CS4501: Introduction to Computer Vision

Dense Stereo and Epipolar Geometry

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).

Last Class

- Camera Calibration
- Stereo Vision

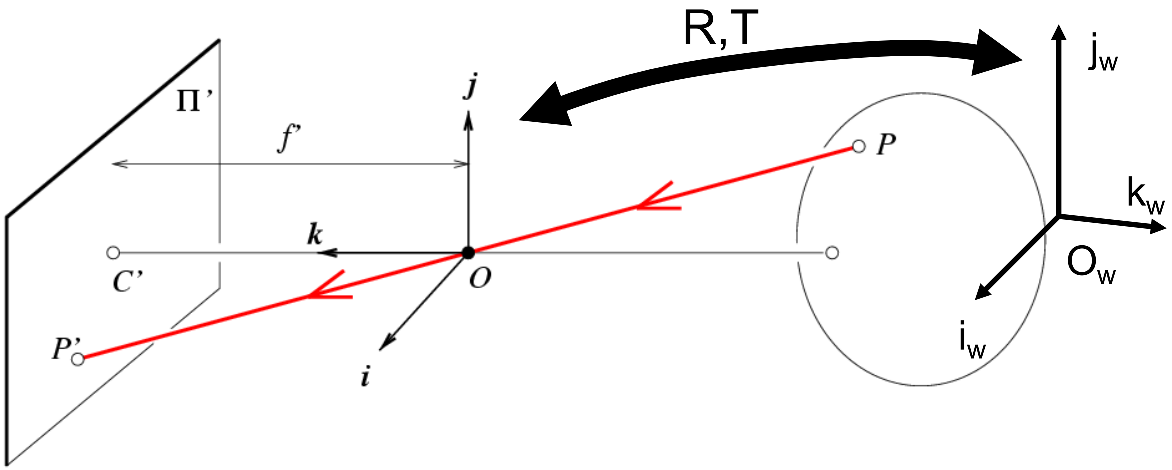
Today's Class

- Stereo Vision – Dense Stereo
- More on Epipolar Geometry

Camera Calibration

- What does it mean?

Recall the Projection matrix



$$\mathbf{x} = \mathbf{K}[\mathbf{R} \quad \mathbf{t}]\mathbf{X}$$

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.
```

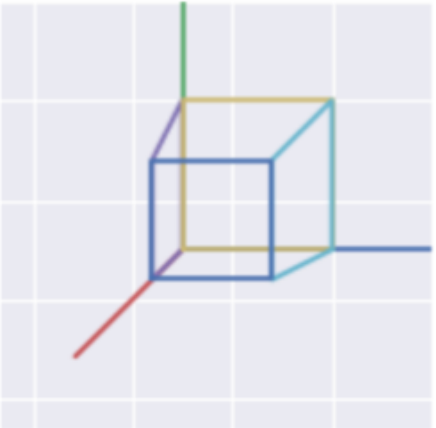
$\mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix} =$

```
# 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 $\mathbf{x}$



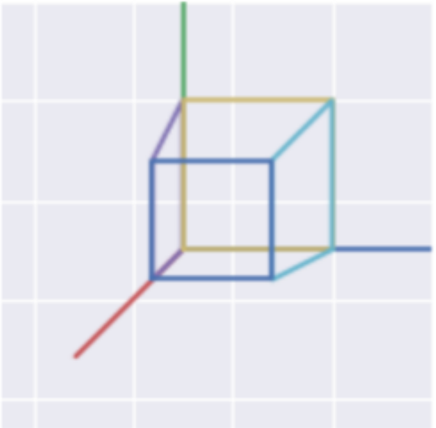
Camera Calibration

$$\mathbf{x} = \mathbf{K}[\mathbf{R} \quad \mathbf{t}]\mathbf{X}$$

$\mathbf{X} =$

```
# 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

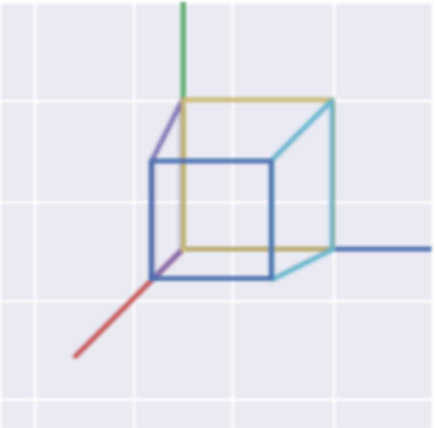
$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

$\mathbf{X} =$

