Embodied Vision

Image Formation and Multi-view Geometry

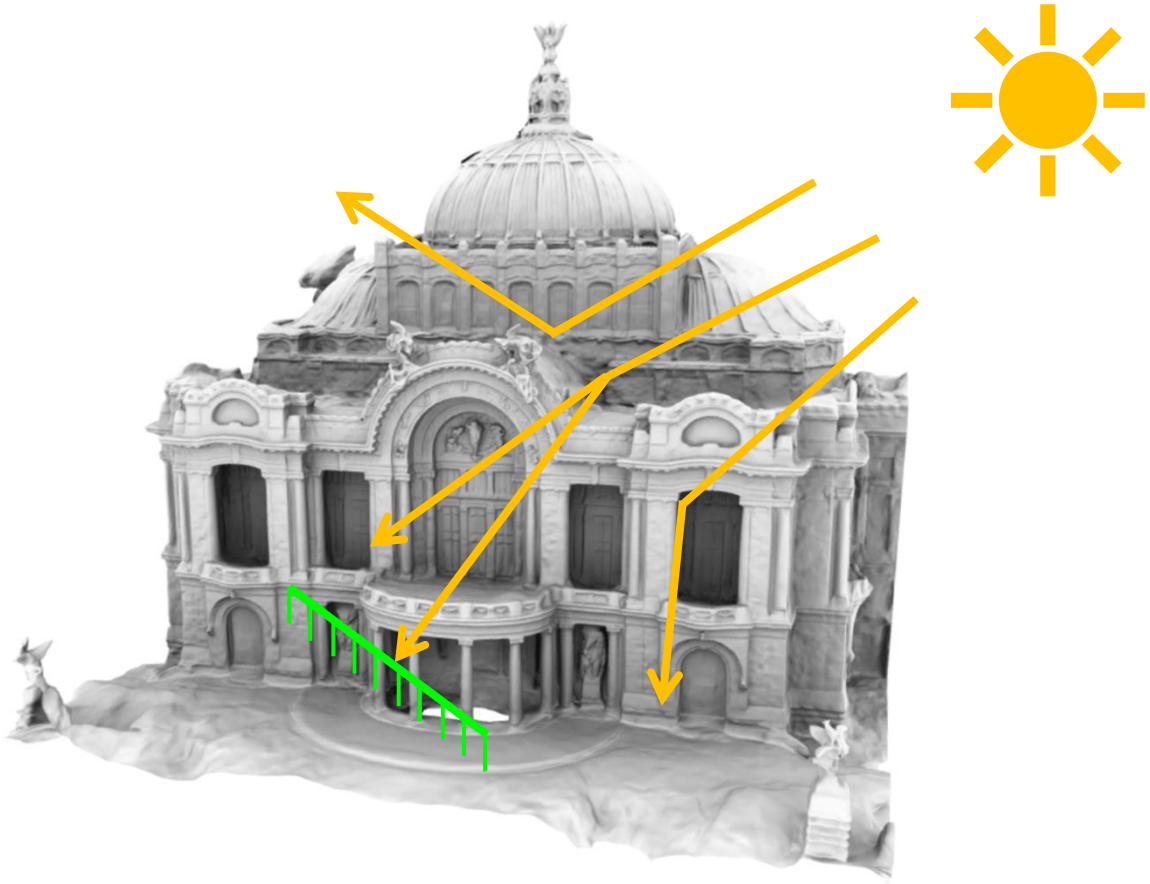
Tsung-Wei Ke

Spring 2026

Disclaimer

- This lecture borrows contents heavily from
  - Machine Learning for Inverse Graphics by Vincent Sitzmann at MIT
  - Computer Vision by Andreas Geiger at University Tübingen
  - Intro to Computer Vision and Computational Photography by Alyosha Efros at UC Berkeley

Image Formation: How Do We See 2D Images from the 3D World?

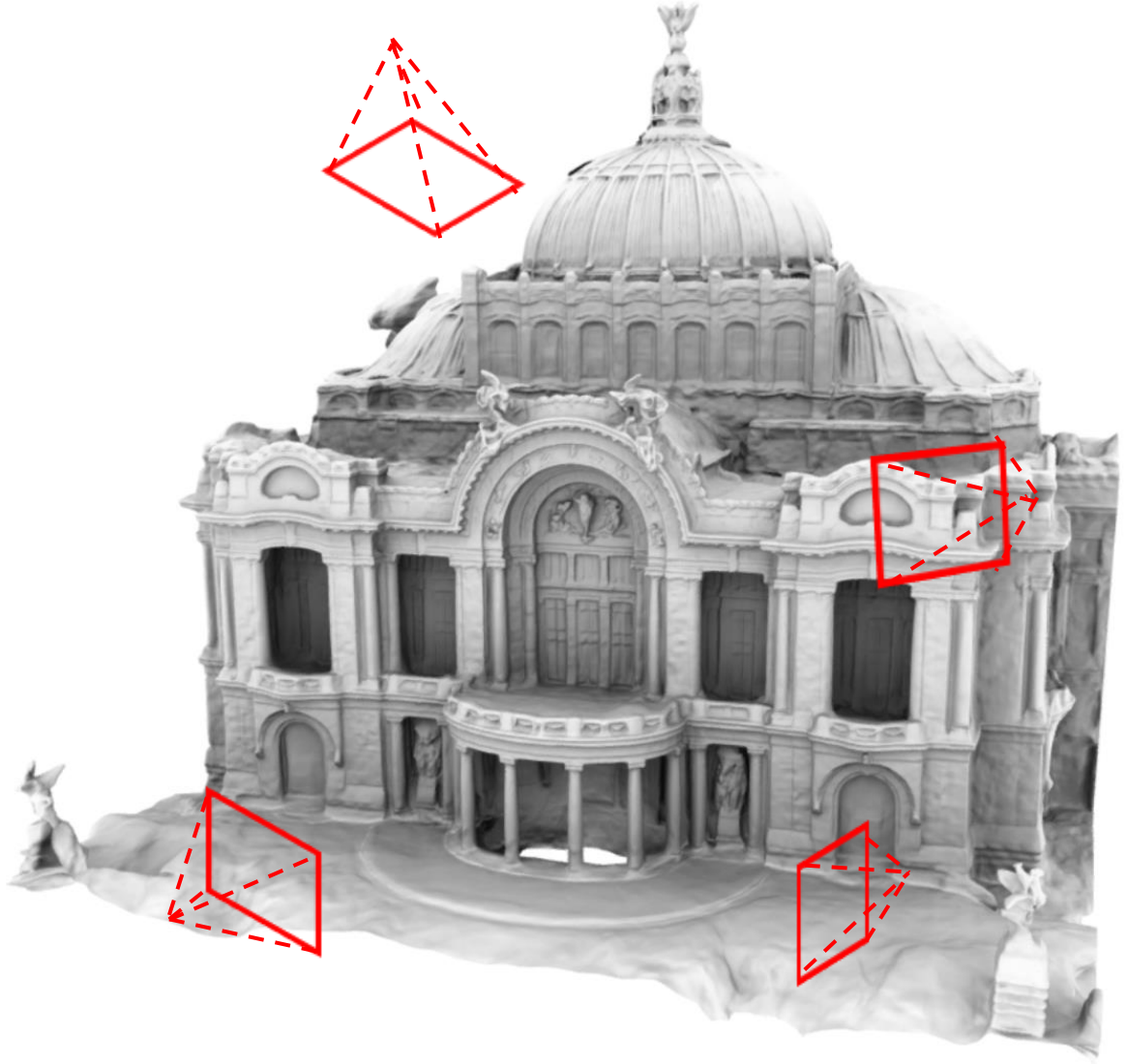


Multi-view Geometry: How Do We Reconstruct the 3D World from 2D Images?









Content

- Image Formation
  - Pinhole camera model
  - 2D transformation
  - 3D transformation
- Geometry Reconstruction
  - Reconstruction with depths
  - Multi-view geometry reconstruction with camera motions and correspondence
  - Epipolar geometry reconstruction

Content

- Image Formation
  - ➢Pinhole camera model
  - ➢2D transformation
  - ➢3D transformation
- Geometry Reconstruction
  - ➢Reconstruction with depths
  - ➢Multi-view geometry reconstruction with camera motions and correspondence
  - ➢Epipolar geometry reconstruction

2D Image Formation from the 3D World: A simplified model of photometric image formation

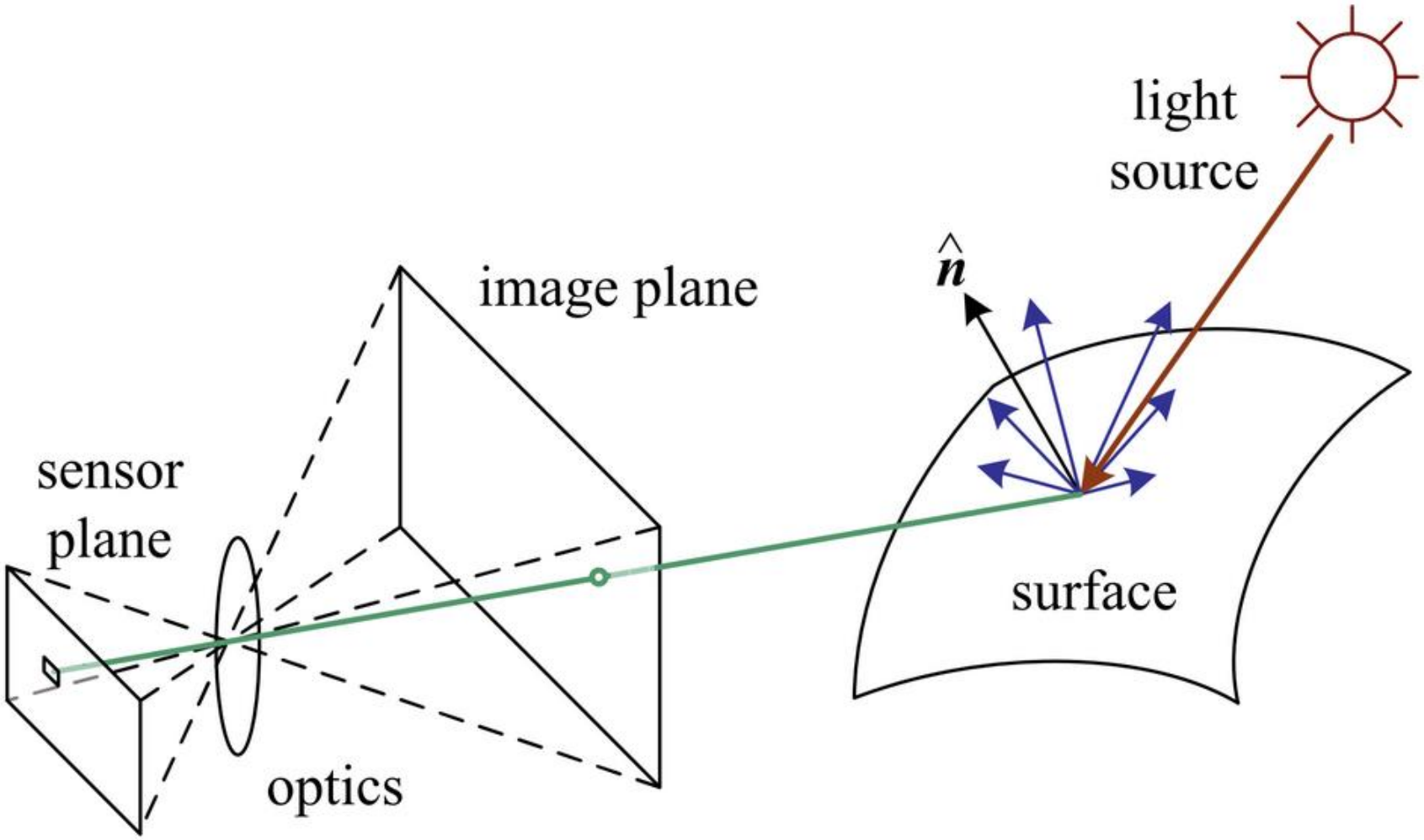
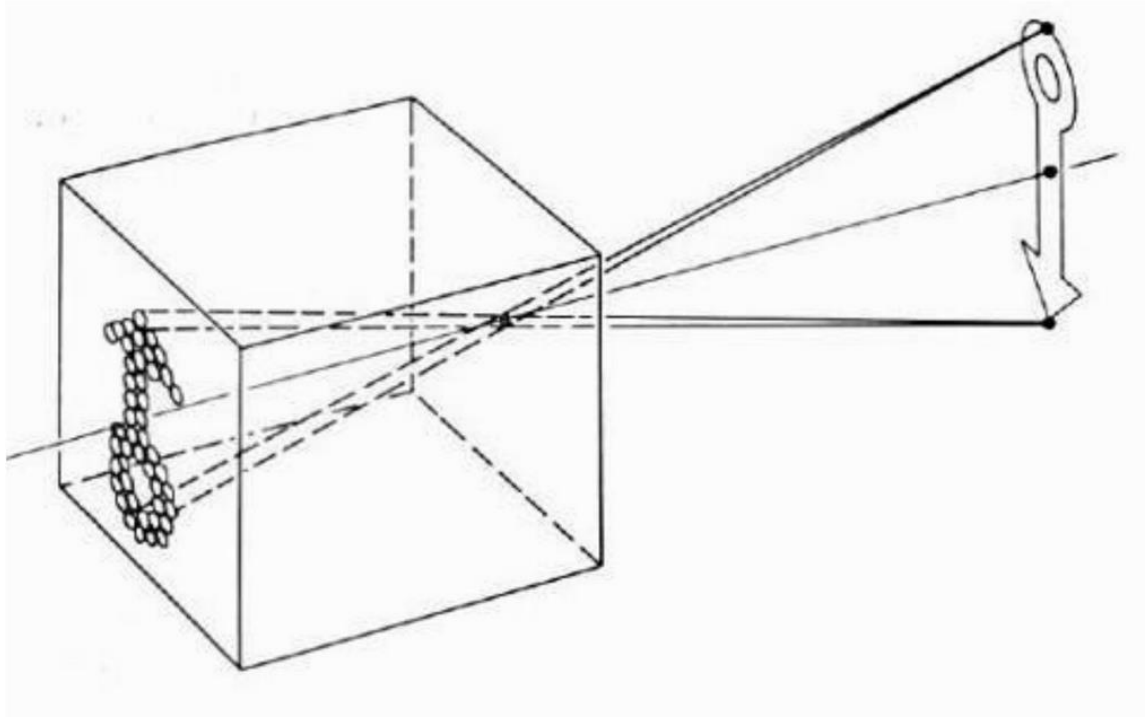
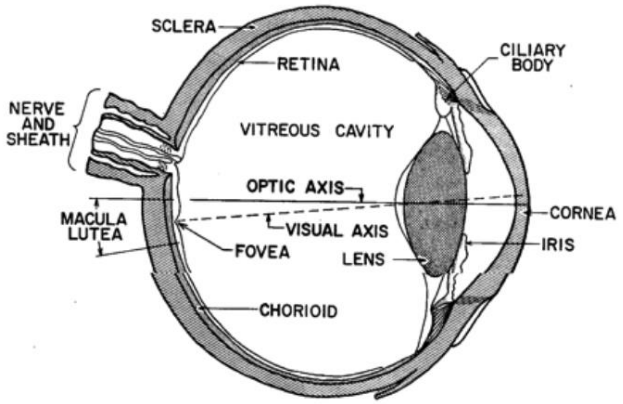


Image source S. Seitz text book

Perspective Projection

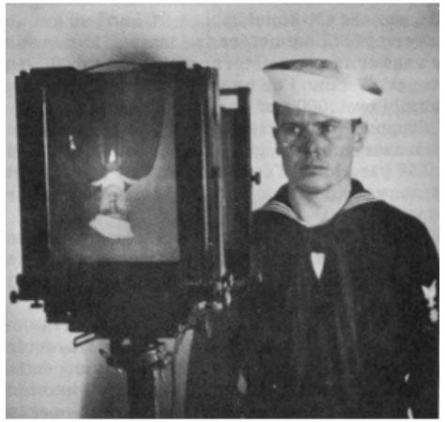




Animal eye: a long time ago

Gemma Frisius, 1558

Pinhole Perspective Projection: Brunelleschi, 15th Century



Photographic camera: Niepce, 1816

Image source:

- US Navy Manual of Basic Optics and Optical Instruments, prepared by Bureau of Naval Personnel. Reprinted by Dover Publications, Inc., 1969.
- Remote Sensing. A.L. Nowicki, “Stereoscopy.” Manual of Photogrammetry, Thompson, Radlinski, and Speert (eds.), third edition, 1966

Other Types of Camera Projection

Ellipsoidal projection

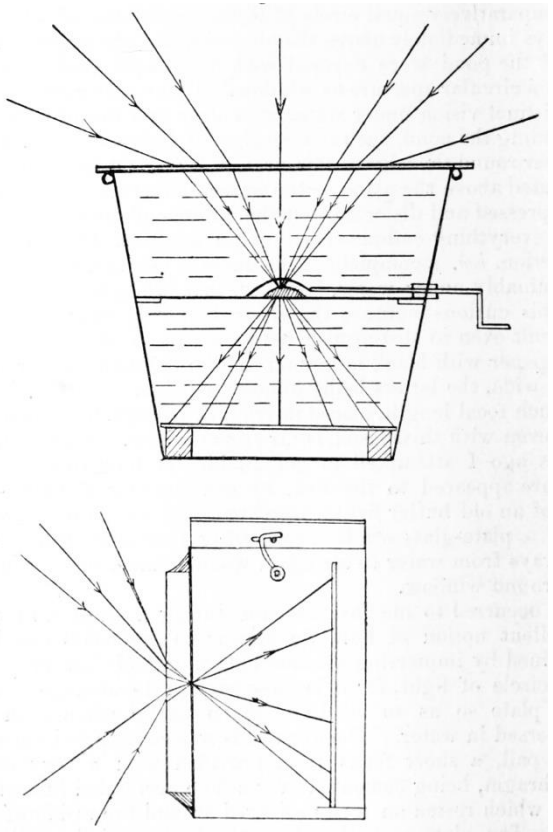
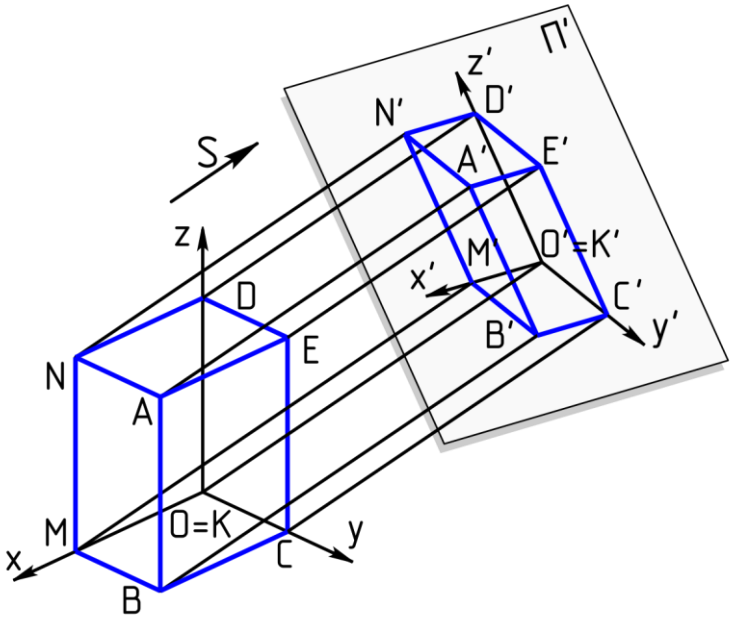
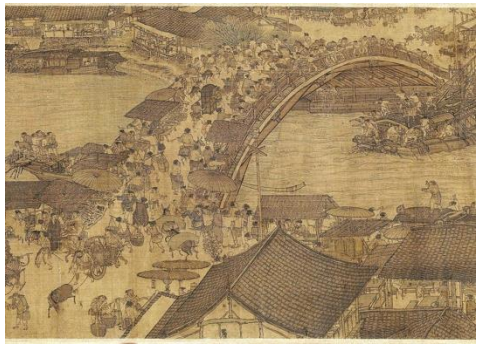


Image source obert W. Wood, "Fish-Eye Views, and Vision under Water" (August 1906)

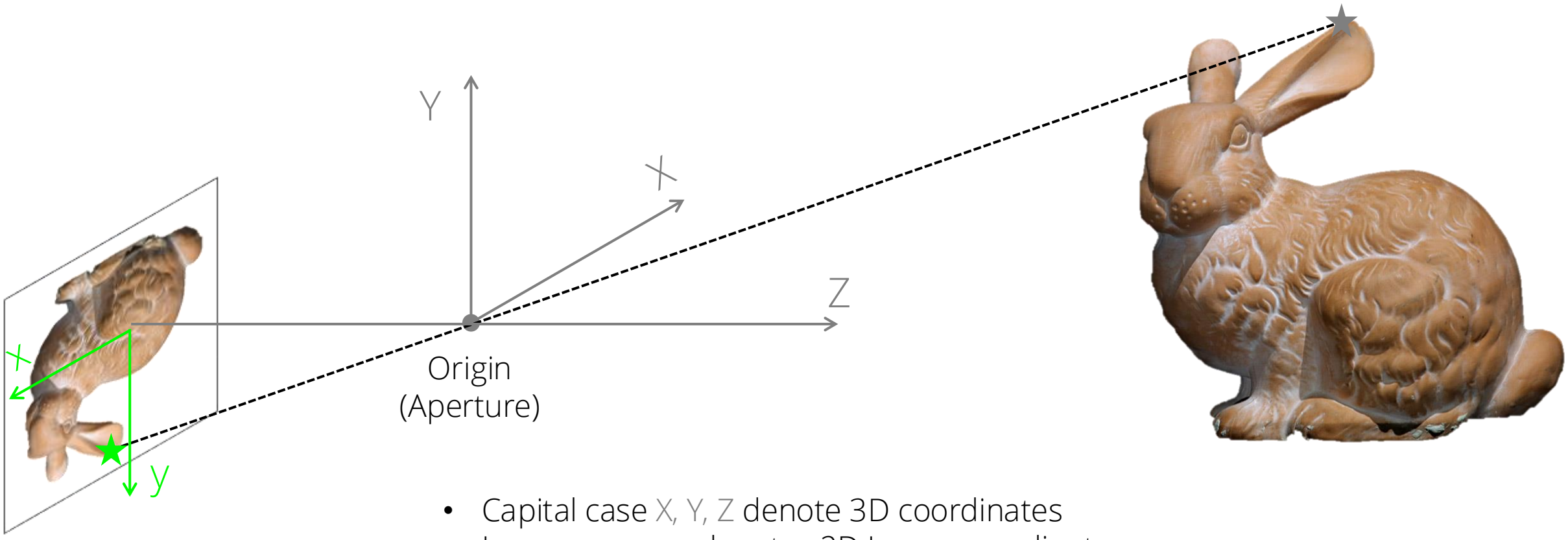
Parallel projection





https://en.wikipedia.org/wiki/Parallel_projection#/media/File:Axonometric_projection.svg

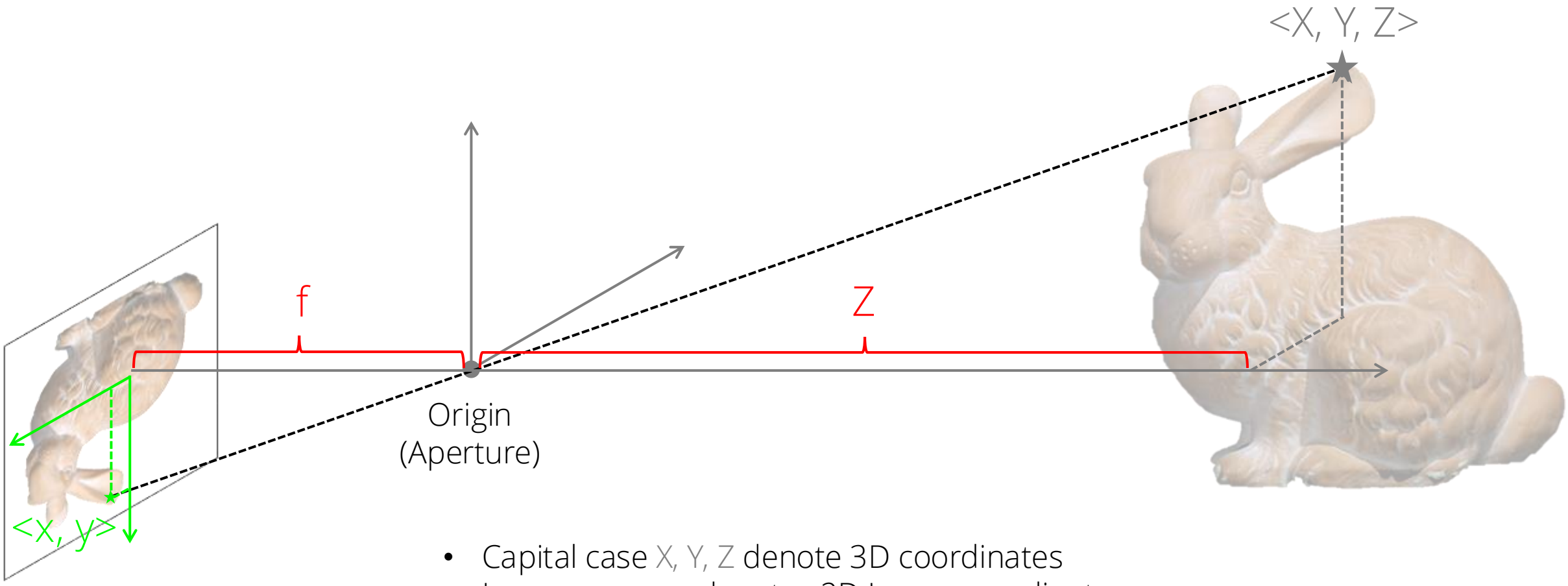
Perspective Projection



- Capital case X, Y, Z denote 3D coordinates
- Lower case x, y denotes 2D Image coordinates

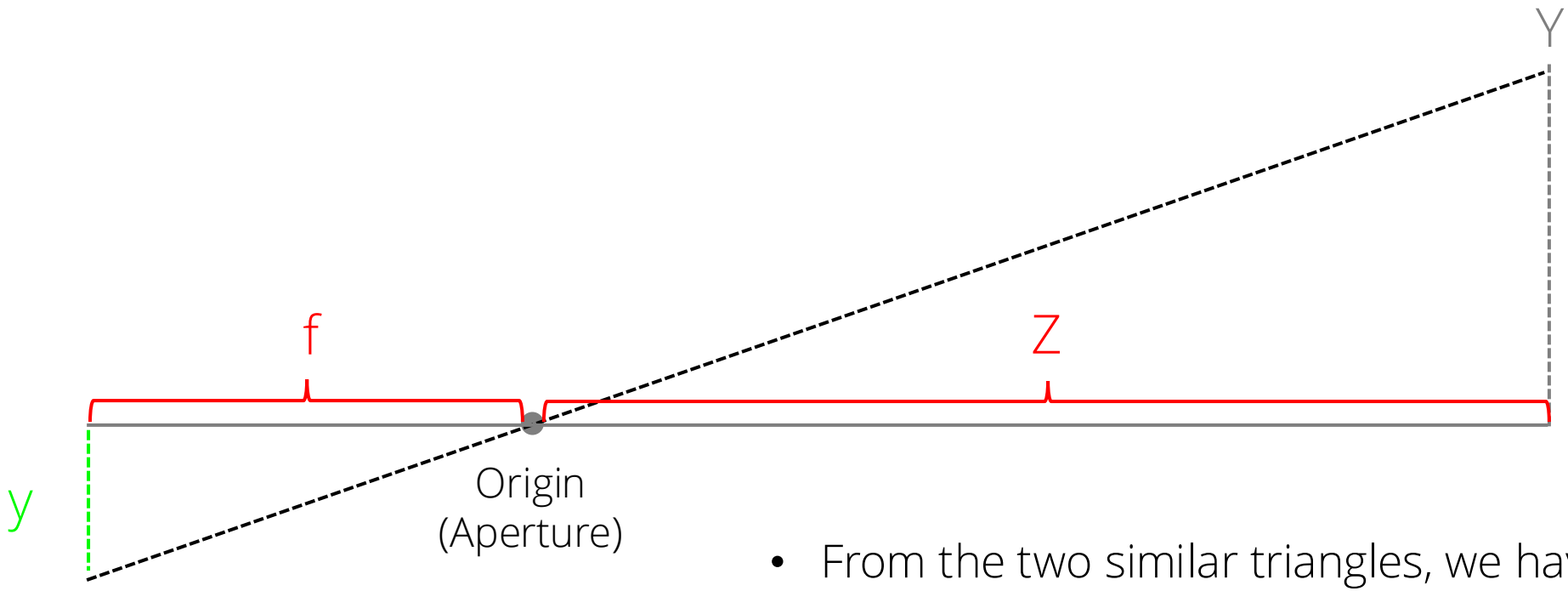
- How to derive the relation between <x, y> and <X, Y, Z> ?

Perspective Projection



- Capital case X, Y, Z denote 3D coordinates
- Lower case x, y denotes 2D Image coordinates
- f denotes the camera focal length

Perspective Projection



- From the two similar triangles, we have:

$$\frac{y}{f} = \frac{Y}{Z} \Rightarrow y = f\frac{Y}{Z}$$

- Likewise

$$\frac{x}{f} = \frac{X}{Z} \Rightarrow x = f\frac{X}{Z}$$

We Can Transfer Between 2D Pixel Coordinates and 3D Coordinates, if f and Z are Known

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

<x, y>

Depth map

<X, Y, Z>

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

Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer. Rantfl et al.

