ADVANCES IN COMPUTER VISION

MODULE 1: Geometry

MULTI-VIEW GEOMETRY

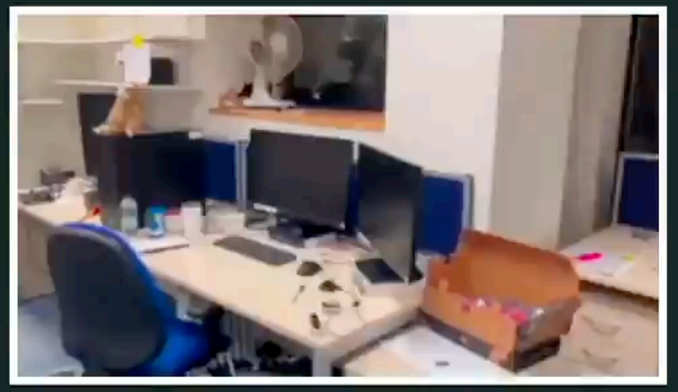


Prof. Vincent Sitzmann

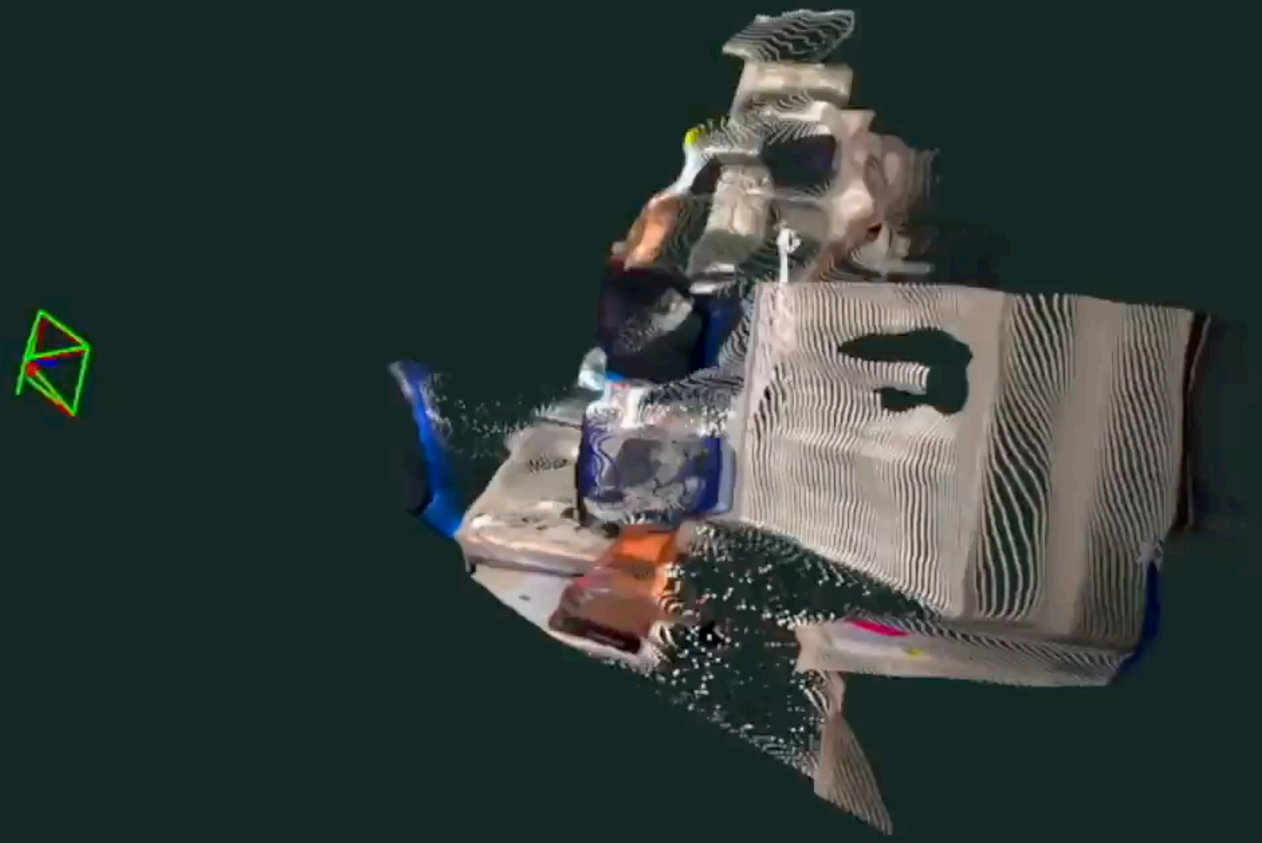
Addendum to flow/matching

- Progression of approaches/definition of correspondence
- 1/ per pixel RGB value, color constancy, 1st order Taylor and image gradient
- 2/ neighborhood match (sum of square difference over a window
- 3/ Sophisticated hand-designed features (SIFT, HOG)
- 4/ learned features, often part of end-to-end

Say more clearly we also get 3D points at the end



Uncalibrated RGB
8x Playback



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

3D and novel view synthesis

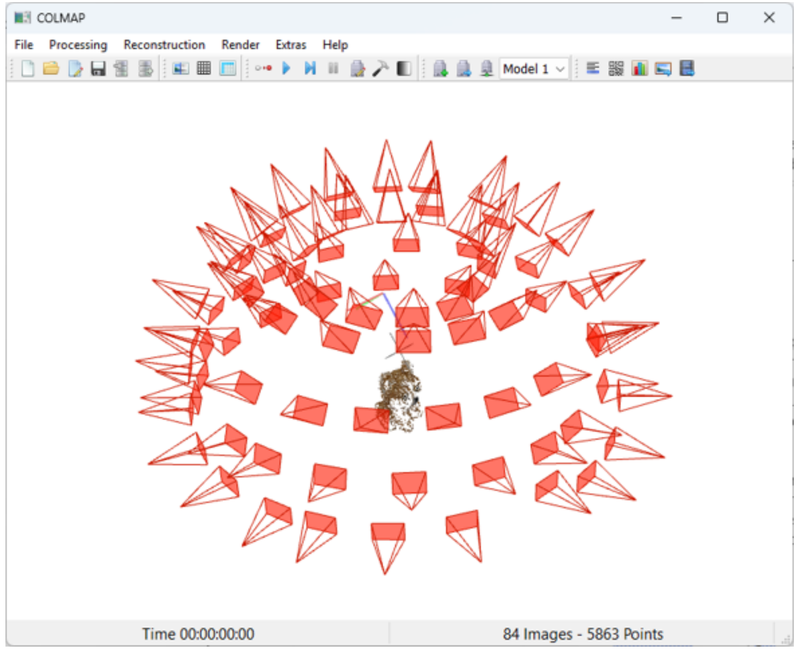
- Camera calibration still largely done with traditional techniques.
  - Typically the software clomp
- But changing rapidly
  - Various degrees of inductive bias

Traditional landscape

- SfM: sparse
- Multiview stereo : dense
  - With SfM providing calibration
  -

Traditional 3D Structure from Motion (e.g. Colmap)

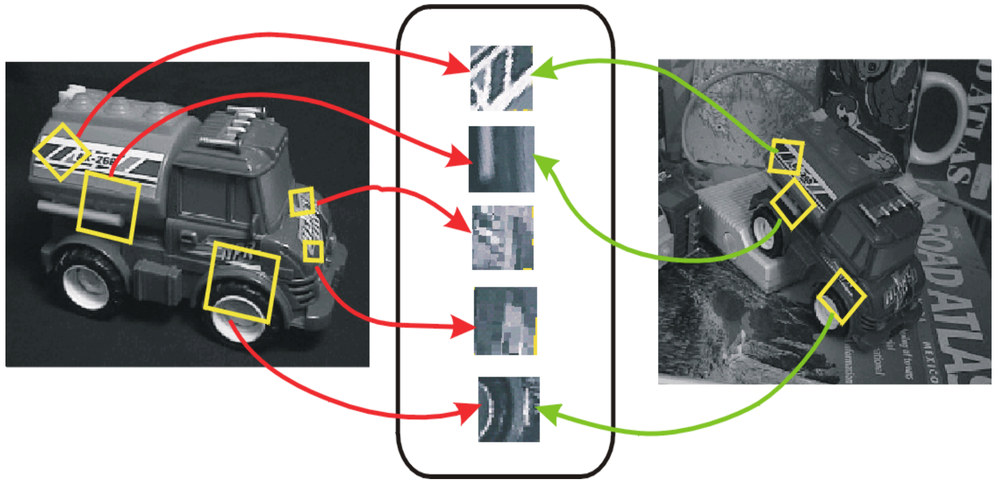
- Feature extraction and matching (SIFT)
- Linear algebra camera calibration given correspondences
  - 8, 7 or 5 point correspondences
  - + RANSAC for robustness
  - Also gives 3D points
- Non-linear bundle adjustment
- Potentially additionally dense 3D reconstruction (e.g. using patch match)

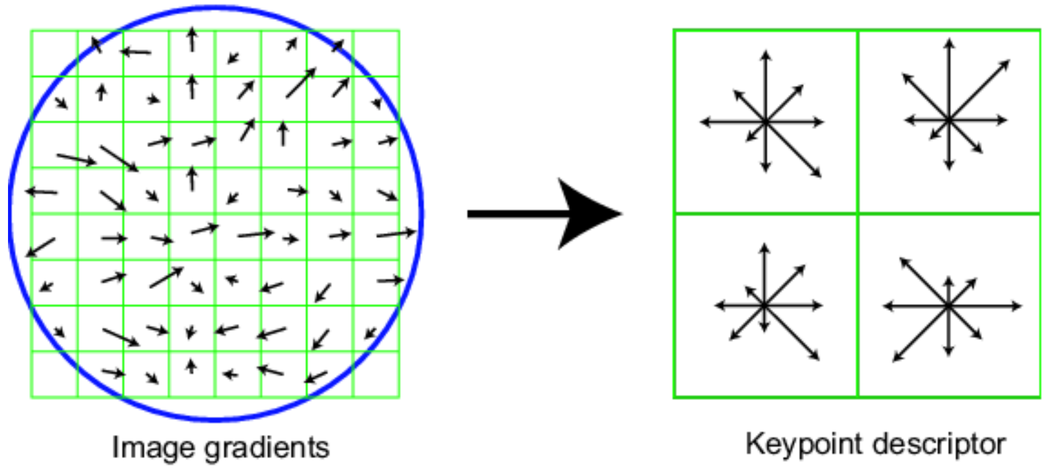


https://smartdatascan.com/tutorials/gaussian-splatting-windows/colmap/adjusted-parameters

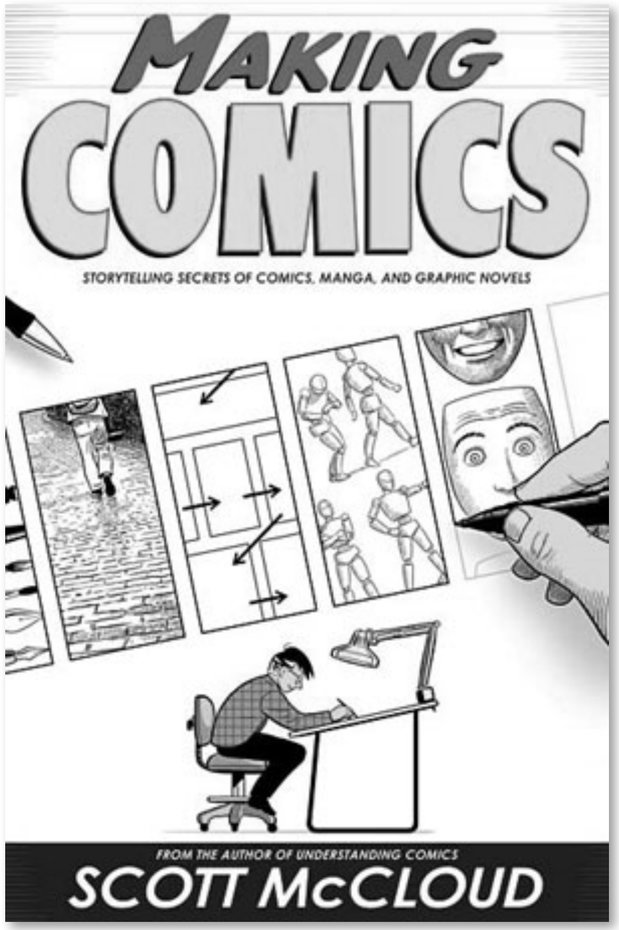
A few words about traditional correspondences

- Sparse + distant in last lecture's taxonomy
- Special points + descriptors (SIFT and teh such)
  - Similar to good points to track, but multi scale
  - + rich hand-coded descriptor (some kind of histogram of gradient orientations)
- (Approximate) Nearest Neighbor
- Other tools =to deal with robustness (some of it in this lecture)

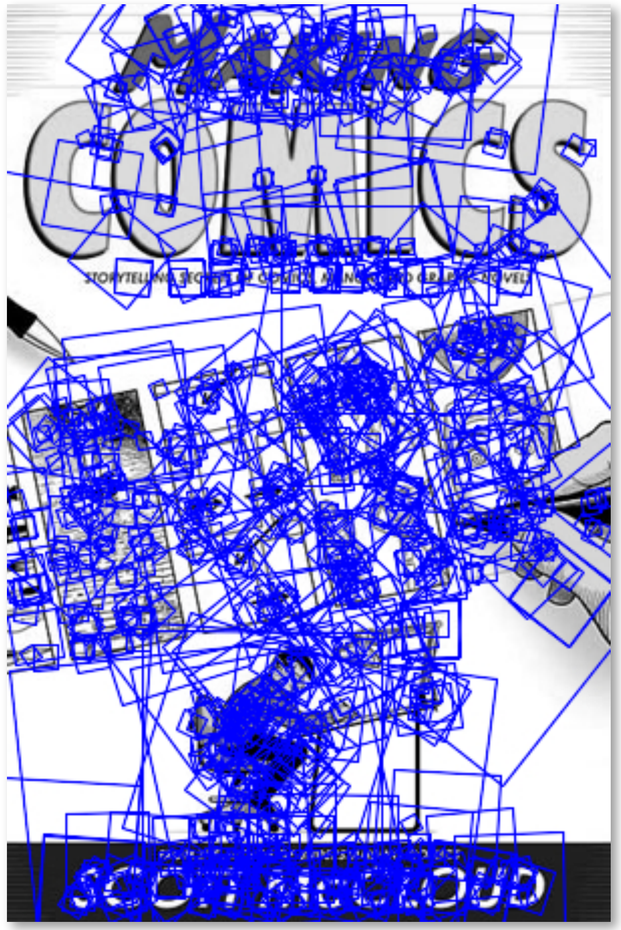




SIFT example



sift



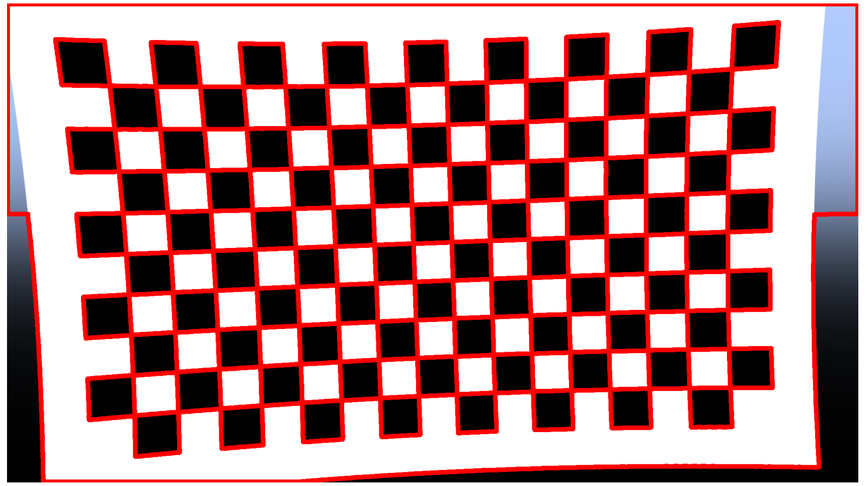
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

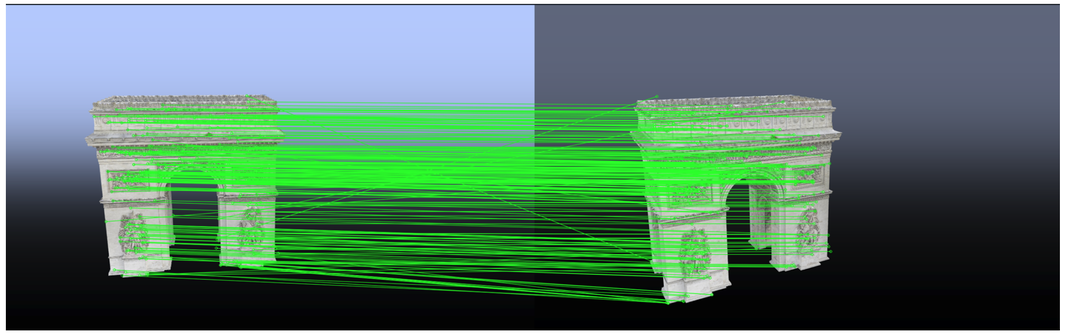
Why not use flow/tracking

- OK if you do SLAM and the camera path is smooth
  - Then the problem is also a tad harder because the baseline is small and some of the triangulation arguments don’t work as well
- But in many cases the camera viewpoints are more distant and we need very non-local matches
  - But then triangulations are better conditioned

Common calibration journey

- **Calibrate intrinsics with known target**
  (e.g. checkerboard)
  - Linear model
  - Plus non-linear effects (radial distortion)
    Not always possible (e.g. “in the wild” data)
- **Get extrinsics with correspondences**
  - SIFT
  - Linear algebra to deduce extrinsic matrix
    (make things more linear by warping images to correct for radial distortion)
  - RANSAC algorithm to make it robust
- **Refine with non-linear optimization**
  - And use more complete camera model (radial distortion)





A few word about intrinsic

- Calibrate radial distortion
  - Detect corners of checkerboard
  - Simple parametric model
  - Minimize parameters so that checkerboard corners are on straight lines
  - Then you can use a warp to straighten everything and work with a linear projection
  - Easy but tedious, use OpenCV

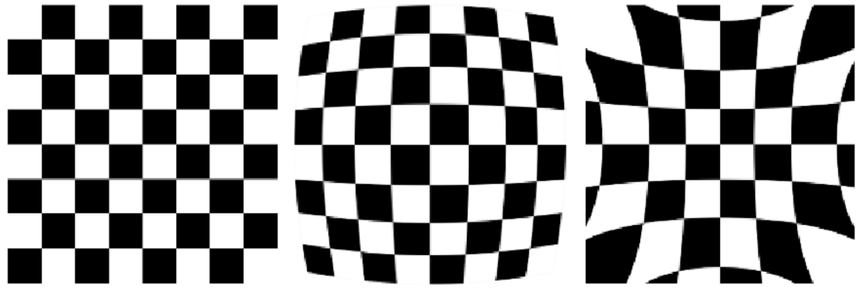
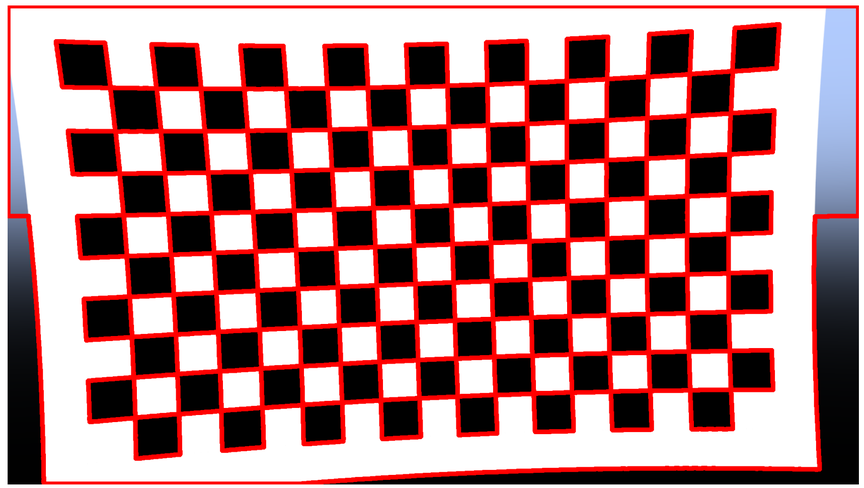


Figure 3. Left: No distortion, Middle: Barrel distortion, Right: Pincushion distortion

$$x_{\text{distorted}} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 xy + p_2(r^2 + 2x^2)$$

$$y_{\text{distorted}} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_2 xy + p_1(r^2 + 2y^2)$$



A few word about intrinsic

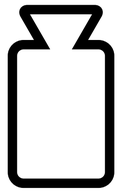
- Multiple (say 2) images of a known object (checkerboard) where the intrinsics are the same for all images
- Assume you know 3D geometry of checkerboard PLUS
  Z=0 for all 3D checkerboard points (without loss of generality),
  which removes one column of the extrinsic matrix
- Each known 2D point and known 3D point gives an equation.
  Unfortunately it includes matrix product of intrinsics and extrinsics.
  Goal : eliminate extrinsic terms
- Leverage the fact that the columns in a rotation matrix should be orthogonal
- Leverage form of K
- Orthogonality of columns of R leads to a dot product equation that involve K-1 on both sides, hence $B = K^{-T}K^{-1}$
- Cholesky decomposition $chol(B) = AA^T$

Plan

- Better understand multi view geometry
  - Establish that the forward model 3D->2D can be linear
  - Understand how to get
- Solve camera calibration as inverse problem
  - Linear algebra !

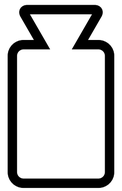




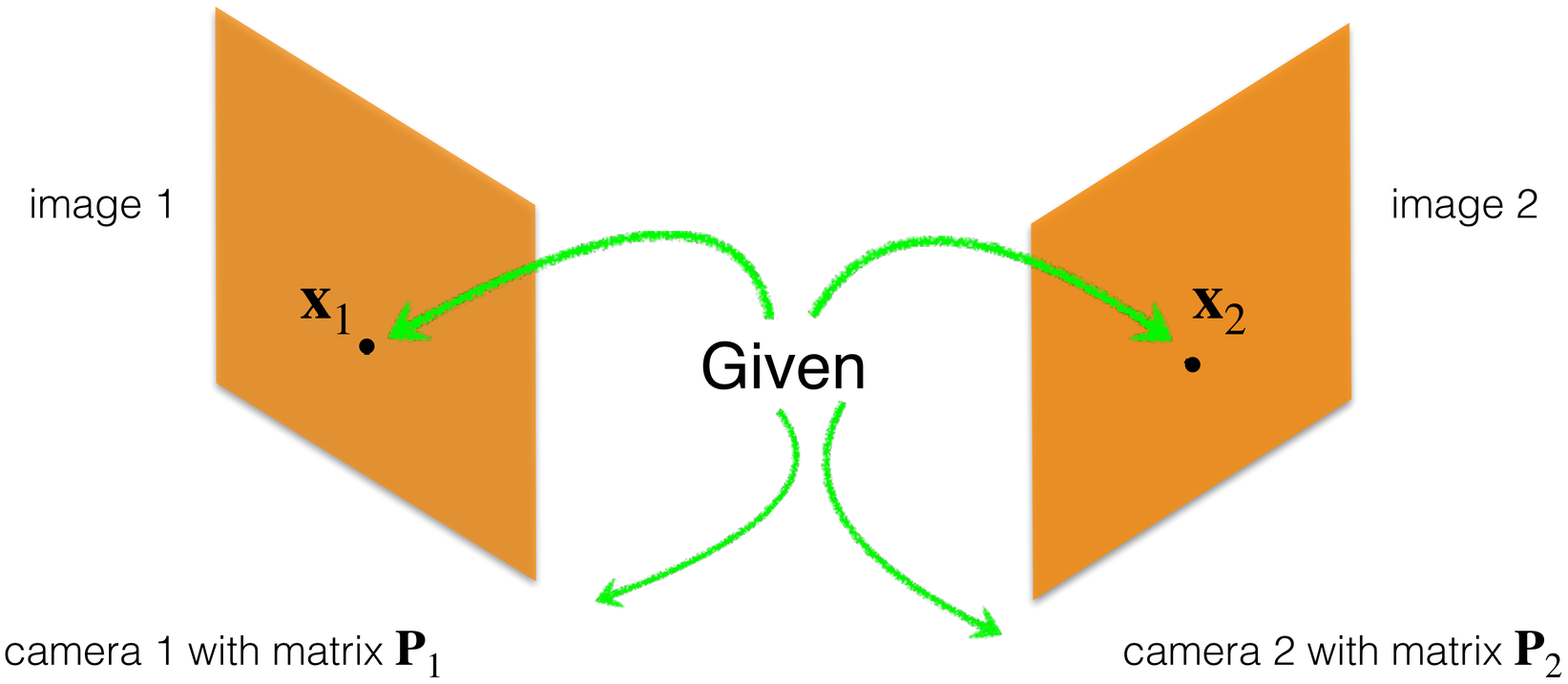


Known $\mathbf{P}_1, \mathbf{P}_2$!

What's 3D location of point?



Triangulation



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

Triangulation

Which 3D points map to $\mathbf{x}_1$?

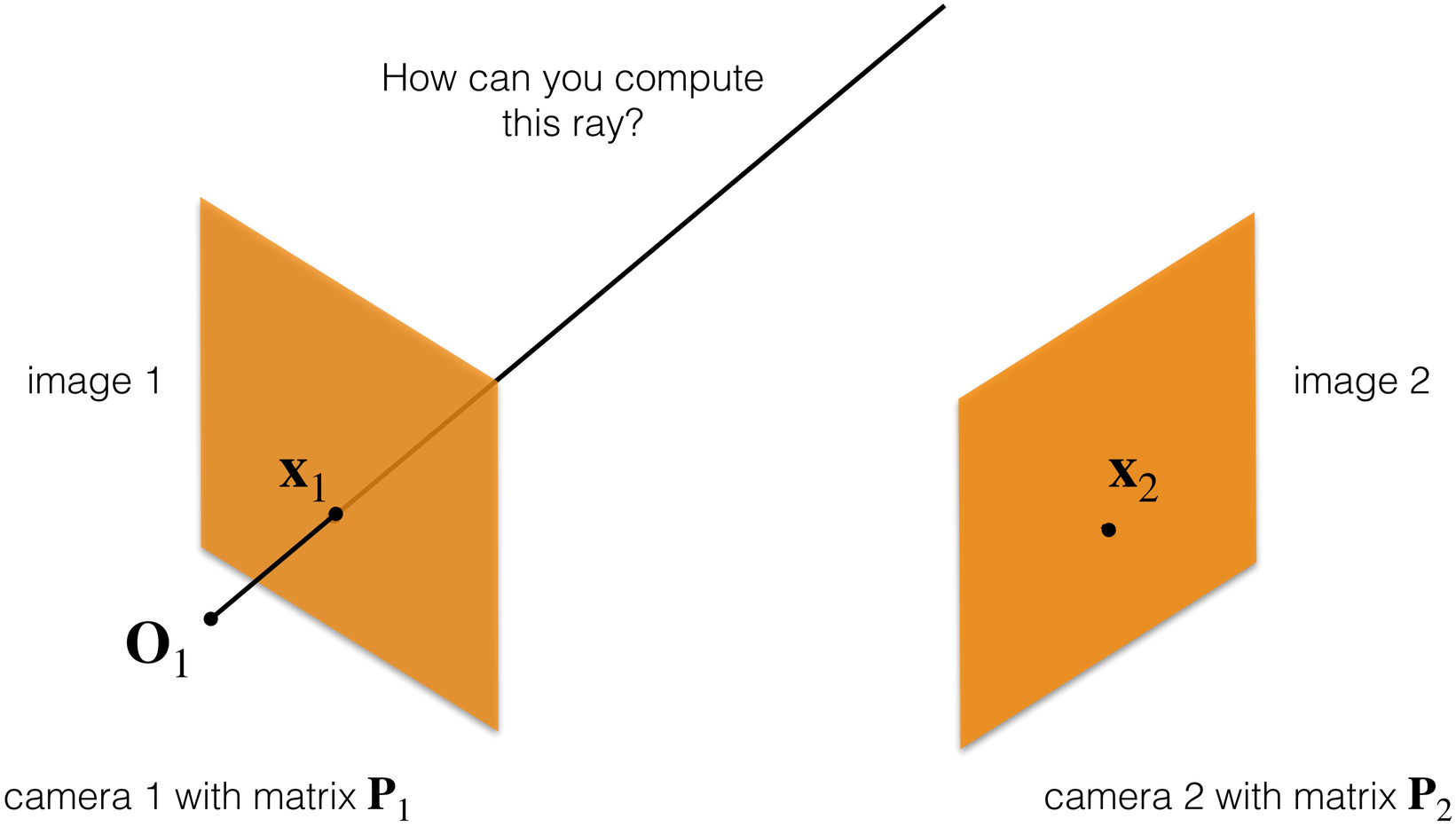


camera 1 with matrix $\mathbf{P}_1$

camera 2 with matrix $\mathbf{P}_2$

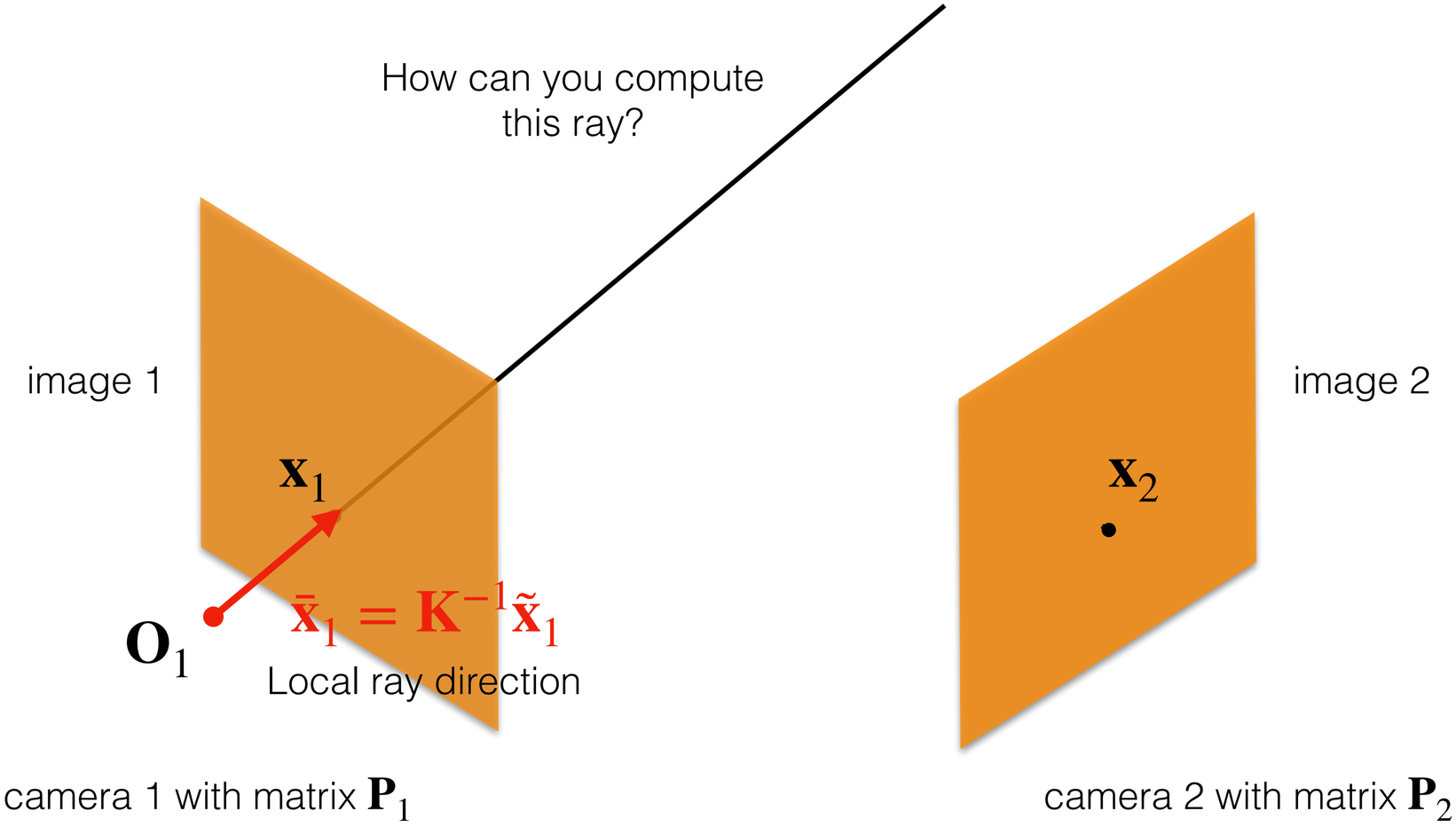
Adapted from: CMU 16-385 (Yannis, Kris)

