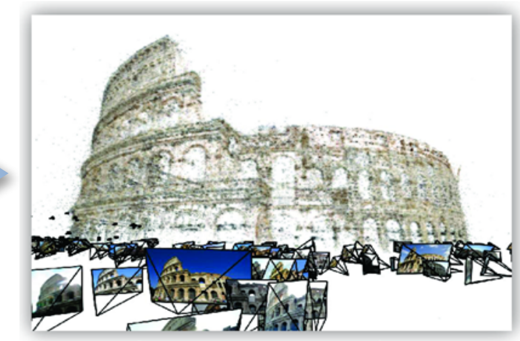


相机模型与多视几何

周晓巍

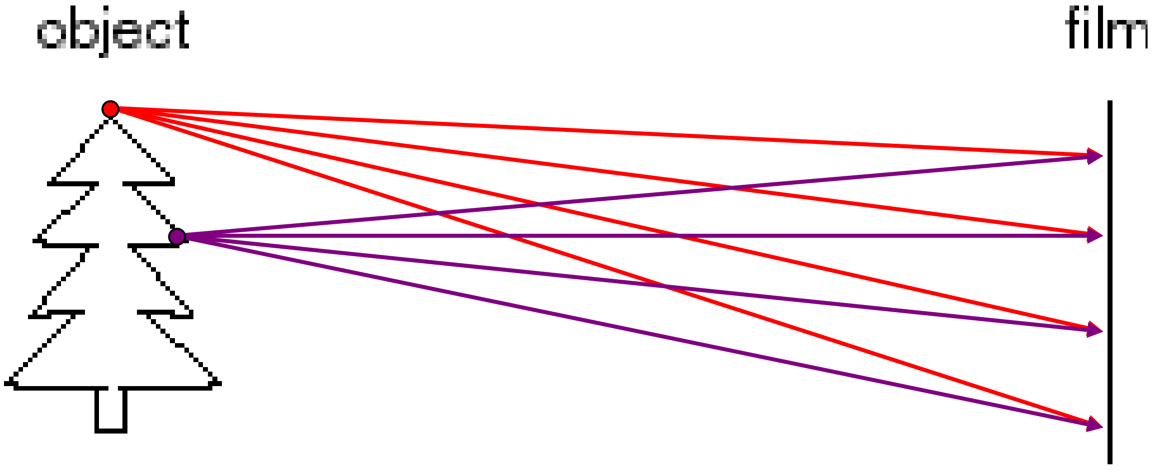
浙江大学CAD&CG实验室

Slides adapted from Noah Snavely, Jia-Bin Huang, S. Seitz and D. Hoiem

Camera Model and Multi-view Geometry

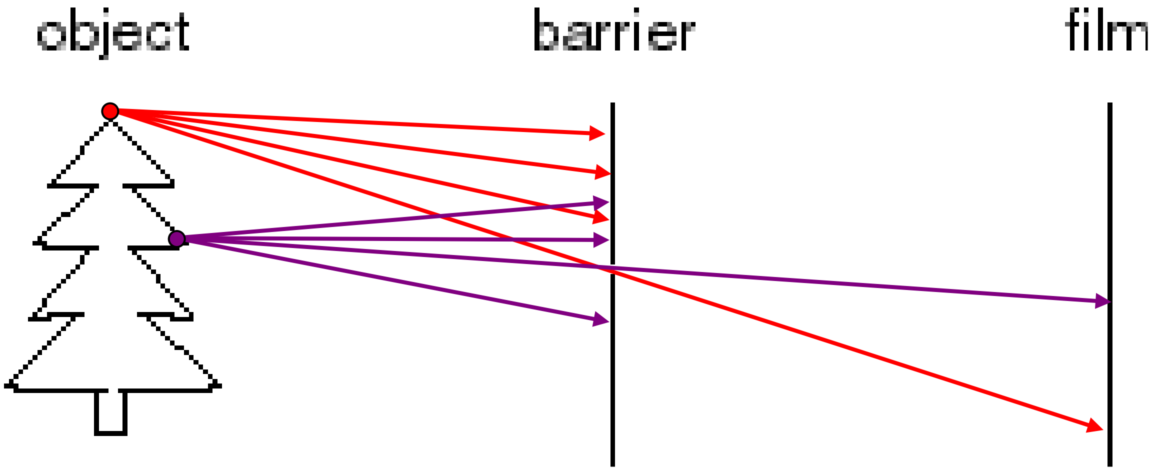
- **Camera Models**（相机模型）
  - What’s the geometric relation between image and world coordinates?
- **Multi-View Geometry** （多视几何）
  - What’s the geometric relation between images taken from different viewpoints?
- **3D Reconstruction**（三维重建）
  - How can we recover 3D geometry of the world from two or multiple images?

Image formation



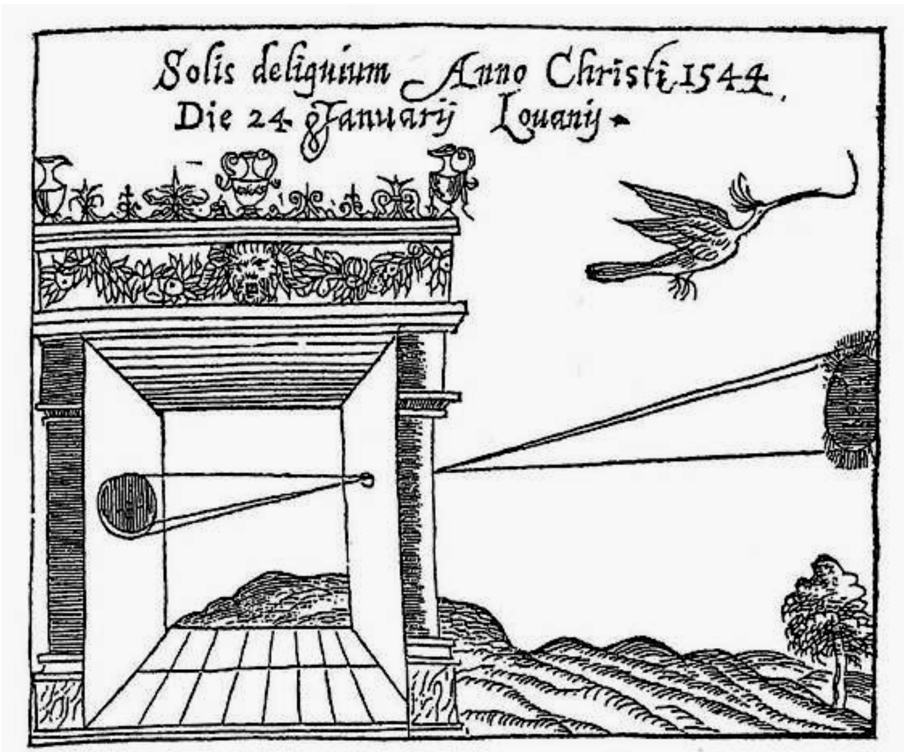
- Let's design a camera
  - Idea 1: put a piece of film in front of an object
  - Do we get a reasonable image?
  - No. This is a bad camera (not one-to-one).

Pinhole camera



- Add a barrier to block off most of the rays
  - The opening known as the **aperture**

Pinhole camera

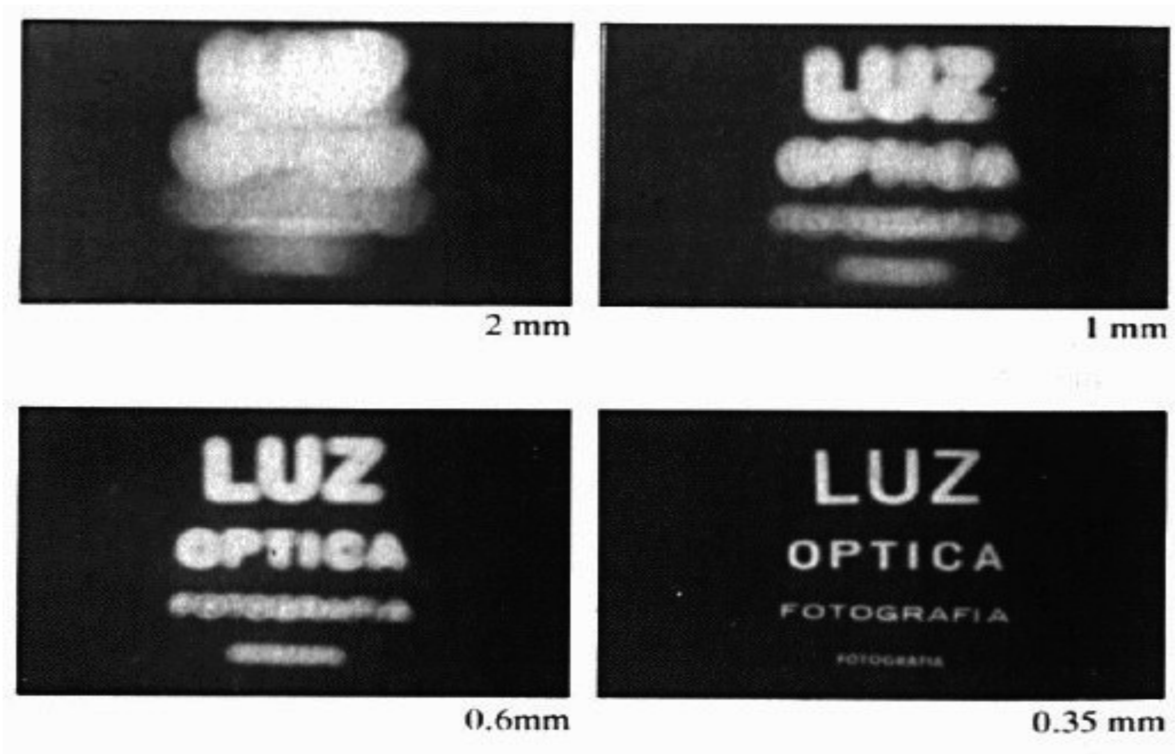


Gemma Frisius, 1558

- Basic principle known to Mozi (470-390 BC), Aristotle (384-322 BC)

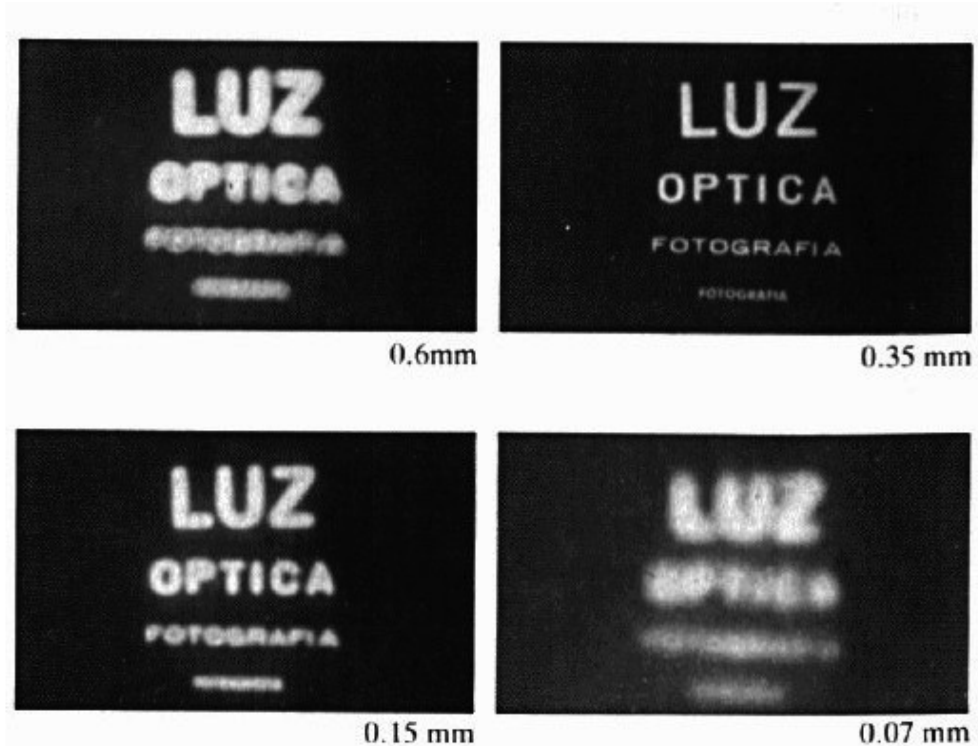
Source: A. Efros

Shrinking the aperture



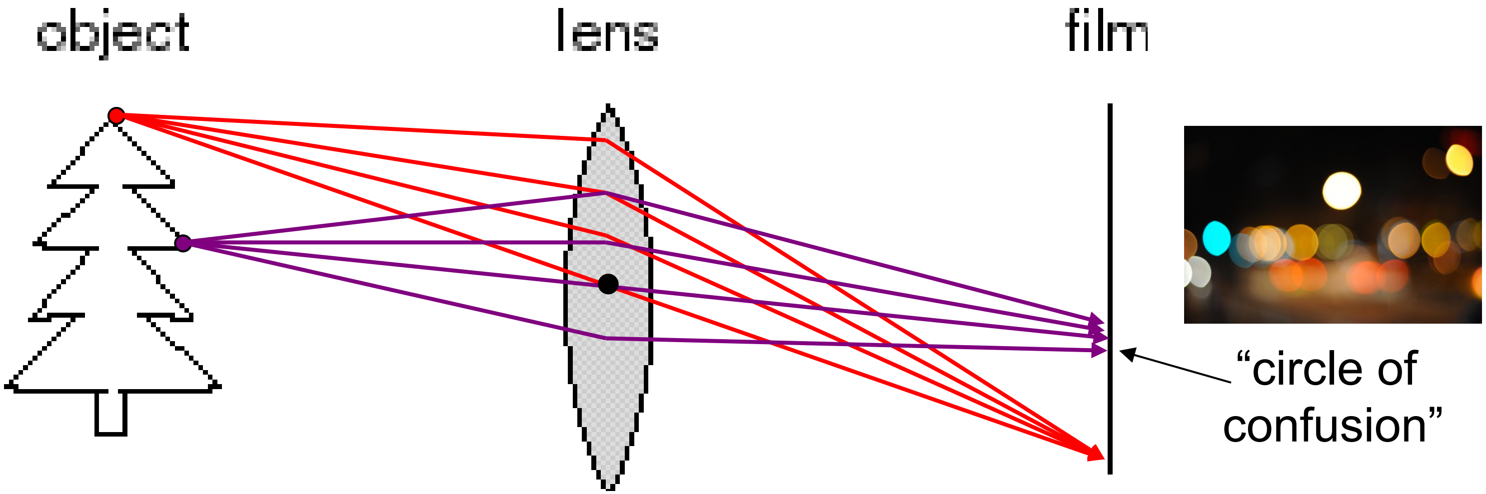
- Why not make the aperture as small as possible?

Shrinking the aperture



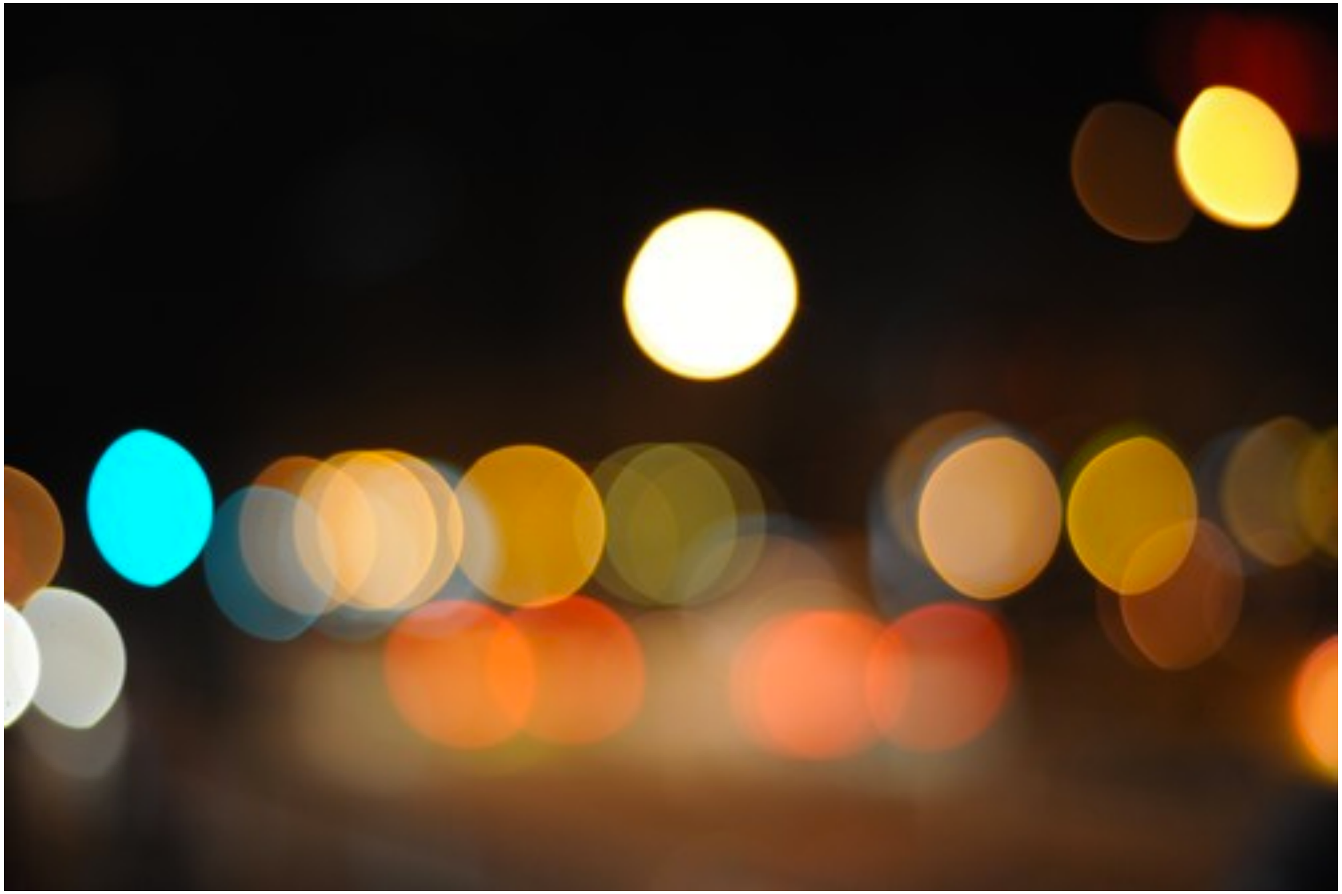
- Why not make the aperture as small as possible?
  - Less light gets through
  - *Diffraction* effects...

Adding a lens

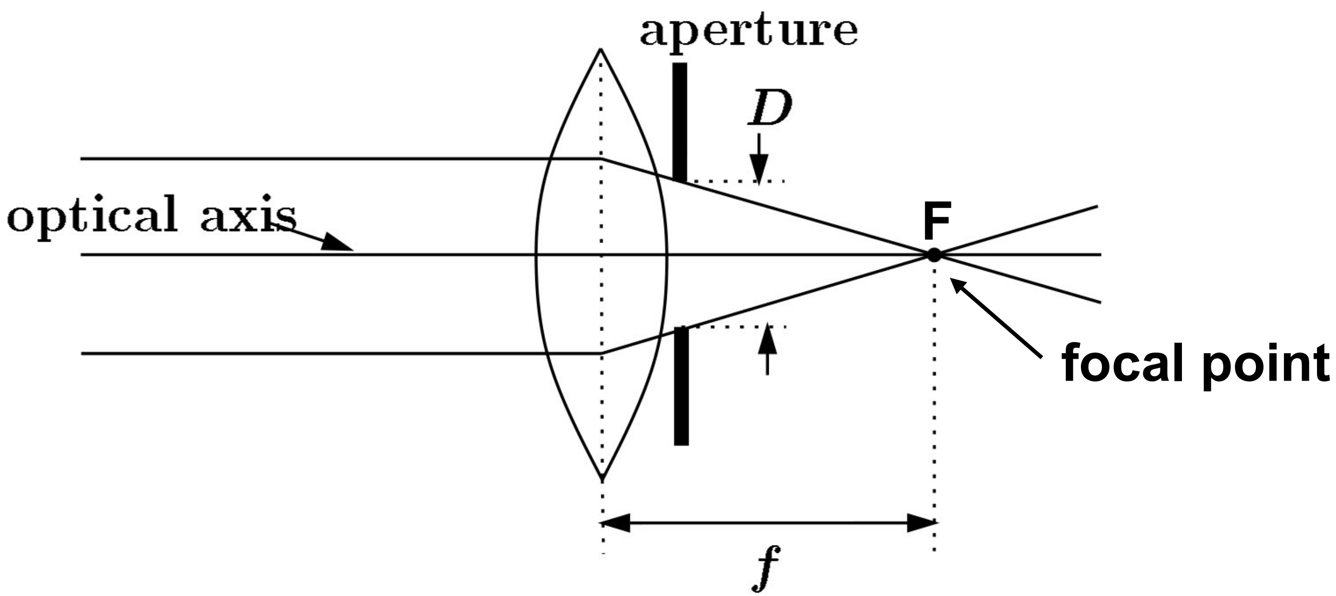


- A lens focuses light onto the film
  - There is a specific distance at which objects are “in focus”
  - other points project to a “circle of confusion” in the image
- Lens equation (thin lens)

$$\frac{1}{d_0} + \frac{1}{d_i} = \frac{1}{f}$$

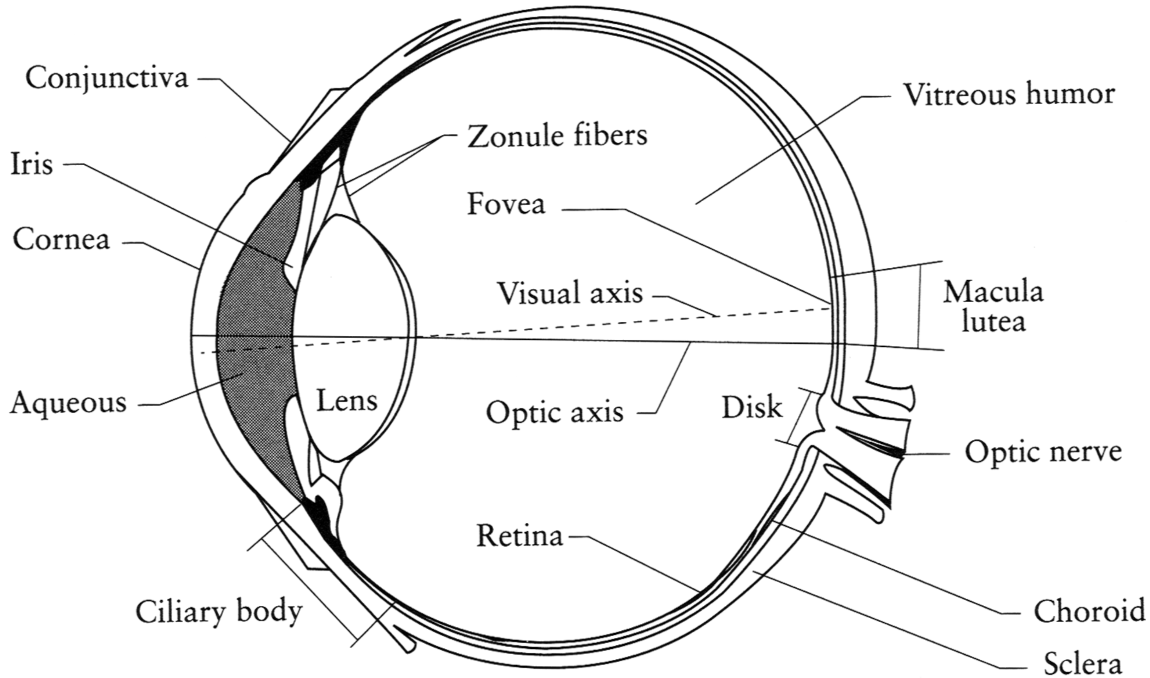


Lenses



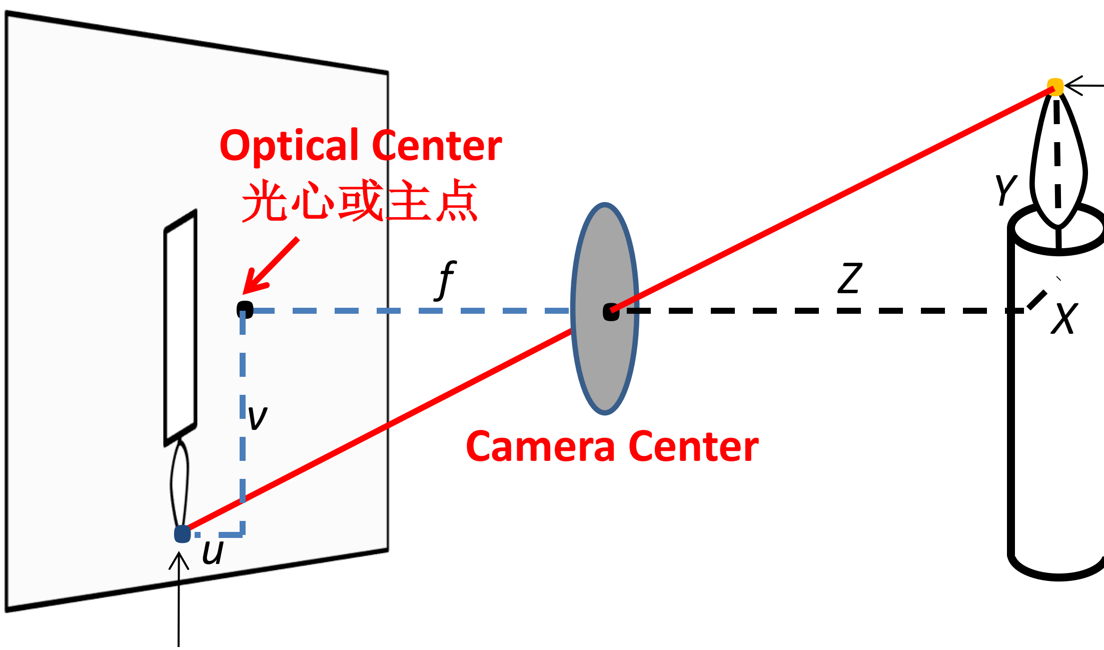
- A lens focuses parallel rays onto a single focal point
  - Focal length (焦距): focal point at a distance *f* beyond the plane of the lens (*f* is a function of the shape and index of refraction of the lens)
  - Aperture (光圈): restricts the range of rays
  - Optical axis (光轴)

The eye



- The human eye is a camera
  - **Lens**（晶状体）
  - **Iris**（虹膜）
  - **Pupil**（瞳孔）
  - **Retina**（视网膜）

Math for Pin-hole camera:
3D world coordinates → 2D image coordinates



$$\mathbf{P} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$\mathbf{p} = \begin{bmatrix} u \\ v \end{bmatrix}$$

How does focal length affect image?