```
# 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 $\mathbf{K}[\mathbf{R} \;\; \mathbf{t}]$

$\mathbf{X} =$

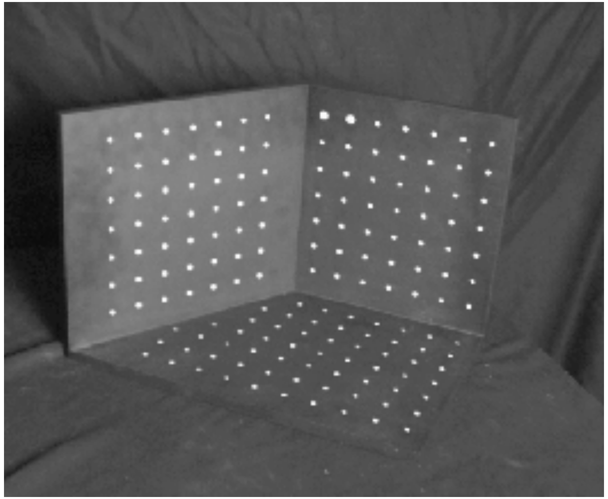


Slide Credit: James Hays

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

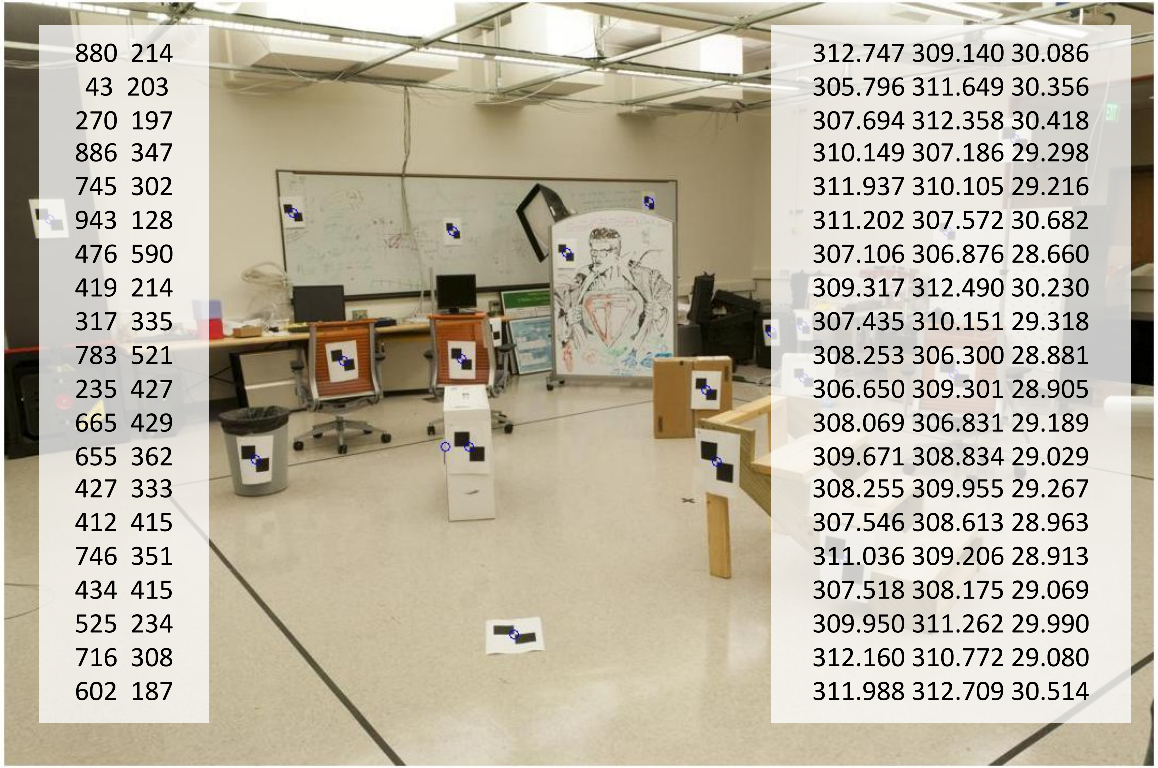
Known 3d locations

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

Unknown Camera Parameters

How do we calibrate a camera?

Slide Credit: James Hays



Known 2d image coords		Known 3d locations		
880	214	312.747	309.140	30.086
43	203	305.796	311.649	30.356
270	197	307.694	312.358	30.418
886	347	310.149	307.186	29.298
745	302	311.937	310.105	29.216