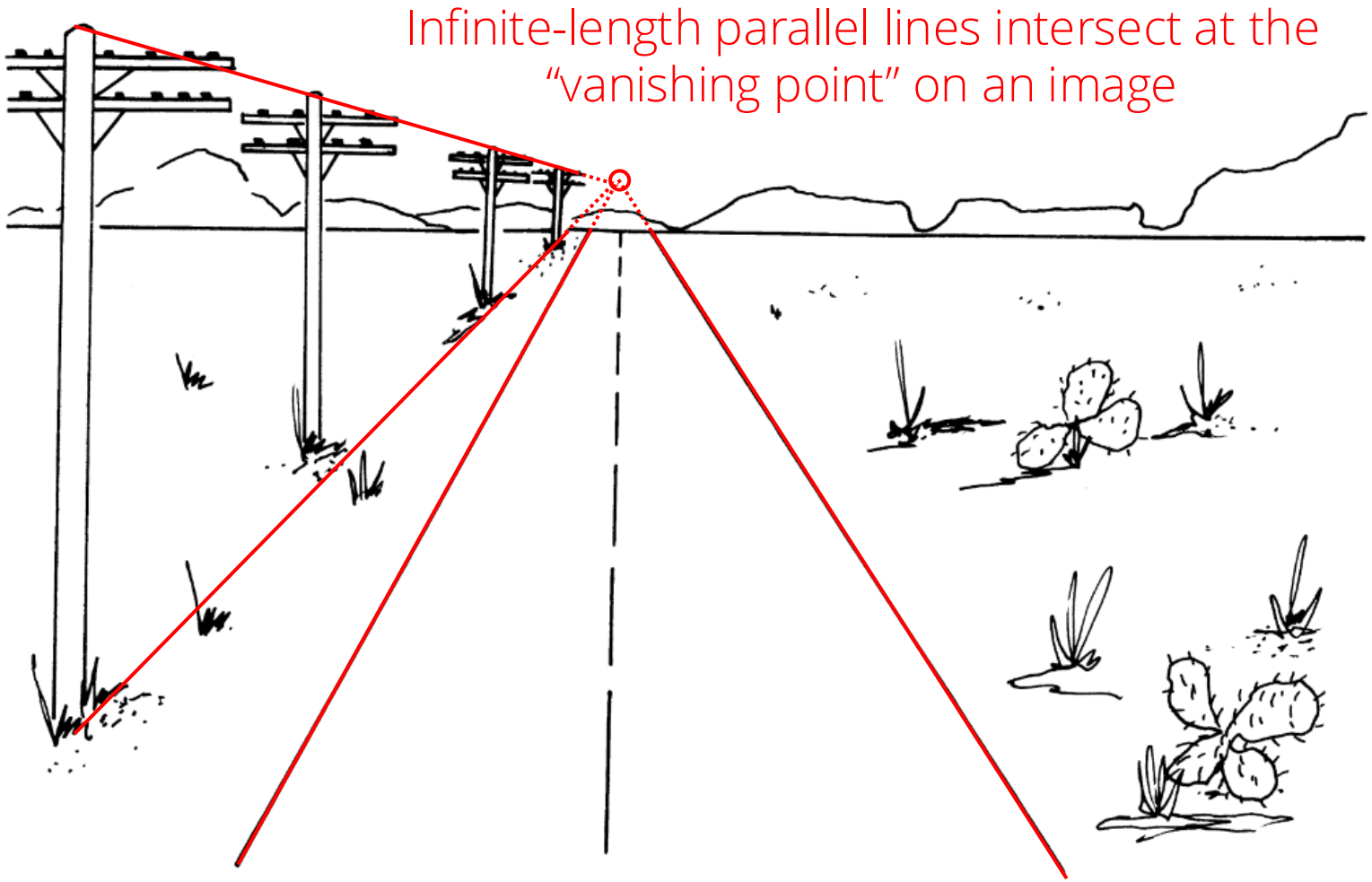
Information Loss by Perspective Projection

Slide credit A. Efros
Image source: Fooling the eye Making of 3D sidewalk art:
http://www.youtube.com/watch?v=3SNYtd0Ayt0

Information Loss by Perspective Projection

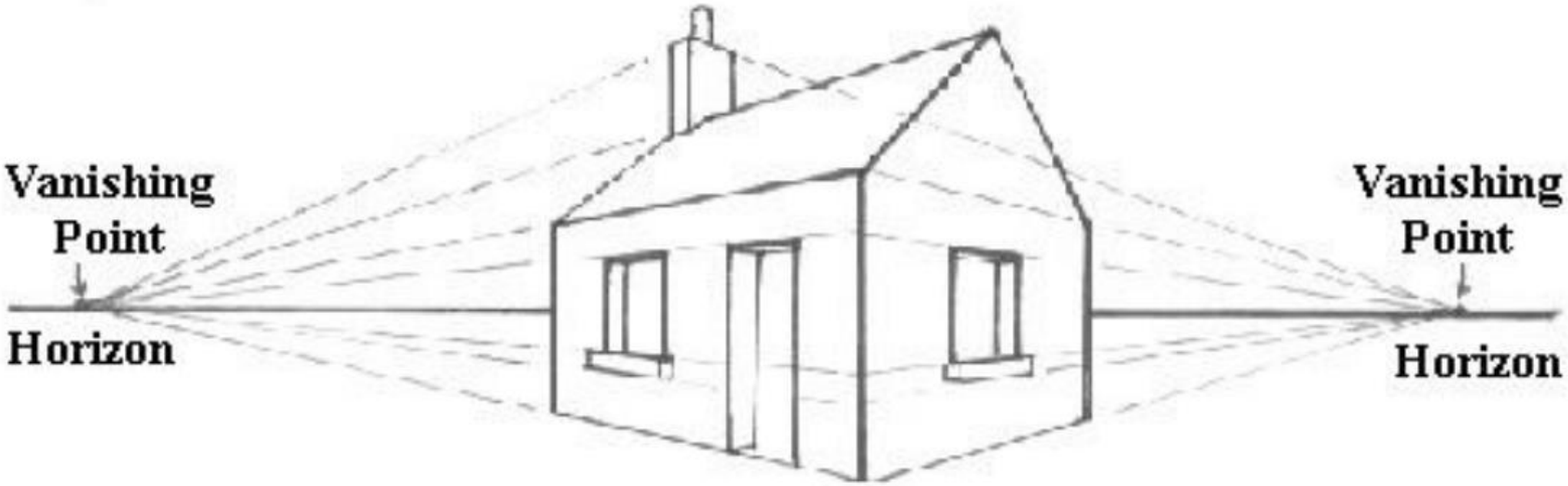
Slide credit A. Efros
Image source: Fooling the eye Making of 3D sidewalk art:
http://www.youtube.com/watch?v=3SNYtd0Ayt0

Angles Are Not Preserved by Perspective Projection



Slide credit J. Malik

Each Family of Parallel Lines has its Own Vanishing Point



Slide credit J. Malik

Proof

- Line in 3D space:

$$X(t) = X_0 + at$$
$$Y(t) = Y_0 + bt$$
$$Z(t) = Z_0 + ct$$

- Perspective projection:

$$y(t) = f\frac{Y(t)}{Z(t)} = f\frac{Y_0 + bt}{Z_0 + ct}$$
$$x(t) = f\frac{X(t)}{Z(t)} = f\frac{X_0 + at}{Z_0 + ct}$$

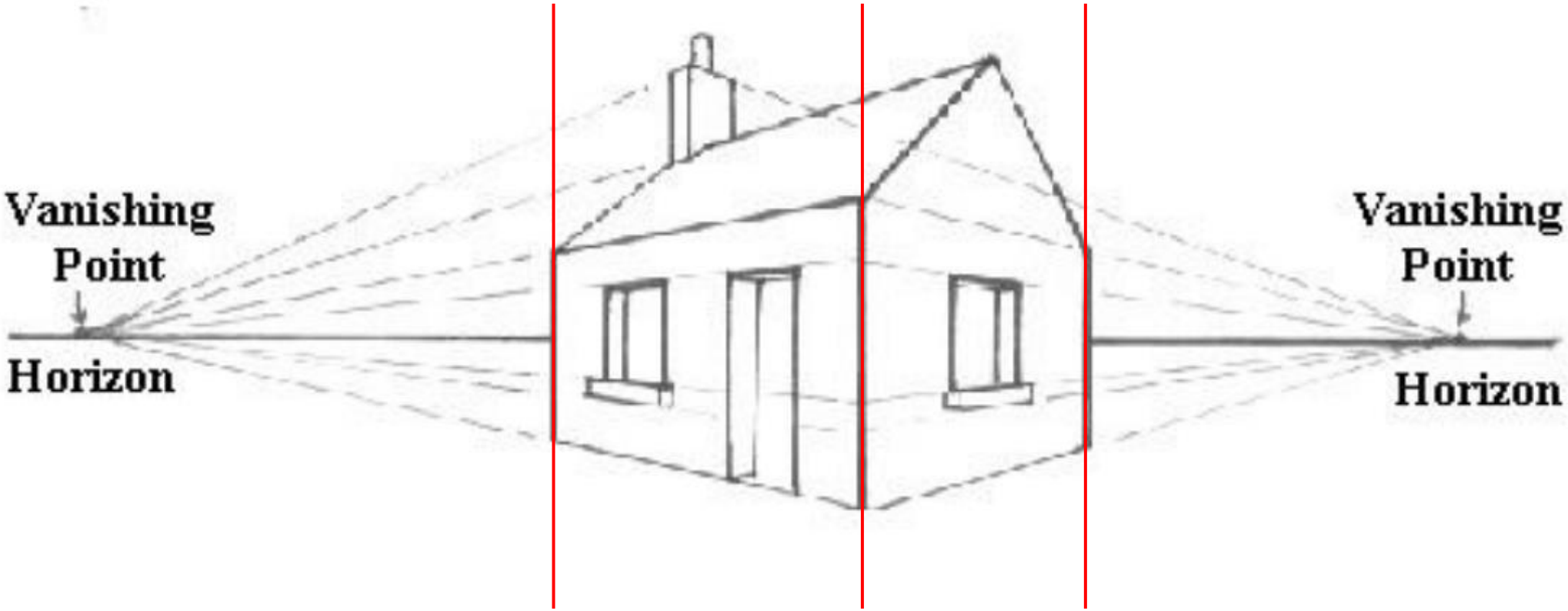
- Projective lines when $t \to \infty$:

$$\lim_{t\to\infty} y(t) = \lim_{t\to\infty} f\frac{Y_0 + bt}{Z_0 + ct} = f\frac{b}{c}$$
$$\lim_{t\to\infty} x(t) = \lim_{t\to\infty} f\frac{X_0 + at}{Z_0 + ct} = f\frac{a}{c}$$

This is a point!

Each Family of Parallel Lines has its Own Vanishing Point

Where is the “vanishing point” of these vertical parallel lines?



Slide credit J. Malik

Proof

- Line in 3D space:

$$X(t) = X_0 + 0t = X_0$$
$$Y(t) = Y_0 + bt$$
$$Z(t) = Z_0 + 0t = Z_0$$

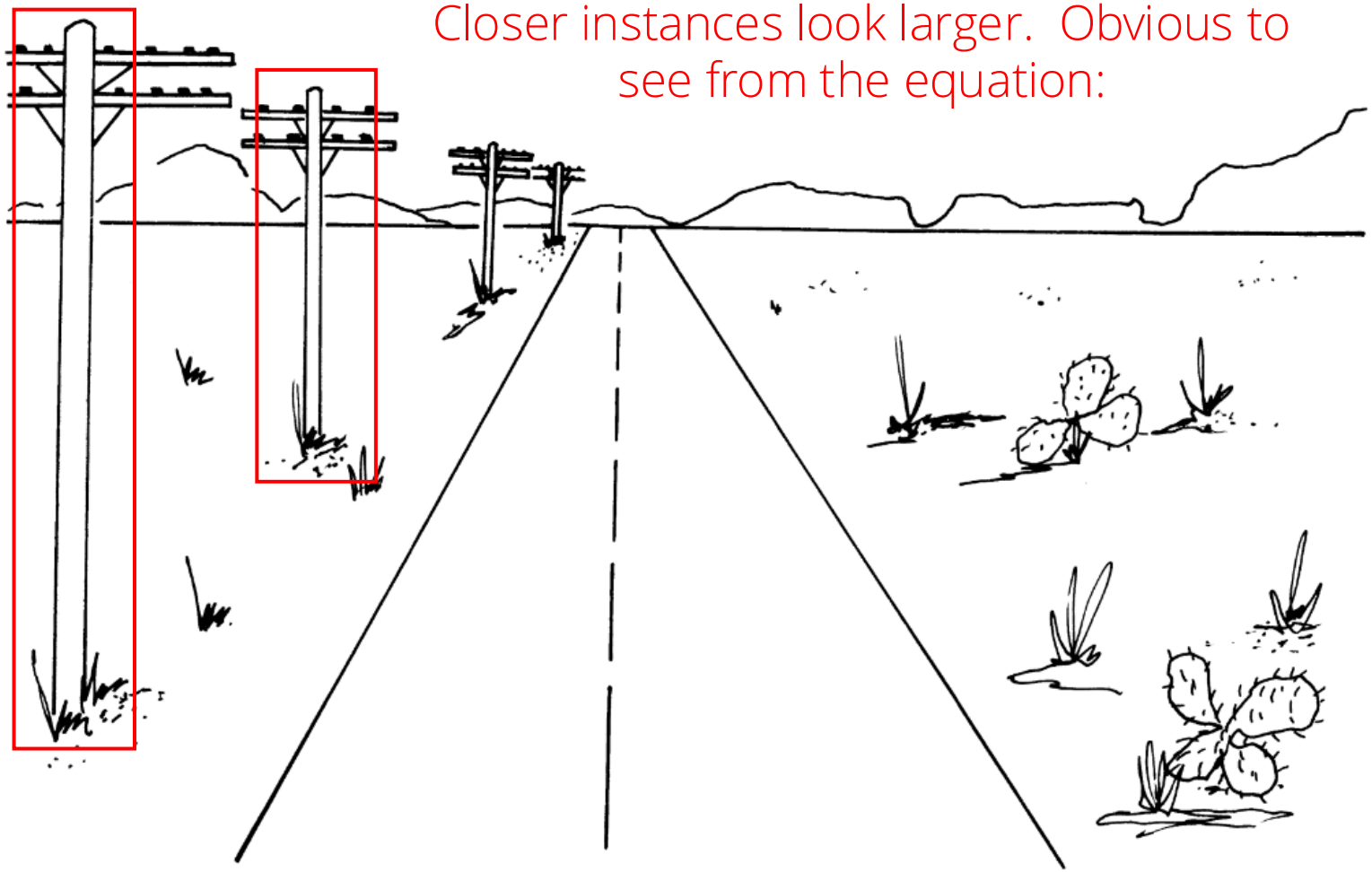
- Perspective projection:

$$y(t) = f\frac{Y(t)}{Z(t)} = f\frac{Y_0 + bt}{Z_0}$$
$$x(t) = f\frac{X(t)}{Z(t)} = f\frac{X_0}{Z_0}$$

- Projective lines when $t \to \infty$:

$$\lim_{t\to\infty} y(t) = \lim_{t\to\infty} f\frac{Y_0 + bt}{Z_0} = f\frac{b}{0}$$
$$\lim_{t\to\infty} x(t) = \lim_{t\to\infty} f\frac{X_0}{Z_0}$$

Scales Are Not Preserved by Perspective Projection



Closer instances look larger. Obvious to see from the equation:

$$y = f\frac{Y}{Z}$$

$$x = f\frac{X}{Z}$$

Illusion of Perspective Projection

Slide credit S. Tulsiani

How to Convert Perspective Transformation into a Matrix Form

- Homogeneous image coordinates: (x, y)
- Homogeneous scene coordinates: (X, Y, Z)
- Perspective projection:

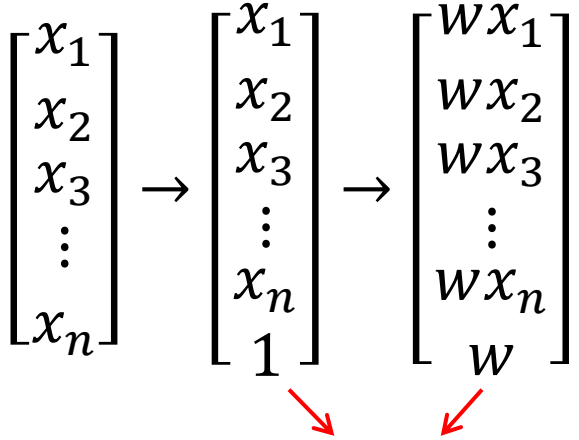
$$y = f\frac{Y}{Z}$$
$$x = f\frac{X}{Z}$$

$$\longrightarrow \quad \begin{bmatrix} & & \\ & & \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix}$$

Dimension is inconsistent, awkward!

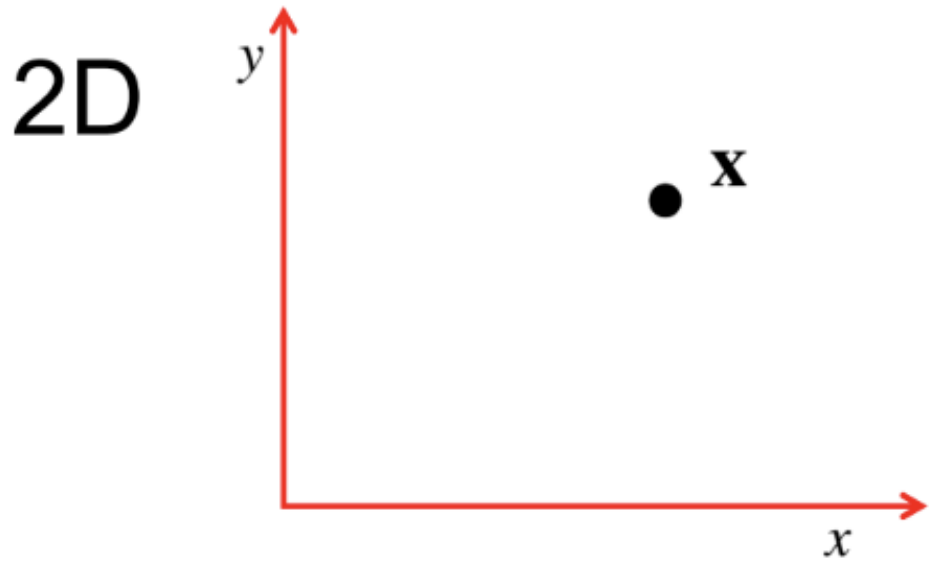
Homogeneous Coordinates

- Representing n-dim coordinates with (n+1)-dim



- We can think of this as a line through the origin in (n+1)-dimensional space

Homogeneous Coordinates



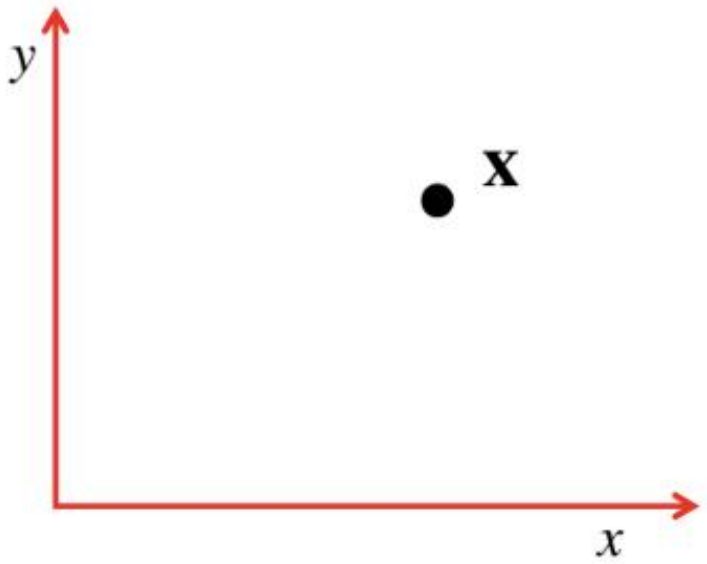
$$\mathbf{x} = (x, y) \quad \Rightarrow \quad \tilde{\mathbf{x}} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

heterogeneous coordinates

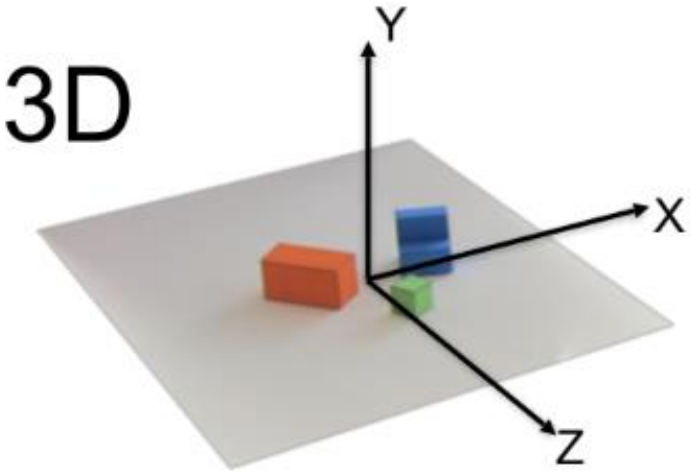
homogeneous coordinates

Homogeneous Coordinates

2D



3D



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

heterogeneous coordinates

homogeneous coordinates

$$\mathbf{X} = (X, Y, Z) \quad \Rightarrow \quad \tilde{\mathbf{X}} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Slide credit V. Sitzmann

Homogeneous Coordinates

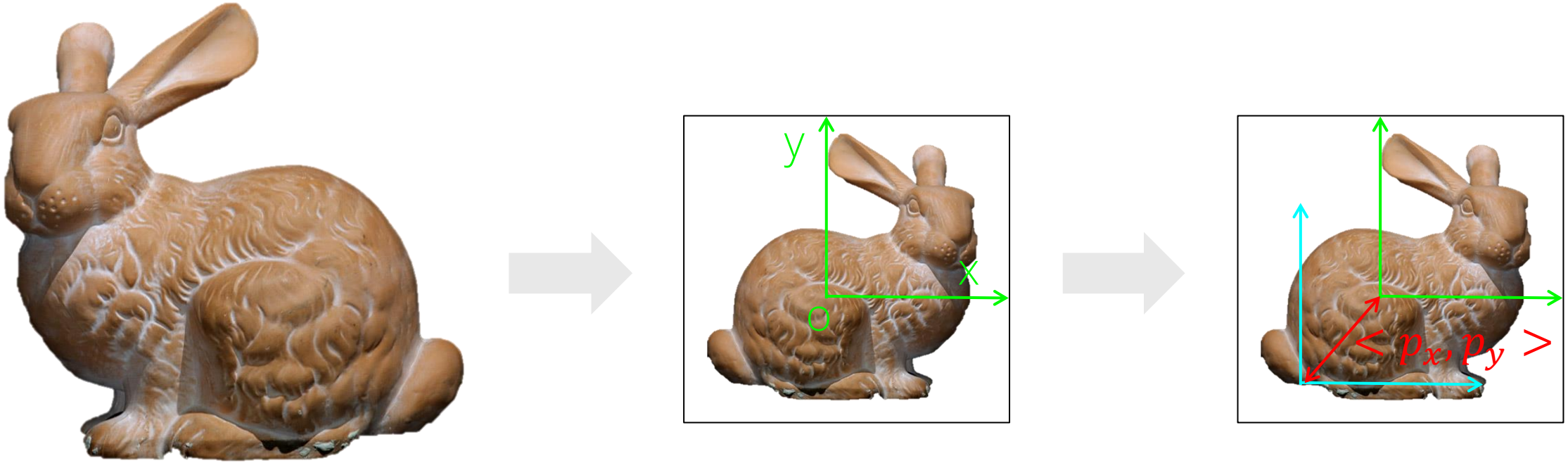
- Homogeneous image coordinates: $(x, y) \Longrightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
- Homogeneous scene coordinates: $(X, Y, Z) \Longrightarrow \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$
- We can represent the pin-hole camera model with homogeneous coordinates

$$\begin{aligned} x &= f\frac{X}{Z} \\ y &= f\frac{Y}{Z} \end{aligned} \longrightarrow \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} \Longrightarrow \begin{bmatrix} f\frac{X}{Z} \\ f\frac{Y}{Z} \\ 1 \end{bmatrix}$$

Content

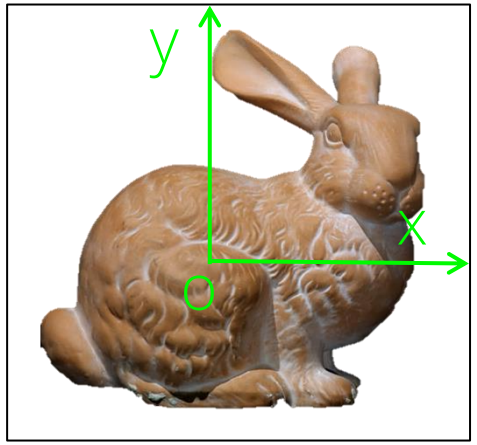
- Image Formation
  - ➢Pinhole camera model
  - ➢2D transformation
  - ➢3D transformation
- Geometry Reconstruction
  - ➢Reconstruction with depths
  - ➢Multi-view geometry reconstruction with camera motions and correspondence
  - ➢Epipolar geometry reconstruction

What is the Matrix Form of Perspective Projection with



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

Decompose the Projection Matrix

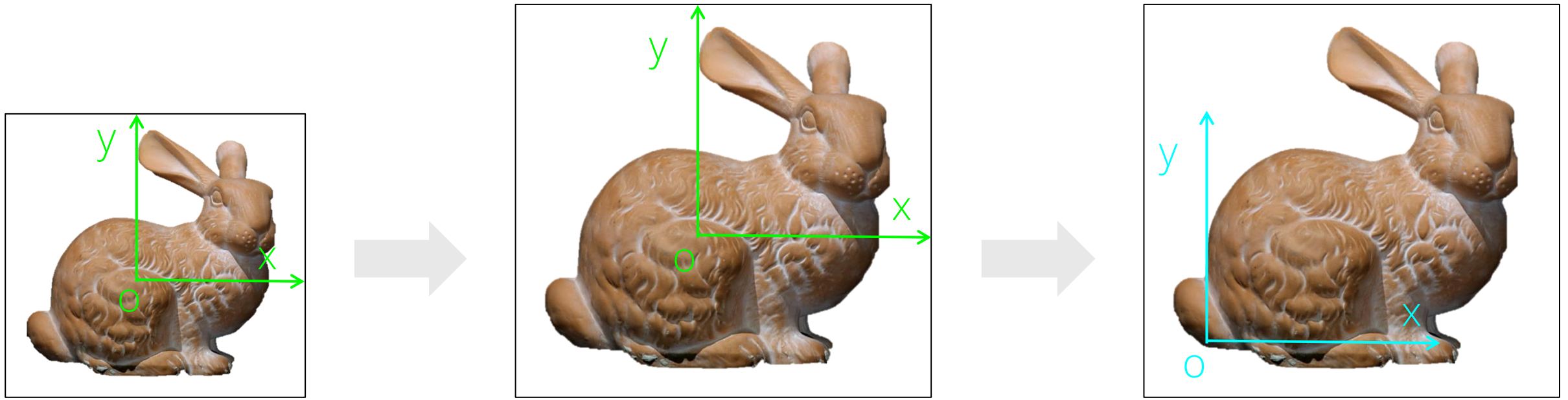


Perspective projection from 3D to 2D, assuming image plane at z=1 and shared camera/image origin

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \Rightarrow \begin{bmatrix} X/Z \\ Y/Z \\ 1 \end{bmatrix}$$

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

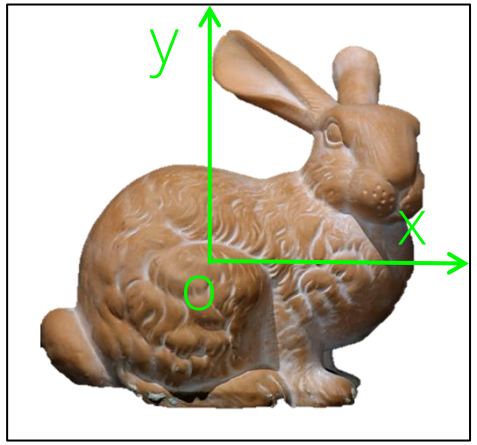
Decompose the Projection Matrix

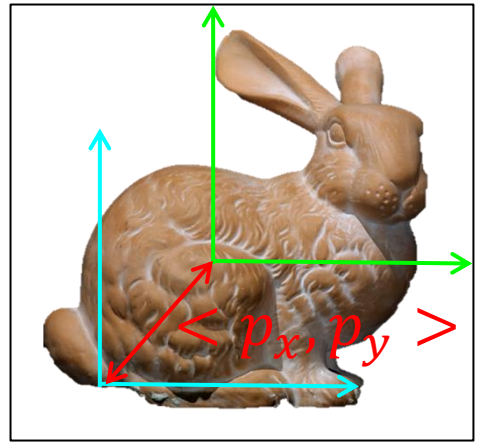


$$\begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X/Z \\ Y/Z \\ 1 \end{bmatrix} = \begin{bmatrix} f\,X/Z + p_x \\ f\,Y/Z + p_y \\ 1 \end{bmatrix}$$

Transformation from 2D to 2D, accounting for not unit focal length and origin shift

What is the Matrix Form of Perspective Projection with





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

K_{img}^{pix} K_{cam}^{img}

$K_{img}^{pix} K_{cam}^{img}$ is also called camera intrinsics matrix

2D-2D Transformation

$$\begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X/Z \\ Y/Z \\ 1 \end{bmatrix} = \begin{bmatrix} f\,X/Z + p_x \\ f\,Y/Z + p_y \\ 1 \end{bmatrix}$$

K X^{old} X^{new}

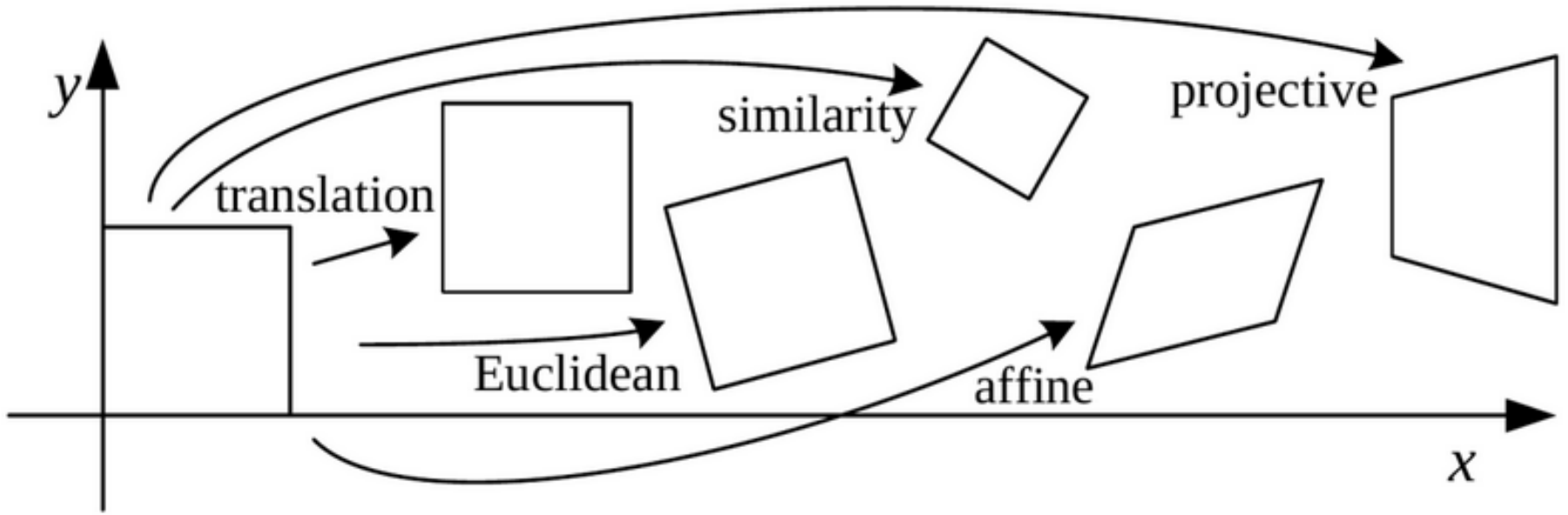


Image source A. Geiger

Euclidean Transformations

$$\mathbf{Rx} + \mathbf{t} = \mathbf{x}' \Leftrightarrow \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\intercal & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

Bold fonts denote vector

- Translation is a special case when $\mathbf{R} = \mathbf{I}$
- $\mathbf{R} \in \mathrm{SO}(2)$ forms a special group

$$\mathrm{SO}(2) = \{\mathbf{M} \mid \mathbf{M}\mathbf{M}^\intercal = \mathrm{I}, \det(\mathbf{M}) = 1\}$$

