CS4501: Introduction to Computer Vision

3D Vision: Camera Calibration and Dense Stereo

Various slides from previous courses by:

D.A. Forsyth (Berkeley / UIUC), I. Kokkinos (Ecole Centrale / UCL). S. Lazebnik (UNC / UIUC), S. Seitz (MSR / Facebook), J. Hays (Brown / Georgia Tech), A. Berg (Stony Brook / UNC), D. Samaras (Stony Brook) . J. M. Frahm (UNC), V. Ordonez (UVA), Steve Seitz (UW).

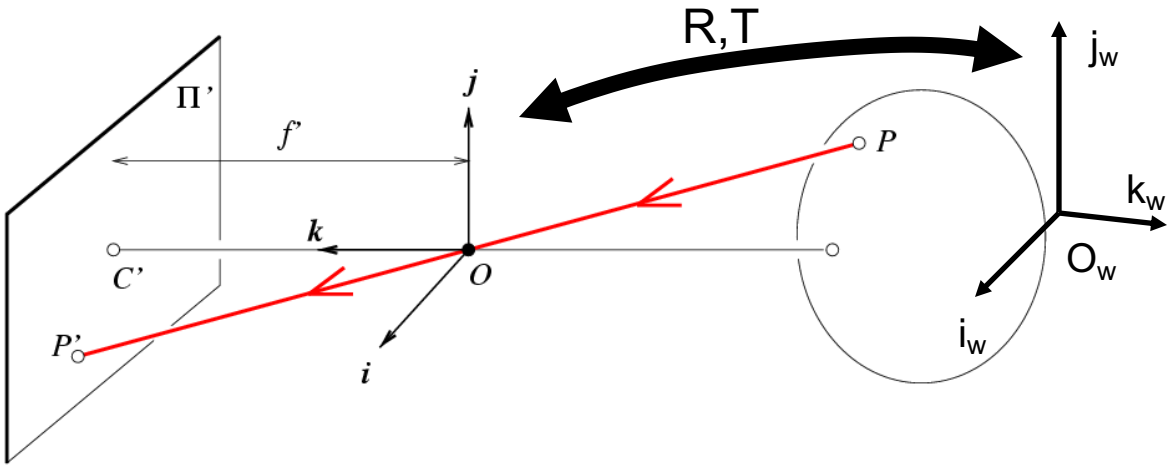
Today's Class

- Camera Calibration
- Stereo Vision – Dense Stereo / Stereo Matching

Camera Calibration

- What does it mean?

Recall the Projection matrix



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

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

Recall the Projection matrix

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

$\mathbf{X} =$

```
# Definition of the faces of the cube.
cube_pts = np.array(
                [[[0,0,0], [0,0,1], [0,1,1], [0,1,0], [0,0,0]],   # Face 1.
                 [[0,0,0], [0,1,0], [1,1,0], [1,0,0], [0,0,0]],   # Face 2.
                 [[1,0,0], [1,0,1], [1,1,1], [1,1,0], [1,0,0]],   # Face 3.
                 [[0,0,1], [0,1,1], [1,1,1], [1,0,1], [0,0,1]]])  # Face 4.
```

$\mathbf{K}[\mathbf{R}\ \mathbf{t}] =$

```
# Intrinsic Camera Matrix.
f = 3.0 # focal length.
K = np.array([[f, 0, 0],
              [0, f, 0],
              [0, 0, 1]])

# Extrinsic Camera Parameters.
Rt = np.array([[1, 0, 0, 1],
               [0, 1, 0, 1],
               [0, 0, 1, 4]])

# Camera matrix.
Camera_matrix = np.dot(K, Rt)
```

Recall the Projection matrix

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

$\mathbf{X} =$

```
# Definition of the faces of the cube.
cube_pts = np.array(
                [[[0,0,0], [0,0,1], [0,1,1], [0,1,0], [0,0,0]],   # Face 1.
                 [[0,0,0], [0,1,0], [1,1,0], [1,0,0], [0,0,0]],   # Face 2.
                 [[1,0,0], [1,0,1], [1,1,1], [1,1,0], [1,0,0]],   # Face 3.
                 [[0,0,1], [0,1,1], [1,1,1], [1,0,1], [0,0,1]]])  # Face 4.
```

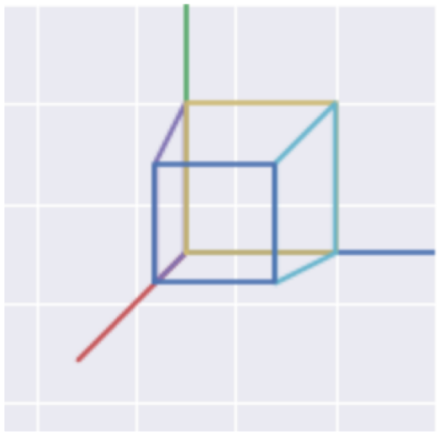
$\mathbf{K}[\mathbf{R}\ \mathbf{t}] =$

```
# Intrinsic Camera Matrix.
f = 3.0 # focal length.
K = np.array([[f, 0, 0],
              [0, f, 0],
              [0, 0, 1]])

# Extrinsic Camera Parameters.
Rt = np.array([[1, 0, 0, 1],
               [0, 1, 0, 1],
               [0, 0, 1, 4]])

# Camera matrix.
Camera_matrix = np.dot(K, Rt)
```

Goal: Find $\mathbf{x}$



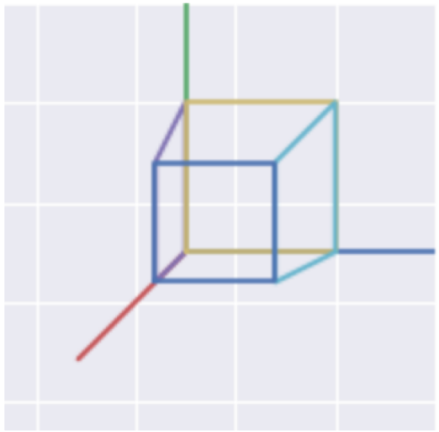
Camera Calibration

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

$\mathbf{X} =$

```
# Definition of the faces of the cube.
cube_pts = np.array(
                [[[0,0,0], [0,0,1], [0,1,1], [0,1,0], [0,0,0]],    # Face 1.
                [[0,0,0], [0,1,0], [1,1,0], [1,0,0], [0,0,0]],     # Face 2.
                [[1,0,0], [1,0,1], [1,1,1], [1,1,0], [1,0,0]],     # Face 3.
                [[0,0,1], [0,1,1], [1,1,1], [1,0,1], [0,0,1]]])    # Face 4.
```

$\mathbf{x} =$



Camera Calibration

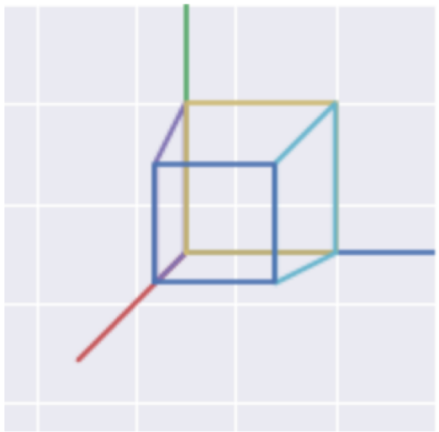
$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

$\mathbf{X} =$

```
# Definition of the faces of the cube.
cube_pts = np.array(
                [[[0,0,0], [0,0,1], [0,1,1], [0,1,0], [0,0,0]],   # Face 1.
                 [[0,0,0], [0,1,0], [1,1,0], [1,0,0], [0,0,0]],   # Face 2.
                 [[1,0,0], [1,0,1], [1,1,1], [1,1,0], [1,0,0]],   # Face 3.
                 [[0,0,1], [0,1,1], [1,1,1], [1,0,1], [0,0,1]]])  # Face 4.
```

Goal: Find $\mathbf{K}[\mathbf{R}\ \mathbf{t}]$

$\mathbf{x} =$

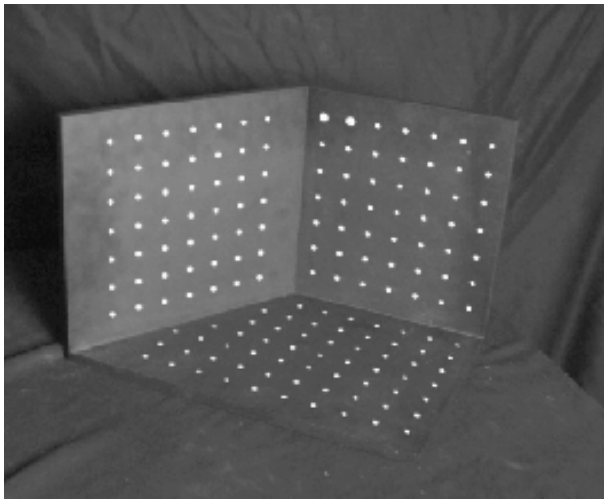


Slide Credit: James Hays

Calibrating the Camera

Use an scene with known geometry

- Correspond image points to 3d points
- Get least squares solution (or non-linear solution)