943	128	311.202	307.572	30.682
476	590	307.106	306.876	28.660
419	214	309.317	312.490	30.230
317	335	307.435	310.151	29.318
783	521	308.253	306.300	28.881
235	427	306.650	309.301	28.905
665	429	308.069	306.831	29.189
655	362	309.671	308.834	29.029
427	333	308.255	309.955	29.267
412	415	307.546	308.613	28.963
746	351	311.036	309.206	28.913
434	415	307.518	308.175	29.069
525	234	309.950	311.262	29.990
716	308	312.160	310.772	29.080
602	187	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]?

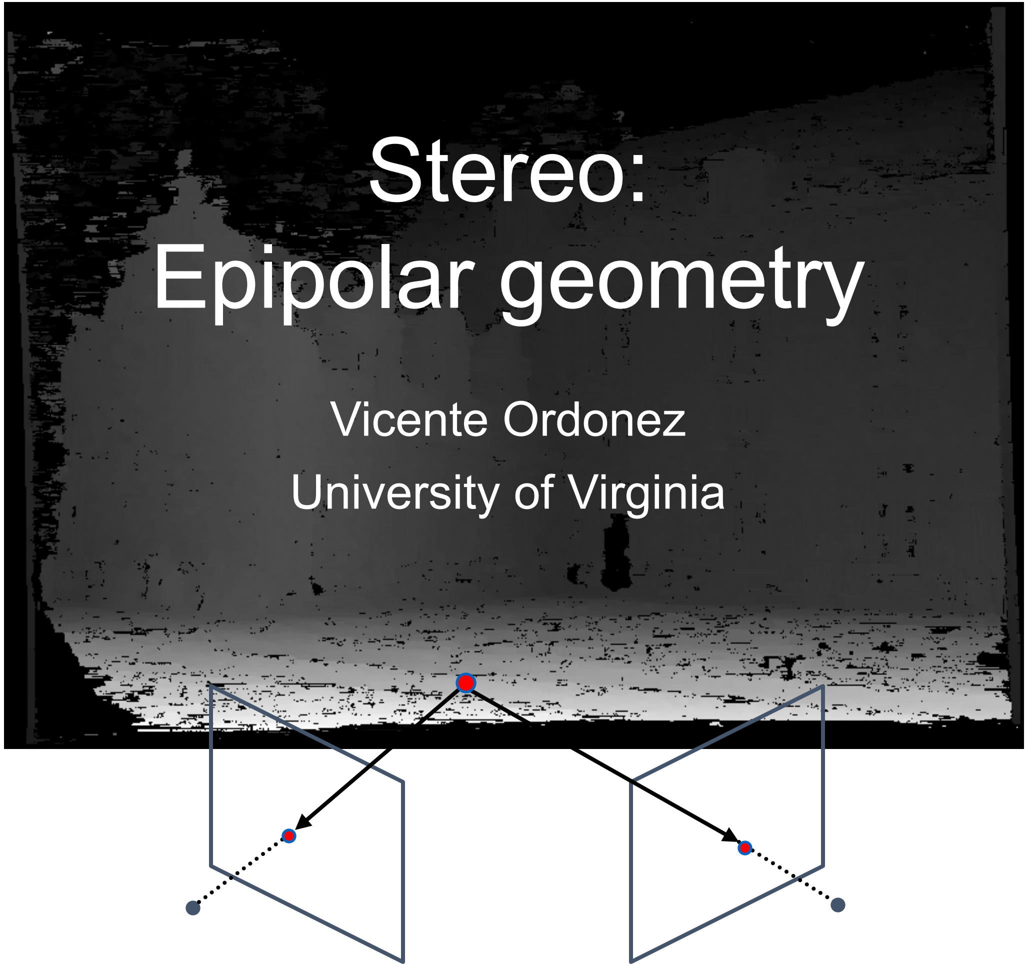
$$\begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} = \mathbf{K}[\mathbf{R} \;\; \mathbf{t}]$$