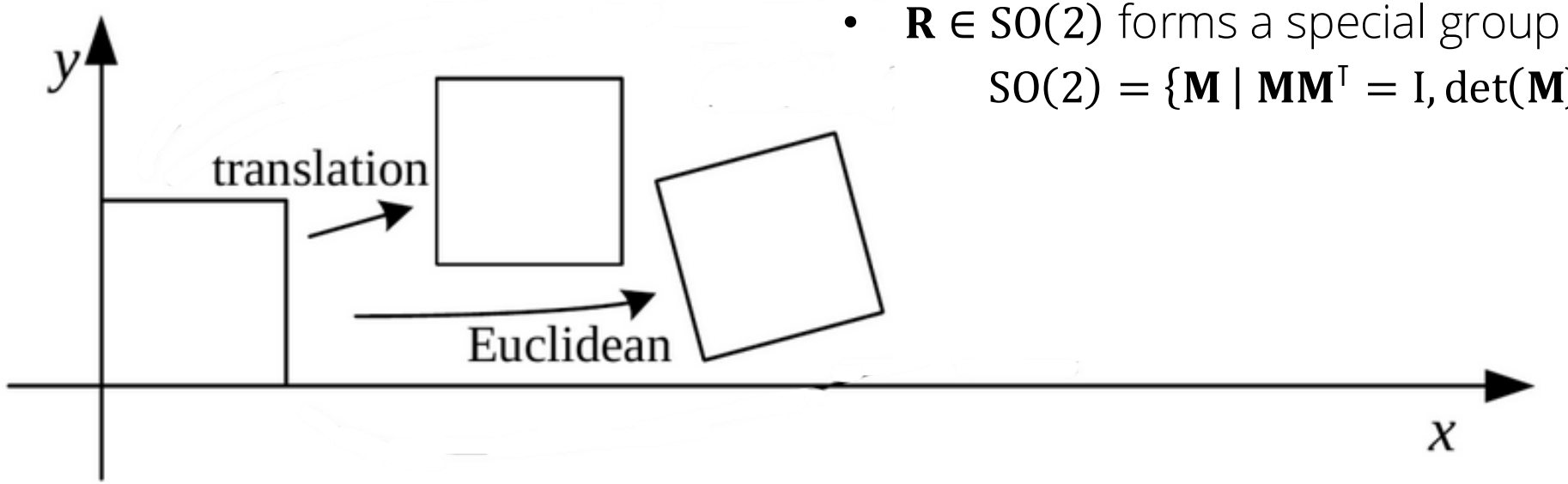


Image source A. Geiger

Similarity Transformations

$$s\mathbf{R}\mathbf{x} + \mathbf{t} = \mathbf{x}' \Leftrightarrow \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

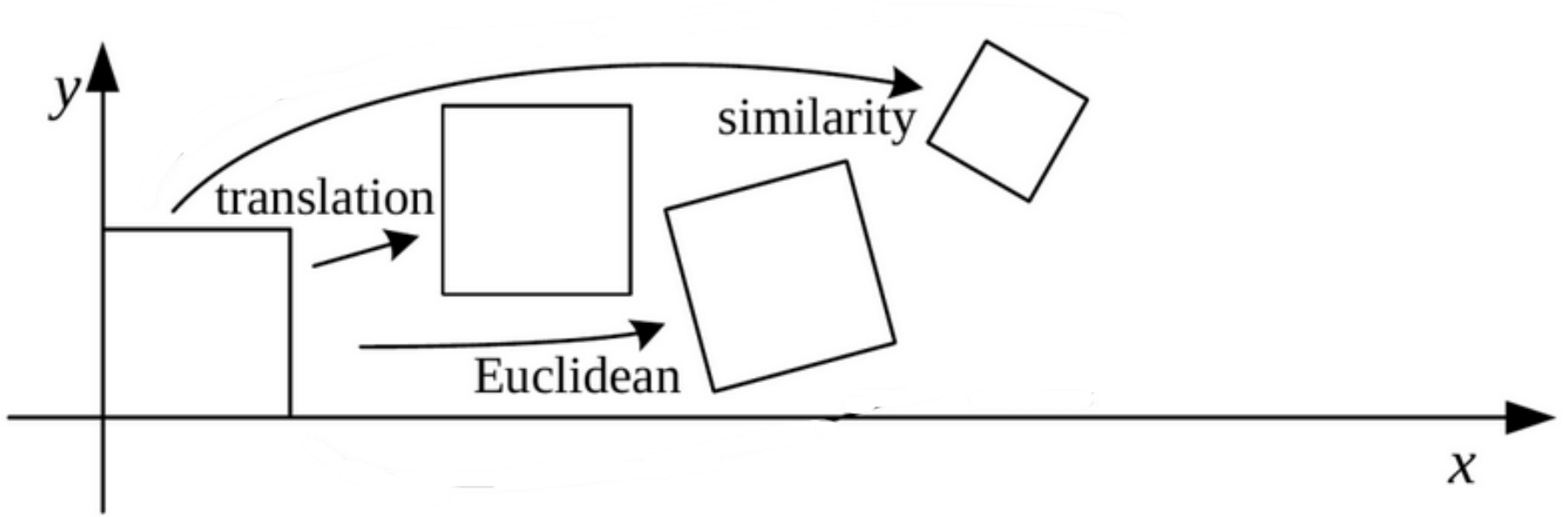


Image source A. Geiger

Affine Transformations

$$\mathbf{Ax} + \mathbf{t} = \mathbf{x}' \Leftrightarrow \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0}^{\mathsf{T}} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

- $\boldsymbol{A}$ is an arbitrary 2×2 matrix
- Parallel lines remain parallel after affine transformations

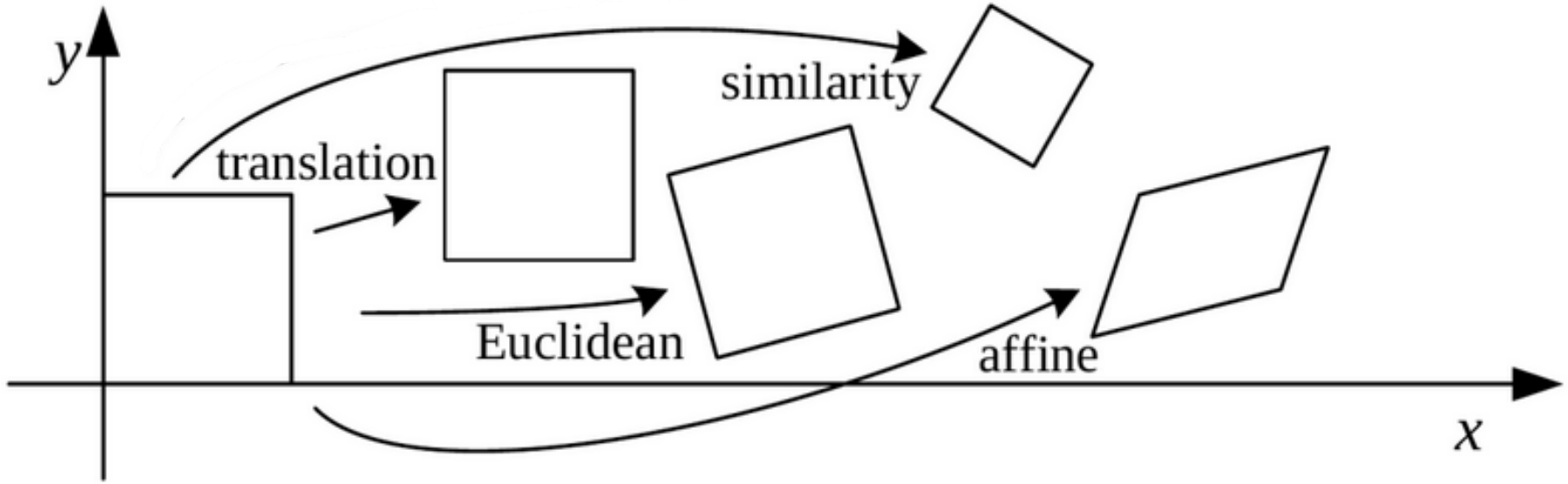


Image source A. Geiger

Projective Transformations

$$\mathbf{Hx} = \mathbf{x}' \Leftrightarrow \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

- $\mathbf{H}$ is an arbitrary $\mathbf{3 \times 3}$ matrix, also called homography matrix
- Projective transformations preserve straight lines

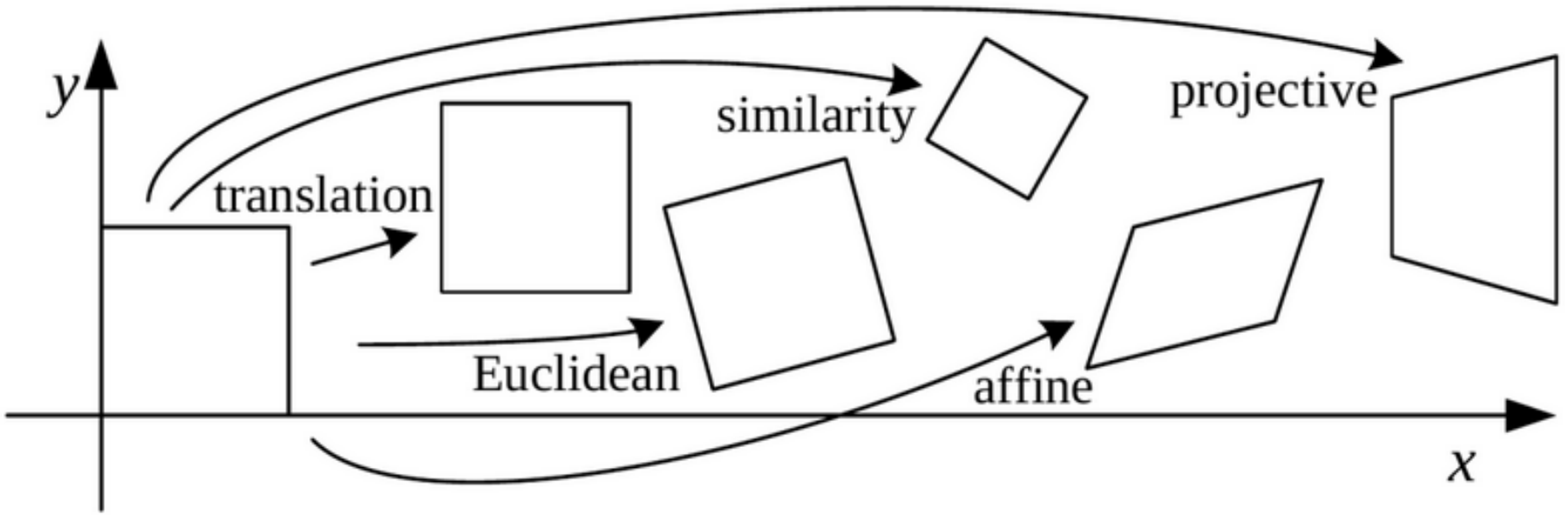
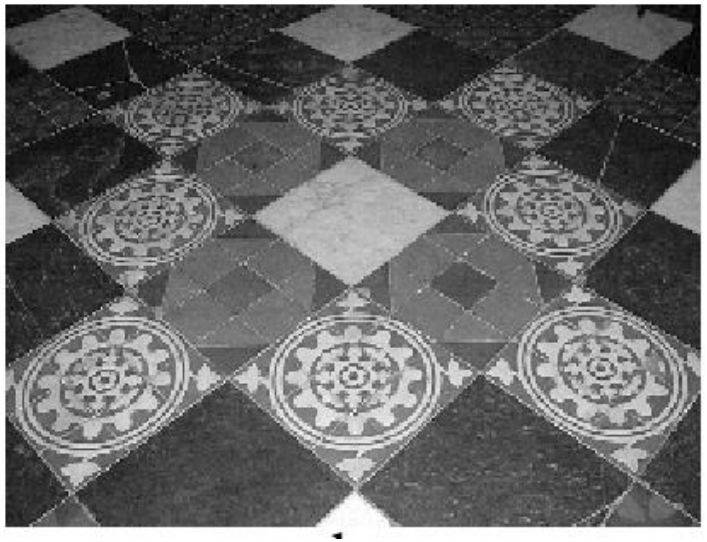


Image source A. Geiger

Application: Image Rectification with Projective Transformations

Image 1

Image 2



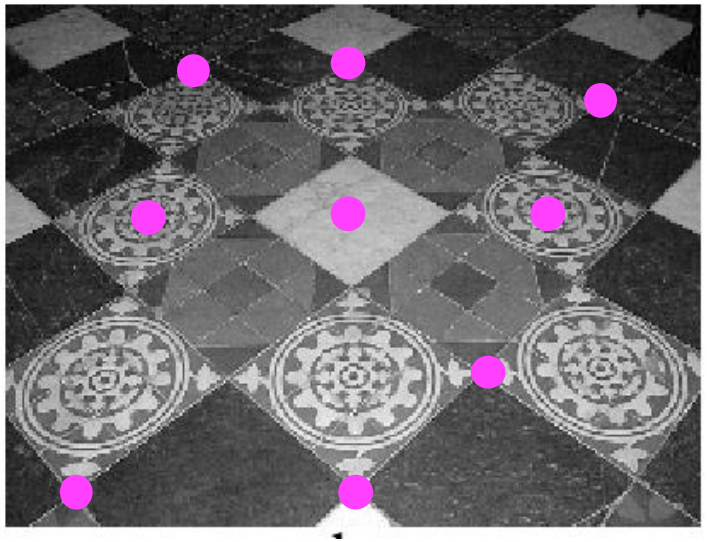
$$\mathbf{Hx} = \mathbf{x}'$$

Image source R. Harley and A. Zisserman

How Does Image Rectification Work?

Image 1

Image 2



- We have a set of points $\mathbf{x}$ from image 2 and our goal is to rectify image 2 so that it looks like image 1
- Transformation matrix $\mathbf{H}$ and transformed points are $\mathbf{x}'$ unknown
- The equation is not solvable with two unknown variables

$$\mathbf{H}\mathbf{x} = \mathbf{x}'$$

Image source R. Harley and A. Zisserman

Idea: Obtain $\mathbf{x}'$ from correspondence

Image 1

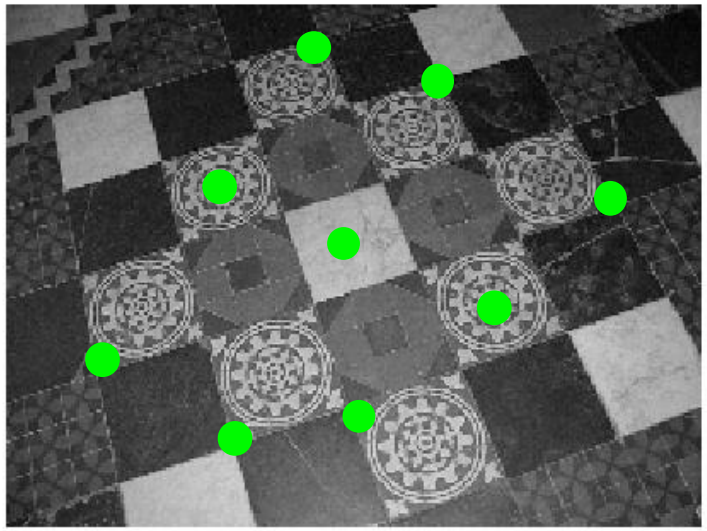
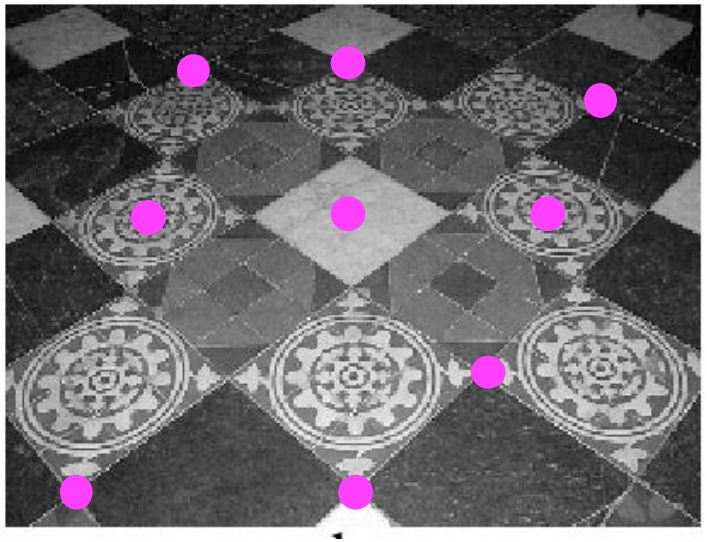


Image 2



- We have a set of points $\mathbf{x}$ from image 2 and our goal is to rectify image 2 so that it looks like image 1
- We obtain $\mathbf{x}'$ through pixel correspondence

$$\mathbf{H}\mathbf{x} = \mathbf{x}'$$

Image source R. Harley and A. Zisserman

Idea: Obtain $\mathbf{x}'$ from correspondence

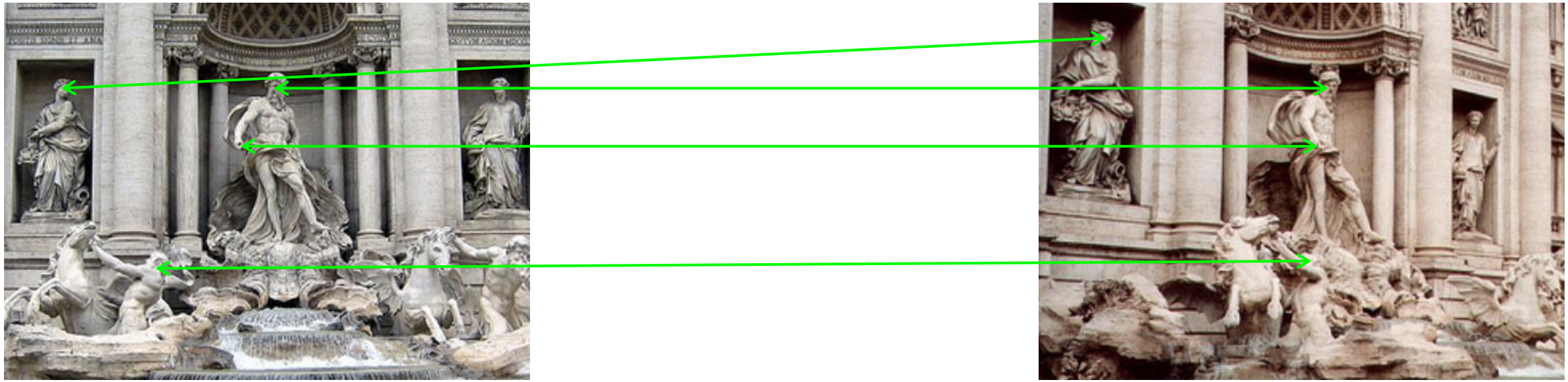
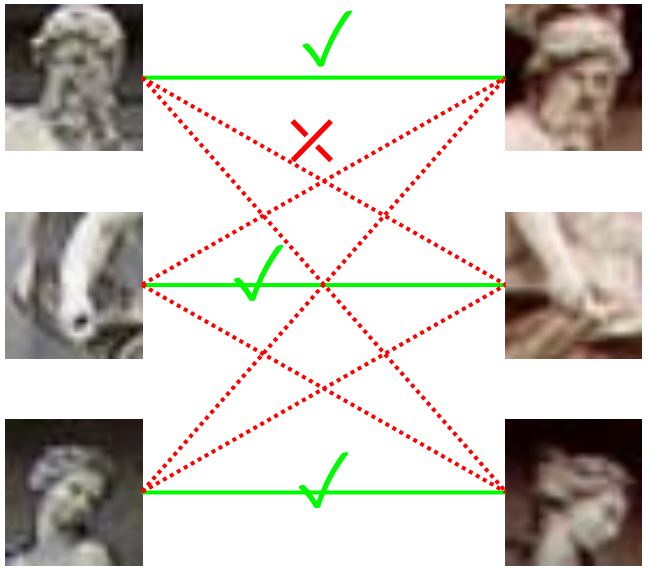


Image source Diva Sian and scgbt

Idea: Obtain $\mathbf{x}'$ from correspondence



- How to obtain pixel correspondence between two images?
  - ➢ Use a good feature extractor (e.g. SIFT / DINO.v2 …) to encode each pixel into a feature vector
  - ➢ Find the best matching between two sets of feature vectors

Image source Diva Sian and scgbt

Solve $\mathbf{H}$ given $\mathbf{x}$ and $\mathbf{x}'$

Image 1

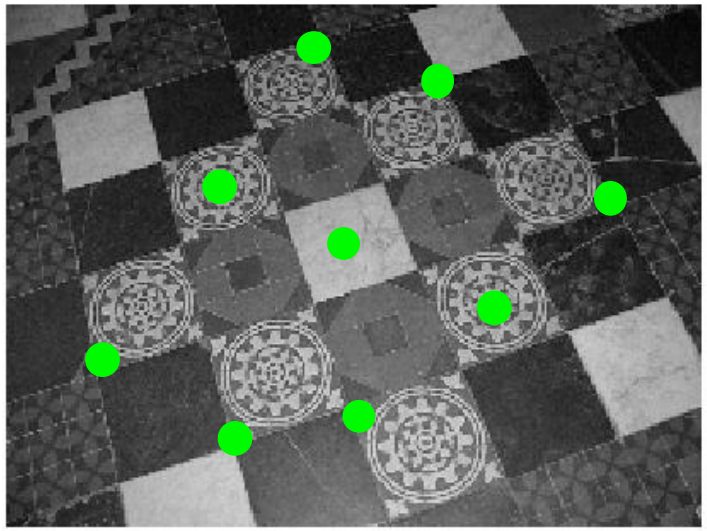
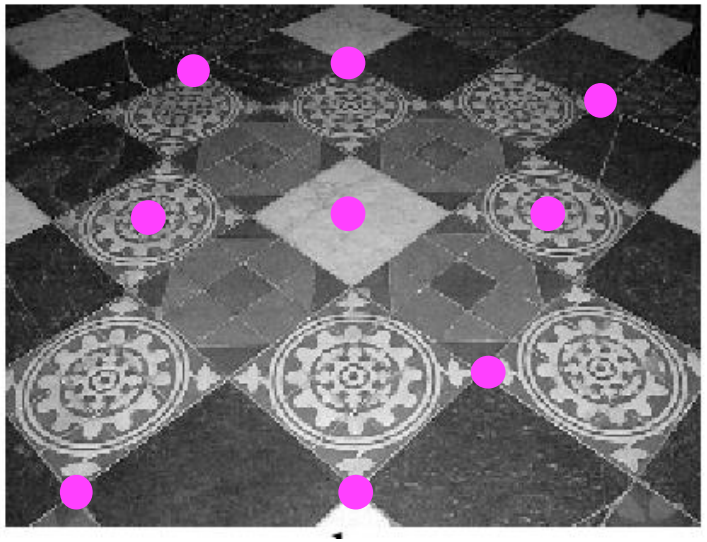


Image 2



$\mathbf{Hx} = \mathbf{x}'$

- We have a set of points $\mathbf{x}$ from image 2 and our goal is to rectify image 2 so that it looks like image 1
- We obtain $\mathbf{x}'$ through pixel correspondence
- Transformation matrix $\mathbf{H}$
- The problem is to find $\mathbf{H}$ such that $\mathbf{Hx} = \mathbf{x}'$

The Direct Linear Transformation Algorithm

- Let's start with one point correspondence: $\mathbf{H}\mathbf{x}_i = \mathbf{x}_i'$
- Note that $\mathbf{x}_i$ and $\mathbf{x}_i'$ are homogeneous coordinates, thus $\mathbf{H}\mathbf{x}_i$ and $\mathbf{x}_i'$ are in the same direction, might differ in magnitude

$$\begin{bmatrix} \boldsymbol{h}_1^\intercal \\ \boldsymbol{h}_2^\intercal \\ \boldsymbol{h}_3^\intercal \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} x_i' \\ y_i' \\ w_i' \end{bmatrix} \Rightarrow \mathbf{x}_i' \times \mathbf{H}\mathbf{x}_i = 0 \Rightarrow \begin{bmatrix} x_i' \\ y_i' \\ w_i' \end{bmatrix} \times \begin{bmatrix} \boldsymbol{h}_1^\intercal \mathbf{x}_i \\ \boldsymbol{h}_2^\intercal \mathbf{x}_i \\ \boldsymbol{h}_3^\intercal \mathbf{x}_i \end{bmatrix} = 0$$

$$\Rightarrow \begin{bmatrix} y_i' \boldsymbol{h}_3^\intercal \mathbf{x}_i - w_i' \boldsymbol{h}_2^\intercal \mathbf{x}_i \\ w_i' \boldsymbol{h}_1^\intercal \mathbf{x}_i - x_i' \boldsymbol{h}_3^\intercal \mathbf{x}_i \\ x_i' \boldsymbol{h}_2^\intercal \mathbf{x}_i - y_i' \boldsymbol{h}_1^\intercal \mathbf{x}_i \end{bmatrix} = 0 \Rightarrow \begin{bmatrix} \mathbf{0}^\intercal & -w_i' \mathbf{x}_i^\intercal & y_i' \mathbf{x}_i^\intercal \\ w_i' \mathbf{x}_i^\intercal & \mathbf{0}^\intercal & -x_i' \mathbf{x}_i^\intercal \\ -y_i' \mathbf{x}_i^\intercal & x_i' \mathbf{x}_i^\intercal & \mathbf{0}^\intercal \end{bmatrix} \begin{pmatrix} \boldsymbol{h}_1 \\ \boldsymbol{h}_2 \\ \boldsymbol{h}_3 \end{pmatrix} = 0$$

3×9 matrix 9×1 vector

Remember bold-font variable denotes vector

The Direct Linear Transformation Algorithm

- Let's start with one point correspondence: $\mathbf{H}\mathbf{x}_i = \mathbf{x}_i'$
- Note that $\mathbf{x}_i$ and $\mathbf{x}_i'$ are homogeneous coordinates, thus $\mathbf{H}\mathbf{x}_i$ and $\mathbf{x}_i'$ are in the same direction, might differ in magnitude

$$\begin{bmatrix} \boldsymbol{h}_1^\intercal \\ \boldsymbol{h}_2^\intercal \\ \boldsymbol{h}_3^\intercal \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} x_i' \\ y_i' \\ w_i' \end{bmatrix} \Rightarrow \mathbf{x}_i' \times \mathbf{H}\mathbf{x}_i = 0 \Rightarrow \begin{bmatrix} x_i' \\ y_i' \\ w_i' \end{bmatrix} \times \begin{bmatrix} \boldsymbol{h}_1^\intercal \mathbf{x}_i \\ \boldsymbol{h}_2^\intercal \mathbf{x}_i \\ \boldsymbol{h}_3^\intercal \mathbf{x}_i \end{bmatrix} = 0$$

$$\Rightarrow \begin{bmatrix} y_i' \boldsymbol{h}_3^\intercal \mathbf{x}_i - w_i' \boldsymbol{h}_2^\intercal \mathbf{x}_i \\ w_i' \boldsymbol{h}_1^\intercal \mathbf{x}_i - x_i' \boldsymbol{h}_3^\intercal \mathbf{x}_i \\ x_i' \boldsymbol{h}_2^\intercal \mathbf{x}_i - y_i' \boldsymbol{h}_1^\intercal \mathbf{x}_i \end{bmatrix} = 0 \Rightarrow \begin{bmatrix} \mathbf{0}^\intercal & -w_i' \mathbf{x}_i^\intercal & y_i' \mathbf{x}_i^\intercal \\ w_i' \mathbf{x}_i^\intercal & \mathbf{0}^\intercal & -x_i' \mathbf{x}_i^\intercal \\ -y_i' \mathbf{x}_i^\intercal & x_i' \mathbf{x}_i^\intercal & \mathbf{0}^\intercal \end{bmatrix} \begin{pmatrix} \boldsymbol{h}_1 \\ \boldsymbol{h}_2 \\ \boldsymbol{h}_3 \end{pmatrix} = 0$$

3×9 matrix with 2×9 degree of freedom

9×1 vector

By adding one constraint $\left\| \begin{pmatrix} \boldsymbol{h}_1 \\ \boldsymbol{h}_2 \\ \boldsymbol{h}_3 \end{pmatrix} \right\| = 1$, we need 4 points with correspondence

The Direct Linear Transformation Algorithm: A Solution with SVD

Objective

Find the general solution to a set of equations $\mathtt{A}\mathbf{x} = \mathbf{b}$ where $\mathtt{A}$ is an $m \times n$ matrix of rank $r < n$.

Algorithm

(i) Find the SVD $\mathtt{A} = \mathtt{U}\mathtt{D}\mathtt{V}^\mathsf{T}$, where the diagonal entries d_i of $\mathtt{D}$ are in descending numerical order.
(ii) Set $\mathbf{b}' = \mathtt{U}^\mathsf{T}\mathbf{b}$.
(iii) Find the vector $\mathbf{y}$ defined by $y_i = b'_i/d_i$ for $i = 1, \ldots, r$, and $y_i = 0$ otherwise.
(iv) The solution $\mathbf{x}$ of minimum norm $\|\mathbf{x}\|$ is $\mathtt{V}\mathbf{y}$.
(v) The general solution is $\mathbf{x} = \mathtt{V}\mathbf{y} + \lambda_{r+1}\mathbf{v}_{r+1} + \ldots + \lambda_n\mathbf{v}_n$, where $\mathbf{v}_{r+1}, \ldots, \mathbf{v}_n$ are the last $n - r$ columns of $\mathtt{V}$.

We only need the eigen vector of the least eigen value for $Ax = b$

Slide adapted from R. Harley and A. Zisserman

The Direct Linear Transformation Algorithm

Objective

Given $n \geq 4$ 2D to 2D point correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{x}'_i\}$, determine the 2D homography matrix $\mathtt{H}$ such that $\mathbf{x}'_i = \mathtt{H}\mathbf{x}_i$.

Algorithm

(i) For each correspondence $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ compute the matrix $\mathtt{A}_i$ from (4.1). Only the first two rows need be used in general.

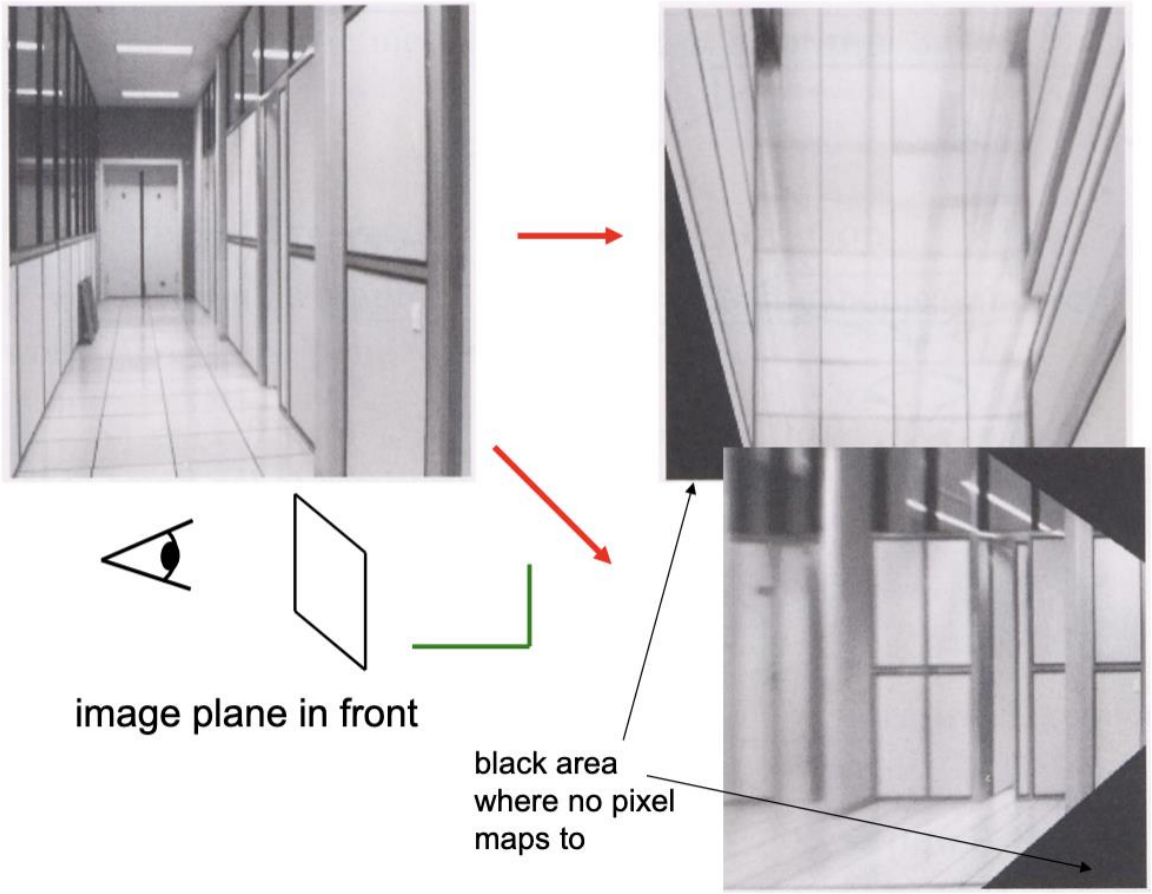
(ii) Assemble the n 2×9 matrices $\mathtt{A}_i$ into a single $2n \times 9$ matrix $\mathtt{A}$.