Focal length

- Can think of as "zoom"

24mm

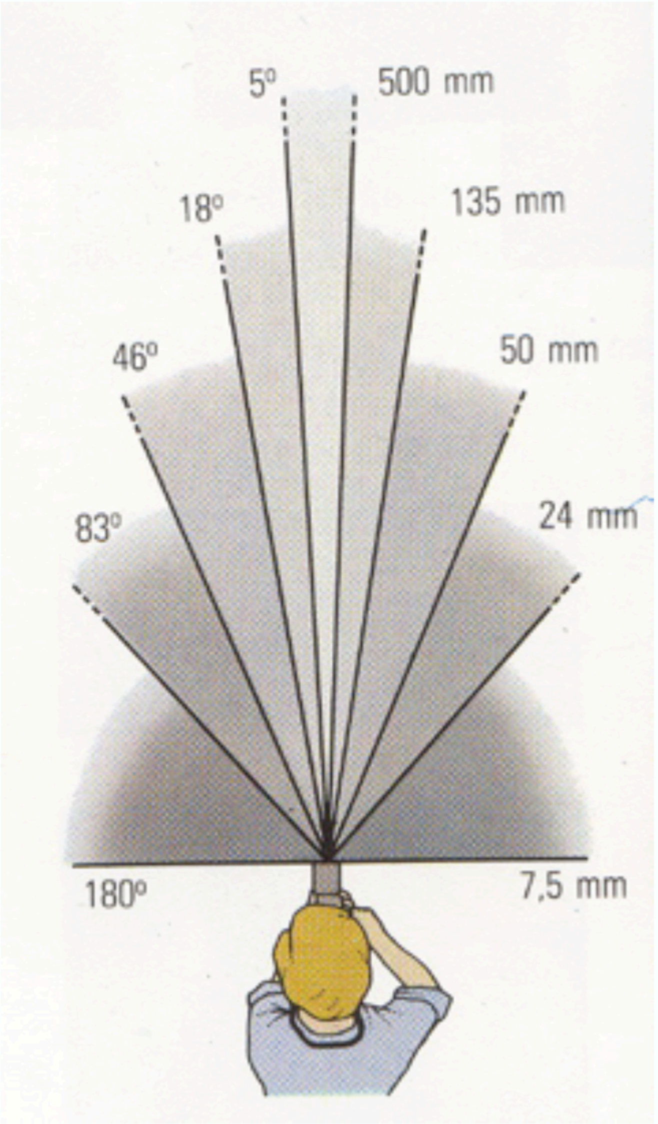
50mm

200mm

800mm

- Also related to *field of view*

Focal length in practice



24mm

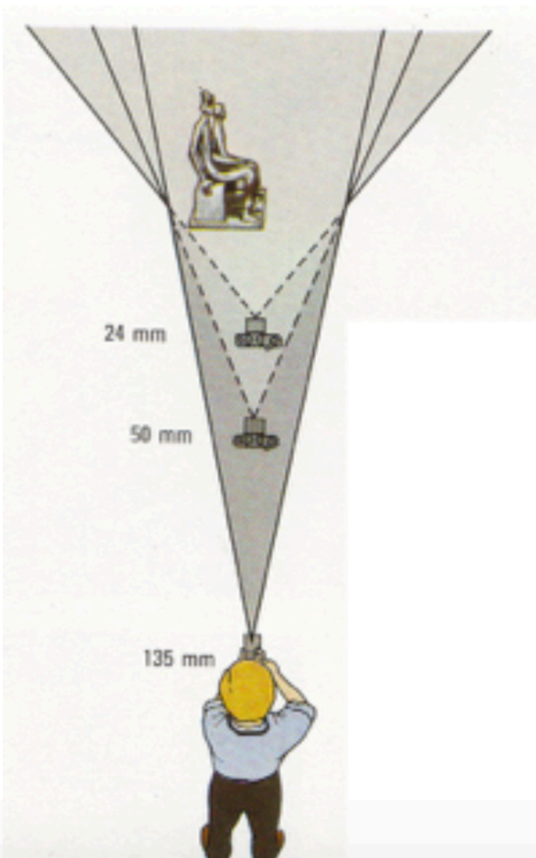
50mm

135mm

Fredo Durand

Focal length vs. viewpoint

- **Telephoto makes it easier to select background (a small change in viewpoint is a big change in background.**



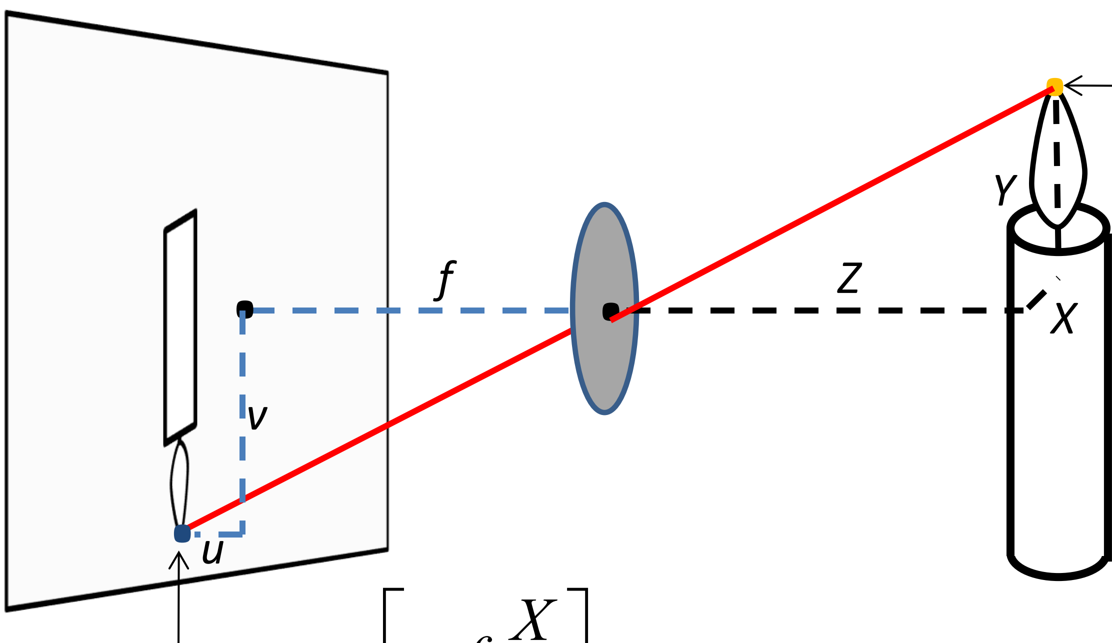
Grand-angulaire 24 mm

Normal 50 mm

Longue focale 135 mm

Fredo Durand

Perspective Projection: 3D world coordinates → 2D image coordinates



$$\mathbf{P} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$\mathbf{p} = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -f\,\dfrac{X}{Z} \\ -f\,\dfrac{Y}{Z} \end{bmatrix}$$

Is this a linear transformation?

Homogeneous coordinates

Converting to homogeneous coordinates

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad (x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Converting from homogeneous coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w) \qquad \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \Rightarrow (x/w, y/w, z/w)$$

Homogeneous coordinates are invariant to scaling

Perspective Projection in homogeneous coordinates

- Projection is a matrix multiplication using homogeneous coordinates

$$
\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} fx \\ fy \\ z \end{bmatrix} \cong \begin{bmatrix} f\frac{x}{z} \\ f\frac{y}{z} \\ 1 \end{bmatrix}
$$

Camera parameters

Assumptions

- Optical center at (0,0)
- Unit aspect ratio
- No skew

$$w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Camera parameters

Assumptions

- ~~Optical center at (0,0)~~
- Unit aspect ratio
- No skew

$$w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & u_0 & 0 \\ 0 & f & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Slide Credit: Saverese

Camera parameters

Assumptions

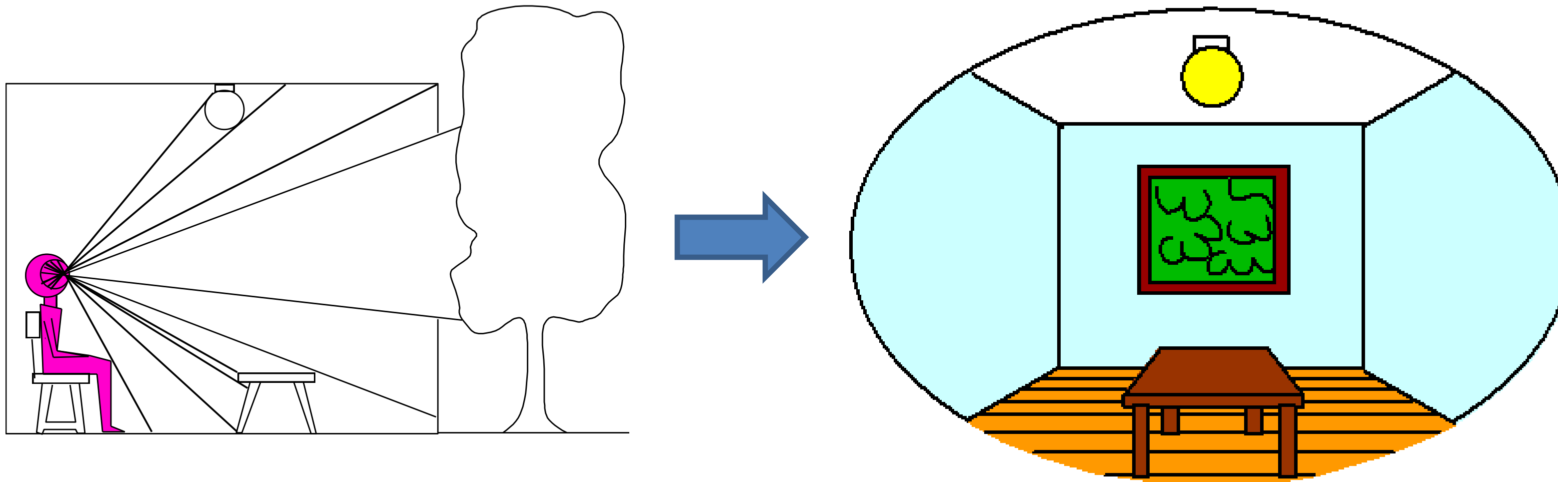
- ~~Optical center at (0,0)~~
- ~~Unit aspect ratio~~
- ~~No skew~~

$$w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & s & u_0 & 0 \\ 0 & \beta & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Slide Credit: Saverese

Dimensionality Reduction Machine (3D to 2D)

3D world *2D image*



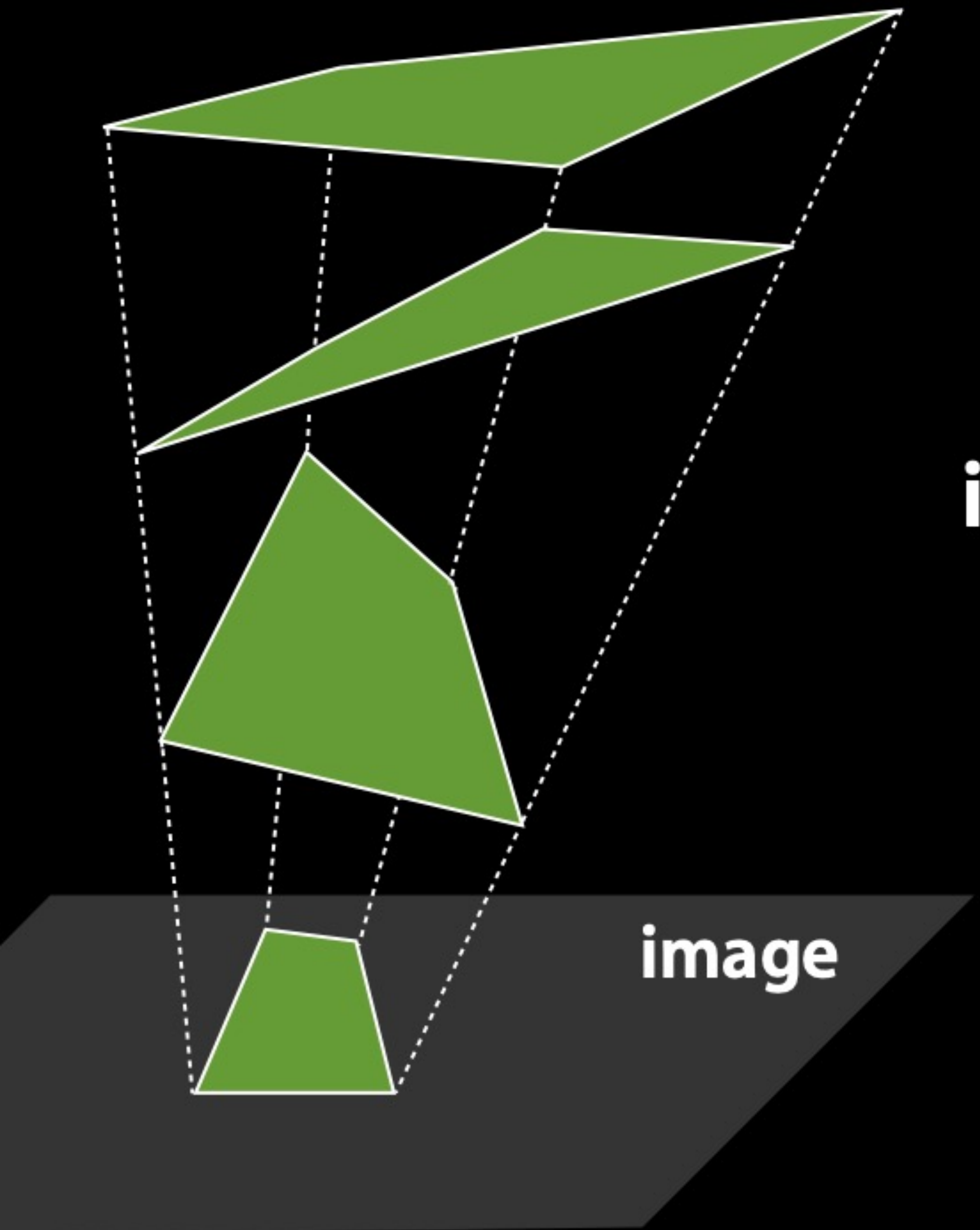
Point of observation

Projection can be tricky…

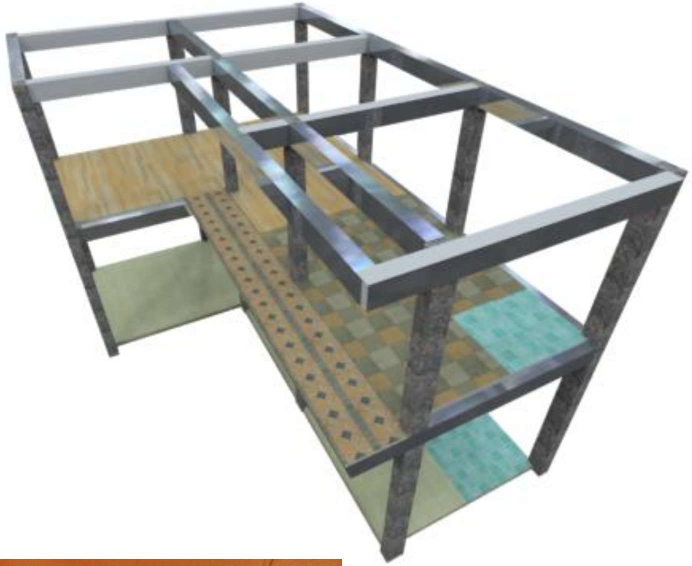
Slide source: Seitz

Projection can be tricky…

Making of 3D sidewalk art: http://www.youtube.com/watch?v=3SNYtd0Ayt0 infinite number of possible shapes



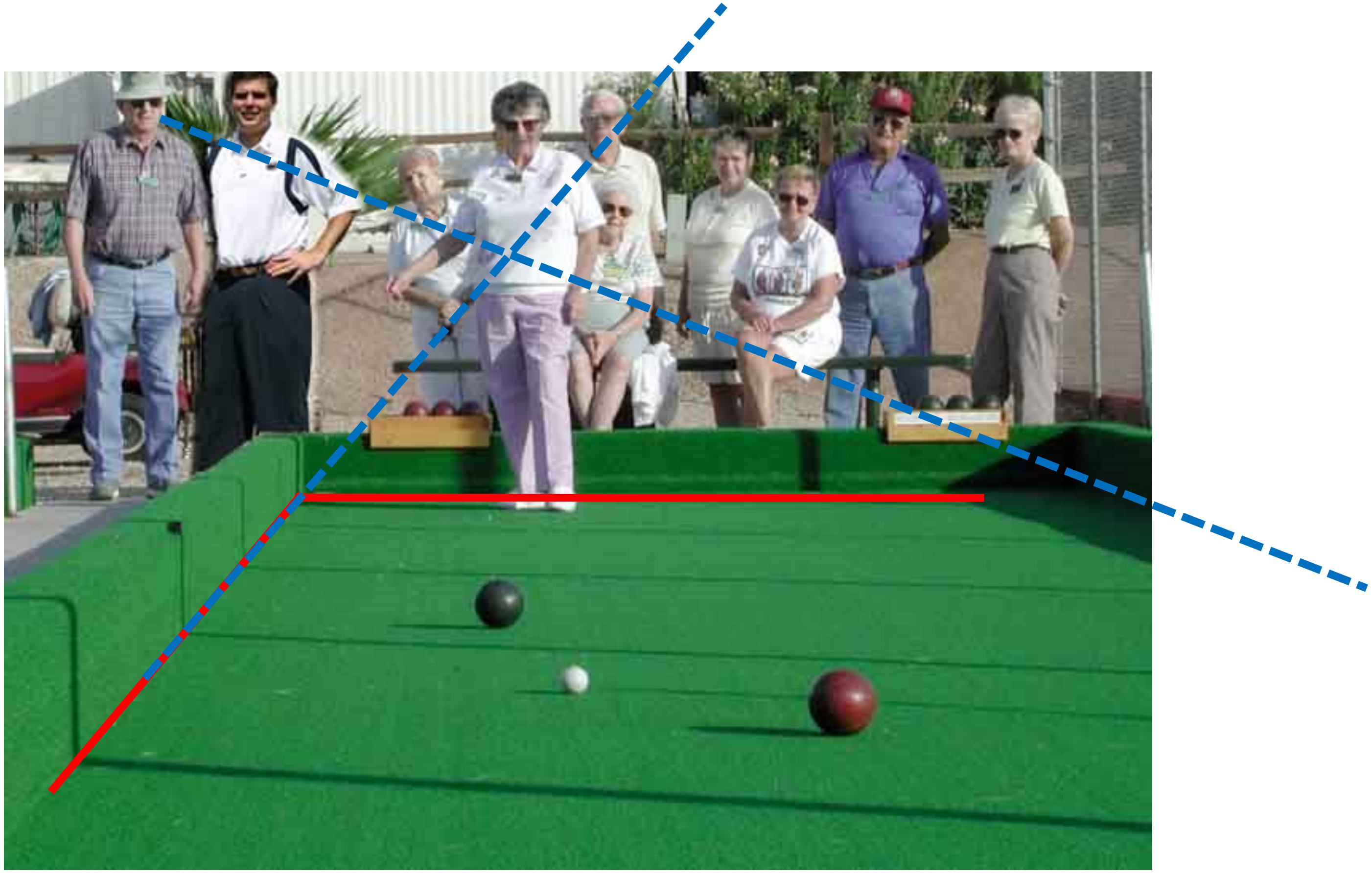
Perspective effect



Projective Geometry

What is preserved?

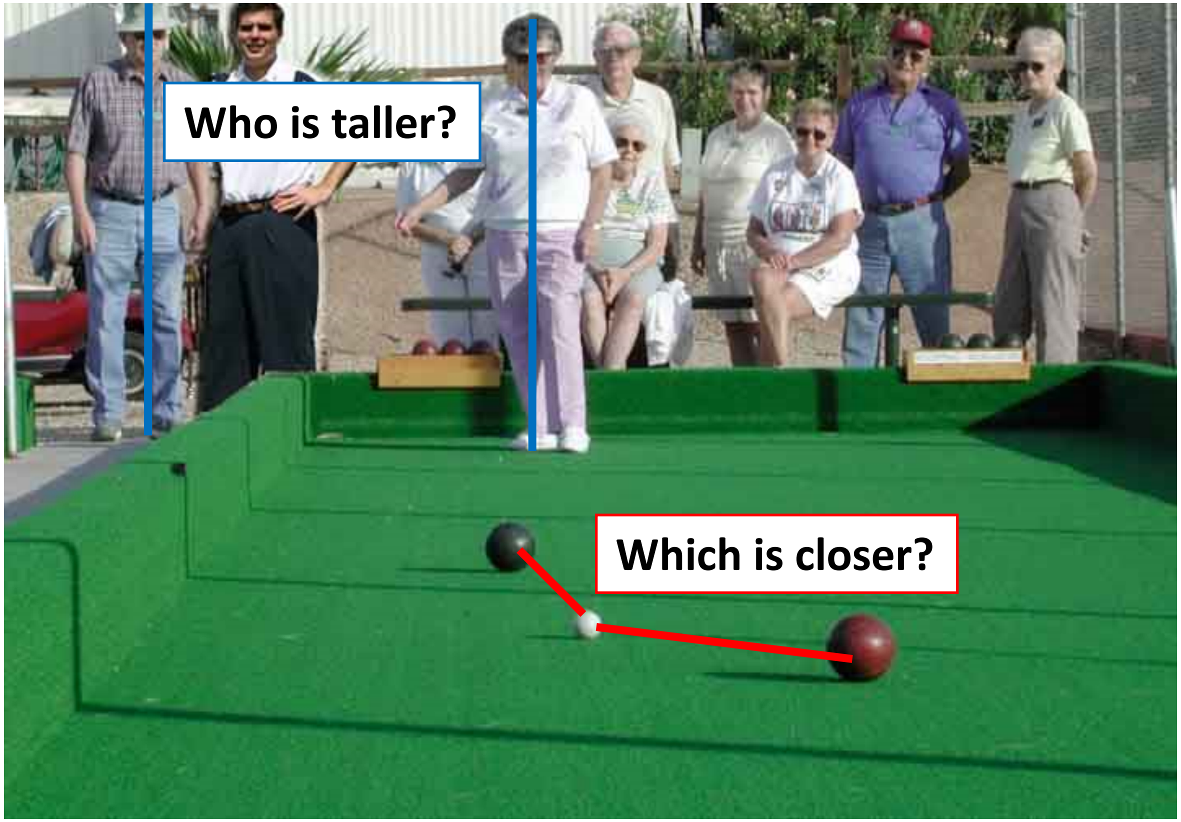
- Straight lines are still straight



Projective Geometry

What is lost?

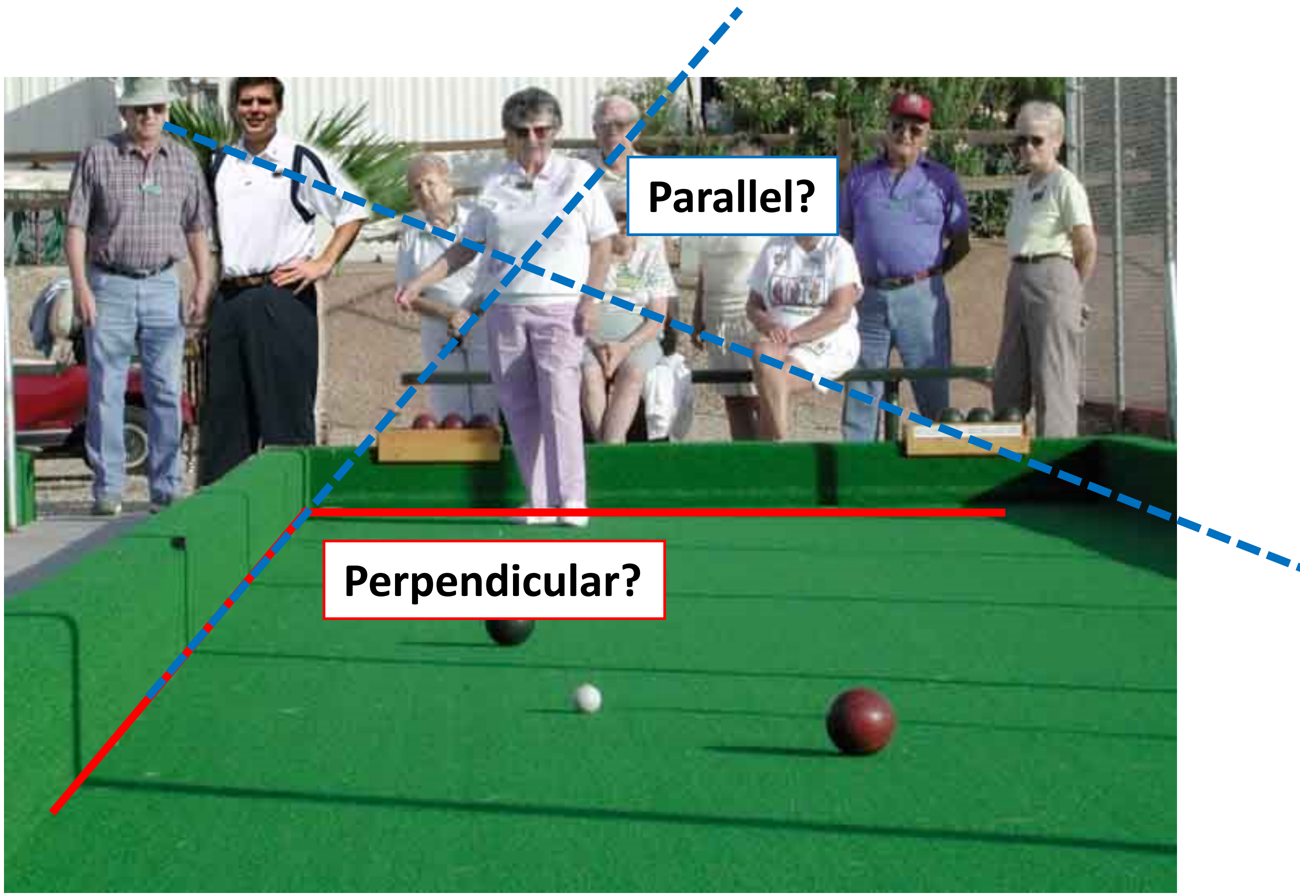
- Length



Projective Geometry

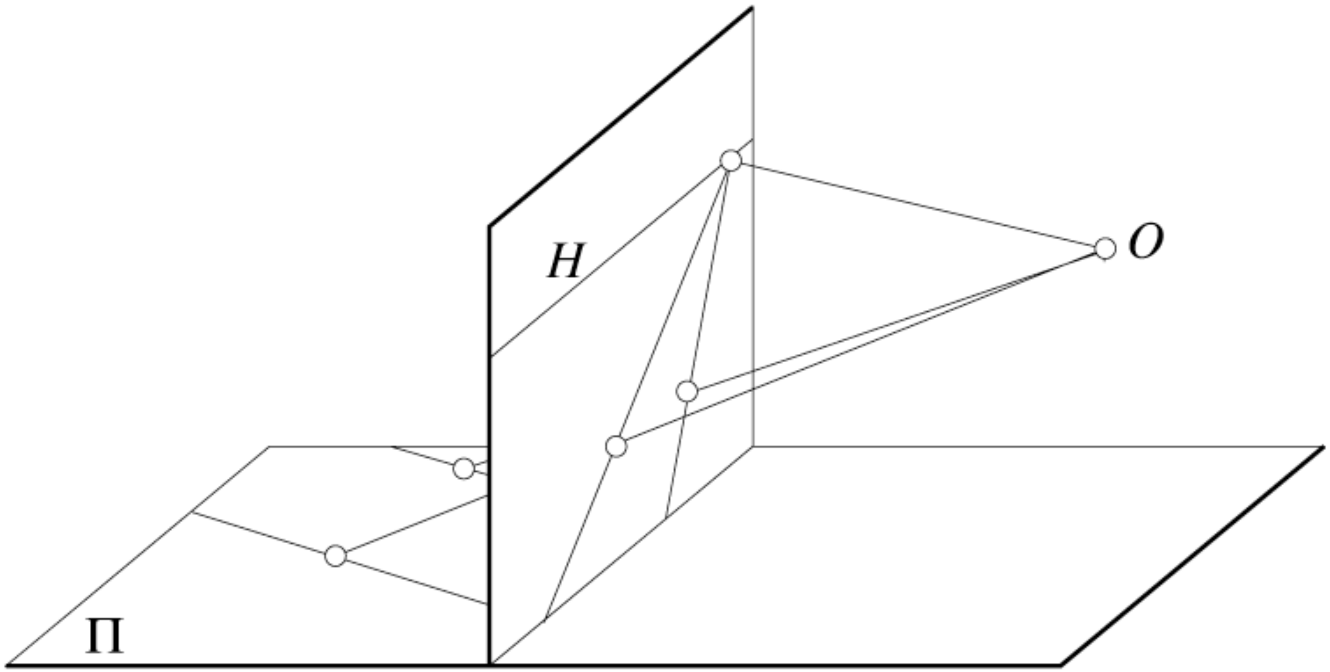
What is lost?

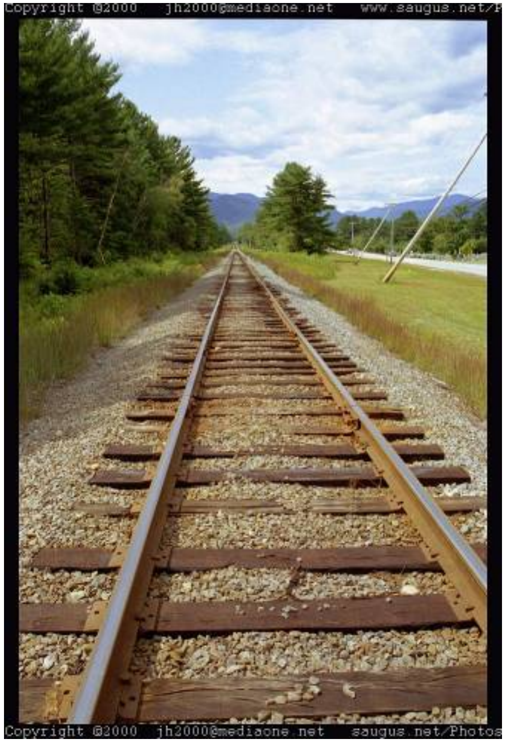
- Length
- Angles



Projection properties

- Parallel lines converge at vanishing point (灭点)
  - Each direction in space has its own vanishing point
  - But parallels parallel to the image plane remain parallel





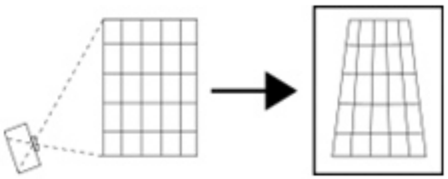
Perspective distortion

- Problem for architectural photography: converging verticals
- The distortion is not due to lens flaws

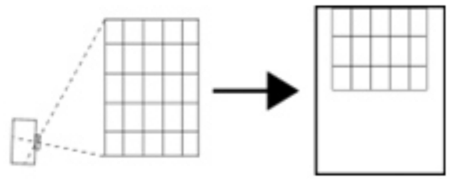
Source: F. Durand

Perspective distortion

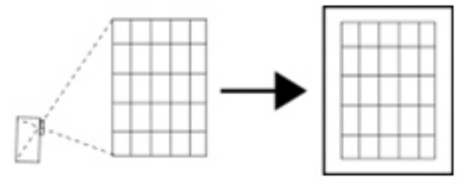
- Problem for architectural photography: converging verticals



Tilting the camera upwards results in converging verticals

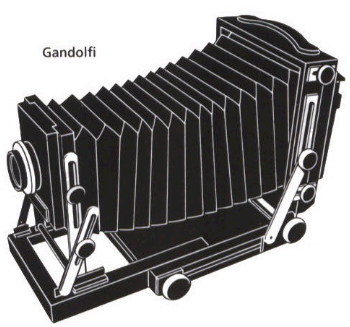


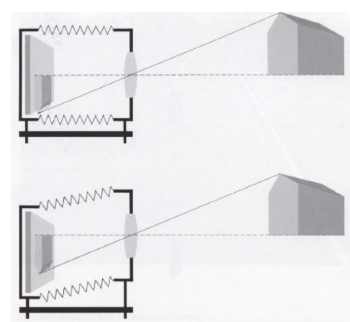
Keeping the camera level, with an ordinary lens, captures only the bottom portion of the building



Shifting the lens upwards results in a picture of the entire subject

- Solution: view camera (lens shifted w.r.t. film)





http://en.wikipedia.org/wiki/Perspective_correction_lens

Source: F. Durand

Perspective distortion

- What does a sphere project to?