- 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

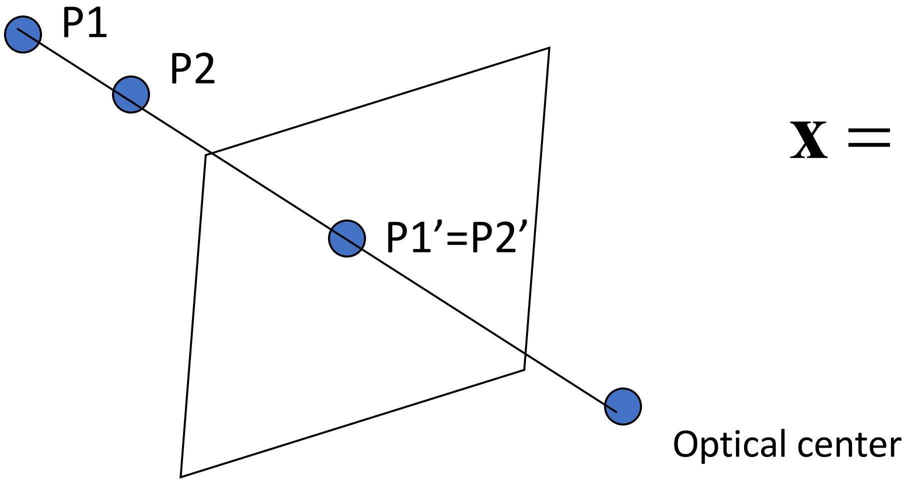
Vicente Ordonez
University of Virginia



Slides by Kristen Grauman

Why multiple views?

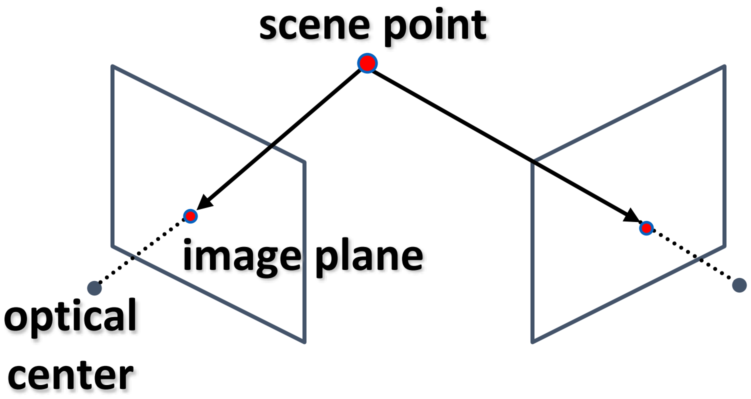
- Structure and depth are inherently ambiguous from single views.