(iii) Obtain the SVD of $\mathtt{A}$ (section A4.4($p585$)). The unit singular vector corresponding to the smallest singular value is the solution $\mathbf{h}$. Specifically, if $\mathtt{A} = \mathtt{UDV}^\mathsf{T}$ with $\mathtt{D}$ diagonal with positive diagonal entries, arranged in descending order down the diagonal, then $\mathbf{h}$ is the last column of $\mathtt{V}$.

(iv) The matrix $\mathtt{H}$ is determined from $\mathbf{h}$ as in (4.2).

Slide adapted from R. Harley and A. Zisserman

Image Rectification: Virtual Camera Rotation with 2D Transformation

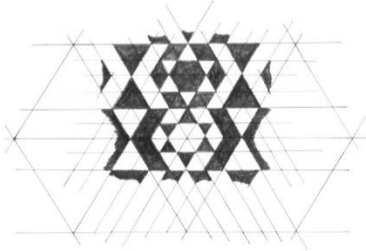


St. Lucy Altarpiece, **D. Veneziano**

Slide from Criminisi

What is the (complicated) shape of the floor pattern?

Automatically rectified floor



From Martin Kemp, *The Science of Art* *(manual reconstruction)*

Stitching Images with Image Rectification

Naïve Stitching

left on top

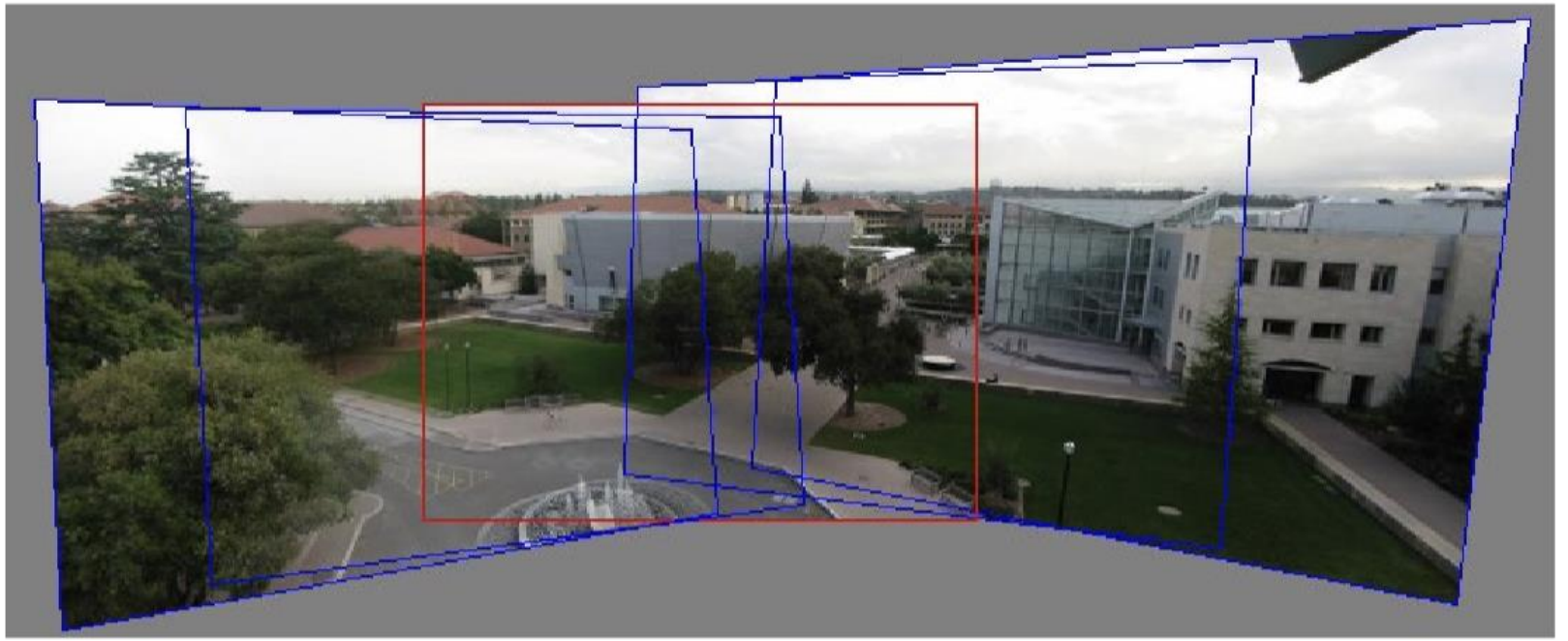
right on top

Translations are not enough to align the images

Slide credit A. Efros

Stitching Images with Image Rectification

Panoramas



1. Pick one image (red)
2. Warp the other images towards it (usually, one by one)
3. blend

Slide credit A. Efros

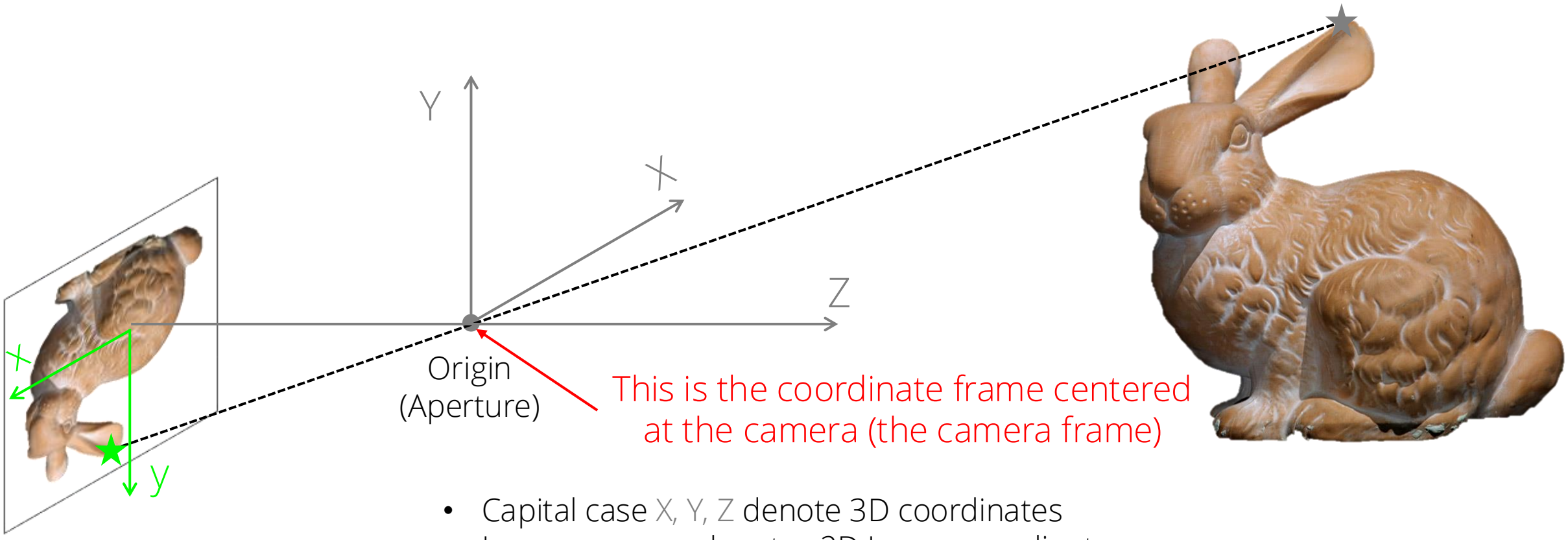
Image Panoramas

Slide credit A. Efros

Content

- Image Formation
  - ➢Pinhole camera model
  - ➢2D transformation
  - ➢3D transformation
- Geometry Reconstruction
  - ➢Reconstruction with depths
  - ➢Multi-view geometry reconstruction with camera motions and correspondence
  - ➢Epipolar geometry reconstruction

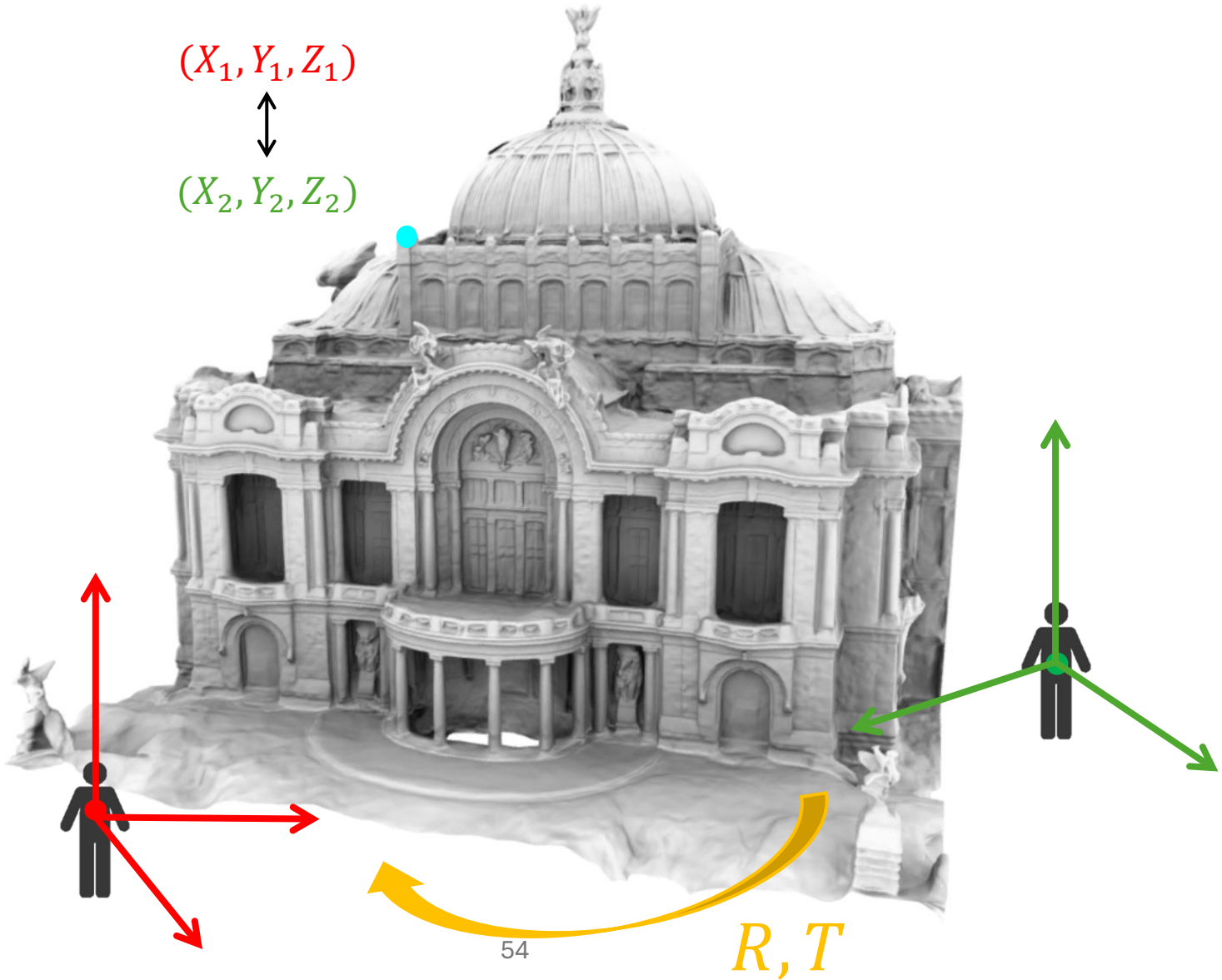
Let's Get Back to Perspective Projection.// Which Coordinate Frame did We Use?



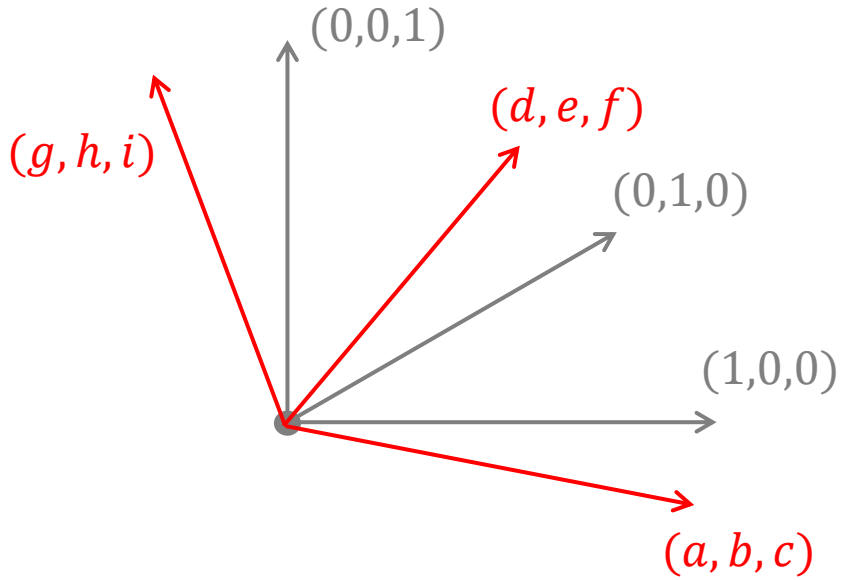
- Capital case X, Y, Z denote 3D coordinates
- Lower case x, y denotes 2D Image coordinates

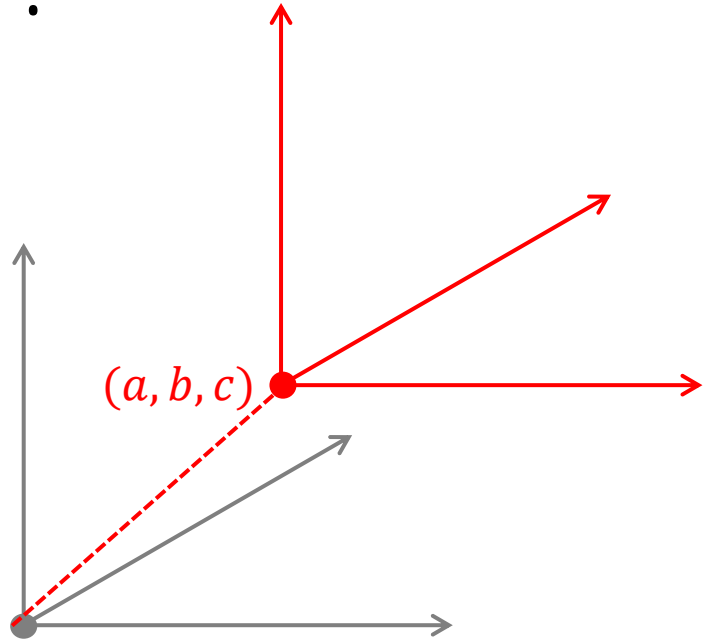
- How to derive the relation between <x, y> and <X, Y, Z> ?

What if You Have More Than 1 Camera? How to Merge 3D Points from Other Cameras?



How to Transform a 3D Point from 1 Coordinate Frame to Another?





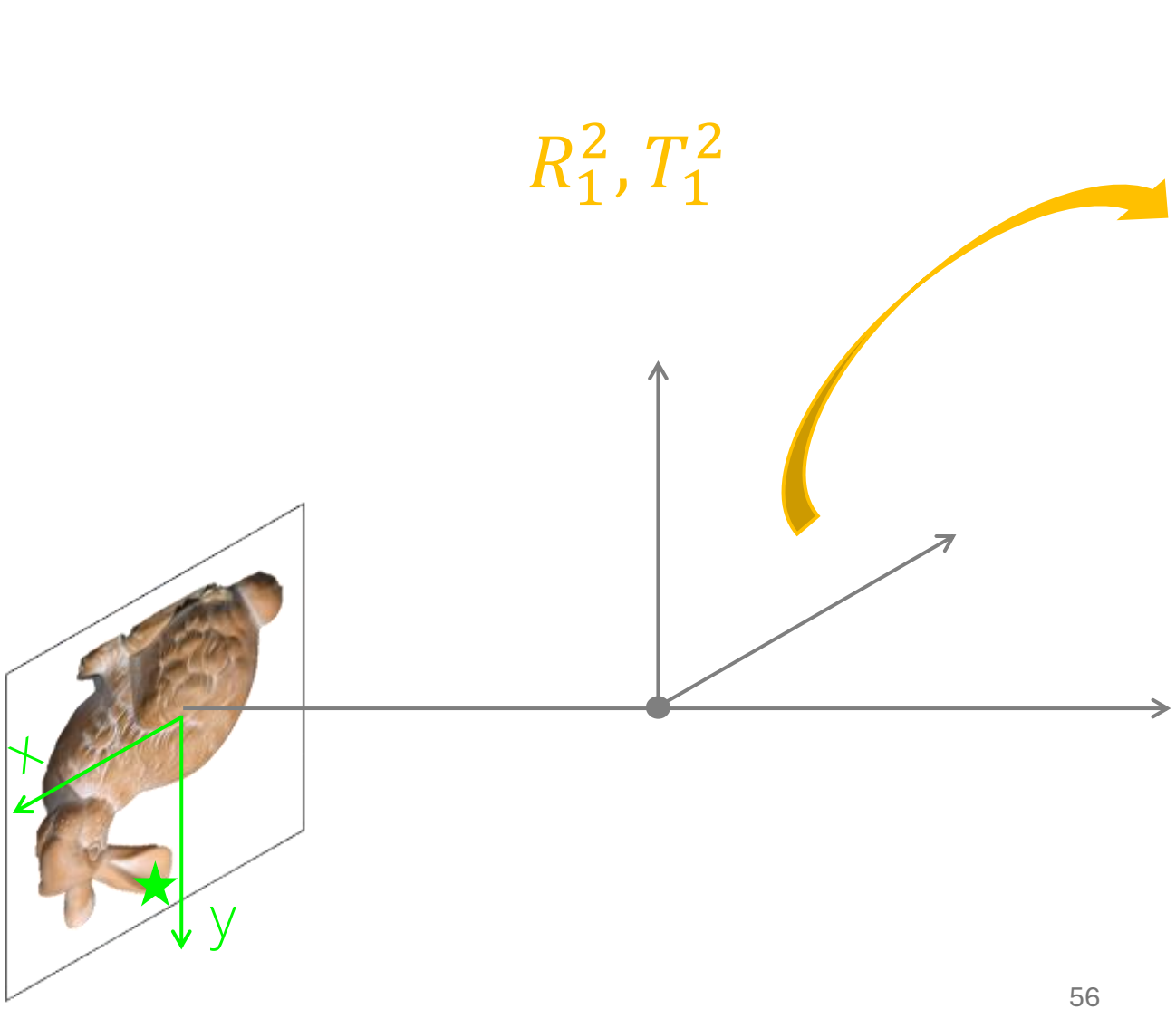
$$R = \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix} \Rightarrow \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} = R \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix}$$

$$T = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \Rightarrow \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} = \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} + \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

How to Transform a 3D Point from 1 Coordinate Frame to Another?

$$\begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} = R_1^2 \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} + T_1^2$$

R_1^2, T_1^2



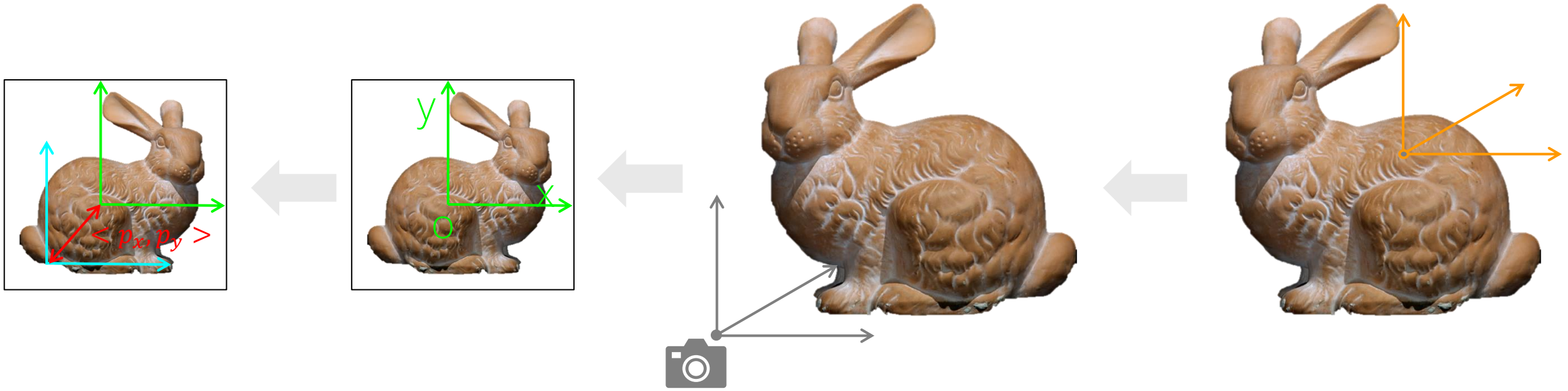
Coordinate Transformation with Homogeneous Coordinates

- We can express 3D coordinate transformation in a matrix form using homogeneous coordinates

$$\begin{bmatrix} X_{cam_2} \\ Y_{cam_2} \\ Z_{cam_2} \\ w \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{R}_{\boldsymbol{cam_1}}^{\boldsymbol{cam_2}} & \boldsymbol{T}_{\boldsymbol{cam_1}}^{\boldsymbol{cam_2}} \\ \mathbf{0} & 1 \end{bmatrix}}_{M} \begin{bmatrix} X_{cam_1} \\ Y_{cam_1} \\ Z_{cam_1} \\ 1 \end{bmatrix}$$

- M is called extrinsic matrix
- In fact, M forms a special group $SE(3)$

Let's Put Everything Together



$$\begin{bmatrix} x_{pix} \\ y_{pix} \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{R_w^c} & \boldsymbol{T_w^c} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

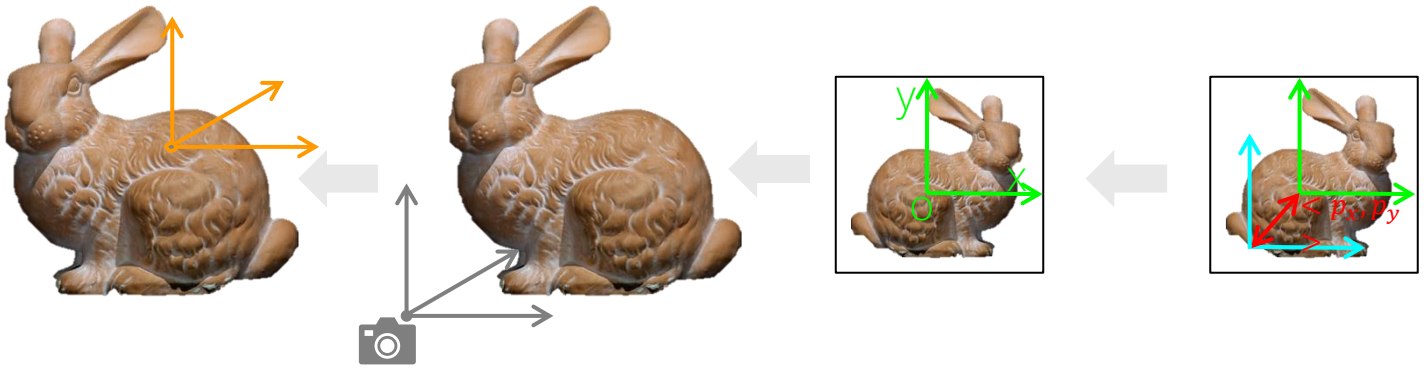
K_{img}^{pix} K_{cam}^{img} M_{world}^{cam}

Content

- Image Formation
  - Pinhole camera model
  - 2D transformation
  - 3D transformation
- Geometry Reconstruction
  - Reconstruction with depths
  - Multi-view geometry reconstruction with camera motions and correspondence
  - Epipolar geometry reconstruction

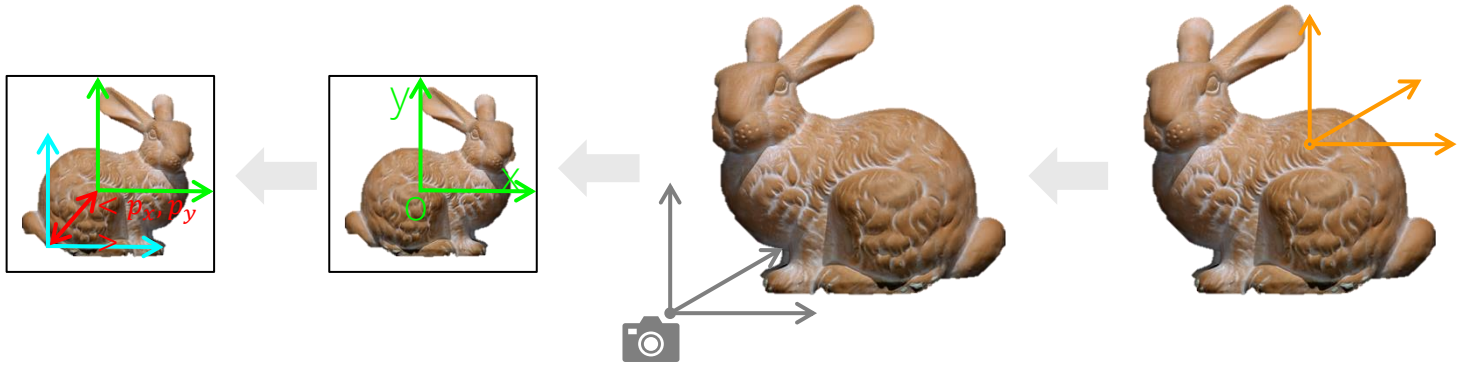
3D Model Reconstruction from 2D Images

2D-to-3D reconstruction



$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R_w^c} & \boldsymbol{T_w^c} \\ \mathbf{0} & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x_{pix} \\ y_{pix} \\ 1 \end{bmatrix}$$

3D-to-2D perspective projection



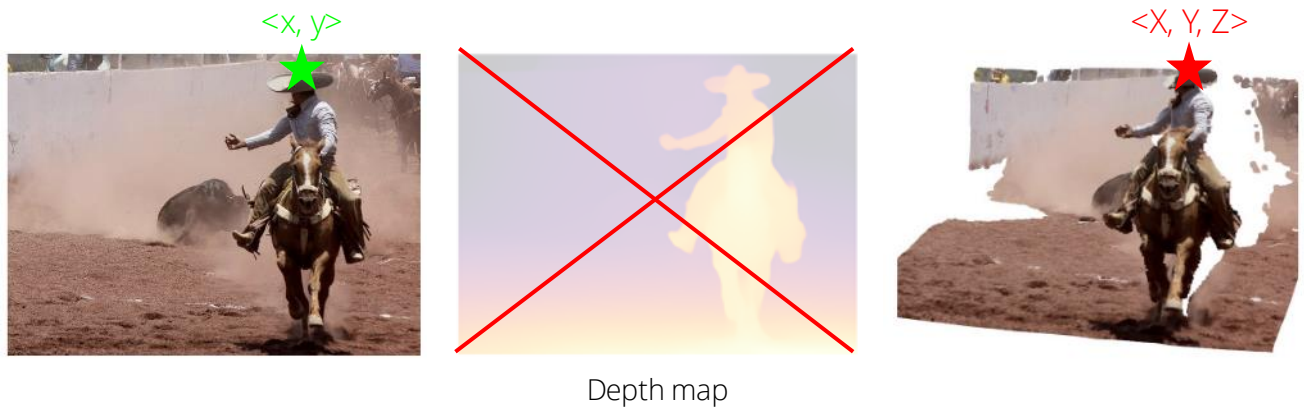
$$\begin{bmatrix} x_{pix} \\ y_{pix} \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{R_w^c} & \boldsymbol{T_w^c} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

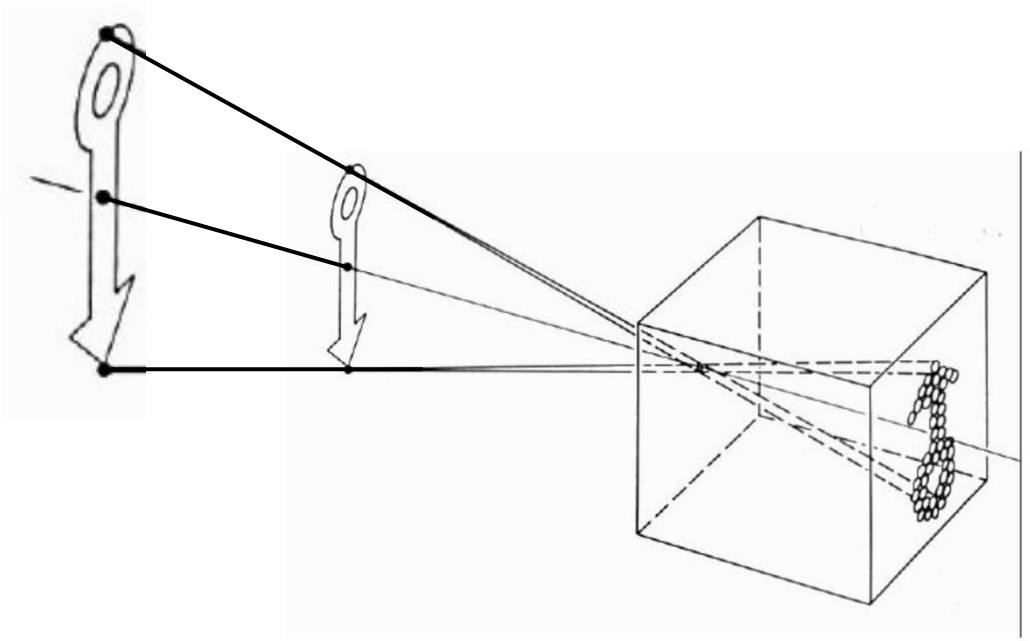
Image source https://3dvision.princeton.edu/courses/SFMedu/
Image credit S. Tulsiani

Content

- Image Formation
  - Pinhole camera model
  - 2D transformation
  - 3D transformation
- Geometry Reconstruction
  - Reconstruction with depths
  - Multi-view geometry reconstruction with camera motions and correspondence
  - Epipolar geometry reconstruction

We Said to Reconstruct 3D Models from a 2D Image with the Depth Map. What if We Don't have the Depth Map?