Known 2d image coords

Known 3d locations

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

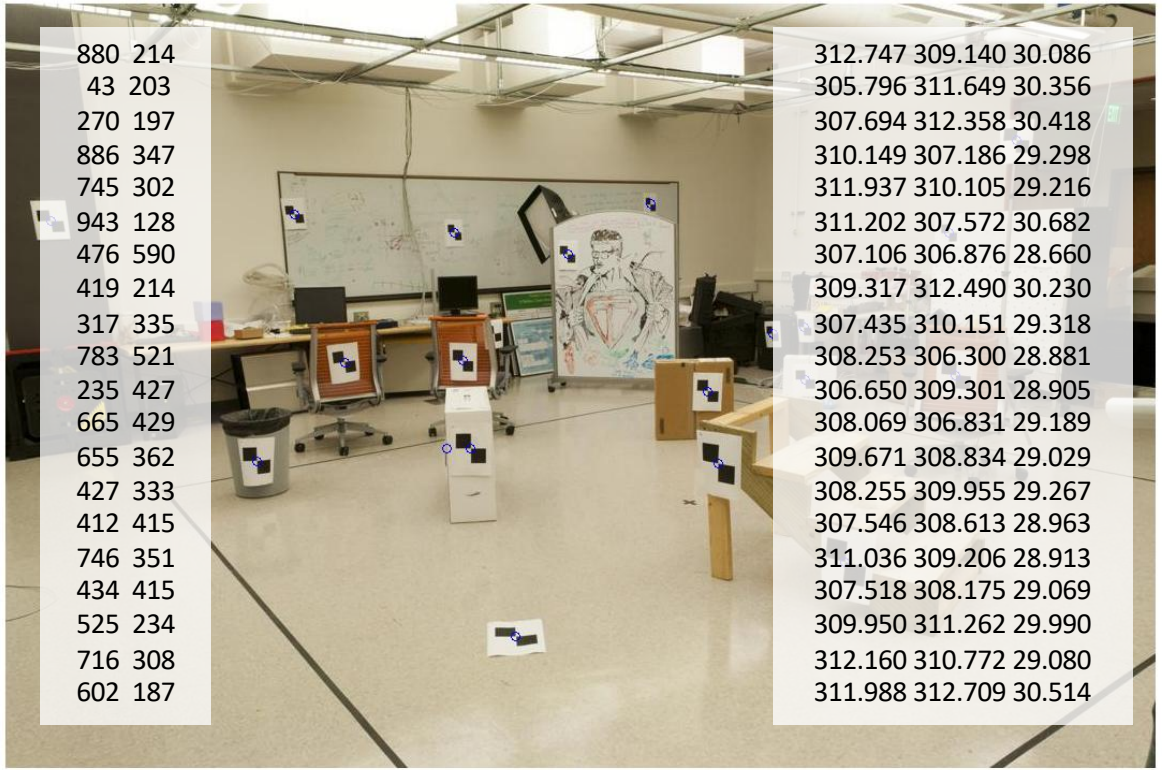
Unknown Camera Parameters

How do we calibrate a camera?

Slide Credit: James Hays

Known 2d image coords

880 214
43 203
270 197
886 347
745 302
943 128
476 590
419 214
317 335
783 521
235 427
665 429
655 362
427 333
412 415
746 351
434 415
525 234
716 308
602 187



Known 3d locations

312.747 309.140 30.086
305.796 311.649 30.356
307.694 312.358 30.418
310.149 307.186 29.298
311.937 310.105 29.216
311.202 307.572 30.682
307.106 306.876 28.660
309.317 312.490 30.230
307.435 310.151 29.318
308.253 306.300 28.881
306.650 309.301 28.905
308.069 306.831 29.189
309.671 308.834 29.029
308.255 309.955 29.267
307.546 308.613 28.963
311.036 309.206 28.913
307.518 308.175 29.069
309.950 311.262 29.990
312.160 310.772 29.080
311.988 312.709 30.514

Unknown Camera Parameters

Slide Credit: James Hays

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

$$su = m_{11}X + m_{12}Y + m_{13}Z + m_{14}$$

$$sv = m_{21}X + m_{22}Y + m_{23}Z + m_{24}$$

$$s = m_{31}X + m_{32}Y + m_{33}Z + m_{34}$$

$$u = \frac{m_{11}X + m_{12}Y + m_{13}Z + m_{14}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}$$

$$v = \frac{m_{21}X + m_{22}Y + m_{23}Z + m_{24}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}$$

Unknown Camera Parameters

Slide Credit: James Hays

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

$$u = \frac{m_{11}X + m_{12}Y + m_{13}Z + m_{14}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}$$

$$v = \frac{m_{21}X + m_{22}Y + m_{23}Z + m_{24}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}$$

$$(m_{31}X + m_{32}Y + m_{33}Z + m_{34})u = m_{11}X + m_{12}Y + m_{13}Z + m_{14}$$
$$(m_{31}X + m_{32}Y + m_{33}Z + m_{34})v = m_{21}X + m_{22}Y + m_{23}Z + m_{24}$$

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

Unknown Camera Parameters

Slide Credit: James Hays

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

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

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

Unknown Camera Parameters

Slide Credit: James Hays

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

$$0 = m_{11}X + m_{12}Y + m_{13}Z + m_{14} - m_{31}uX - m_{32}uY - m_{33}uZ - m_{34}u$$

$$0 = m_{21}X + m_{22}Y + m_{23}Z + m_{24} - m_{31}vX - m_{32}vY - m_{33}vZ - m_{34}v$$

- Method 1 – homogeneous linear system. Solve for m's entries using linear least squares

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

```
[U, S, V] = svd(A);
M = V(:,end);
M = reshape(M,[],3)';
```

Unknown Camera Parameters

Slide Credit: James Hays

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

- Method 2 – nonhomogeneous linear system. Solve for m's entries using linear least squares

Ax=b form

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

```
M = A\Y;
M = [M;1];
M = reshape(M,[],3)';
```

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

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} = \mathbf{K}[\mathbf{R}\ \mathbf{t}]$$

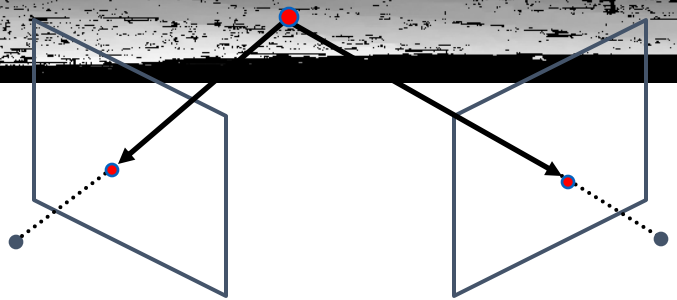
- Yes!
- You can use *RQ* factorization (note – not the more familiar *QR* factorization). *R* (right diagonal) is K, and *Q* (orthogonal basis) is R. T, the last column of [R | T], is inv(K) * last column of M.
  - But you need to do a bit of post-processing to make sure that the matrices are valid. See http://ksimek.github.io/2012/08/14/decompose/

Credit: James Hays

Stereo: Epipolar geometry

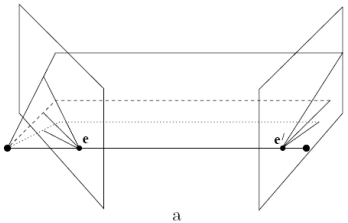
Vicente Ordonez

University of Virginia

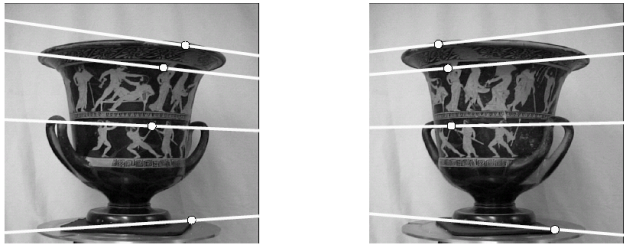


Slides by Kristen Grauman

Multiple views

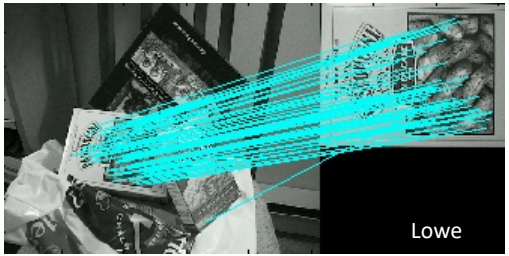


a



Hartley and Zisserman

Multi-view geometry, matching, invariant features, stereo vision



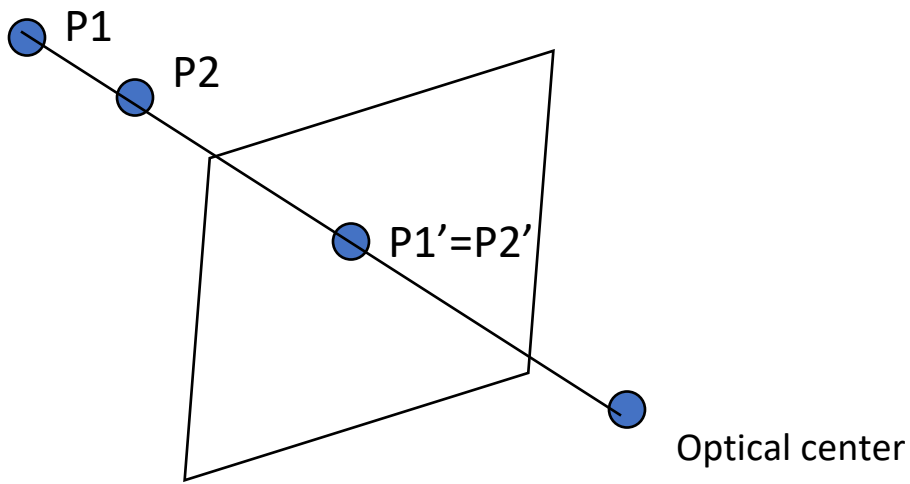
Why multiple views?

- Structure and depth are inherently ambiguous from single views.