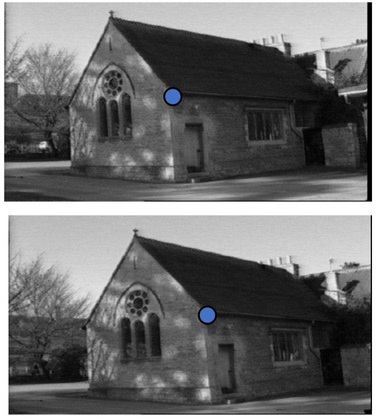


$$\mathbf{x} = \mathbf{K}[\mathbf{R} \quad \mathbf{t}]\mathbf{X}$$

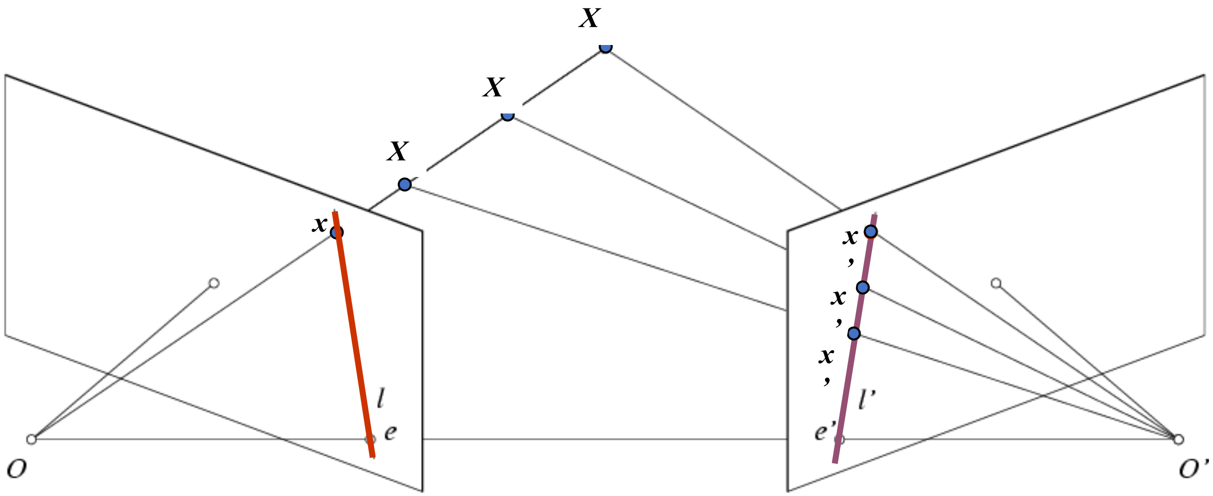
Estimating depth with stereo

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





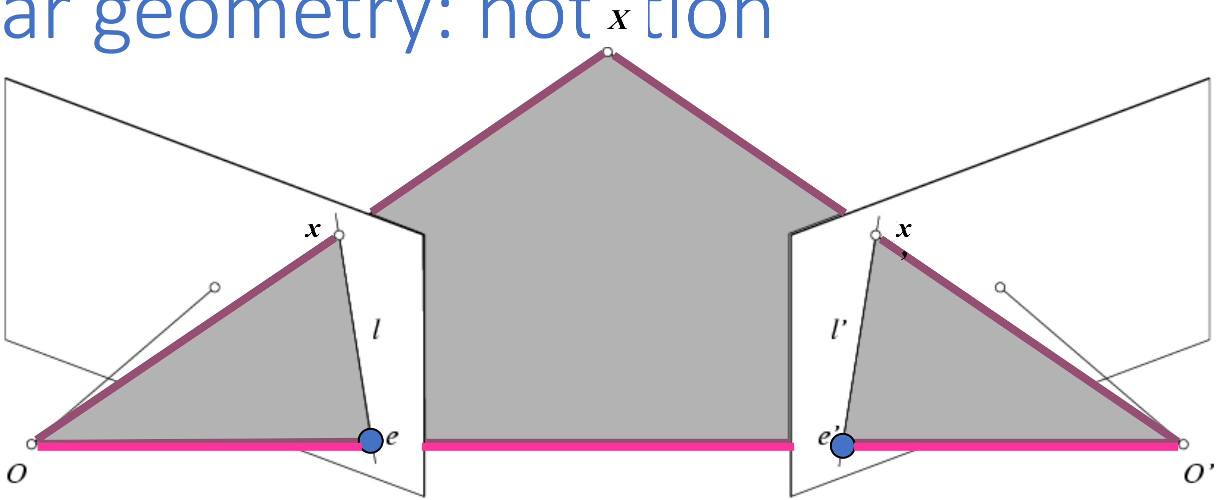
Key idea: Epipolar constraint



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