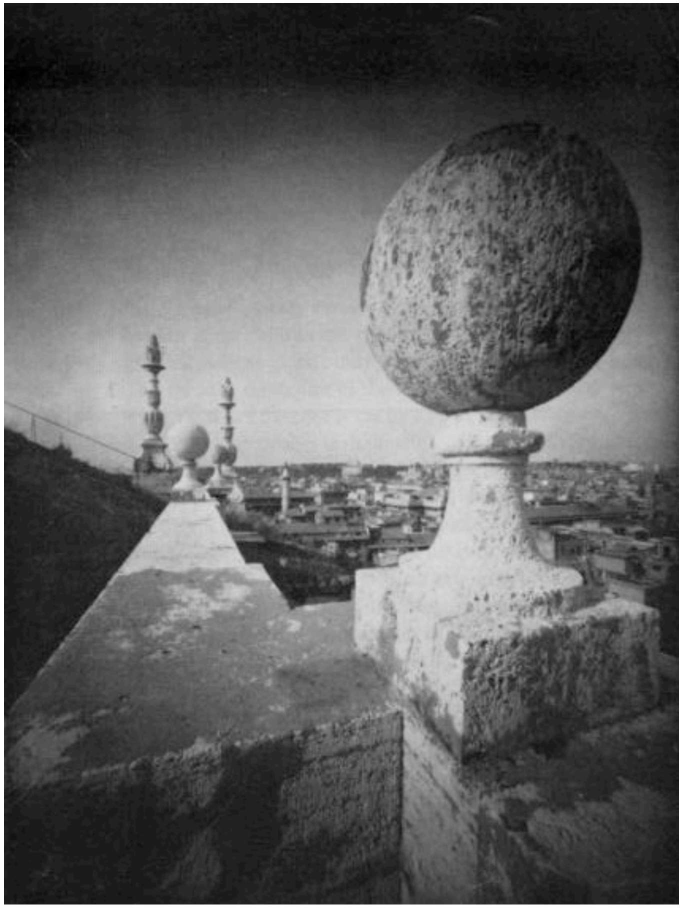
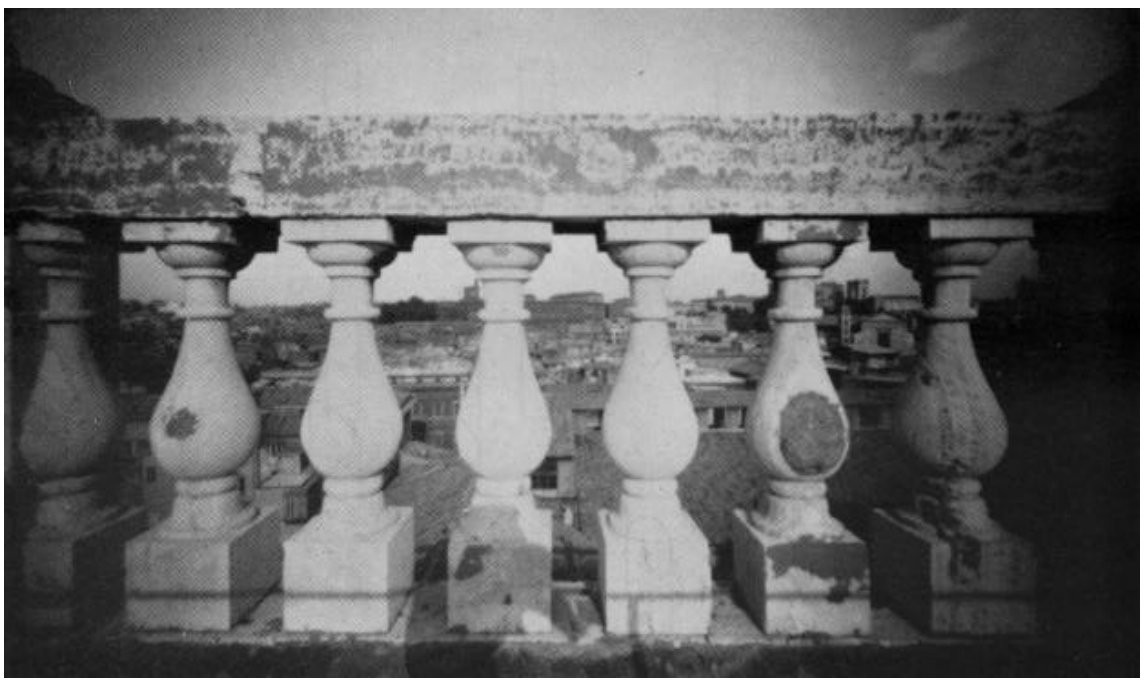
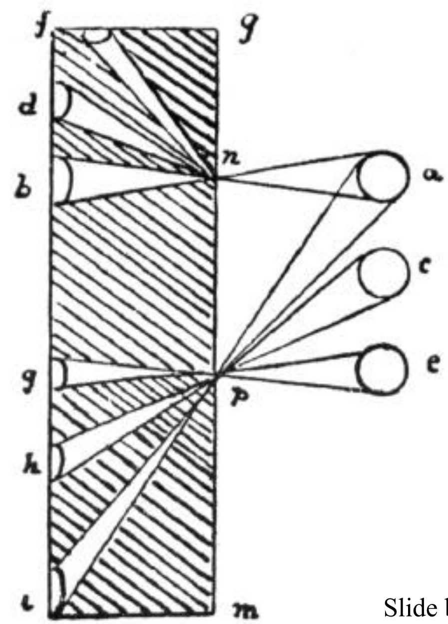


Image source: F. Durand

Perspective distortion

- The exterior columns appear bigger
- Problem pointed out by Da Vinci

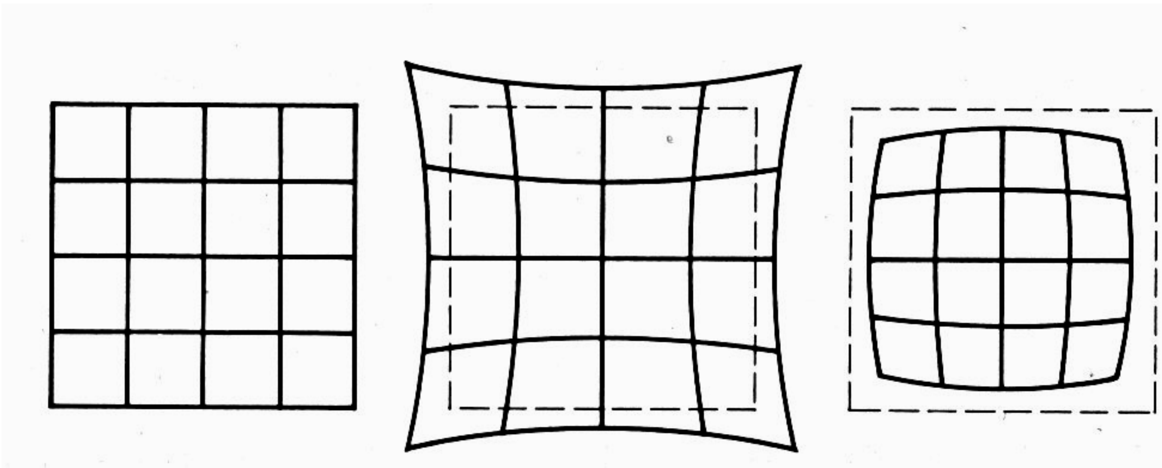




Slide by F. Durand

Perspective distortion: People

Radial distortion



No distortion Pin cushion Barrel

- Radial distortion of the image
  - Caused by imperfect lenses
  - Deviations are most noticeable for rays that pass through the edge of the lens


©2004 Vincent Bockaert 123di.com


©2004 Vincent Bockaert 123di.com

Wide angle

Standard

Telephoto

http://petapixel.com/2013/01/11/how-focal-length-affects-your-subjects-apparent-weight-as-seen-with-a-cat/

Modeling distortion

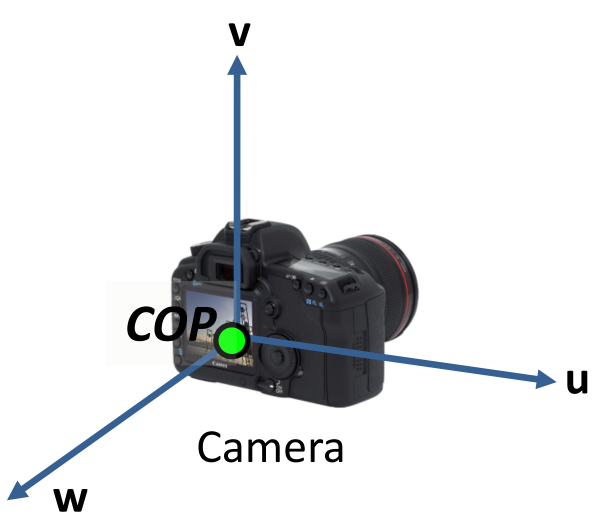
$$
\begin{aligned}
r^2 &= {x'_n}^2 + {y'_n}^2 \\
x'_d &= x'_n(1 + \kappa_1 r^2 + \kappa_2 r^4) \\
y'_d &= y'_n(1 + \kappa_1 r^2 + \kappa_2 r^4)
\end{aligned}
$$

Correcting radial distortion

from Helmut Dersch

Camera coordinates

- How can we model the viewpoint of a camera?



Camera

Two important coordinate systems:

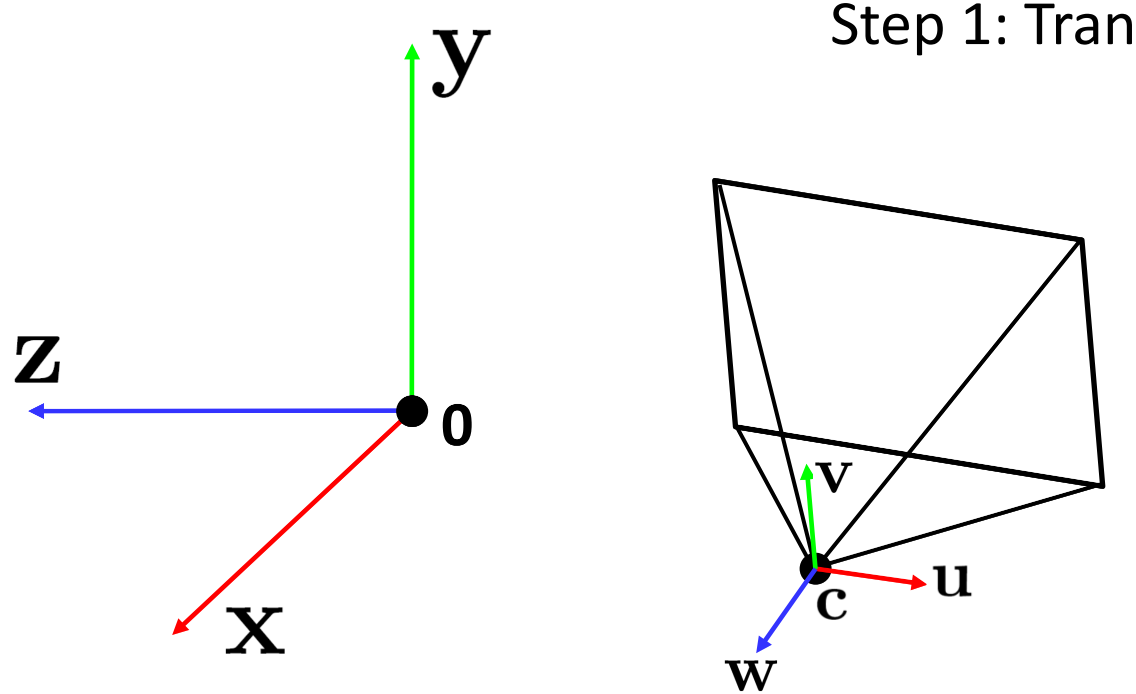
1. *World* coordinate system
2. *Camera* coordinate system

“The World”

Extrinsic parameters

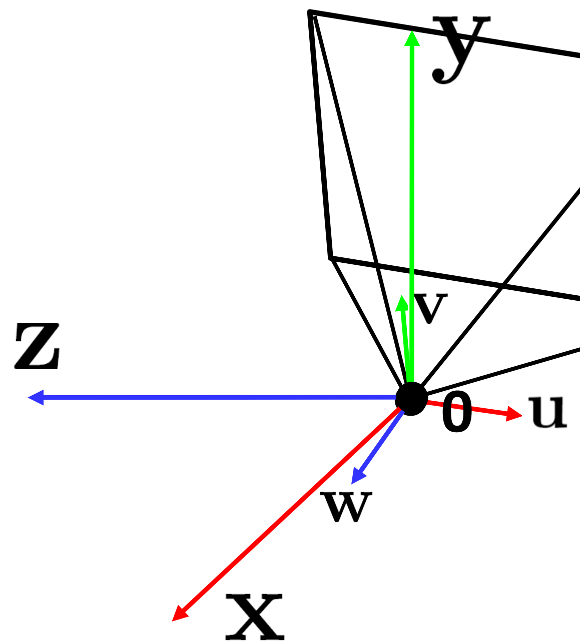
- How do we align two coordinate systems?

Step 1: Translate by -**c**



Extrinsic parameters

- How do we align two coordinate systems?



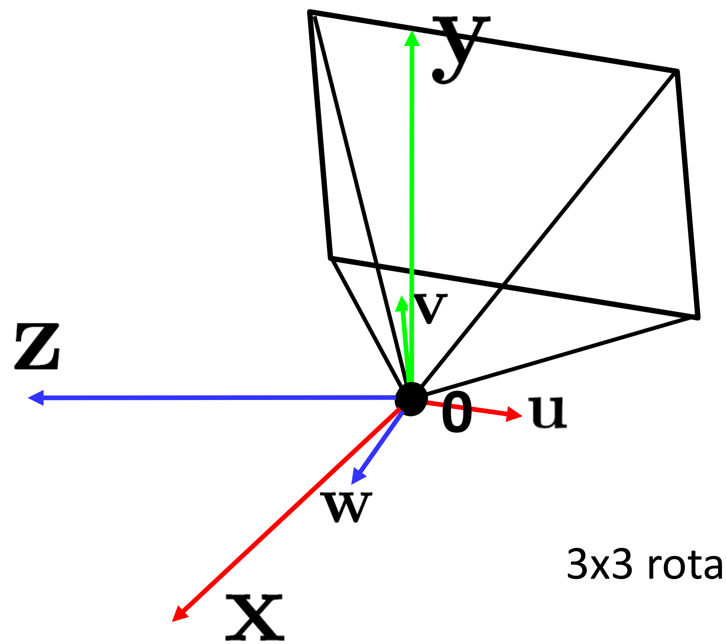
Step 1: Translate by -**c**

How do we represent translation as a matrix multiplication?

$$\mathbf{T} = \begin{bmatrix} \mathbf{I}_{3\times 3} & -\mathbf{c} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}$$

Extrinsic parameters

- How do we align two coordinate systems?



Step 1: Translate by -**c**

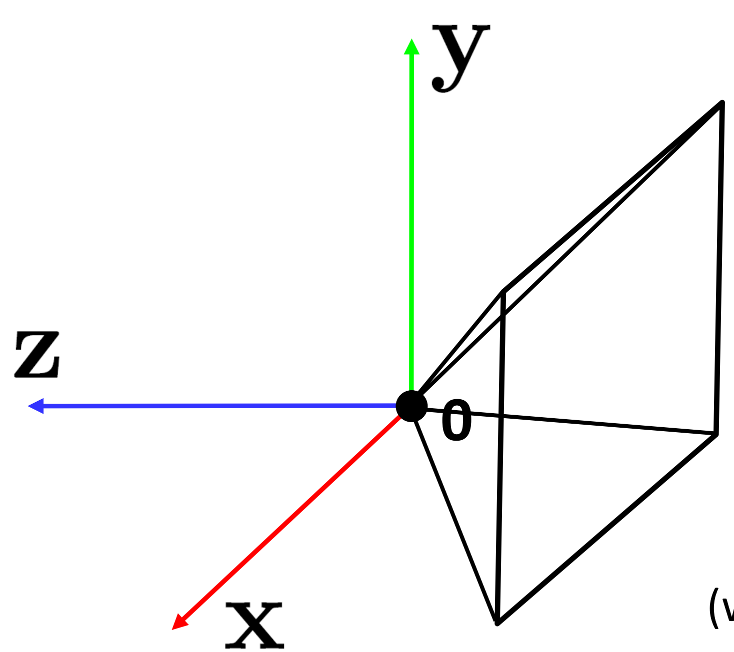
Step 2: Rotate by **R**

$$\mathbf{R} = \begin{bmatrix} \mathbf{u}^T \\ \mathbf{v}^T \\ \mathbf{w}^T \end{bmatrix}$$

3x3 rotation matrix

Extrinsic parameters

- How do we align two coordinate systems?



Step 1: Translate by -**c**

Step 2: Rotate by **R**

$$\mathbf{R} = \begin{bmatrix} \mathbf{u}^T \\ \mathbf{v}^T \\ \mathbf{w}^T \end{bmatrix}$$

(with extra row/column of [0 0 0 1])

Extrinsic parameters

- Rigid transformation in 3D

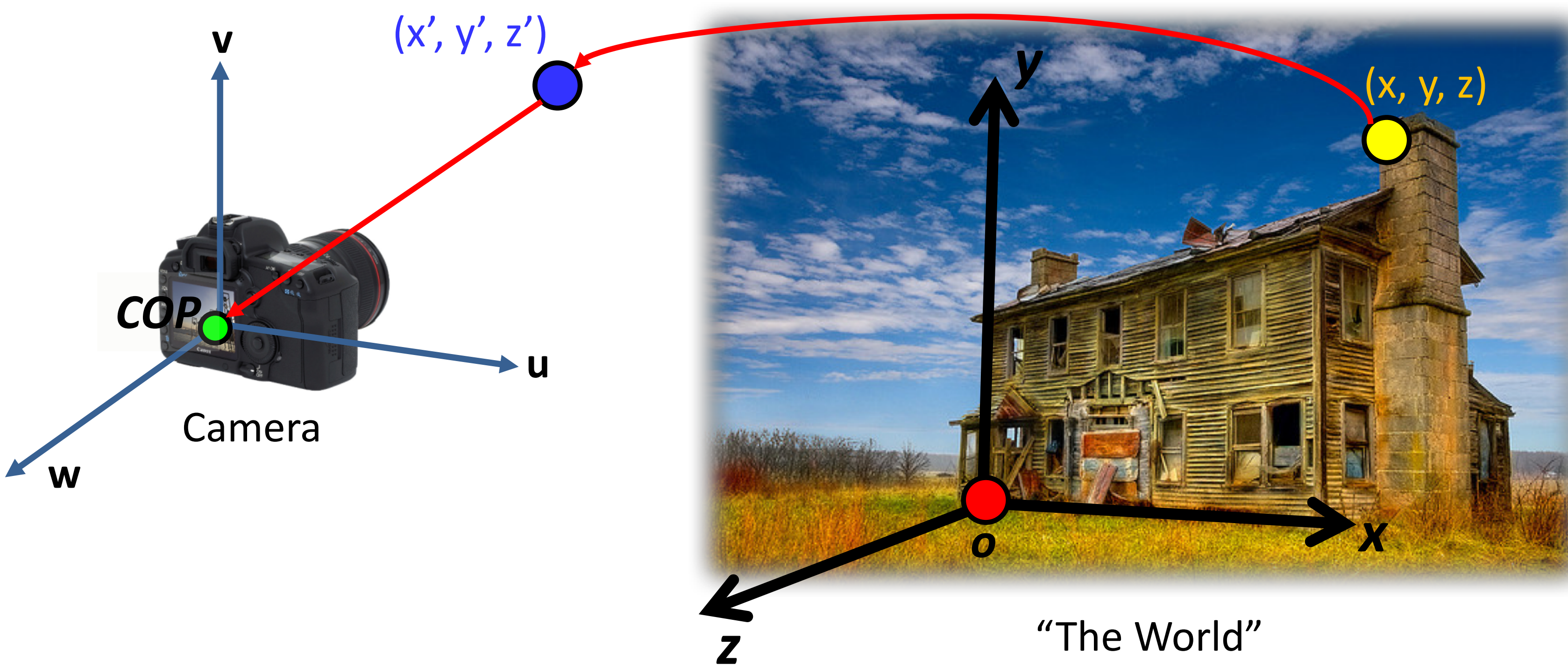
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Camera parameters

How to project a point (x,y,z) in *world* coordinates into a camera?

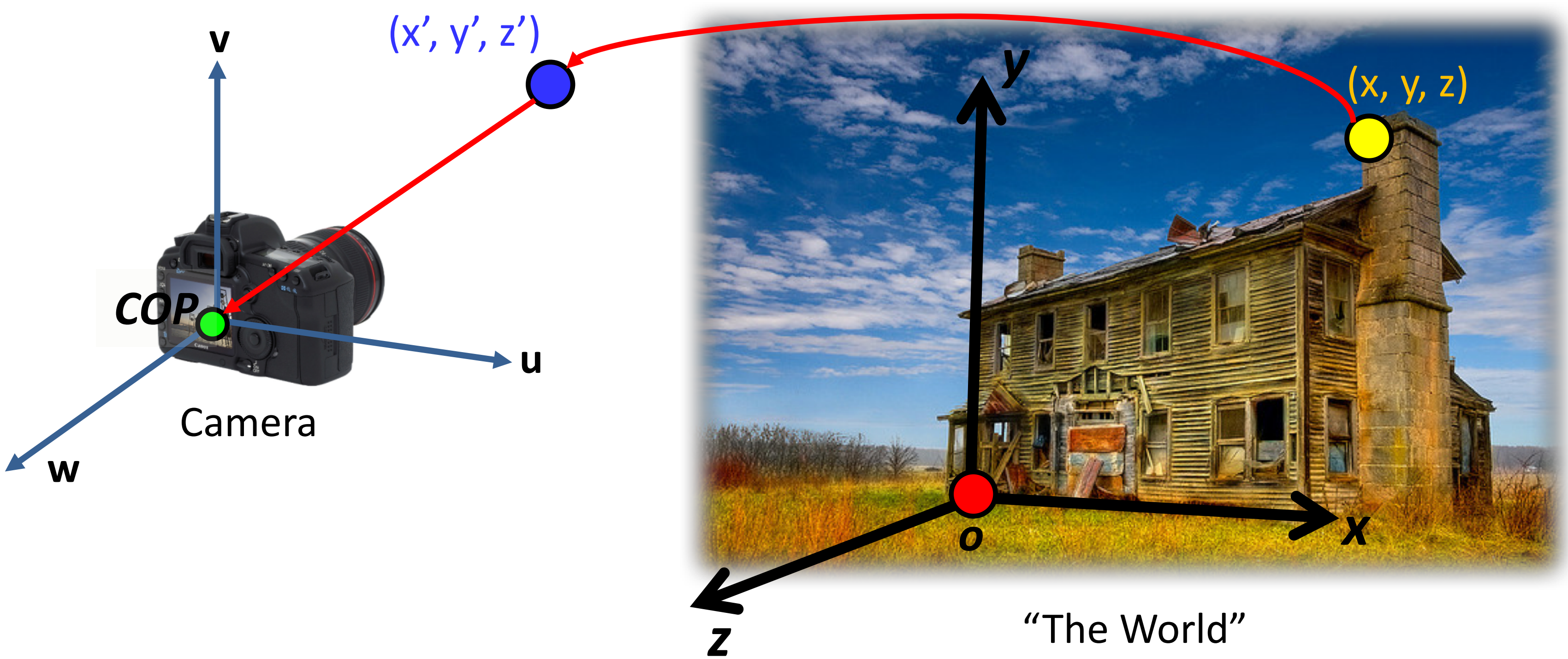
- First transform (x,y,z) into *camera* coordinates
  - Need to know camera extrinsic parameters（外参）
- Then project into the image plane to get a pixel coordinate
  - Need to know camera intrinsic parameters（内参）

Perspective Projection Matrix

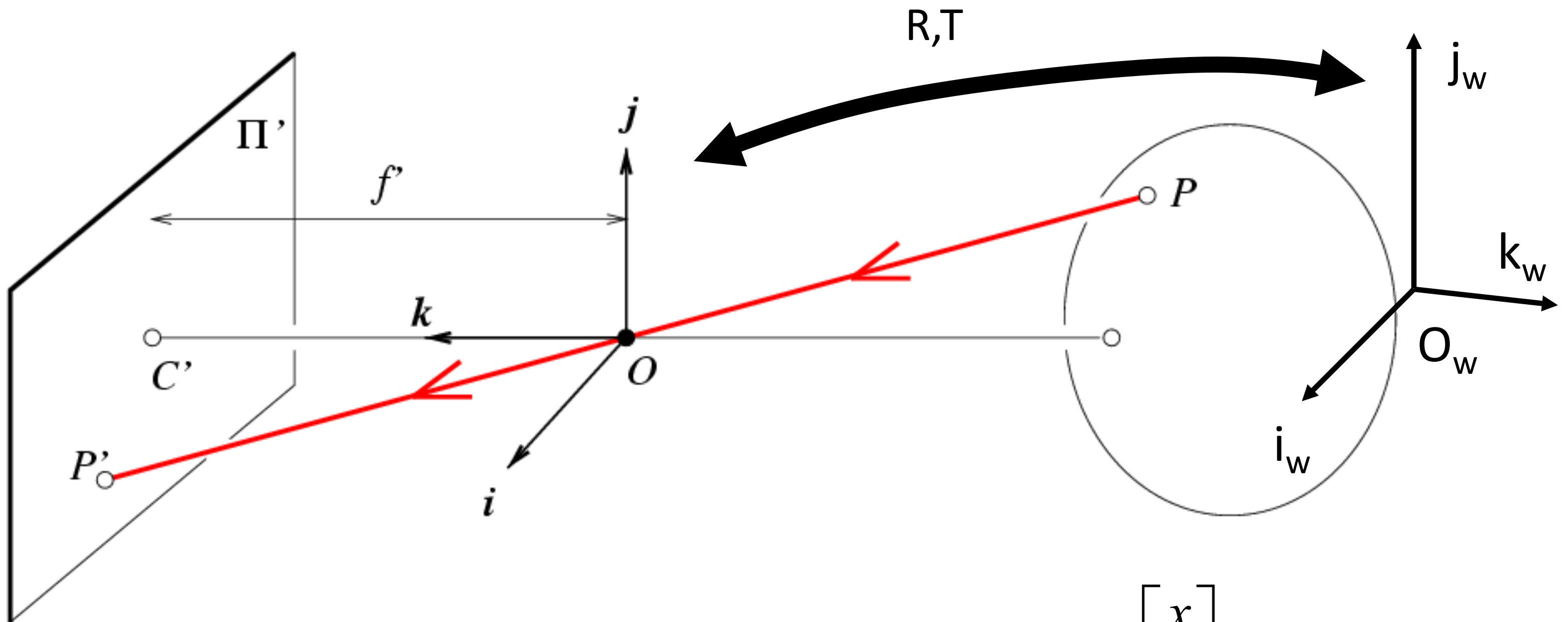


$$\begin{pmatrix}\text{2D}\\ \text{point}\\ \text{(3x1)}\end{pmatrix} = \underbrace{\begin{pmatrix}\text{Perspective}\\ \text{projection matrix}\\ \text{(3x4)}\end{pmatrix}}_{\text{Camera intrincis}} \underbrace{\begin{pmatrix}\text{World to}\\ \text{camera coord.}\\ \text{trans. matrix}\\ \text{(4x4)}\end{pmatrix}}_{\text{Camera extrinsics}} \begin{pmatrix}\text{3D}\\ \text{point}\\ \text{(4x1)}\end{pmatrix}$$

Perspective Projection Matrix



$$
w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & s & u_0 & 0 \\ 0 & \beta & u_v & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}
$$



$$w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\mathbf{x} = \mathbf{K}\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}\mathbf{X}$$

x: Image Coordinates: (u,v,1)
K: Intrinsic Matrix (3x3)
R: Rotation (3x3)
t: Translation (3x1)
X: World Coordinates: (X,Y,Z,1)

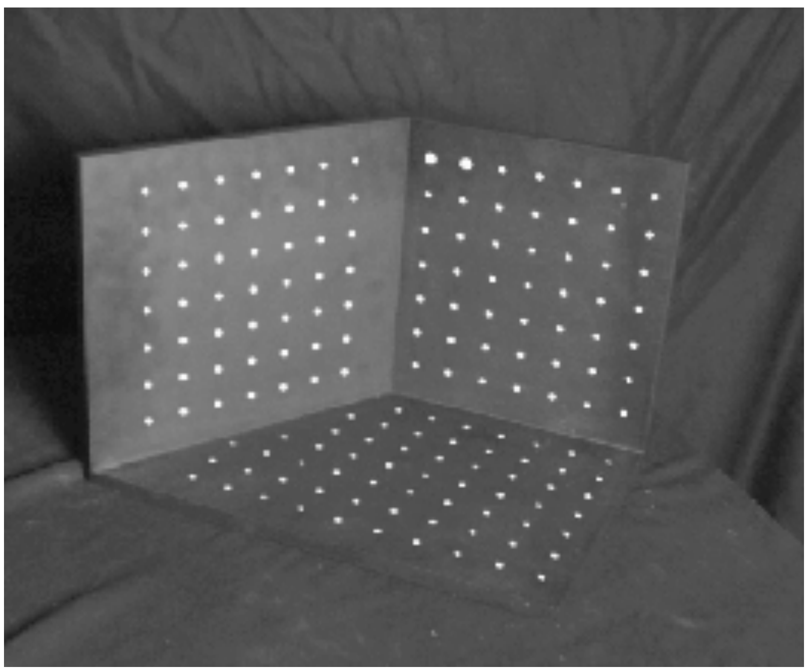
Camera calibration: how to obtain the camera parameters?

$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

$$\begin{bmatrix} wu \\ wv \\ w \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Calibrating the Camera

Use an object with known geometry
(calibration grid)



Known 2d image coordinates

Known 3d locations

$$\begin{bmatrix} wu \\ wv \\ w \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Unknown Camera Parameters

Unknown Camera Parameters

Known 2d image coords

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Known 3d locations

$$0 = m_{11}X + m_{12}Y + m_{13}Z + m_{14} - m_{31}uX - m_{32}uY - m_{33}uZ - m_{34}u$$
$$0 = m_{21}X + m_{22}Y + m_{23}Z + m_{24} - m_{31}vX - m_{32}vY - m_{33}vZ - m_{34}v$$

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 & -u_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_1 & -v_1 \\ & & & & & & \vdots & & & & & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_nX_n & -u_nY_n & -u_nZ_n & -u_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_nX_n & -v_nY_n & -v_nZ_n & -v_n \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \\ m_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

Can we factorize M back to K [R | T]?