Triangulation



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

Triangulation

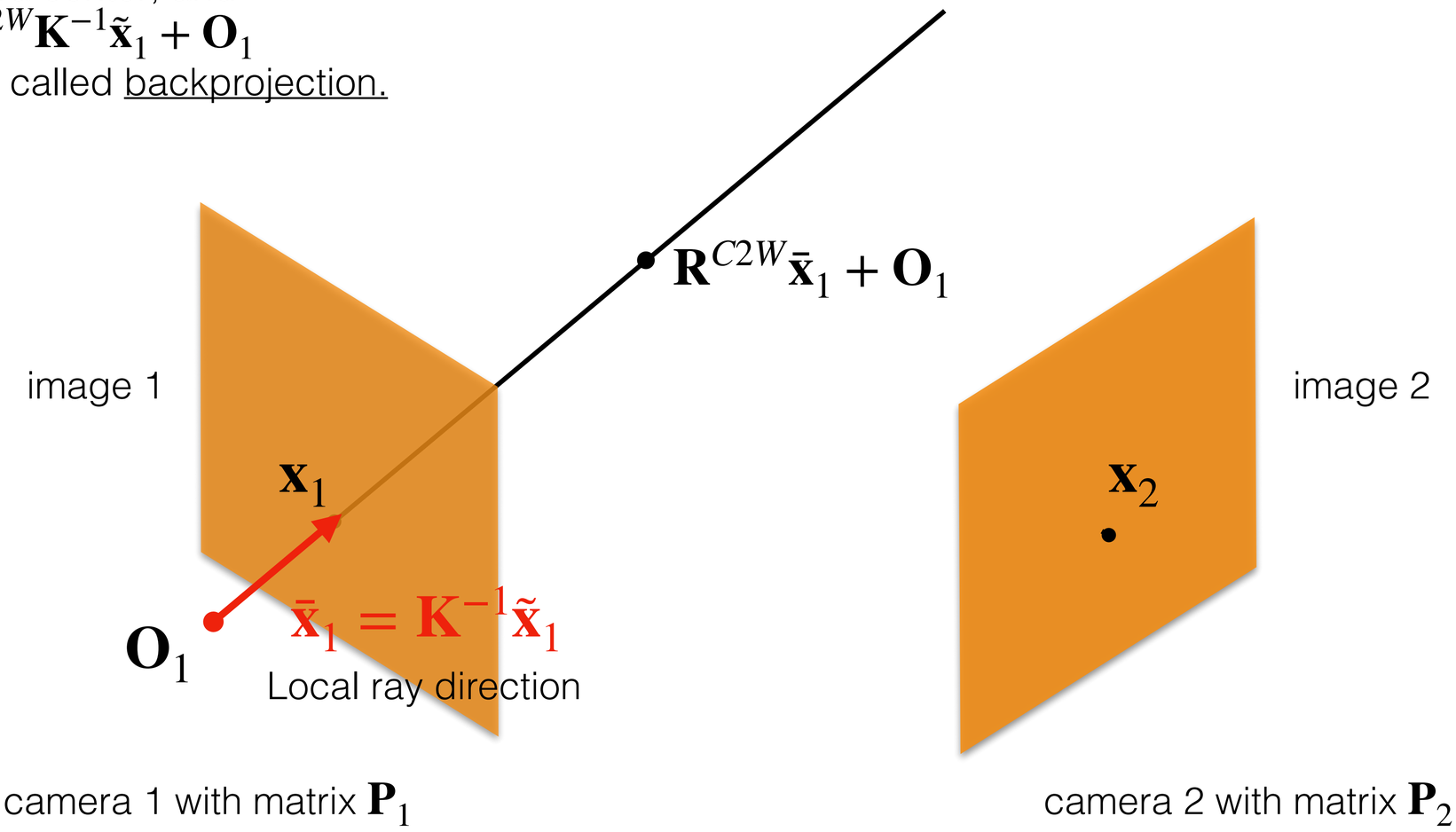


Adapted from: CMU 16-385 (Yannis, Kris)

Triangulation

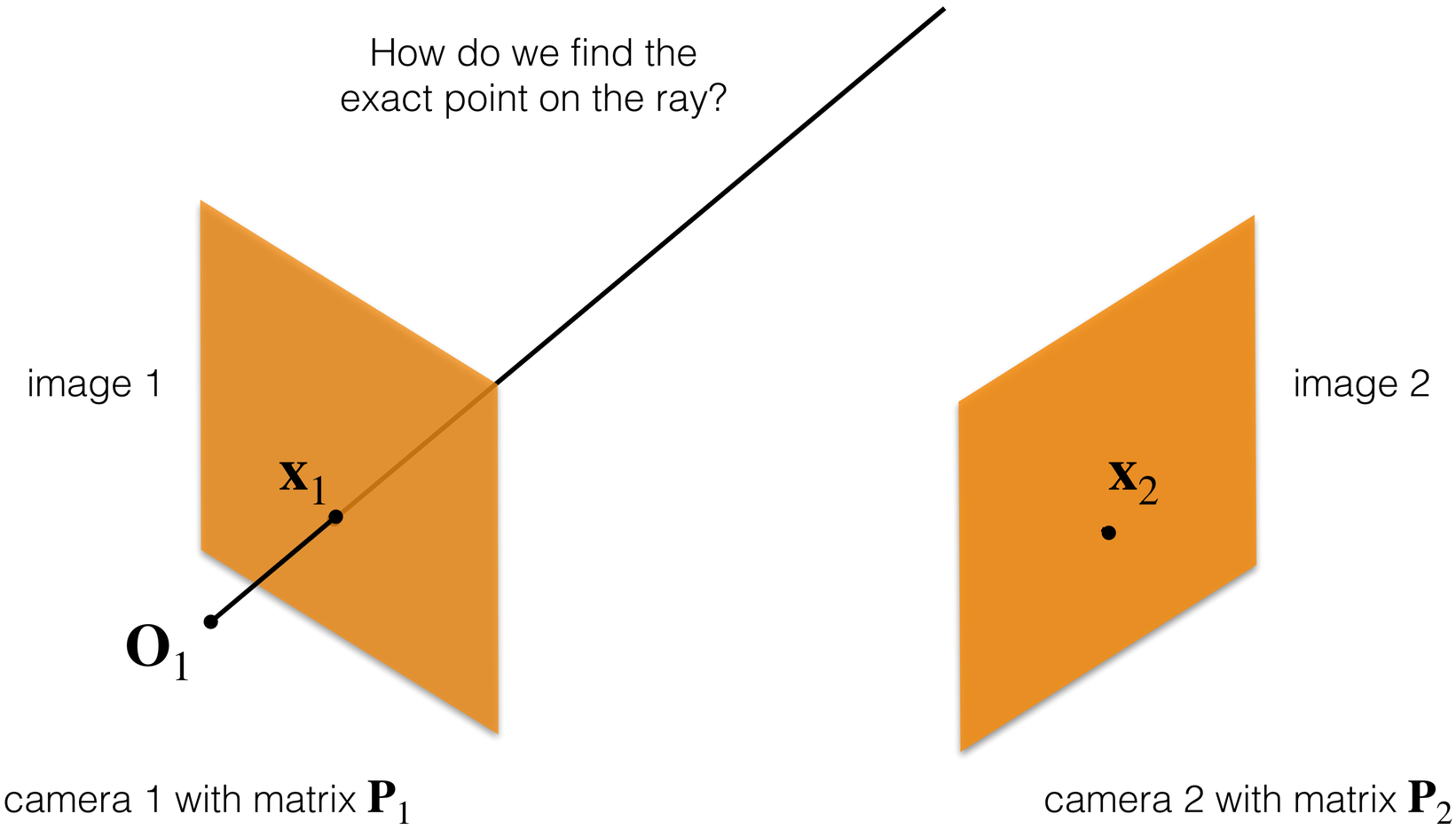
Create two points on the ray:
1) find the camera center; and
2) Compute $\mathbf{R}^{C2W}\mathbf{K}^{-1}\tilde{\mathbf{x}}_1 + \mathbf{O}_1$

This procedure is called backprojection.



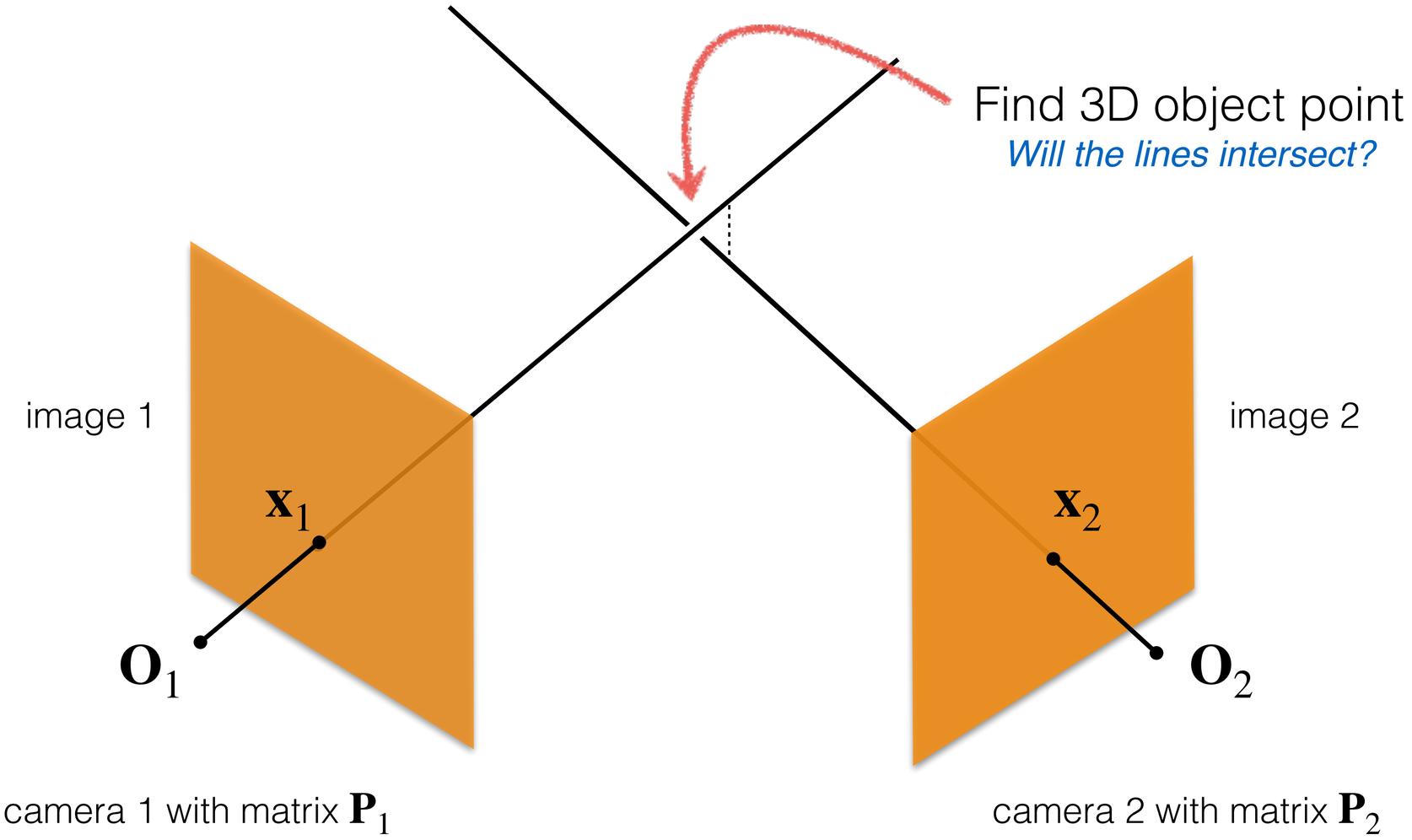
Adapted from: CMU 16-385 (Yannis, Kris)

Triangulation



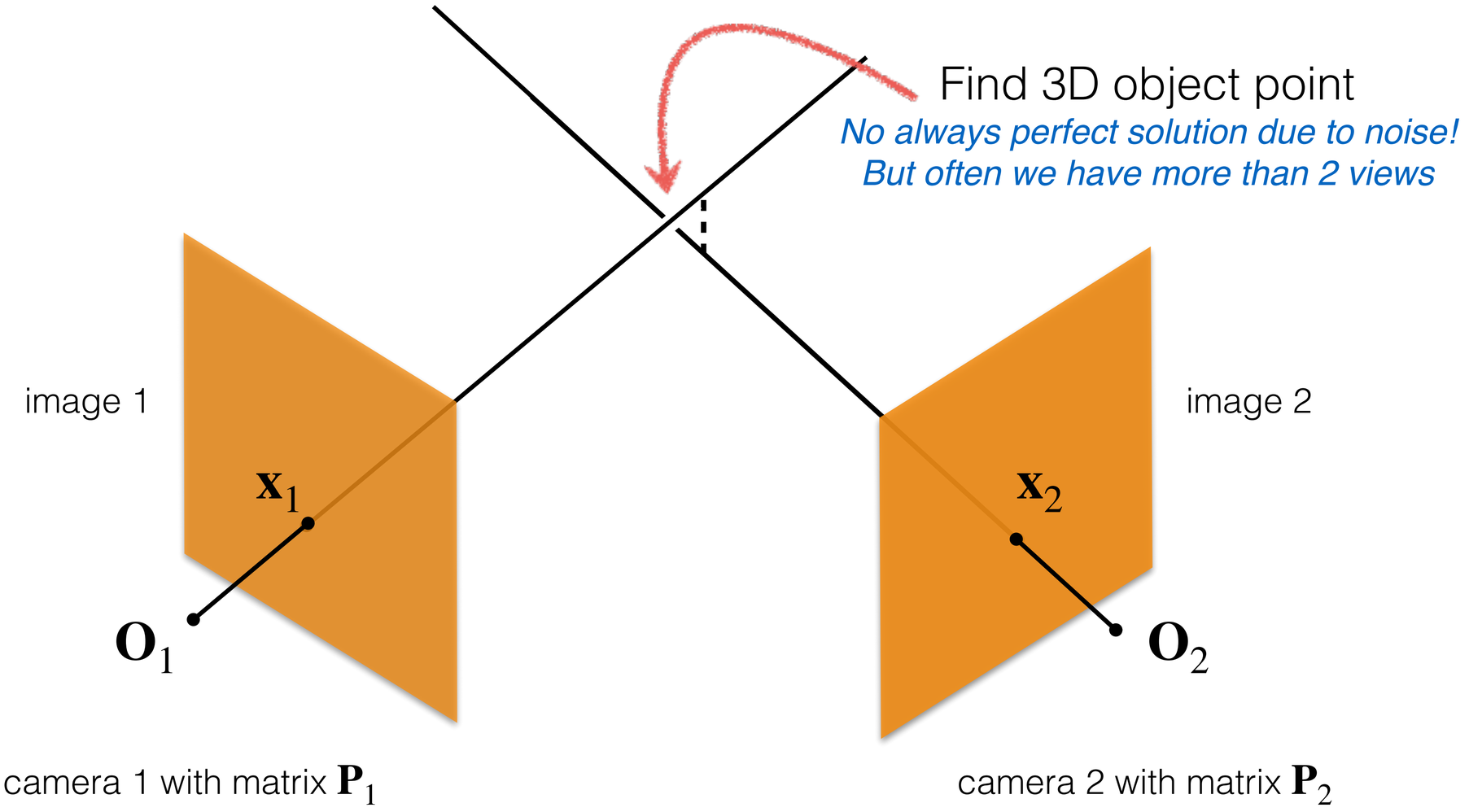
Adapted from: CMU 16-385 (Yannis, Kris)

Triangulation



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

Triangulation



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

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

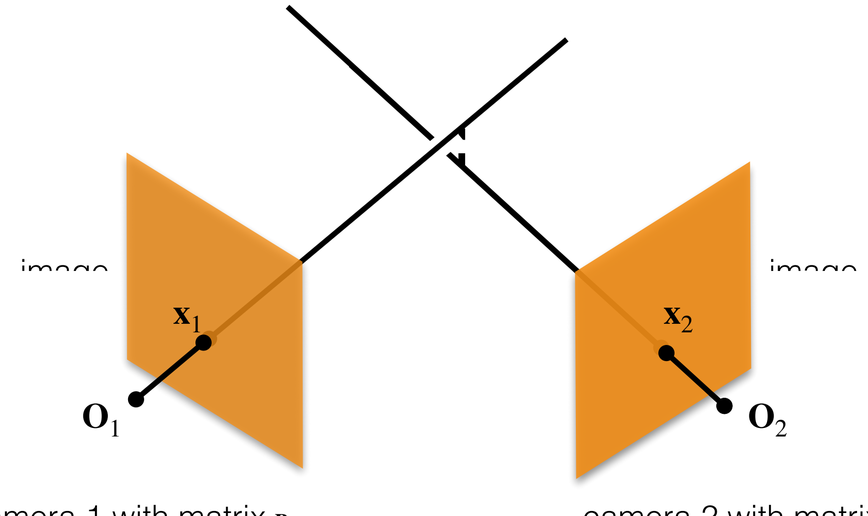
Then we can solve a little least squares problem:

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

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

In a nutshell

- If we know the camera matrices, 3D->2D perspective is linear in 3D point
- Given 2D point correspondences (and known camera matrices), we can solve for 3D coordinates with linear algebra
  - Direct solver or least squares version to handle noise and more views
- In order to solve for the camera matrices, we will seek the opposite: cast the problem as linear in (some) matrices.



More nutshell

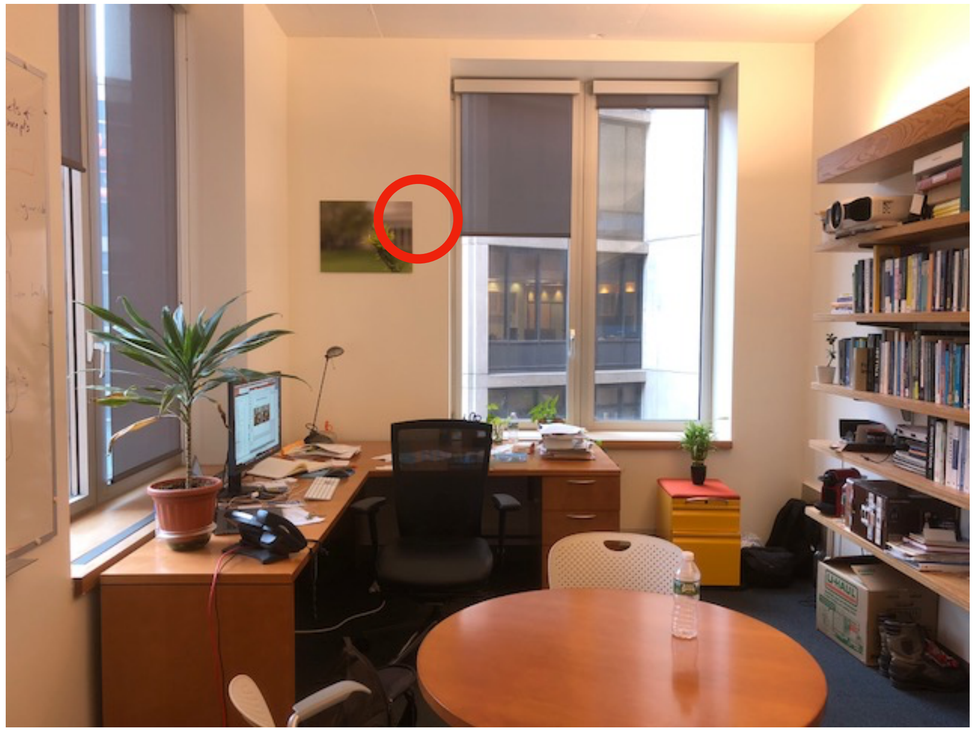
$$x_0 = \mathbf{K}_0[\mathbf{I}\,|\,0]\mathbf{C}_0^{W2C}\tilde{\mathbf{X}} = P_0 X$$

$$x_1 = \mathbf{K}_1[\mathbf{I}\,|\,0]\mathbf{C}_1^{W2C}\tilde{\mathbf{X}} = P_1 X$$

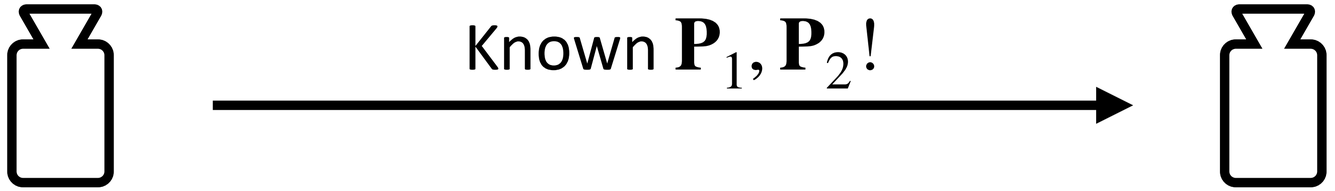
Motivation: Epipolar Lines

Where to look for the match?

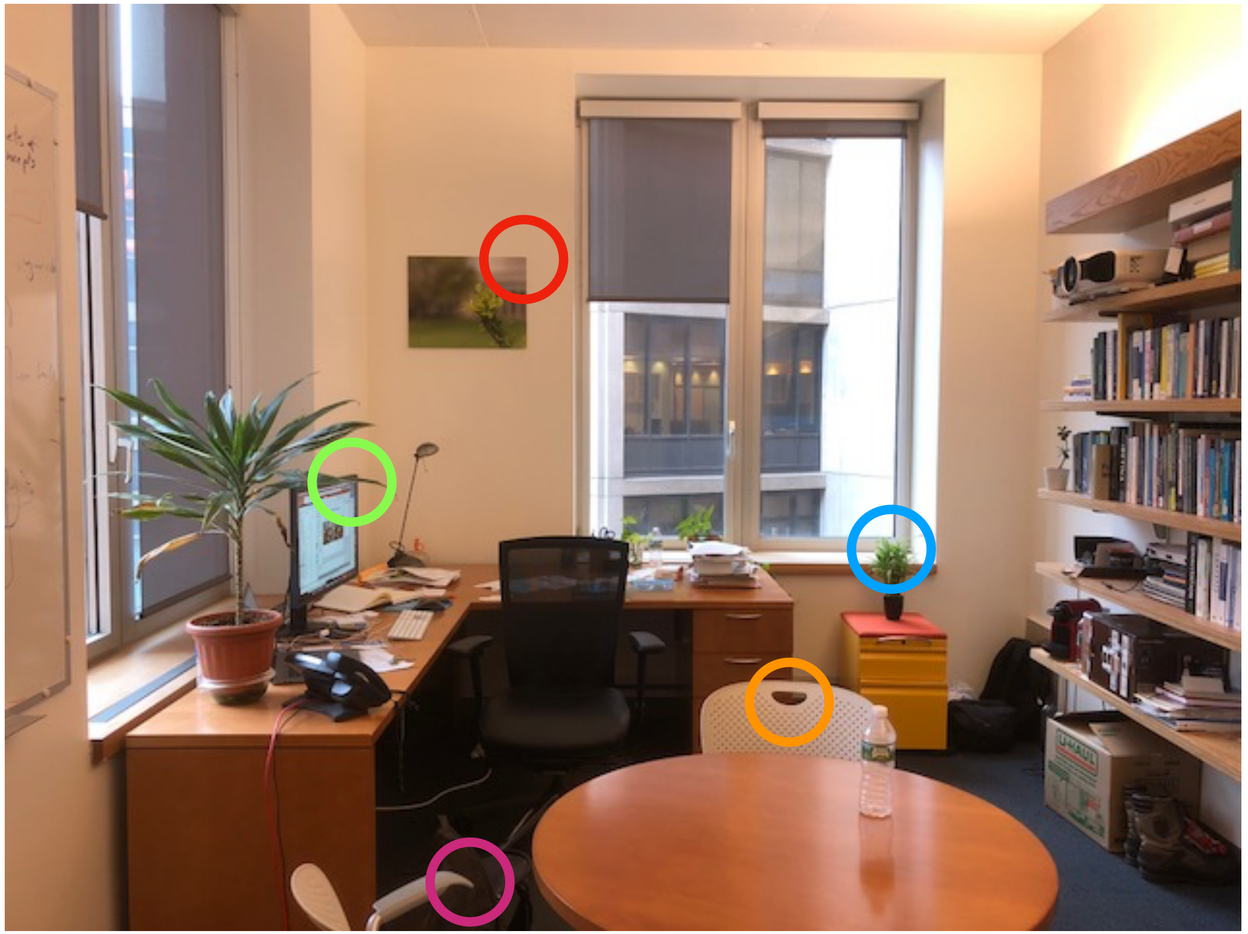
$O(n^2)$ search problem…

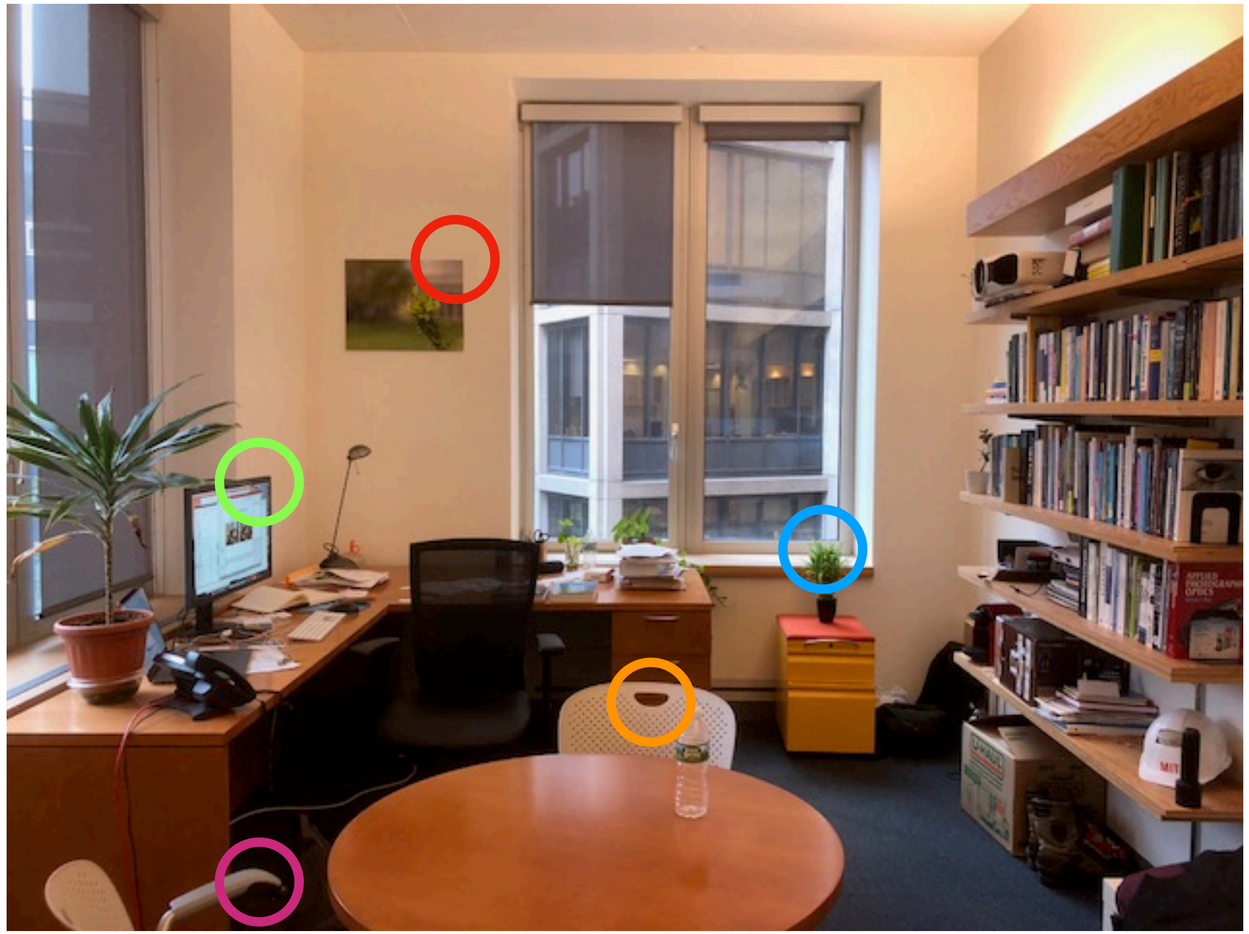


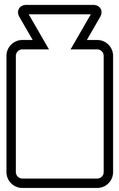




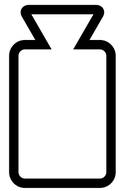
Motivation: Epipolar Lines



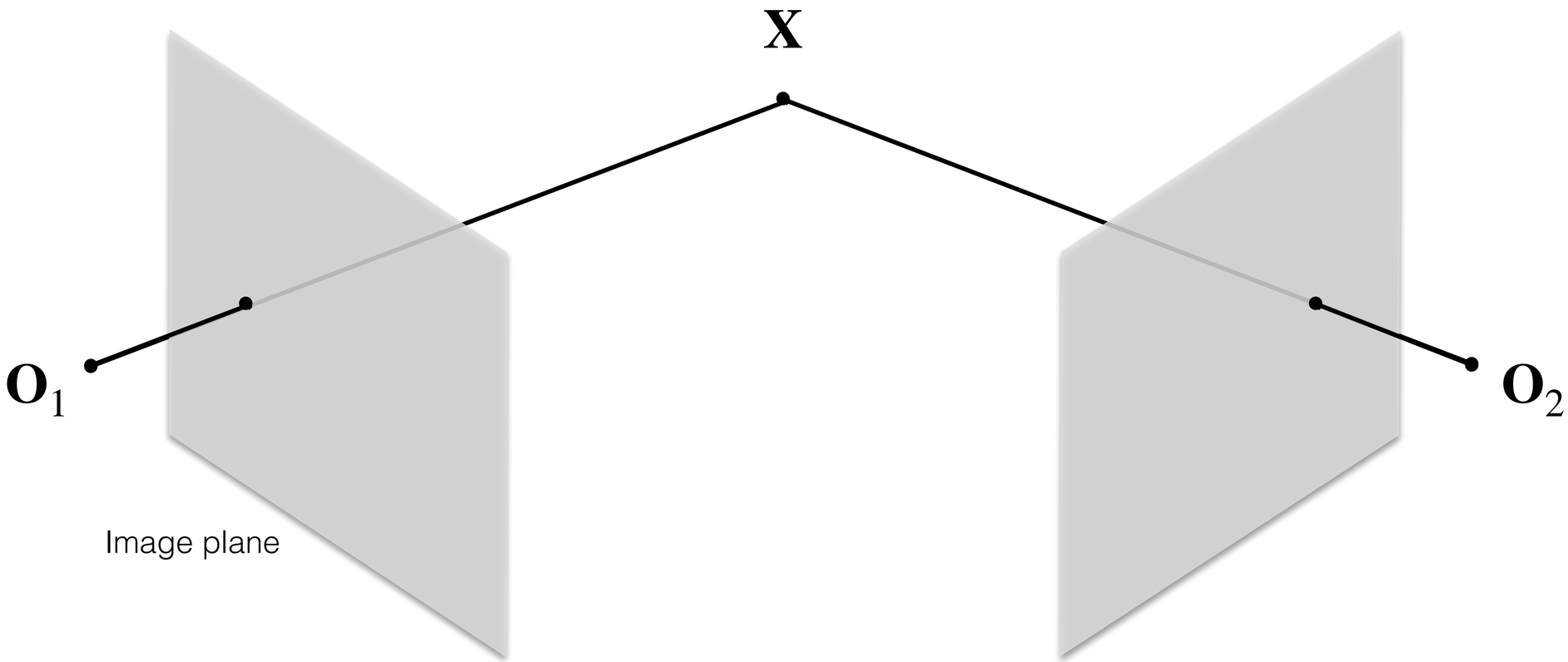




What if we know correspondences, but $\mathbf{P}_1, \mathbf{P}_2$ *aren't* known?