Images from Lana Lazebnik

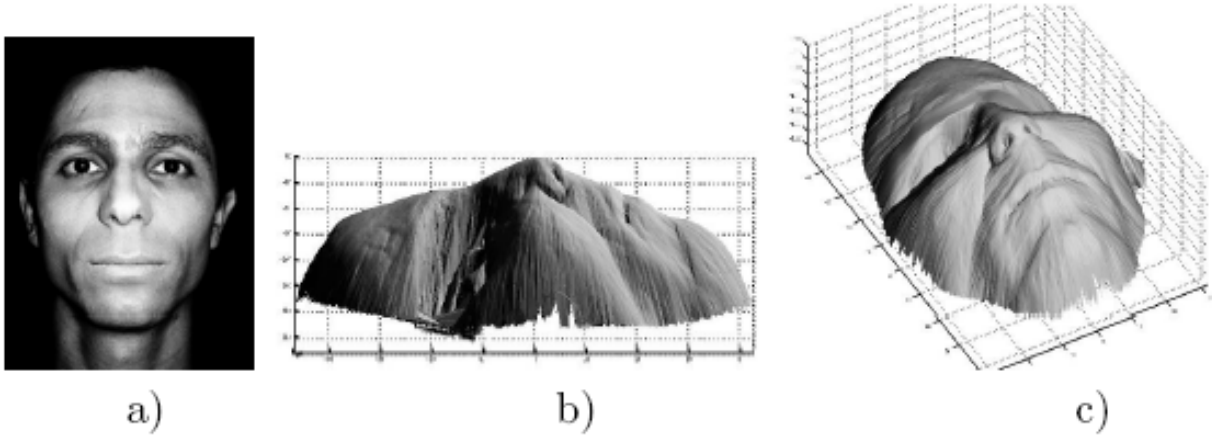
Why multiple views?

- Structure and depth are inherently ambiguous from single views.



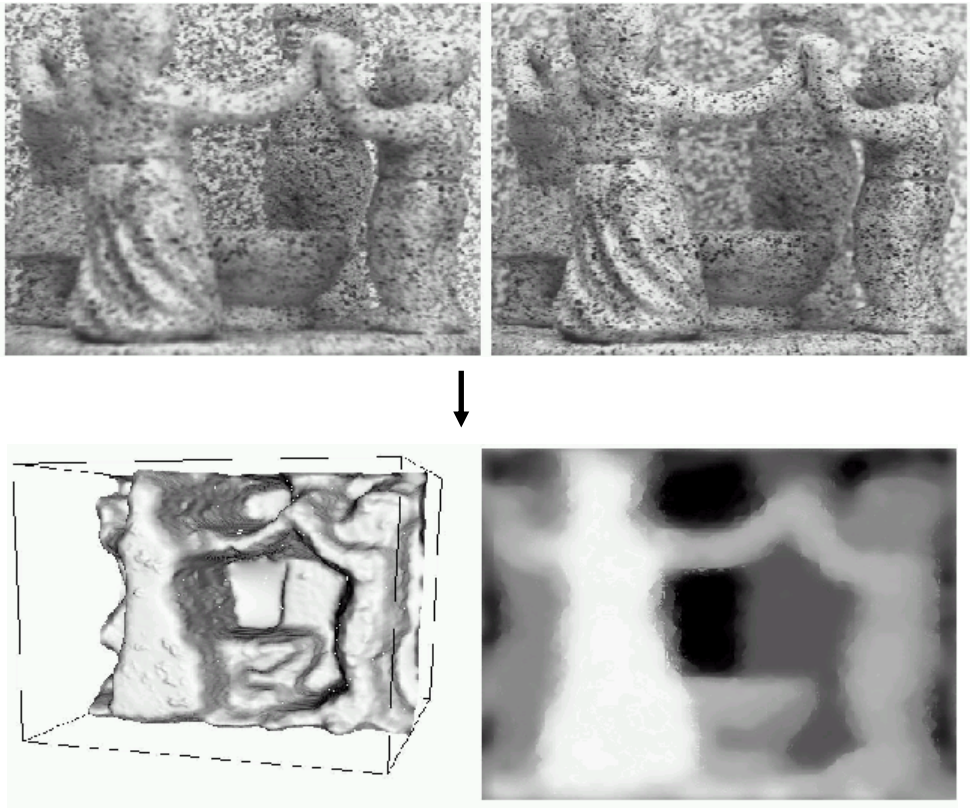
- What cues help us to perceive 3d shape and depth?

Shading



[Figure from Prados & Faugeras 2006]

Focus/defocus

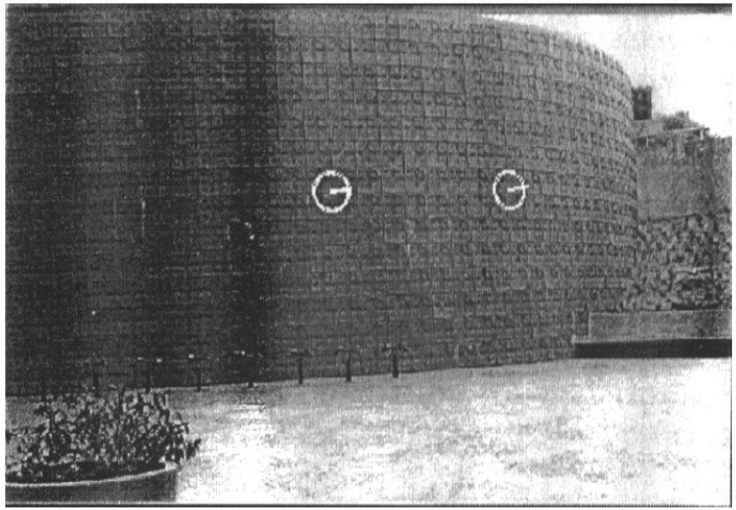


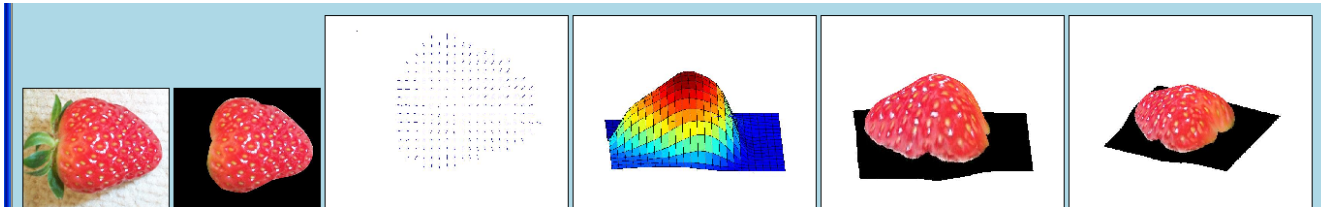
Images from same point of view, different camera parameters

3d shape / depth estimates

[figs from H. Jin and P. Favaro, 2002]

Texture



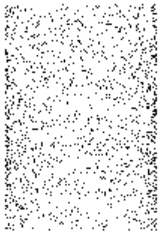


[From A.M. Loh. The recovery of 3-D structure using visual texture patterns. PhD thesis]

Perspective effects

Image credit: S. Seitz

Motion



Figures from L. Zhang

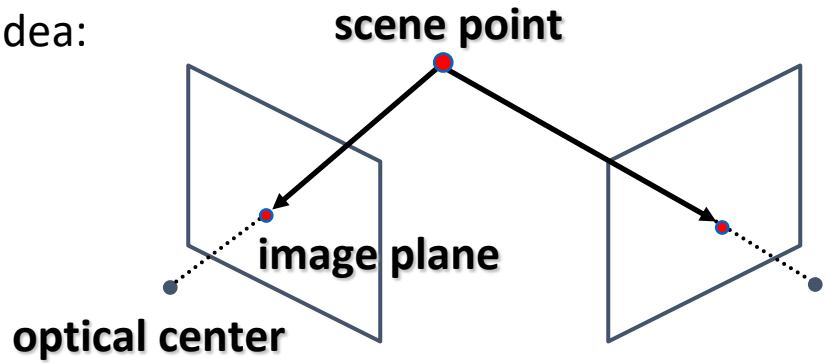
http://www.brainconnection.com/teasers/?main=illusion/motion-shape

Estimating scene shape

- "Shape from X": Shading, Texture, Focus, Motion…

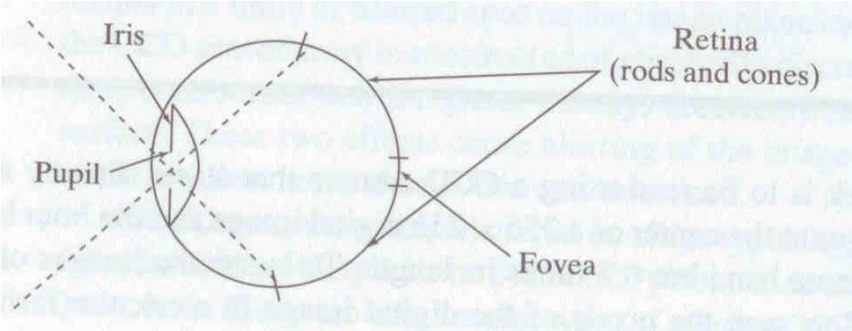
- **Stereo**:
  - shape from "motion" between two views
  - infer 3d shape of scene from two (multiple) images from different viewpoints

Main idea:



Human eye

Rough analogy with human visual system:



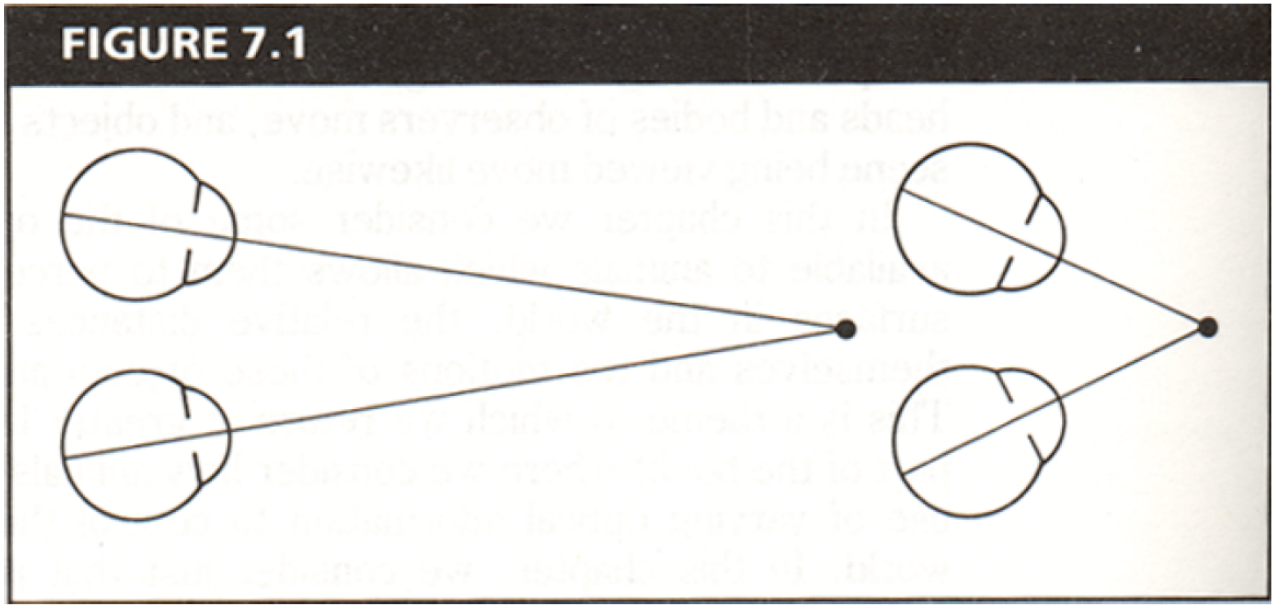
Pupil/Iris – control amount of light passing through lens

Retina - contains sensor cells, where image is formed

Fovea – highest concentration of cones

Fig from Shapiro and Stockman

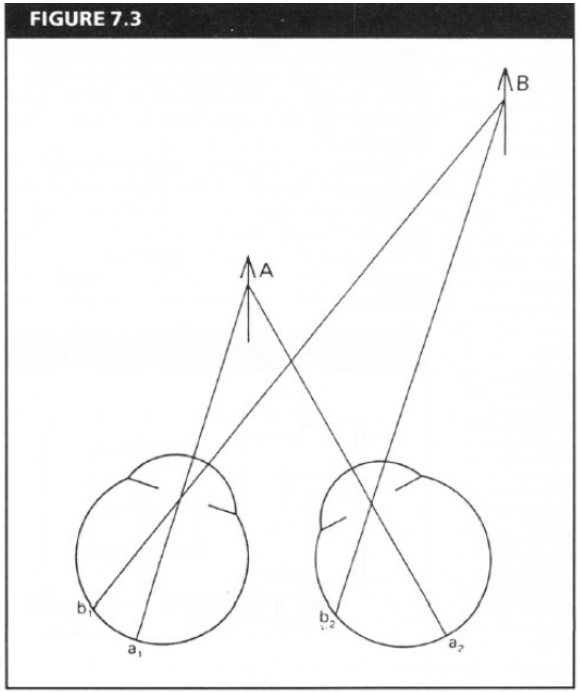
Human stereopsis: disparity



From Bruce and Green, Visual Perception, Physiology, Psychology and Ecology

Human eyes **fixate** on point in space – rotate so that corresponding images form in centers of fovea.

Human stereopsis: disparity



From Bruce and Green, Visual Perception, Physiology, Psychology and Ecology

Disparity occurs when eyes fixate on one object; others appear at different visual angles

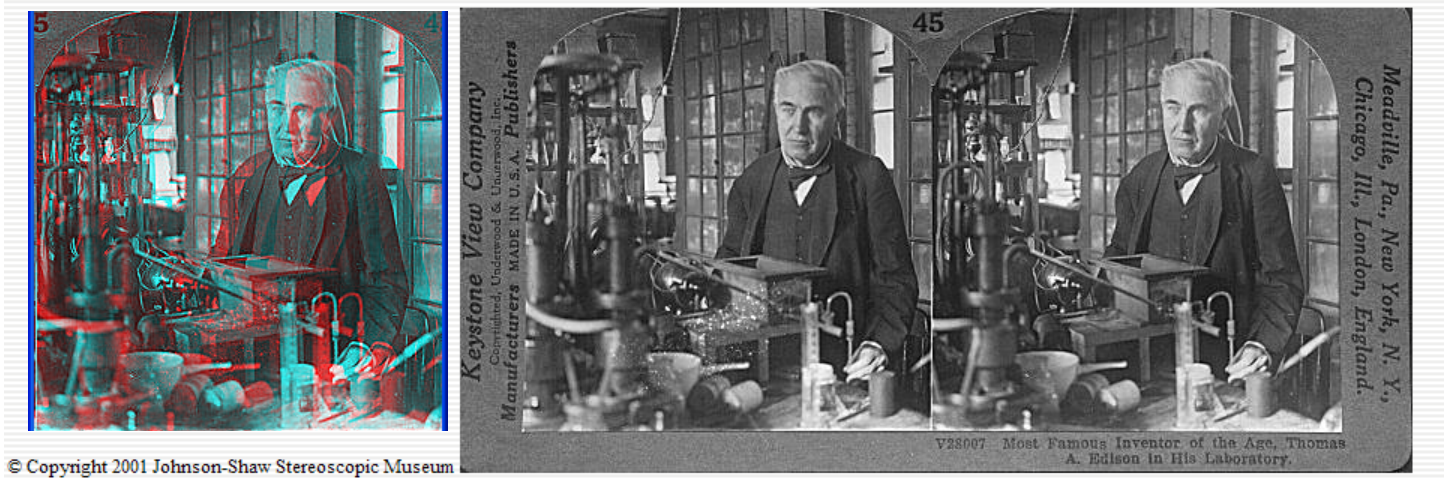
Adapted from David Forsyth, UC Berkeley

Stereo photography and stereo viewers

Take two pictures of the same subject from two slightly different viewpoints and display so that each eye sees only one of the images.

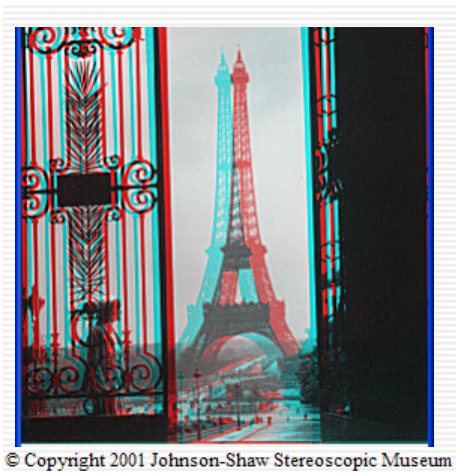
Invented by Sir Charles Wheatstone, 1838

Image from fisher-price.com



© Copyright 2001 Johnson-Shaw Stereoscopic Museum

http://www.johnsonshawmuseum.org



© Copyright 2001 Johnson-Shaw Stereoscopic Museum

http://www.johnsonshawmuseum.org

Public Library, Stereoscopic Looking Room, Chicago, by Phillips, 1923

http://www.well.com/~jimg/stereo/stereo_list.html

Autostereograms

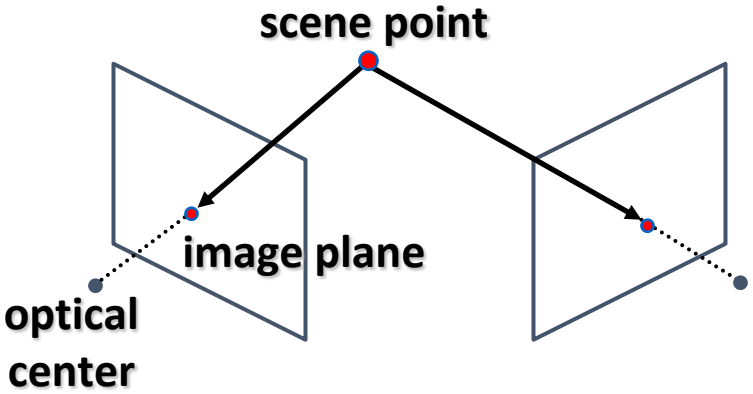
Exploit disparity as depth cue using single image.