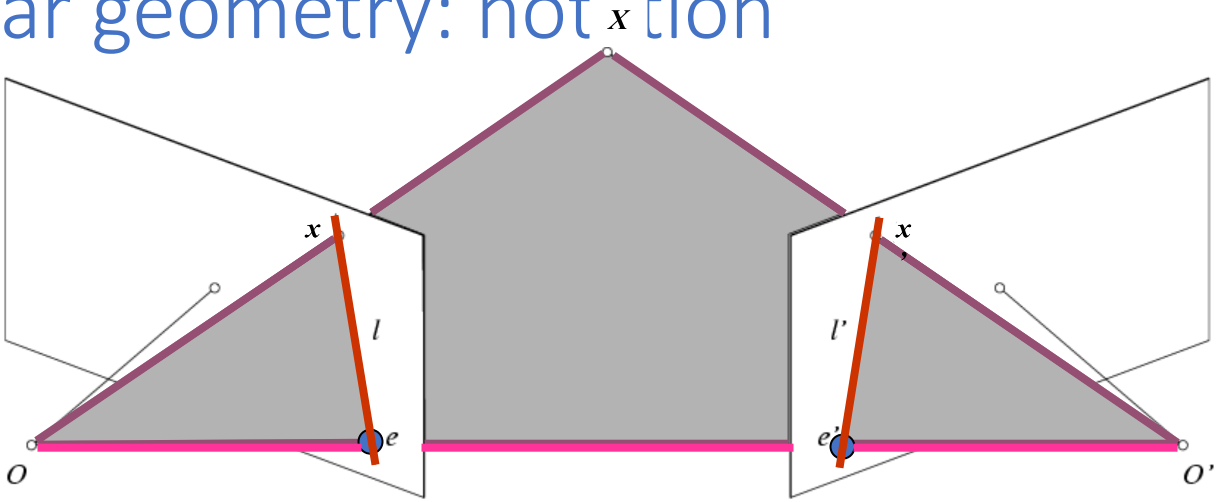


- **Baseline** – line connecting the two camera centers
- **Epipoles**

= intersections of baseline with image planes
= projections of the other camera center

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

Epipolar geometry: notation



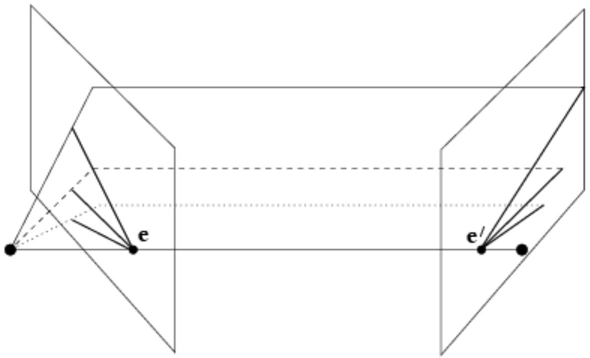
- **Baseline** – line connecting the two camera centers
- **Epipoles**