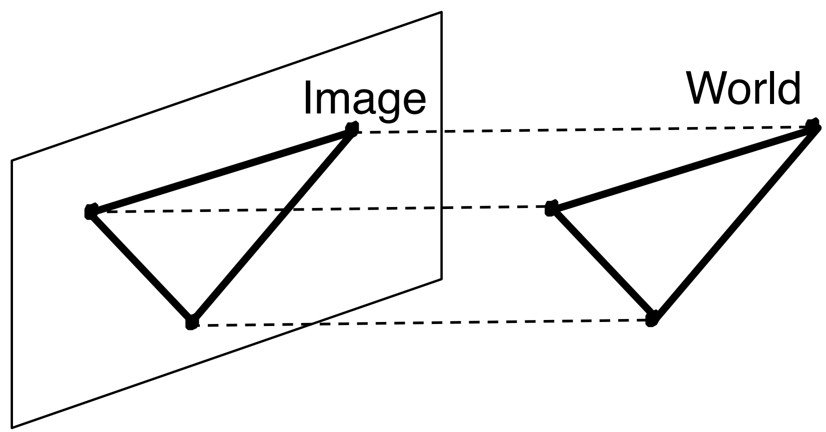
- Yes!
- You can use *RQ* factorization
  - (note – not the more familiar *QR* factorization).

- *R* (right diagonal) is K, and *Q* (orthogonal basis) is R. T, the last column of [R | T], is inv(K) * last column of M.
  - Need post-processing to make sure that the matrices are valid.
  - See http://ksimek.github.io/2012/08/14/decompose/

Slide credit: J. Hays

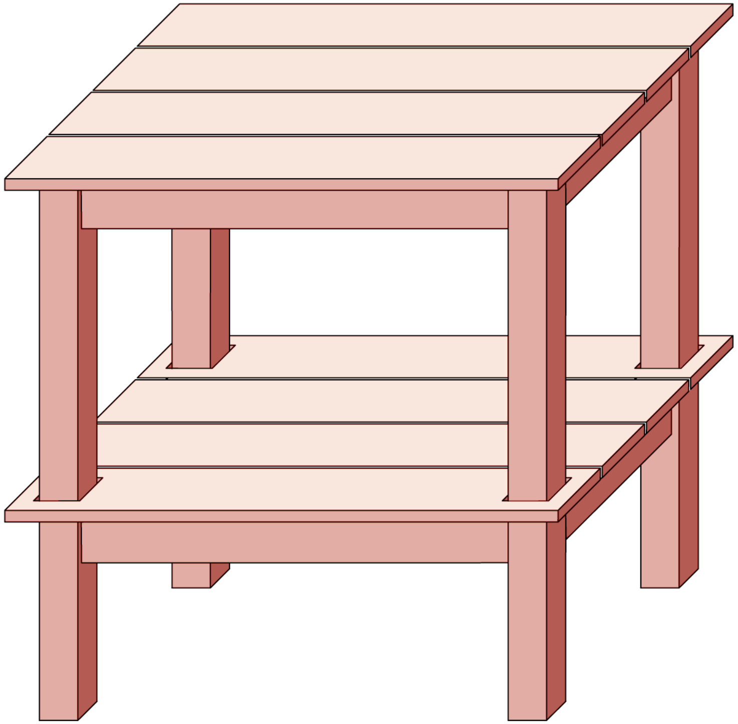
Orthographic projection

- Special case of perspective projection
  - Distance from the COP to the PP is infinite



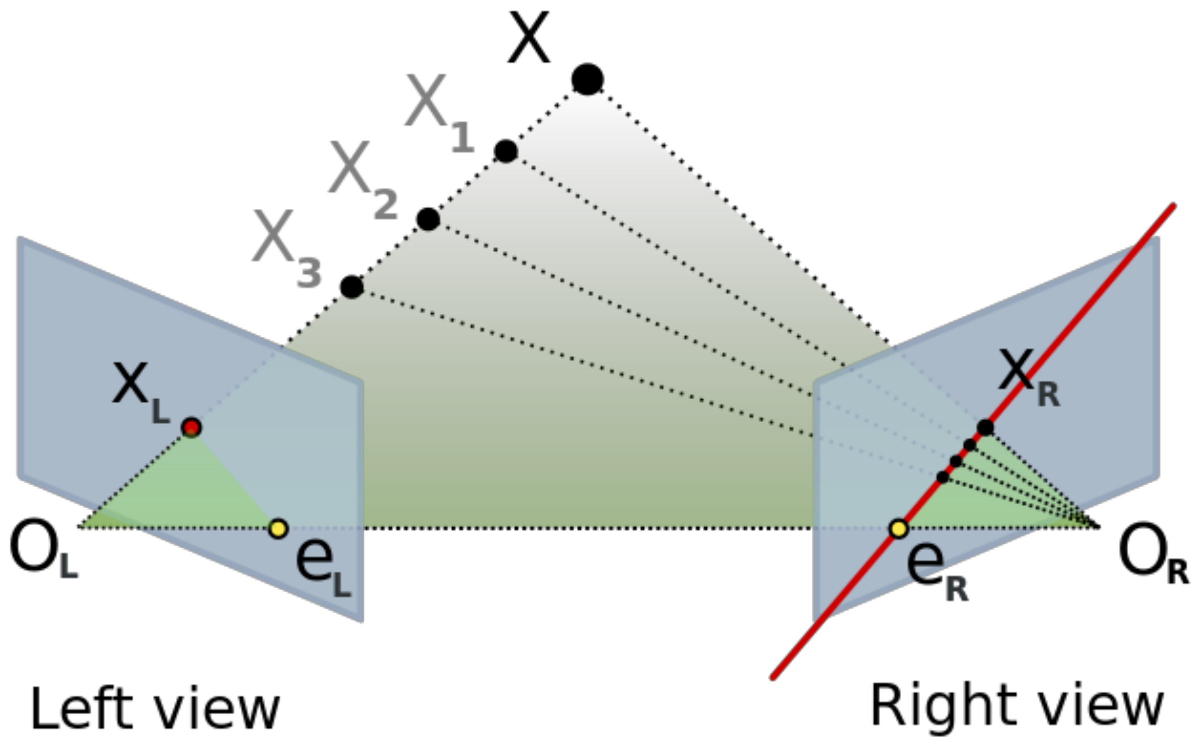
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow (x, y)$$

Orthographic projection



Questions?

Epipolar Geometry

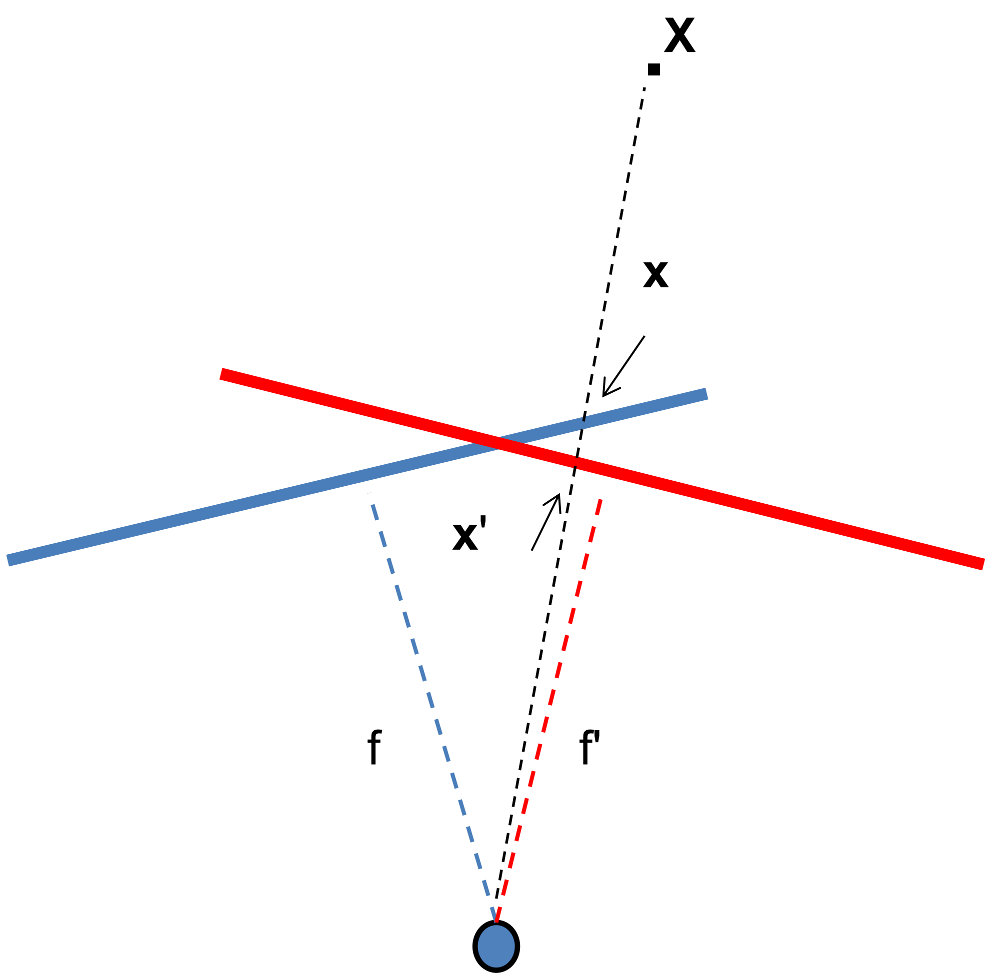


Two-View Geometry

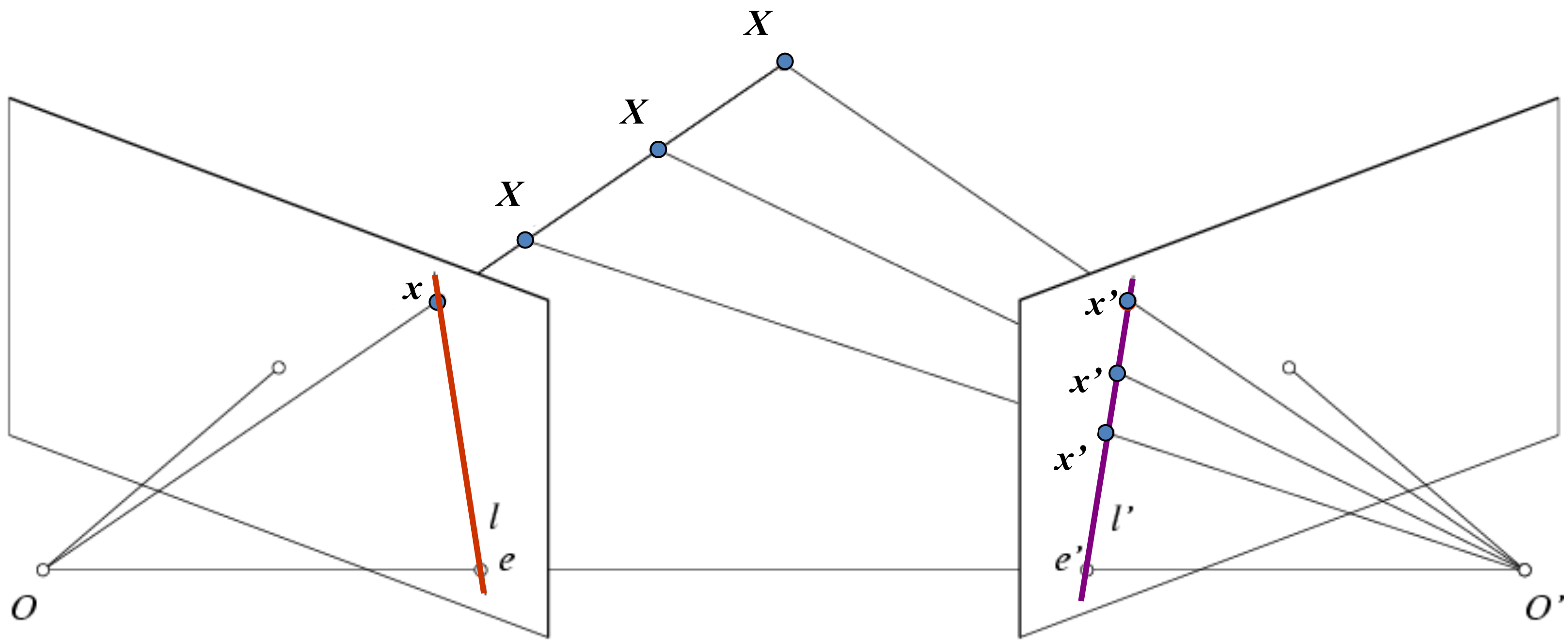
- Epipolar geometry
  - Relates two images taken from two positions
- Two-view reconstruction

Last class: Image Stitching

- Two images with rotation/zoom but no translation



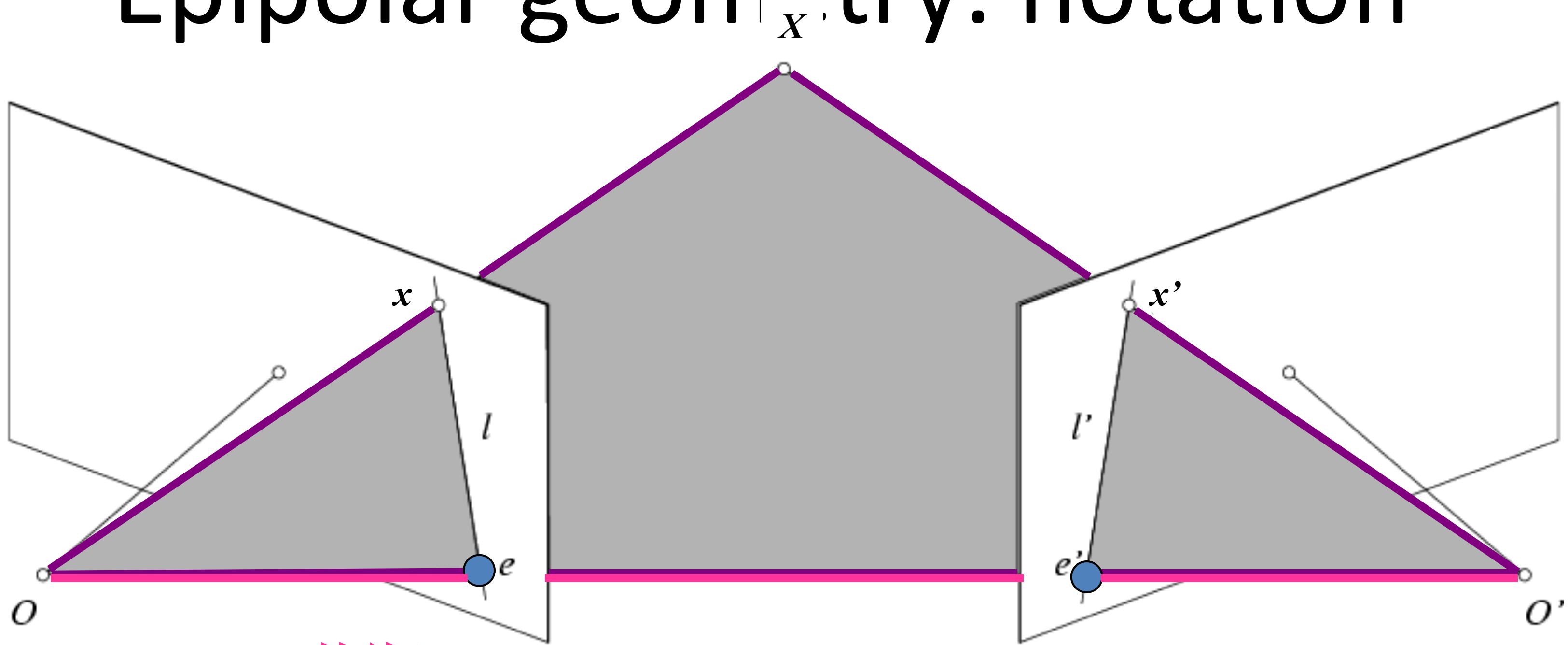
Key idea: Epipolar constraint



Potential matches for *x* have to lie on the corresponding line *l'*.

Potential matches for *x'* have to lie on the corresponding line *l*.

Epipolar geometry: notation



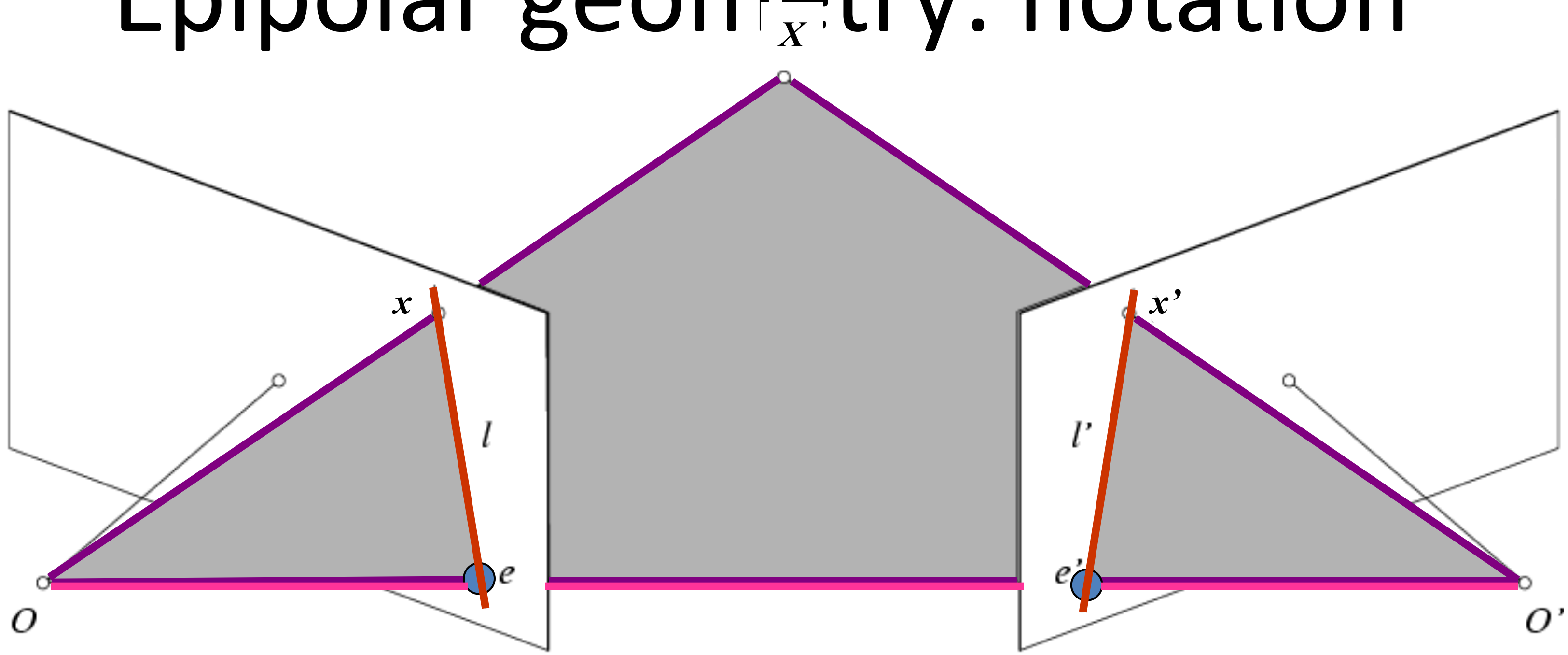
- **Baseline（基线）** – line connecting the two camera centers
- **Epipoles （极点）**

= intersections of baseline with image planes

= projections of the other camera center

- **Epipolar Plane（极平面）** – plane containing baseline (1D family)

Epipolar geometry: notation



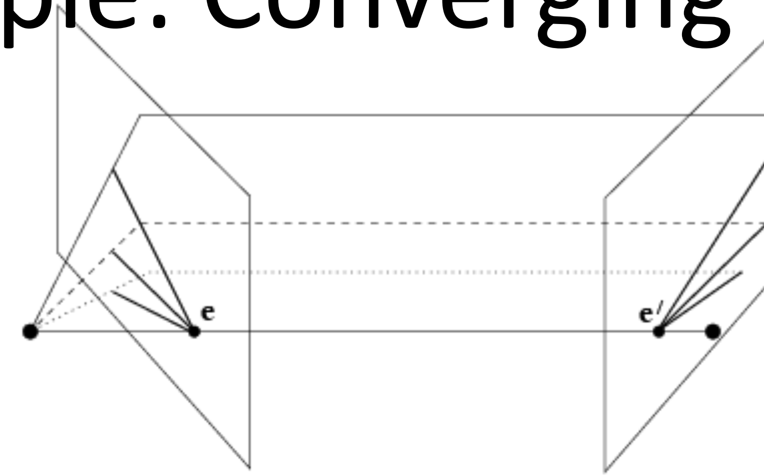
- **Baseline** – line connecting the two camera centers
- **Epipoles**

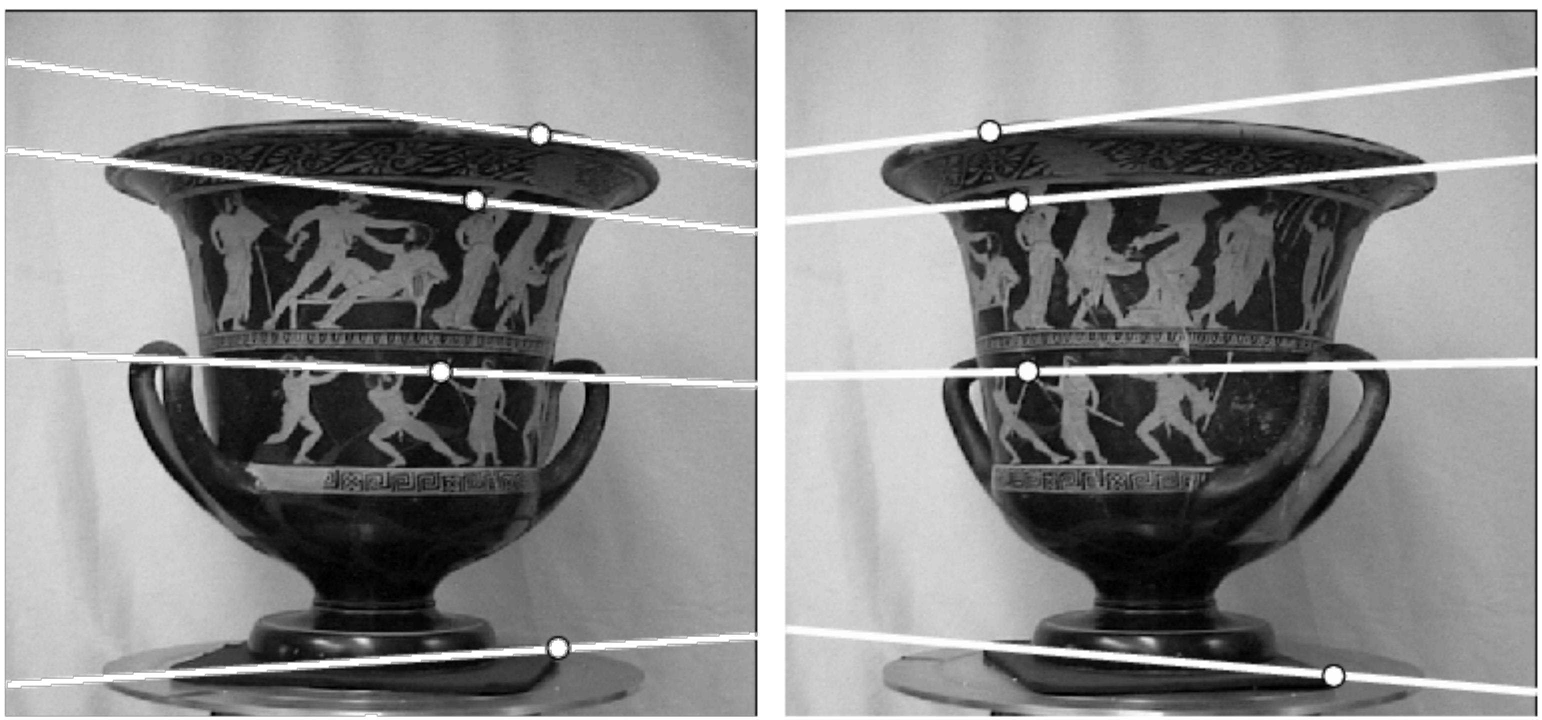
= intersections of baseline with image planes

= projections of the other camera center

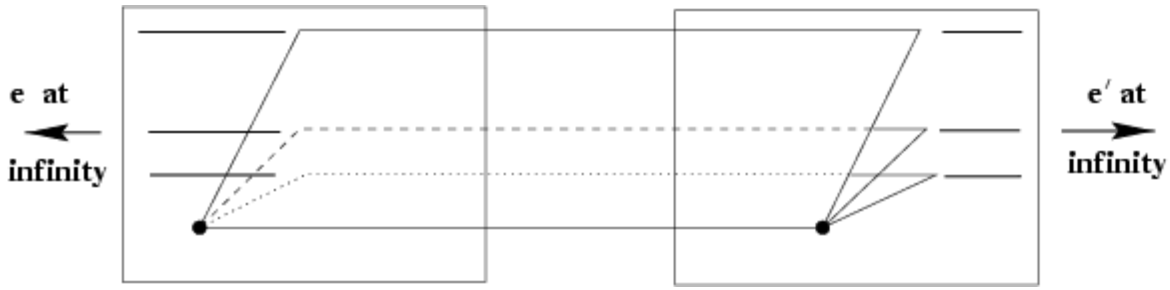
- **Epipolar Plane** – plane containing baseline (1D family)
- **Epipolar Lines**（极线）- intersections of epipolar plane with image planes (always come in corresponding pairs)

Example: Converging cameras





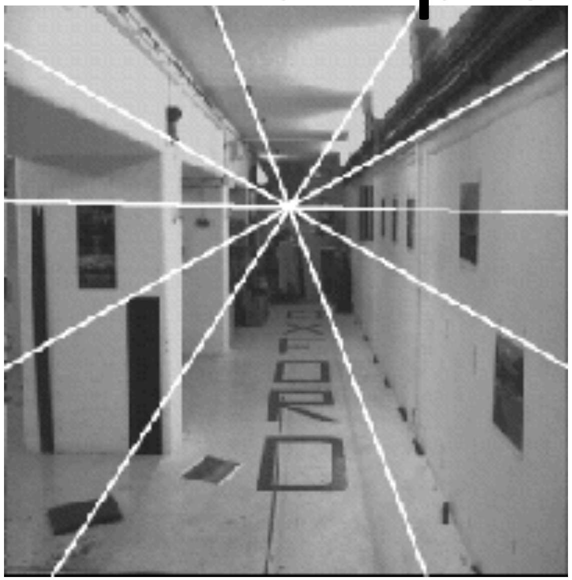
Example: Motion parallel to image plane

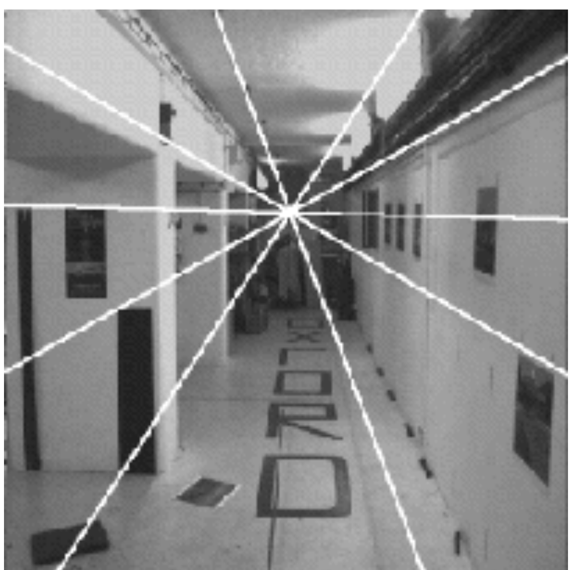


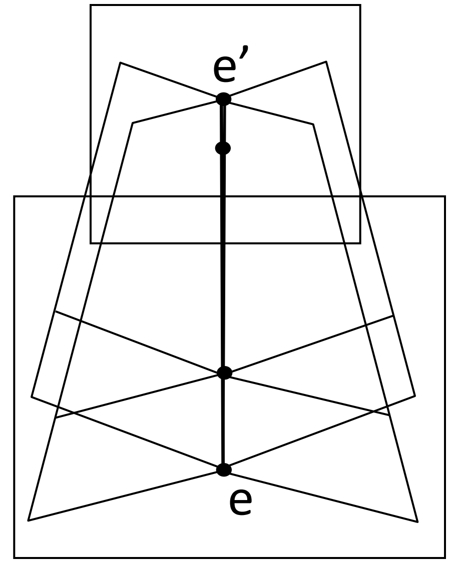
Example: Forward motion

What would the epipolar lines look like if the camera moves directly forward?