2D to 3D is inherently ambiguous:

$$x = PX = PHH^{-1}X = \tilde{P}\tilde{X}$$

Resolve Ambiguous 3D Model Reconstruction with Multiview 2D Images

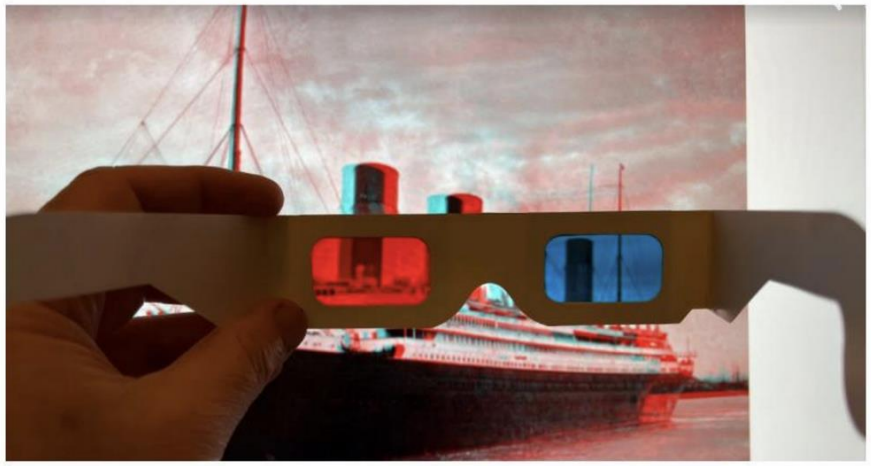
Stereoscope

Image credit A. Efors

Image source https://3dvision.princeton.edu/courses/SFMedu/
Image credit S. Tulsiani

Resolve Ambiguous 3D Model Reconstruction with Multiview 2D Images

(a)



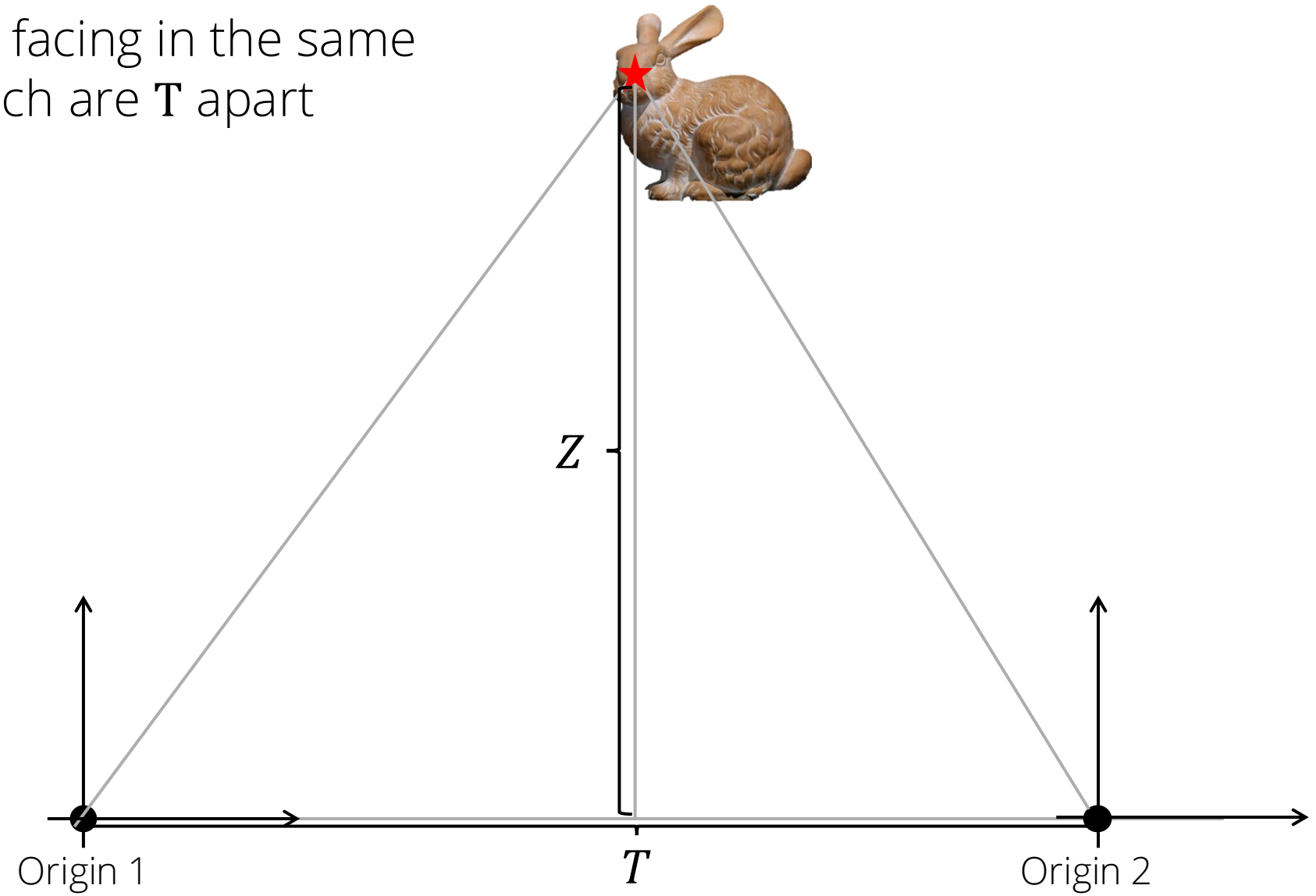
(b)

Figure 1.1: (a) Stereo anaglyph of the ocean liner, the Titanic [McManus2022]. The red image shows the right eye's view, and cyan the left eye's view. When viewed through stereo red/cyan stereo glasses, as in (b), the cyan contrast appears in the left eye image and the red variations appear to the right eye, creating a the perception of 3d.

Slide credit A. Efros

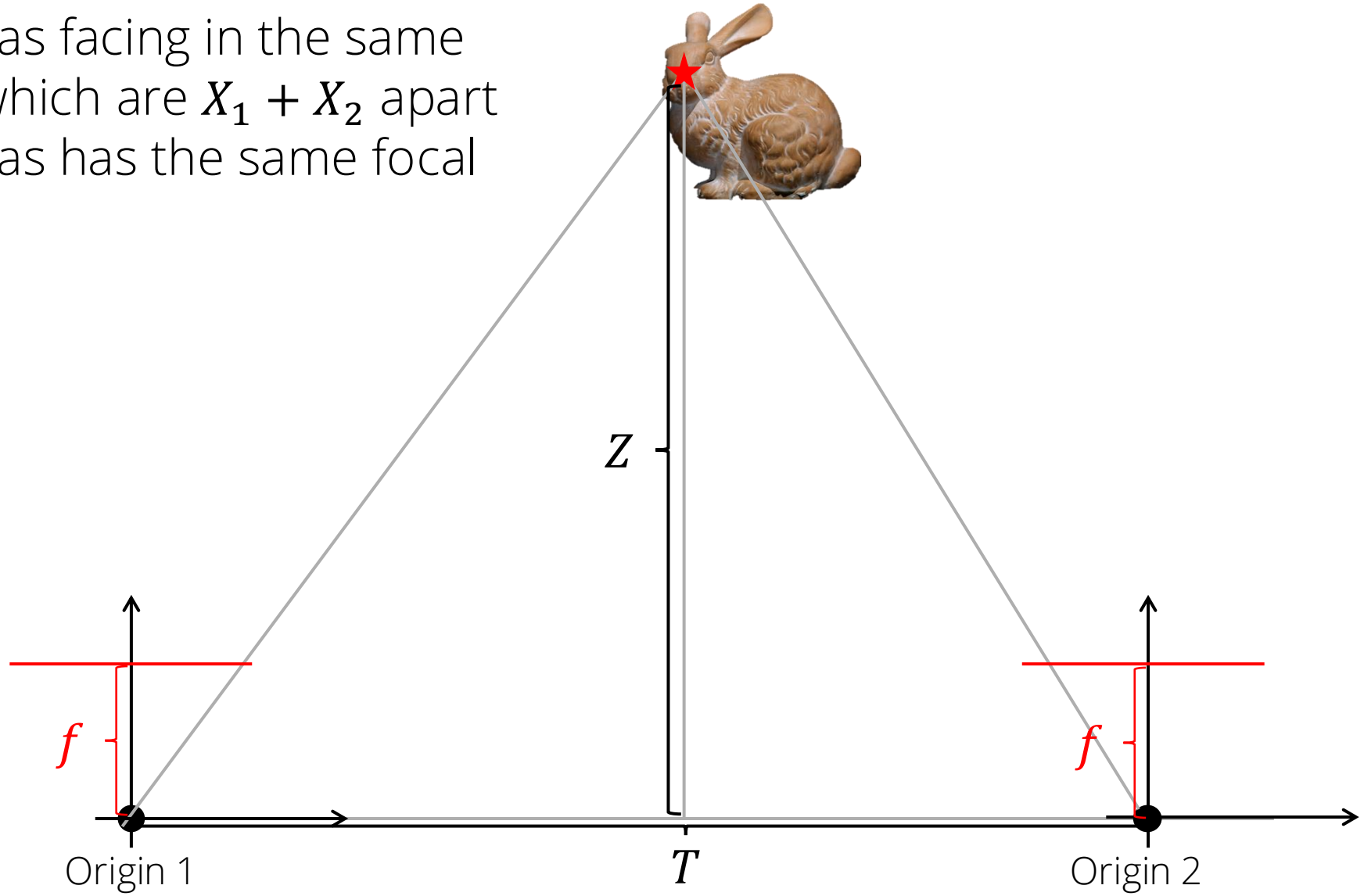
Geometry for a Simple Stereo System

- Two cameras facing in the same direction, which are $\mathbf{T}$ apart



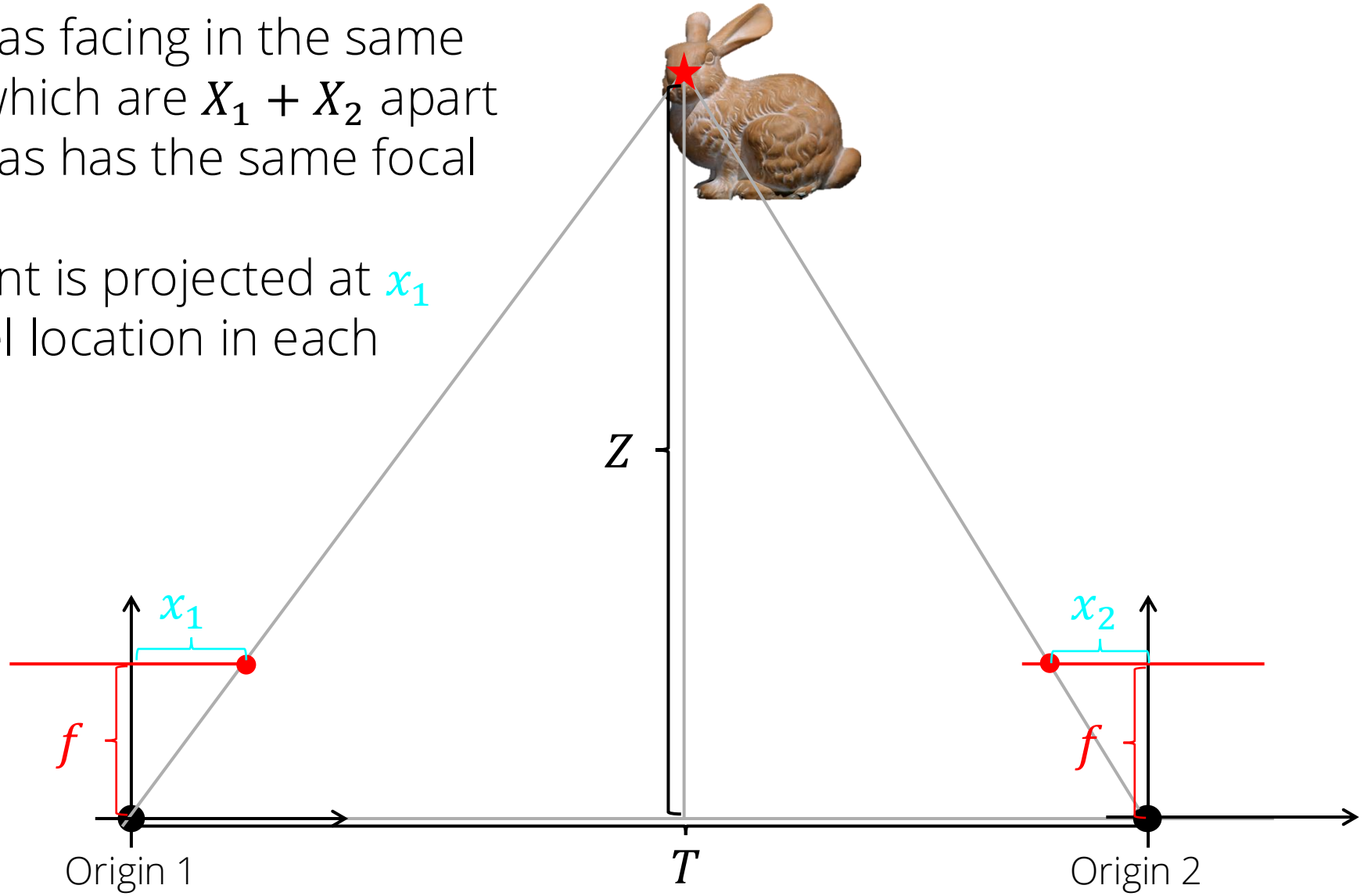
Geometry for a Simple Stereo System

- Two cameras facing in the same direction, which are $X_1 + X_2$ apart
- Two cameras has the same focal length



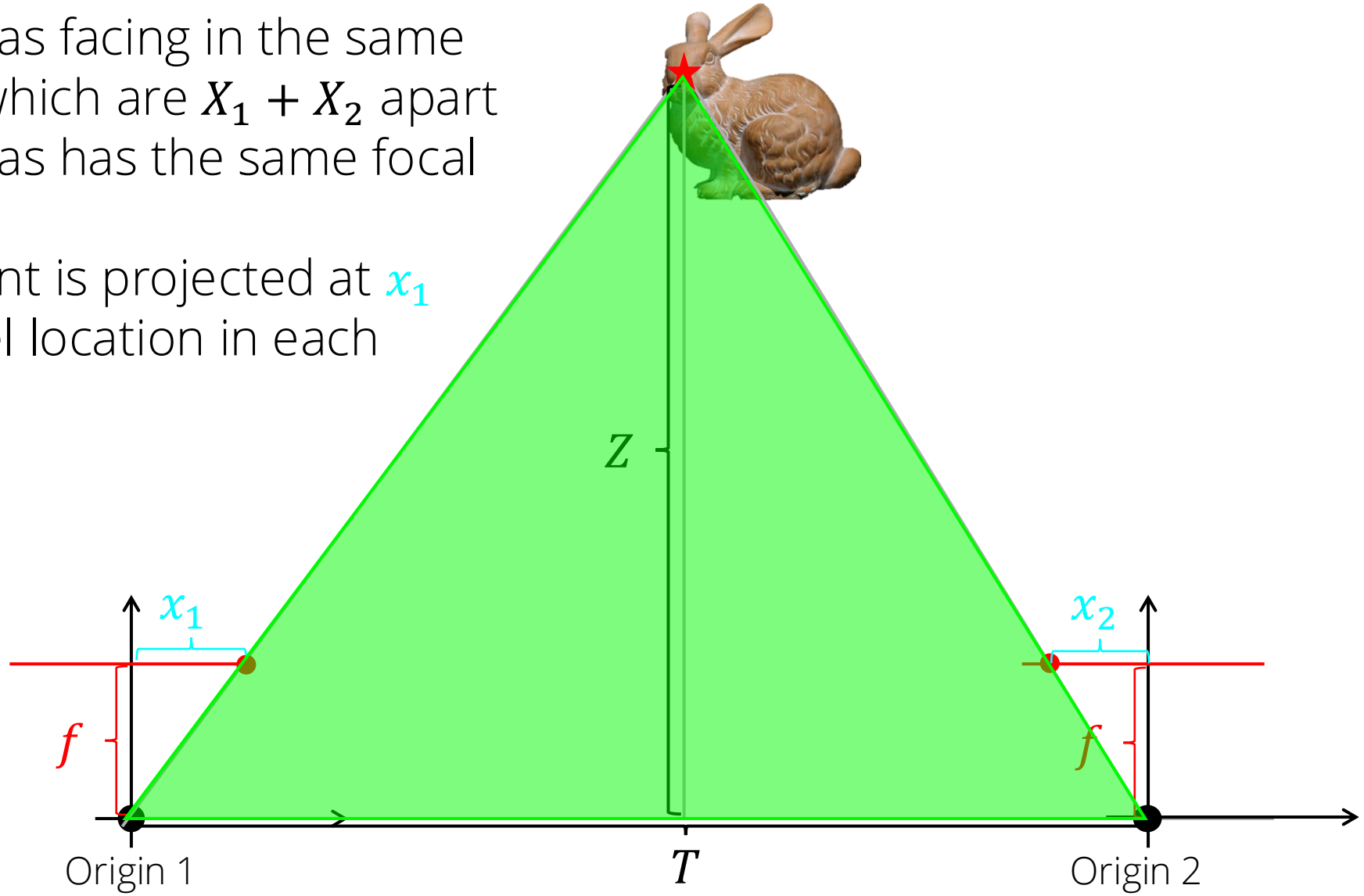
Geometry for a Simple Stereo System

- Two cameras facing in the same direction, which are $X_1 + X_2$ apart
- Two cameras has the same focal length
- The 3D point is projected at x_1 and x_1 pixel location in each camera



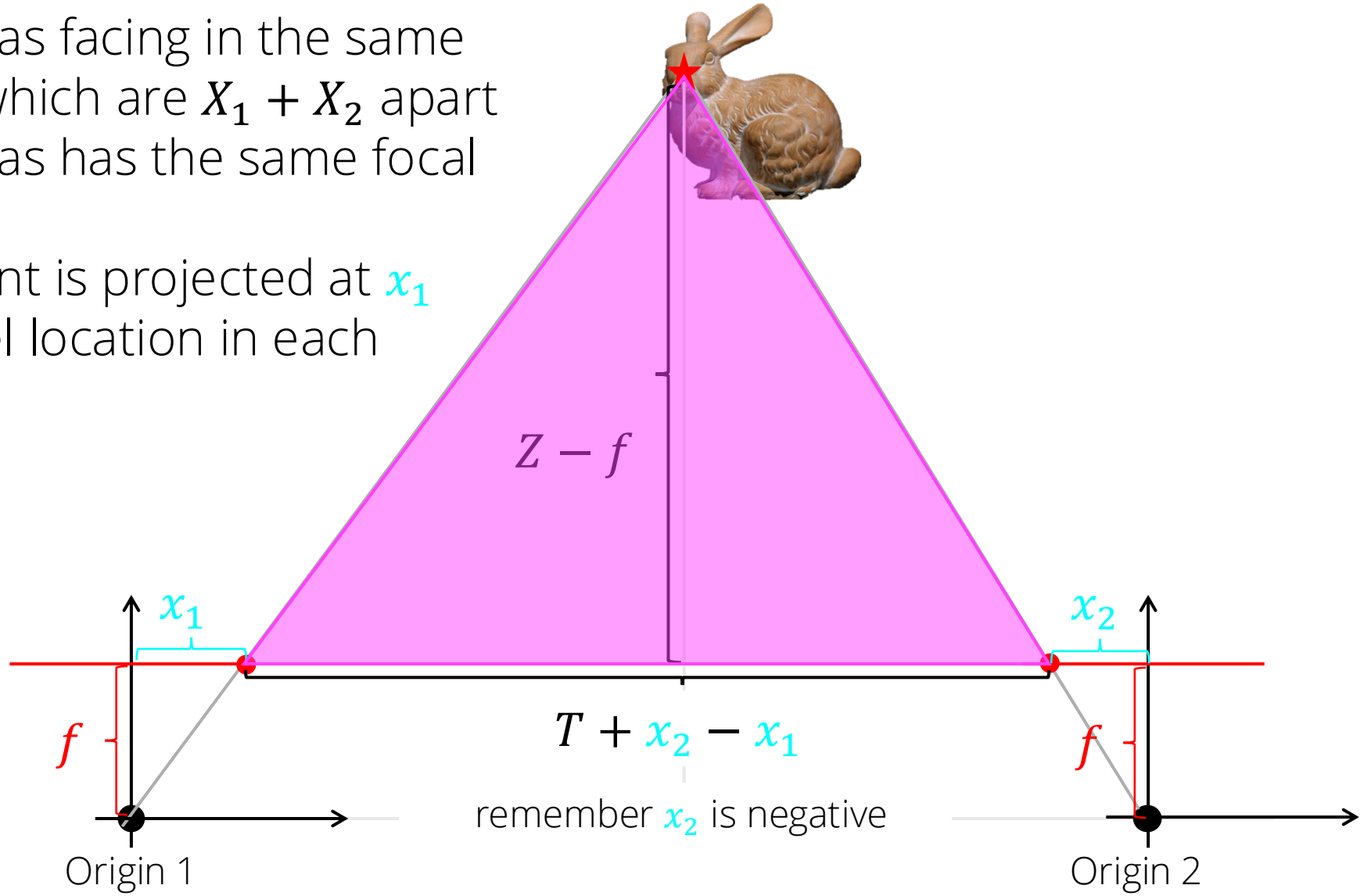
Geometry for a Simple Stereo System

- Two cameras facing in the same direction, which are $X_1 + X_2$ apart
- Two cameras has the same focal length
- The 3D point is projected at x_1 and x_1 pixel location in each camera



Geometry for a Simple Stereo System

- Two cameras facing in the same direction, which are $X_1 + X_2$ apart
- Two cameras has the same focal length
- The 3D point is projected at x_1 and x_1 pixel location in each camera

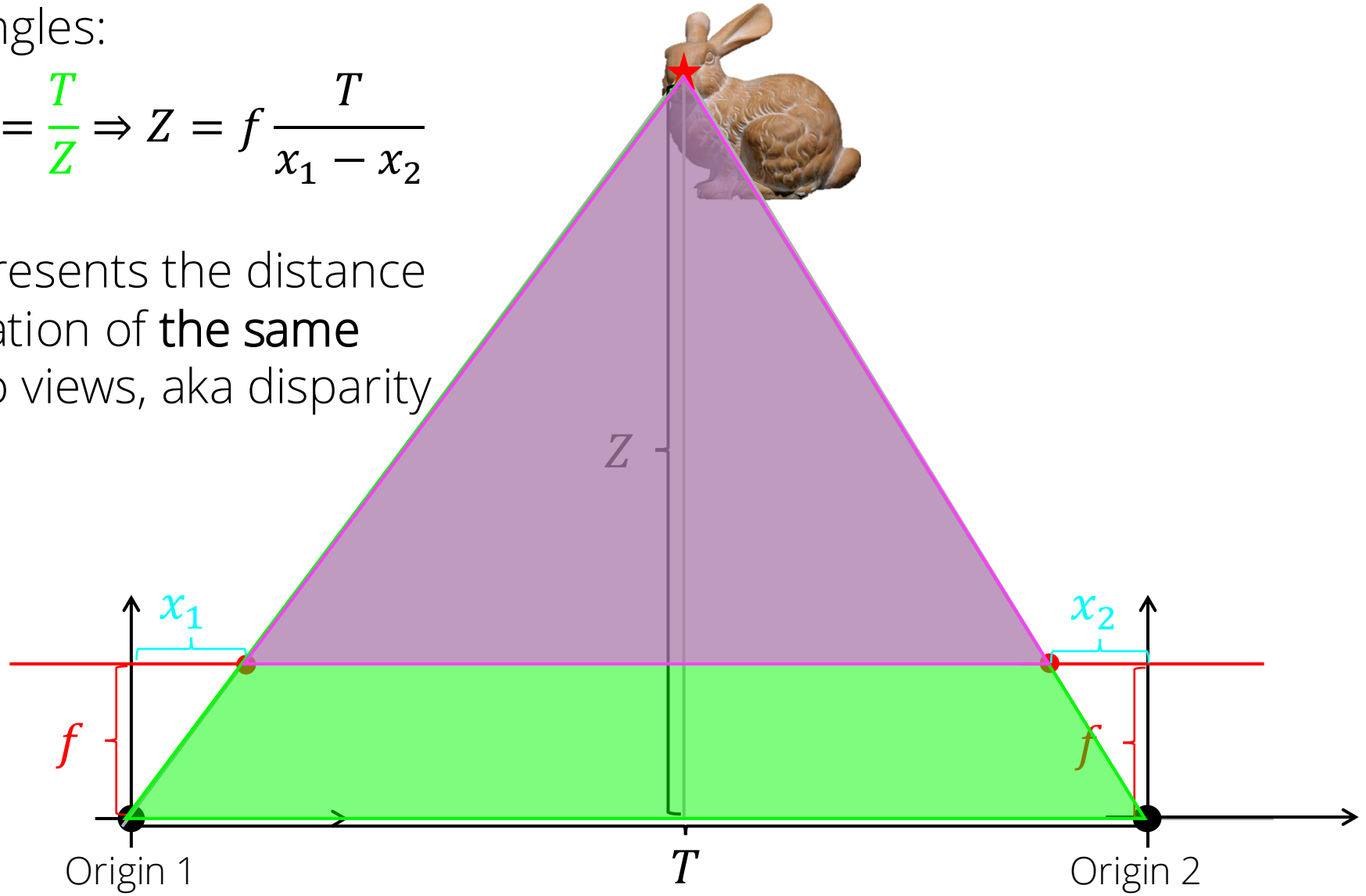


Geometry for a Simple Stereo System

- Similar triangles:

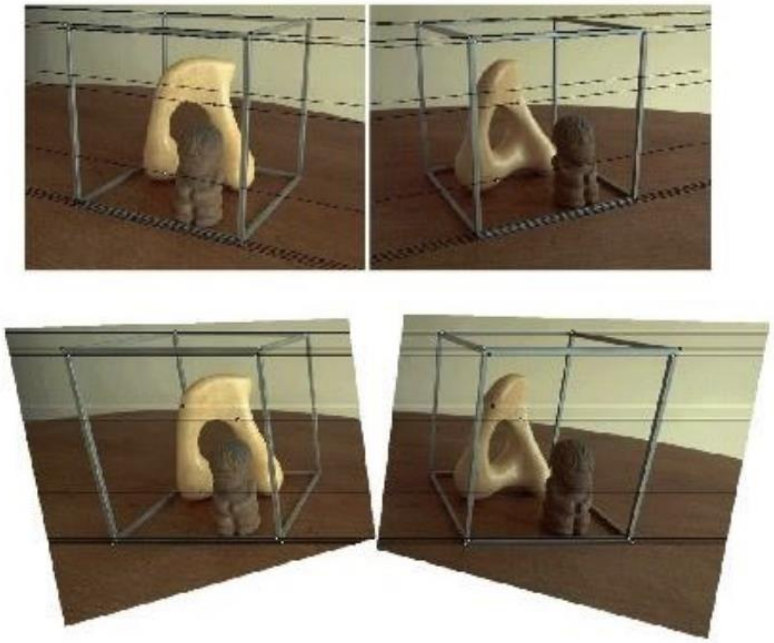
$$\frac{T + x_2 - x_1}{Z - f} = \frac{T}{Z} \Rightarrow Z = f\frac{T}{x_1 - x_2}$$

- $x_1 - x_2$ represents the distance of pixel location of **the same point** in two views, aka disparity



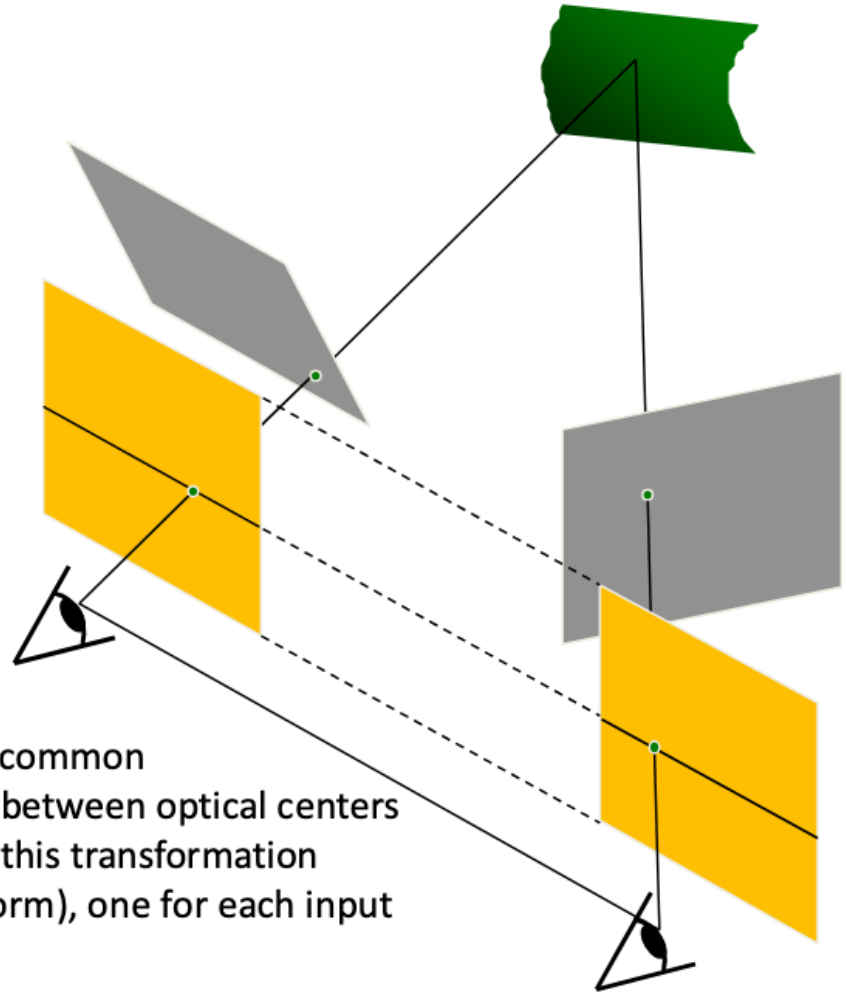
What if the Two Cameras Are Not Parallel?