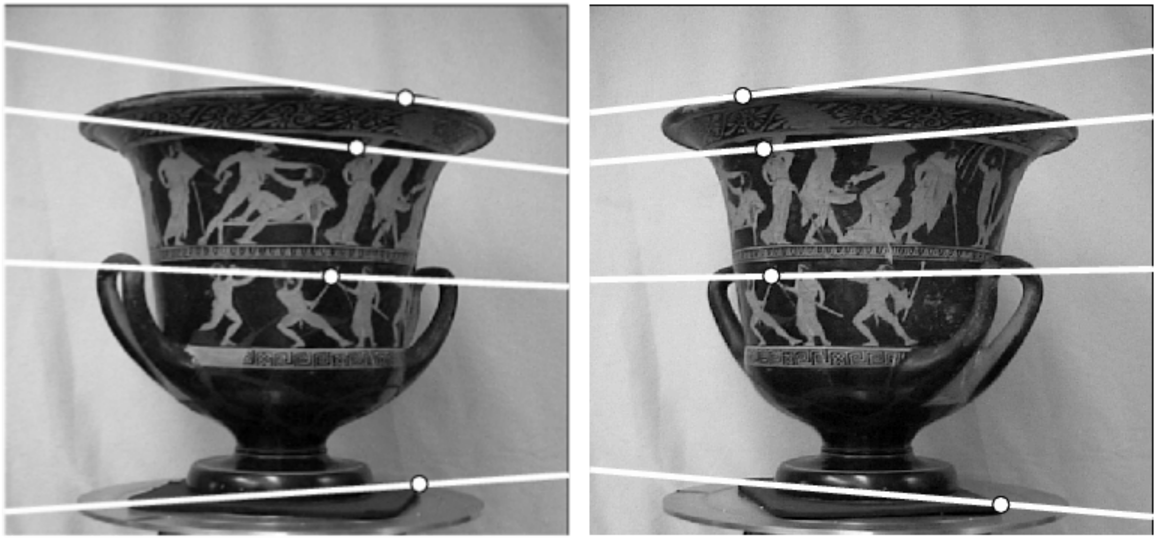
= intersections of baseline with image planes