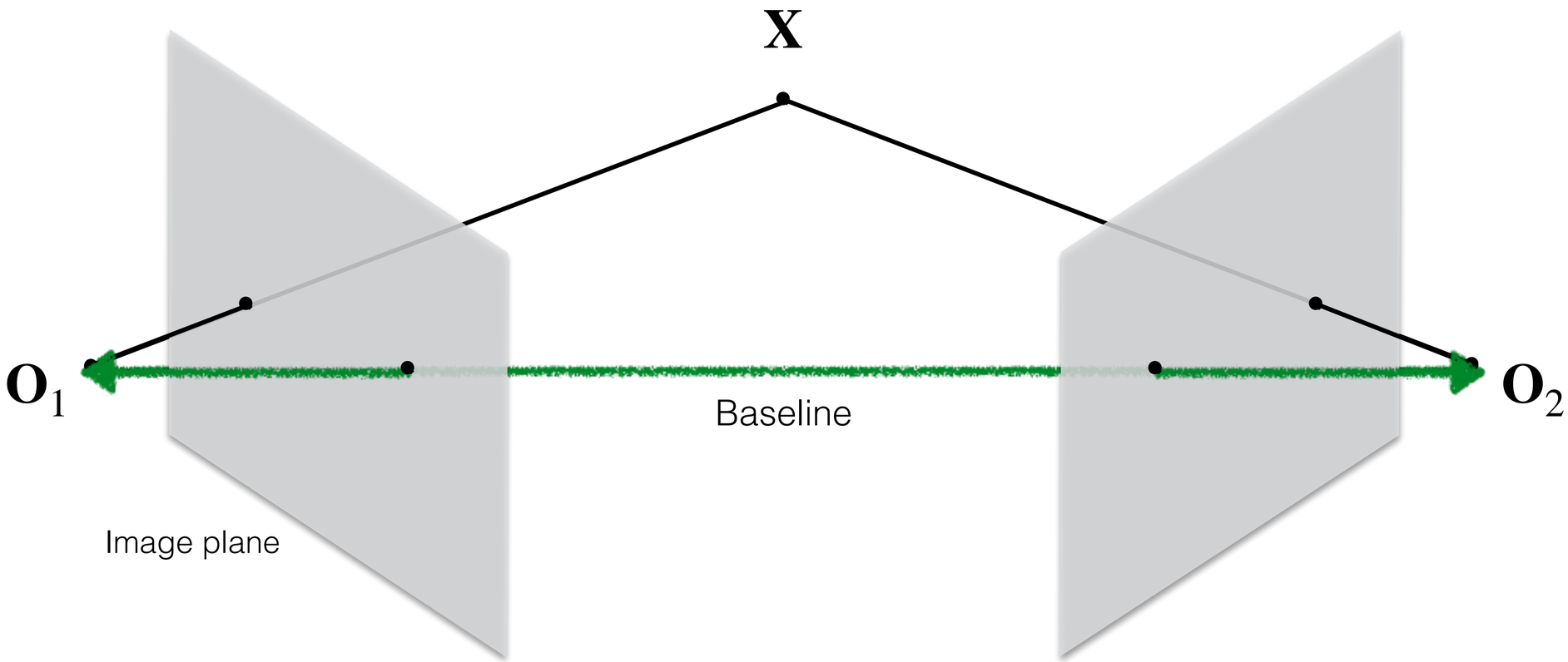


Epipolar geometry



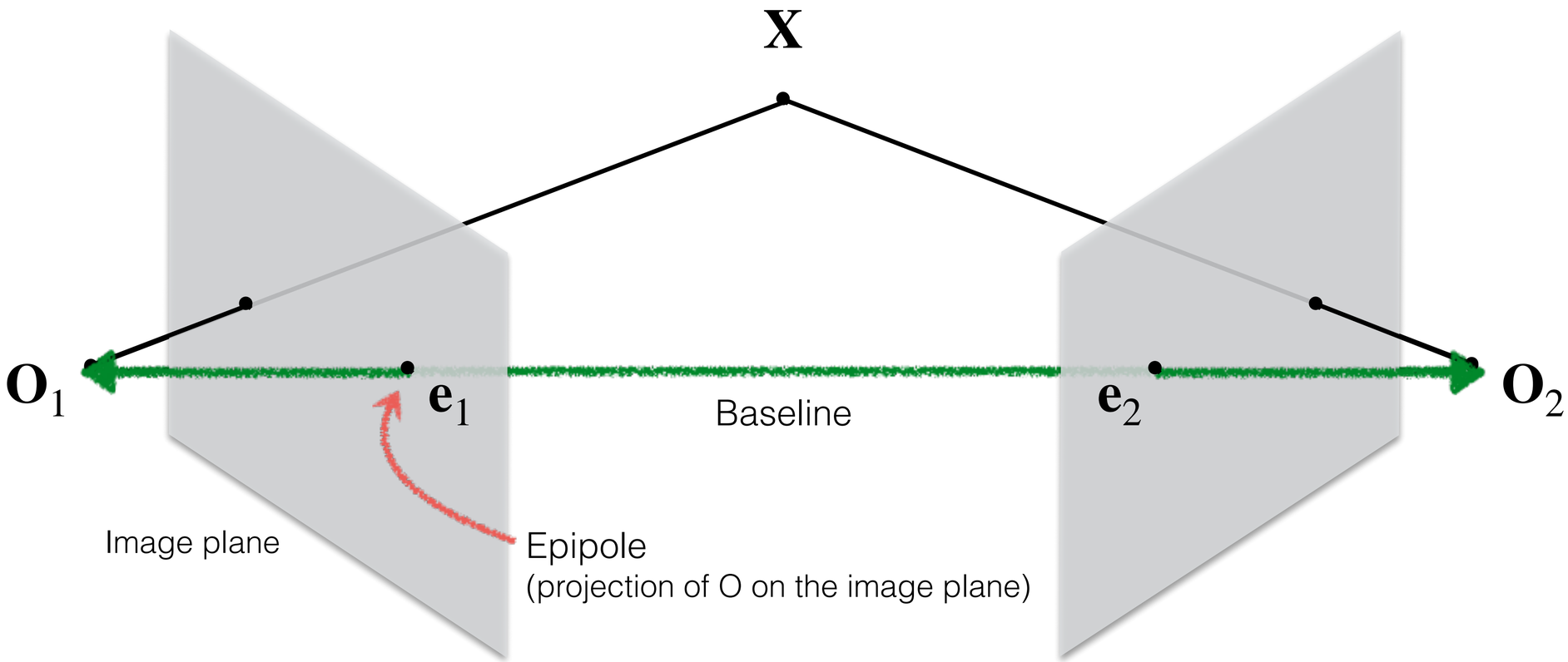
Adapted from: CMU 16-385 (Yannis, Kris)

Epipolar geometry



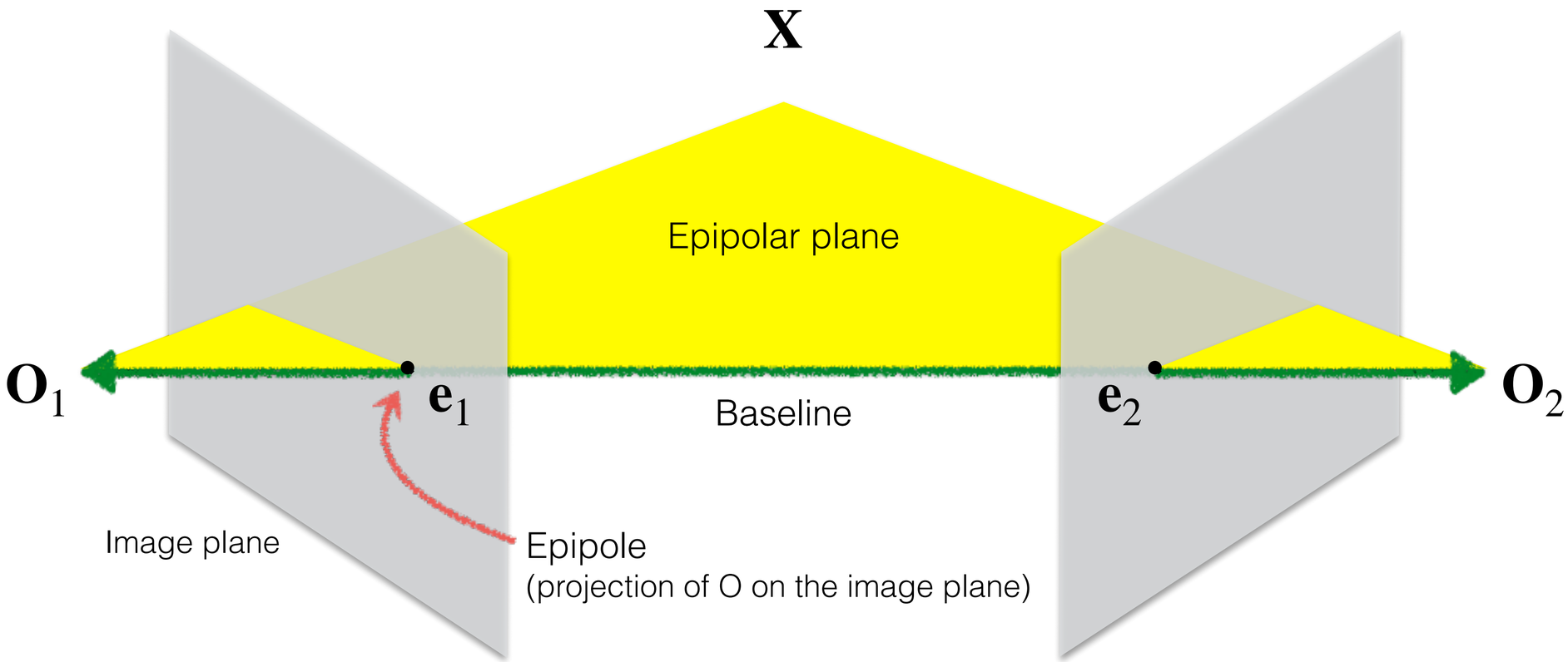
Adapted from: CMU 16-385 (Yannis, Kris)

Epipolar geometry



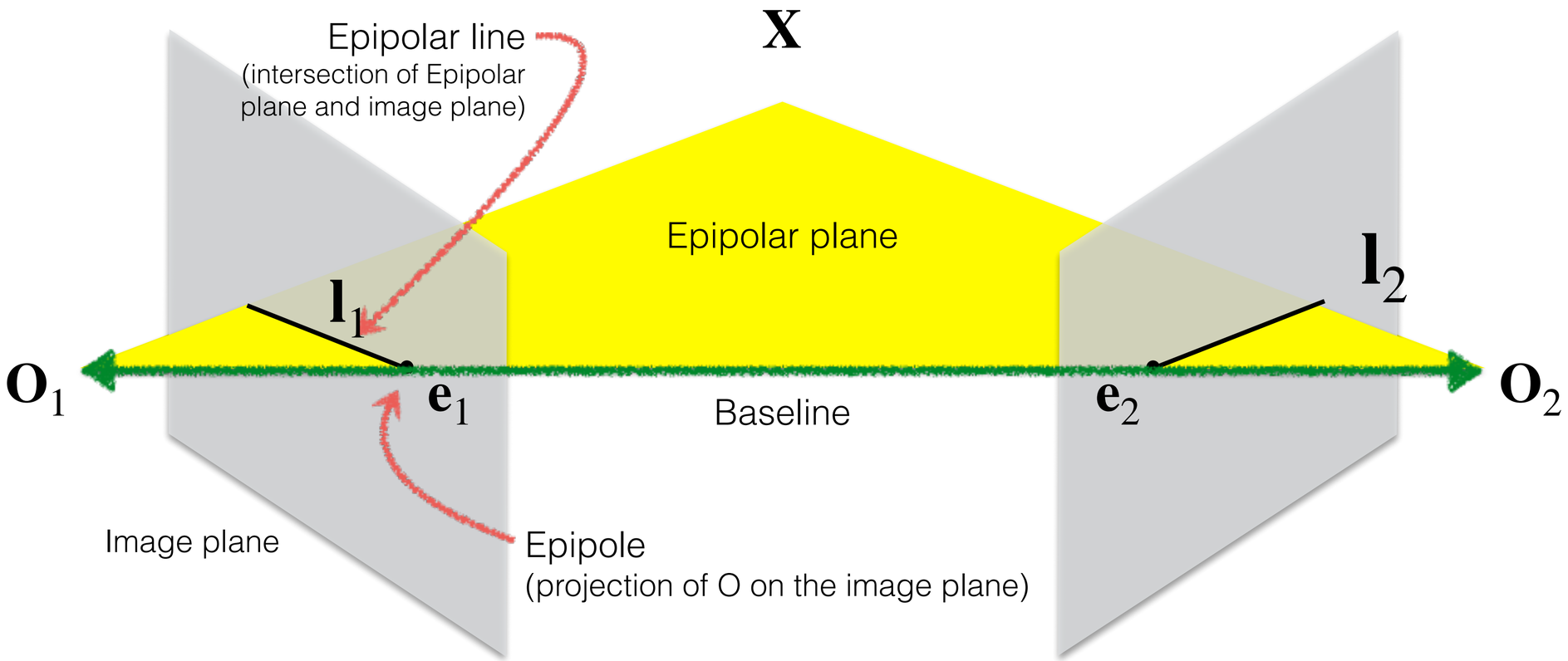
Adapted from: CMU 16-385 (Yannis, Kris)

Epipolar geometry



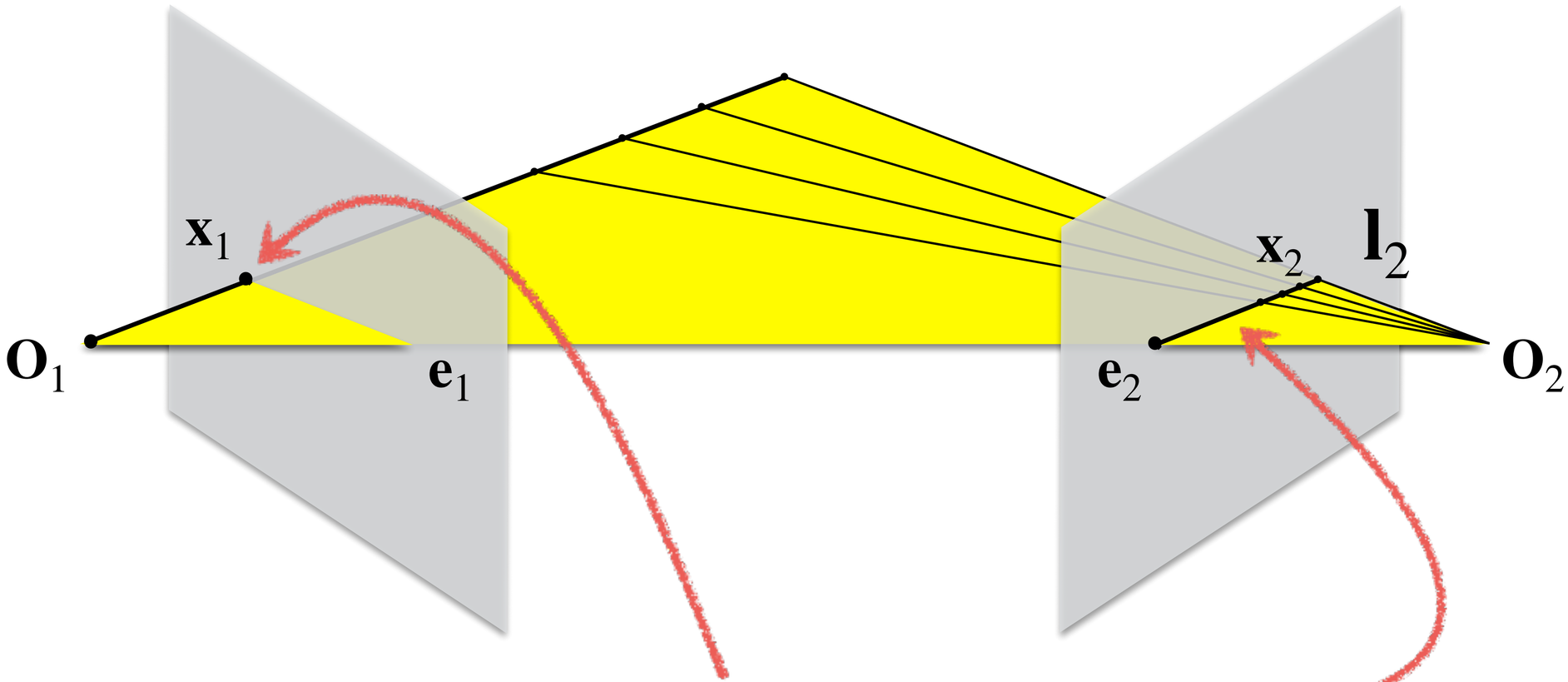
Adapted from: CMU 16-385 (Yannis, Kris)

Epipolar geometry



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

Epipolar constraint

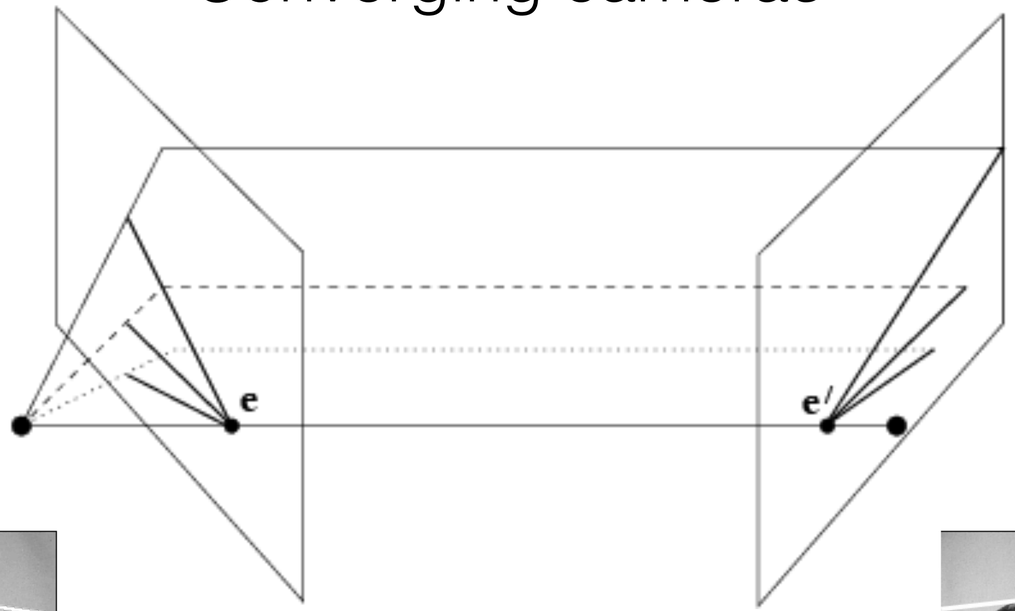


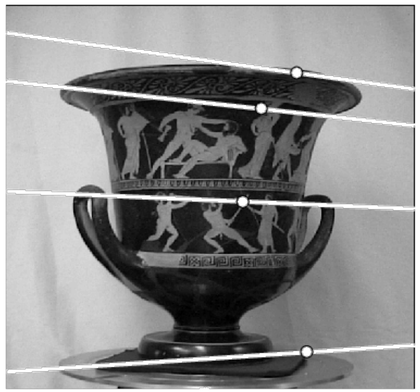
Potential matches for $\mathbf{x}_1$ lie on the epipolar line $\mathbf{l}_2$

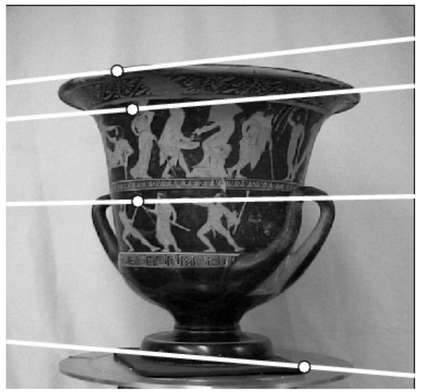
Adapted from: CMU 16-385 (Yannis, Kris)

Discuss that epipole belongs to all Epipolar lines

Converging cameras



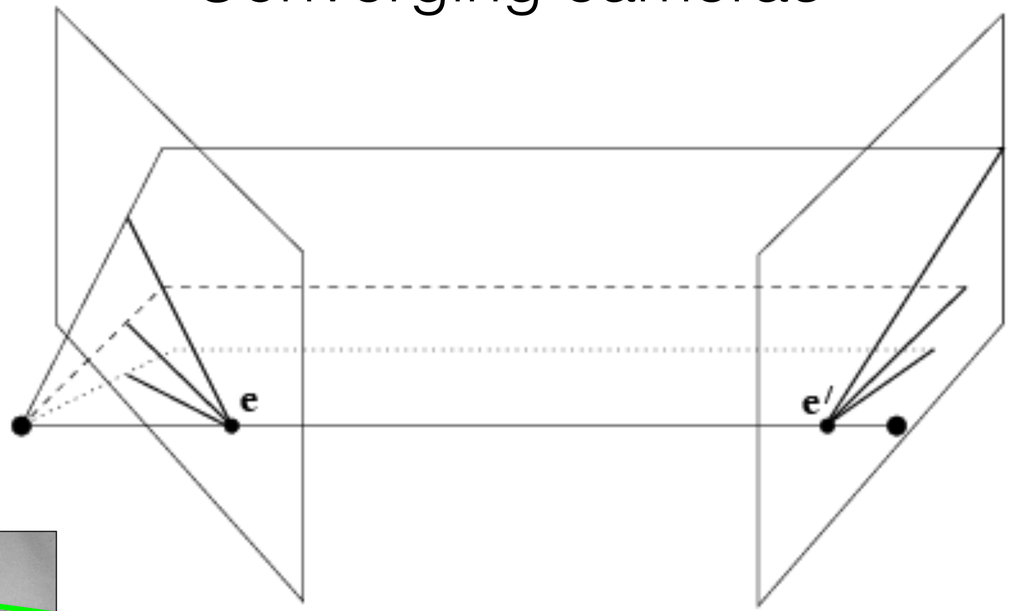


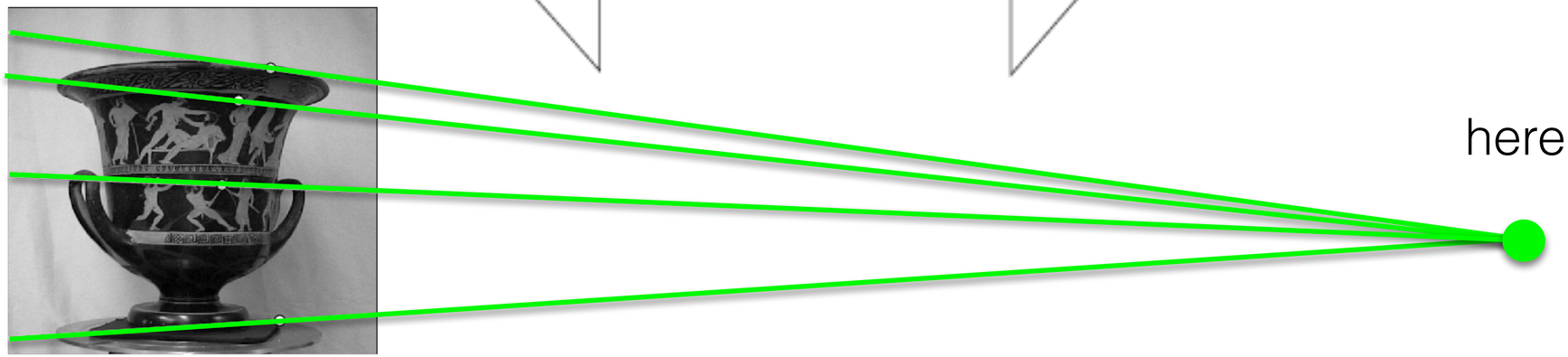


Where is the epipole in this image?

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

Converging cameras





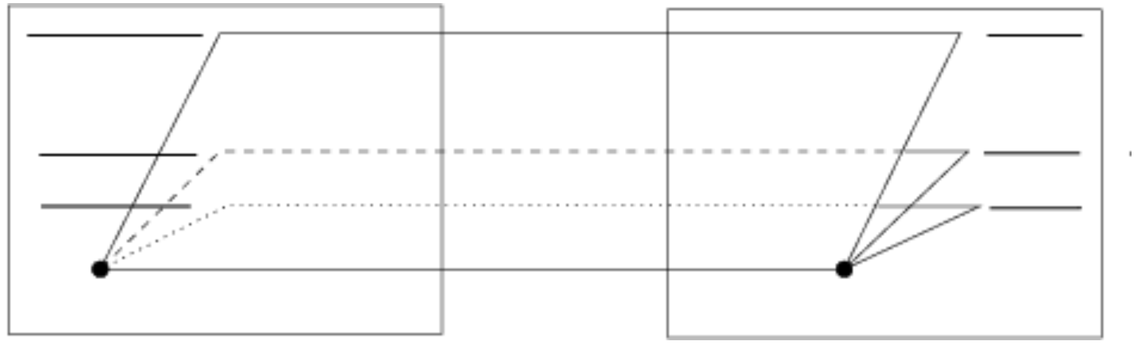
here!

Where is the epipole in this image?

It's not always in the image

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

Parallel cameras



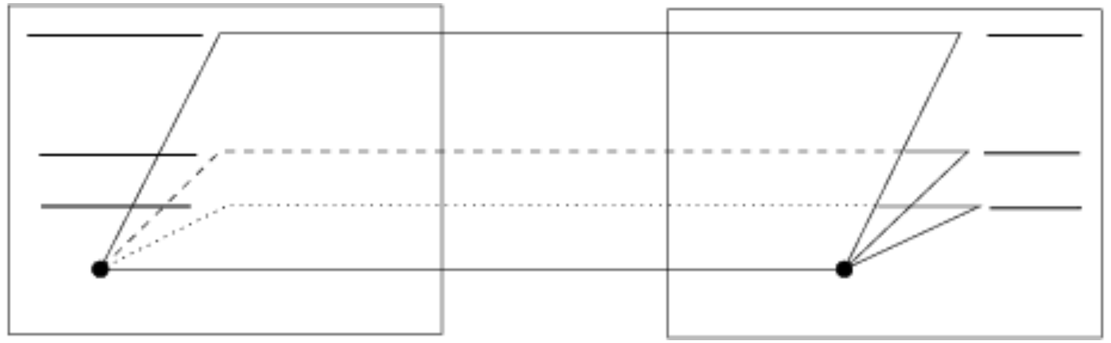


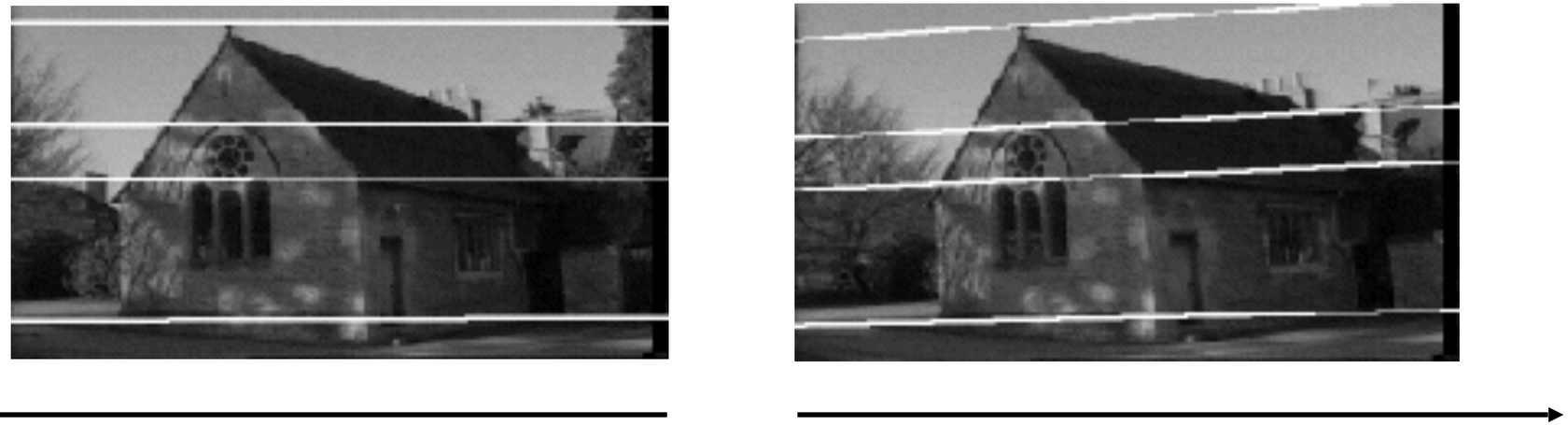


Where is the epipole?