Rectification example



- We'll talk about how to obtain the homography projection that makes 2 cameras parallel next

Stereo image rectification

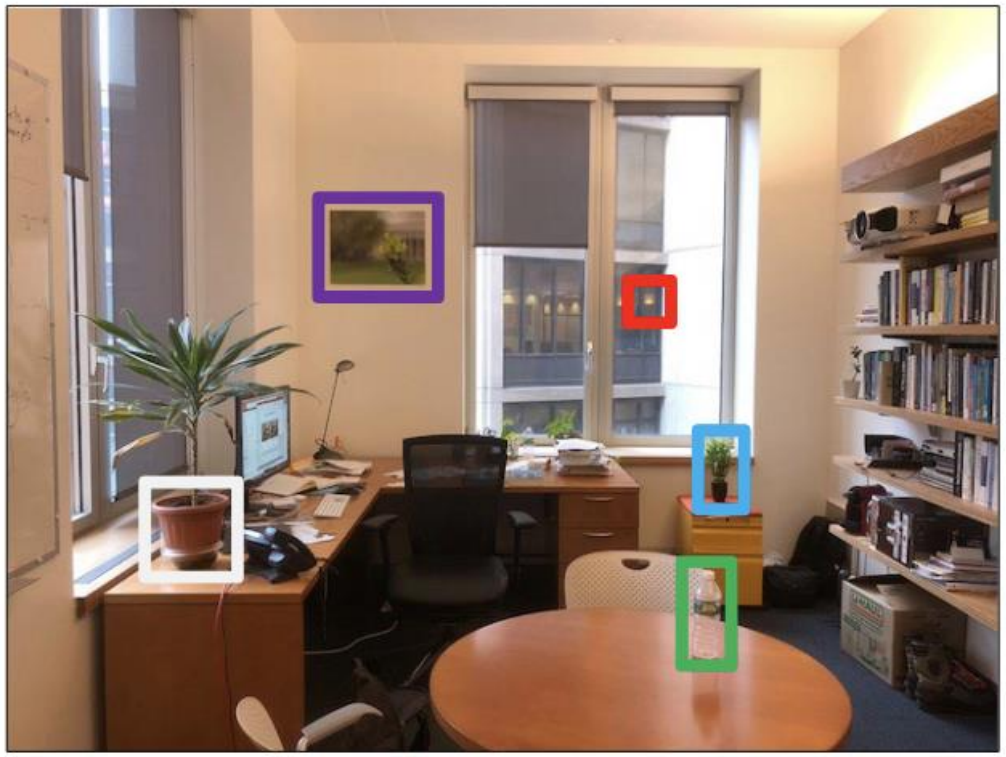


- Reproject image planes onto a common
- plane parallel to the line between optical centers
- Pixel motion is horizontal after this transformation
- Two homographies (3x3 transform), one for each input image reprojection

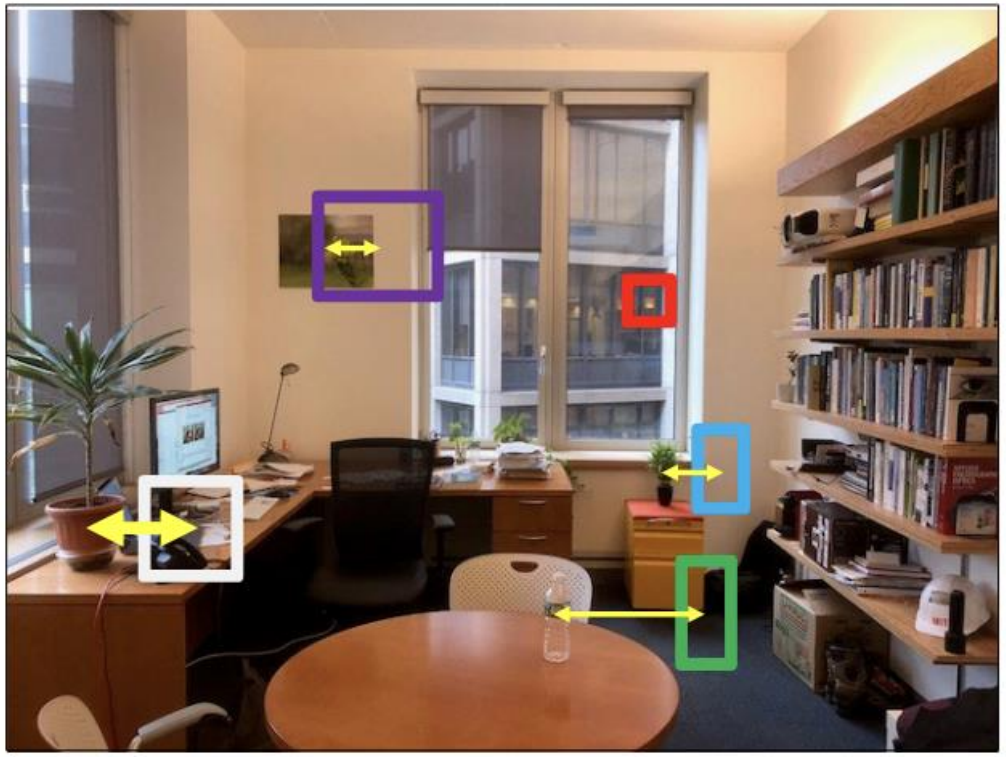
•C. Loop and Z. Zhang. Computing Rectifying Homographies for Stereo Vision. IEEE Conf. Computer Vision and Pattern Recognition, 1999.

Slide credit J. Malik

Disparity Map: the distance of pixel location of the same point in two views



Left image



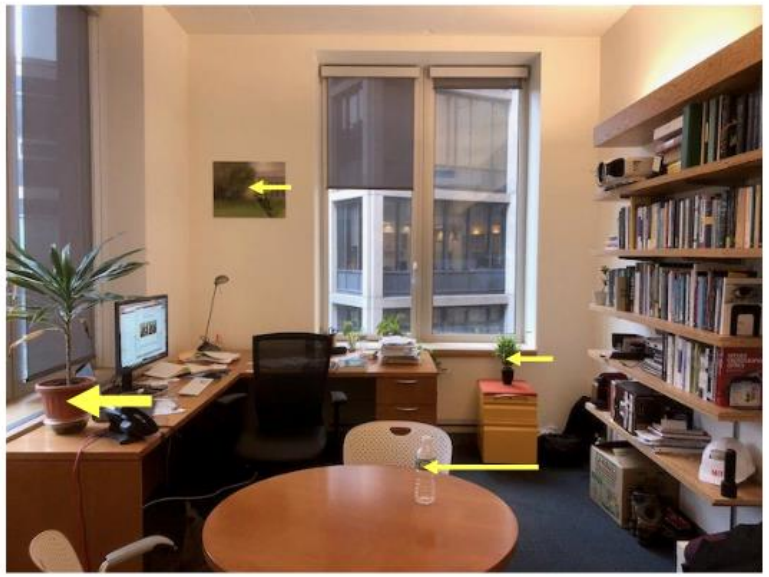
Right image

Slide credit A. Efros

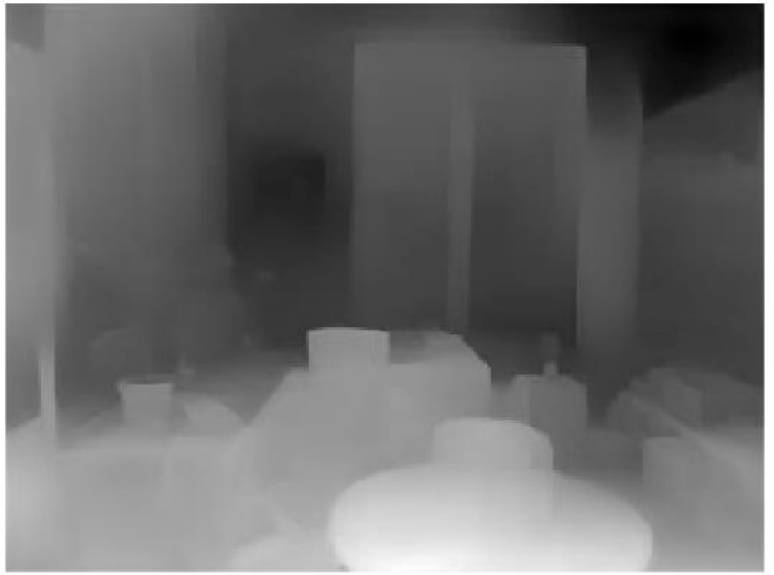
Disparity Map: the distance of pixel location of the same point in two views

I(x,y)

I'(x,y) = I(x+D(x,y), y)



D(x,y)



- Distance derived from the disparity map

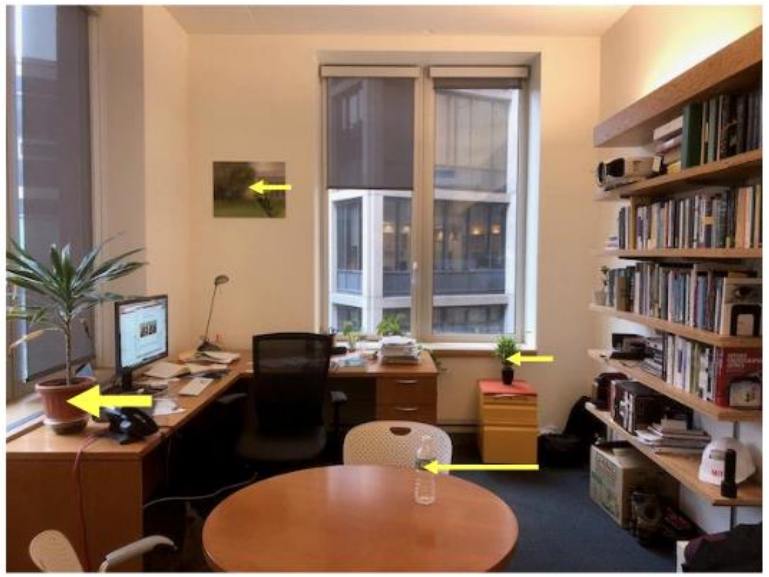
$$Z(x,y) = \frac{a}{D(x,y)}$$

Slide credit A. Efros

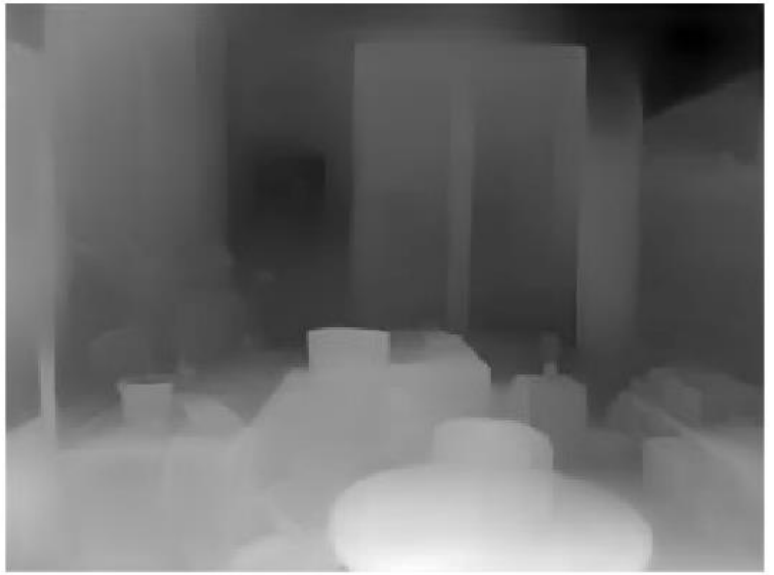
Disparity Map: the distance of pixel location of the same point in two views

I(x,y)

I'(x,y) = I(x+D(x,y), y)



D(x,y)



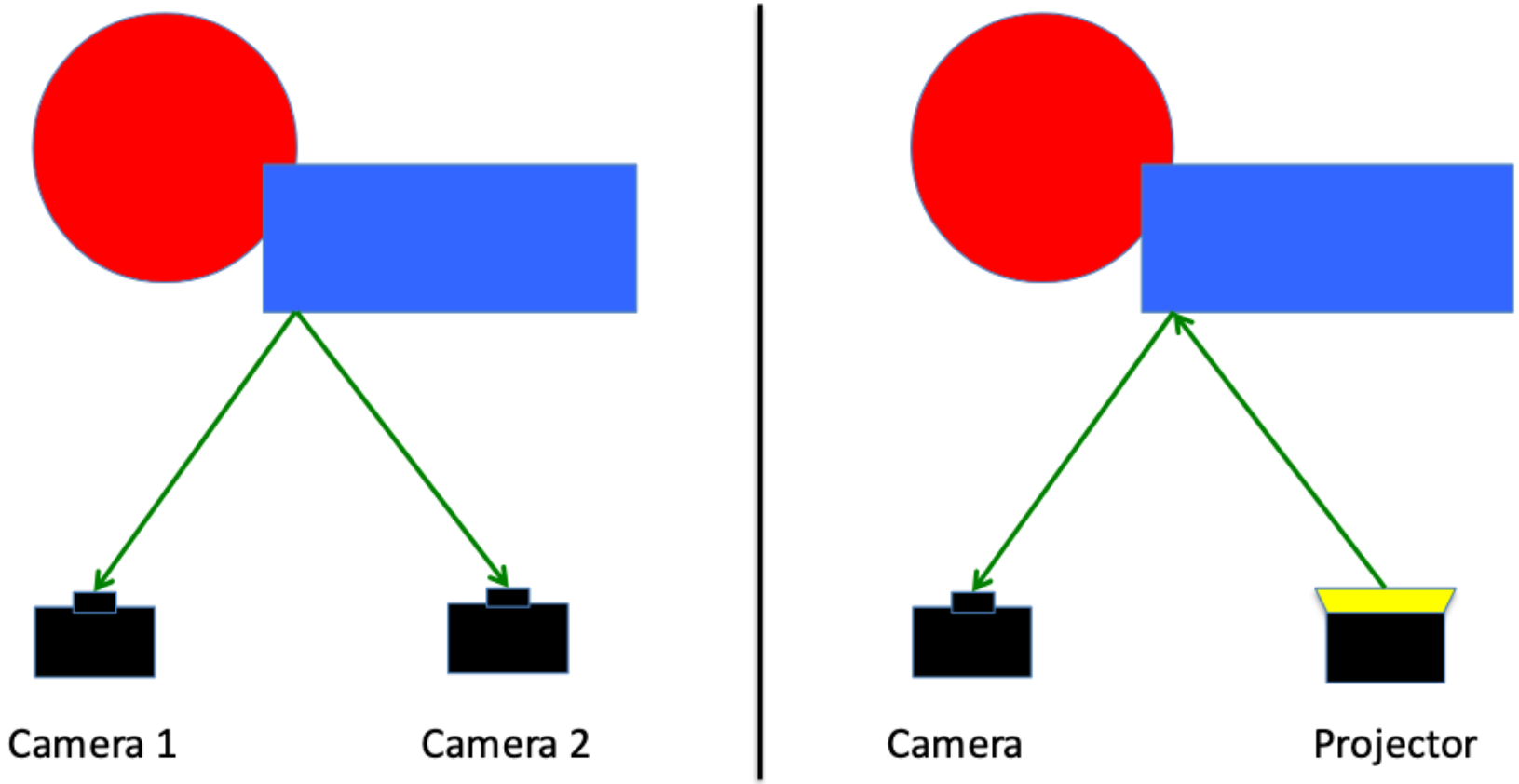
- Disparity map is derived from pixel correspondence
- How to derive pixel correspondence?

- Distance derived from the disparity map

$$Z(x,y) = \frac{a}{D(x,y)}$$

Slide credit A. Efros

Depth from Triangulation



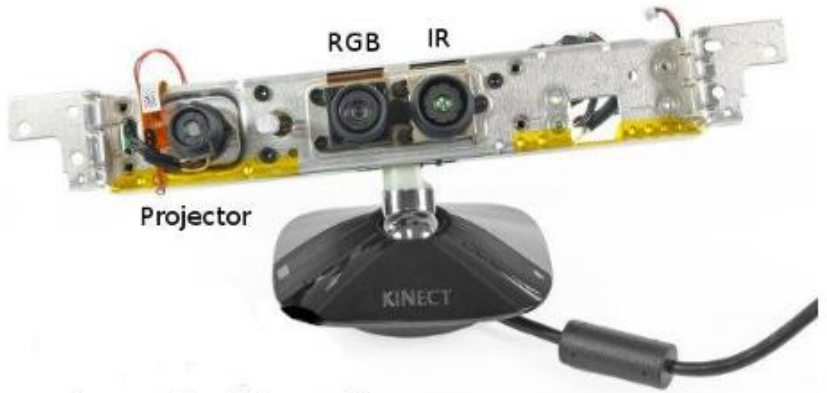
Passive Stereopsis

Active Stereopsis

Active sensing simplifies the problem of estimating point correspondences

Slide credit J. Malik

Range Sensors



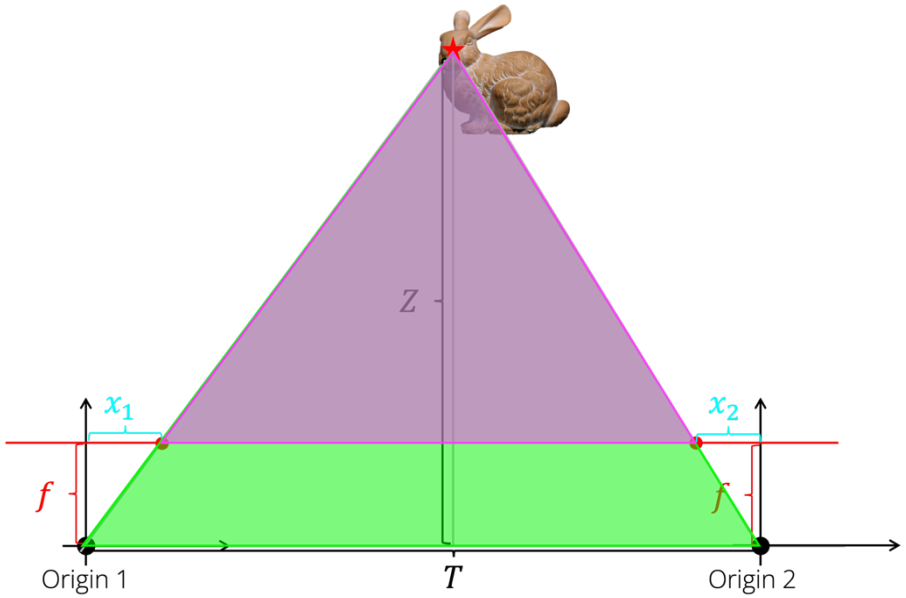
primesense sensor (used in Kinect)

Velodyne LIDAR Sensor

http://www.primesense.com/, http://www.ifixit.com/,
http://mirror.umd.edu/roswiki/kinect_calibration(2f)technical.html
http://velodynelidar.com/lidar/lidar.aspx

Slide credit J. Malik

Let's Take a Short Break



$$Z = f\frac{T}{x_1 - x_2}$$

- $x_1 - x_2$ represents the disparity, which requires pixel correspondence
- T represents the relative camera pose

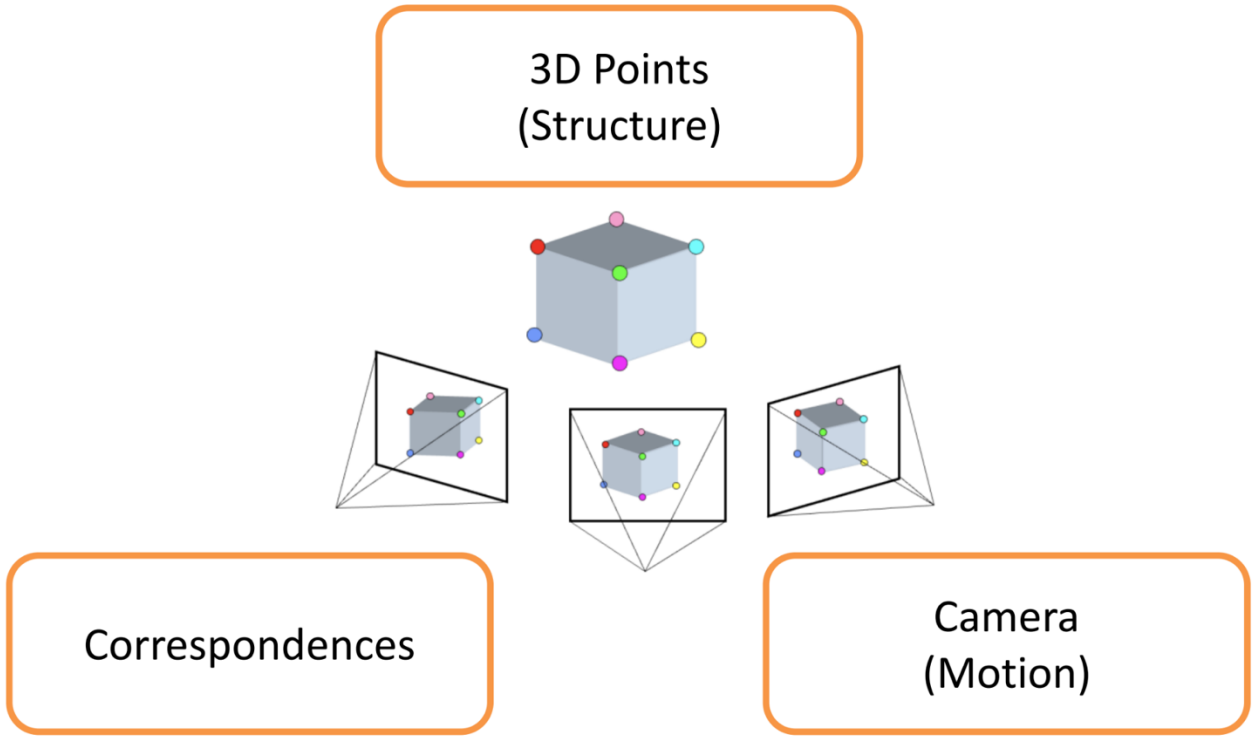


Image credit A. Efros

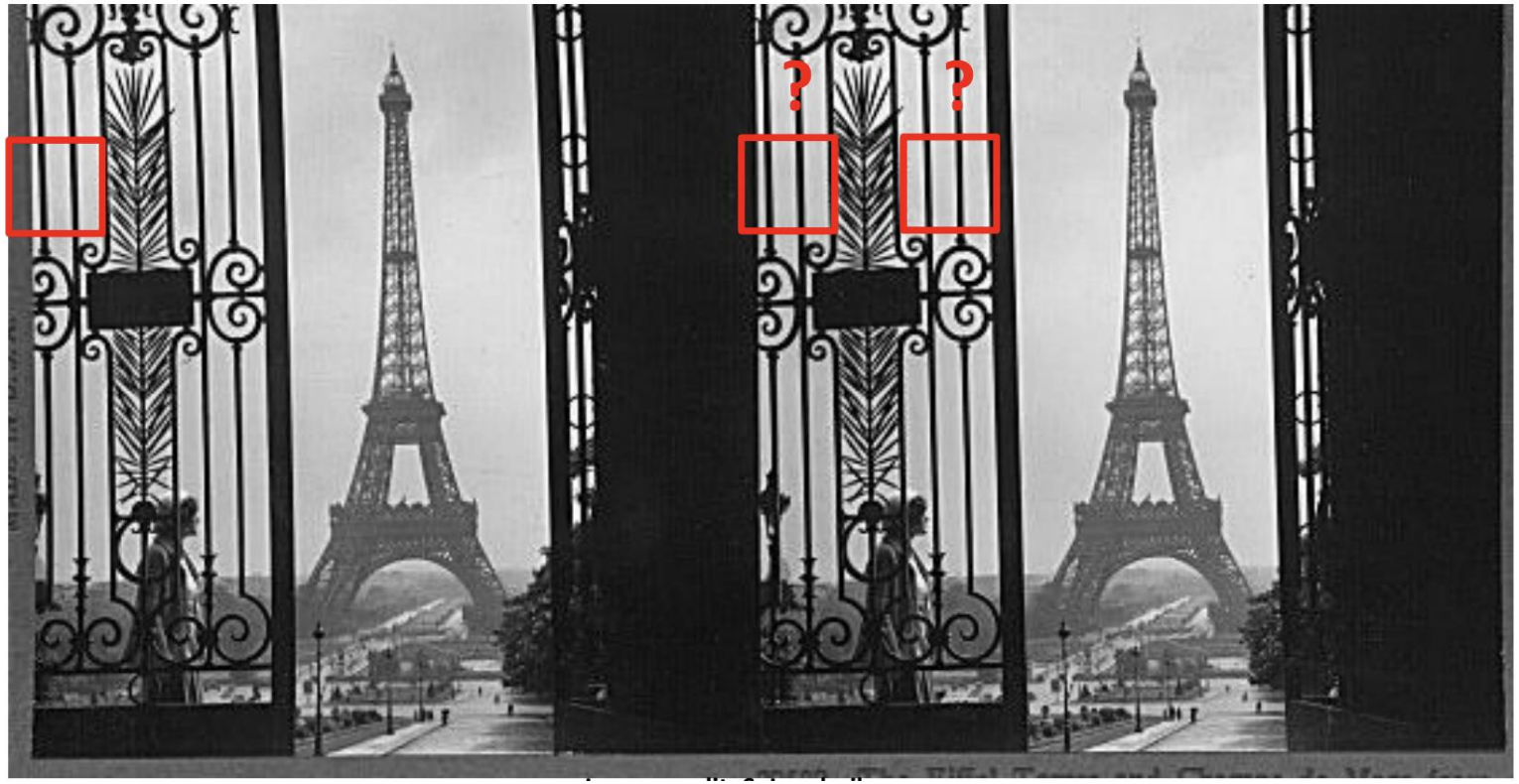
Content

- Image Formation
  - Pinhole camera model
  - 2D transformation
  - 3D transformation
- Geometry Reconstruction
  - Reconstruction with depths
  - Multi-view geometry reconstruction with camera motions and correspondence
  - Epipolar geometry reconstruction

How Can I Find the Corresponding Pixel in Another View?

- We assume the camera extrinsics are known
- Idea:
  1. Extract features: SIFT / Deep Neural Networks
  2. Do Nearest Neighbor Search on all pixels in another view

Failures of Correspondence Search



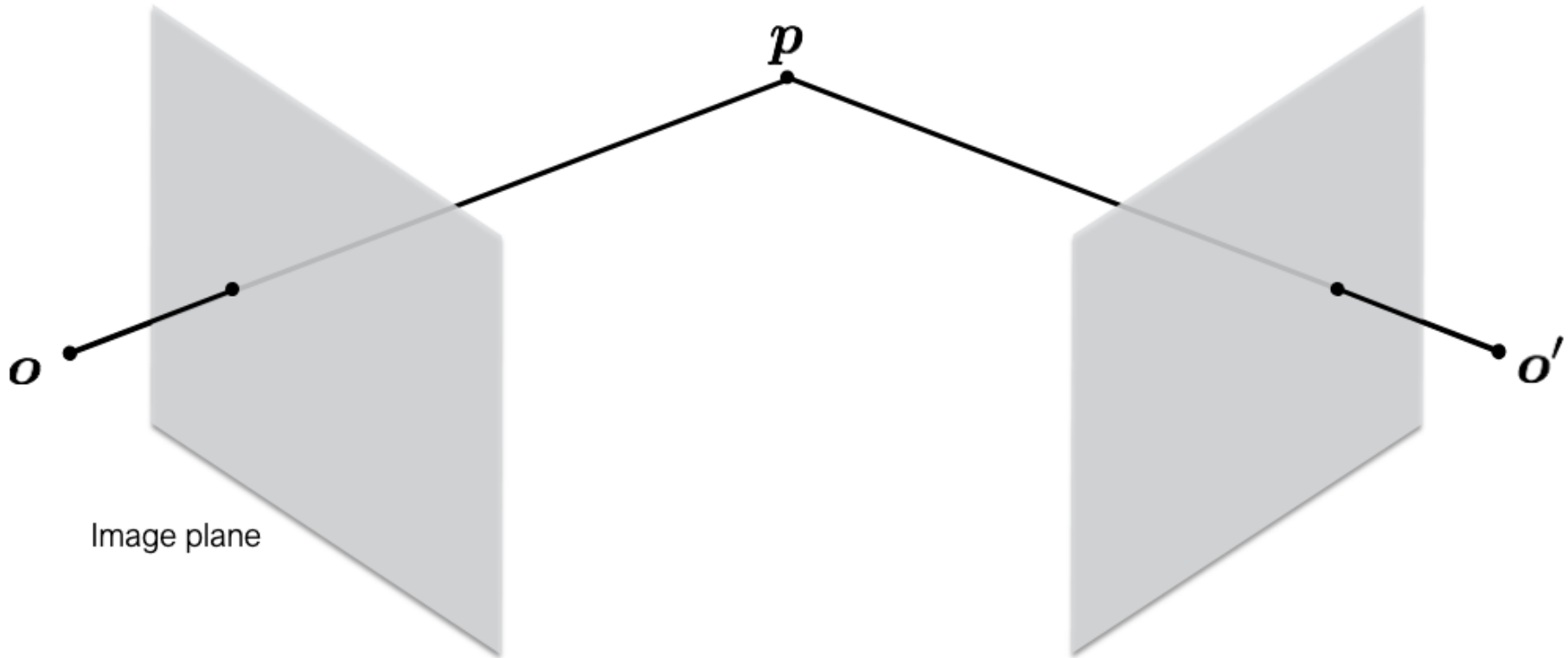
Is there any geometrical structure embedded in multiple views, that could facilitate correspondence search?

Image credit L. Lazebnik

If Camera Poses are Known, the Search Space of Pixel Correspondence is Constraint

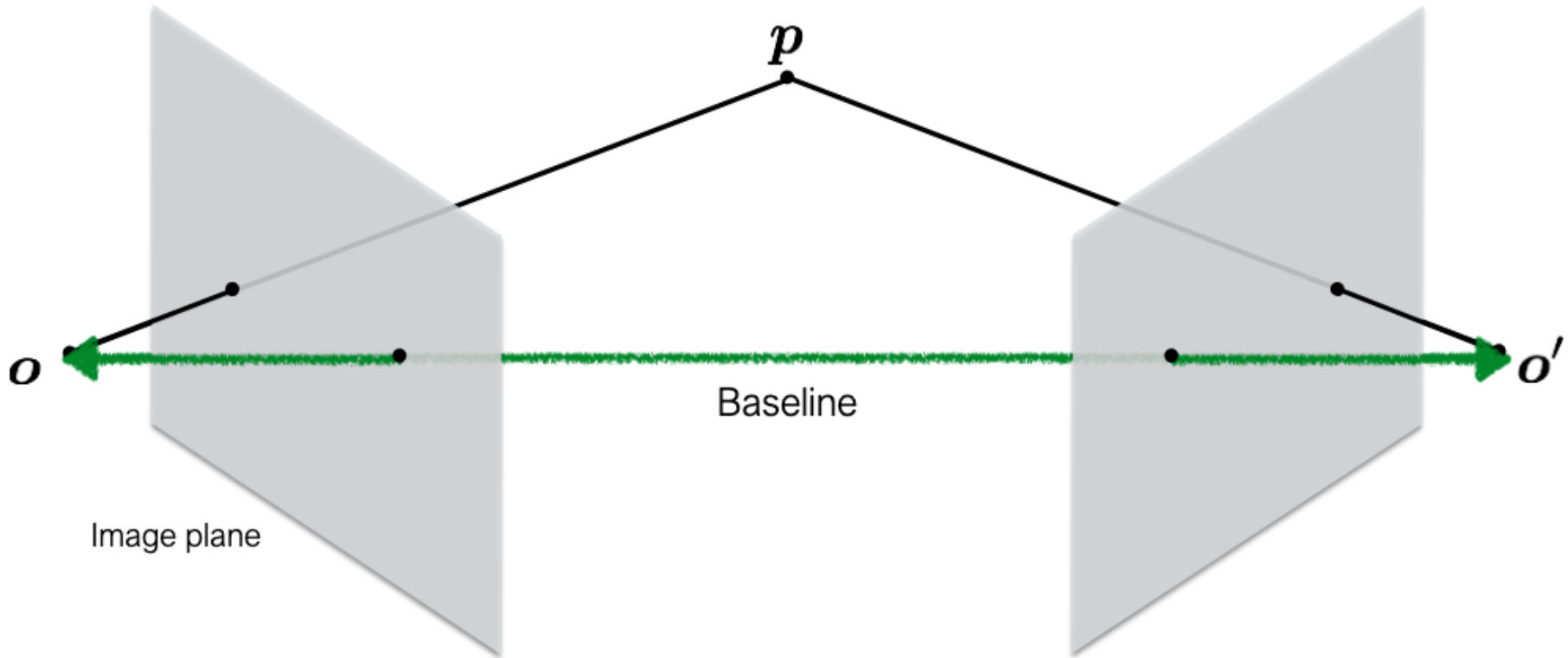
We don't need to search over the whole image, but the patches along the cyan line.

