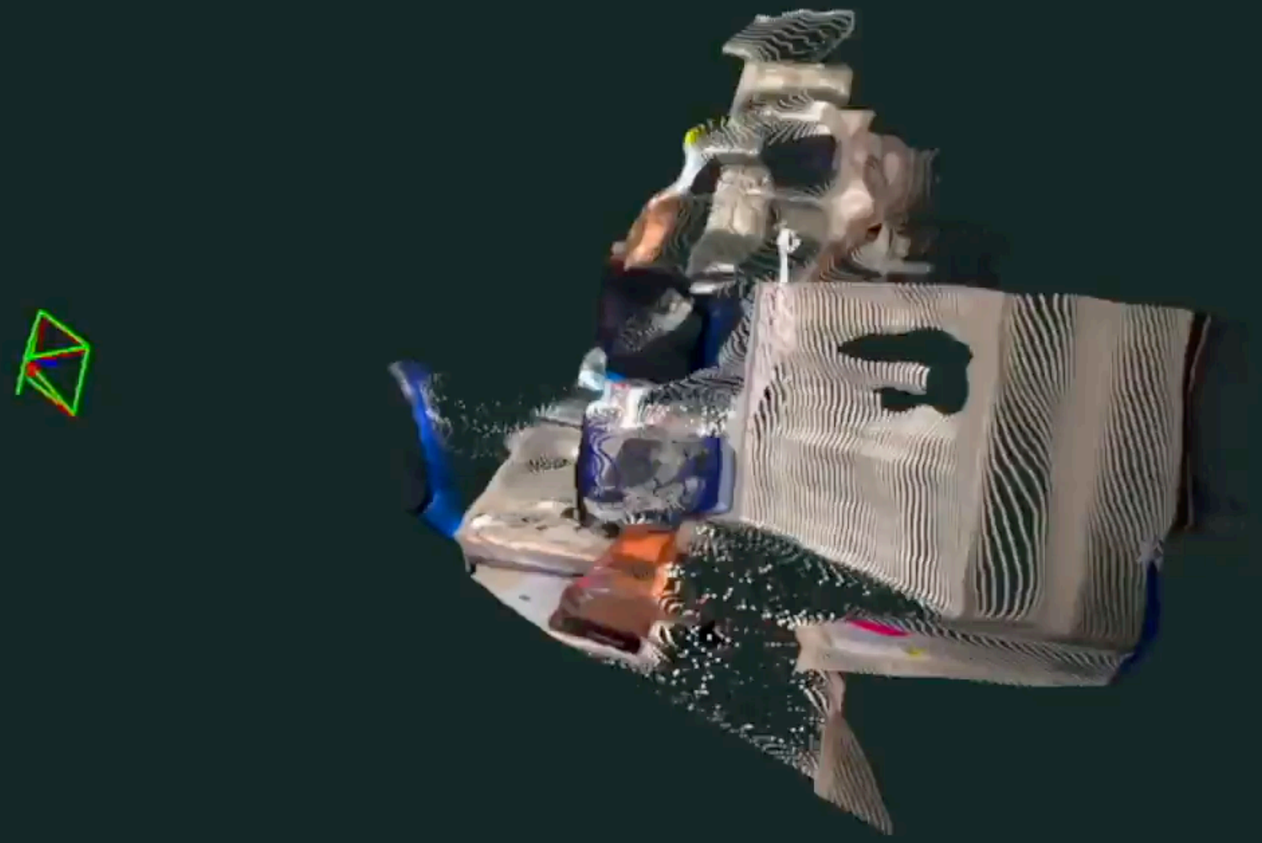
Leroy et al., Grounding Image Matching in 3D with MASt3R, CVPR 2024

Uncalibrated RGB
8x Playback

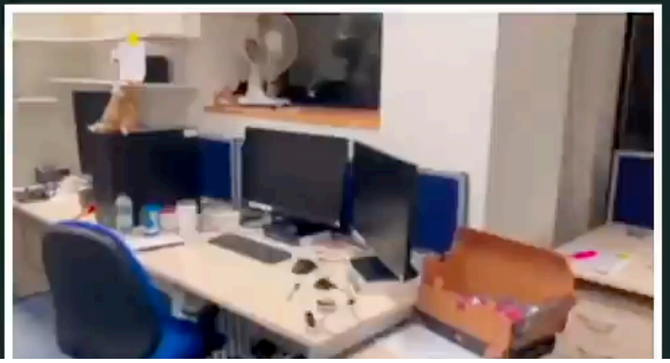
Uncalibrated RGB
8x Playback



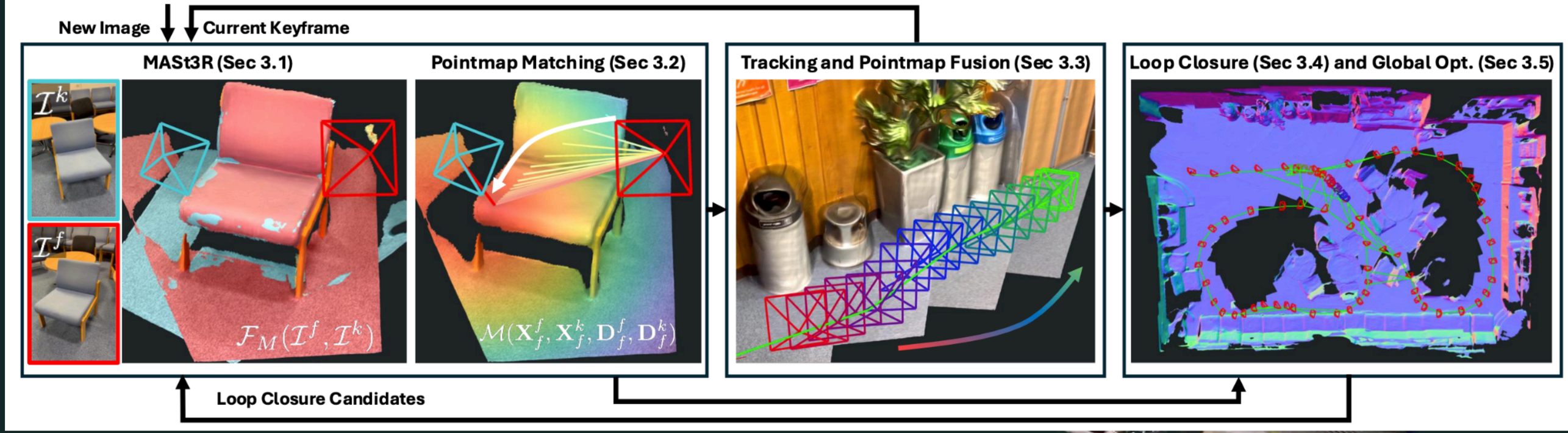
Uncalibrated RGB
8x Playback



Murai, Dexheimer et al., MASt3R-SLAM: Real-Time Dense SLAM with 3D Reconstruction Priors, CVPR 2025



Uncalibrated RGB
8x Playback



Murai, Dexheimer et al., MASt3R-SLAM: Real-Time Dense SLAM with 3D Reconstruction Priors, CVPR 2025

Conclusion on correspondence and 3D vision

For any given *task* (correspondence estimation, 3D reconstruction, depth prediction…): collect a supervised training set and deploy a generalist-as-possible architecture.

To collect the dataset, you either simulate or you rely on hand-crafted prior work.

Network architecture and training details matter, but you should get a sign of life even if you get these details slightly wrong.

You will notice once you try to apply these to real-world problems (robotics, self-driving, …) that there is always more complexity and generality hidden away!