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

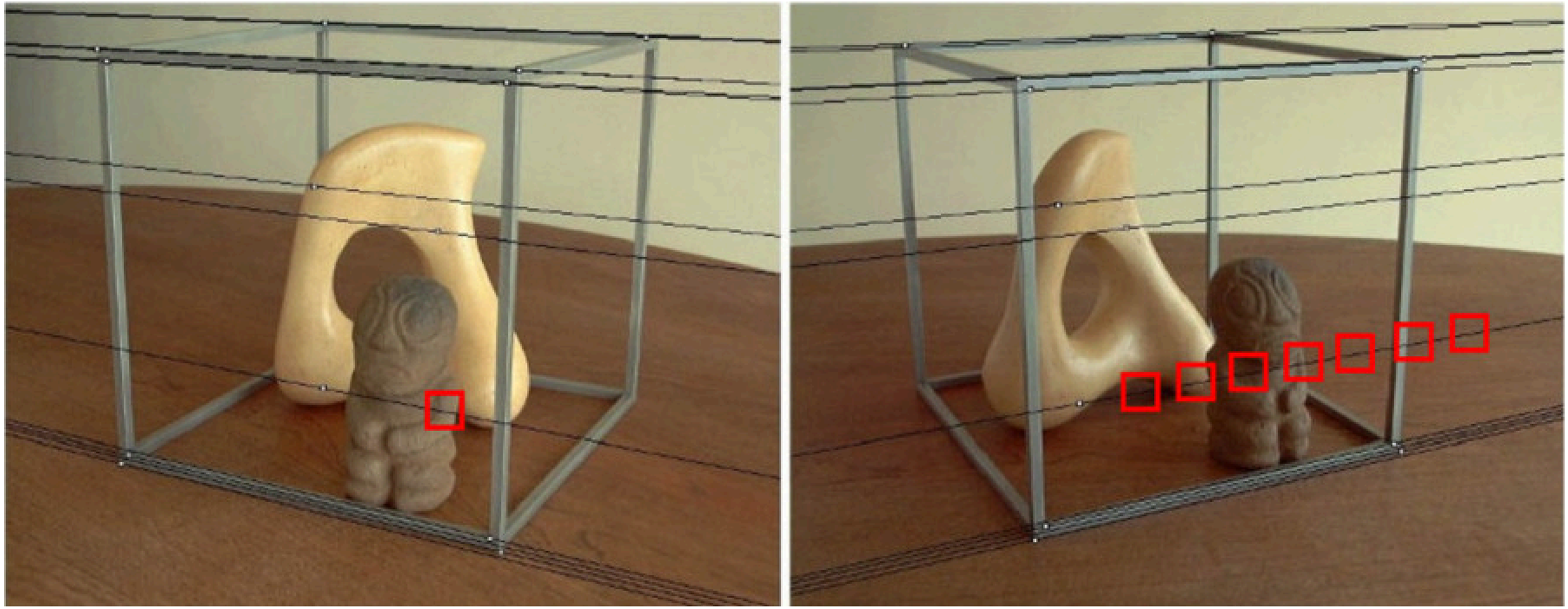
Parallel cameras





epipole at infinity

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



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

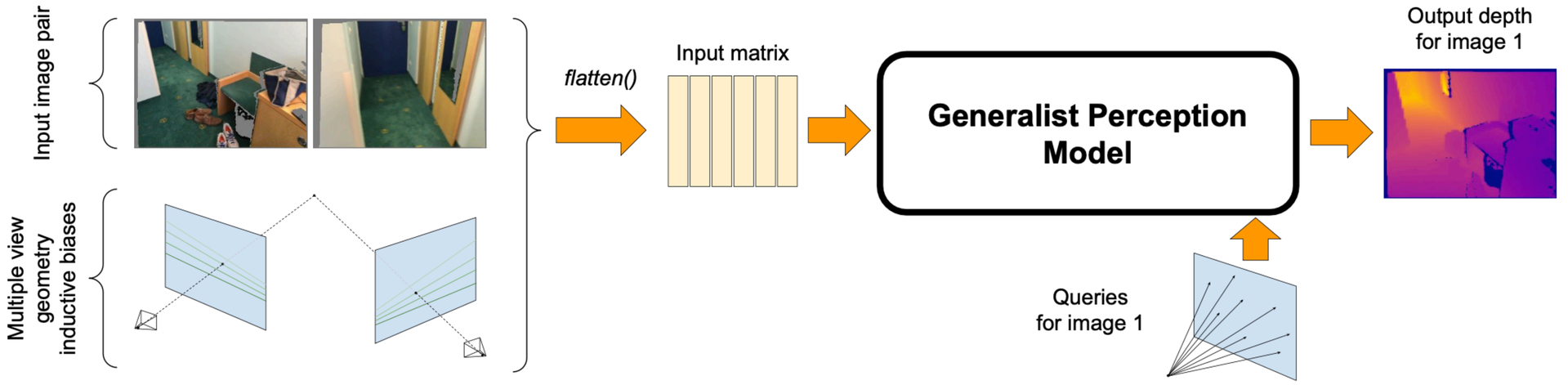
Generalizable Patch-Based Neural Rendering

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

Input-level Inductive Biases for 3D Reconstruction

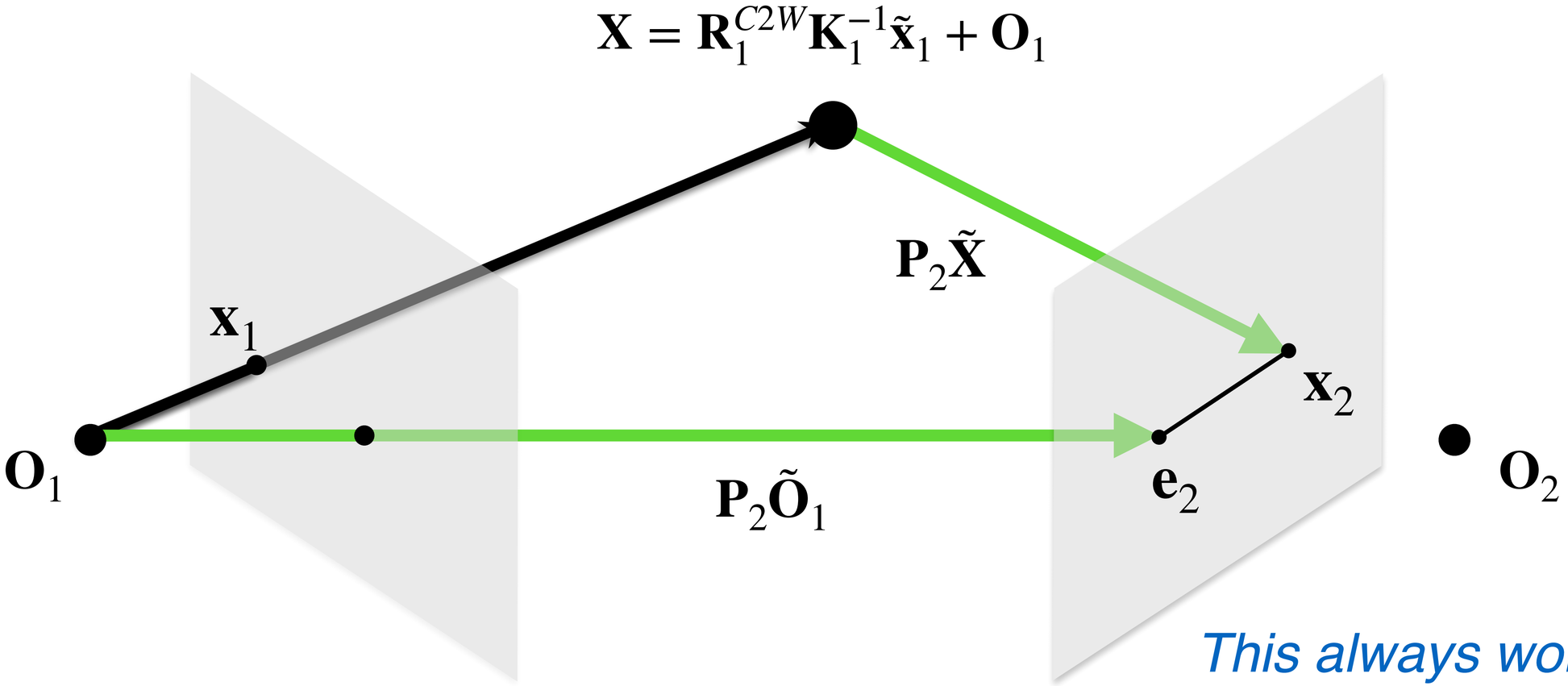
Wang Yifan[1]* Carl Doersch[2] Relja Arandjelović[2] João Carreira[2] Andrew Zisserman[2,3]

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



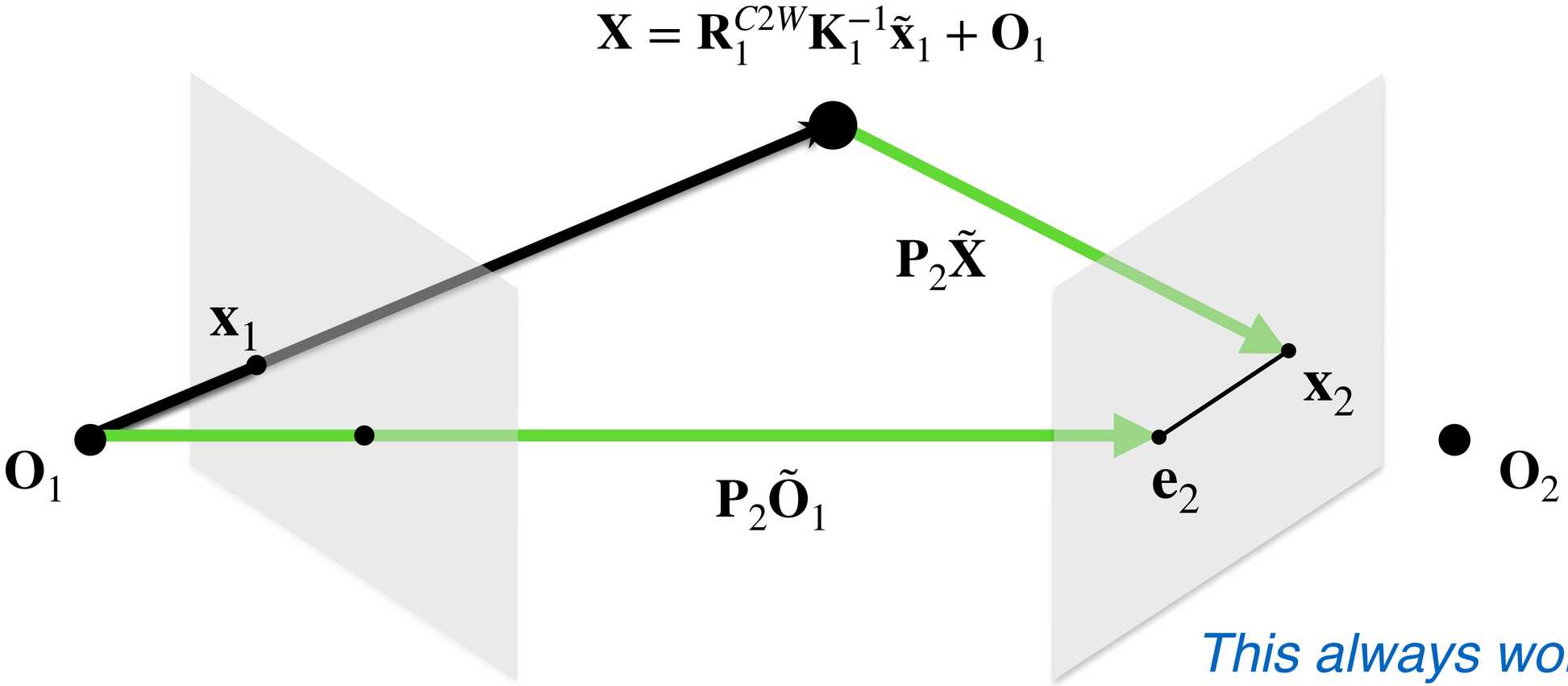
9 Mar 2022

Epipolar Lines: The Tedious Way



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

Epipolar Lines: The Tedious Way



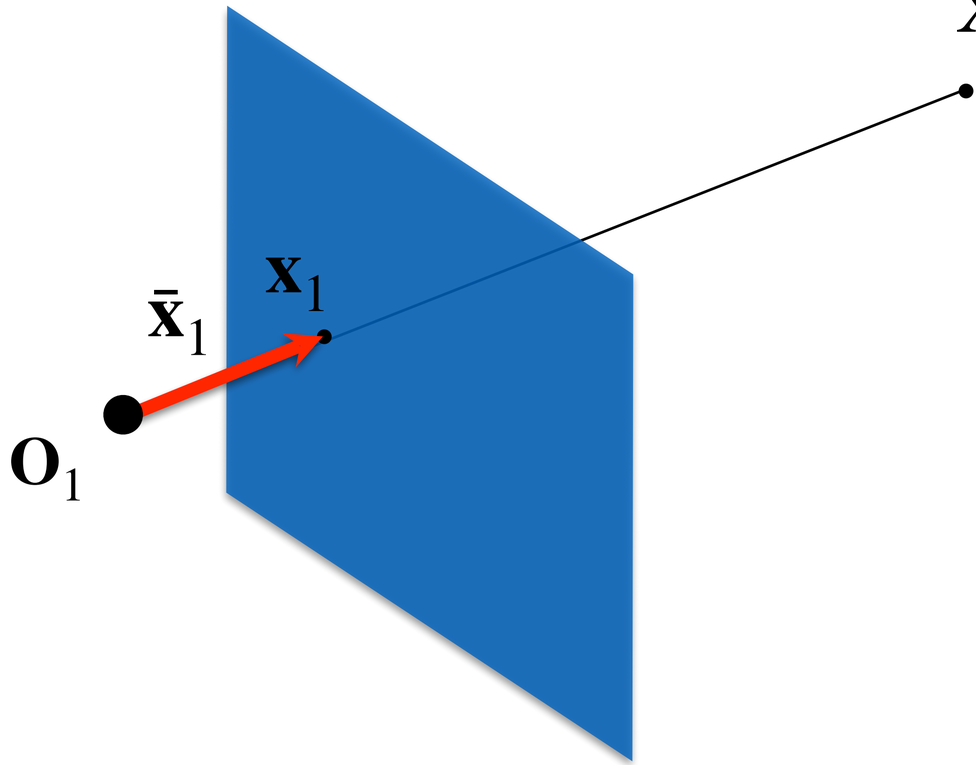
This always works ;)
But: Not clean, many steps.
Is there a better way?

Provide more intuition

- Start by explaining we are looking for a plane equation
- Look for normal. Get reference dot product.
-

Homogeneous **3D point** at depth d in coordinate system of Camera 1

$$\tilde{\mathbf{X}}_1 = \begin{pmatrix} d\tilde{\mathbf{x}}_1 \\ 1 \end{pmatrix}$$



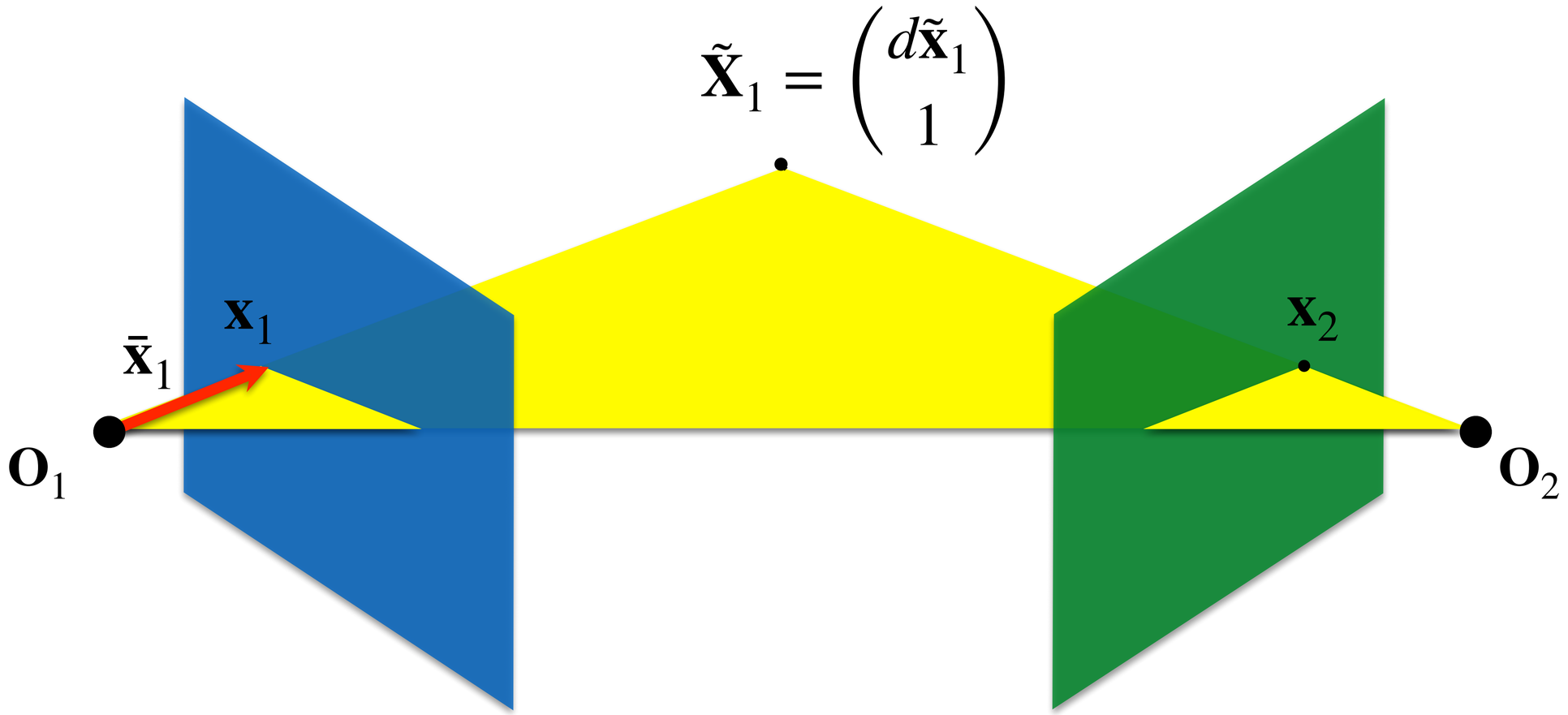
$$\bar{\mathbf{x}} = \mathbf{K}^{-1}\tilde{\mathbf{x}}$$

Pixel ray direction

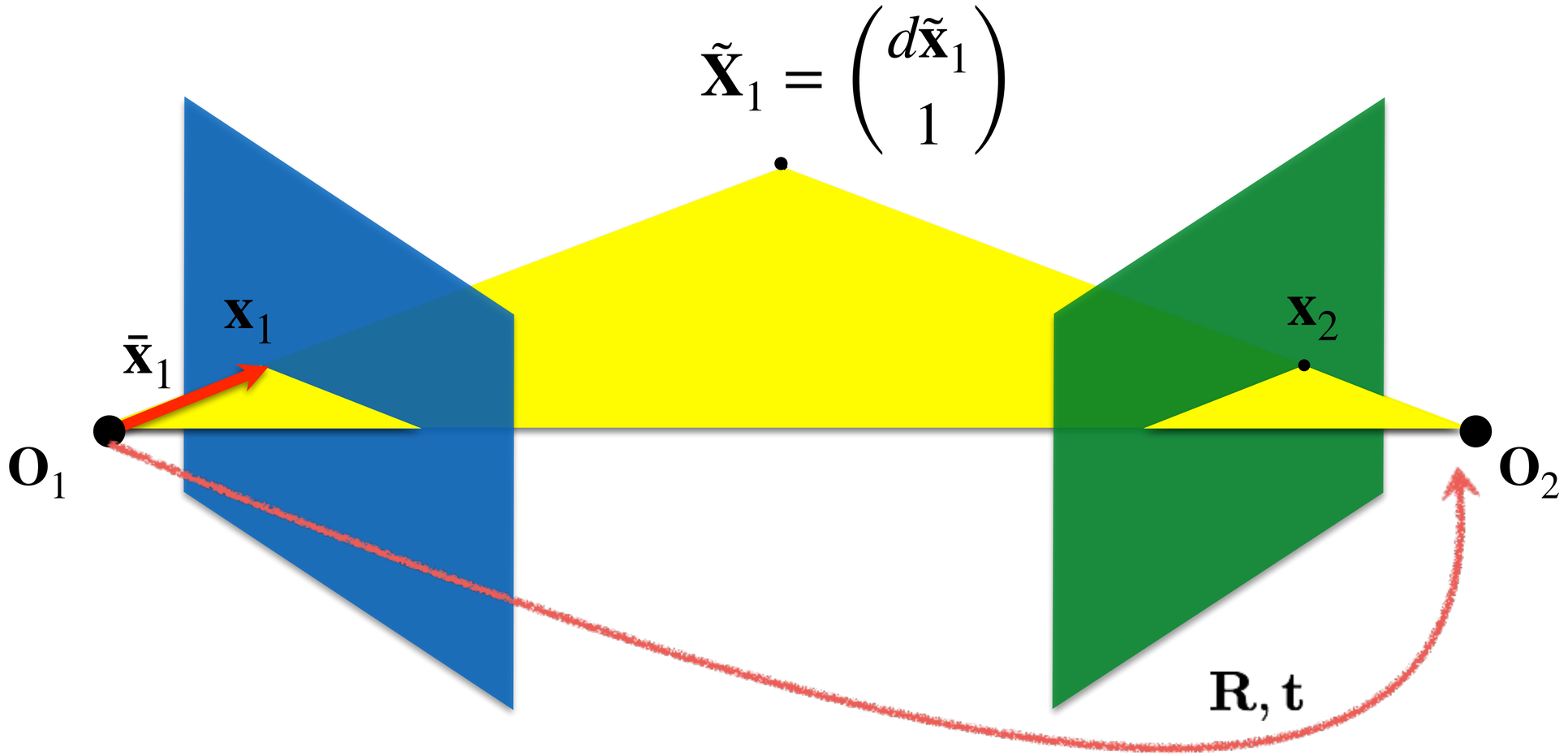
$$\tilde{\mathbf{x}} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Homogenized pixel coordinate

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



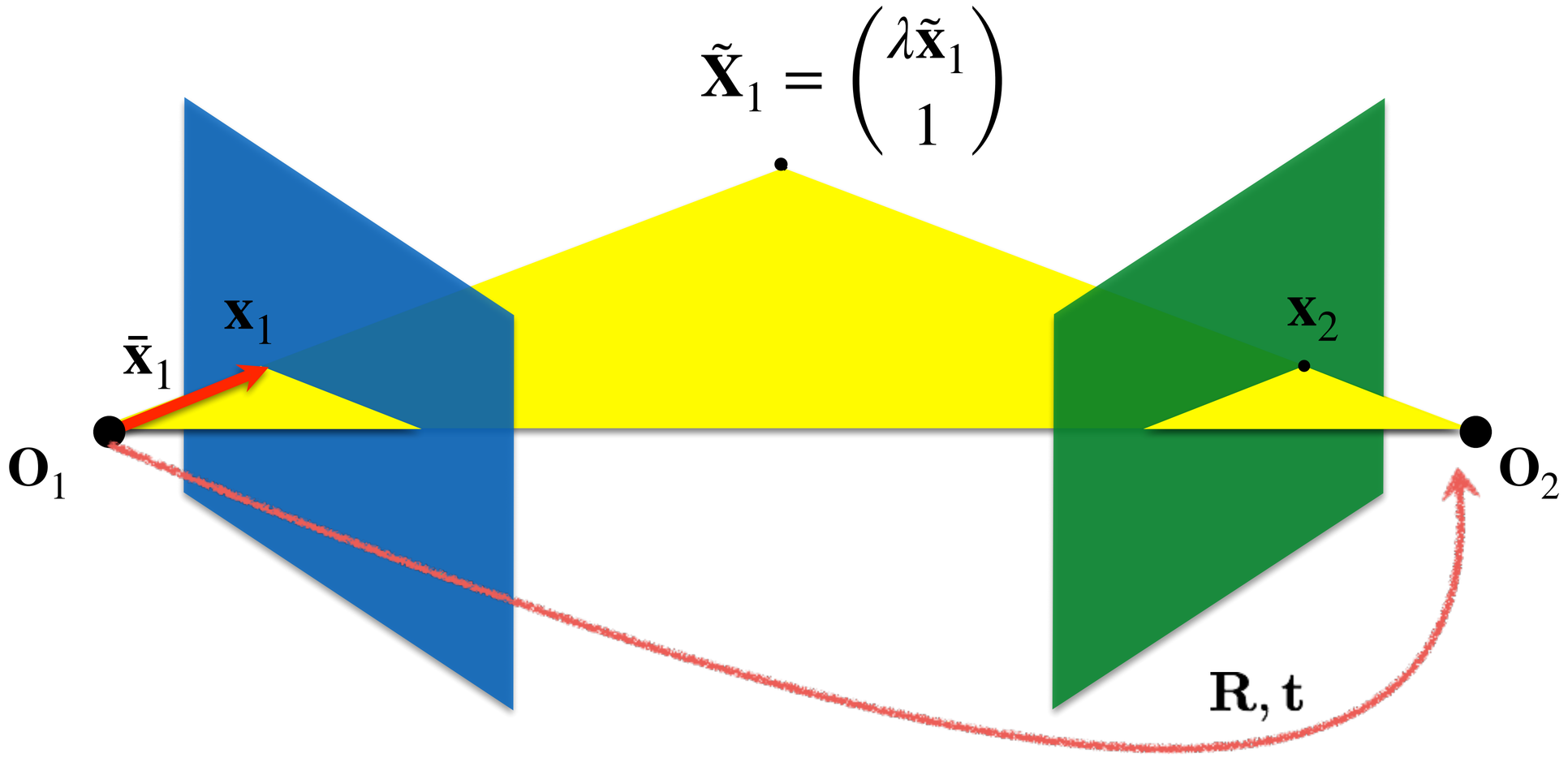
Adapted from: CMU 16-385 (Yannis, Kris)



$$\mathbf{x}_2 \sim \mathbf{K}_2[\mathbf{R} \mid \mathbf{t}] \begin{pmatrix} d\mathbf{K}_1^{-1}\mathbf{x}_1 \\ 1 \end{pmatrix}$$

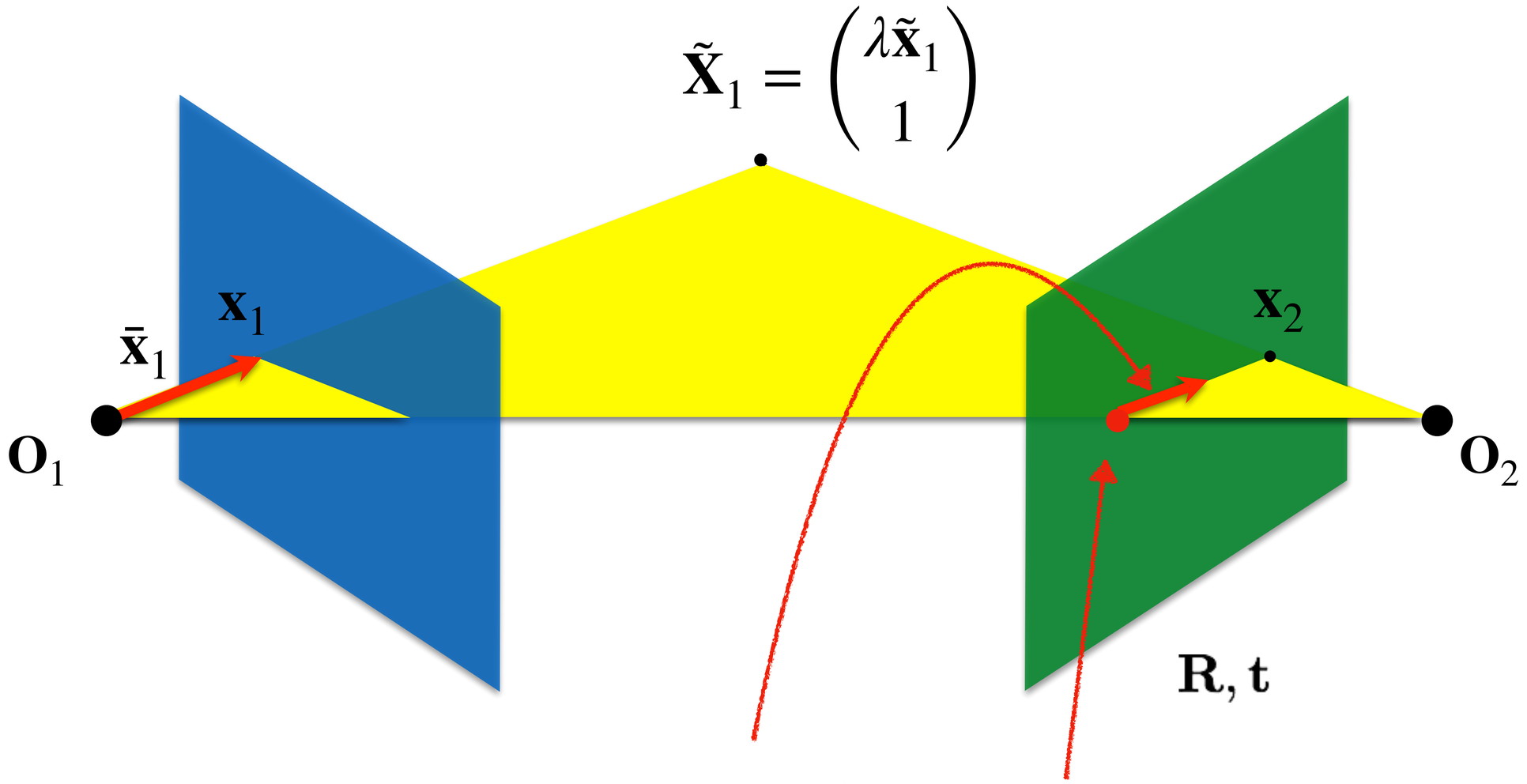
↓ Equal up to scale factor

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



$$\mathbf{x}_2 \sim d(\mathbf{K}_2\mathbf{R}\mathbf{K}_1^{-1}\mathbf{x}_1) + (\mathbf{K}_2\mathbf{t})$$

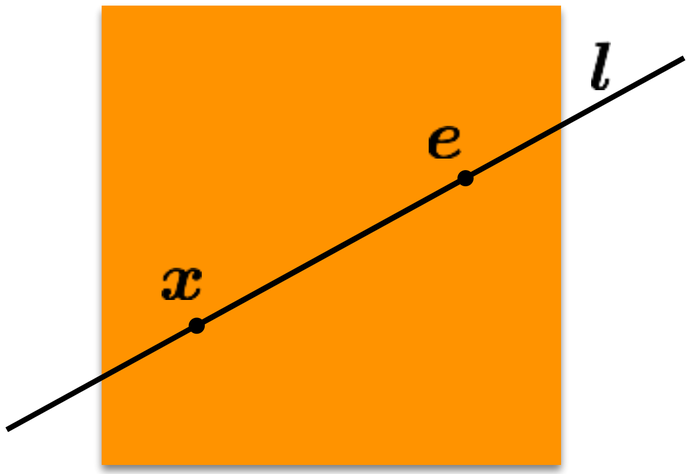
Adapted from: CMU 16-385 (Yannis, Kris)



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

Lines in Homogeneous Coordinates

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



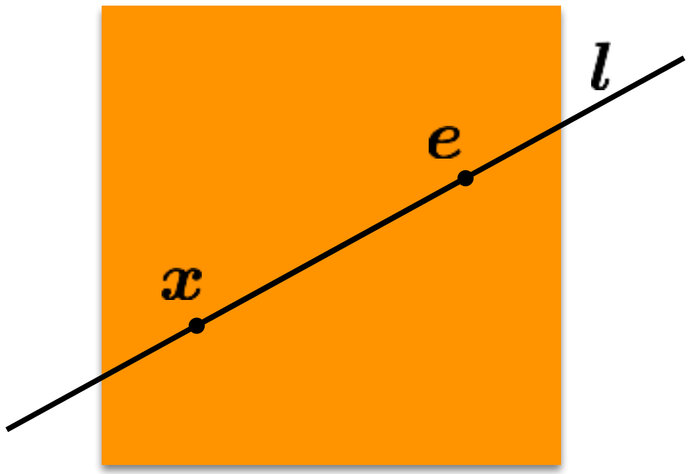
If the point $\mathbf{x}$ is on the epipolar line $\mathbf{l}$ then

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

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

Lines in Homogeneous Coordinates

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



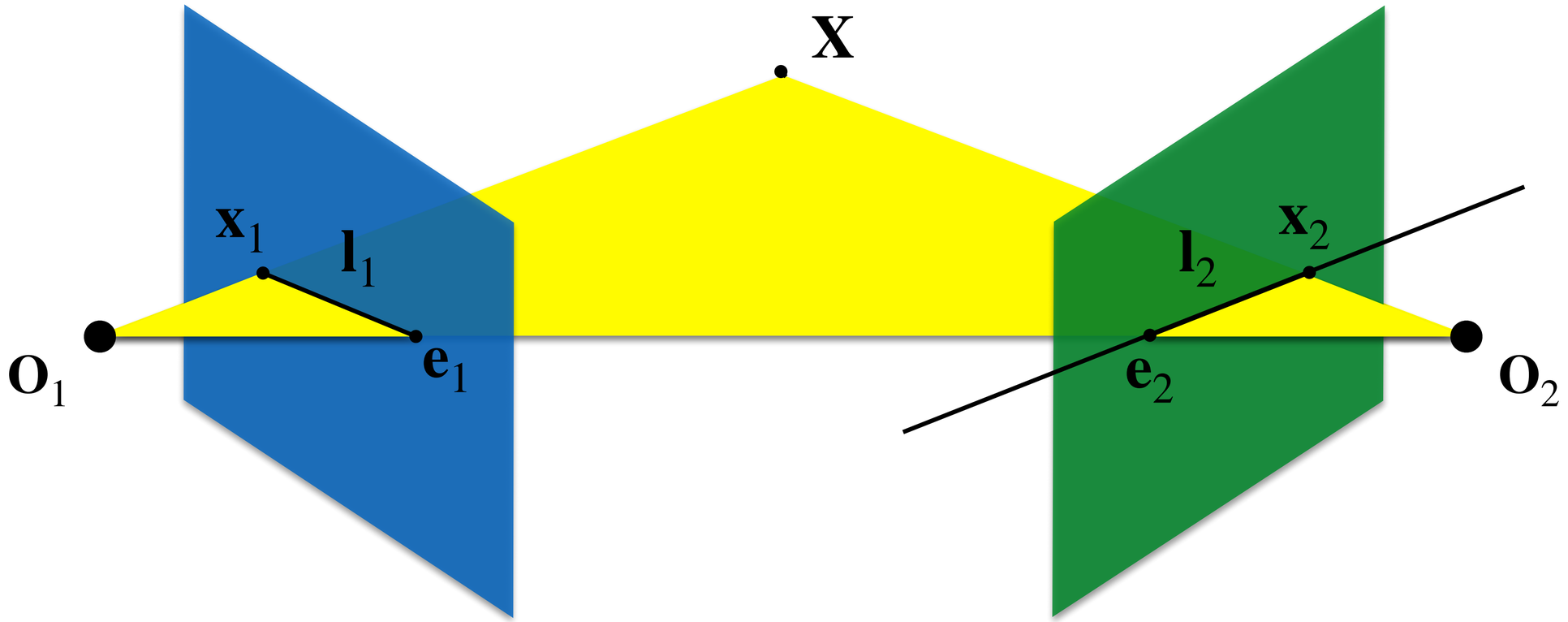
If the point $\mathbf{x}$ is on the epipolar line $\mathbf{l}$ then

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

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)

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

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.