Epipolar geometry



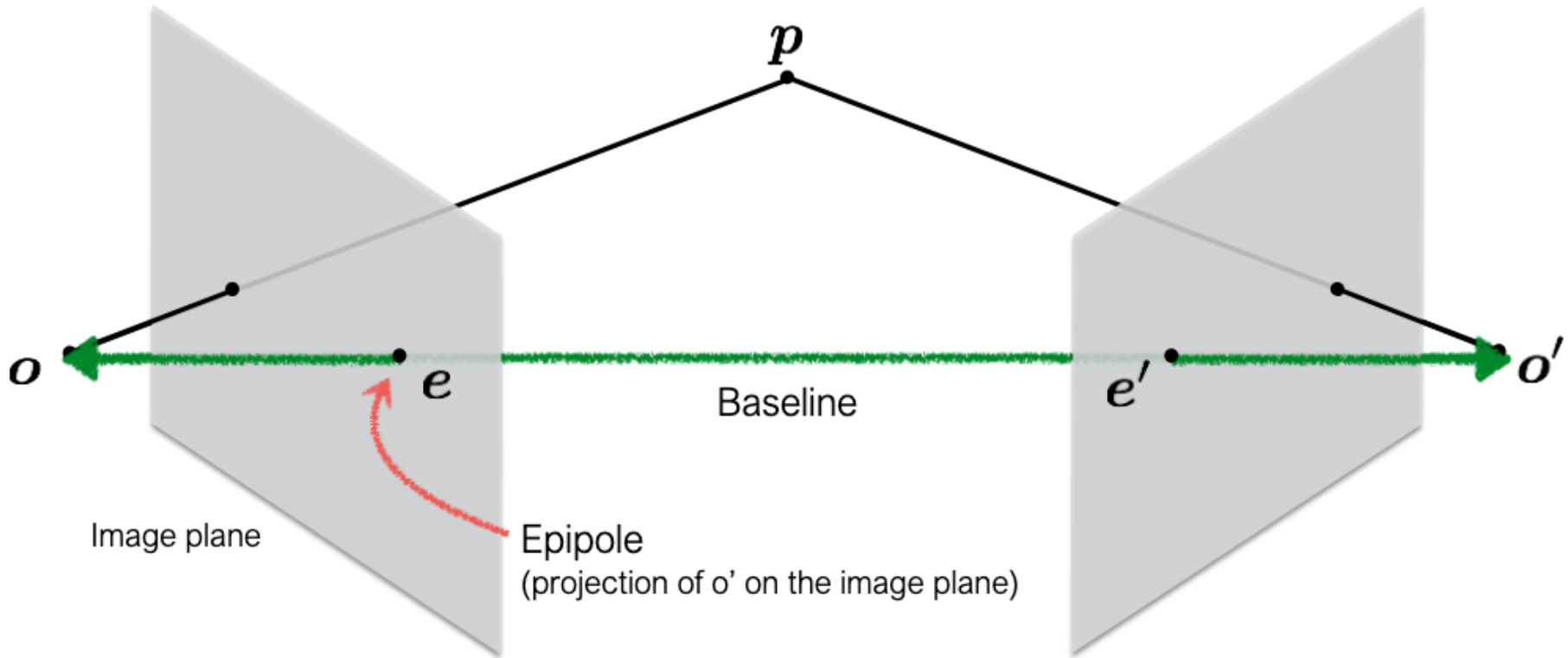
Slide credit K. Kitani and I. Gkioulekas

Epipolar geometry



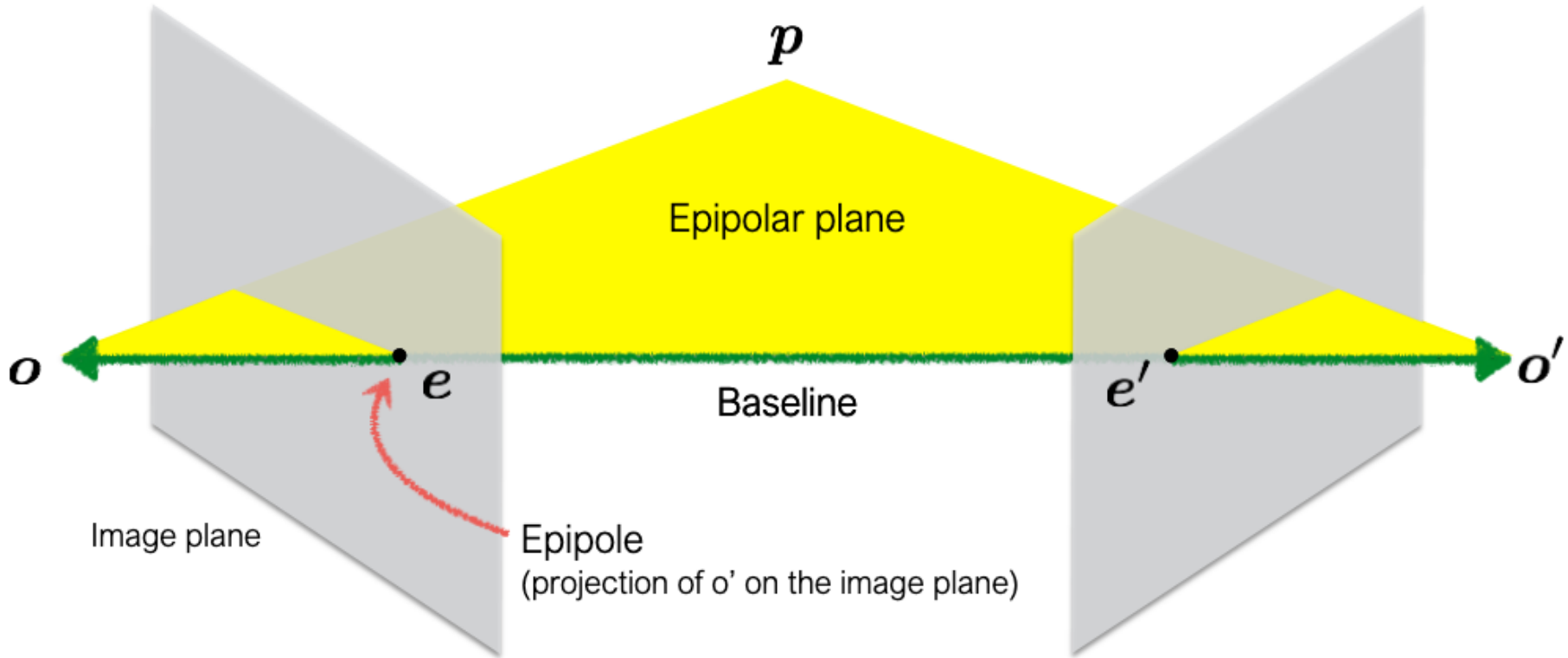
Slide credit K. Kitani and I. Gkioulekas

Epipolar geometry



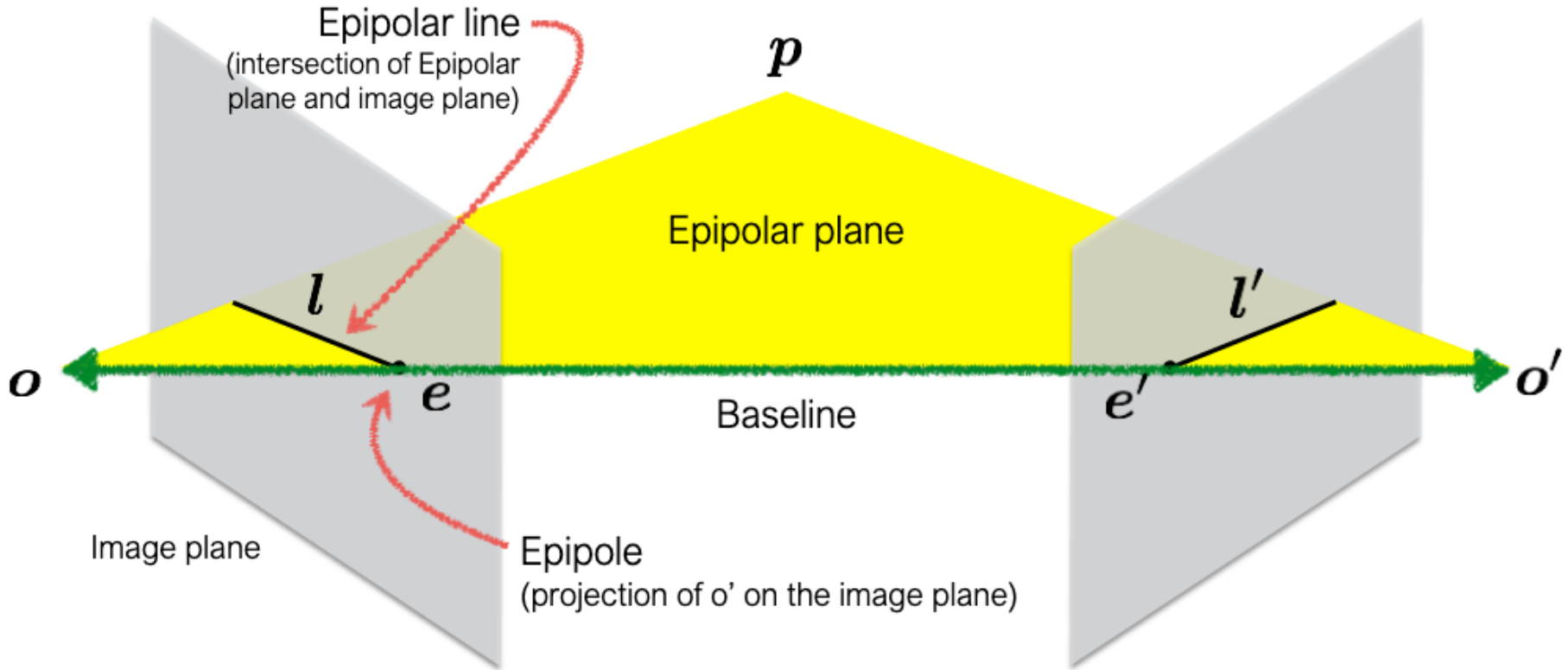
Slide credit K. Kitani and I. Gkioulekas

Epipolar geometry



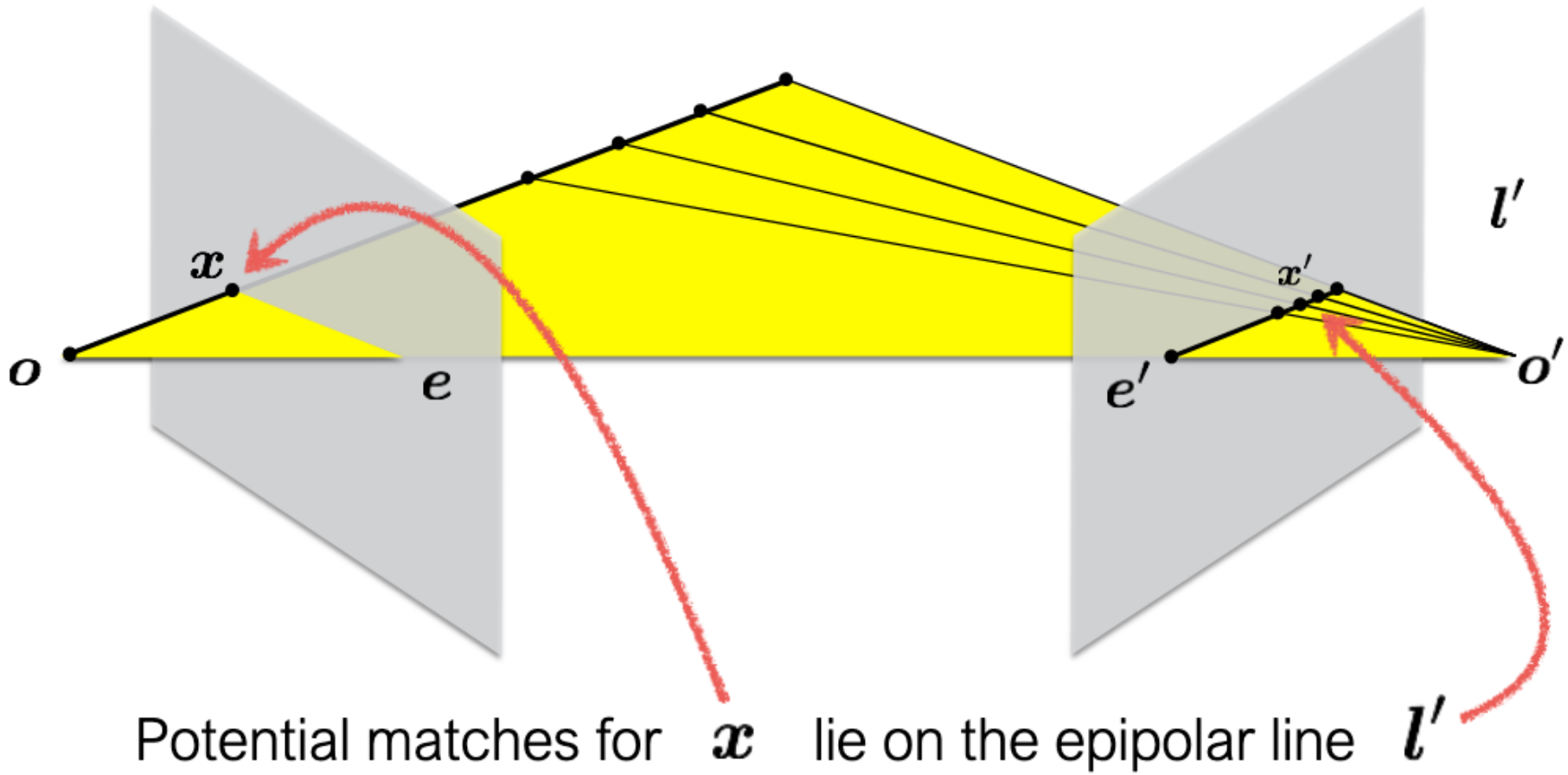
Slide credit K. Kitani and I. Gkioulekas

Epipolar geometry



Slide credit K. Kitani and I. Gkioulekas

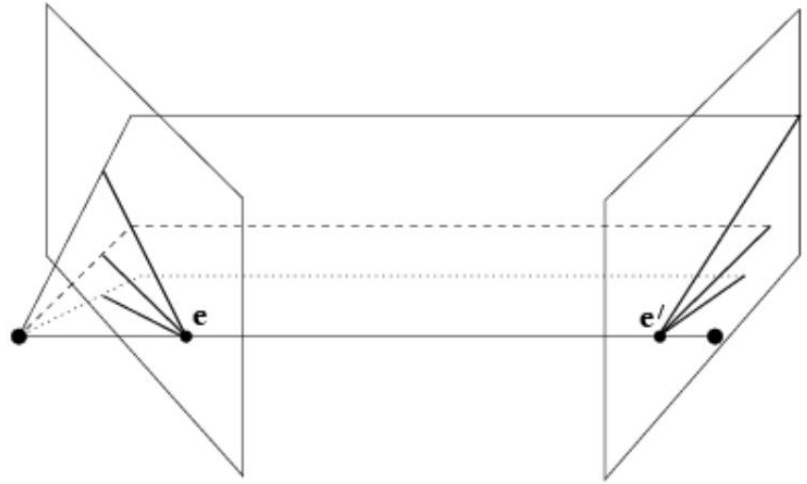
Epipolar constraint



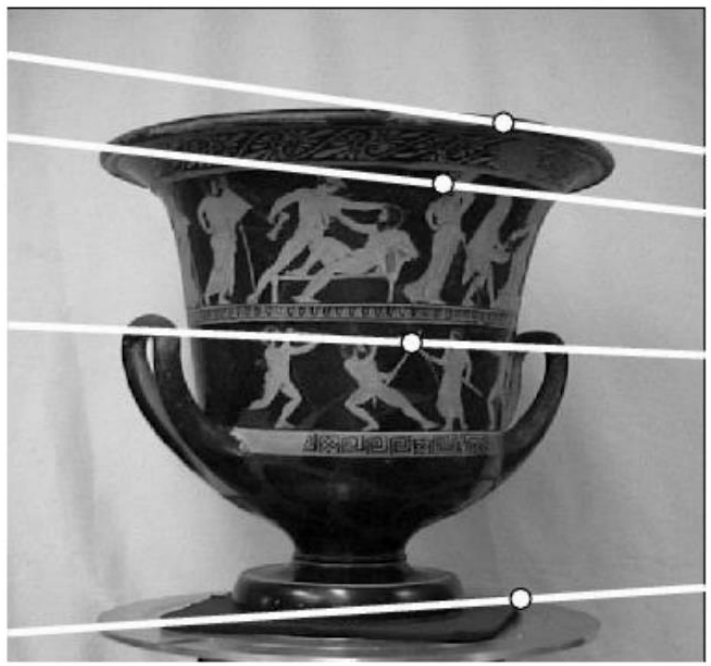
If Camera Poses are Known, the Search Corresponded Pixels Along the Epipolar Line

We don't need to search over the whole image, but the patches along the epipolar line.

Example: converging cameras



As position of 3d point varies, epipolar lines “rotate” about the baseline



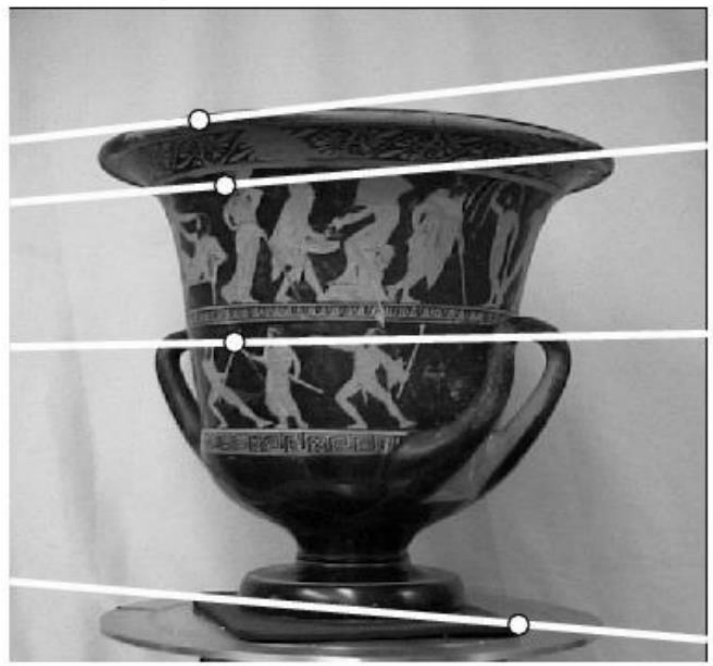
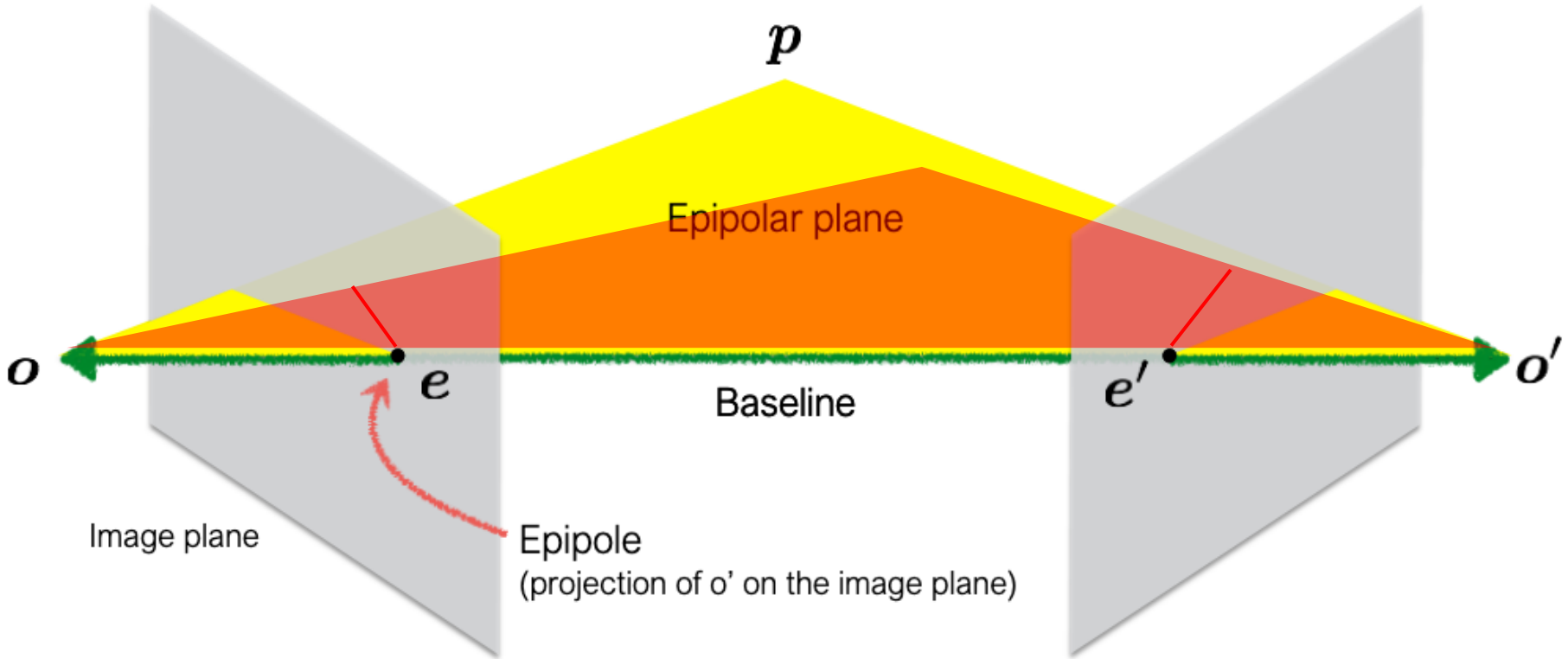


Figure from Hartley & Zisserman

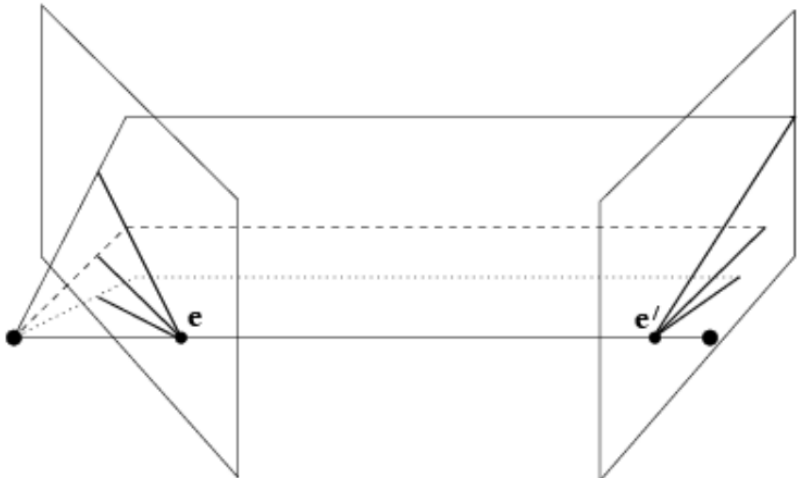
Slide credit K. Graumen and A. Efros

Example: Converging Camera

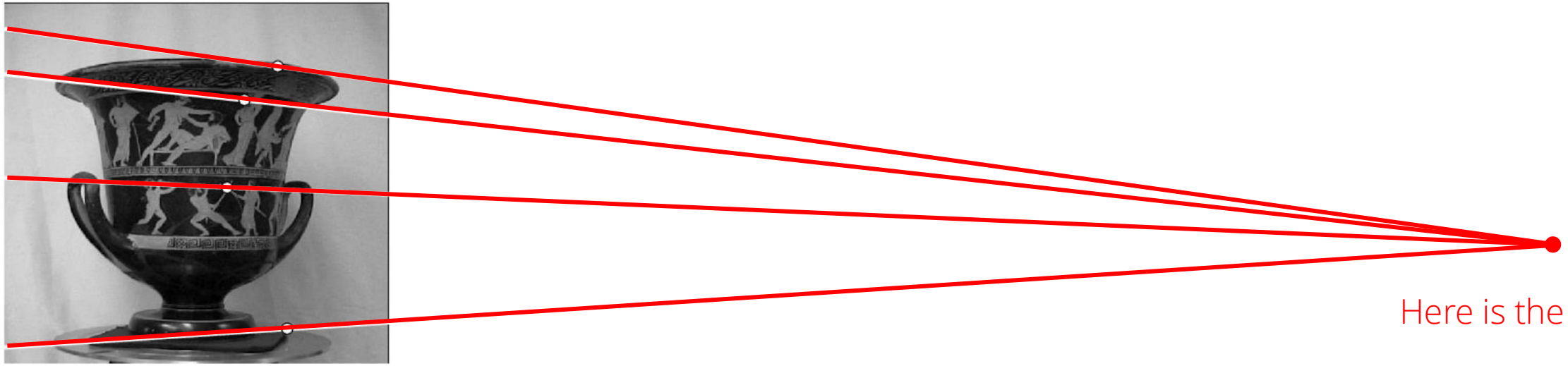


Slide credit K. Kitani and I. Gkioulekas

Example: converging cameras

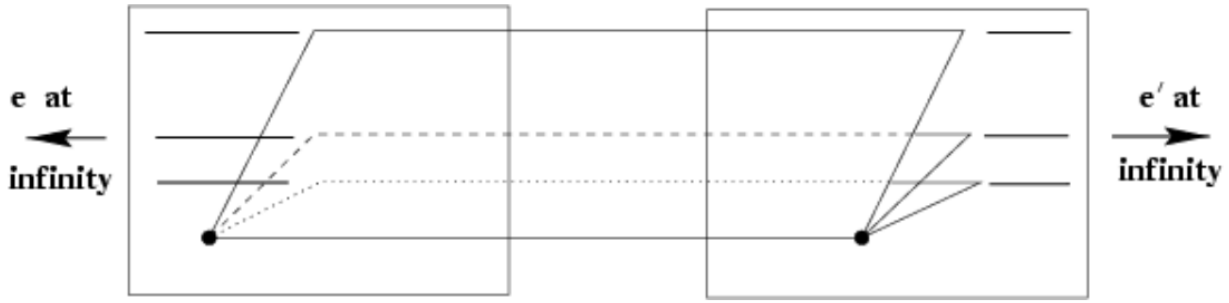


As position of 3d point varies, epipolar lines “rotate” about the baseline



Here is the Epipole!

Example: motion parallel with image plane

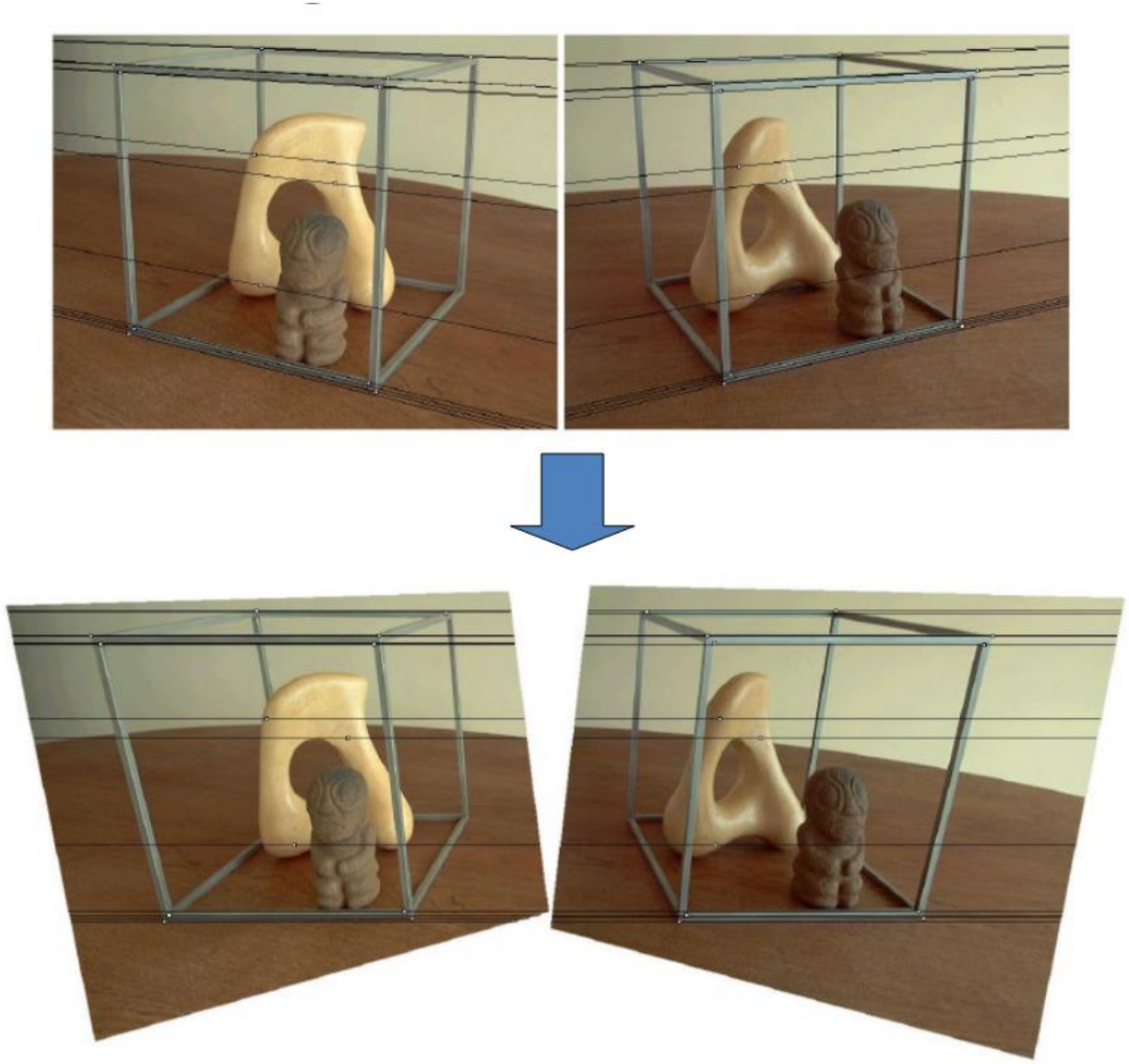


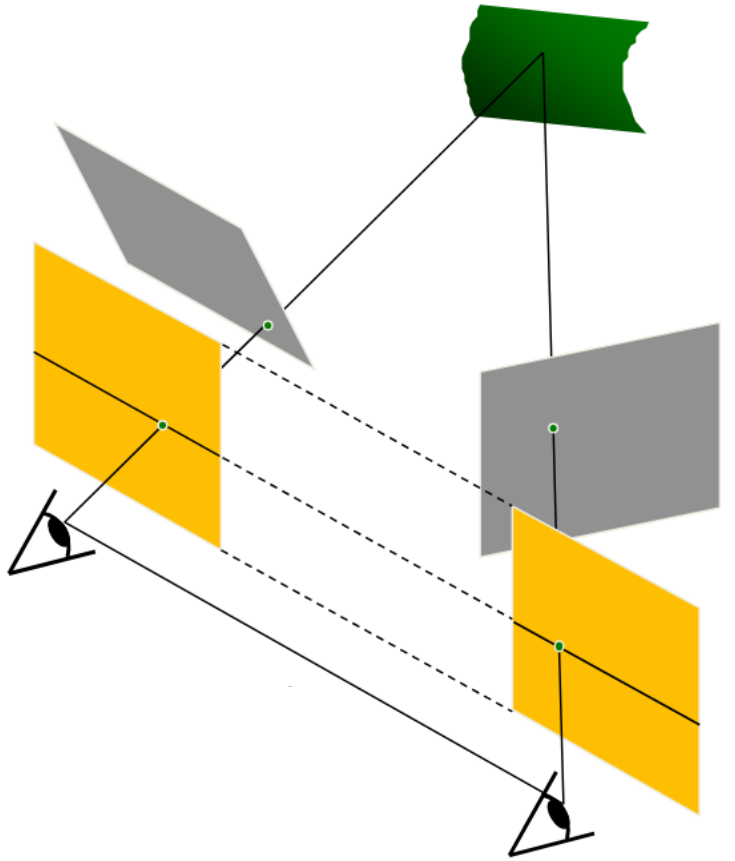
Epipolar lines are horizontally parallel

Figure from Hartley & Zisserman

Slide credit K. Graumen and A. Efros

Stereo Image Rectification



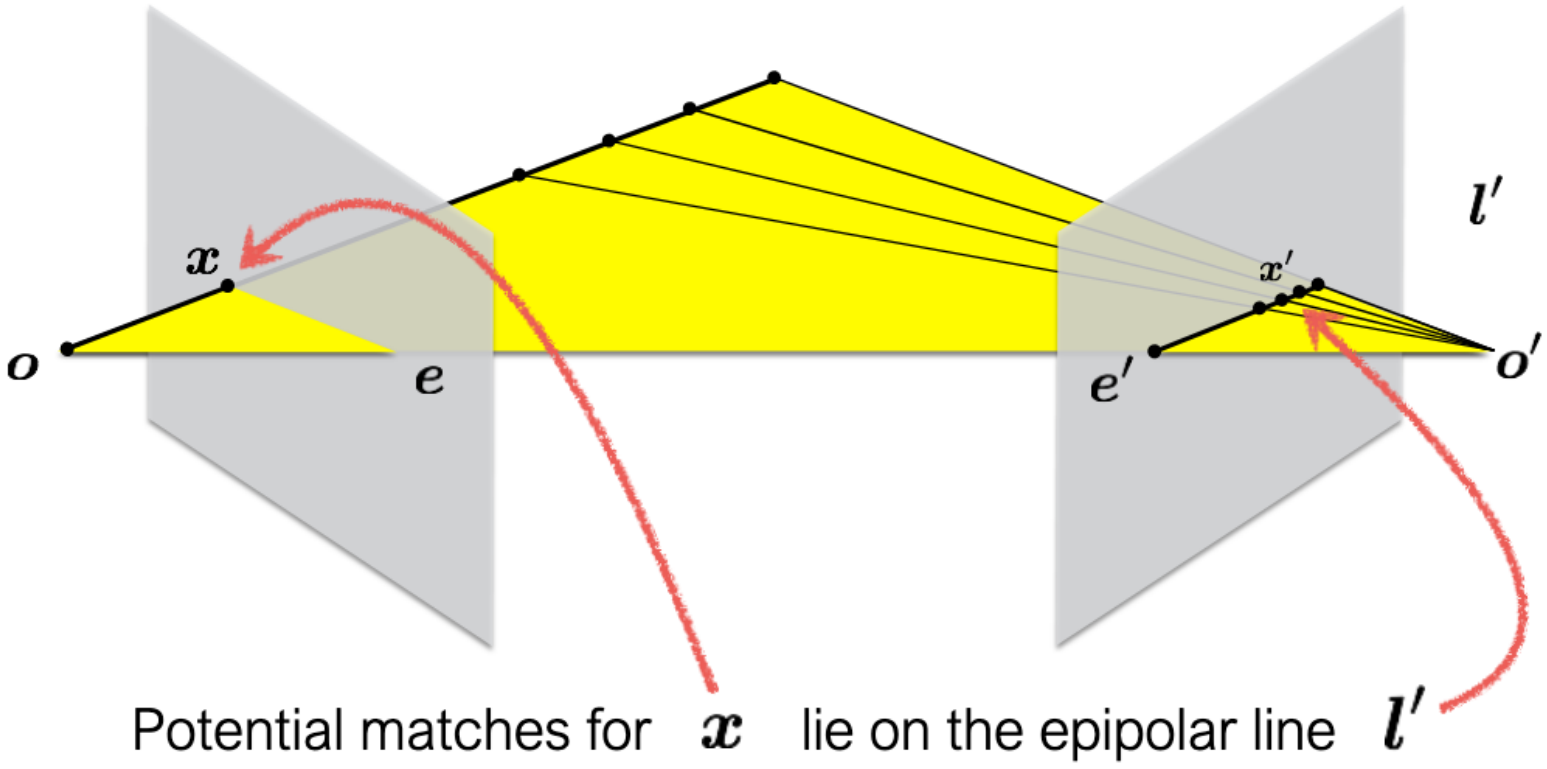


Slide credit J. Malik

Can We Represent the Epipolar Constraints Mathematically?