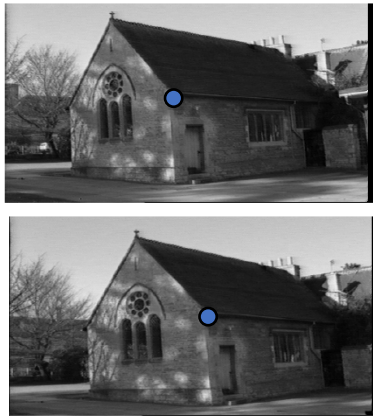
(Single image random dot stereogram, Single image stereogram)

Images from magiceye.com

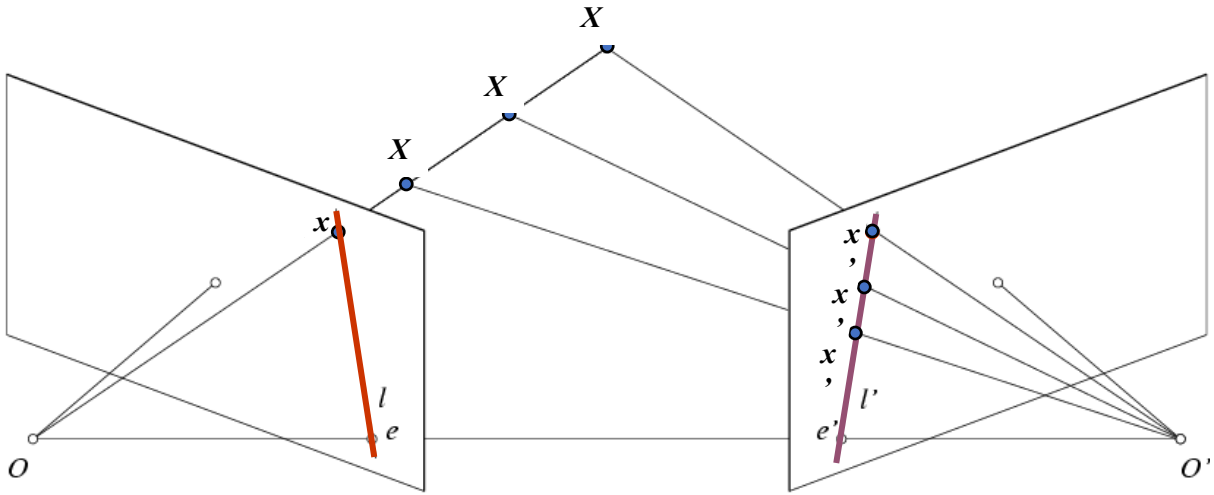
Estimating depth with stereo

- **Stereo**: shape from “motion” between two views
- We’ll need to consider:
  - Info on camera pose (“calibration”)
  - Image point correspondences





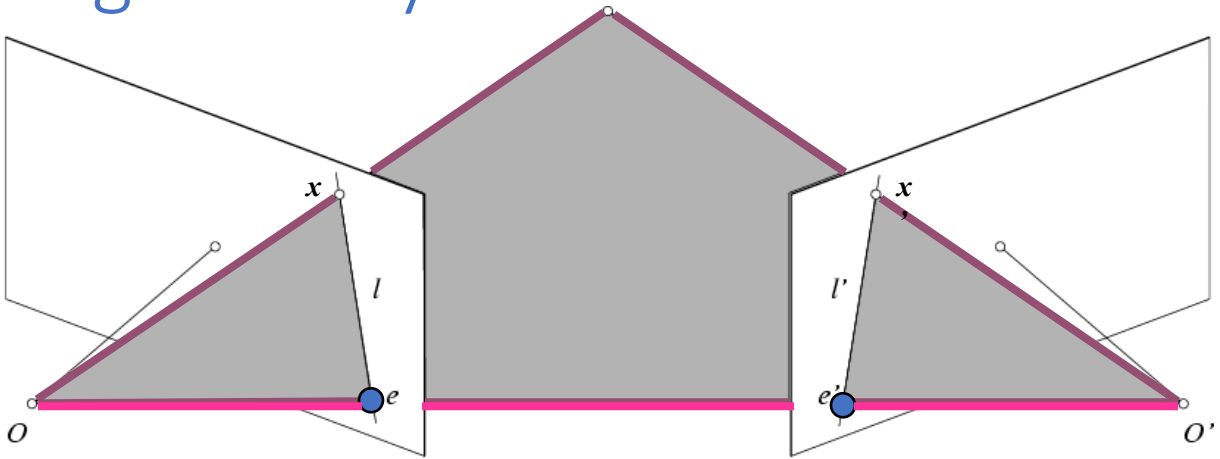
Key idea: Epipolar constraint



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



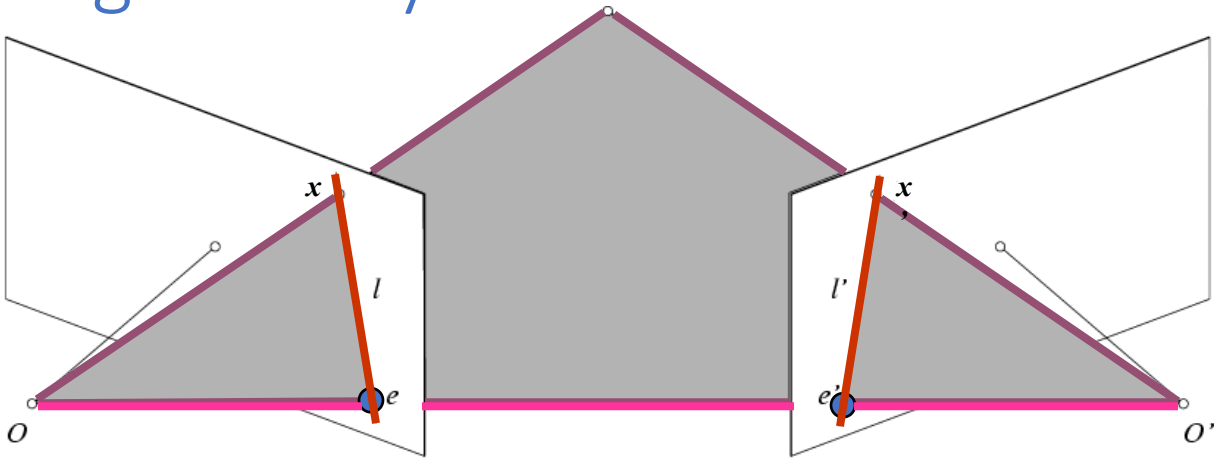
- **Baseline** – line connecting the two camera centers
- **Epipoles**

= intersections of baseline with image planes

= projections of the other camera center

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

Epipolar geometry: notation



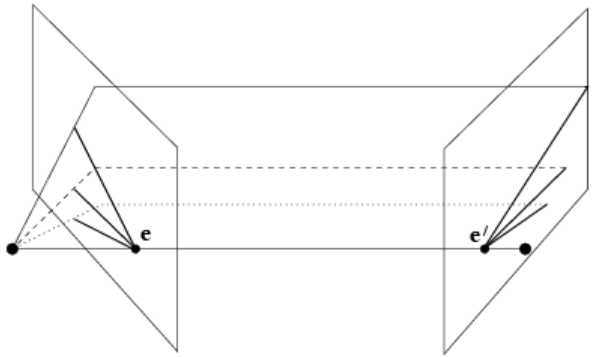
- **Baseline** – line connecting the two camera centers
- **Epipoles**

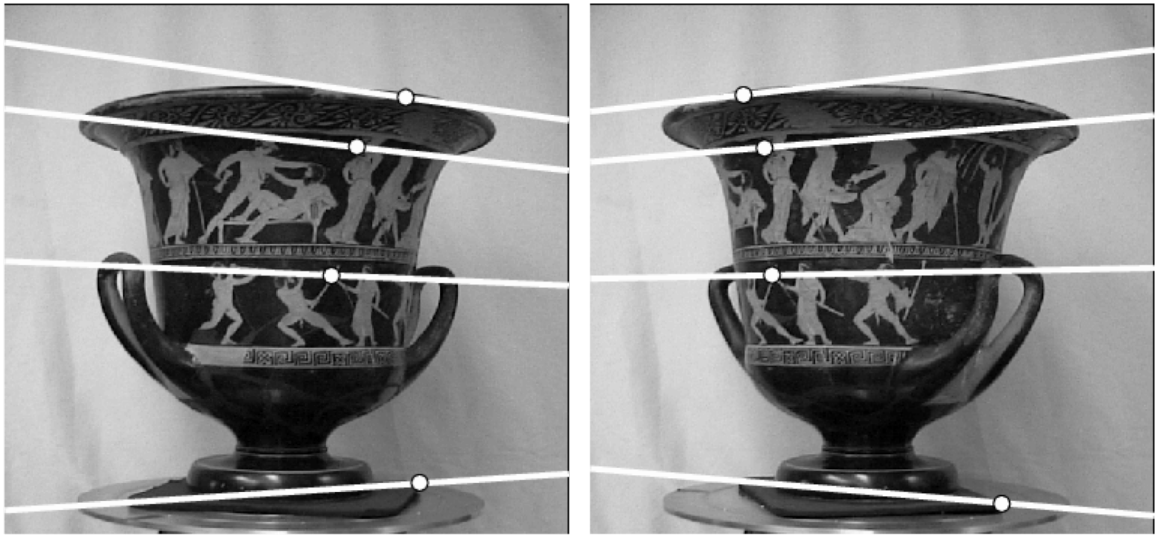
= intersections of baseline with image planes