For a different derivation, see also
https://visionbook.mit.edu/3d_scene_understanding_stereo.html

Definition of
Fundamental Matrix

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

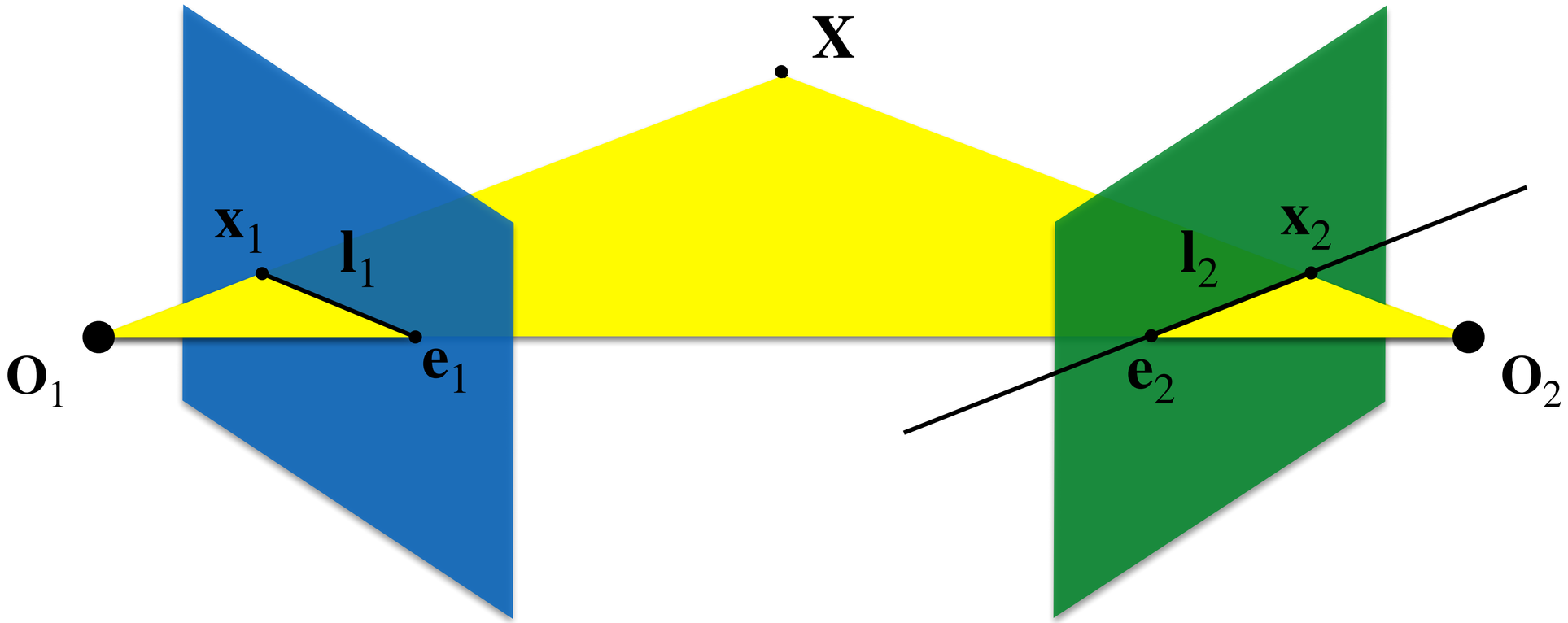
Point $\tilde{\mathbf{x}}_2$ lies on epipolar line $\mathbf{l}_2$

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

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

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

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

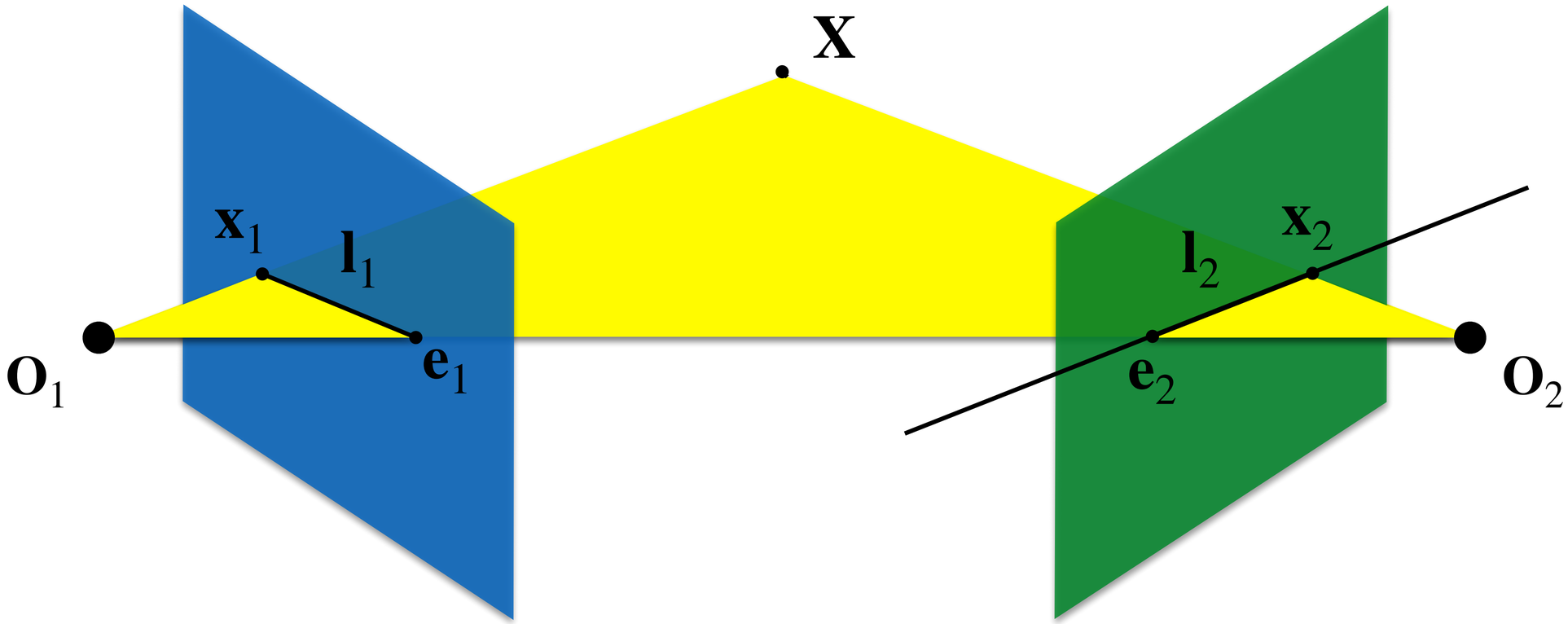


Adapted from: CMU 16-385 (Yannis, Kris)

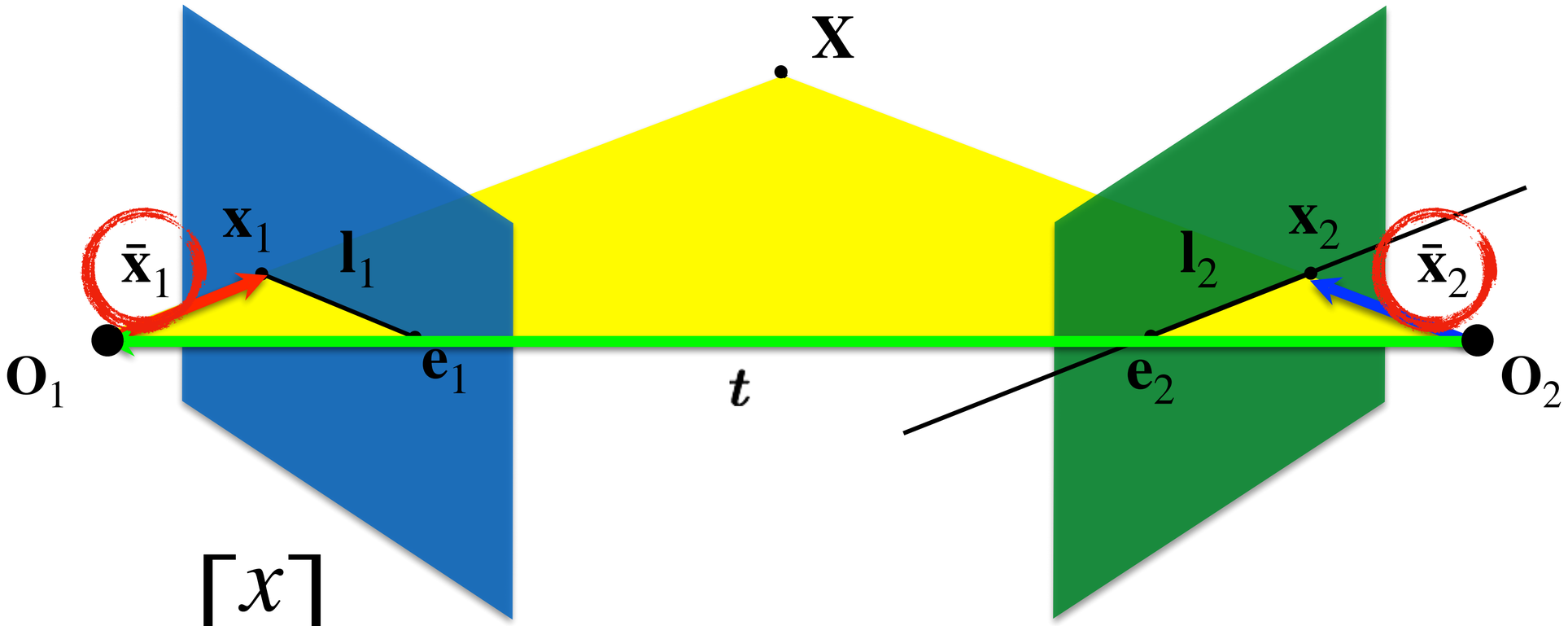
We'll now work off of this constraint to derive an analytical formula for $\mathbf{F}$.

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

Note the symmetry (bilinear form)



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



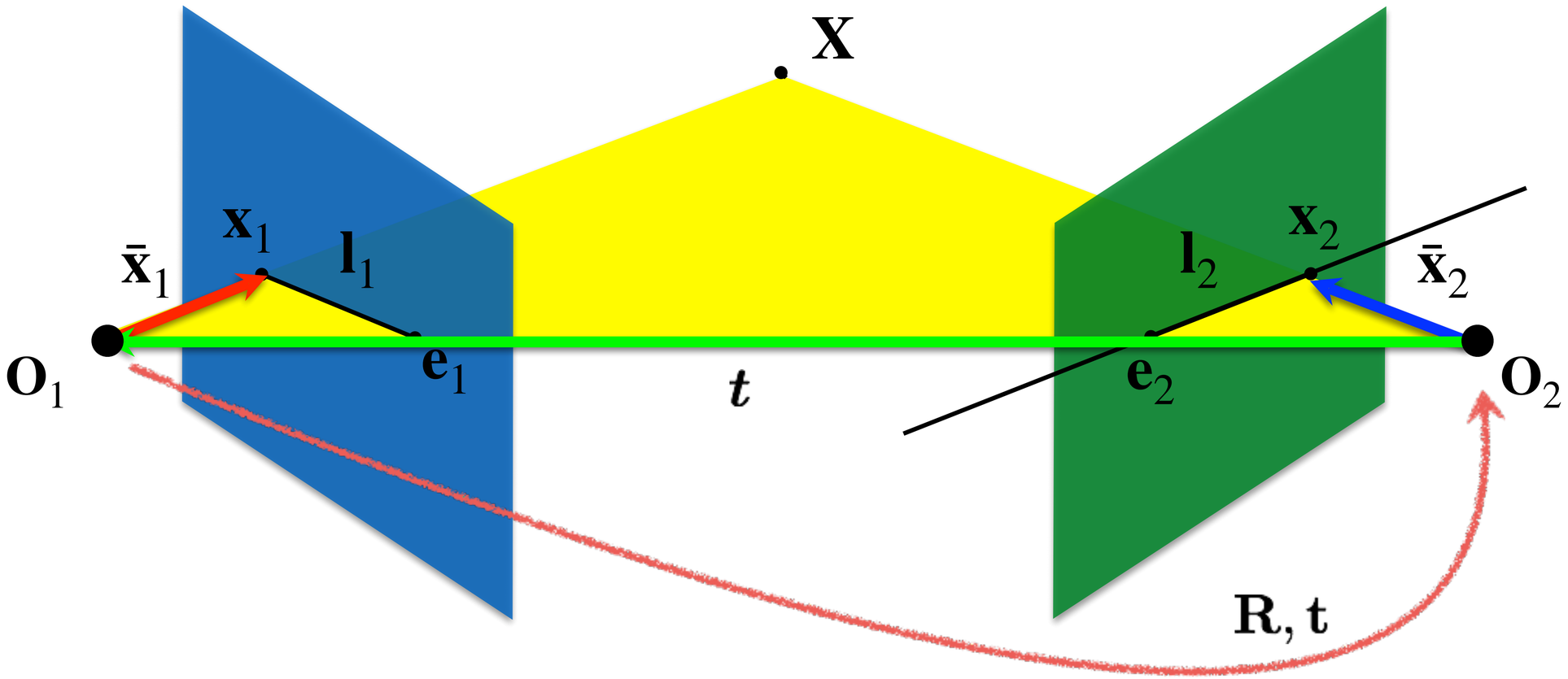
$$\tilde{\mathbf{x}} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Homogenized pixel coordinate

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

The local ray direction!

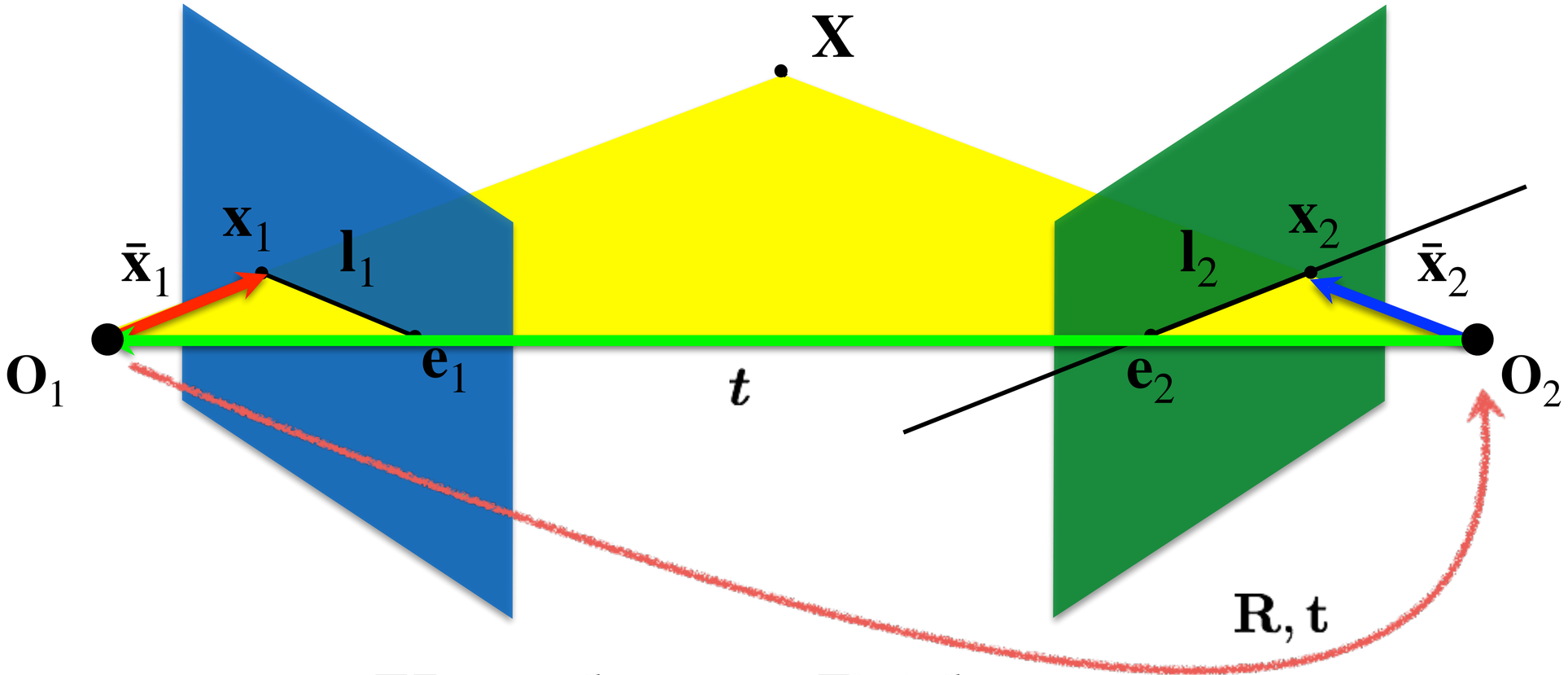
Adapted from: CMU 16-385 (Yannis, Kris)



$$\mathbf{X} = d_2\bar{\mathbf{x}}_2 = \mathbf{R}(d_1\bar{\mathbf{x}}_1 - \mathbf{t})$$

Where d_i is the depth (scalar)

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

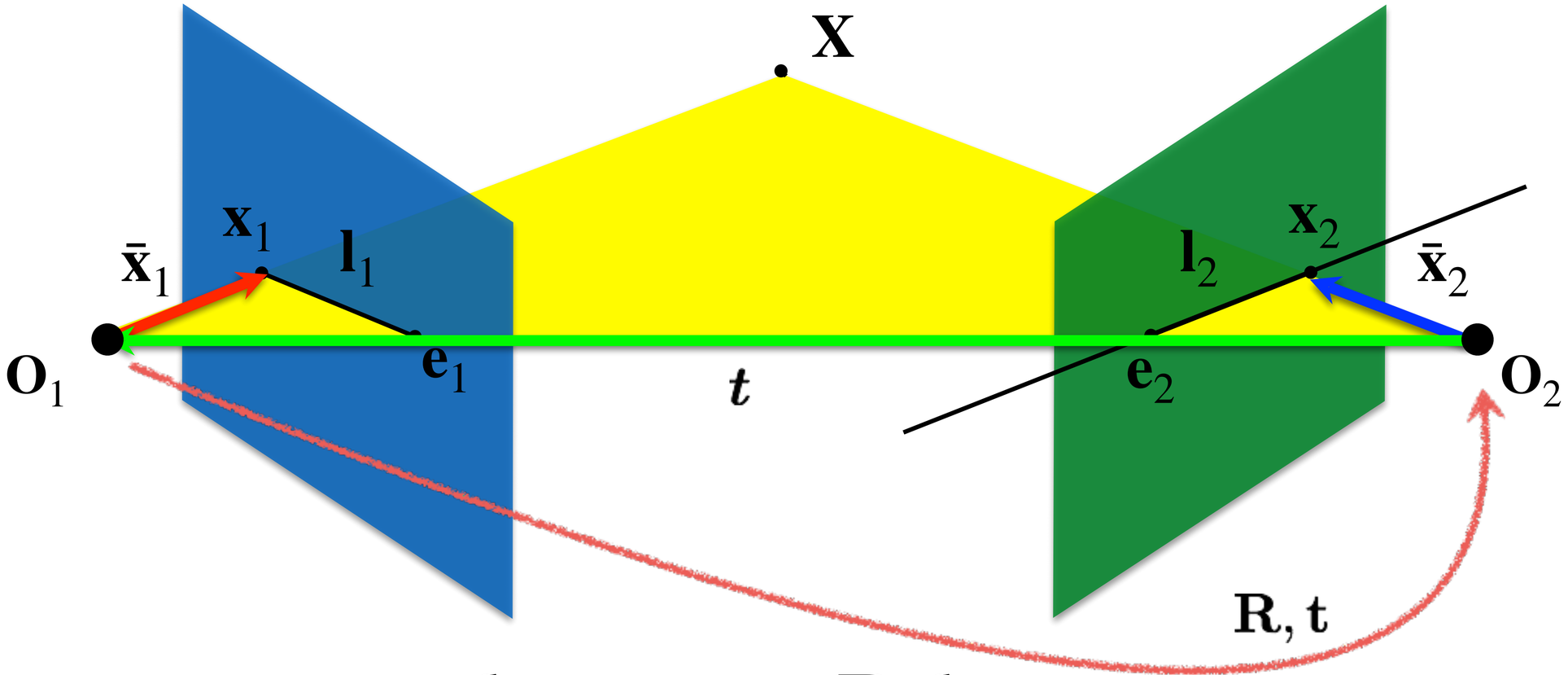


$$\mathbf{X} = d_2\bar{\mathbf{x}}_2 = \mathbf{R}(d_1\bar{\mathbf{x}}_1 - \mathbf{t})$$

Cross product with t

$$t \times d_2\bar{\mathbf{x}}_2 = t \times \mathbf{R}(d_1\bar{\mathbf{x}}_1 - \mathbf{t})$$

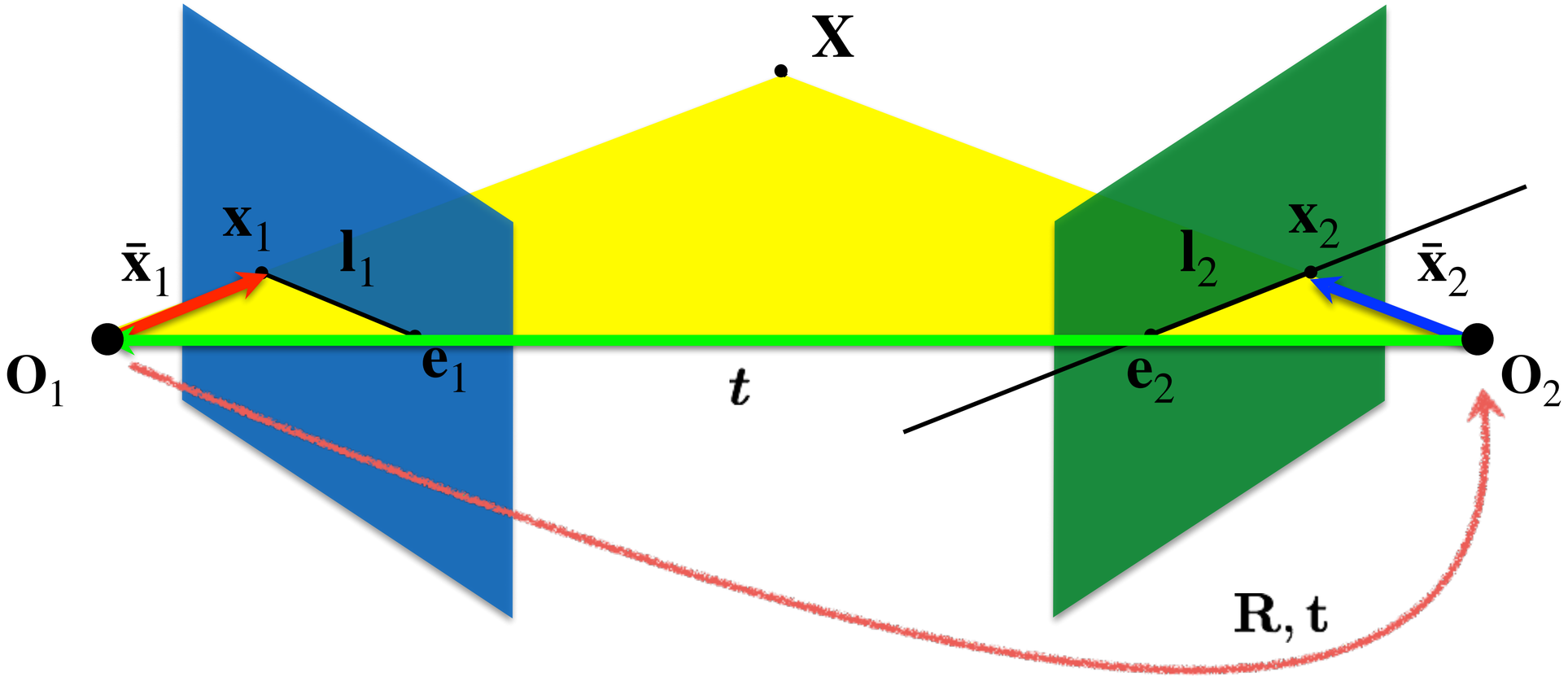
$$t \times d_2\bar{\mathbf{x}}_2 = t \times \mathbf{R}d_1\bar{\mathbf{x}}_1$$



$$t \times d_2 \bar{\mathbf{x}}_2 = t \times \mathbf{R} d_1 \bar{\mathbf{x}}_1$$

Dot product with x2

$$\bar{\mathbf{x}}_2^T \quad t \times d_2 \bar{\mathbf{x}}_2 = \bar{\mathbf{x}}_2^T \quad t \times \mathbf{R} d_1 \bar{\mathbf{x}}_1$$



$$\bar{\mathbf{x}}_2^T \quad t \times d_2 \bar{\mathbf{x}}_2 = \bar{\mathbf{x}}_2^T \quad t \times \mathbf{R} d_1 \bar{\mathbf{x}}_1$$

Must be zero, plus divide everything by d1

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

Turning cross products into matrices

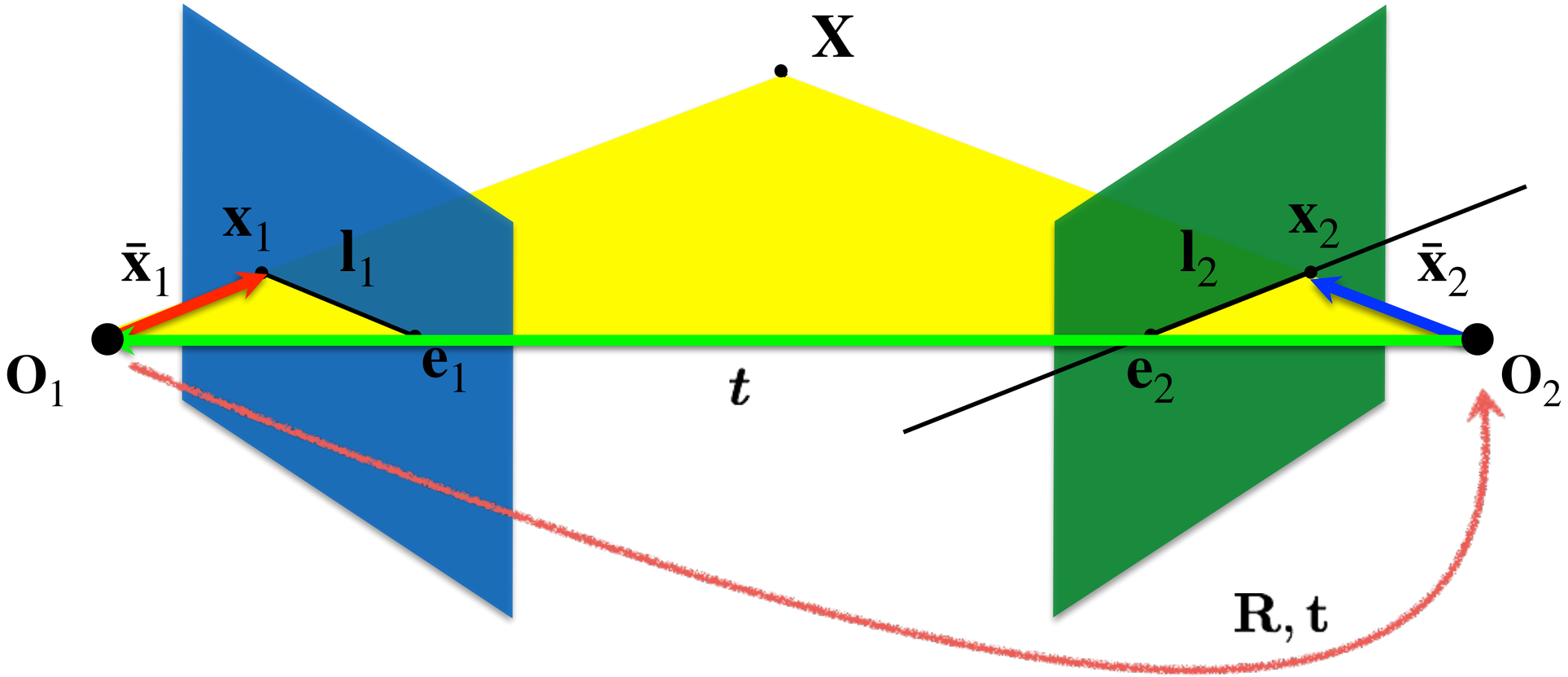
$$\boldsymbol{a} \times \boldsymbol{b} = \begin{bmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{bmatrix}$$

$$\boldsymbol{a} \times \boldsymbol{b} = [\boldsymbol{a}]_\times \boldsymbol{b} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Skew symmetric $[a]_\times = -[a]_\times^T$

$[a]_\times$ *means matrix form of cross product*

https://www.cs.cmu.edu/~16385/s15/lectures/Lecture18.pdf

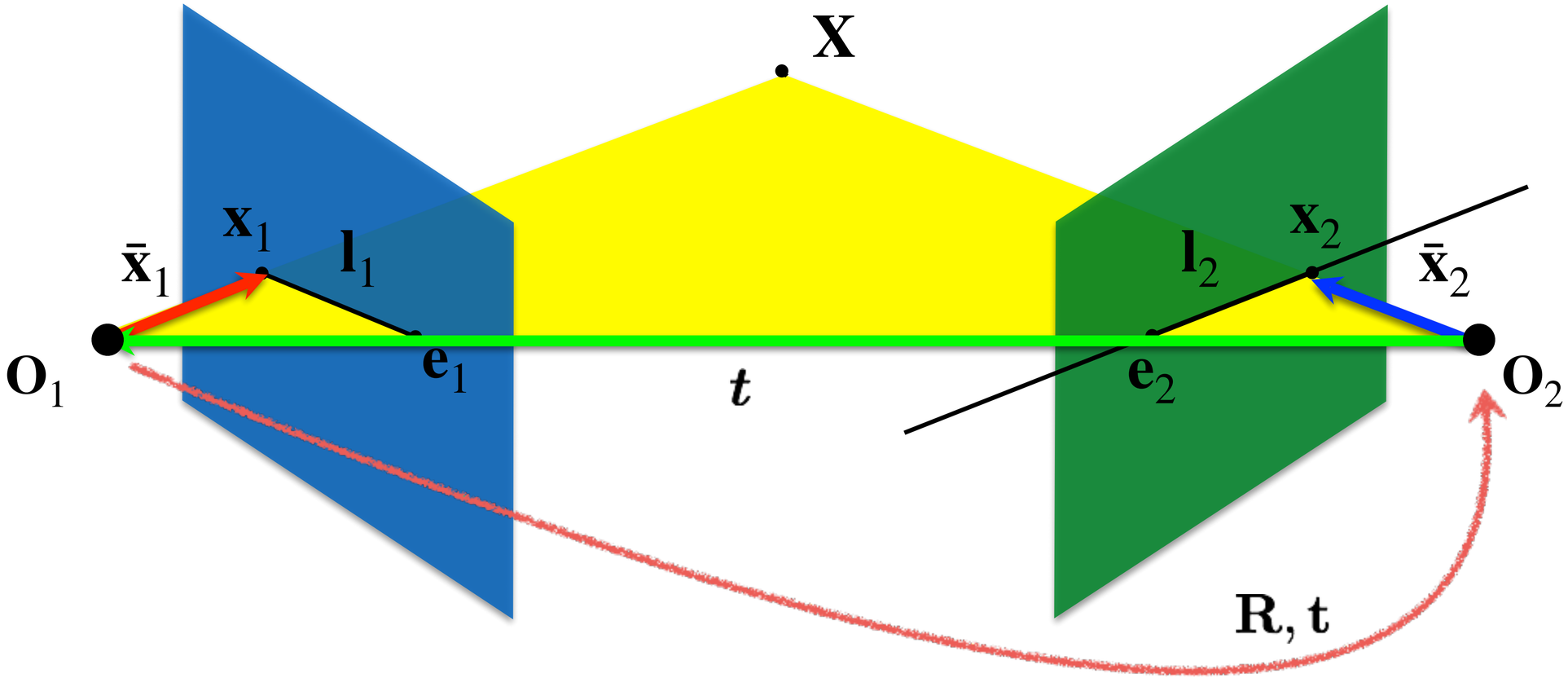


$$\bar{\mathbf{x}}_2^T \quad t \times d_2 \bar{\mathbf{x}}_2 = \bar{\mathbf{x}}_2^T \quad t \times \mathbf{R} d_1 \bar{\mathbf{x}}_1$$

Must be zero, plus divide everything by d1

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

Essential matrix



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

Essential matrix

Going back to image coordinates with $\bar{\mathbf{x}} = \mathbf{K}^{-1}\tilde{\mathbf{x}}$