Example: Forward motion

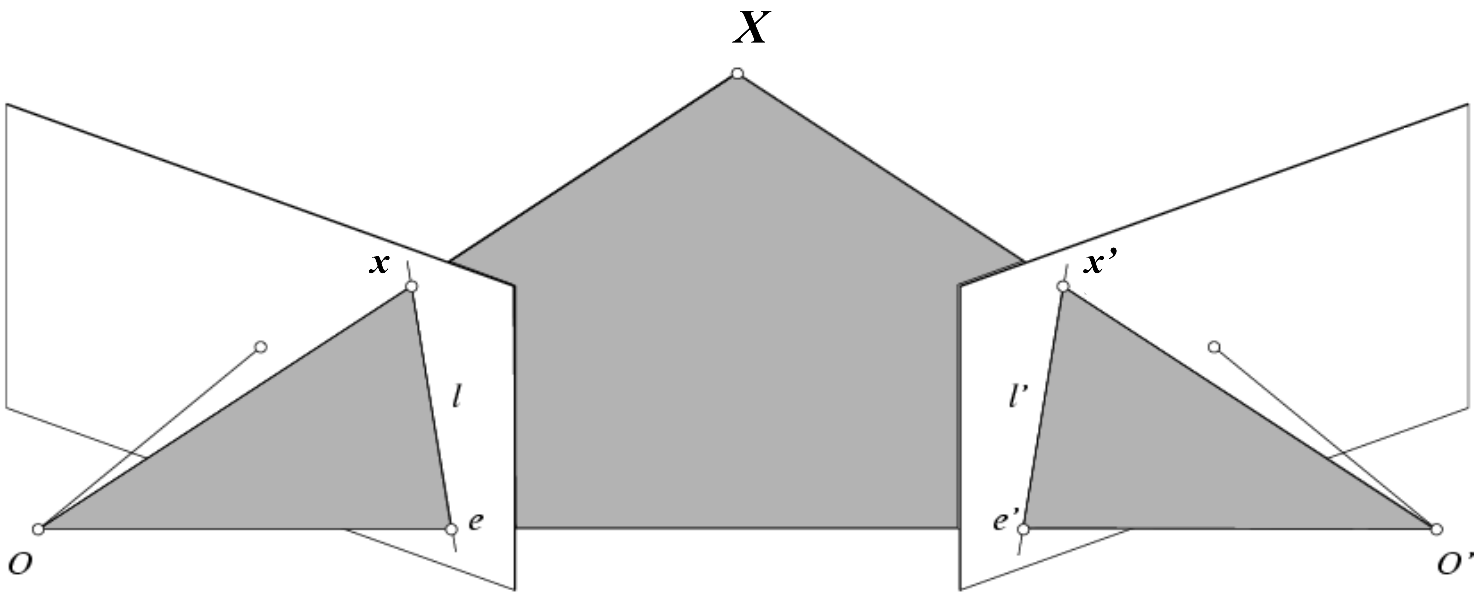






Epipole has same coordinates in both images.
Points move along lines radiating from e:
“Focus of expansion”

Epipolar constraint: Calibrated case



Given the intrinsic parameters of the cameras:

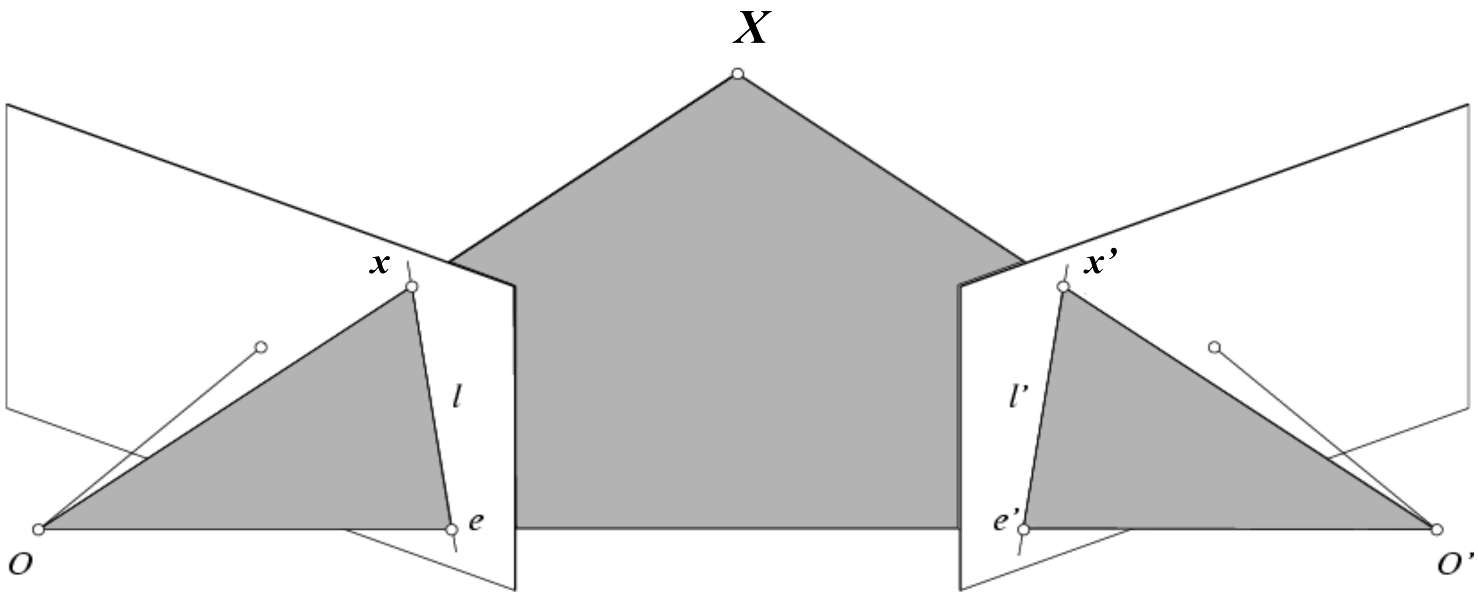
1. Convert to normalized coordinates by pre-multiplying all points with the inverse of the calibration matrix

$$\hat{x} = K^{-1} x \qquad\qquad \hat{x}' = K'^{-1} x'$$

Normalized coordinate
(3D ray towards X)

Image coordinate
(pixel location)

Epipolar constraint: Calibrated case



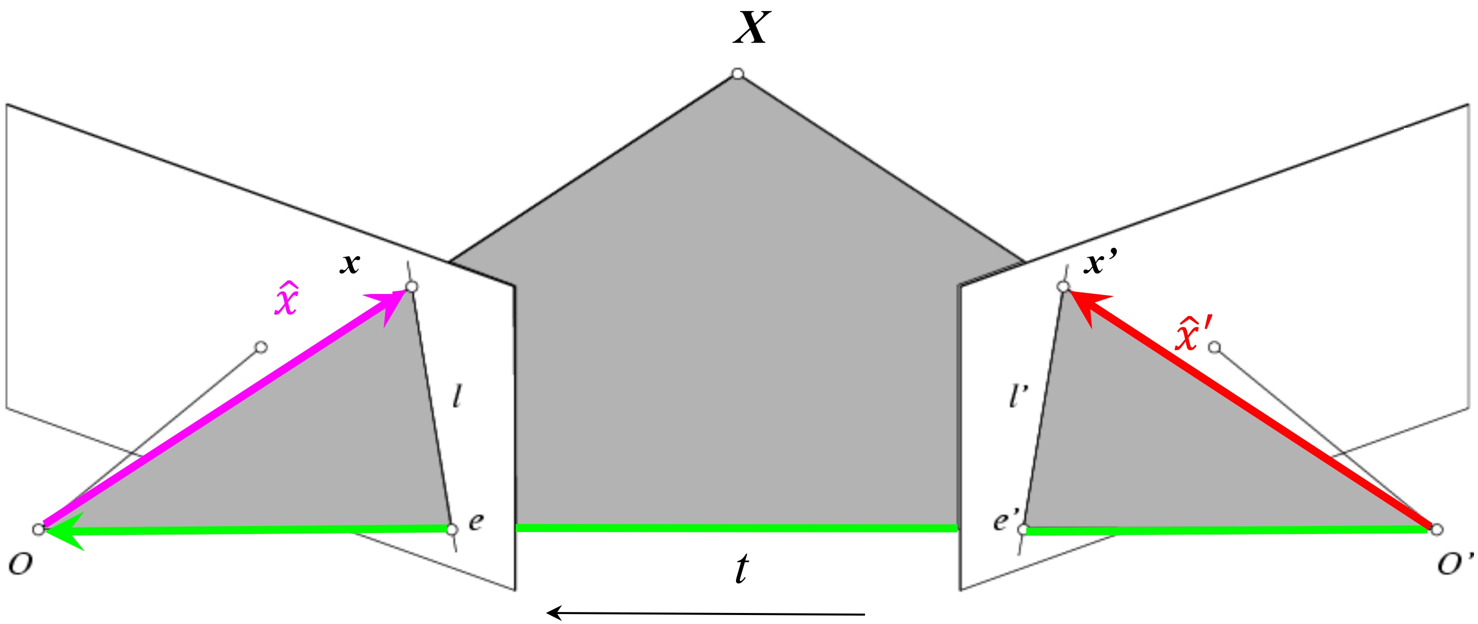
Given the intrinsic parameters of the cameras:

1. Convert to normalized coordinates by pre-multiplying all points with the inverse of the calibration matrix
2. Define some *R* and *t* that relate x to x' as below

$$\hat{x} = K^{-1}x \qquad \hat{x}' = K'^{-1}x'$$

$$\boxed{\hat{x} = R\hat{x}' + t}$$

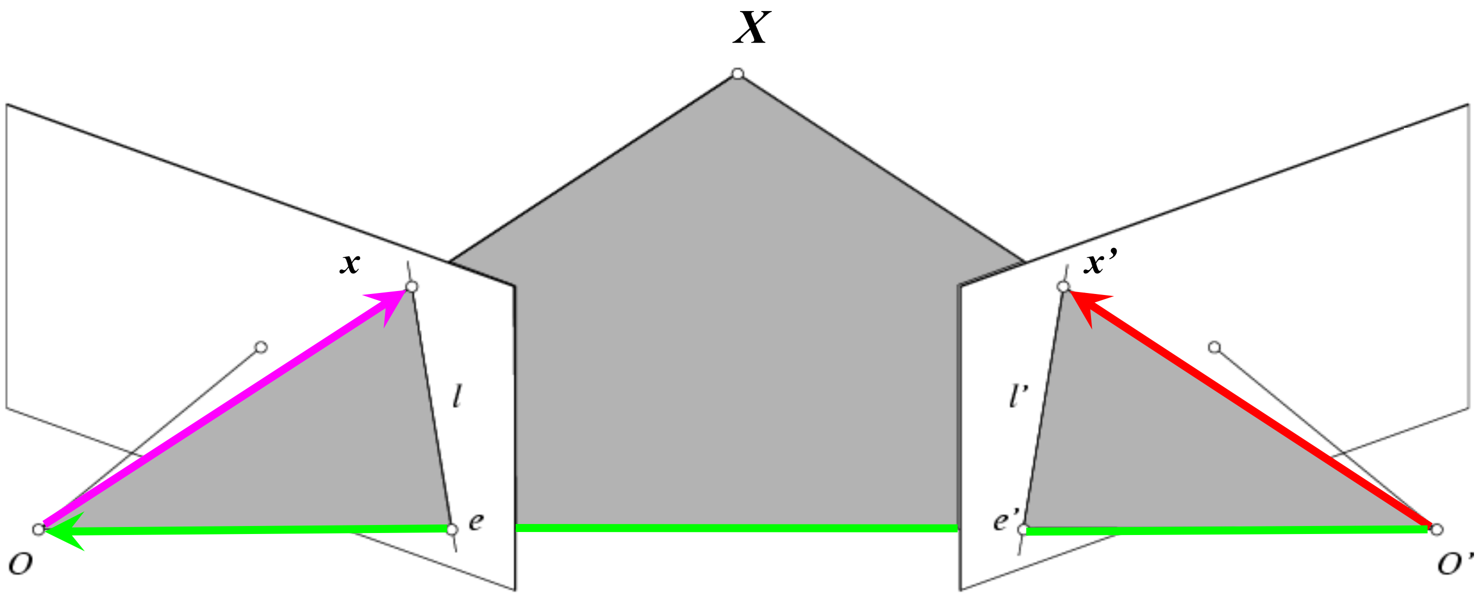
Epipolar constraint: Calibrated case



$$\hat{x} = R\hat{x}' + t \quad \Rightarrow \quad \hat{x} \cdot [t \times (R\hat{x}')] = 0$$

(because $\hat{x}$, $R\hat{x}'$, and t are co-planar)

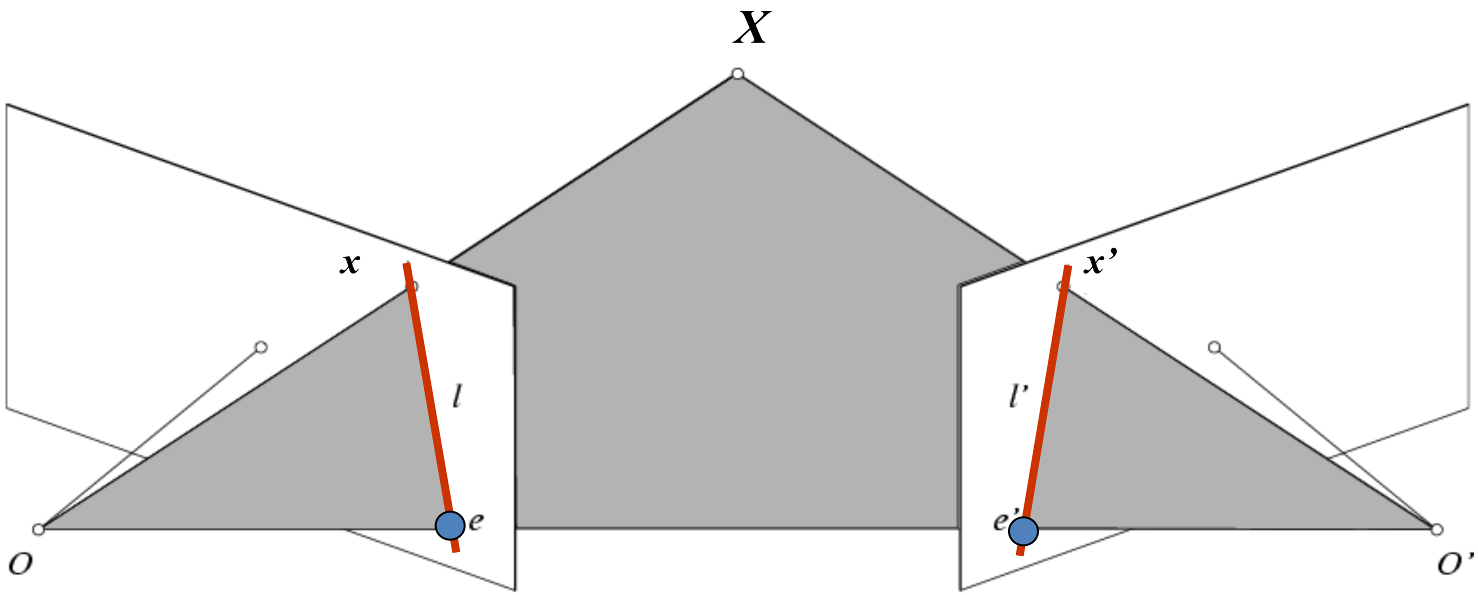
Essential matrix



$$\hat{x} \cdot [t \times (R\hat{x}')] = 0 \quad \Rightarrow \quad \hat{x}^T E \hat{x}' = 0 \quad \text{with} \quad E = [t]_\times R$$

Essential Matrix
(Longuet-Higgins, 1981)

Properties of the Essential matrix



$$\hat{x} \cdot [t \times (R\hat{x}')] = 0 \quad \Rightarrow \quad \hat{x}^T E \hat{x}' = 0 \quad \text{with} \quad E = [t]_\times R$$

Drop ^ below to simplify notation

Skew-symmetric matrix

- $E x'$ is the epipolar line associated with x' ($l = E x'$)
- $E^T x$ is the epipolar line associated with x ($l' = E^T x$)
- $E e' = 0$ and $E^T e = 0$
- E is singular (rank two)
- E has five degrees of freedom
  - (3 for R, 2 for t because it's up to a scale)

The Fundamental Matrix

Without knowing K and K’, we can define a similar relation using image coordinates

$$\hat{x}^T E \hat{x}' = 0$$

$$\hat{x} = K^{-1} x$$

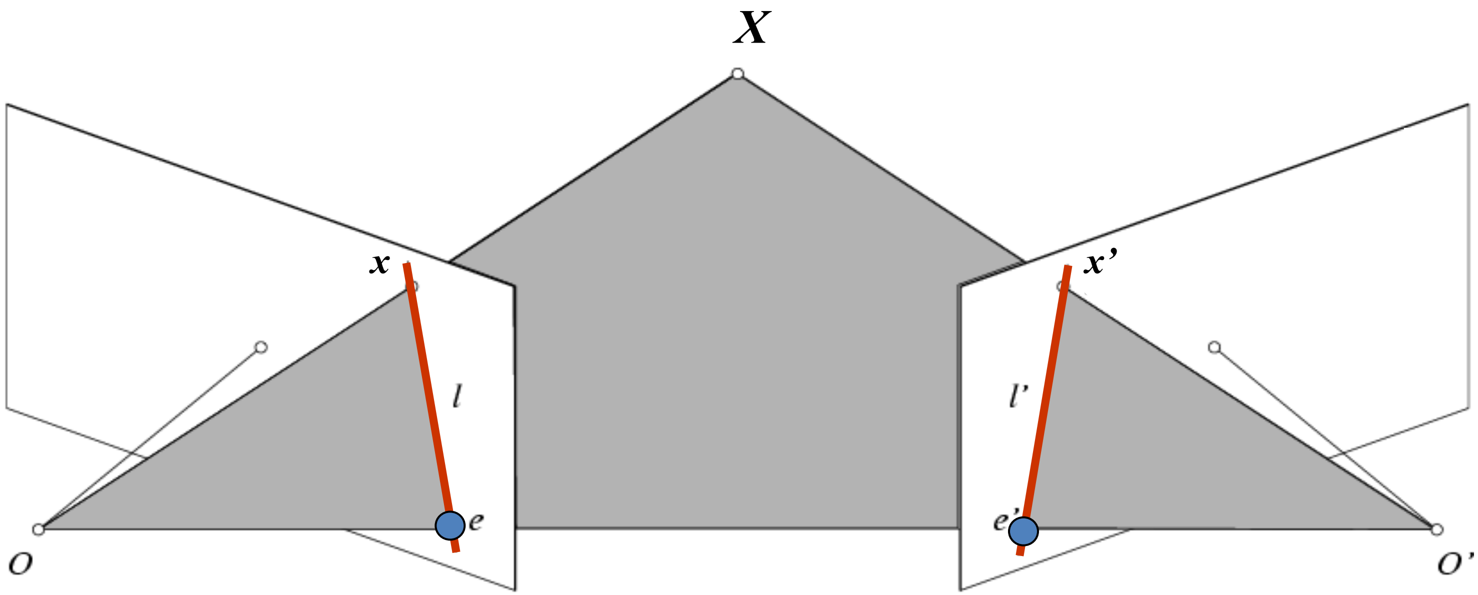
$$\hat{x}' = K'^{-1} x'$$

$$\Rightarrow \quad x^T F x' = 0 \quad \text{with} \quad F = K^{-T} E K'^{-1}$$

Fundamental Matrix
(Faugeras and Luong, 1992)

- *F* is singular (rank two): det(F)=0
- *F* has seven degrees of freedom: 9 entries but defined up to scale, det(F)=0

Properties of the Fundamental matrix



$$x^T F x' = 0 \quad \text{with} \quad F = K^{-T} E K'^{-1}$$

- $F x'$ is the epipolar line associated with x' ($l = F x'$)
- $F^T x$ is the epipolar line associated with x ($l' = F^T x$)
- $F e' = 0$ and $F^T e = 0$
- F is singular (rank two): det(F)=0
- F has seven degrees of freedom: 9 entries but defined up to scale, det(F)=0

How to solve F?

Write down the system of equations

$$\mathbf{x}^T F \mathbf{x}' = 0$$

$$uu'f_{11} + uv'f_{12} + uf_{13} + vu'f_{21} + vv'f_{22} + vf_{23} + u'f_{31} + v'f_{32} + f_{33} = 0$$

$$\mathrm{A}\boldsymbol{f} = \begin{bmatrix} u_1u_1' & u_1v_1' & u_1 & v_1u_1' & v_1v_1' & v_1 & u_1' & v_1' & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_nu_v' & u_nv_n' & u_n & v_nu_n' & v_nv_n' & v_n & u_n' & v_n' & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ \vdots \\ f_{33} \end{bmatrix} = \mathbf{0}$$

How many equations are needed?

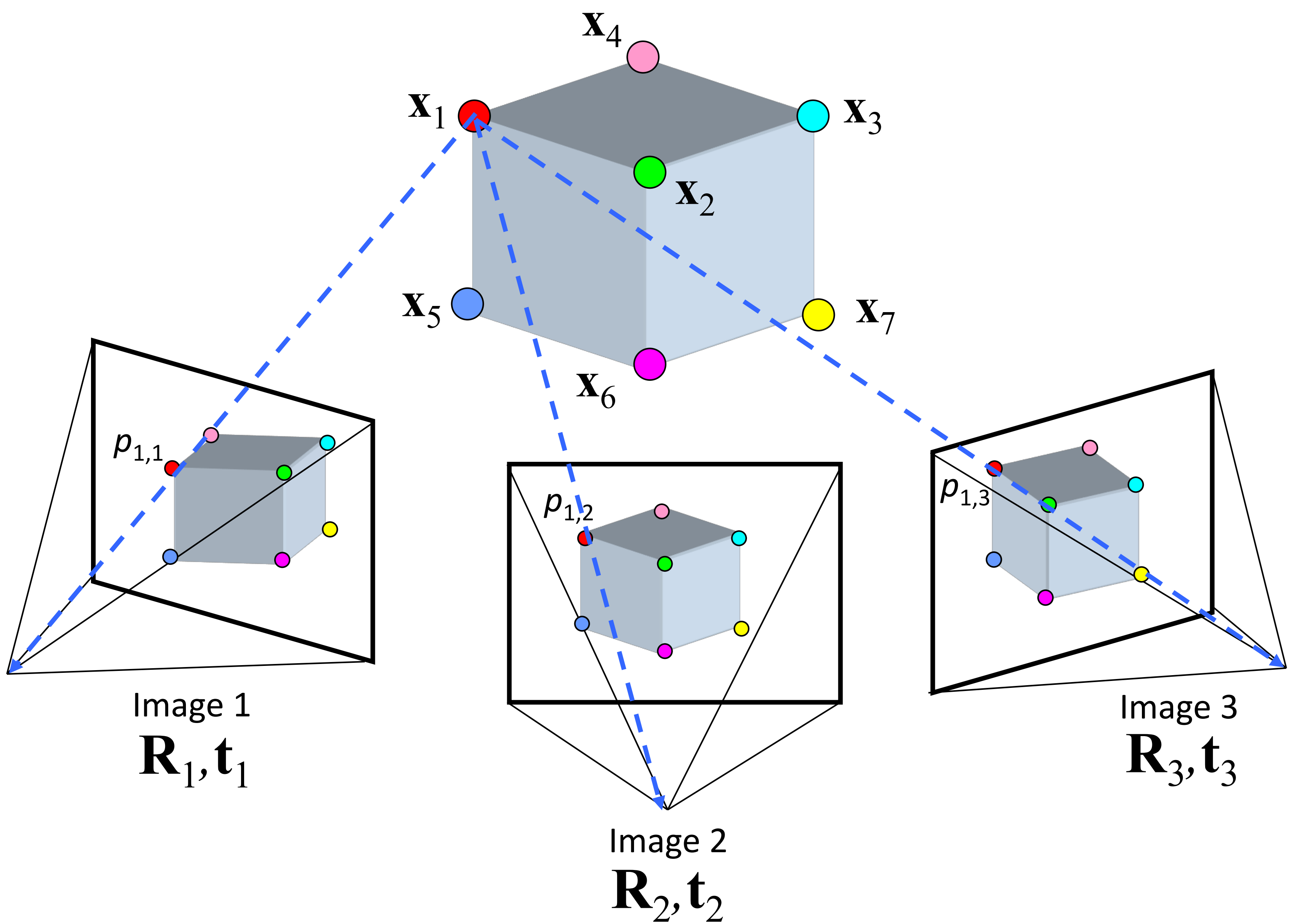
8-point algorithm

1. Solve a system of homogeneous linear equations
   a. Write down the system of equations
   b. Solve **f** from A**f**=**0** using SVD
2. Resolve det(F) = 0 constraint by SVD

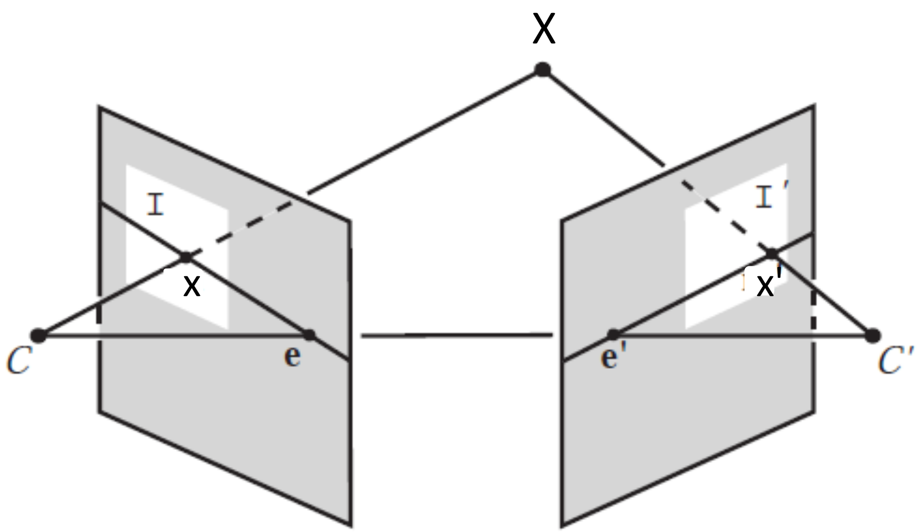
Notes:

- Use RANSAC to deal with outliers (sample 8 points)
  - How to test for outliers? $|x'F\, x\,| < threshold?$

Triangulation



Triangulation: Linear Solution



$$\mathbf{x} = \mathbf{P}\mathbf{X} \qquad \mathbf{x}' = \mathbf{P}'\mathbf{X}$$

$$\mathbf{A}\mathbf{X} = \mathbf{0} \qquad \mathbf{A} = \begin{bmatrix} u\mathbf{p}_3^T - \mathbf{p}_1^T \\ v\mathbf{p}_3^T - \mathbf{p}_2^T \\ u'\mathbf{p}_3'^T - \mathbf{p}_1'^T \\ v'\mathbf{p}_3'^T - \mathbf{p}_2'^T \end{bmatrix}$$

- Generally, rays C→x and C'→x' will not exactly intersect
- Solve via SVD:
  A least squares solution to a system of equations

Further reading: HZ p. 312-313

Triangulation: Non-linear Solution

- Minimize projected error while satisfying

$$\hat{\boldsymbol{x}}'^{T}\boldsymbol{F}\hat{\boldsymbol{x}}=0$$

$$cost(\boldsymbol{X}) = dist(\boldsymbol{x}, \hat{\boldsymbol{x}})^2 + dist(\boldsymbol{x}', \hat{\boldsymbol{x}}')^2$$

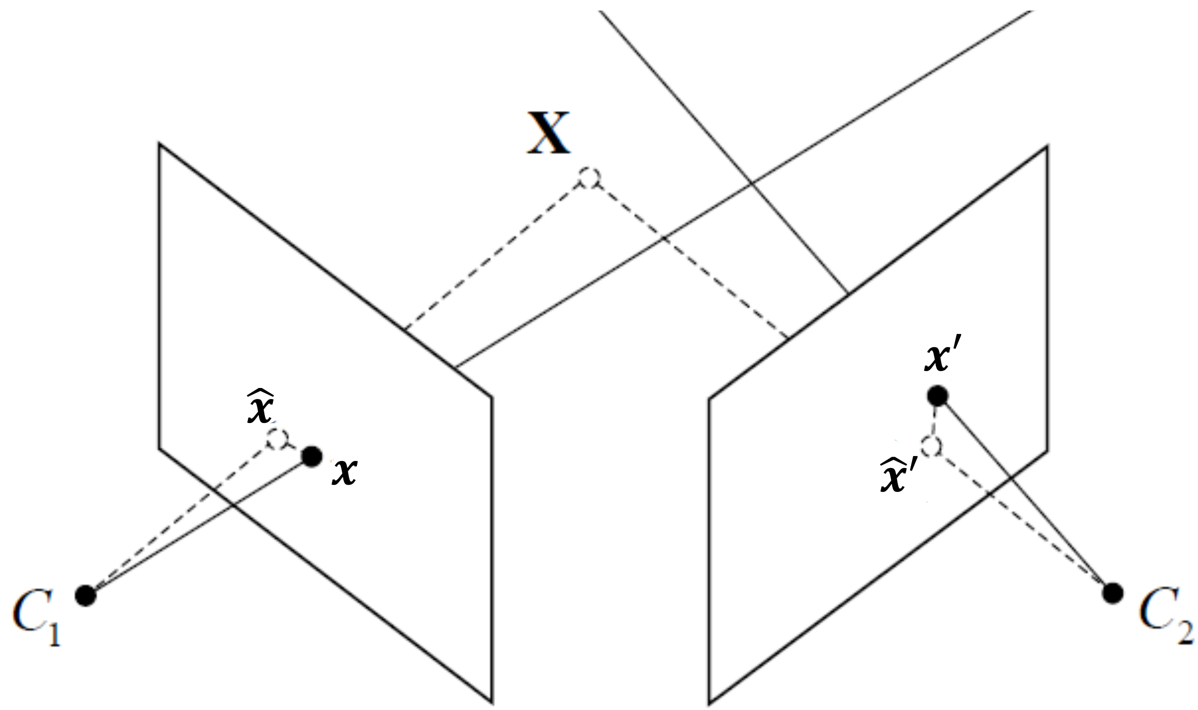
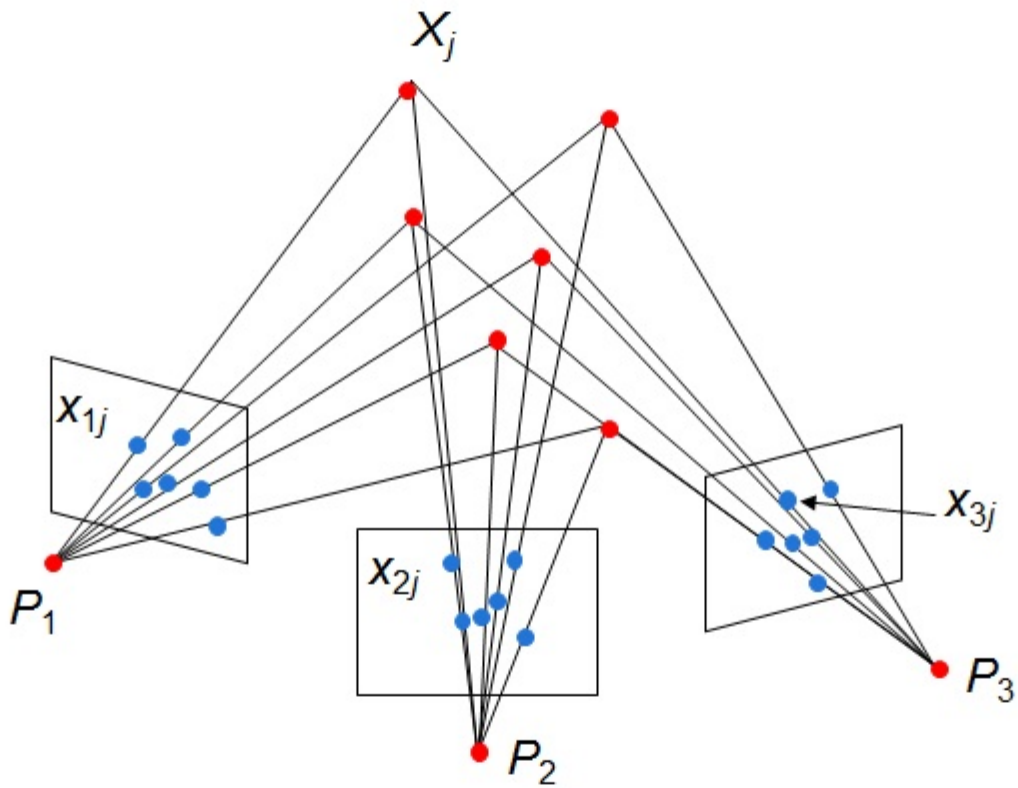


Figure source: Robertson and Cipolla (Chpt 13 of Practical Image Processing and Computer Vision)

Questions?