= projections of the other camera center

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

Example: Converging cameras

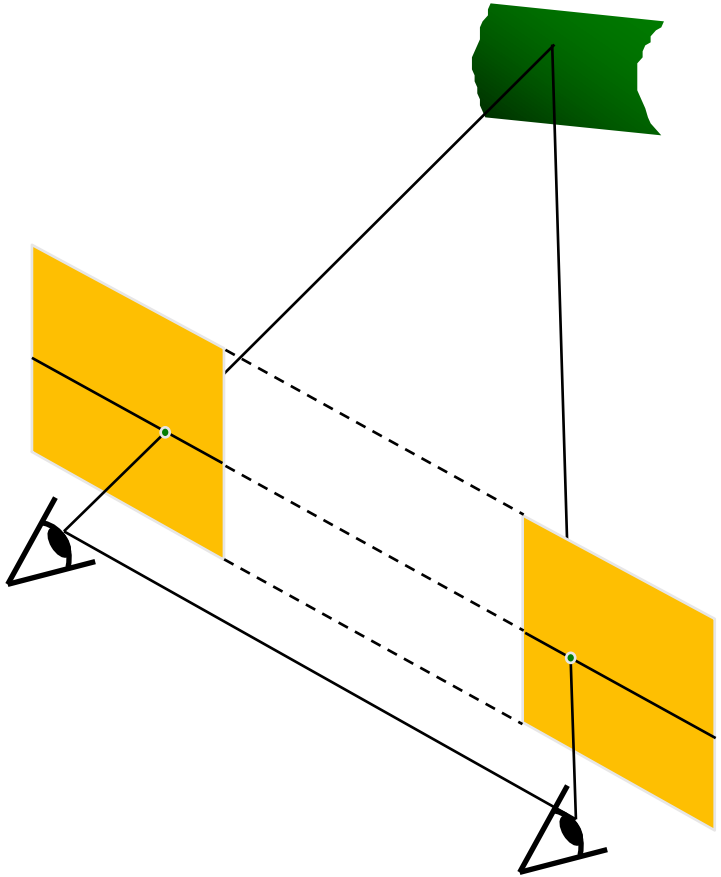




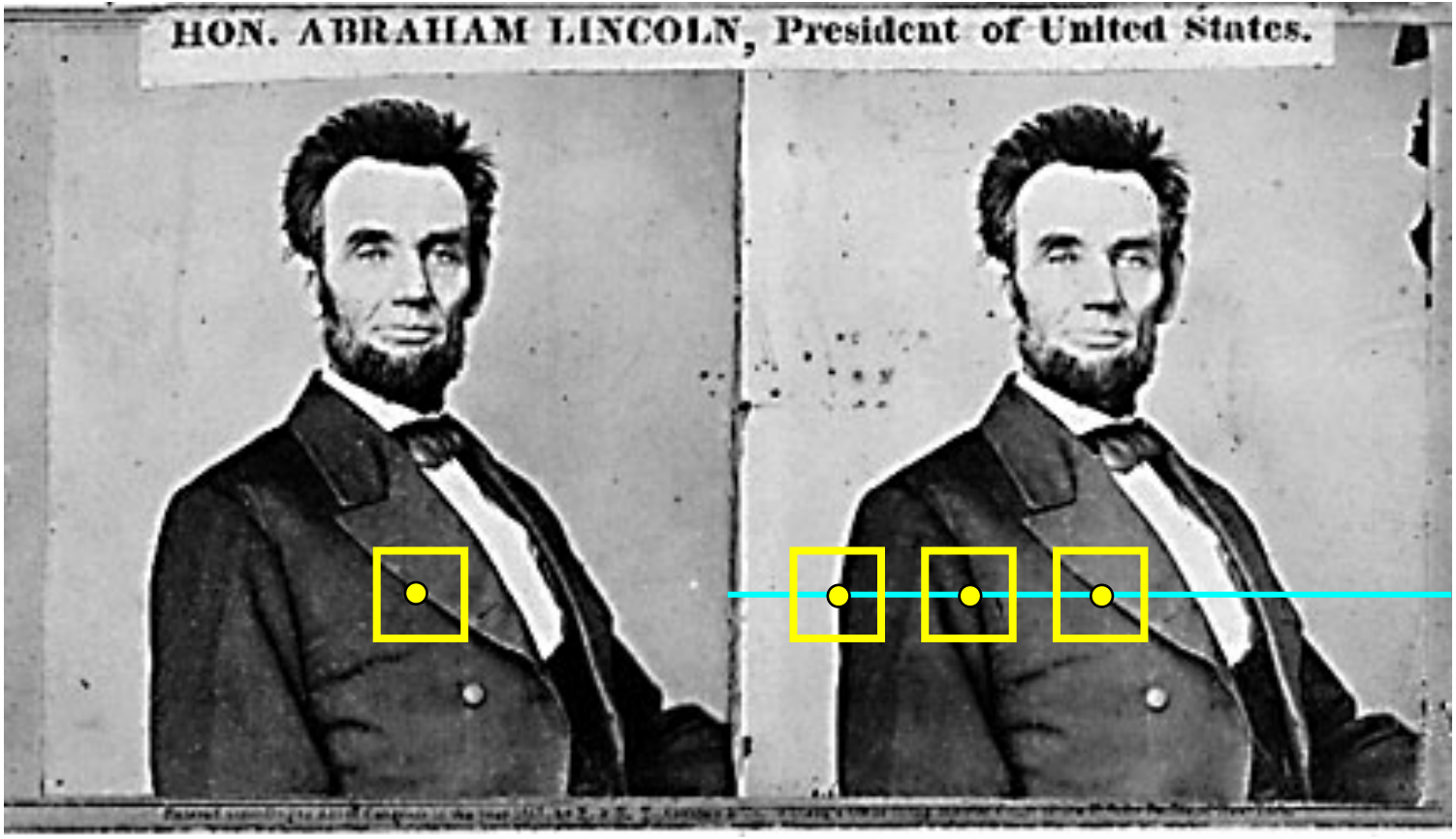
Geometry for a simple stereo system

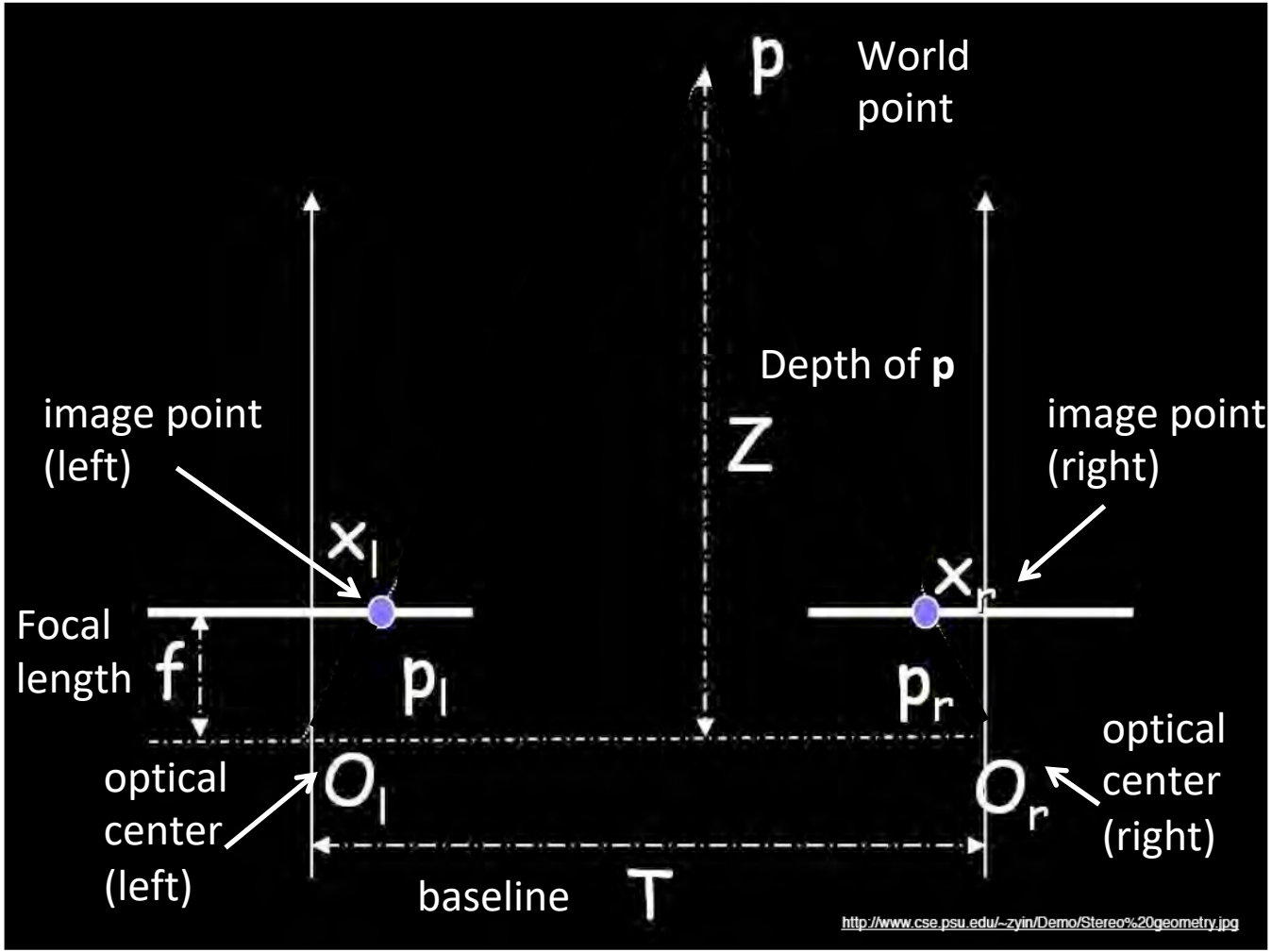
- First, assuming parallel optical axes, known camera parameters (i.e., calibrated cameras):

Simplest Case: Parallel images



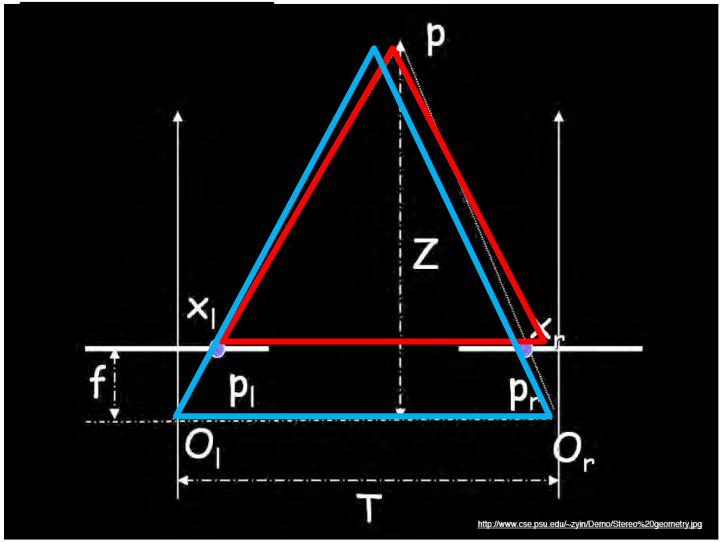
- Image planes of cameras are parallel to each other and to the baseline
- Camera centers are at same height
- Focal lengths are the same
- Then epipolar lines fall along the horizontal scan lines of the images