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

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

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

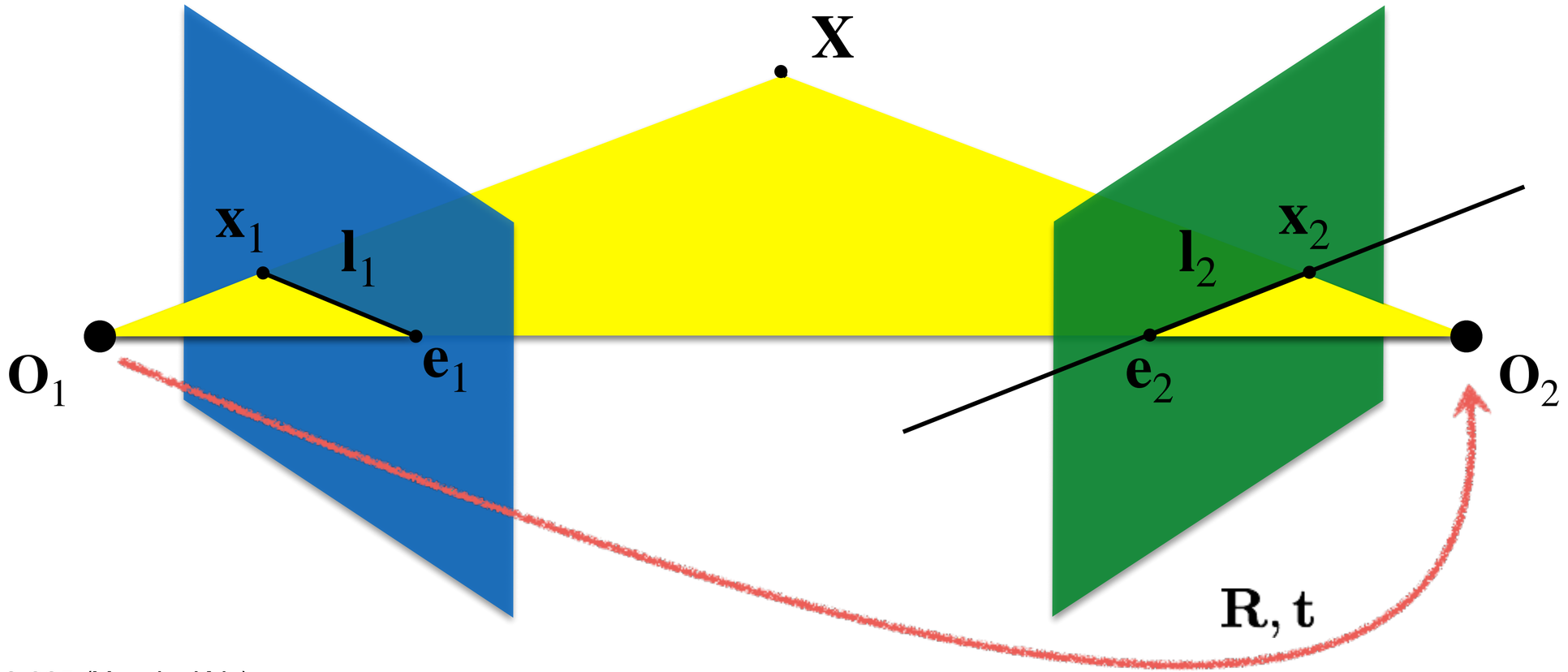


Figure credit: CMU 16-385 (Yannis, Kris)

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

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

Consistent with our former line equation:

$$\mathbf{x}_2 \sim \mathbf{K}_2[\mathbf{R} \mid \mathbf{t}]\begin{pmatrix}\lambda\mathbf{K}_1^{-1}\mathbf{x}_1 \\ 1\end{pmatrix}$$

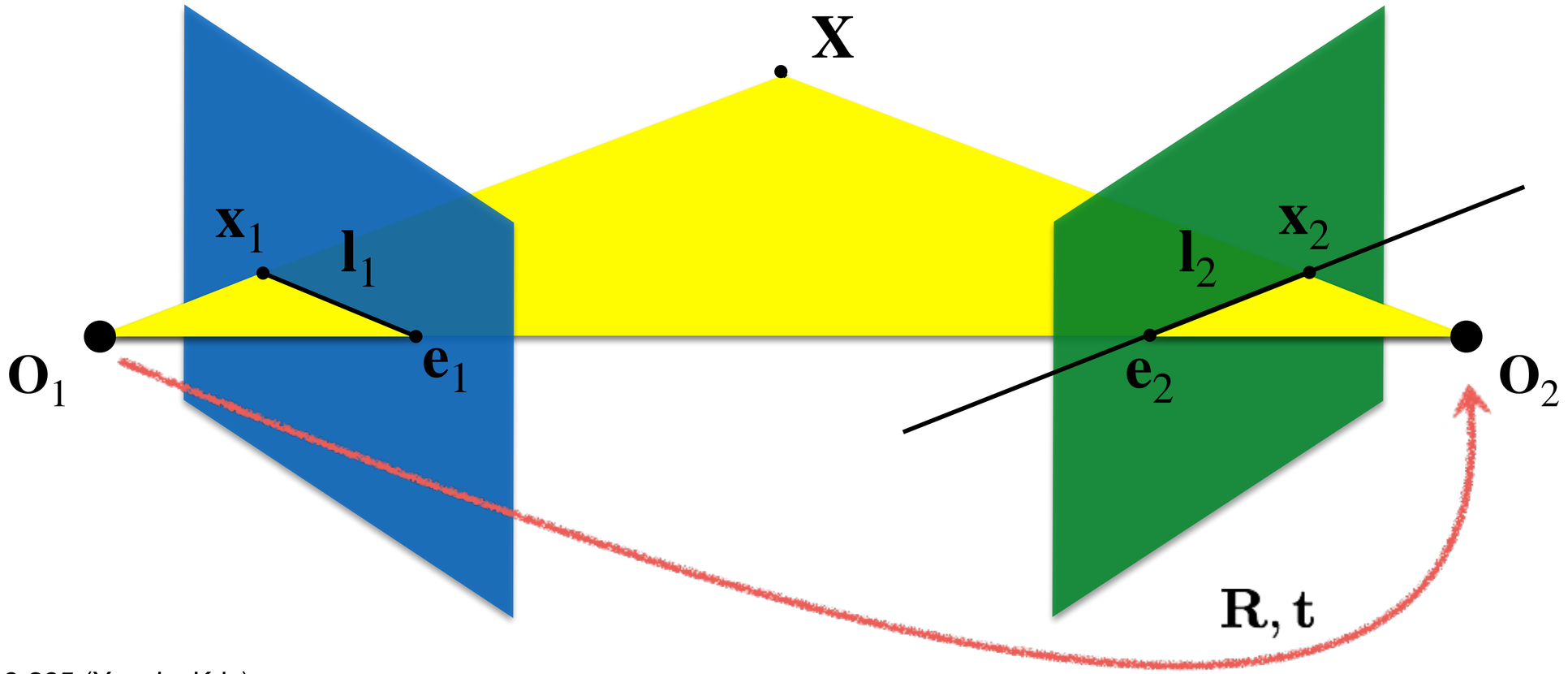


Figure credit: CMU 16-385 (Yannis, Kris)

Recap

- Epipolar Geometry/lines can be expressed as

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

Or

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

- Where F is called the fundamental matrix

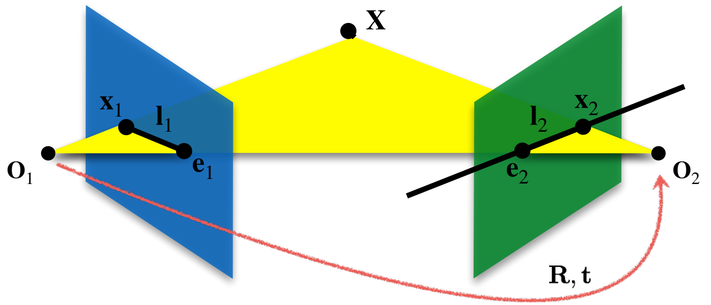
$$\mathbf{F} = \mathbf{K}_2^{-T}(\mathbf{R}[\mathbf{t}]_\times)\mathbf{K}_1^{-1}$$

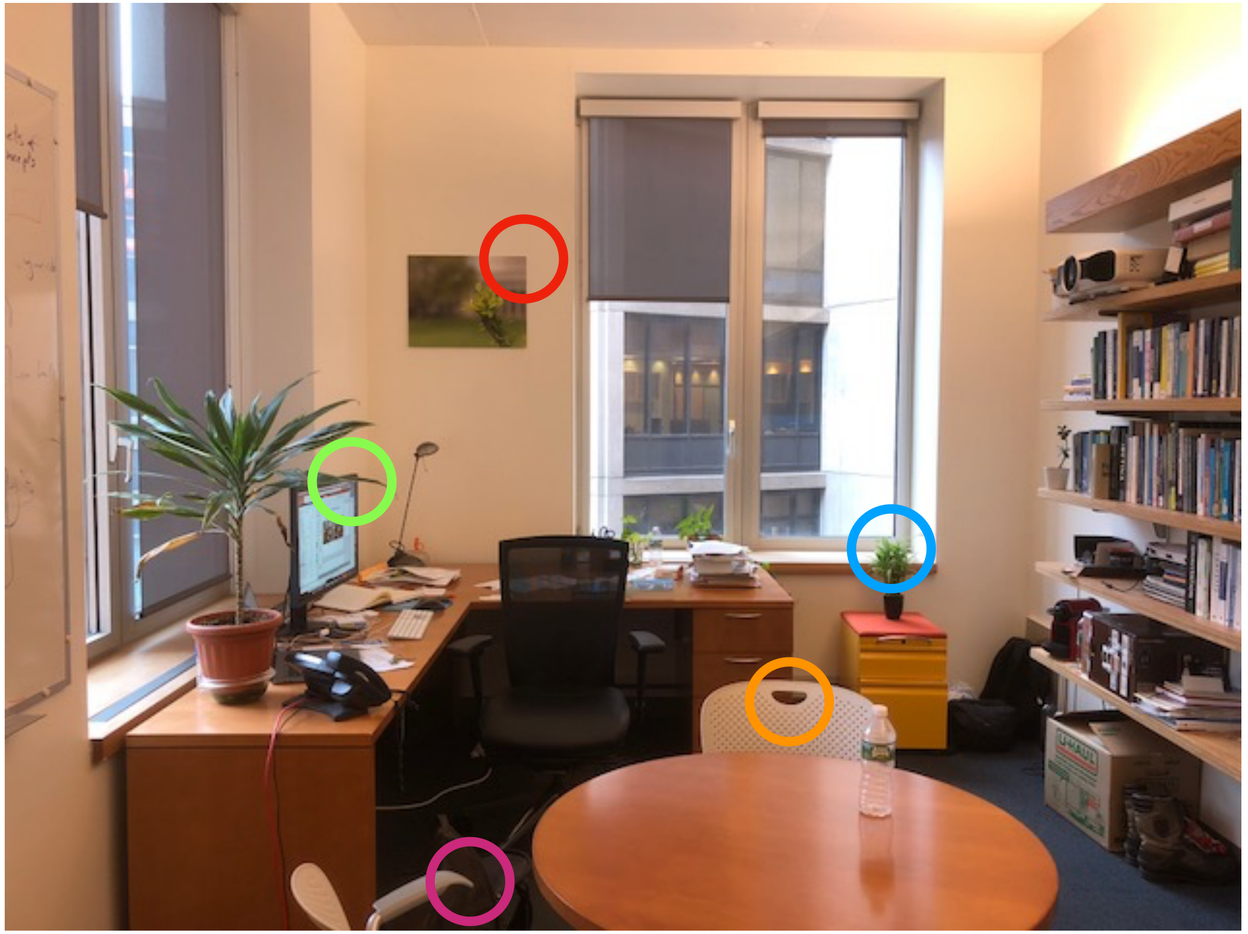
- If you express things in normalized camera coordinates, you remove the K matrices and it's called the essential matrix

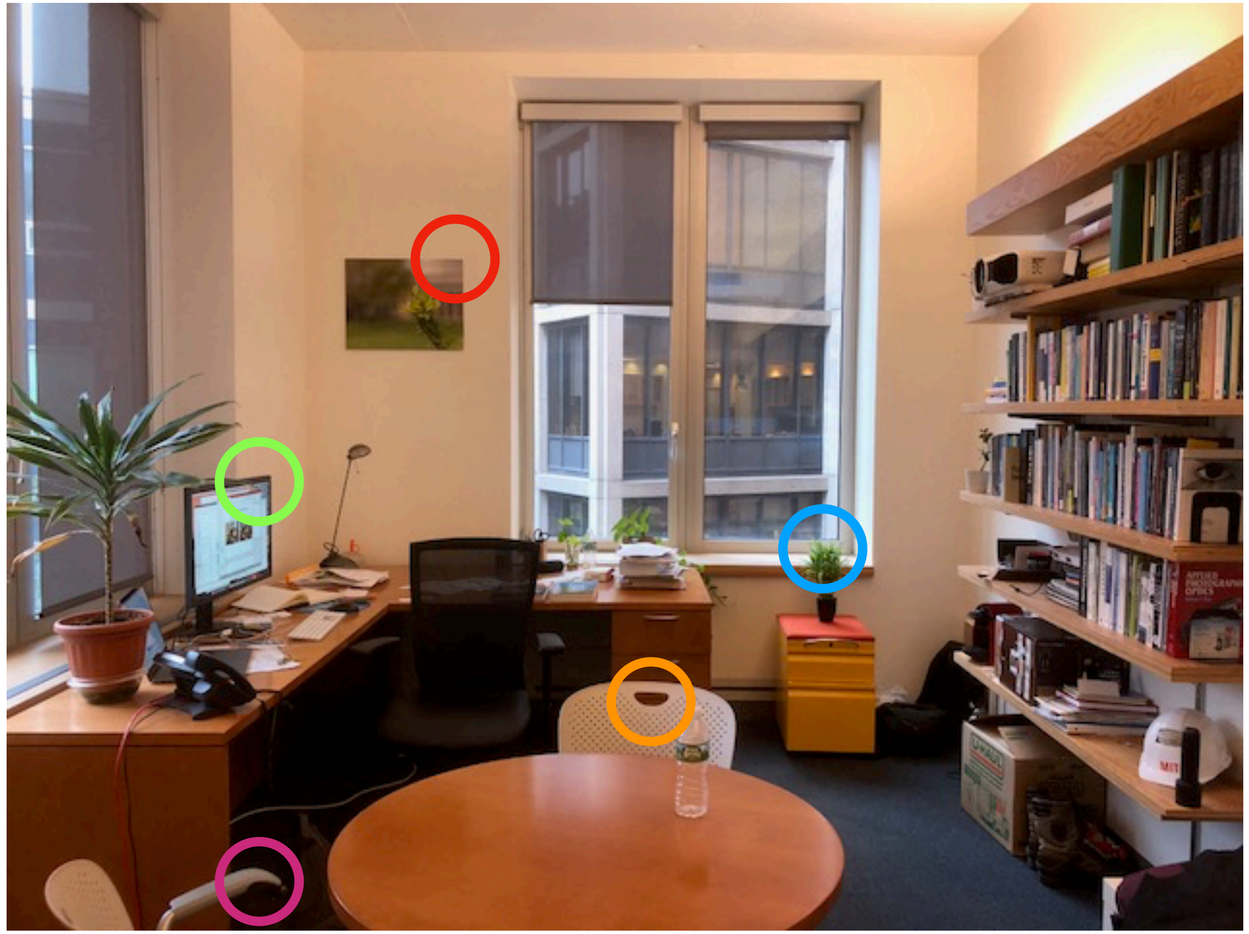
$$\mathbf{E} = \mathbf{R}[\mathbf{t}]_\times$$

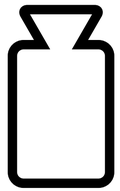
$$\mathbf{F} = \mathbf{K}_2^{-T}\mathbf{E}\mathbf{K}_1^{-1}$$

- For calibration, F is unknown and we seek to find its coefficients from know x1s and x2s

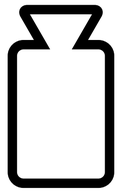








What if $\mathbf{P}_1, \mathbf{P}_2$ *aren't* known?



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

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

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

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.*

Recap

Epipolar geometry captured by fundamental matrix $\tilde{\mathbf{x}}_2^T \mathbf{F} \tilde{\mathbf{x}}_1 = 0$

Can be obtained using given correspondences and linear regression $\arg \min_{\|\mathbf{f}\|=1} \|\mathbf{W}\mathbf{f}\|_2^2$ (each correspondence gives one equation)

If intrinsic matrix known, done.

Otherwise factorize based on properties and assumptions about the intrinsic vs extrinsics.

Better if K is known because we can unwarp and the linear assumption holds better

Robustness

Challenge : not all our correspondences may be correct

  Even worse, it’s not noise, it can be arbitrarily far

Least squares helps but can still be arbitrarily bad

  One single bad correspondence can have an arbitrary impact

RANSAC

Challenge : not all our correspondences may be correct

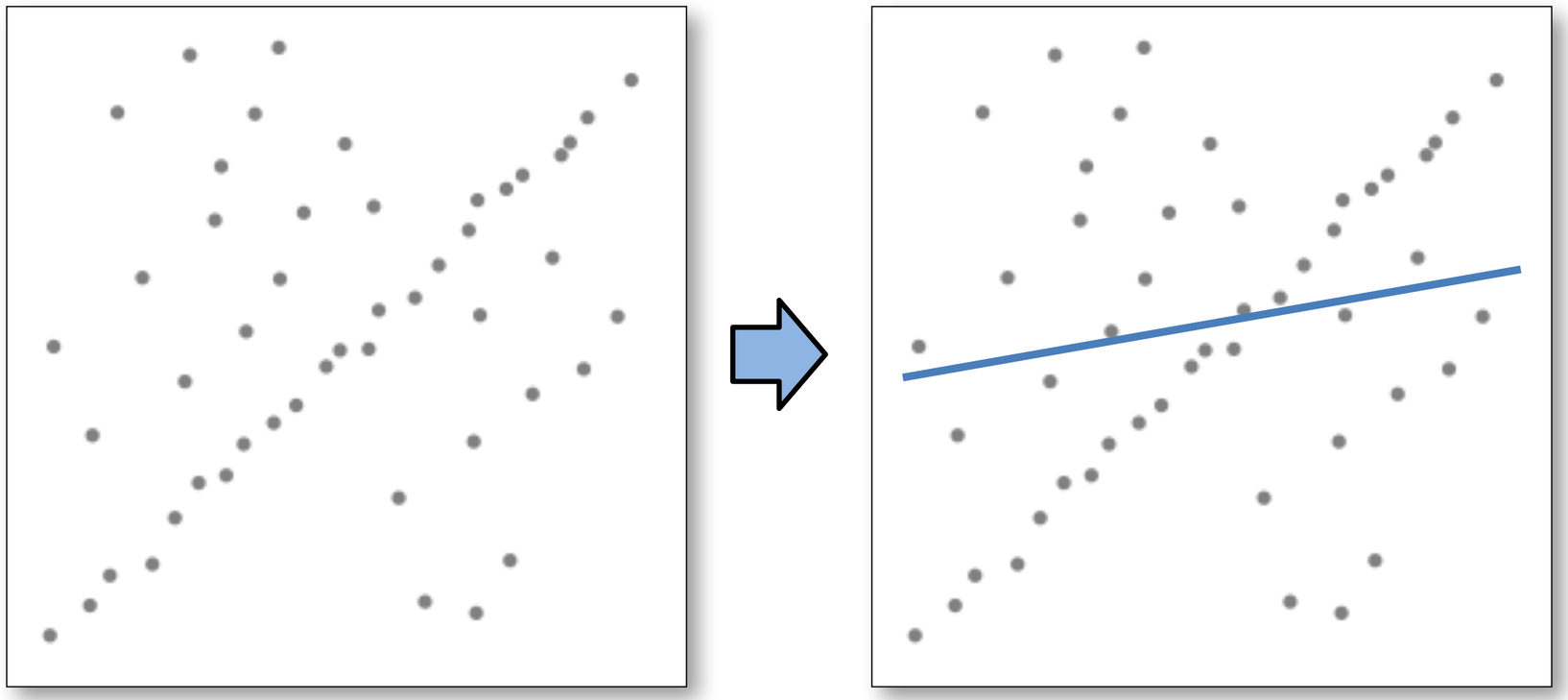
Least squares helps but can still be arbitrarily bad

Solution:

- Leverage the oversupply of correspondences (usually more than 8)
- Hope that there exists a subset that are all correct enough
- Randomly try subsets until we find one that “agrees” with the maximum number of other correspondences
- Can be seen as using L0 instead of L2

Robustness

- Let’s consider the problem of linear regression



Problem: Fit a line to these datapoints

Least squares fit

- How can we fix this?

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

We need a better cost function…

- Suggestions?

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

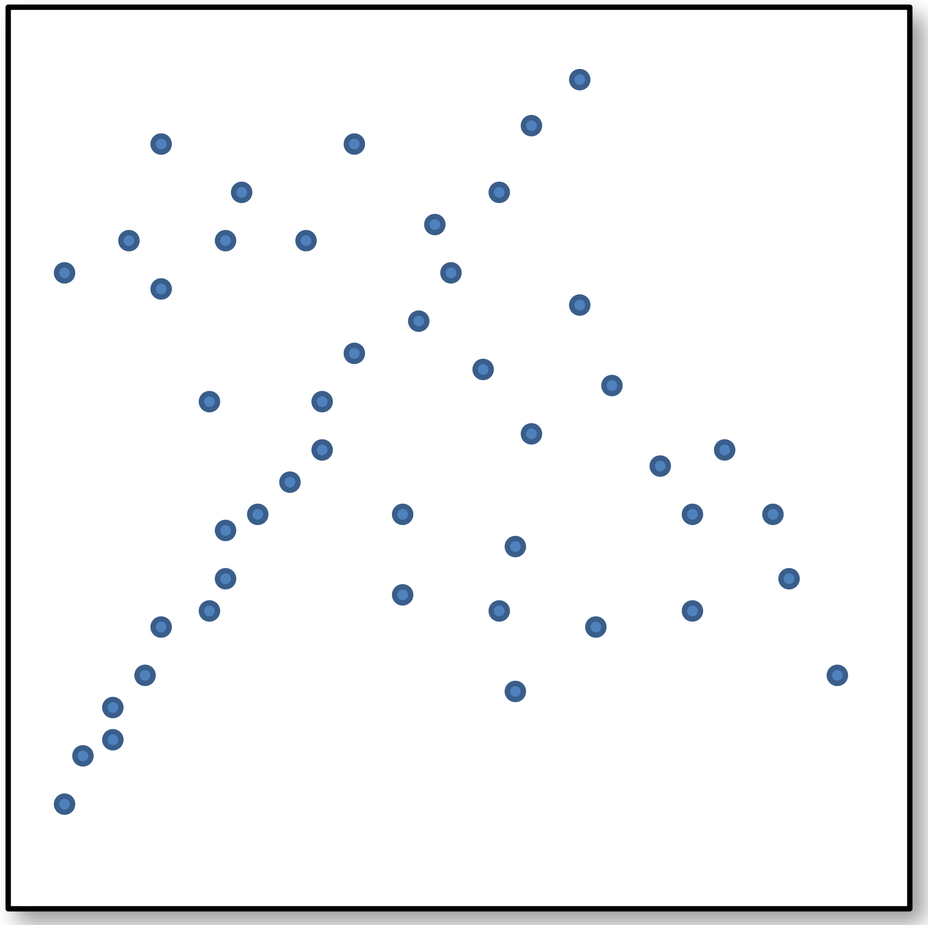
Idea

- Given a hypothesized line
- Count the number of points that “agree” with the line
  - “Agree” = within a small distance of the line
  - I.e., the **inliers** to that line

- For all possible lines, select the one with the largest number of inliers

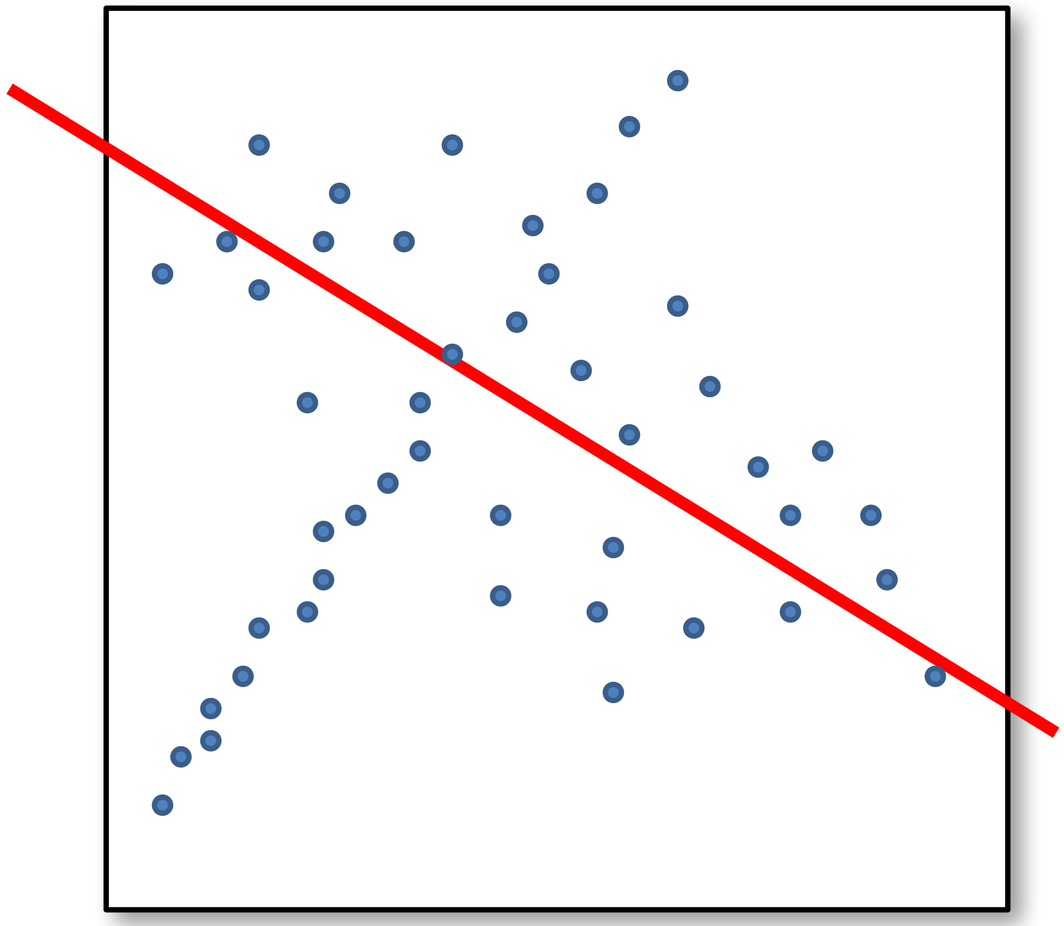
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Counting inliers



Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

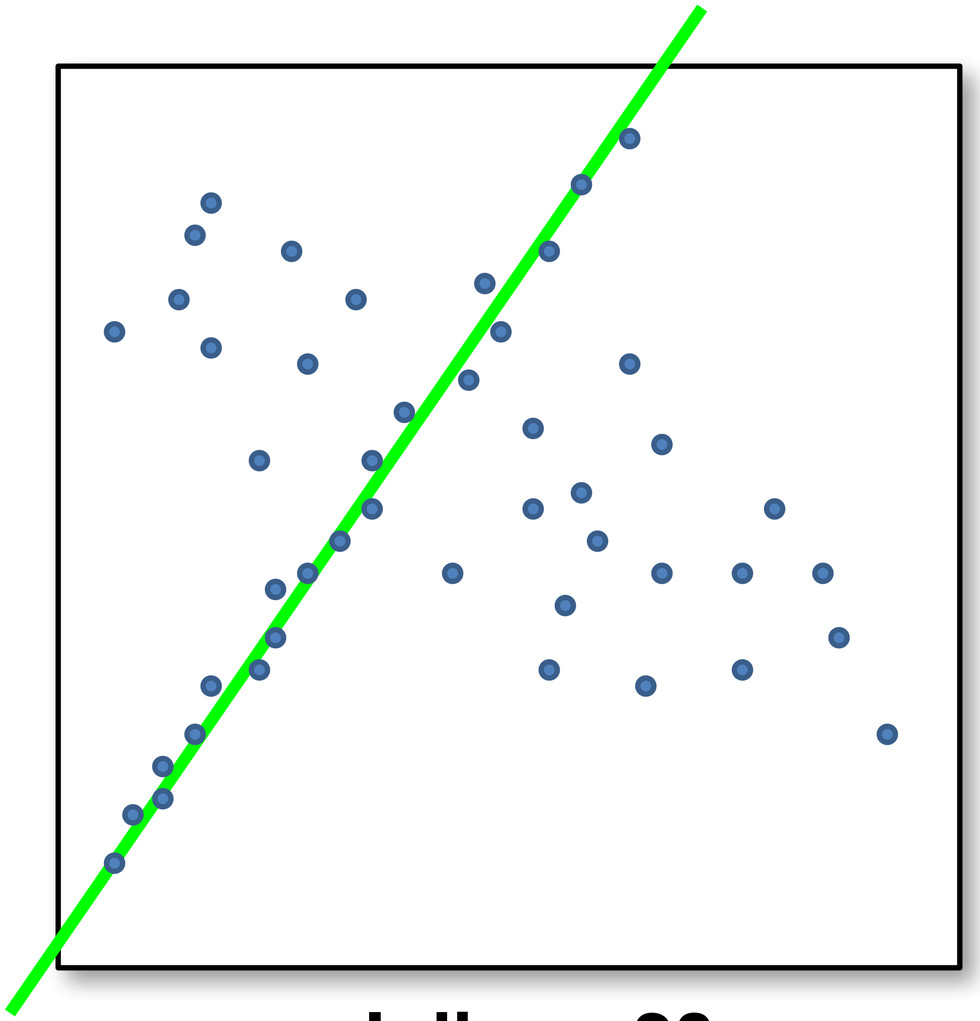
Counting inliers



Inliers: 3

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Counting inliers



Inliers: 20

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

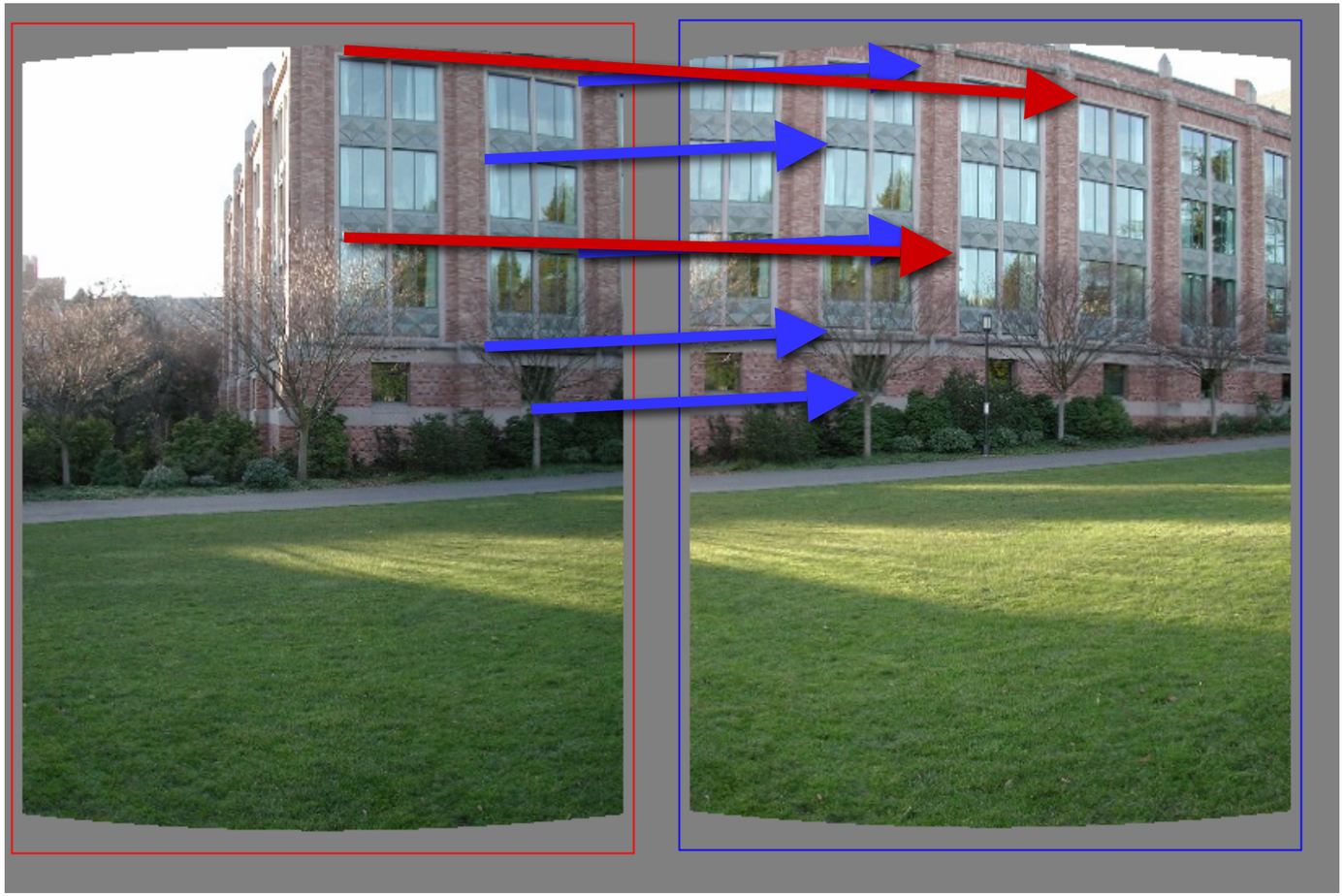
How do we find the best line?

- Unlike least-squares, no simple closed-form solution

- Hypothesize-and-test
  - Try out many lines, keep the best one
  - Which lines?