Structure from Motion



Structure

3D Point Cloud of the Scene

Motion

Camera Location and Orientation

Structure from Motion (SfM)

Get the Point Cloud from Moving Cameras

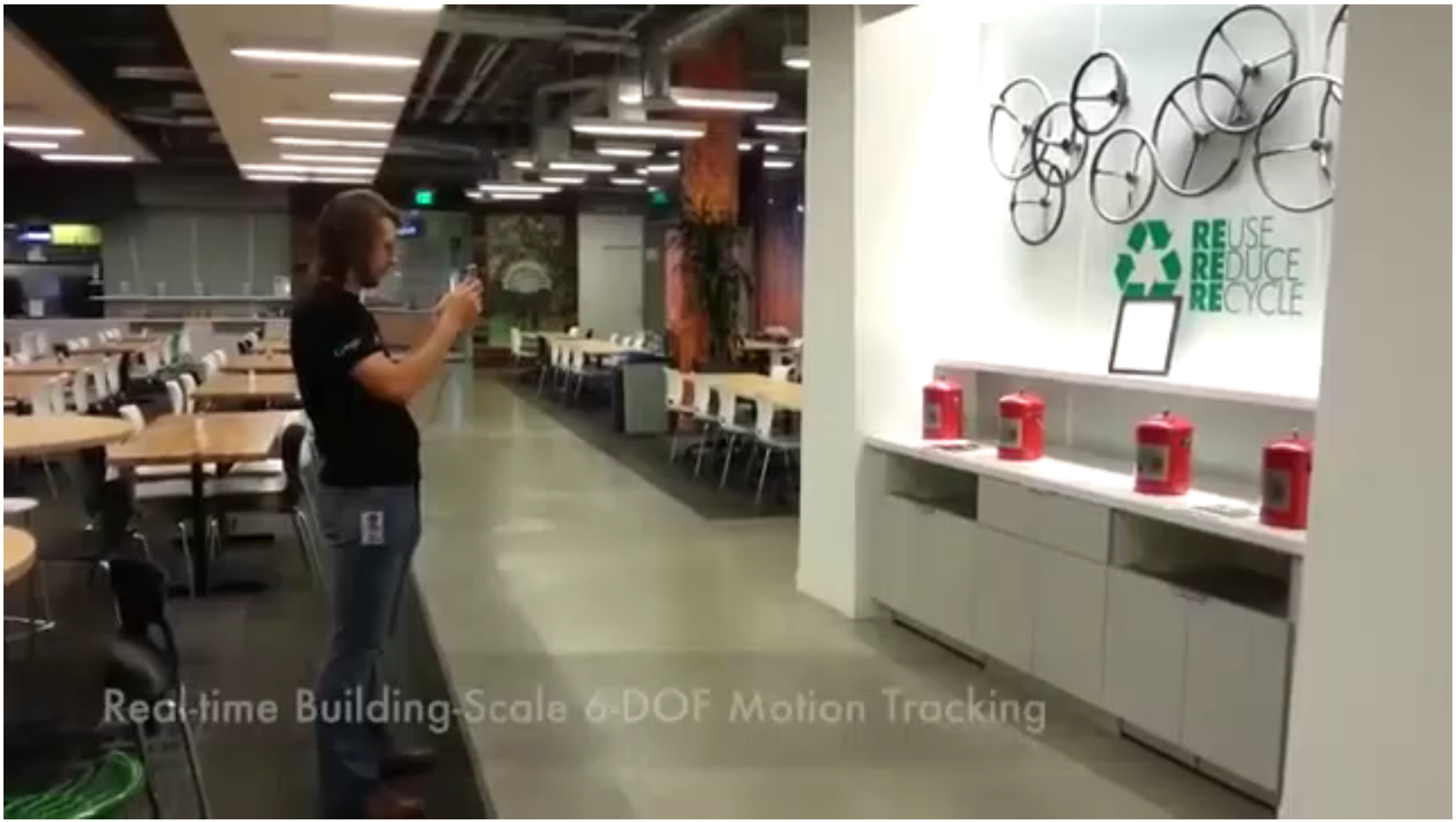
SfM Applications – 3D Modeling

http://www.3dcadbrowser.com/download.aspx?3dmodel=40454

SfM Applications – Surveying cultural heritage structure analysis

Guidi et al. High-accuracy 3D modeling of cultural heritage, 2004

SfM Applications – localization and mapping (SLAM)



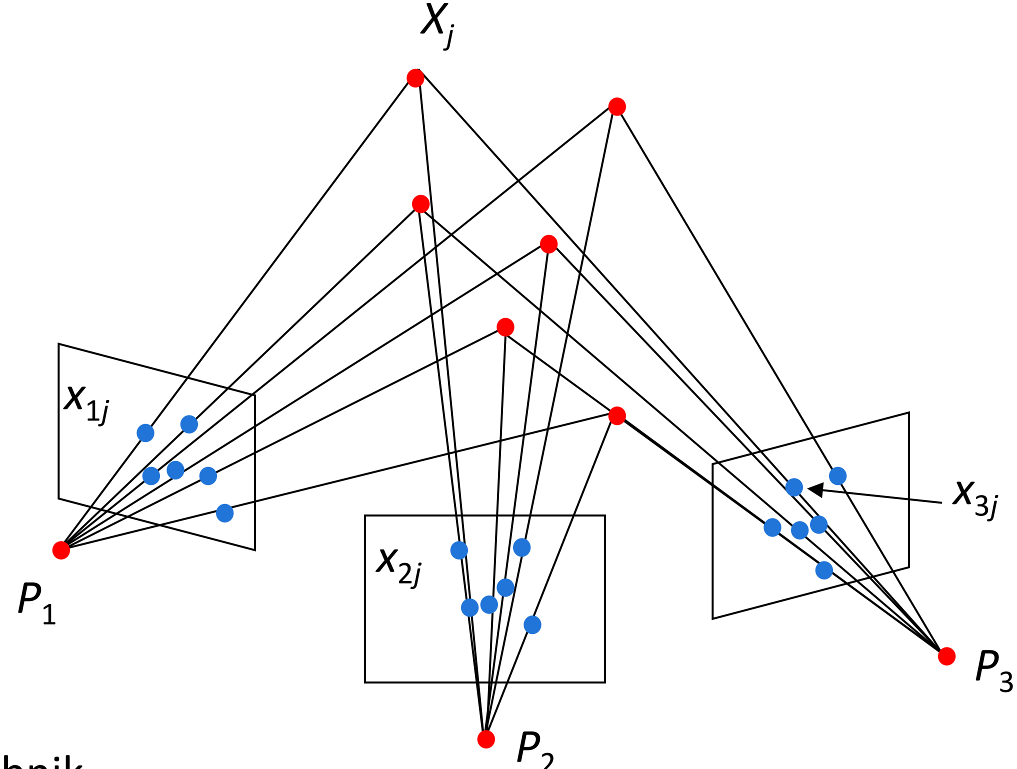
https://www.youtube.com/watch?v=1HhOmF22oYA

Structure from motion

- Given: *m* images of *n* fixed 3D points

$$\mathbf{x}_{ij} = \mathbf{P}_i \mathbf{X}_j, \quad i = 1, \dots, m, \quad j = 1, \dots, n$$

- Problem: estimate *m* projection matrices $\mathbf{P}_i$ and *n* 3D points $\mathbf{X}_j$ from the *mn* corresponding 2D points $\mathbf{x}_{ij}$



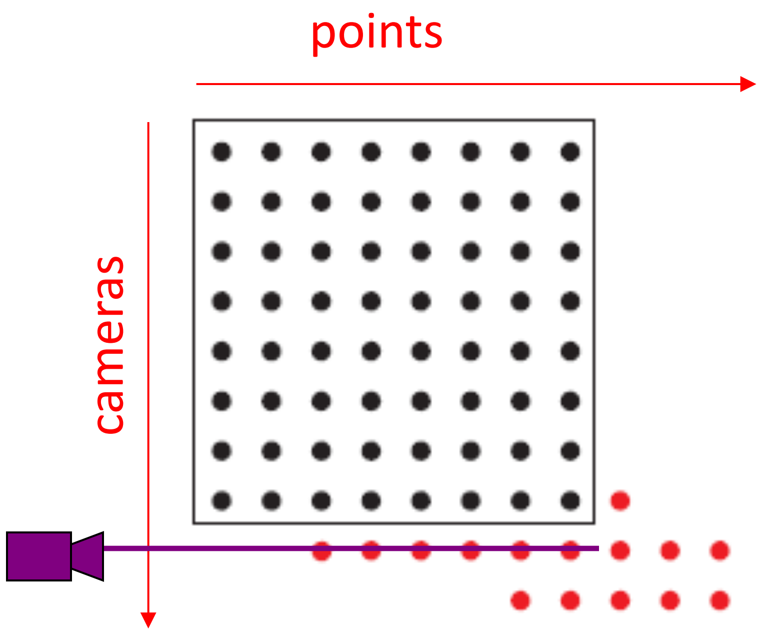
Slides from Lana Lazebnik

Sequential structure from motion

- Initialize motion (calibration) from two images using fundamental matrix

- Initialize structure by triangulation

- For each additional view:
  - Determine projection matrix of new camera using all the known 3D points that are visible in its image – *calibration/resectioning*



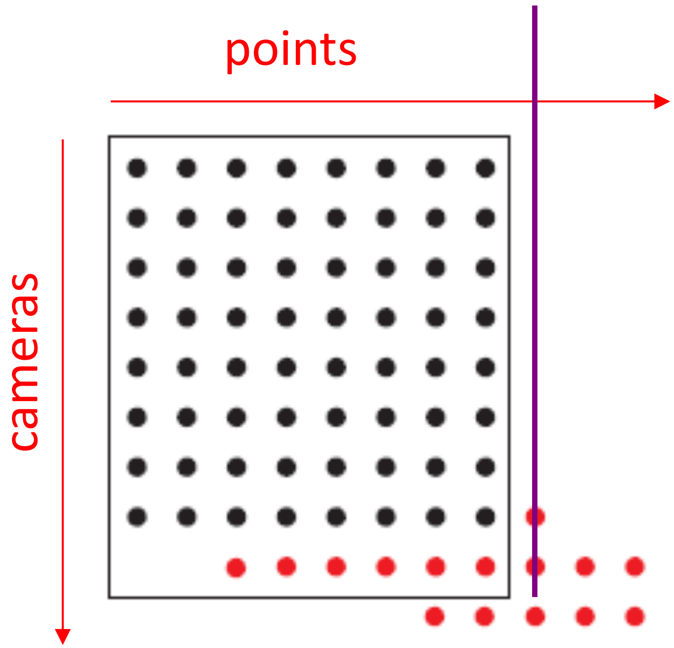
Sequential structure from motion

- Initialize motion from two images using fundamental matrix

- Initialize structure by triangulation

- For each additional view:
  - Determine projection matrix of new camera using all the known 3D points that are visible in its image – *calibration*

  - Refine and extend structure: compute new 3D points, re-optimize existing points that are also seen by this camera – *triangulation*



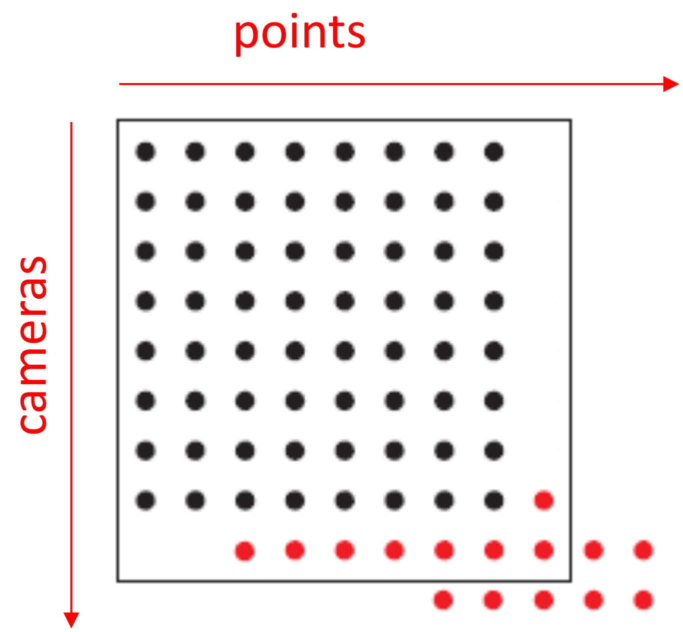
Sequential structure from motion

- Initialize motion from two images using fundamental matrix

- Initialize structure by triangulation

- For each additional view:
  - Determine projection matrix of new camera using all the known 3D points that are visible in its image – *calibration*
  - Refine and extend structure: compute new 3D points, re-optimize existing points that are also seen by this camera – *triangulation*

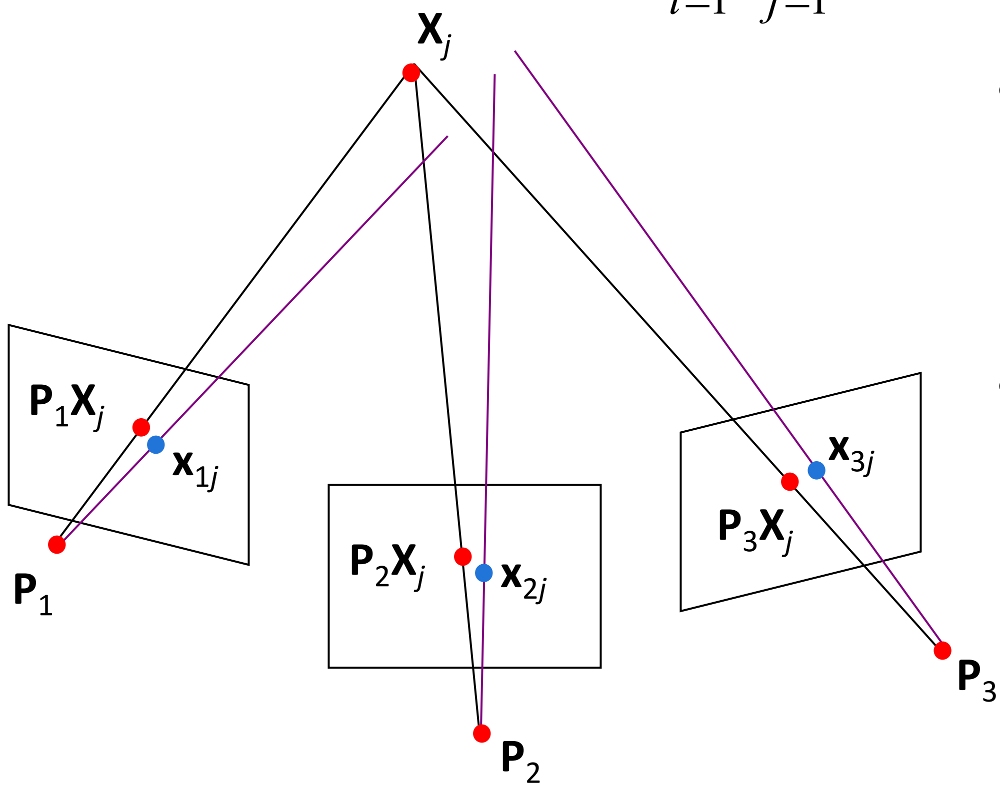
- Refine structure and motion: bundle adjustment



Bundle adjustment

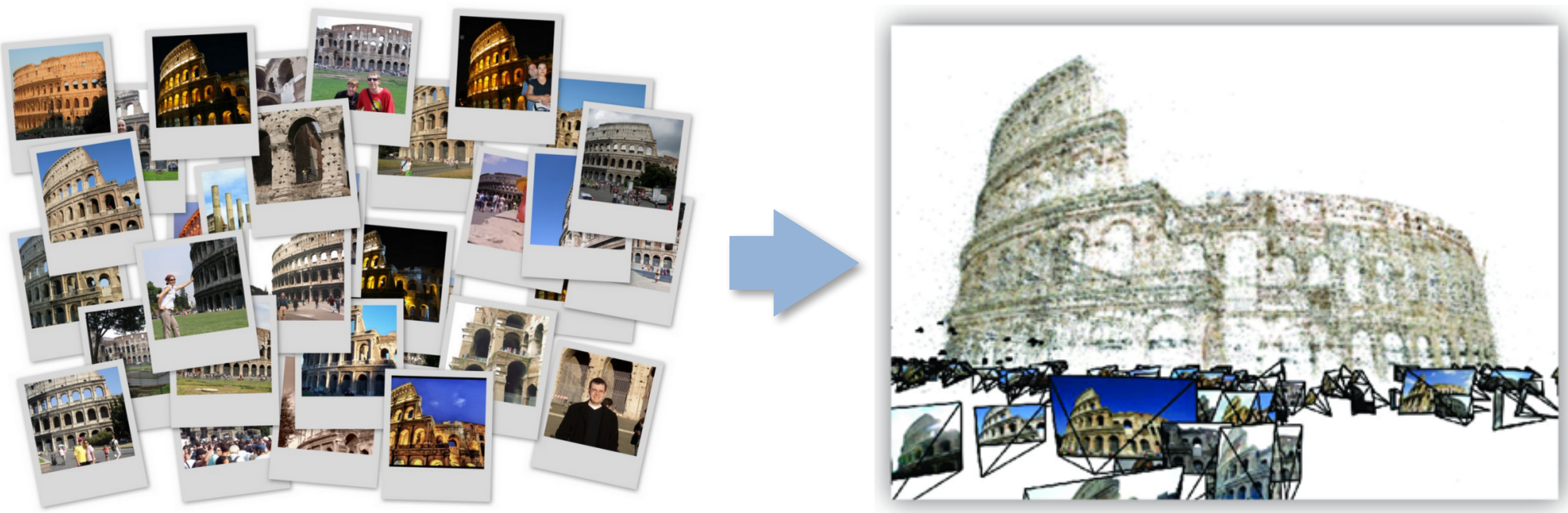
- Non-linear method for refining structure and motion
- Minimizing reprojection error

$$E(\mathbf{P},\mathbf{X}) = \sum_{i=1}^{m}\sum_{j=1}^{n} D\left(\mathbf{x}_{ij},\mathbf{P}_i\mathbf{X}_j\right)^2$$



- Theory:
  The Levenberg–Marquardt algorithm
- Practice:
  The Ceres-Solver from Google

3D from multiple images



Building Rome in a Day: Agarwal et al. 2009

3D from multiple images

Building Rome in a Day: Agarwal et al. 2009

Steps

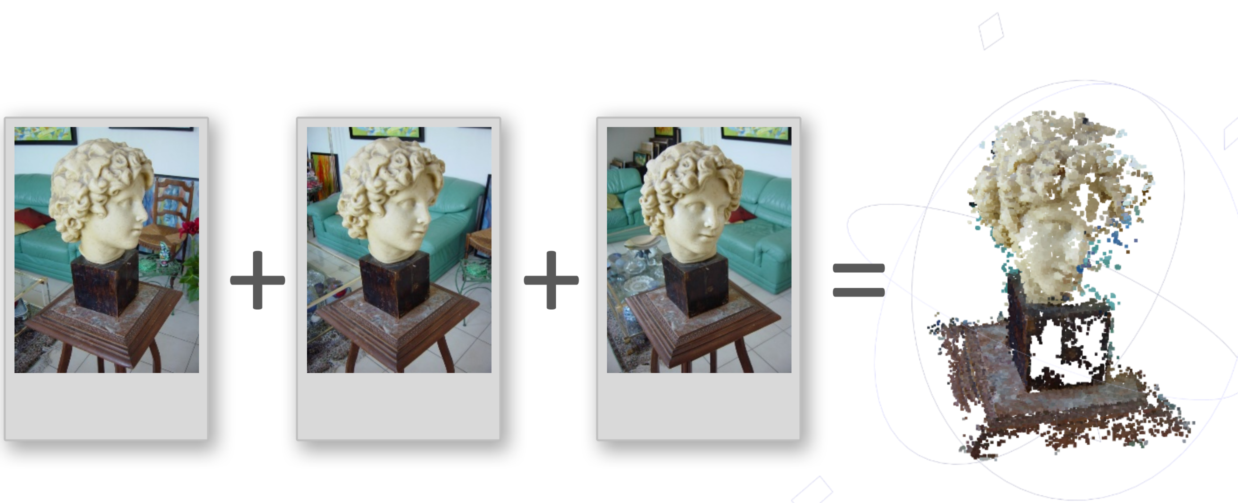
Images → Points: Structure from Motion

Points → More points: Multiple View Stereo

Points → Meshes: Model Fitting

\+ Meshes → Models: Texture Mapping

= Images → Models: Image-based Modeling



Slide credit: J. Xiao

Steps

Images → Points: Structure from Motion

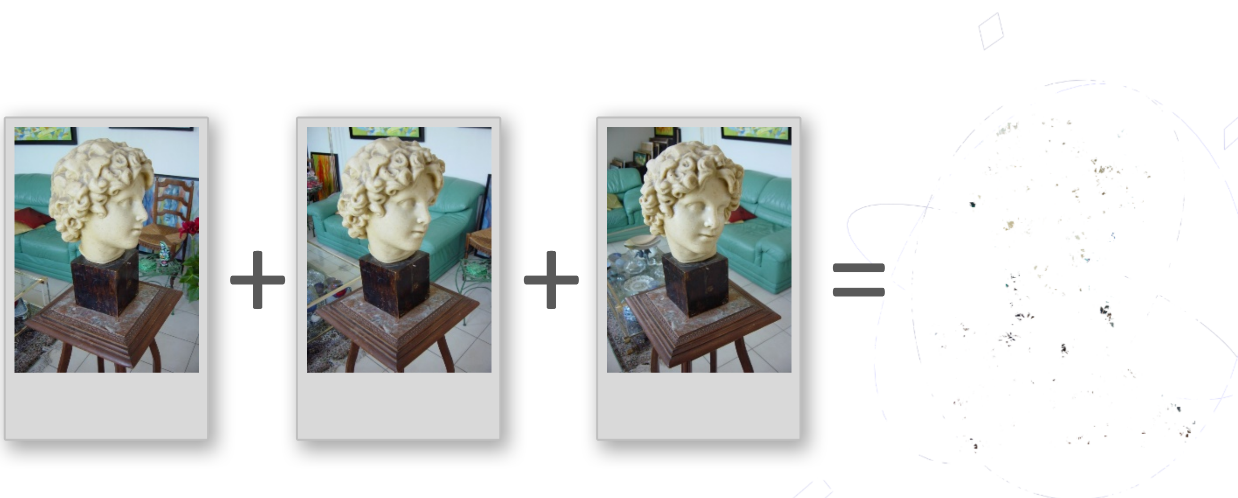
Points → More points: Multiple View Stereo

Points → Meshes: Model Fitting

Meshes → Models: Texture Mapping

=

Images → Models: Image-based Modeling



Slide credit: J. Xiao

Steps

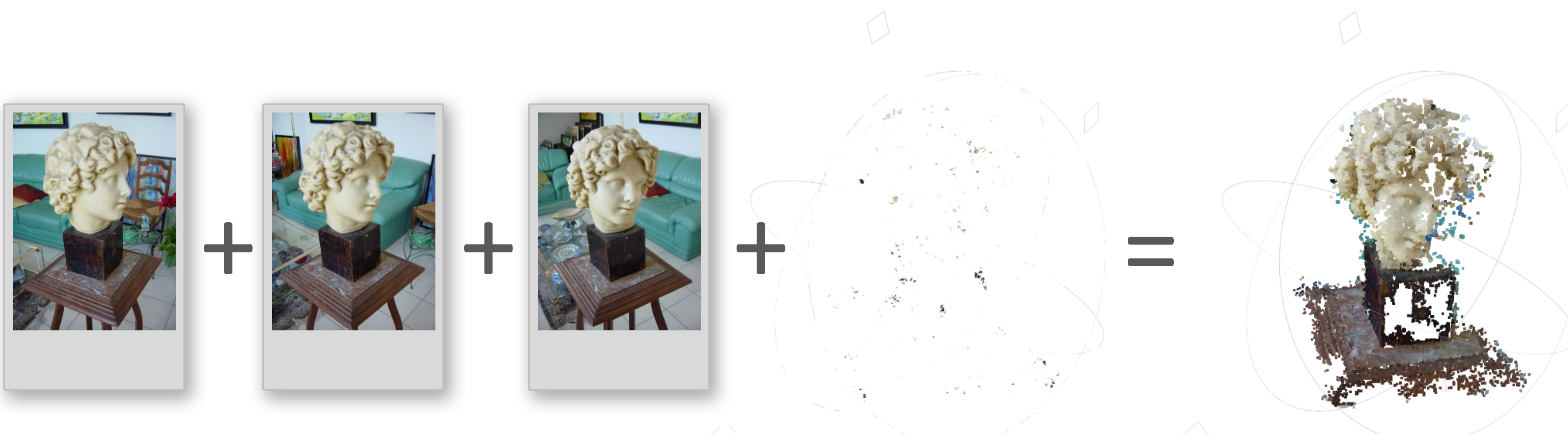
Images → Points: Structure from Motion

Points → More points: Multiple View Stereo

Points → Meshes: Model Fitting

\+ Meshes → Models: Texture Mapping

= Images → Models: Image-based Modeling



Slide credit: J. Xiao

Steps

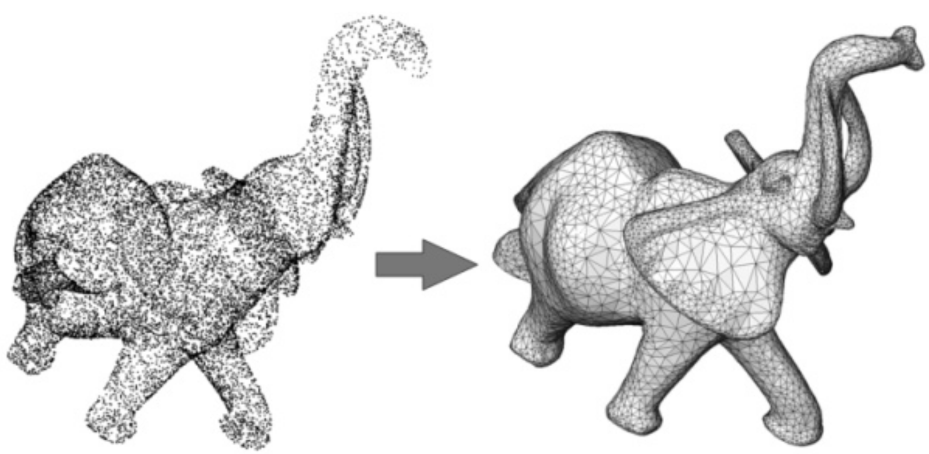
Images → Points: Structure from Motion

Points → More points: Multiple View Stereo

Points → Meshes: Model Fitting

\+ Meshes → Models: Texture Mapping

= Images → Models: Image-based Modeling



Slide credit: J. Xiao

Steps

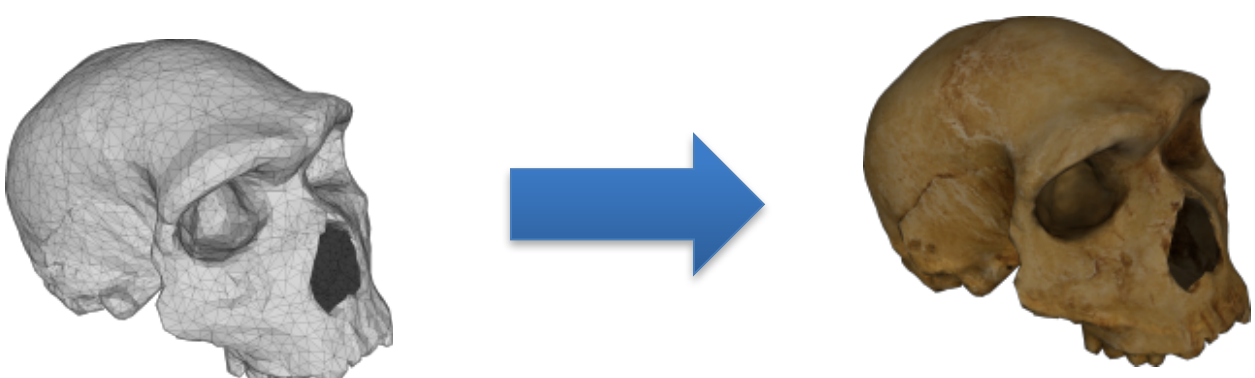
Images → Points: Structure from Motion

Points → More points: Multiple View Stereo

Points → Meshes: Model Fitting

\+ Meshes → Models: Texture Mapping

\= Images → Models: Image-based Modeling



Slide credit: J. Xiao

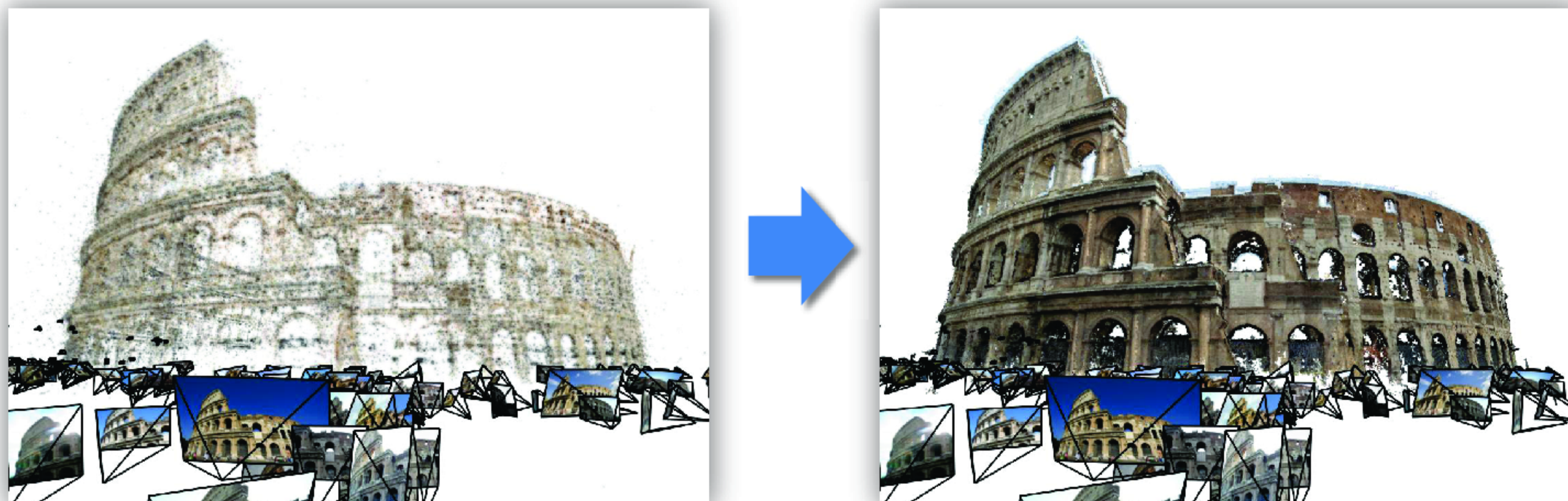
Steps

	Images → Points:	Structure from Motion
	Points → More points:	Multiple View Stereo
+	Points → Meshes:	Model Fitting
	Meshes → Models:	Texture Mapping
=	Images → Models:	$\boxed{\text{Image-based Modeling}}$