Given a pixel x in one view, describe the epipolar line l' in another view

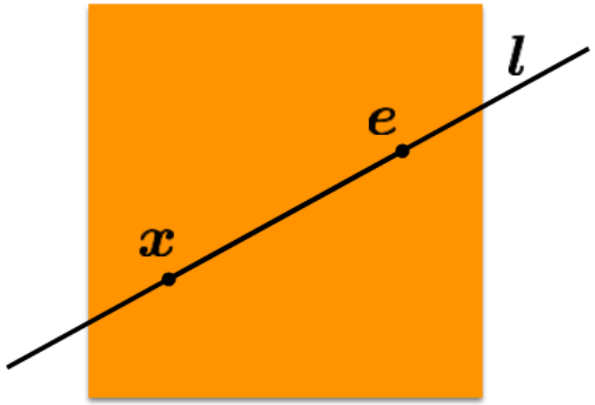
- If the camera poses are known, $l' = Ex$, where E is the essential matrix
- If the camera poses are unknown, $l' = Fx$, where F is the fundamental matrix



Slide credit K. Kitani

How to Represent a Line?

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

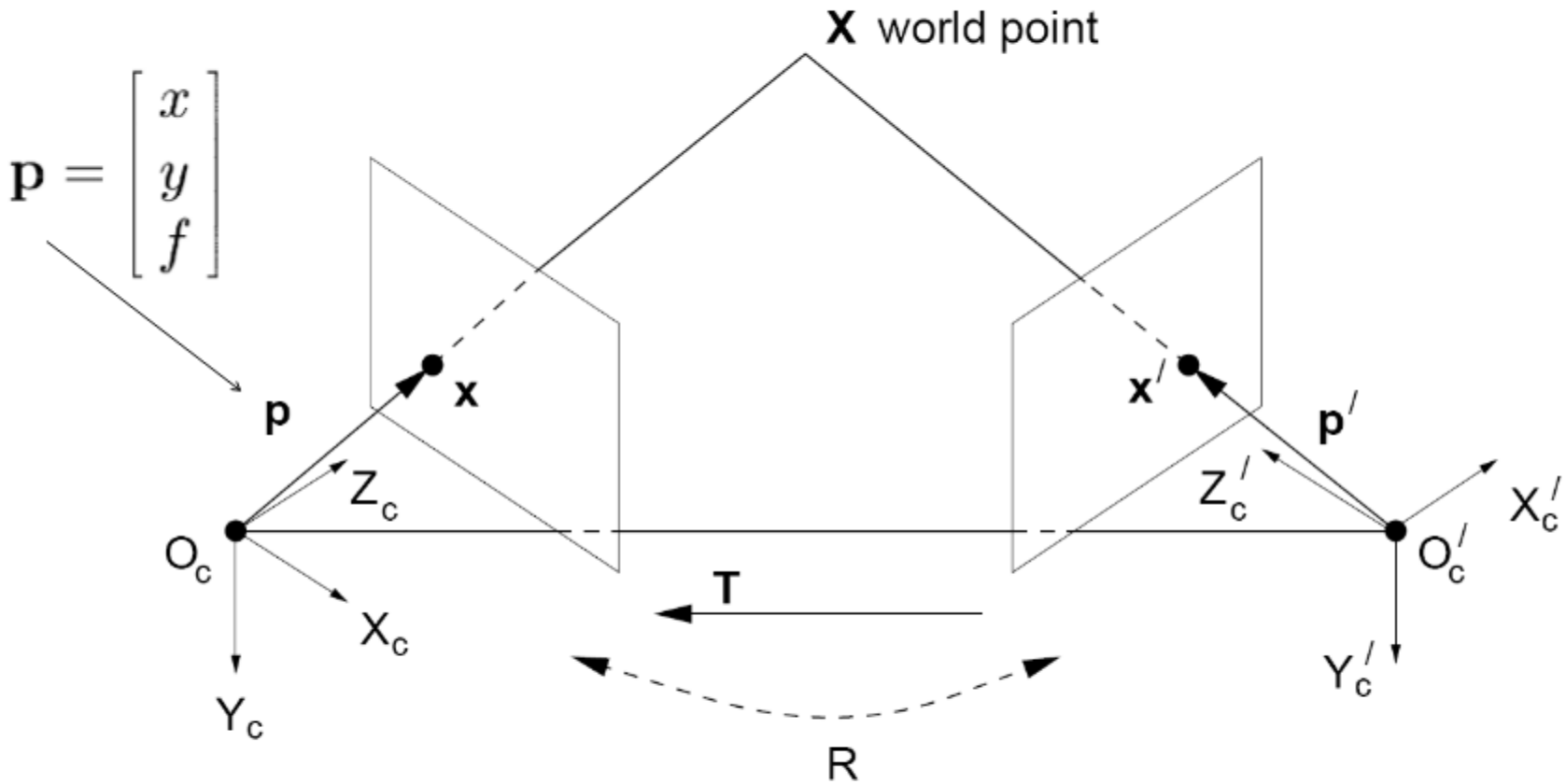


If the point $\boldsymbol{x}$ is on the epipolar line $\boldsymbol{l}$ then

$$\boldsymbol{x}^\top \boldsymbol{l} = 0$$

Deriving the Essential Matrix:

Stereo geometry, with calibrated cameras



Camera-centered coordinate systems are related by known rotation **R** and translation **T**:

$$\mathbf{X}' = \mathbf{R}\mathbf{X} + \mathbf{T}$$

Review: Cross product

$$\vec{a} \times \vec{b} = \vec{c}$$

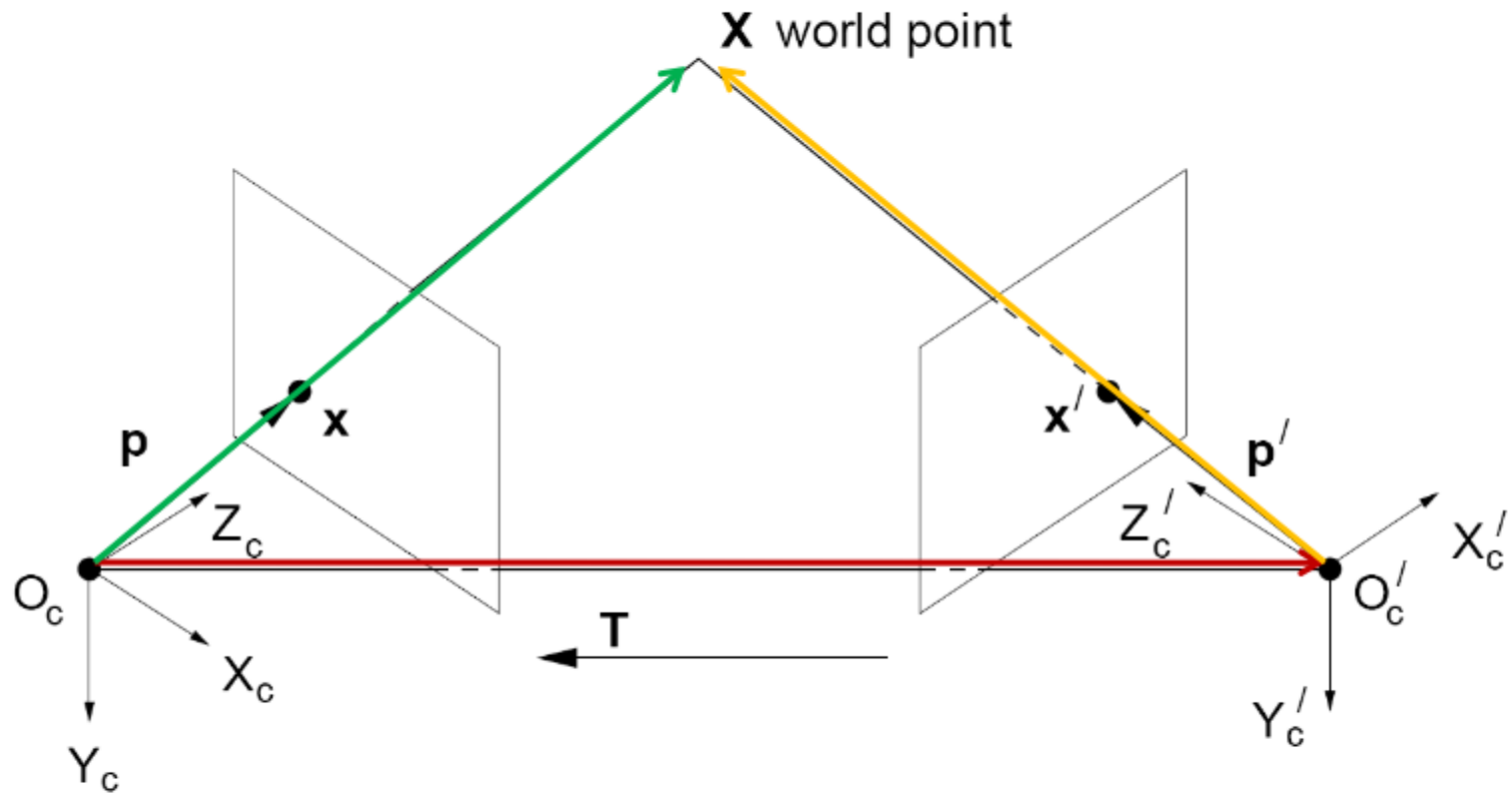
$$\vec{a} \cdot \vec{c} = 0$$
$$\vec{b} \cdot \vec{c} = 0$$

Vector cross product takes two vectors and returns a third vector that's perpendicular to both inputs.

So here, c is perpendicular to both a and b, which means the dot product = 0.

Deriving the Essential Matrix:

From geometry to algebra



$$\mathbf{X}' = \mathbf{R}\mathbf{X} + \mathbf{T}$$

$$\mathbf{T} \times \mathbf{X}' = \underbrace{\mathbf{T} \times \mathbf{X}'}_{\text{Normal to the plane}} =$$

$$= \mathbf{T} \times \mathbf{R}\mathbf{X}$$

$$\mathbf{X}' \cdot (\mathbf{T} \times \mathbf{X}') = \mathbf{X}' \cdot (\mathbf{T} \times \mathbf{R}\mathbf{X})$$

$$= 0$$

Slide credit A. Efros

Grauman

Representing Cross Product in a Matrix Form

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

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

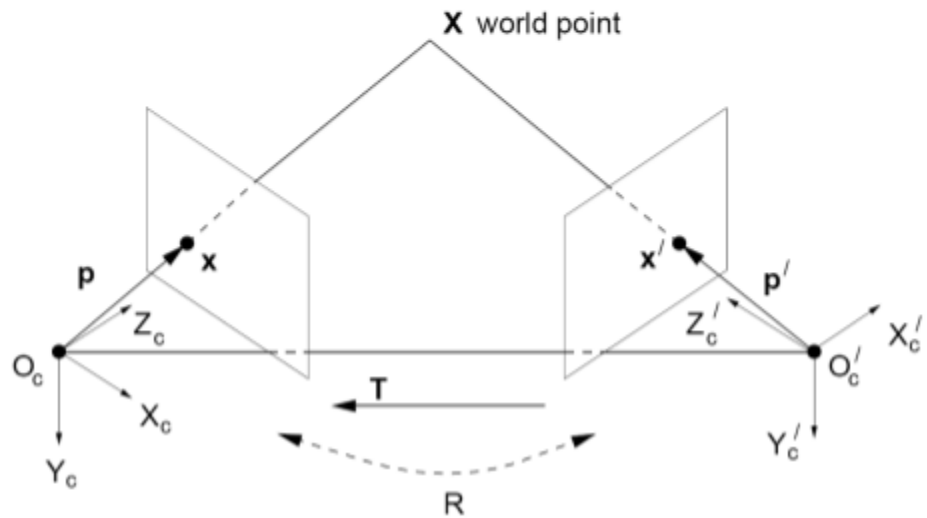
$[a]_\times$: skew matrix

Slide credit A. Efros

Essential matrix

$$\mathbf{X}' \cdot \left(\mathbf{T}_x \ \ \mathbf{R}\mathbf{X}\right) = 0$$

Let $\mathbf{E} = \mathbf{T}_x\mathbf{R}$



This holds for the rays **p** and **p'** that are parallel to the camera-centered position vectors **X** and **X'**, so we have:

$$\mathbf{p}'^T \mathbf{E}\mathbf{p} = 0$$

E is called the **essential matrix**, which relates corresponding image points [Longuet-Higgins 1981]

Slide credit A. Efros

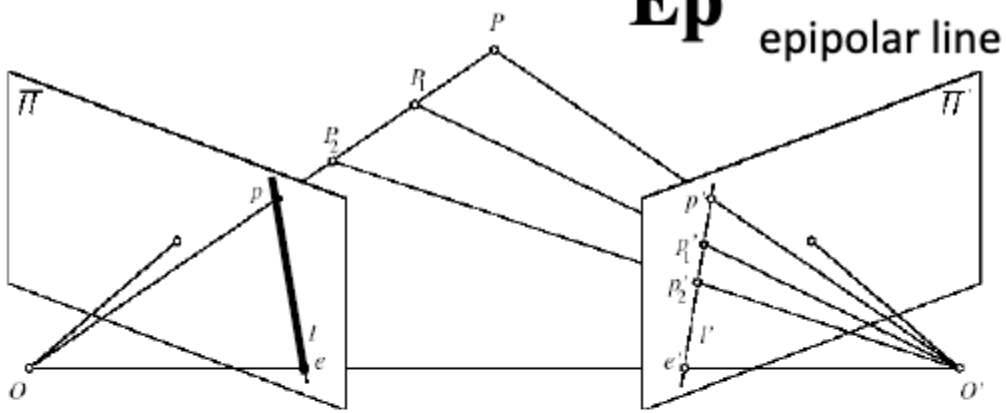
Grauman

Essential matrix and epipolar lines

$$\mathbf{p'}^{\mathrm{T}}\mathbf{E}\mathbf{p} = 0$$

Epipolar constraint: if we observe point **p** in one image, then its position **p'** in second image must satisfy this equation.

$\mathbf{E}\mathbf{p}$ is the coordinate vector representing the epipolar line associated with point p

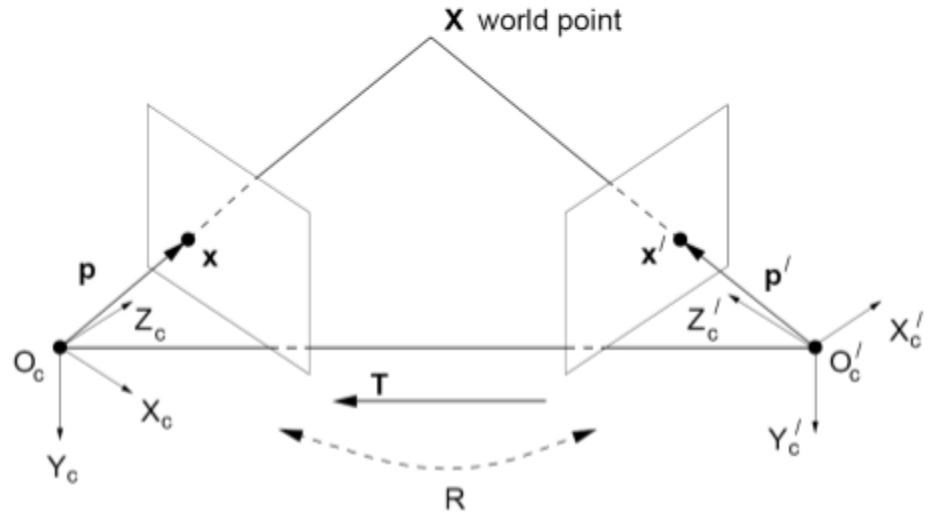


$\mathbf{E}^T\mathbf{p'}$ is the coordinate vector representing the epipolar line associated with point p'

Uncalibrated cameras

$$\mathbf{E} = \mathbf{T}_x\mathbf{R}$$

$$\mathbf{p}'^T\,\mathbf{E}\mathbf{p} = 0$$



- For an *uncalibrated* stereo rig, can we we express the epipolar constraints algebraically via the **Essential Matrix**?
- No, we do not know T or R
- However we can use the **Fundamental Matrix**
  - Estimate epipolar geometry from a (redundant) set of point correspondences between two uncalibrated cameras

Slide credit A. Efros

Uncalibrated case

For a given camera:

$$\overline{\mathbf{p}} = \mathbf{M}_{\text{int}}\mathbf{p}$$

Camera coordinates

So, for two cameras (left and right):

$$\mathbf{p}_{(left)} = \mathbf{M}^{-1}_{left,\text{int}}\overline{\mathbf{p}}_{(left)}$$

$$\mathbf{p}_{(right)} = \mathbf{M}^{-1}_{right,\text{int}}\overline{\mathbf{p}}_{(right)}$$

Camera coordinates

Image pixel coordinates

Internal calibration matrices, one per camera

Uncalibrated case:
Fundamental matrix

$$\mathbf{p}_{(left)} = \mathbf{M}^{-1}_{left,\text{int}}\,\overline{\mathbf{p}}_{(left)}$$

$$\mathbf{p}_{(right)} = \mathbf{M}^{-1}_{right,\text{int}}\,\overline{\mathbf{p}}_{(right)}$$

$$^{c}\mathbf{p}_{(right)}{}^{\mathrm{T}}\mathbf{E}\mathbf{p}_{(left)} = 0$$

From before, the **essential** matrix **E.**

$$\left(\mathbf{M}^{-1}_{right,\text{int}}\,\overline{\mathbf{p}}_{right}\right)^{\mathrm{T}}\mathbf{E}\left(\mathbf{M}^{-1}_{left,\text{int}}\,\overline{\mathbf{p}}_{left}\right) = 0$$

$$\overline{\mathbf{p}}^{\mathrm{T}}_{right}\left(\mathbf{M}^{-\mathrm{T}}_{right,\text{int}}\mathbf{E}\mathbf{M}^{-1}_{left,\text{int}}\right)\overline{\mathbf{p}}_{left} = 0$$

$$\overline{\mathbf{p}}^{\mathrm{T}}_{right}\mathbf{F}\,\overline{\mathbf{p}}_{left} = 0$$

Fundamental matrix

Slide credit A. Efros

Fundamental matrix

- Relates pixel coordinates in the two views
- More general form than essential matrix: we remove need to know intrinsic parameters
- If we estimate fundamental matrix from correspondences in pixel coordinates, can reconstruct epipolar geometry without intrinsic or extrinsic parameters

If Intrinsics are Unknown, How to Get Fundamental Matrix? Correspondence!

Each point correspondence generates one constraint on F

$$\overline{\mathbf{p}}_{right}^{\mathrm{T}}\mathbf{F}\overline{\mathbf{p}}_{left}=0$$

$$\begin{bmatrix} u' & v' & 1 \end{bmatrix}\begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}=0$$

- (u', v') and (u, v) are pixel coordinates of a pair of corresponding pixel from two views
- $\boldsymbol{F}$ has 9 variables but we can impose a constraint on the scale of $\boldsymbol{F}$, ending up with 8 degree of freedom.
- The Eight-point algorithm: we can solve the system with at least 8 pairs of corresponding pixels.

Slide credit A. Efros

If Intrinsics are Unknown, How to Get Fundamental Matrix? Correspondence!

Each point correspondence generates one constraint on F

$$\overline{\mathbf{p}}_{right}^{\mathrm{T}}\mathbf{F}\overline{\mathbf{p}}_{left}=0$$

$$\begin{bmatrix} u' & v' & 1 \end{bmatrix}\begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}=0$$

Collect n of these constraints

$$\begin{bmatrix} u'_1u_1 & u'_1v_1 & u'_1 & v'_1u_1 & v'_1v_1 & v'_1 & u_1 & v_1 & 1 \end{bmatrix}\begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix}=\mathbf{0}$$

Solve for f , vector of parameters.

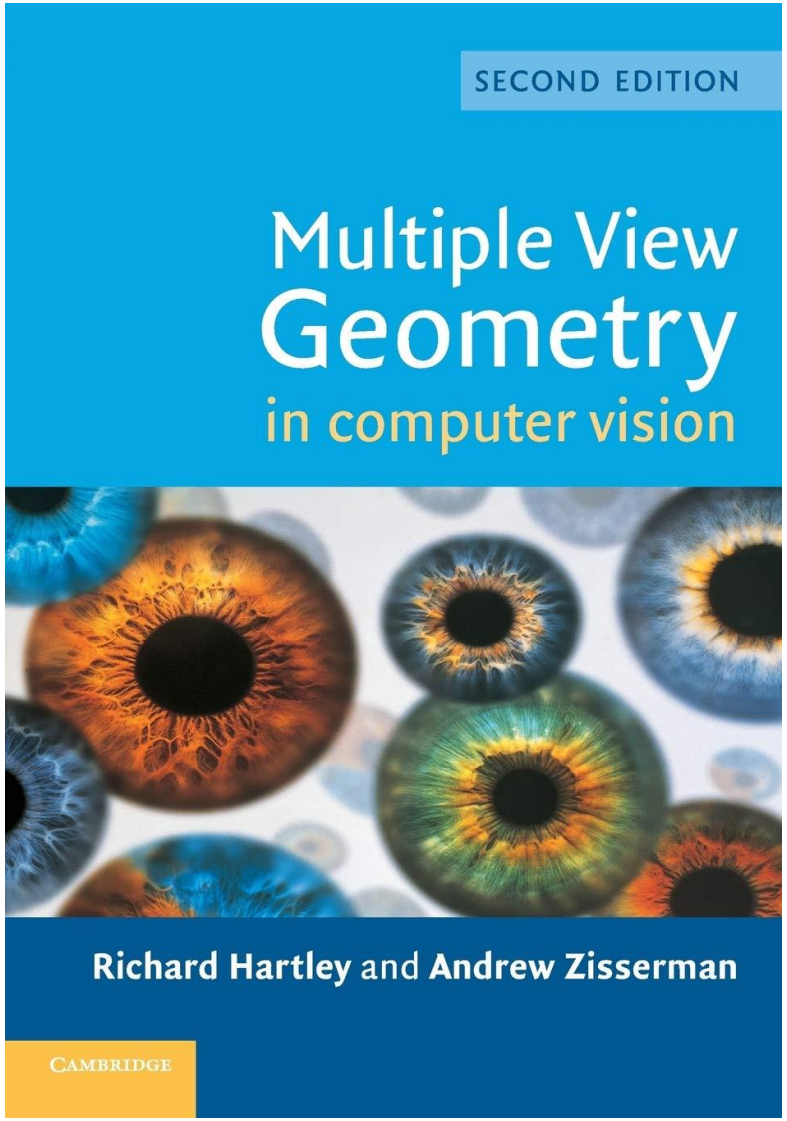
Grauman

Slide credit A. Efros

Why Do We Need to Known Multi-view Geometry?

Building Rome in a Day. Agarwal et al.

Learn More About Multiview Geometry

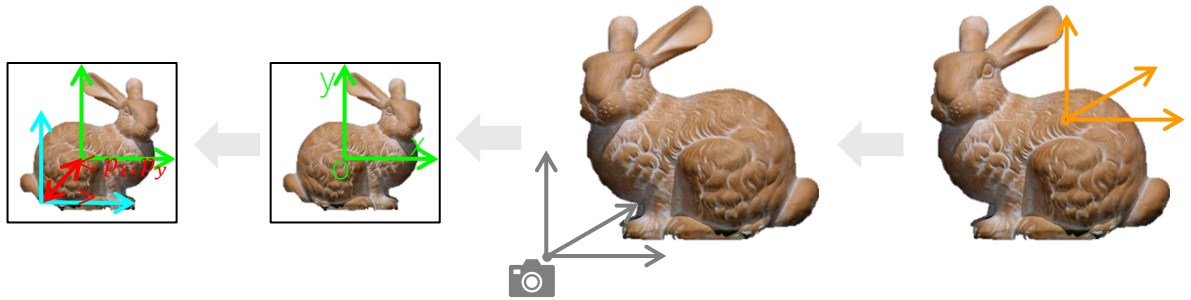


Learn More About Multiview Geometry

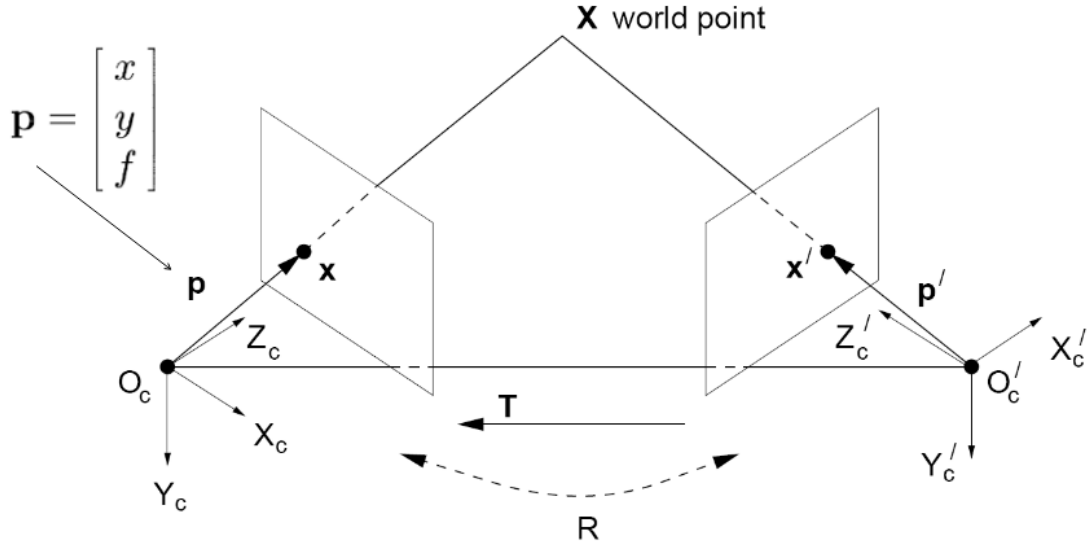
- A more advanced lecture in geometry-based vision
  - ➢ Geometry-based Methods in Vision by Shubham Tulsiani at CMU

Summary

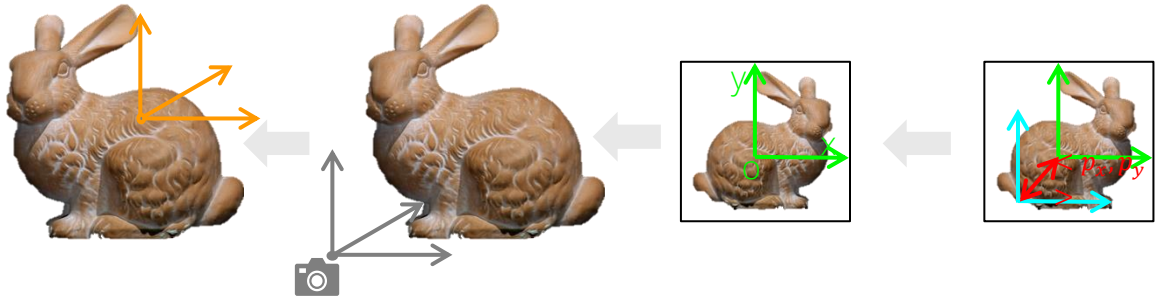
3D-to-2D perspective projection



$$\begin{bmatrix} x_{pix} \\ y_{pix} \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{R}_{\boldsymbol{w}}^{\boldsymbol{c}} & \boldsymbol{T}_{\boldsymbol{w}}^{\boldsymbol{c}} \\ \mathbf{0} & 1 \end{bmatrix} \quad \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$



2D-to-3D reconstruction



$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{\boldsymbol{w}}^{\boldsymbol{c}} & \boldsymbol{T}_{\boldsymbol{w}}^{\boldsymbol{c}} \\ \mathbf{0} & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x_{pix} \\ y_{pix} \\ 1 \end{bmatrix}$$

What We Will Cover Next Week

- Structure from motion and 3D foundation models
- 3D Scene Representations