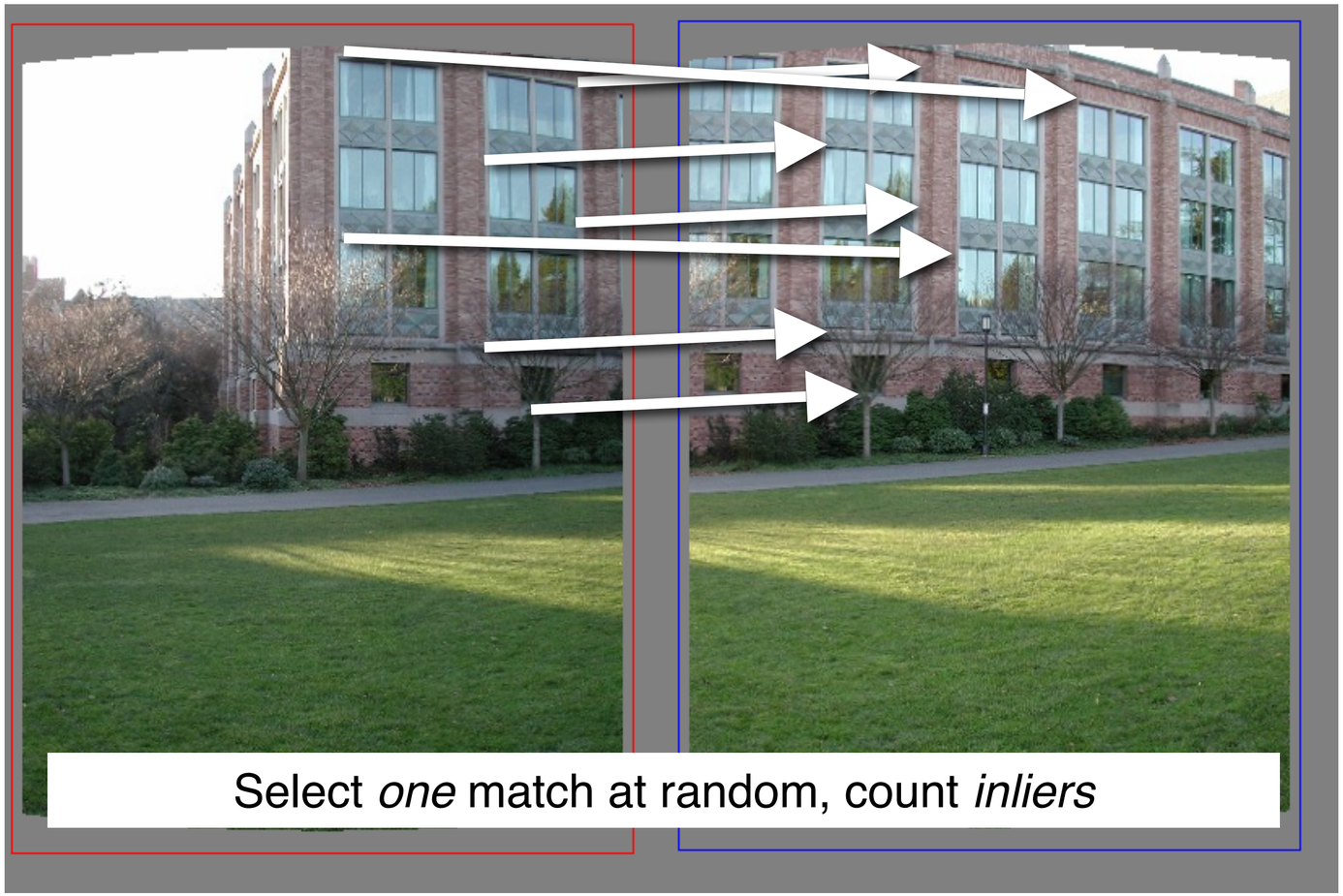
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Translations



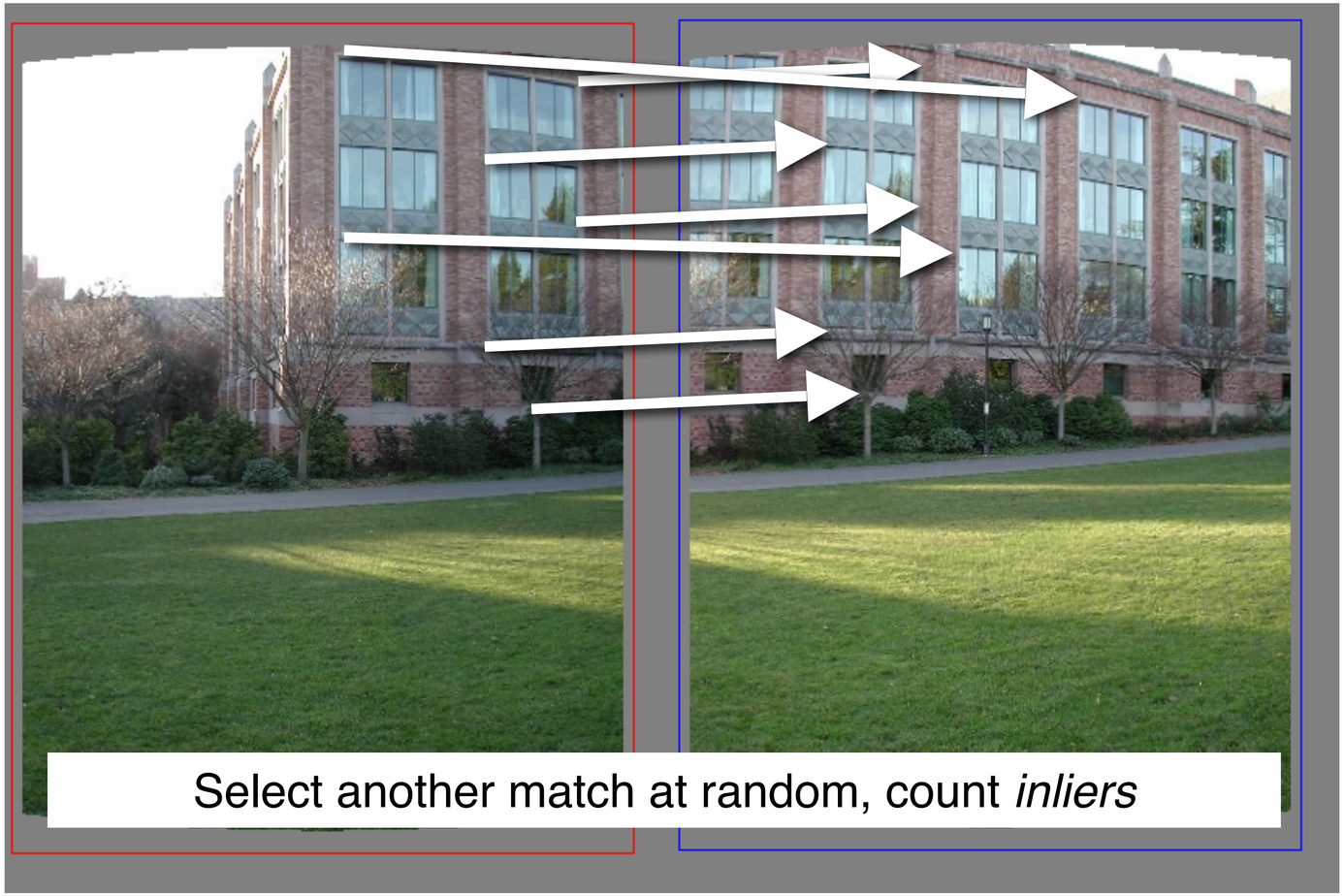
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RAndom SAmple Consensus



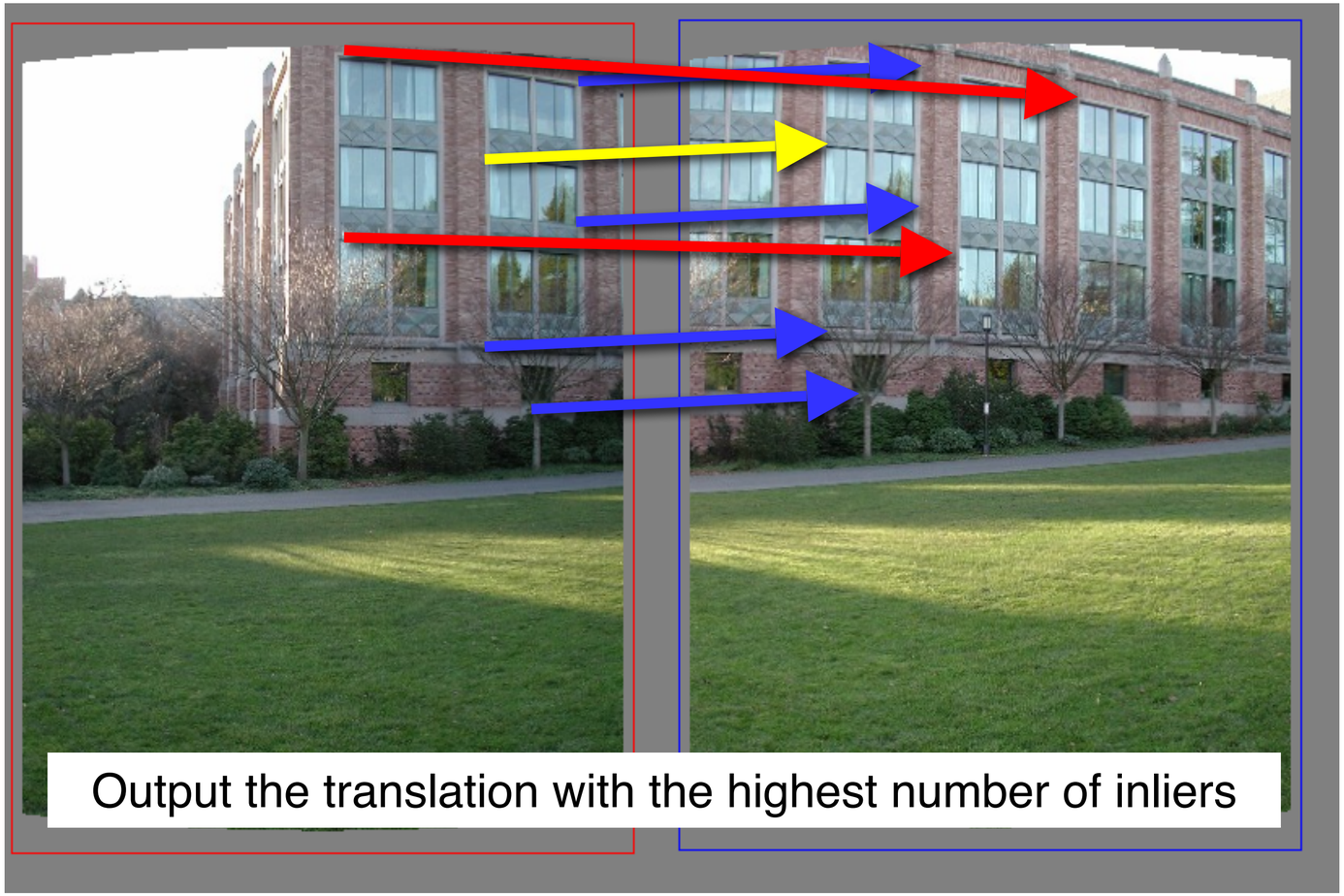
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RAndom SAmple Consensus



Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RAndom SAmple Consensus



Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RANSAC

- Idea:
  - All the inliers will agree with each other on the translation vector; the (hopefully small) number of outliers will (hopefully) disagree with each other
    - RANSAC only has guarantees if there are < 50% outliers
  - “All good matches are alike; every bad match is bad in its own way.”

    – Tolstoy via Alyosha Efros

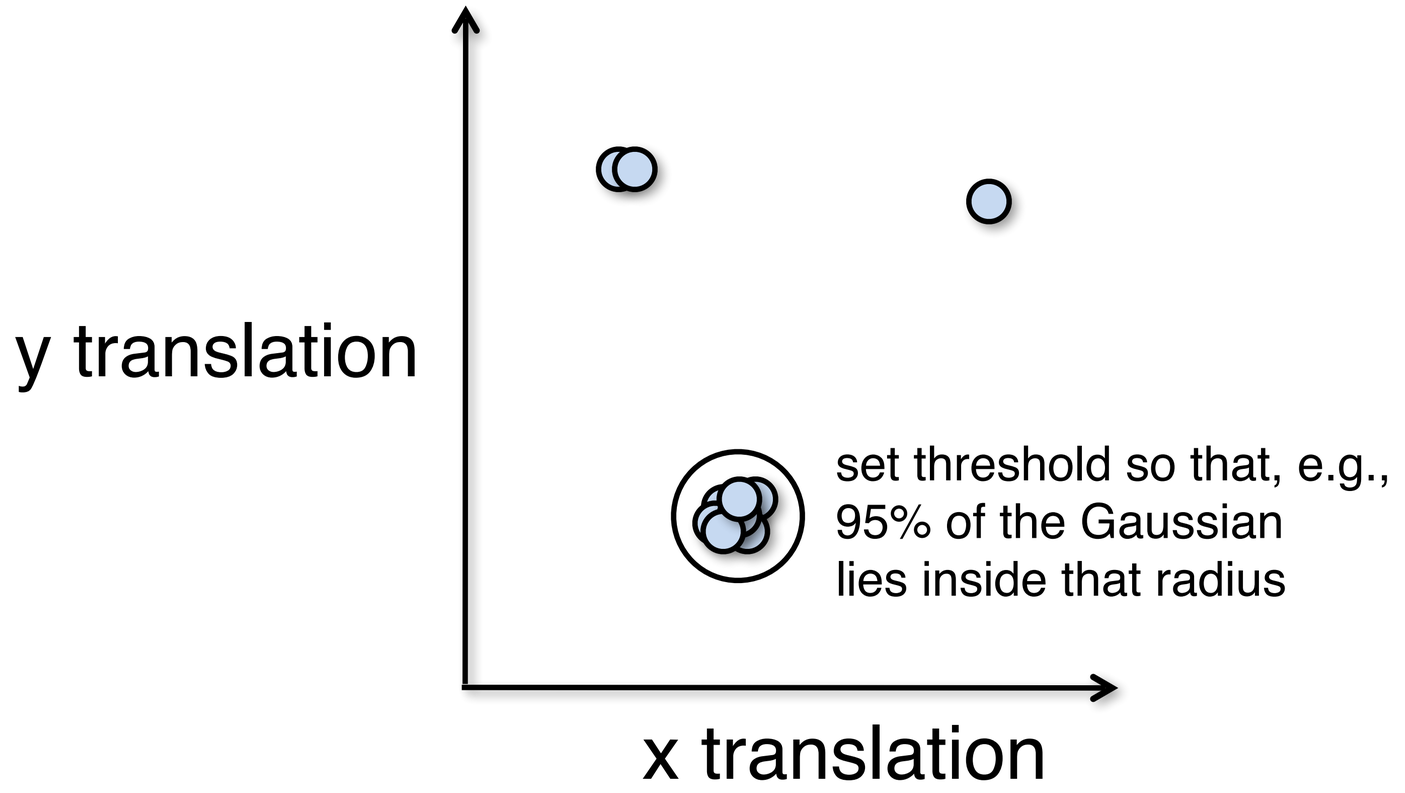
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RANSAC

- **Inlier threshold** related to the amount of noise we expect in inliers
  - Often model noise as Gaussian w/ some standard deviation (e.g. 3 pixels)
- **Number of rounds** related to the percentage of outliers we expect, and the probability of success we'd like to guarantee
  - Suppose there are 20% outliers, and we want to find the correct answer with at least 99% probability
  - How many rounds do we need?

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RANSAC: Another view

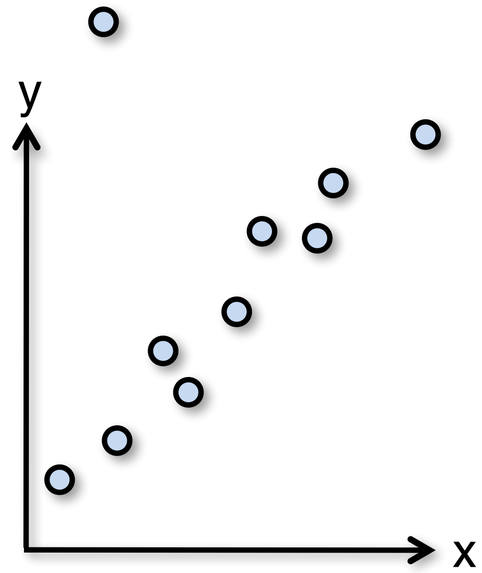


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RANSAC



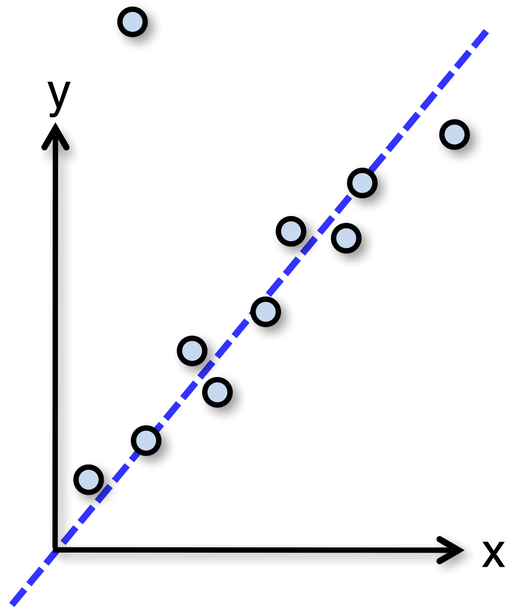
- Back to linear regression
- How do we generate a hypothesis?



Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RANSAC

- Back to linear regression
- How do we generate a hypothesis?



Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RANSAC General version:

1. Randomly choose *s* samples
   - Typically *s* = minimum sample size that lets you fit a model
2. Fit a model (e.g., line) to those samples
3. Count the number of inliers that approximately fit the model
4. Repeat *N* times
5. Choose the model that has the largest set of inliers

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RANSAC for fundamental matrix

1. Randomly choose *8* correspondences
2. Fit a fundamental matrix F to those samples (homogenous linear equation, SVD)
3. Count the number of inliers that approximately fit the model (i.e. where $\tilde{\mathbf{x}}_2^T \mathbf{F} \tilde{\mathbf{x}}_1 = 0$)
4. Repeat *N* times
5. Choose the model that has the largest set of inliers

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

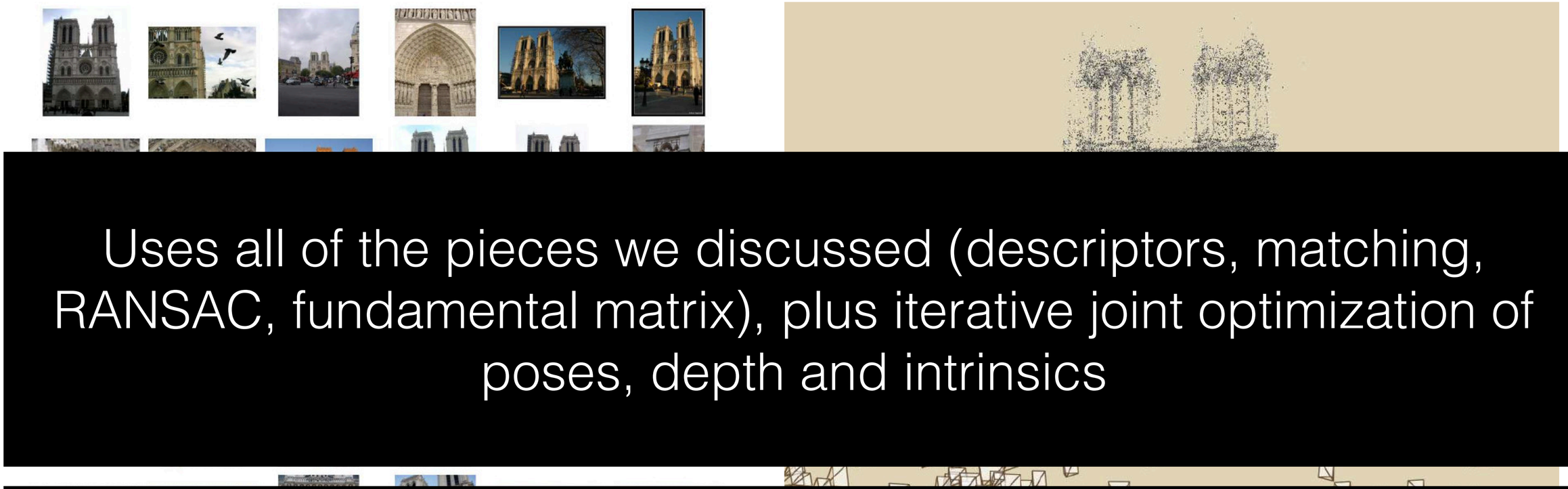
Questions?

1. Randomly choose *8* correspondences
2. Fit a fundamental matrix F to those samples (homogenous linear equation, SVD)
3. Count the number of inliers that approximately fit the model (i.e. where $\tilde{\mathbf{x}}_2^T \mathbf{F} \tilde{\mathbf{x}}_1 = 0$)
4. Repeat *N* times
5. Choose the model that has the largest set of inliers

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

What if we have **many** views?

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

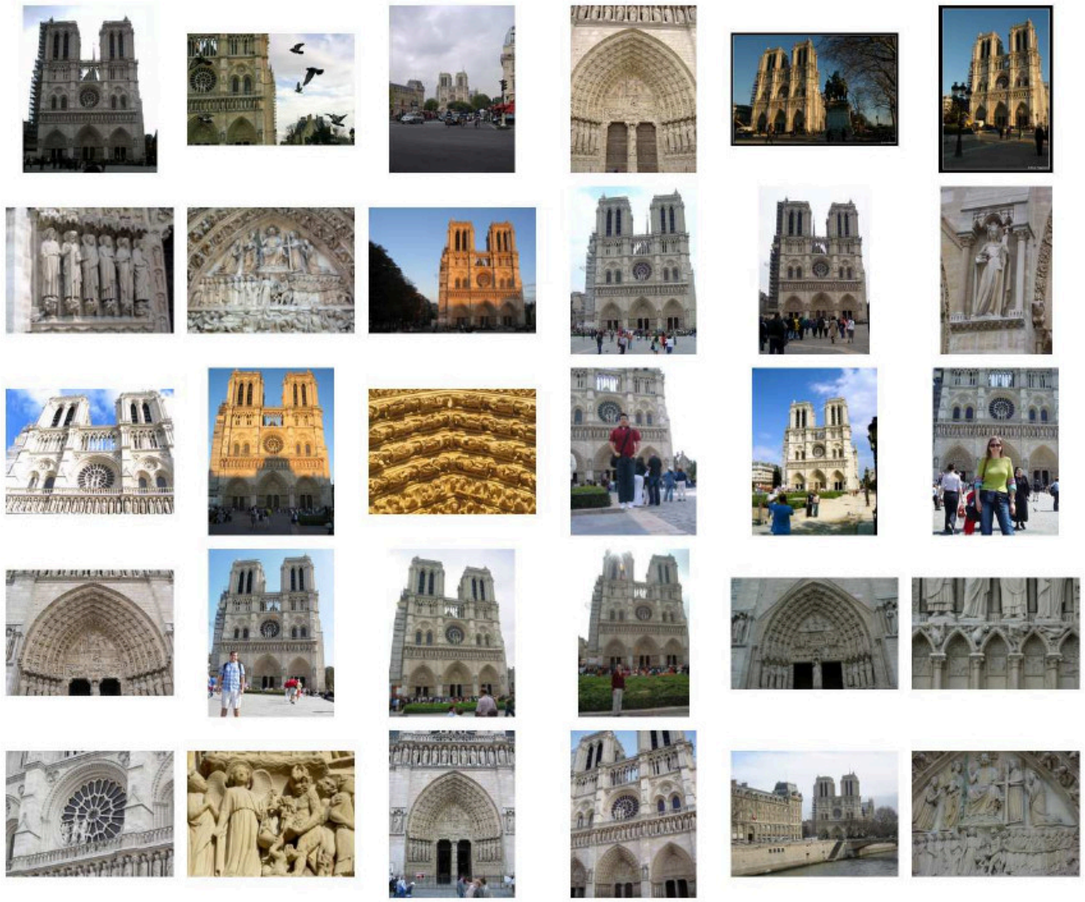


- COLMAP features a second **multi-view stereo stage** to obtain dense geometry

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



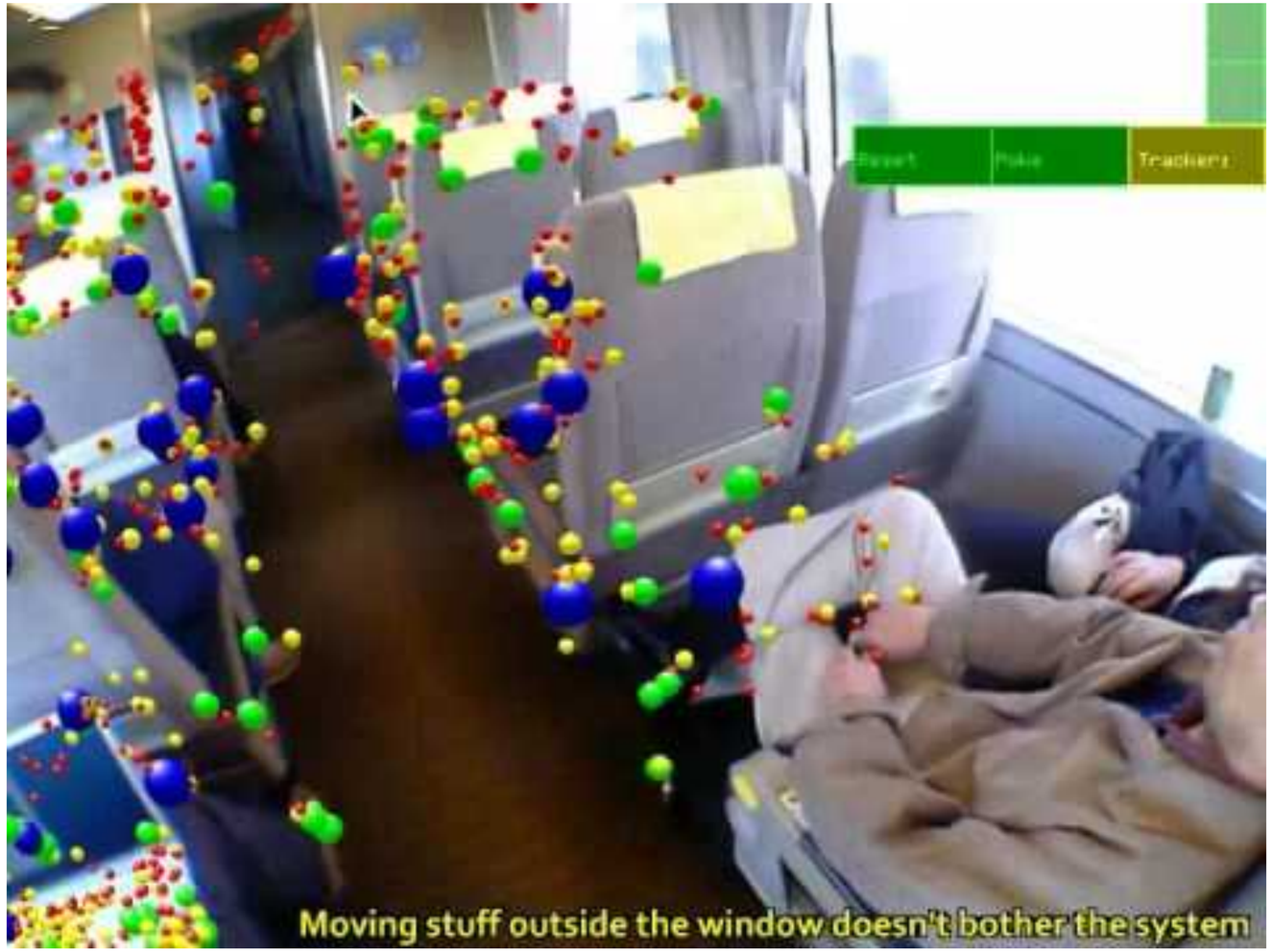


- **Photo Tourism / PhotoSynth** allows for exploring photo collections in 3D

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)

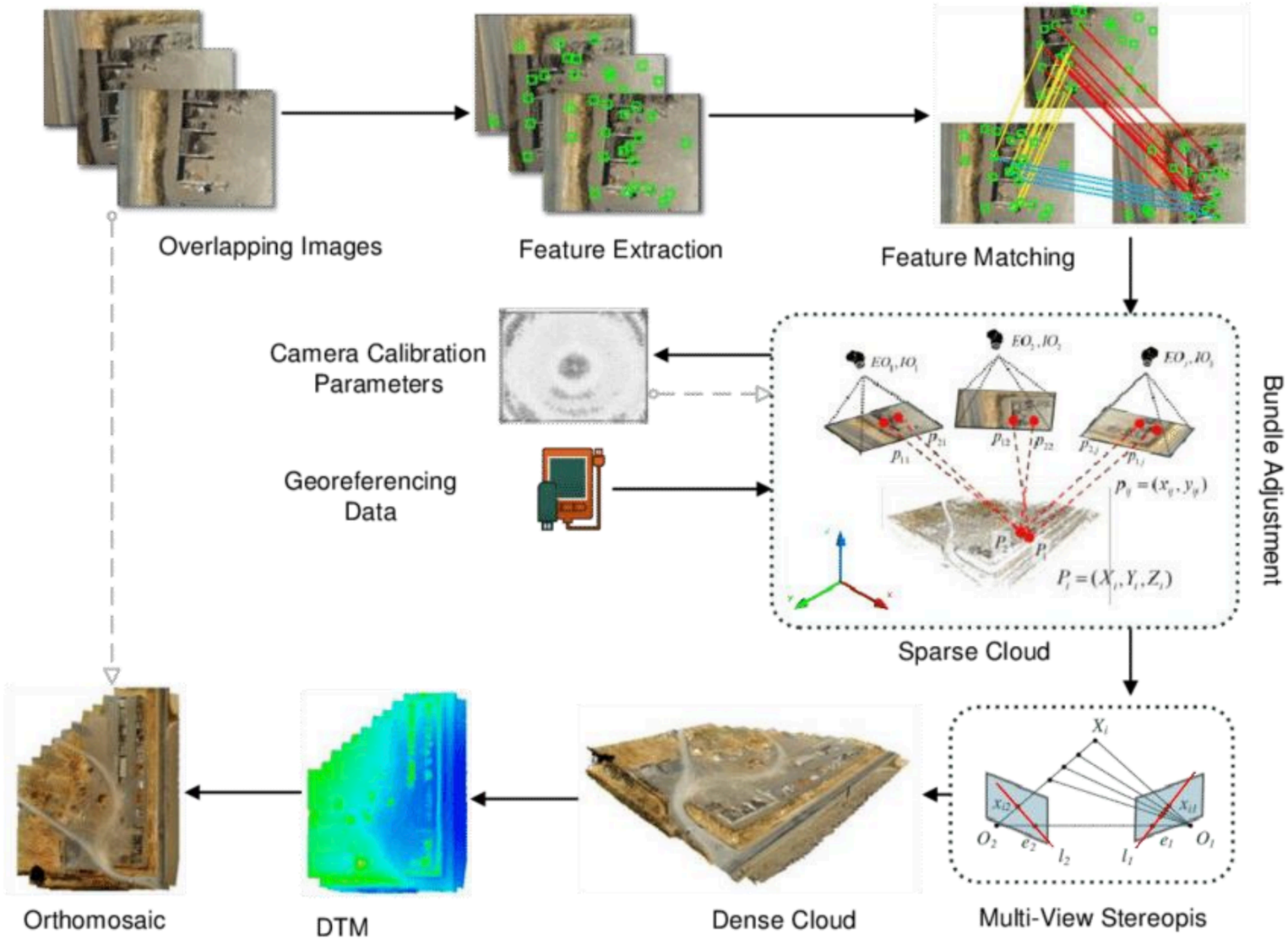


- **PTAM** demonstrates real-time tracking and mapping of small workspaces

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

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

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

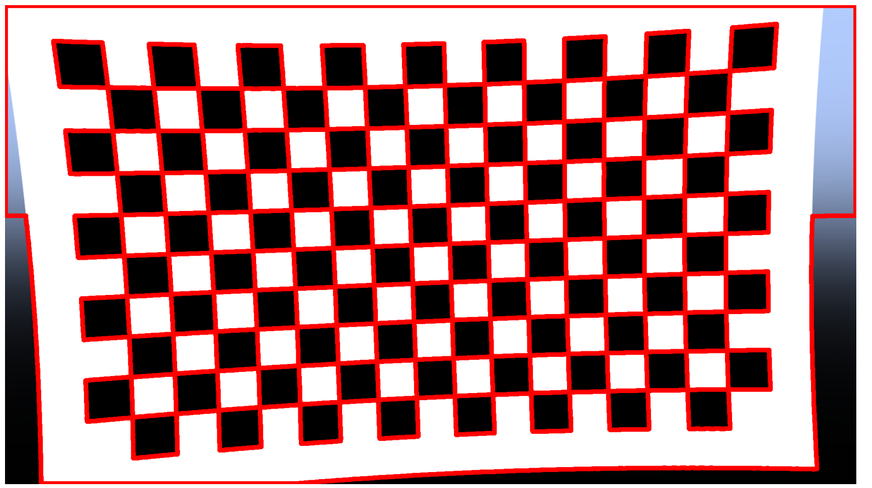


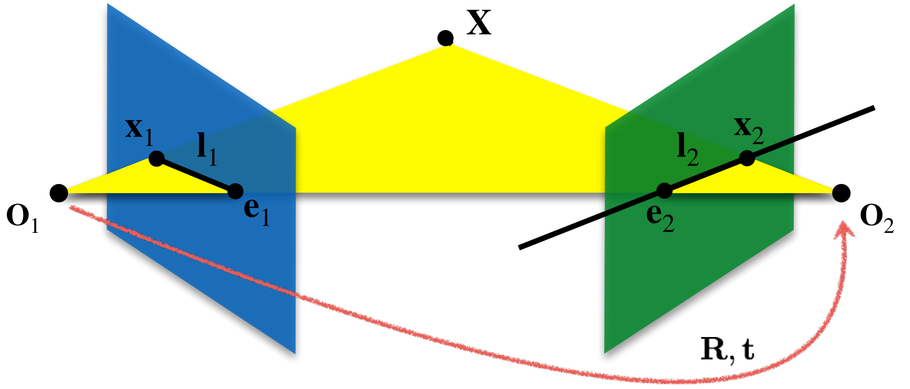
More at

- https://visionbook.mit.edu/imaging_geometry.html#sec-camera_calibration
- https://users.cecs.anu.edu.au/~hartley/Papers/fundamental/electronic-submission/fundamental.pdf
- https://colmap.github.io/
-

Recap

- Sparse features + correspondences
- Intrinsic + radial calibration
  - Known target (checkerboard)
- Epipolar Geometry, fundamental matrix
- Extrinsic camera calibration with linear regression + RANSAC
- Refined with non-linear bundle adjustment (including intrinsics, extrinsics, radial)
- We also get sparse 3D points







- Excellent open-source software exists (colmap, OpenMVG, etc.)