Example: https://photosynth.net/

Slide credit: J. Xiao

Multi-view stereo



Moving on to stereo…

Compute a depth image

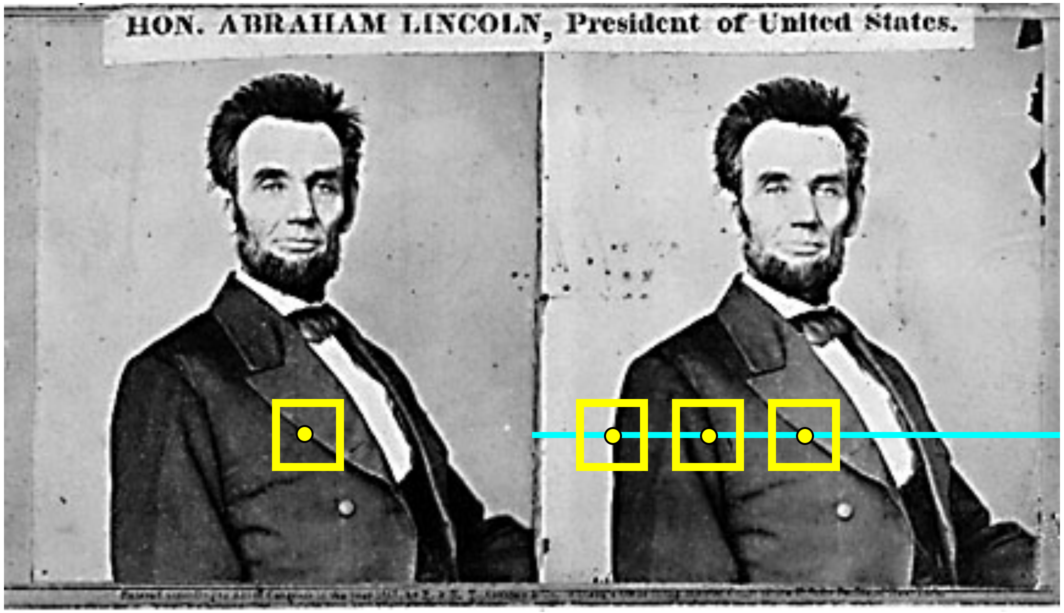
image 1

image 2

Dense depth map

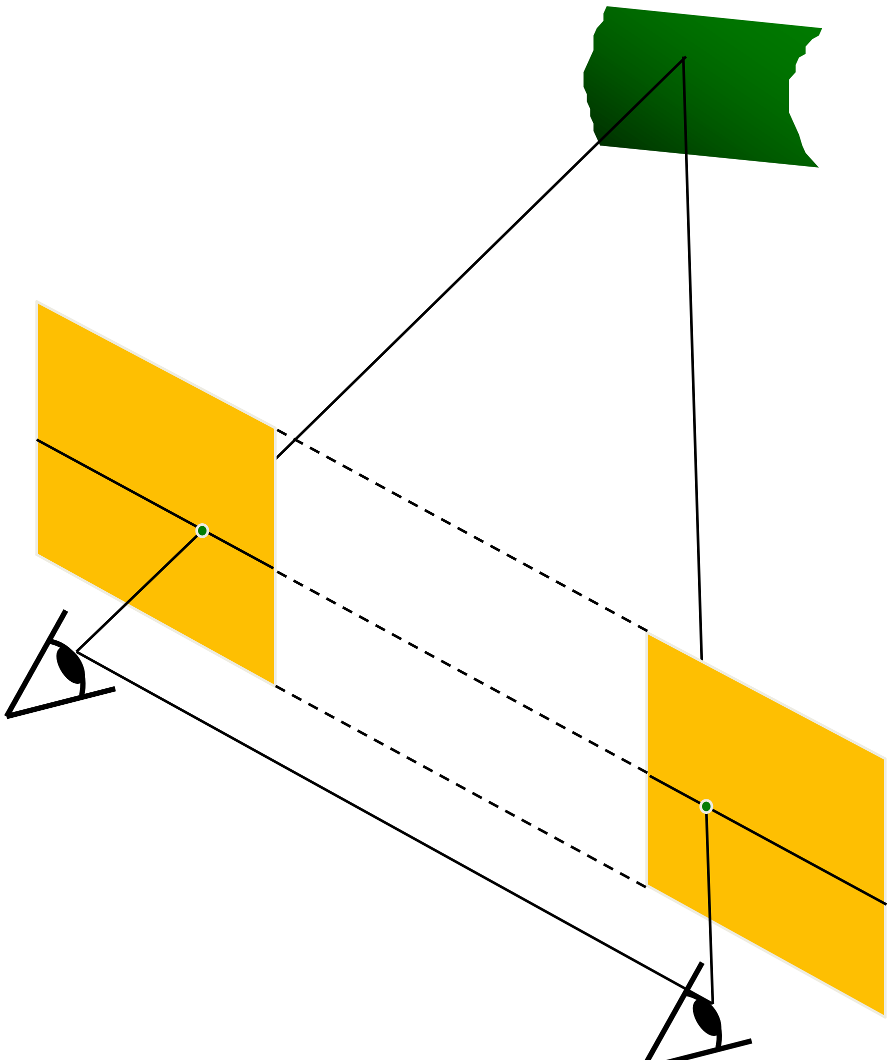
Many of these slides adapted from
Steve Seitz and Lana Lazebnik

Basic stereo matching algorithm



- For each pixel in the first image
  - Find corresponding epipolar line in the right image
  - Search along epipolar line and pick the best match
  - Triangulate the matches to get depth information
- Simplest case: epipolar lines are scanlines
  - When does this happen?

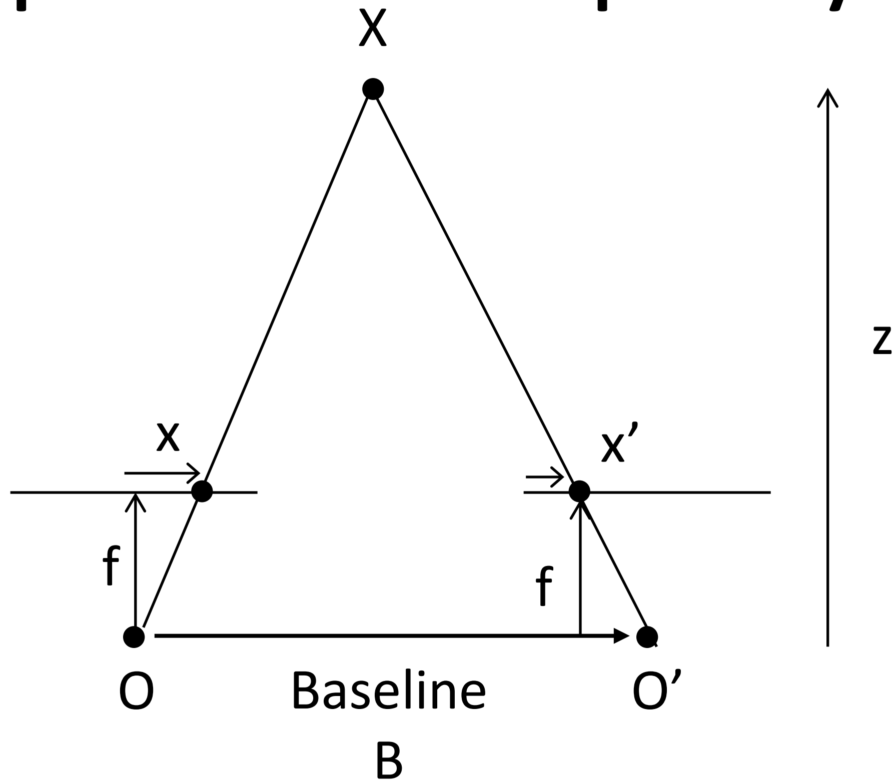
Simplest Case: Parallel images



- Image planes of cameras are parallel to each other and to the baseline
- Camera centers are at same height
- Focal lengths are the same
- Then, epipolar lines fall along the horizontal scan lines of the images

Depth from disparity

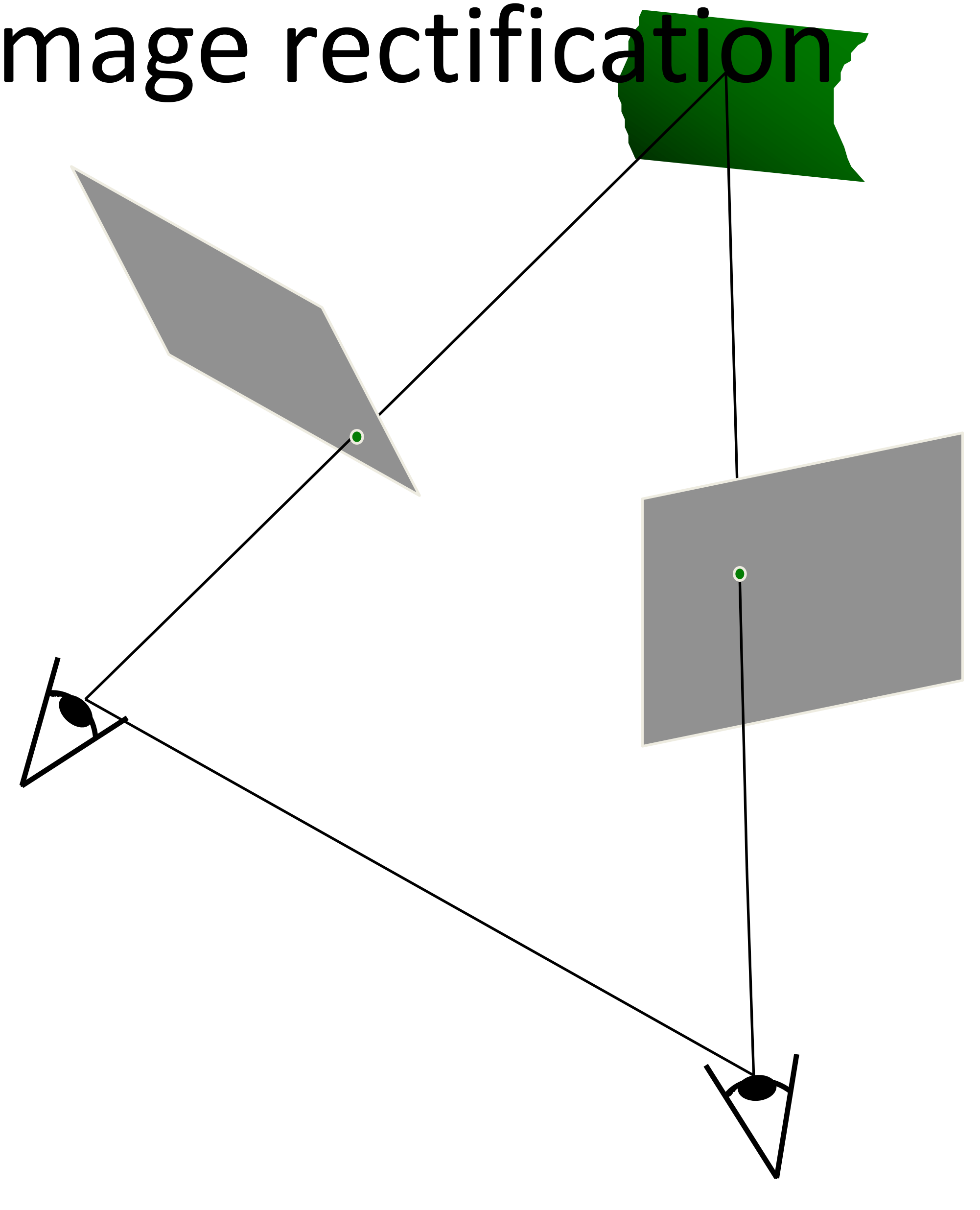
$$\frac{x - x'}{O - O'} = \frac{f}{z}$$



$$disparity = x - x' = \frac{B \cdot f}{z}$$

Disparity is inversely proportional to depth.

Stereo image rectification

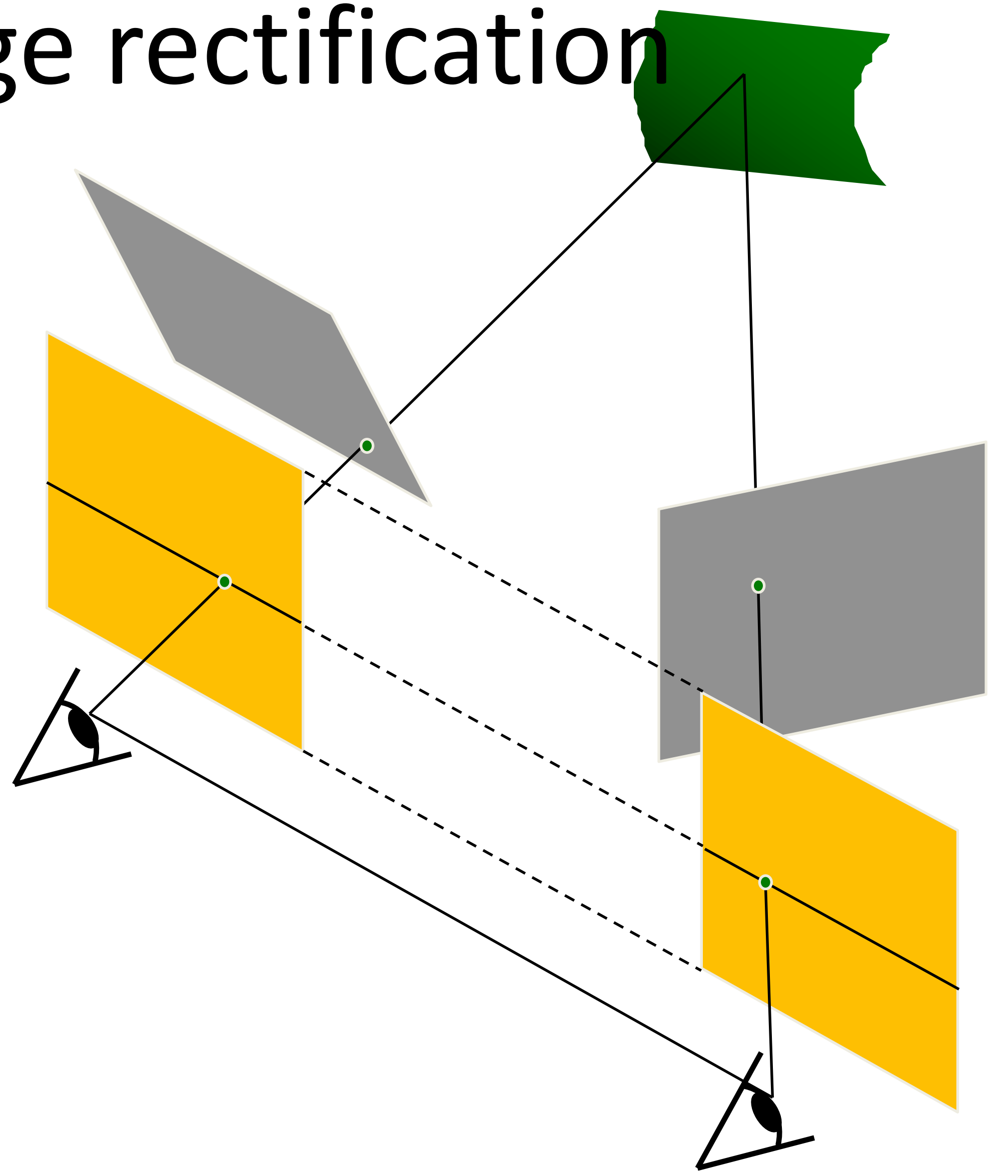


Stereo image rectification

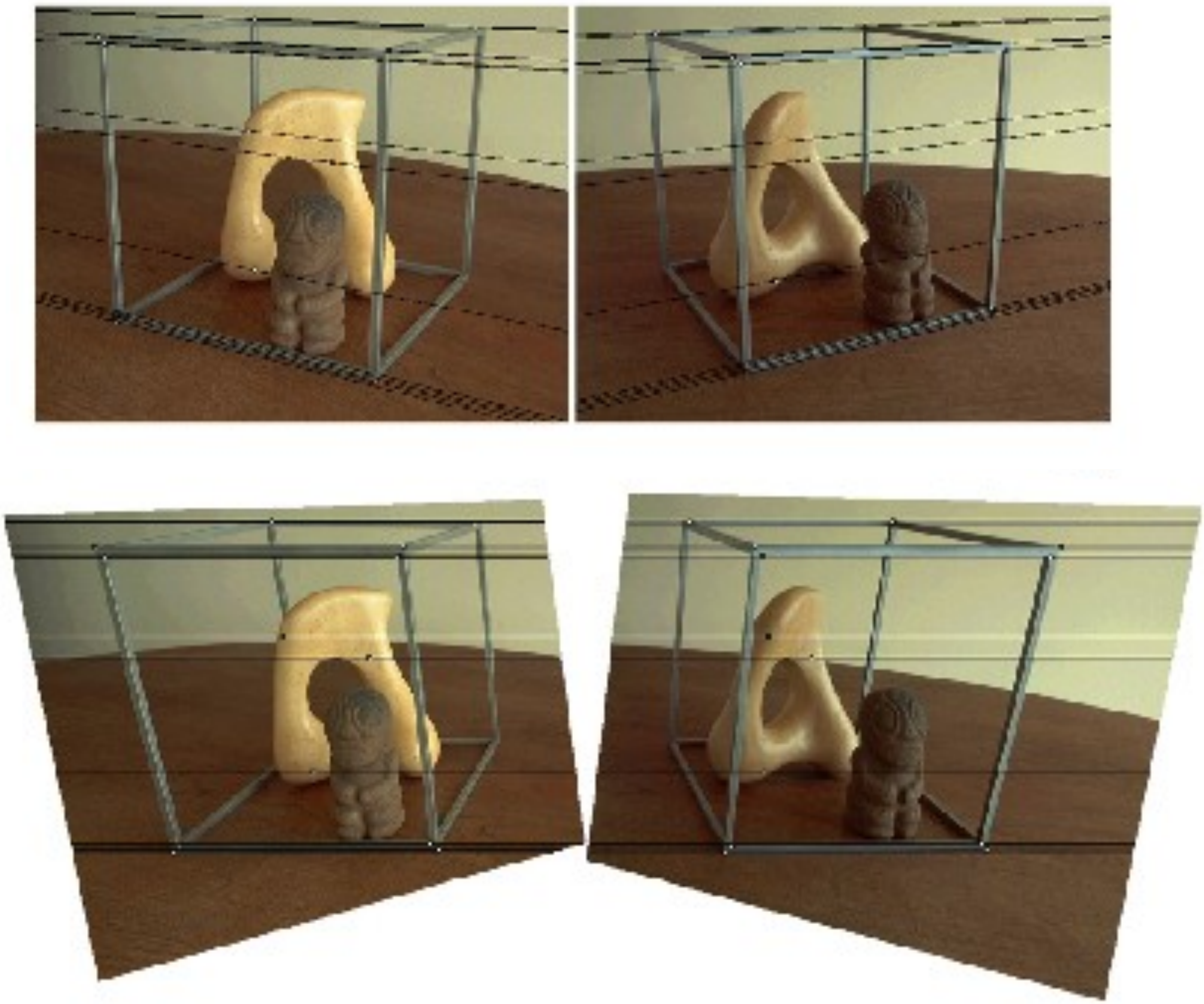
- Reproject image planes onto a common plane parallel to the line between camera centers

- Two homographies (3x3 transform), one for each input image reprojection

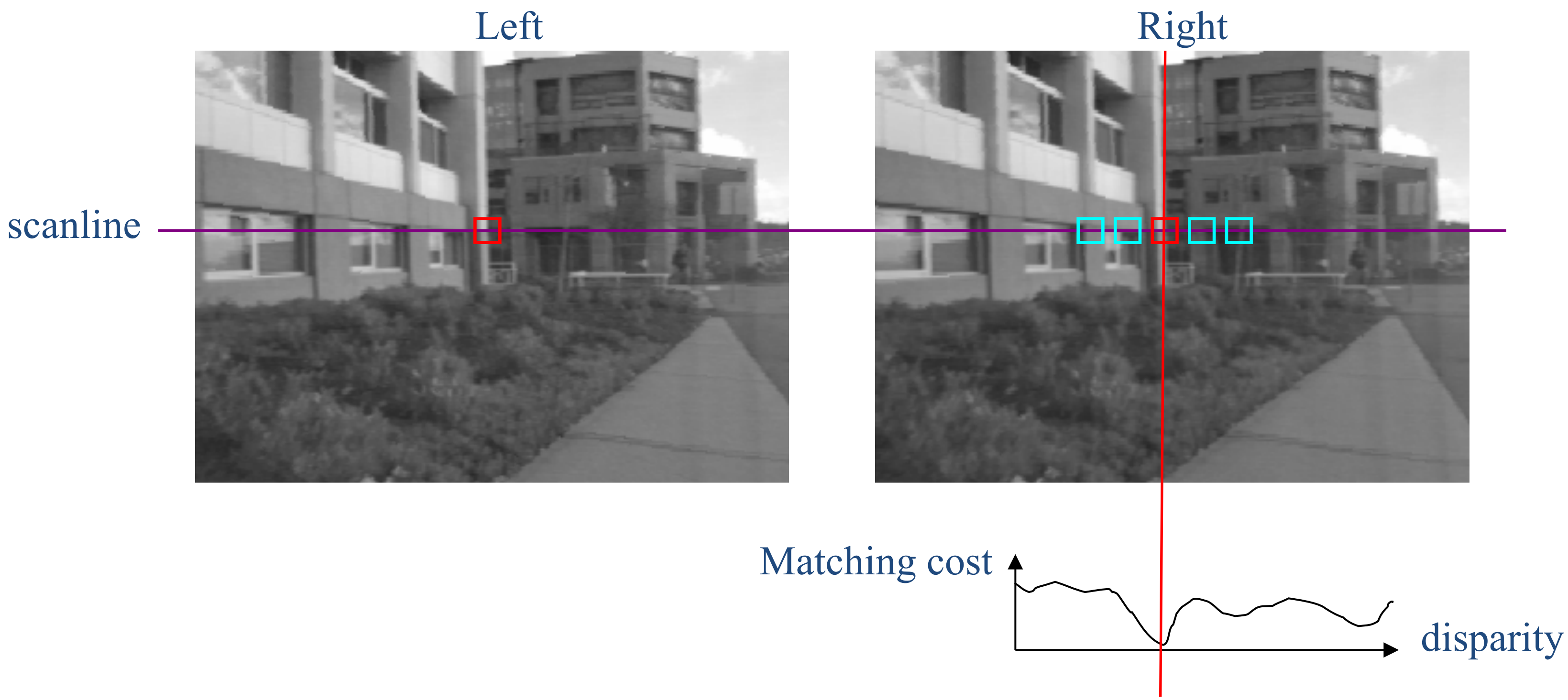
➢ C. Loop and Z. Zhang. [Computing Rectifying Homographies for Stereo Vision](). IEEE Conf. Computer Vision and Pattern Recognition, 1999.