HON. ABRAHAM LINCOLN, President of United States.


p
World point
Depth of p
Z
image point (left)
image point (right)
x_l
x_r
Focal length
f
p_l
p_r
optical center (left)
O_l
O_r
optical center (right)
baseline
T
http://www.cse.psu.edu/~zyin/Demo/Stereo%20geometry.jpg

Geometry for a simple stereo system

- Assume parallel optical axes, known camera parameters (i.e., calibrated cameras). **What is expression for Z?**



Similar triangles (p_l, P, p_r) and (O_l, P, O_r):

$$\frac{T + x_l - x_r}{Z - f} = \frac{T}{Z}$$

$$Z = f\frac{T}{x_r - x_l}$$

disparity

Depth from disparity

image I(x,y)

Disparity map D(x,y)

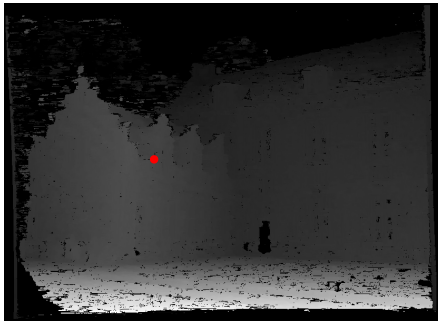
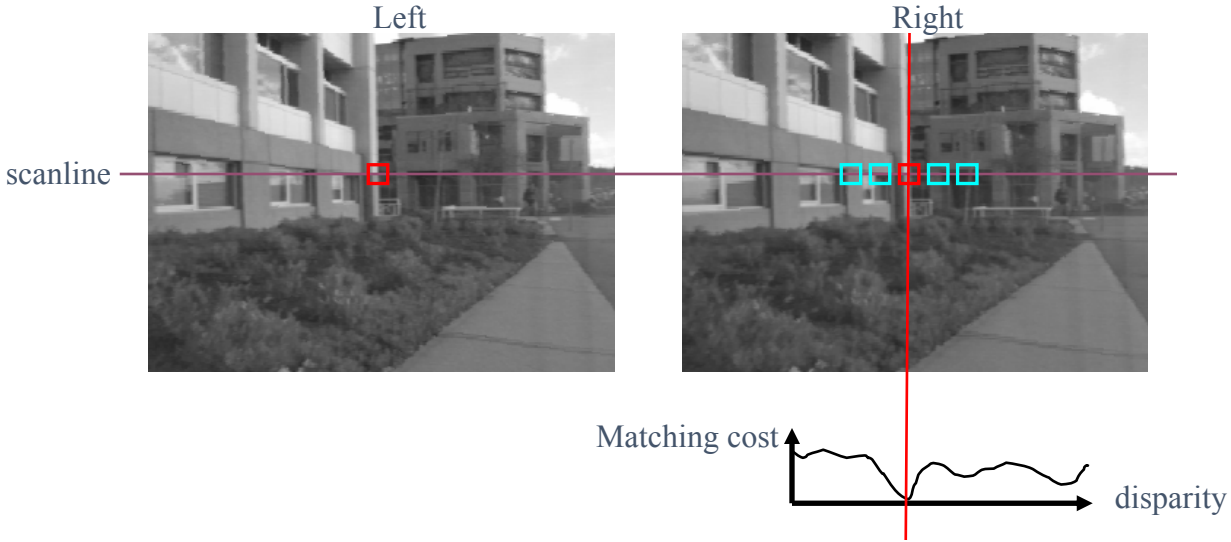


image I´(x´,y´)

$(x´,y´)=(x+D(x,y), y)$

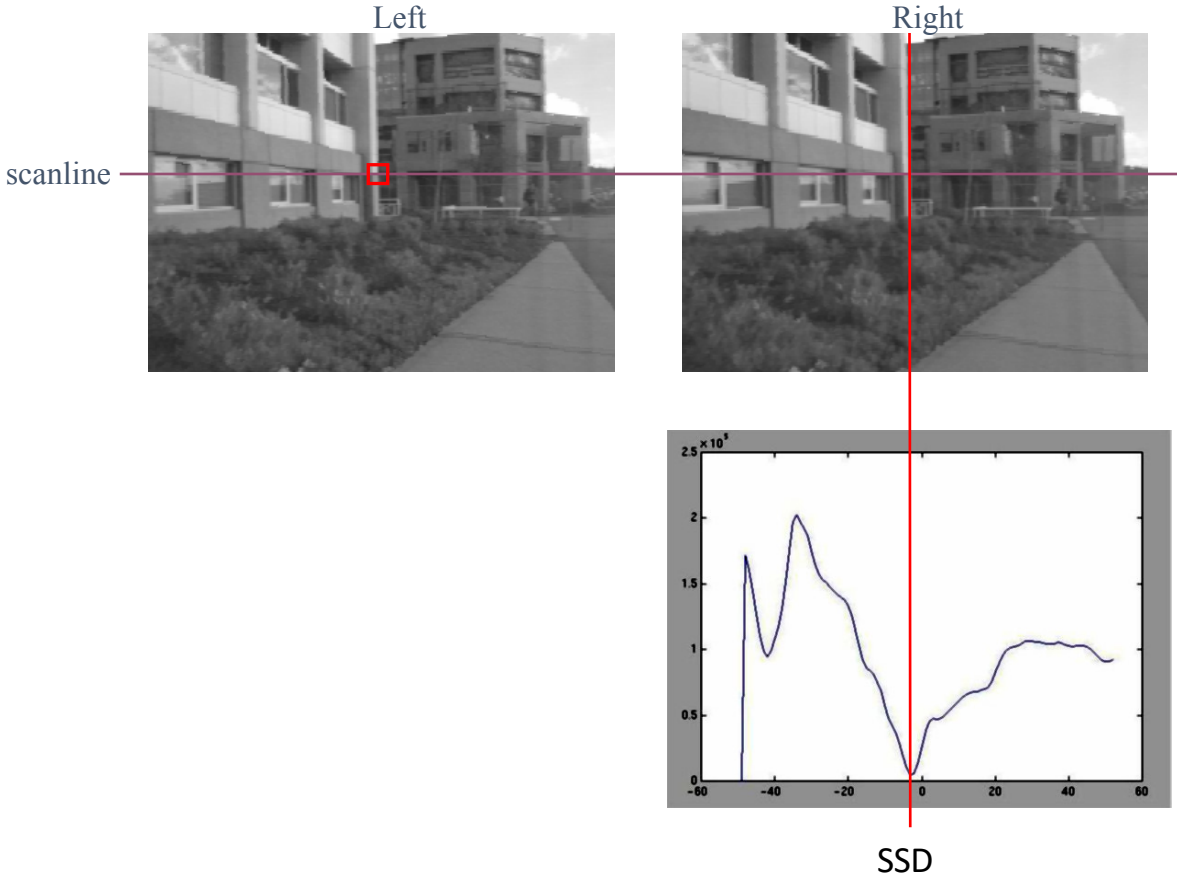
So if we could find the **corresponding points** in two images, we could **estimate relative depth**…

Correspondence search

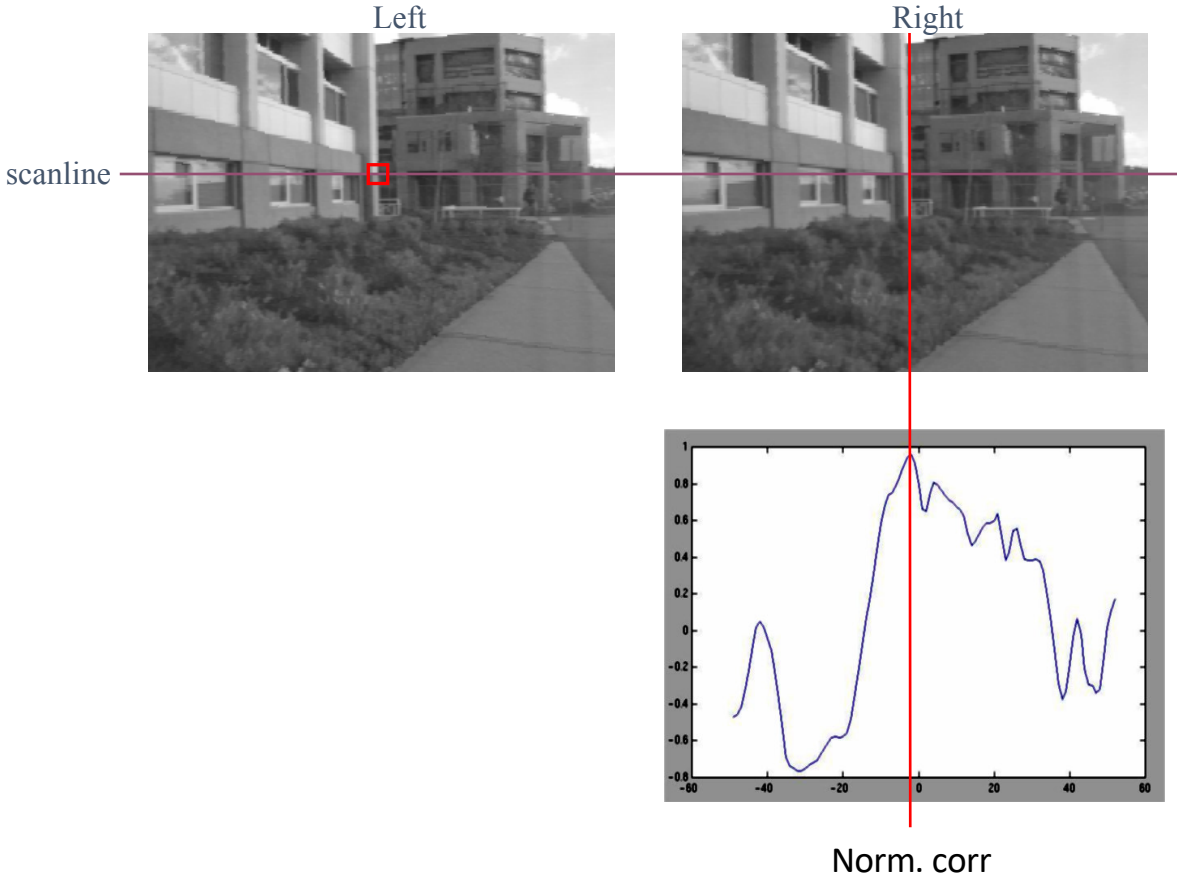


- Slide a window along the right scanline and compare contents of that window with the reference window in the left image
- Matching cost: SSD or normalized correlation