Deep Learning Era

Not covered, will do next time.

Skipping intermediate steps

- All modules of SfM have undergone neural improvements
  - advanced feature description,
  - more accurate image matching,
  - feature metric refinement,
  - and neural bundle adjustment+.

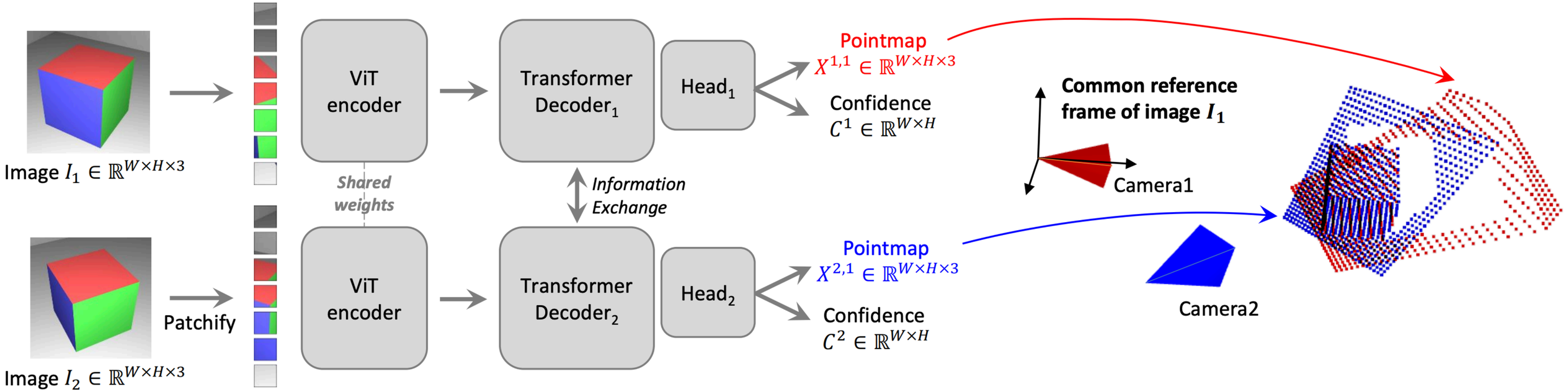
DUST3R: Dense Unconstrained Stereo 3D Reconstruction

- Challenges with traditional approaches
  - Mistakes/error accumulate
- Opportunity: it’s all minimization anyway
- Supervised (synthetically generated, reconstructed from SfM softwares, or captured using dedicated sensors)
- Monocular + stereo + multiple views
- Directly on 3D point maps
- Loss : 3D distance on ground truth point maps
- +confidence
- Camera info inferred as post-process (e.g. relative pose as 3D regression on point maps)
- Only 2 images

Point map

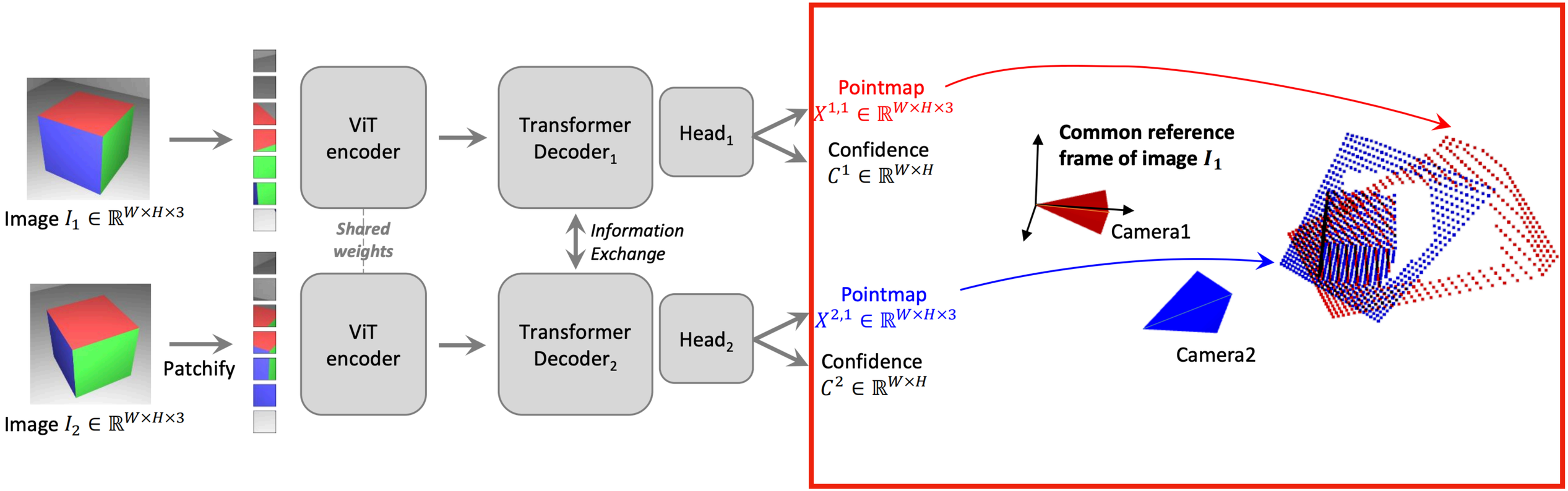
- Dense 2D field of 3D points (un-projection of an image according to depth)

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

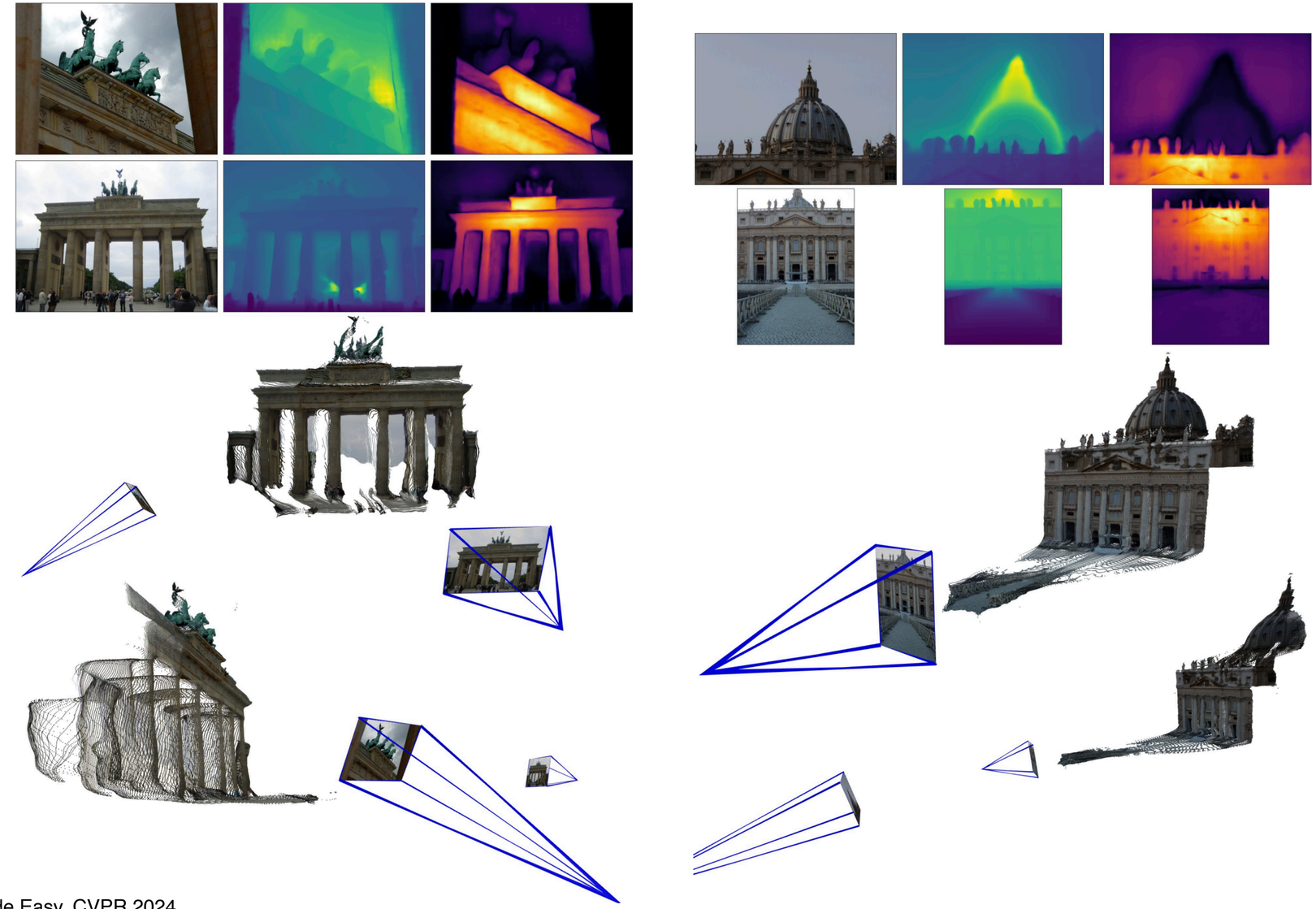
DUST3R: Pose SfM as supervised learning problem

Input

RGB images

No calibration, No poses

DUST3R: Pose SfM as supervised learning problem



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

Follow ups

MAST3R

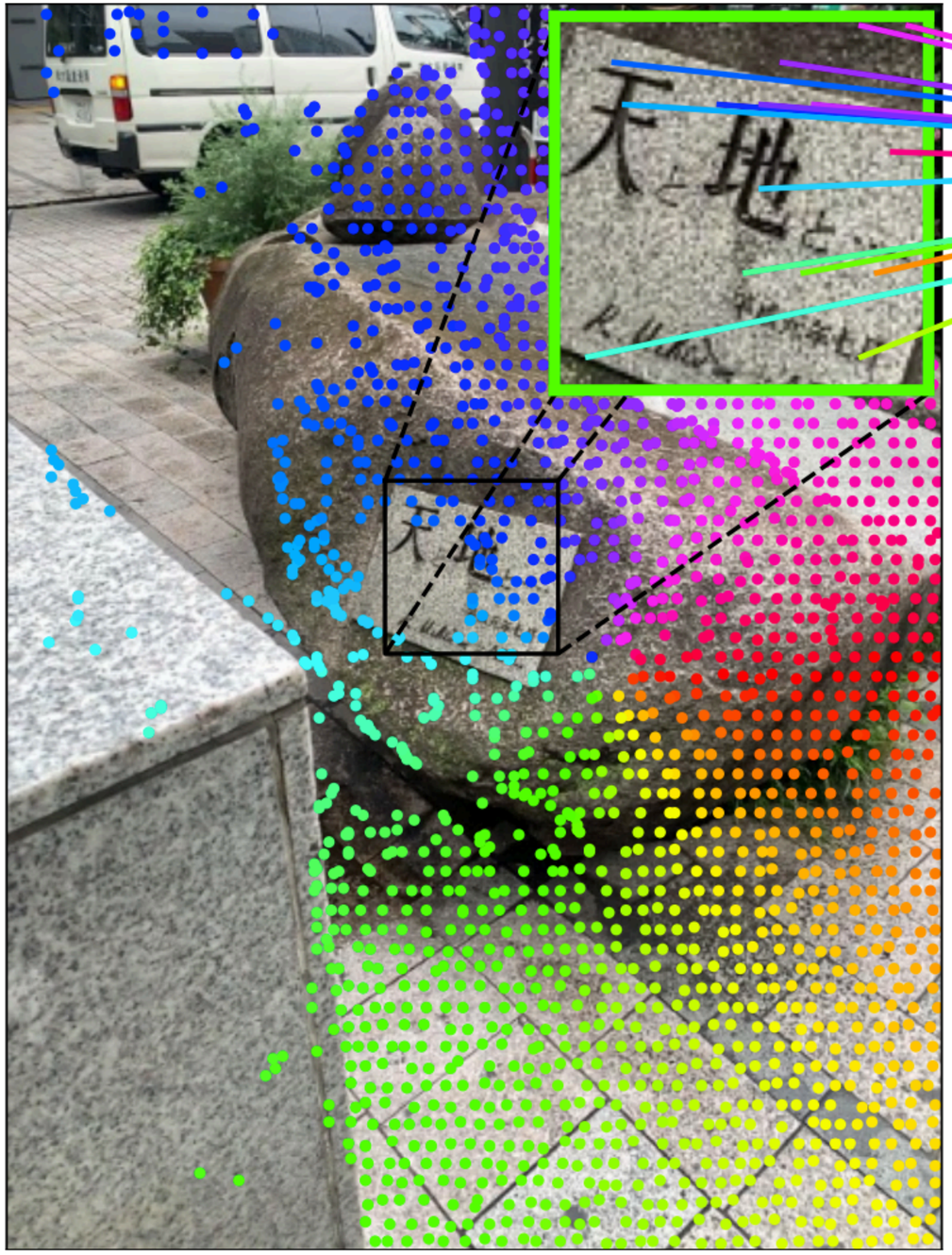
FAST3R

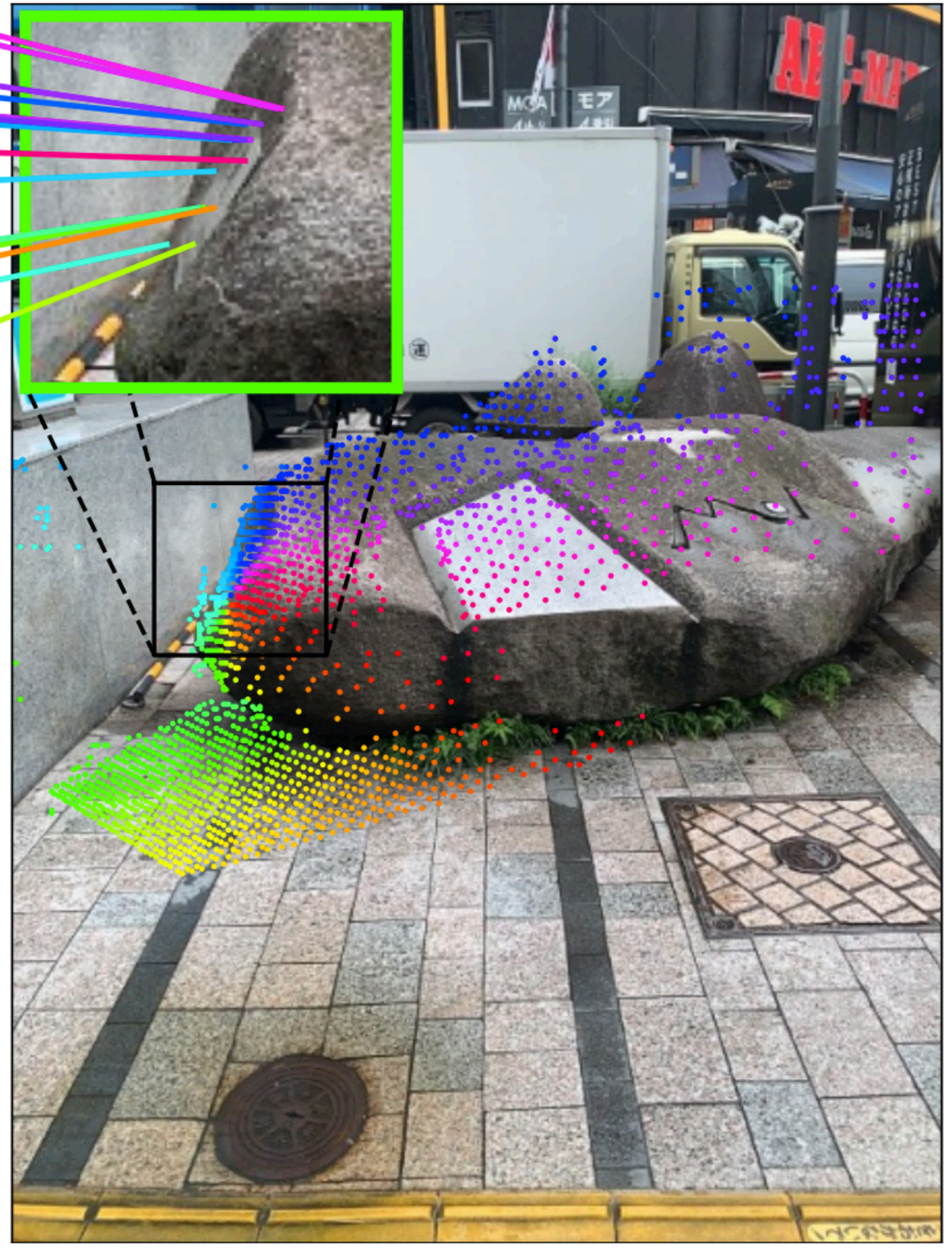
Instasplat

G

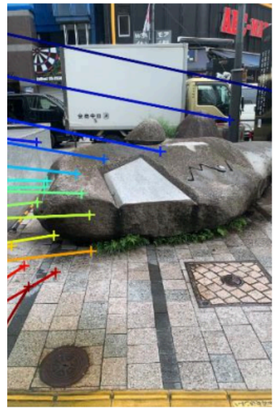








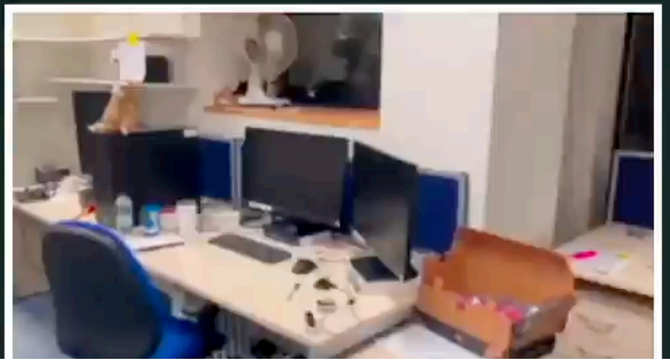
St3R



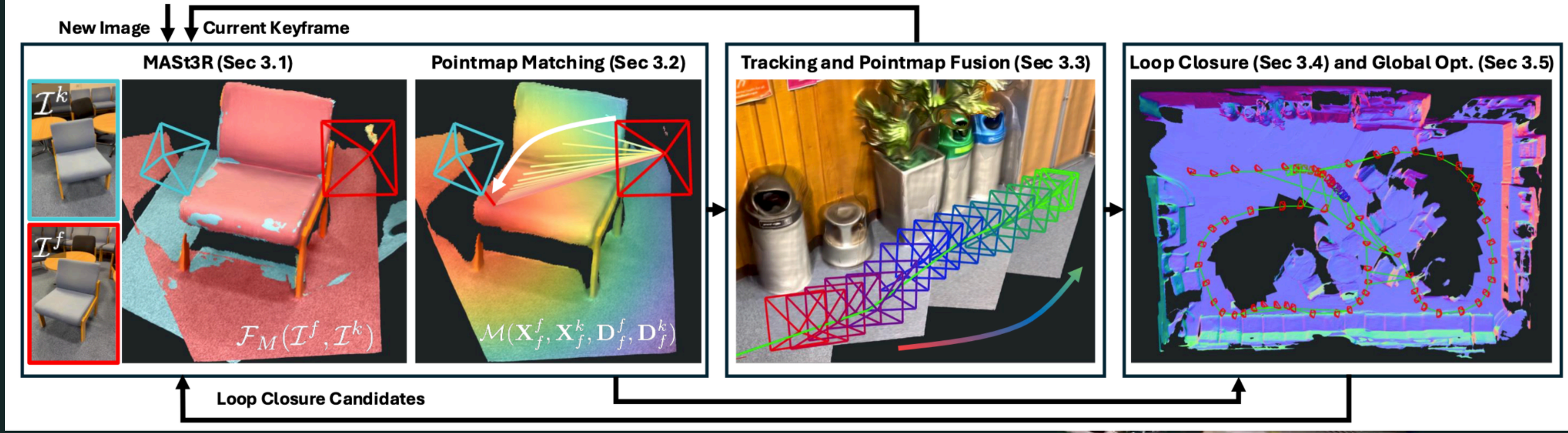
Leroy et al., Grounding Image Matching in 3D with MASt3R, CVPR 2024



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



Uncalibrated RGB
8x Playback



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

VGGT: Visual Geometry Grounded Transformer

Jianyuan Wang[1,2] Minghao Chen[1,2] Nikita Karaev[1,2] Andrea Vedaldi[1,2]
Christian Rupprecht[1] David Novotny[2]

[1]Visual Geometry Group, University of Oxford [2]Meta AI

1651v1 [cs.CV] 14 Mar 2025



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

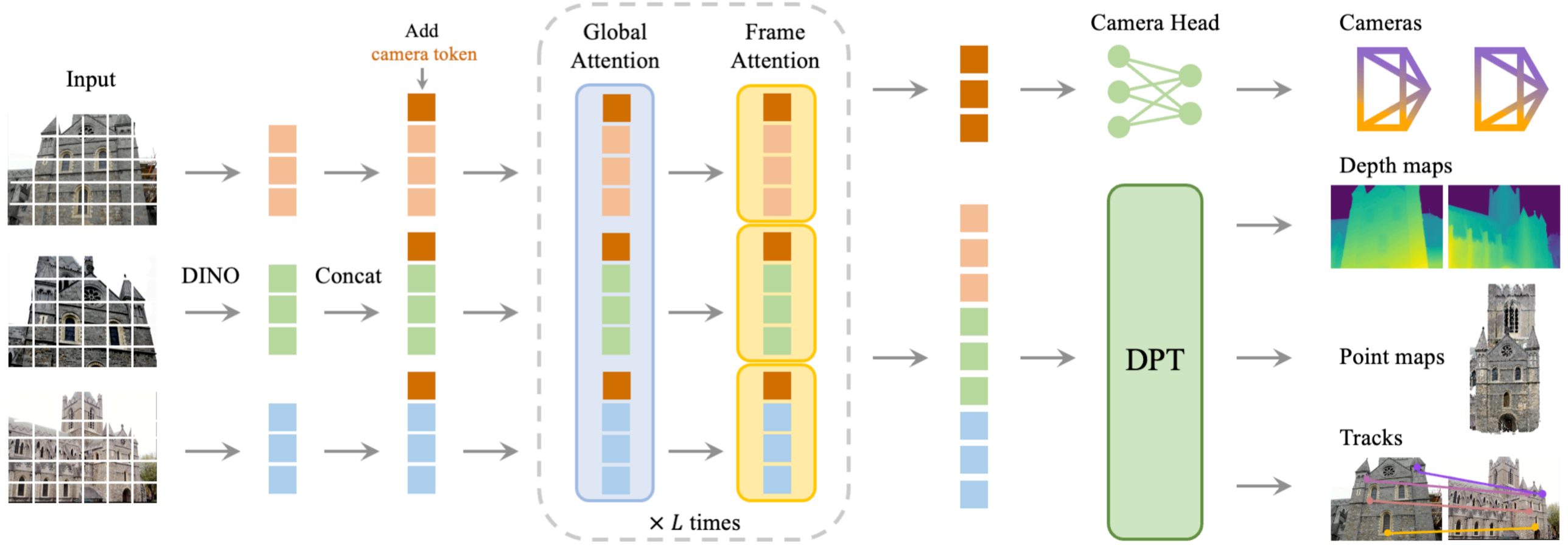
https://vgg-t.github.io/

VGGT

- Multiple images
- Multiple tasks:
  - camera parameter estimation,
  - multi-view depth estimation,
  - dense point cloud reconstruction,
  - point tracking
- Minimal 3D or other inductive biases
  (except for alternating between frame-wise and global attention, and parameterization of the virus problems)
- But (supervised) trained on a large number of publicly available datasets with 3D annotations.

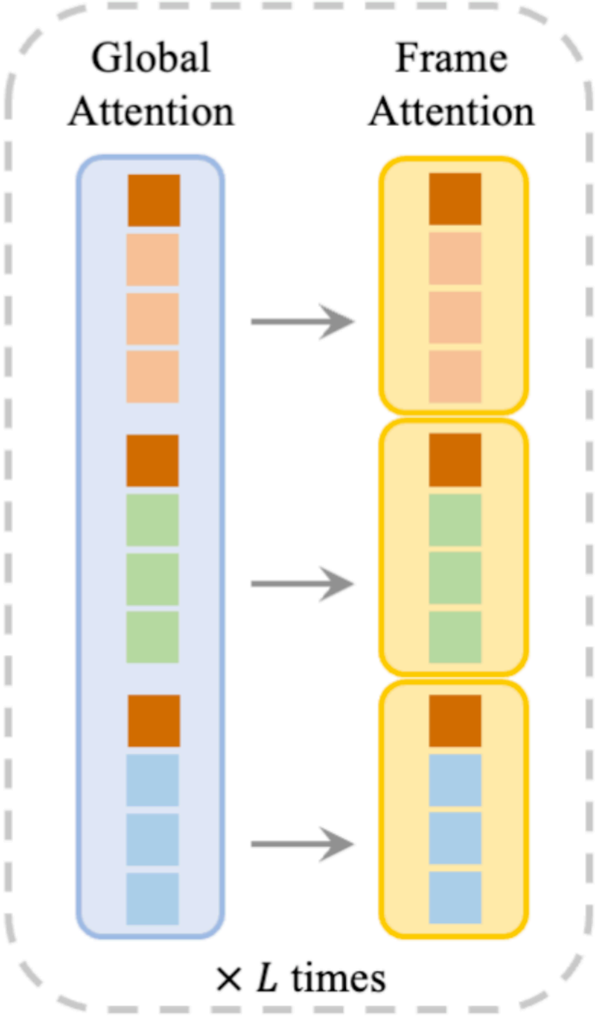
32 Views




Input
Add
camera token
Global
Attention
Frame
Attention
Camera Head
Cameras
DINO
Concat
Depth maps
DPT
Point maps
Tracks
× L times

Alternating attention

- frame-wise self-attention attends to the tokens tIk within each frame separately,
- global self-attention attends to the tokens tI across all frames jointly.
- This strikes a balance between integrating information across different images and normalizing the activations for the tokens within each image
- L= 24 layers of global and frame-wise attention



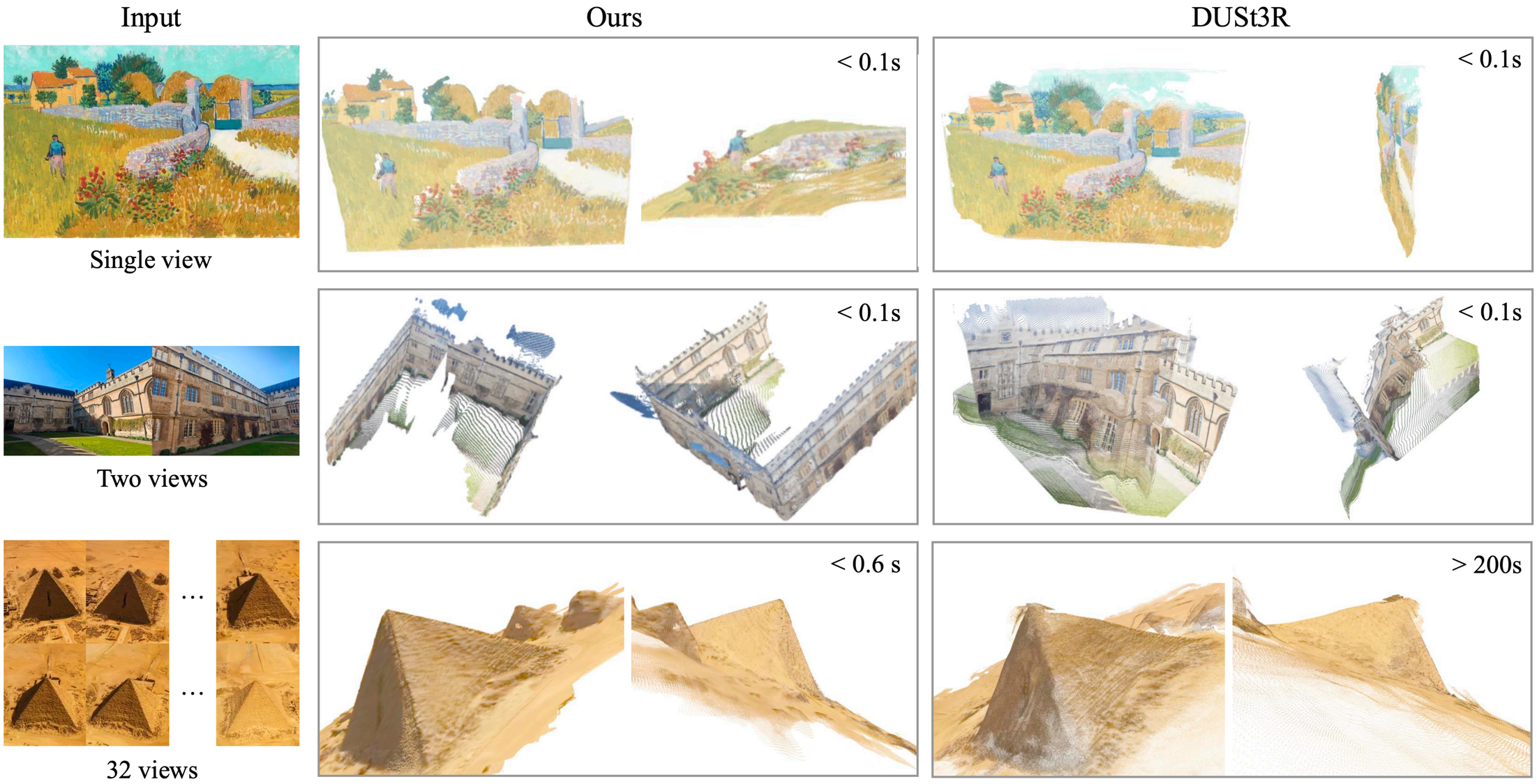


Figure 3. **Qualitative comparison of our predicted 3D points to DUSt3R on in-the-wild images.** As shown in the top row, our method successfully predicts the geometric structure of an oil painting, while DUSt3R predicts a slightly distorted plane. In the second row, our method correctly recovers a 3D scene from two images with no overlap, while DUSt3R fails. The third row provides a challenging example with repeated textures, while our prediction is still high-quality. We do not include examples with more than 32 frames, as DUSt3R runs out of memory beyond this limit.

32 Views



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!