Rectification example

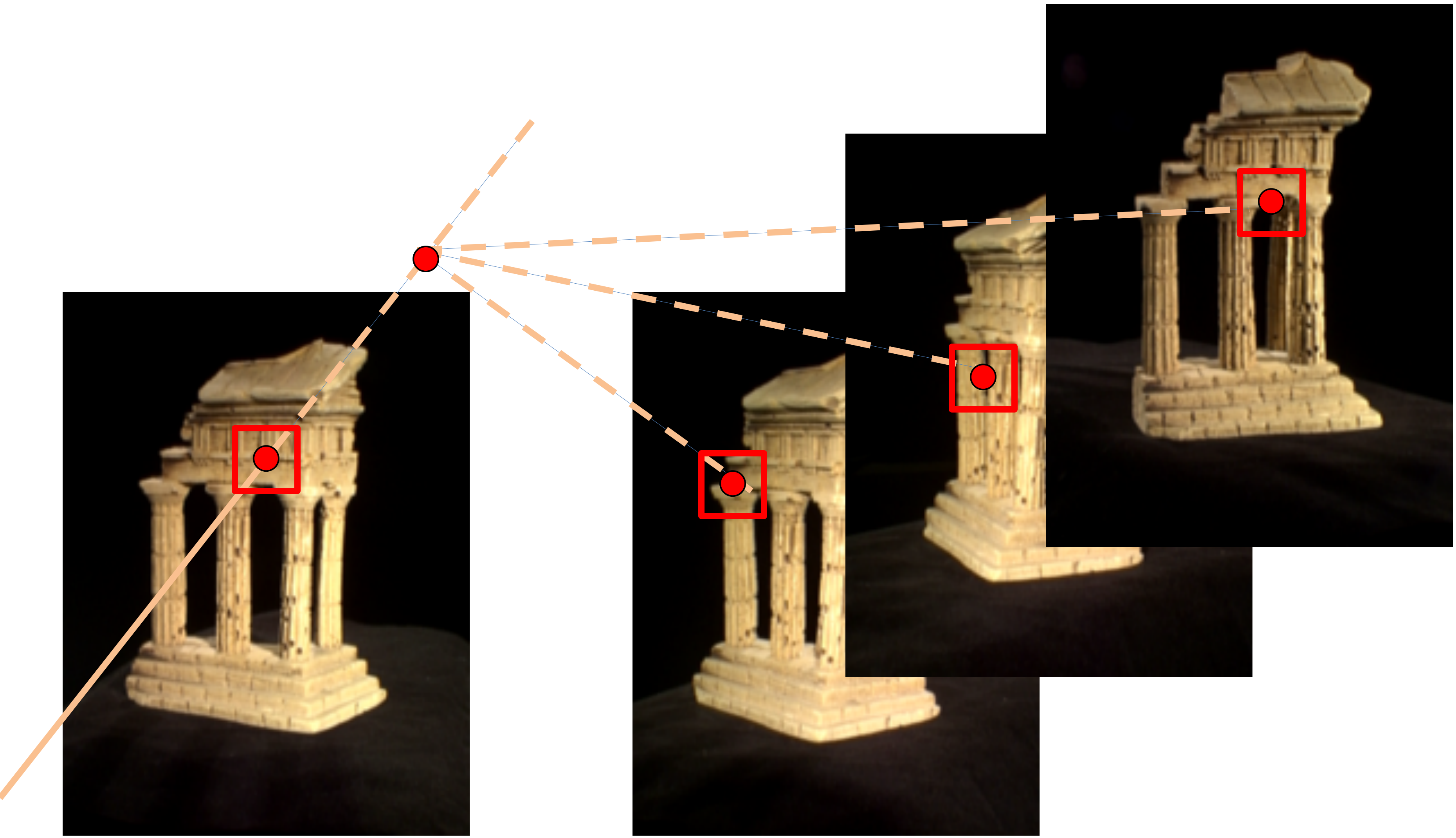


Correspondence search



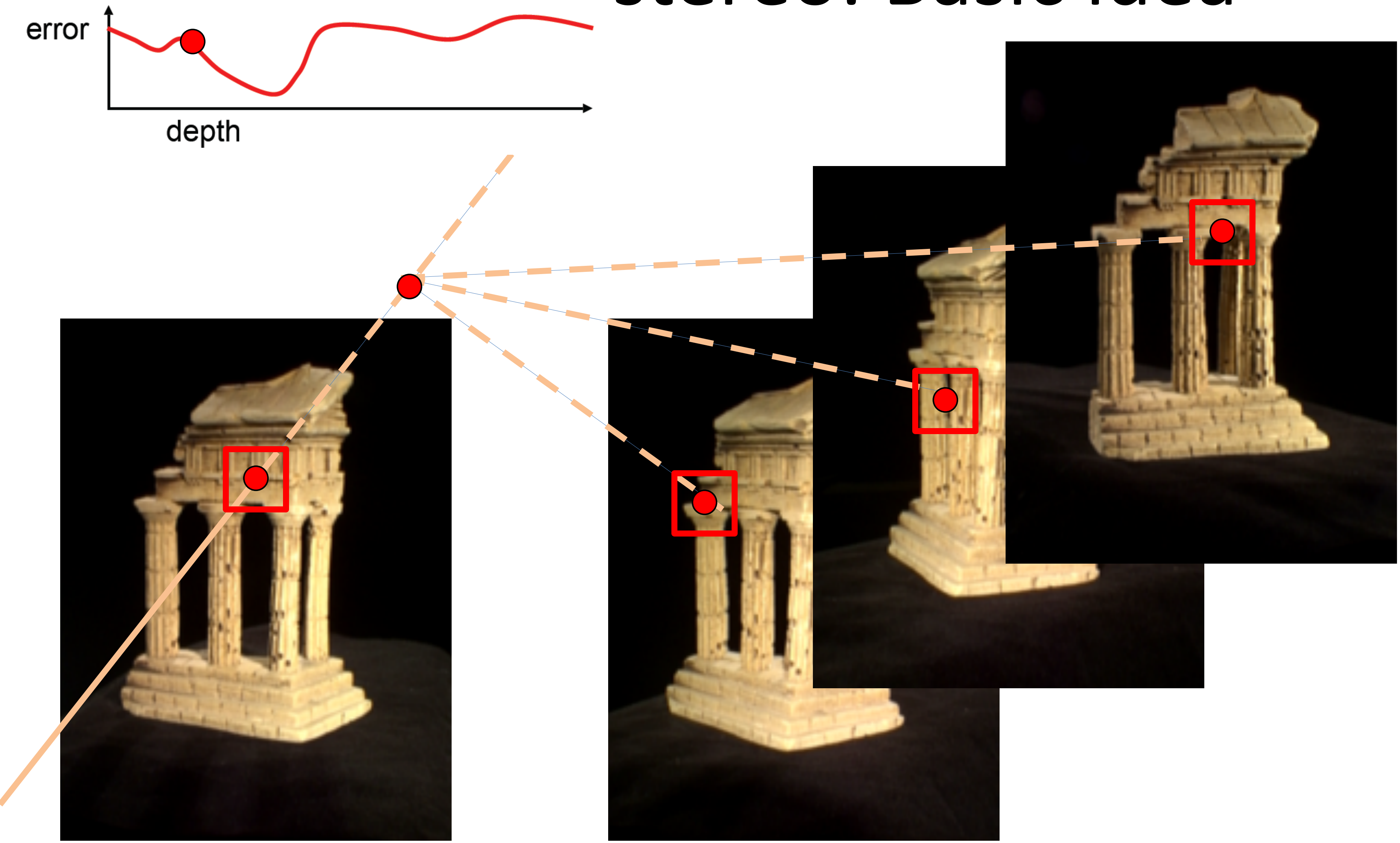
- Slide a window along the right scanline and compare contents of that window with the reference window in the left image
- Matching cost: SSD or normalized correlation

Multi-view stereo: Basic idea



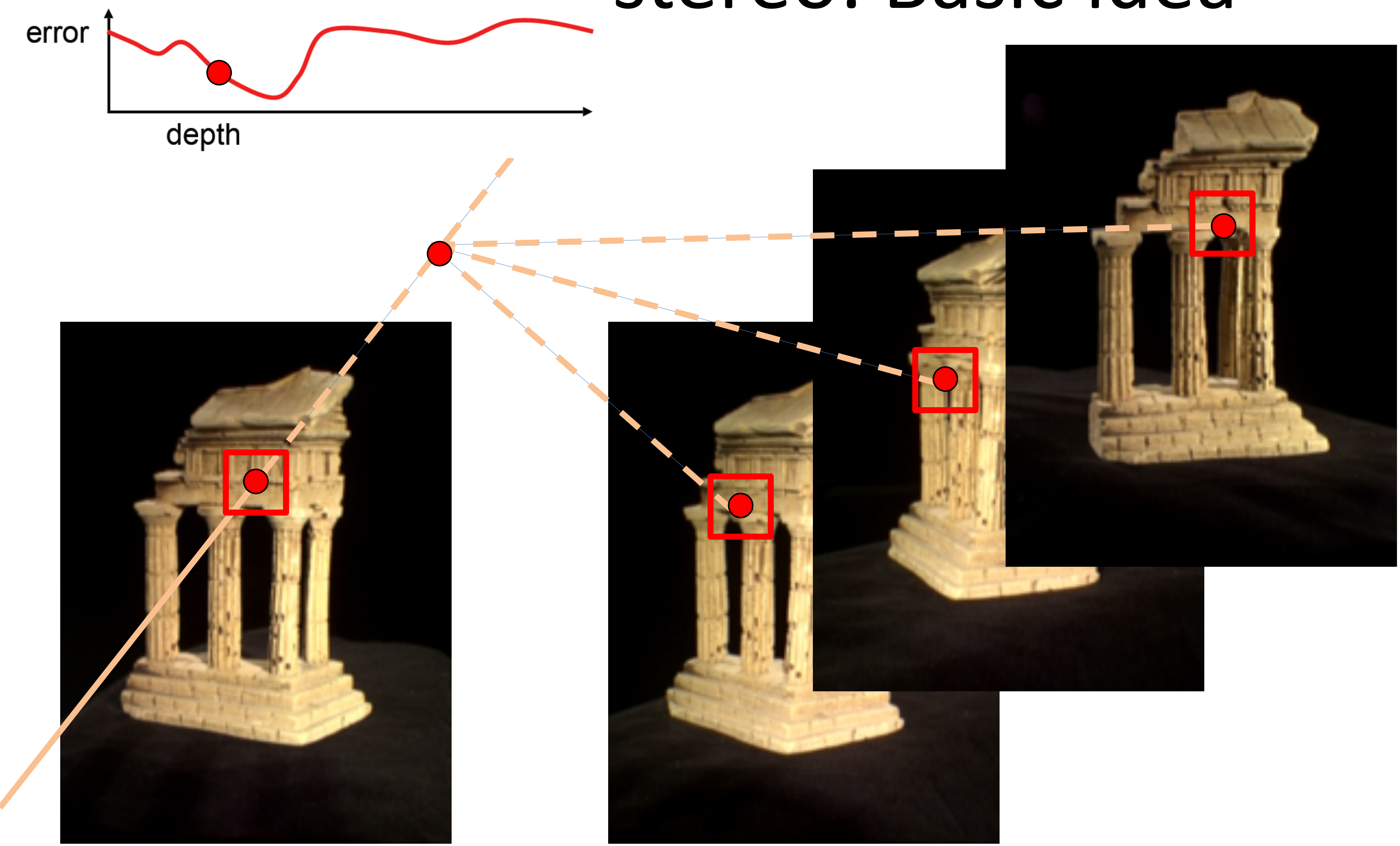
Source: Y. Furukawa

Multi-view stereo: Basic idea



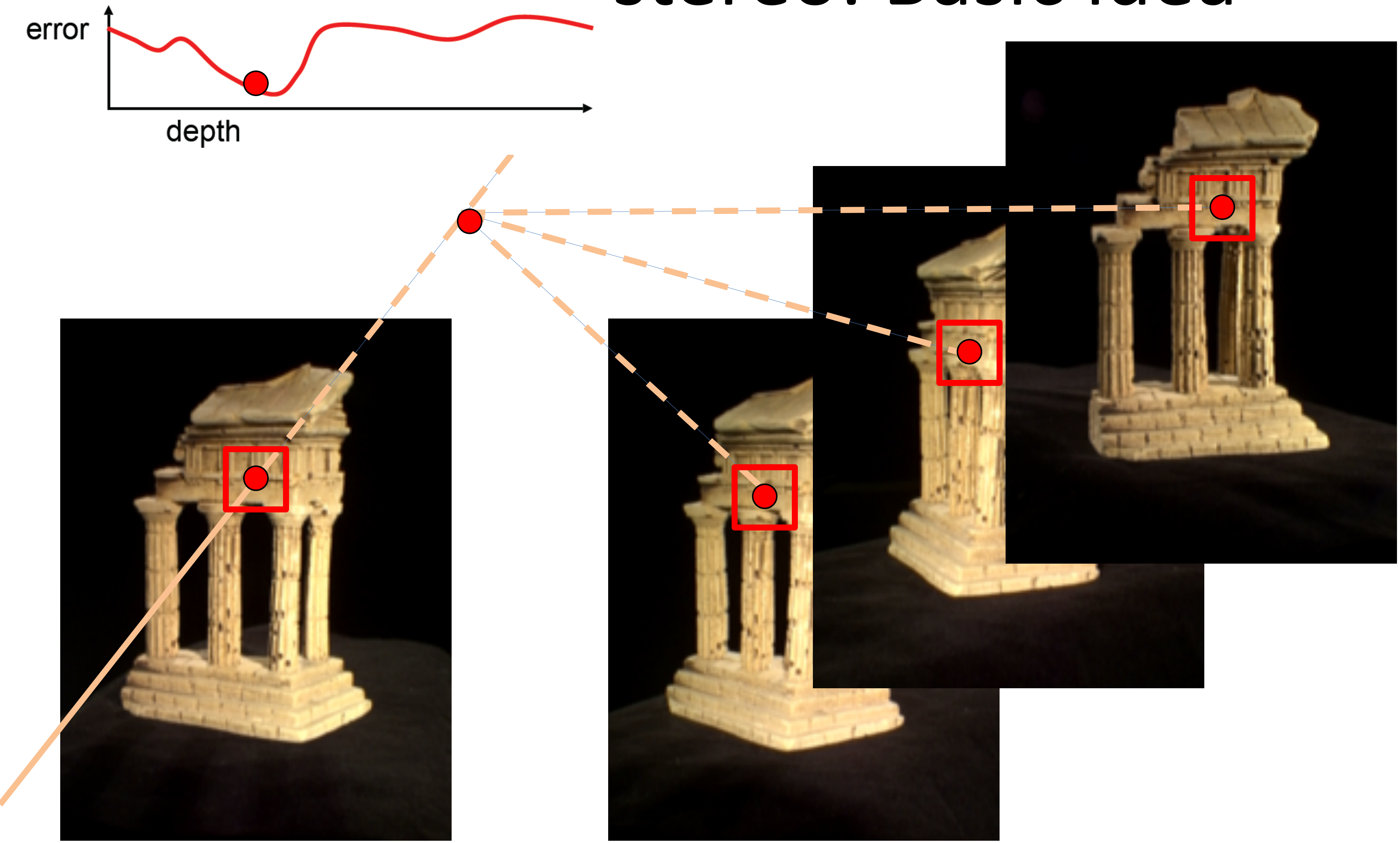
Source: Y. Furukawa

Multi-view stereo: Basic idea



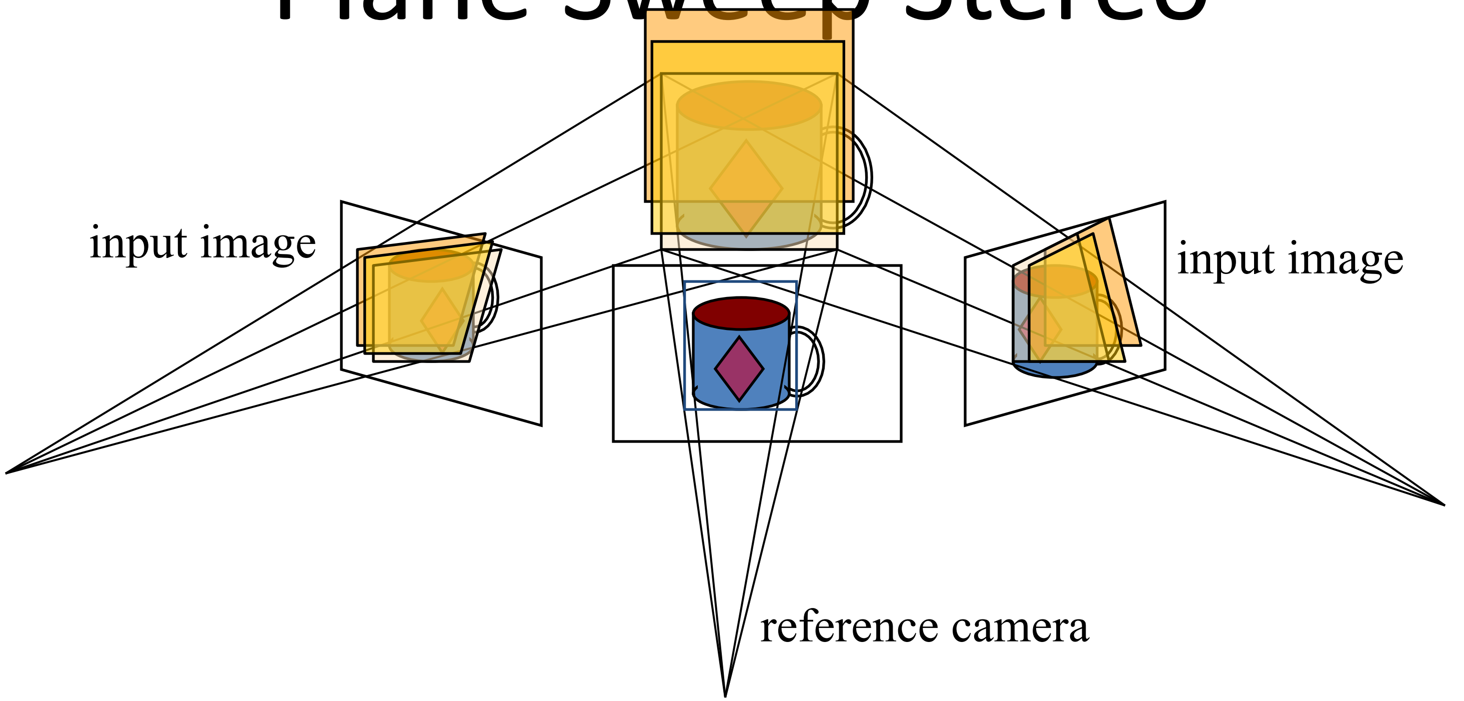
Source: Y. Furukawa

Multi-view stereo: Basic idea



Source: Y. Furukawa

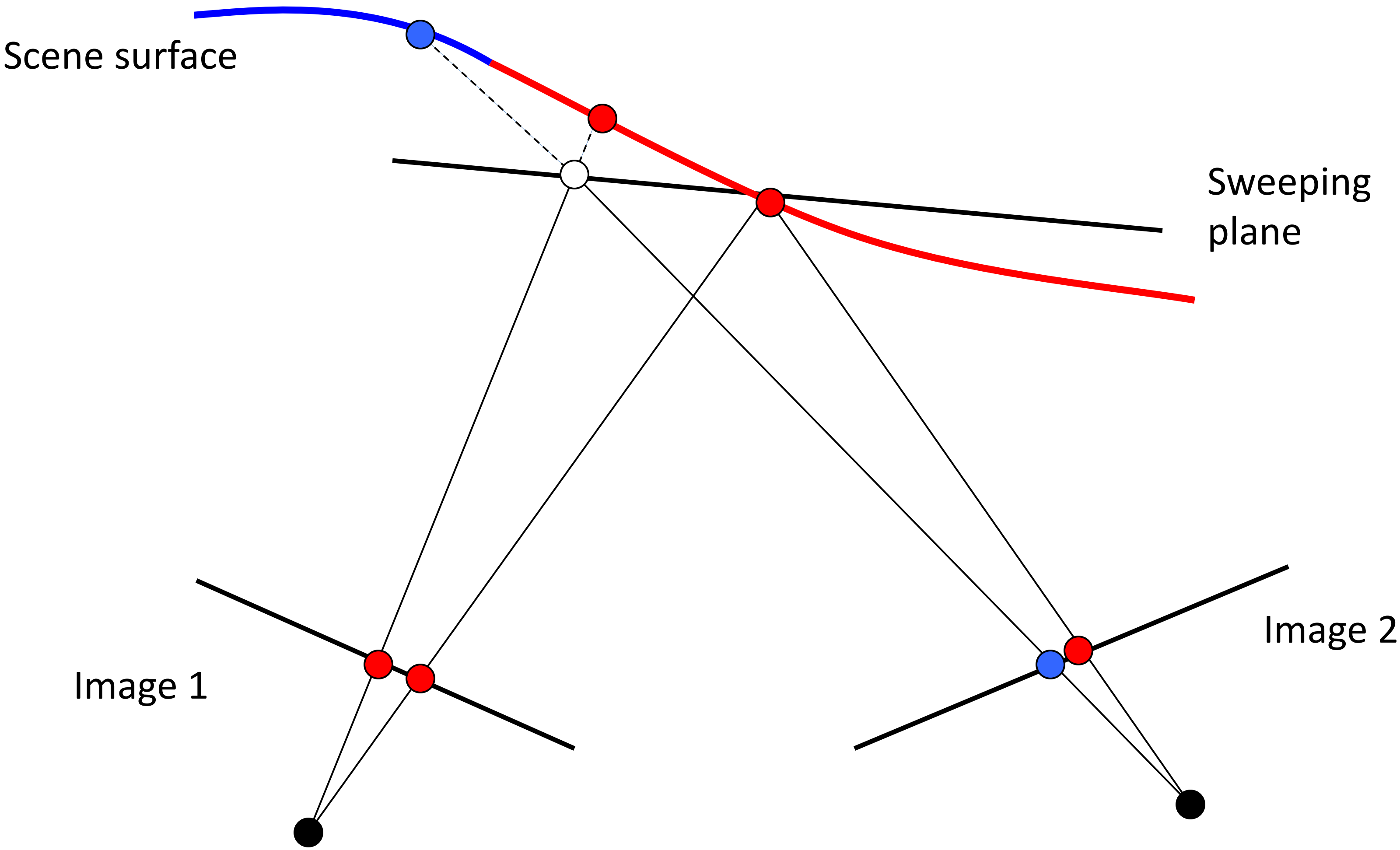
Plane Sweep Stereo



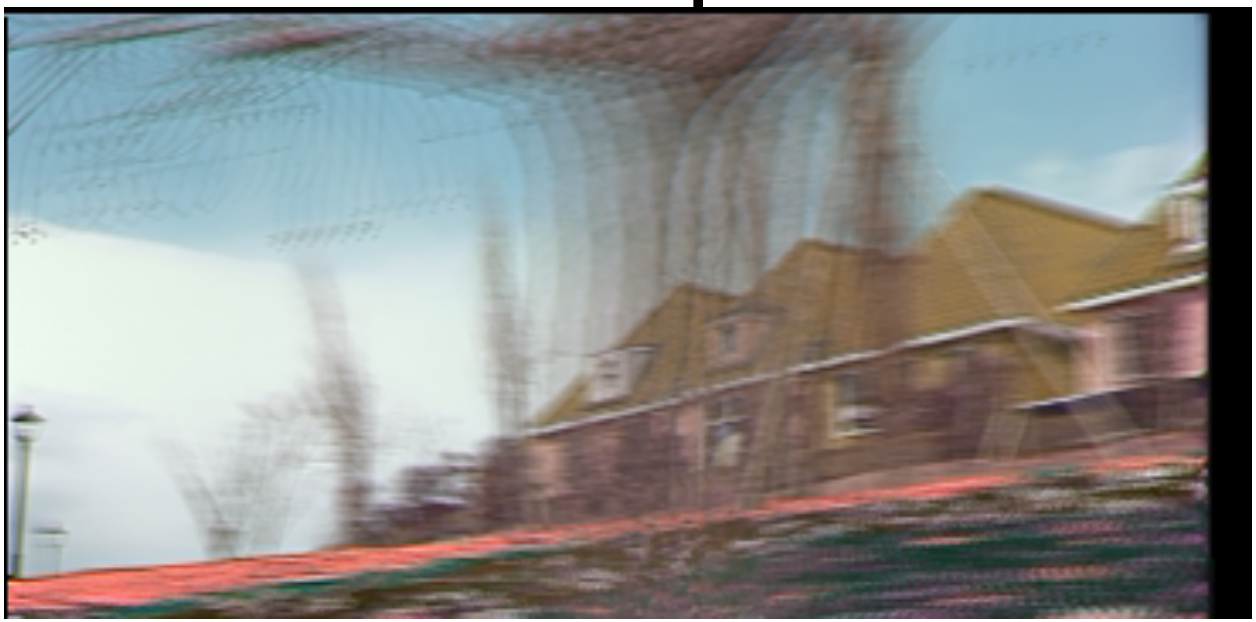
- Sweep family of planes at different depths w.r.t. a reference camera
  - For each depth, project each input image onto that plane
  - This is equivalent to a homography warping each input image into the reference view
- What can we say about the scene points that are at the right depth?

R. Collins. [A space-sweep approach to true multi-image matching.]() CVPR 1996.

Plane Sweep Stereo



Plane Sweep Stereo



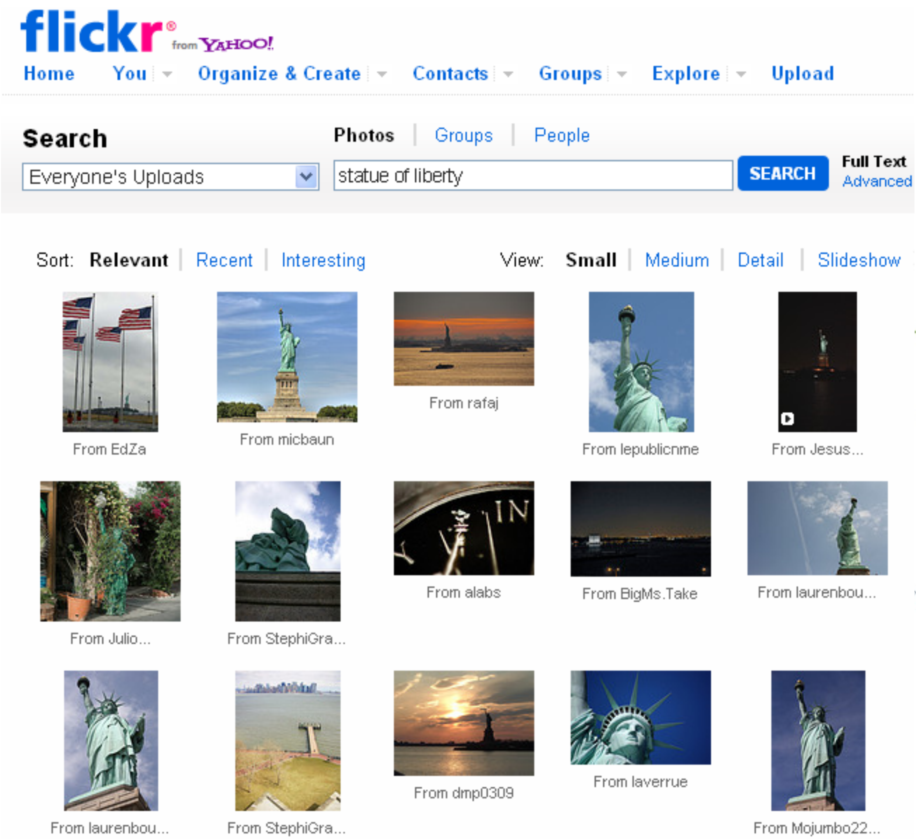
- For each depth plane
  - For each pixel in the composite image stack, compute the variance
- For each pixel, select the depth that gives the lowest variance

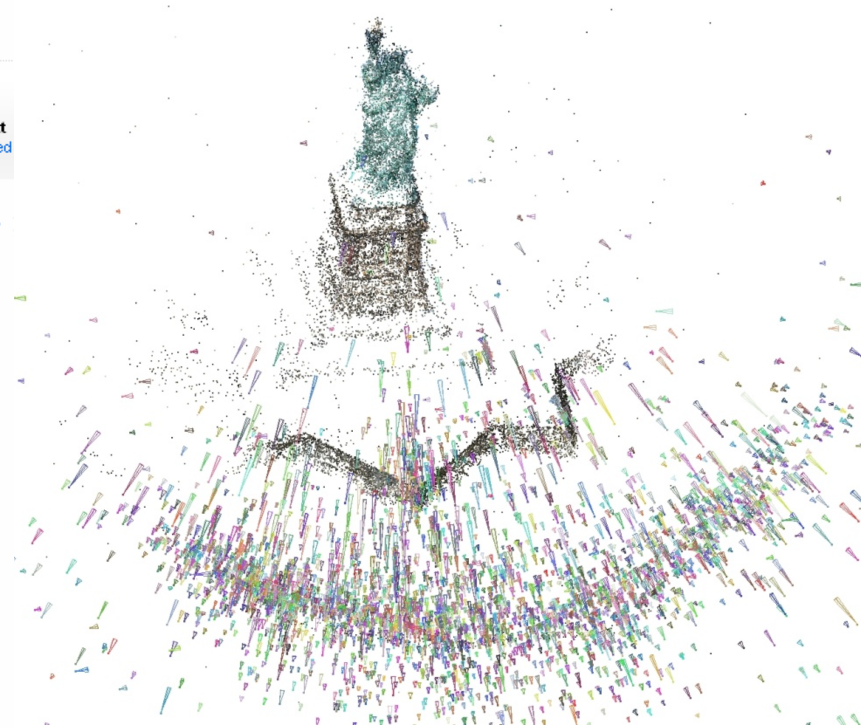
- Can be accelerated using graphics hardware

R. Yang and M. Pollefeys. *[Multi-Resolution Real-Time Stereo on Commodity Graphics Hardware]()*, CVPR 2003

Stereo from community photo collections

- Need *structure from motion* to recover unknown camera parameters
- Need *view selection* to find good groups of images on which to run dense stereo



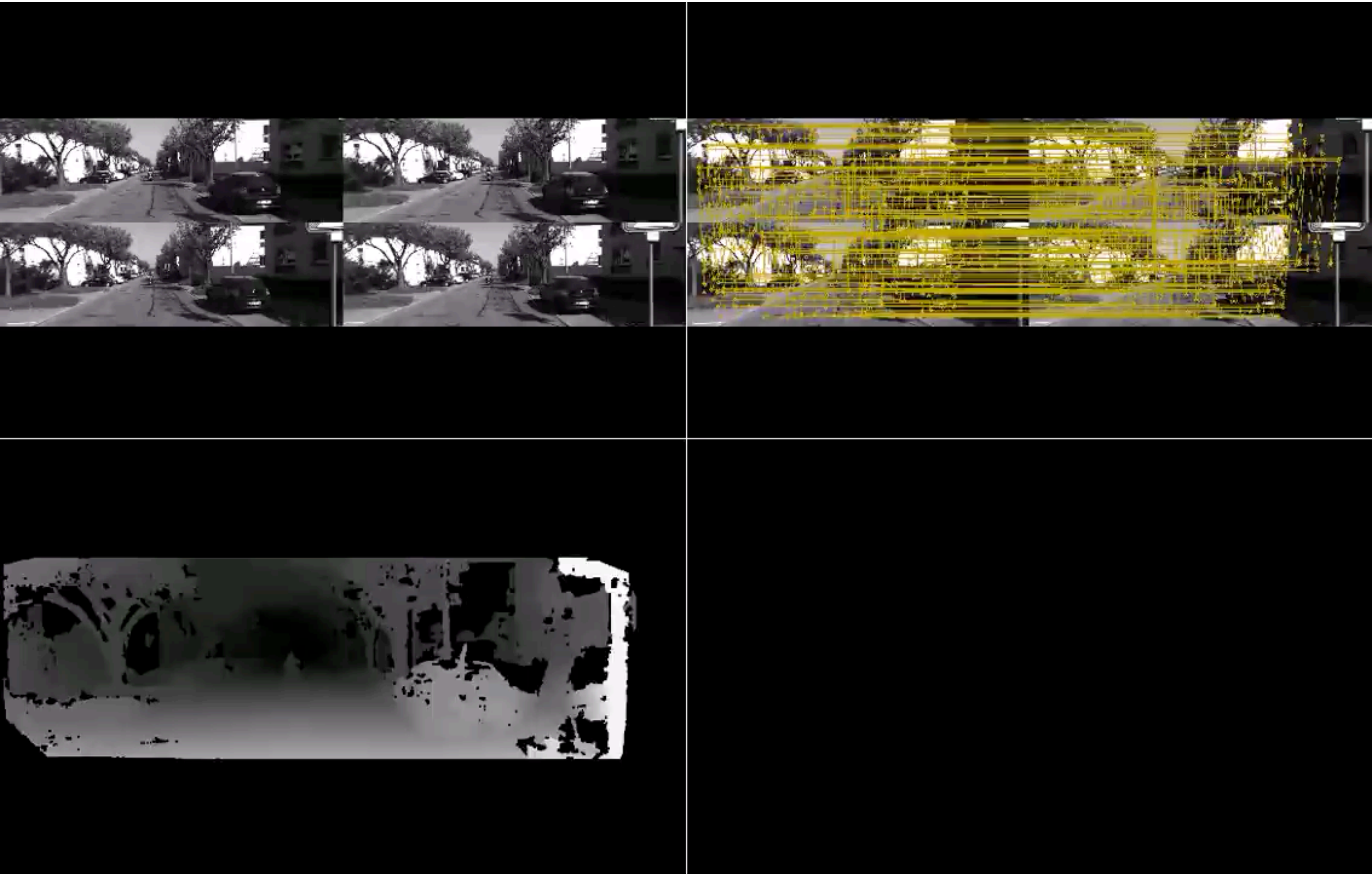


Towards Internet-Scale Multi-View Stereo

Yasutaka Furukawa, Brian Curless, Steven M. Seitz and Richard Szeliski, [Towards Internet-scale Multi-view Stereo](), CVPR 2010.

Internet-Scale Multi-View Stereo

Applications: SLAM



Applications – Hyperlapse

Applications: Visual Reality & Augmented Reality

Questions?