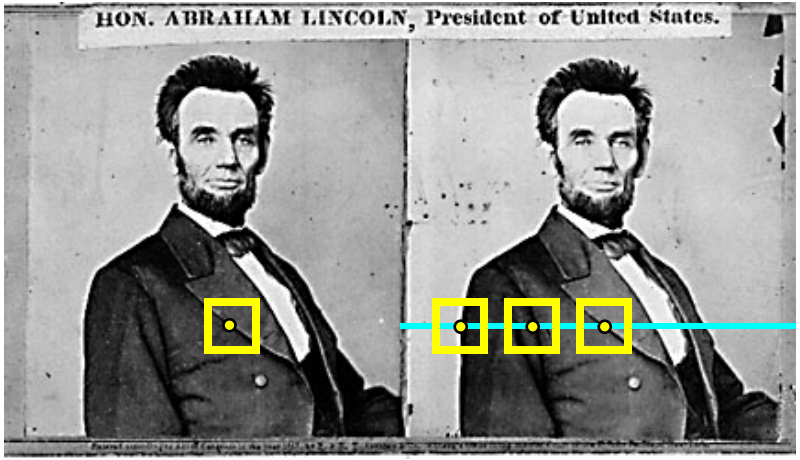
Correspondence search



Correspondence search

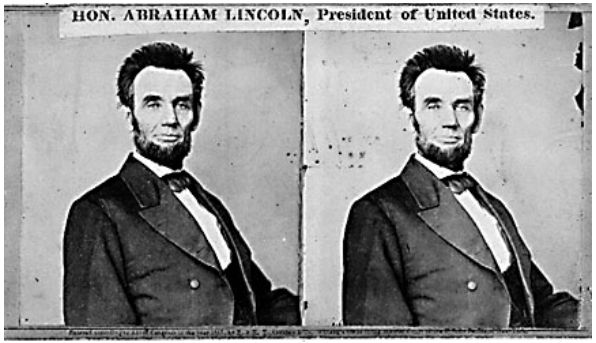


Basic stereo matching algorithm



- If necessary, rectify the two stereo images to transform epipolar lines into scanlines
- For each pixel *x* in the first image
  - Find corresponding epipolar scanline in the right image
  - Examine all pixels on the scanline and pick the best match *x'*
  - Compute disparity *x–x'* and set depth(*x*) = *B*f*/(*x–x'*)

Failures of correspondence search



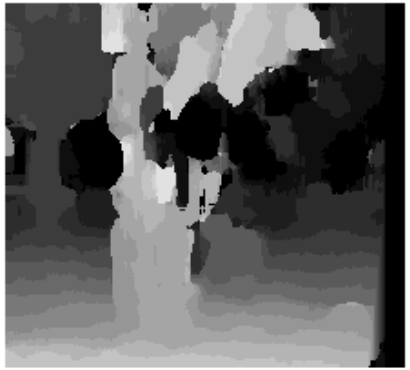
Textureless surfaces

Occlusions, repetition

Non-Lambertian surfaces, specularities

Effect of window size

W = 3



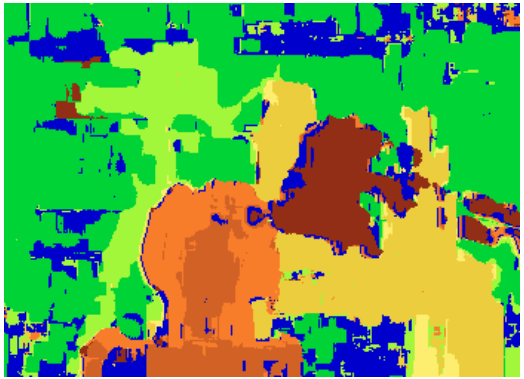
W = 20

- Smaller window
  - \+ More detail
  - More noise
- Larger window
  - \+ Smoother disparity maps
  - Less detail

Results with window search

Data

Window-based matching



Ground truth

Better methods exist...

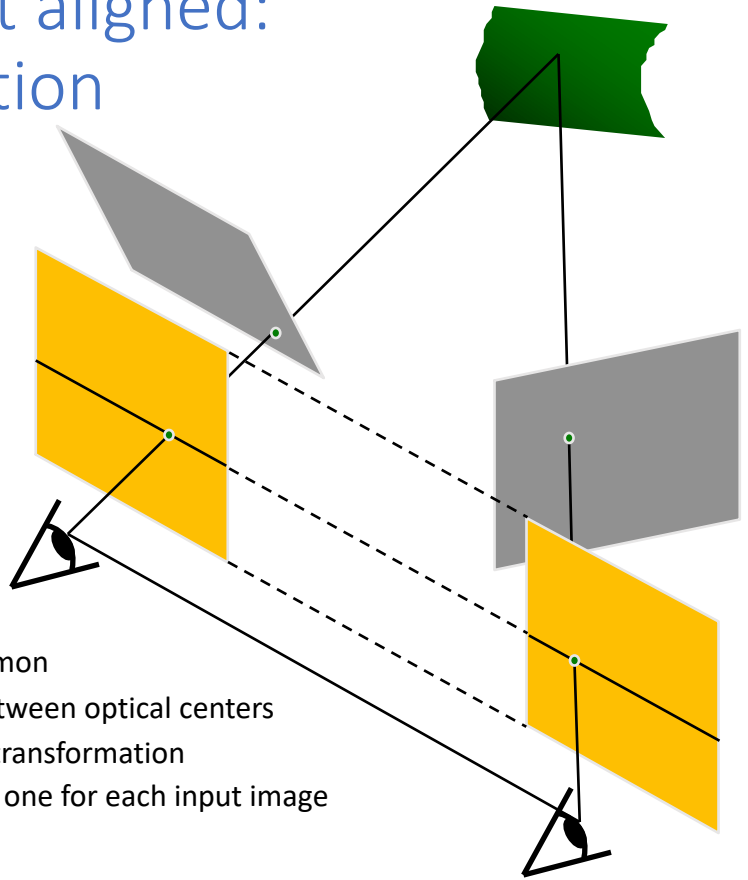
Graph cuts

Ground truth

Y. Boykov, O. Veksler, and R. Zabih, Fast Approximate Energy Minimization via Graph Cuts, PAMI 2001

For the latest and greatest: http://www.middlebury.edu/stereo/

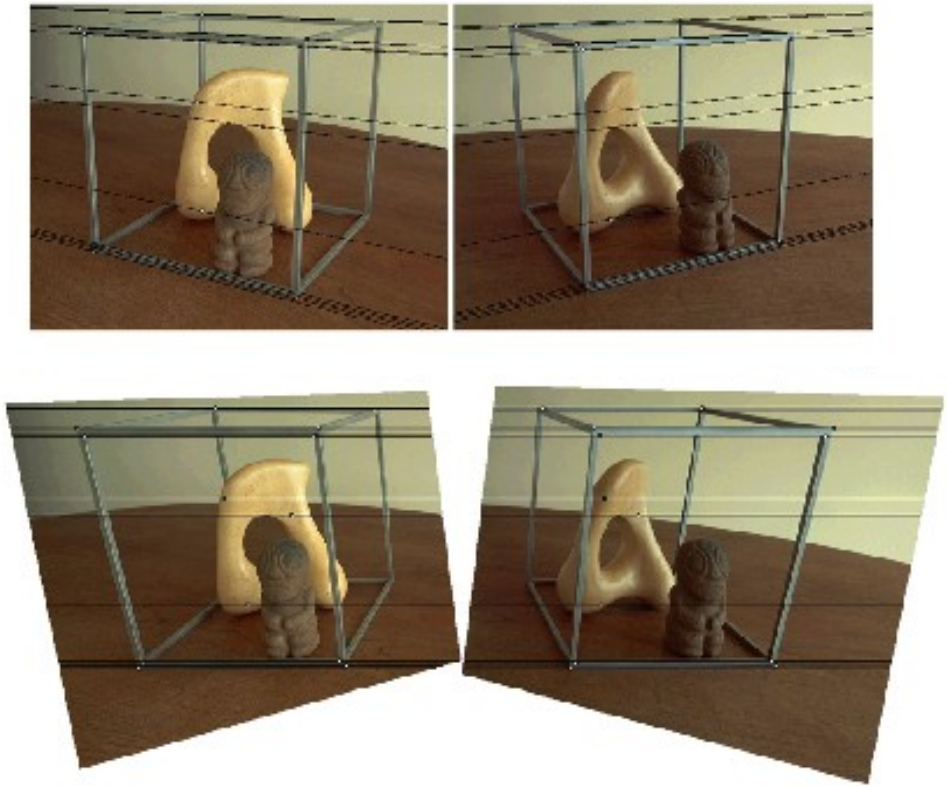
When cameras are not aligned: Stereo image rectification



- 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.

Rectification example



Questions?