= 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

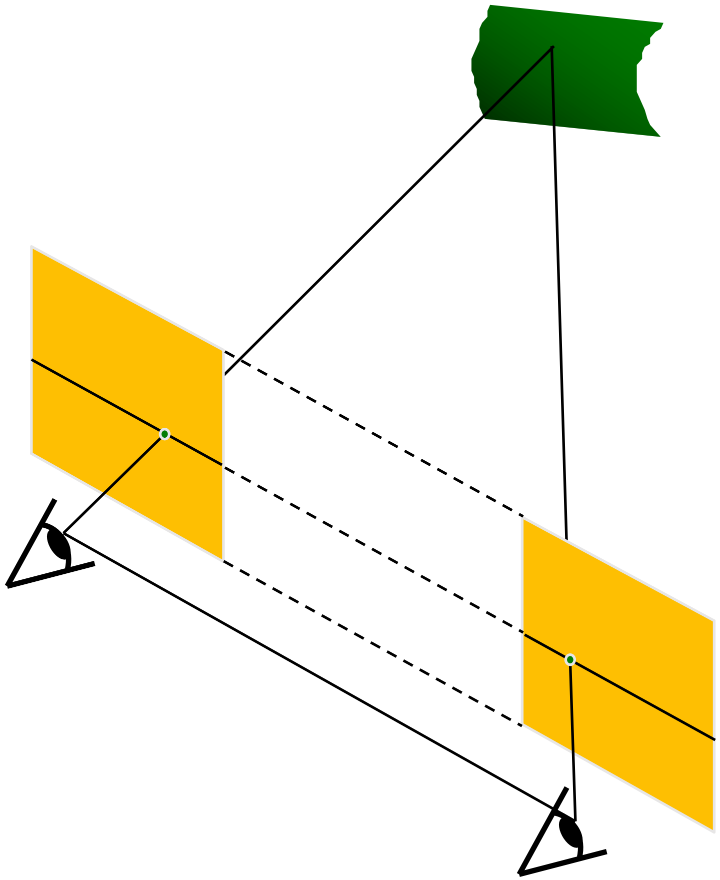




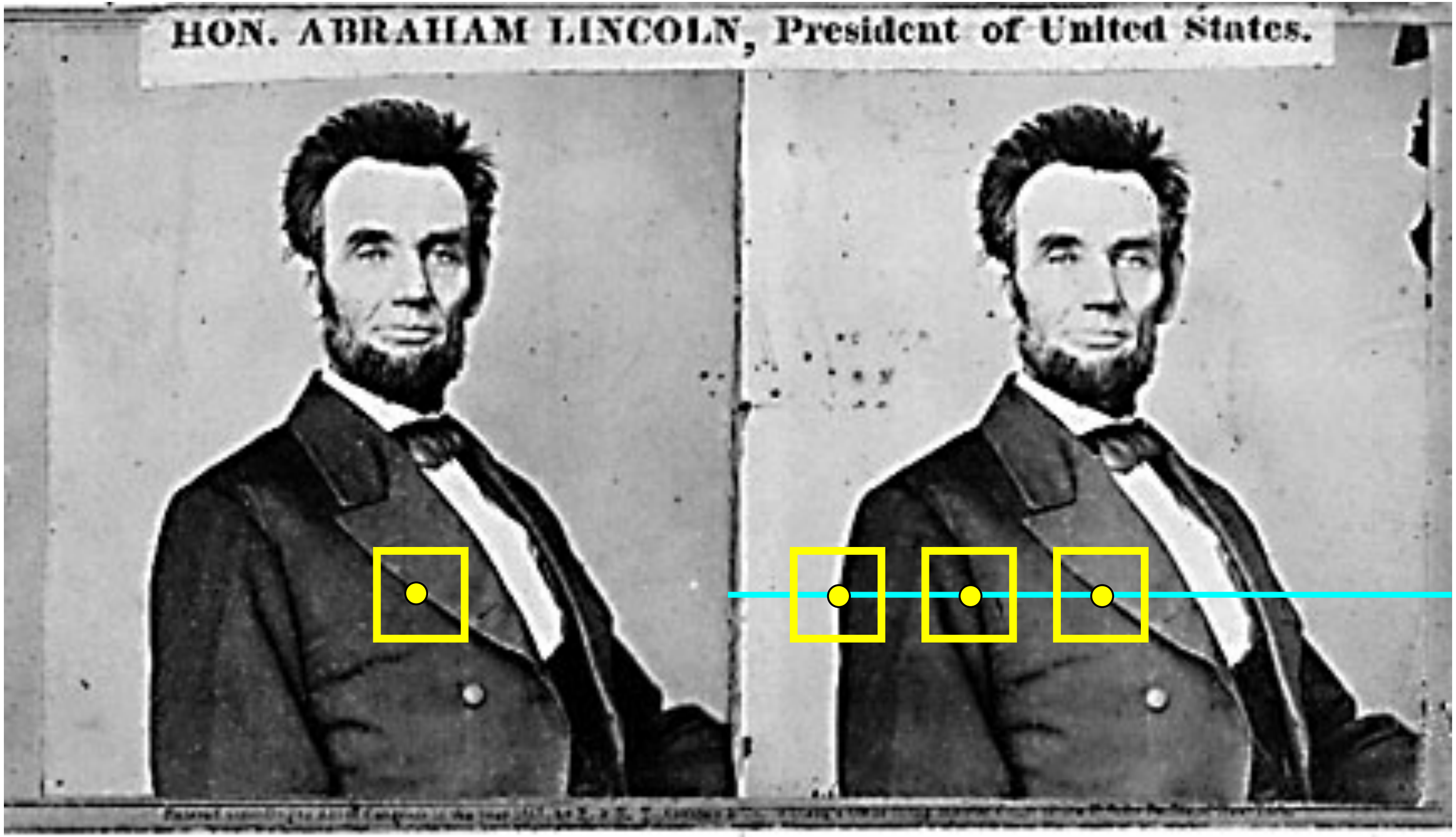
Geometry for a simple stereo system

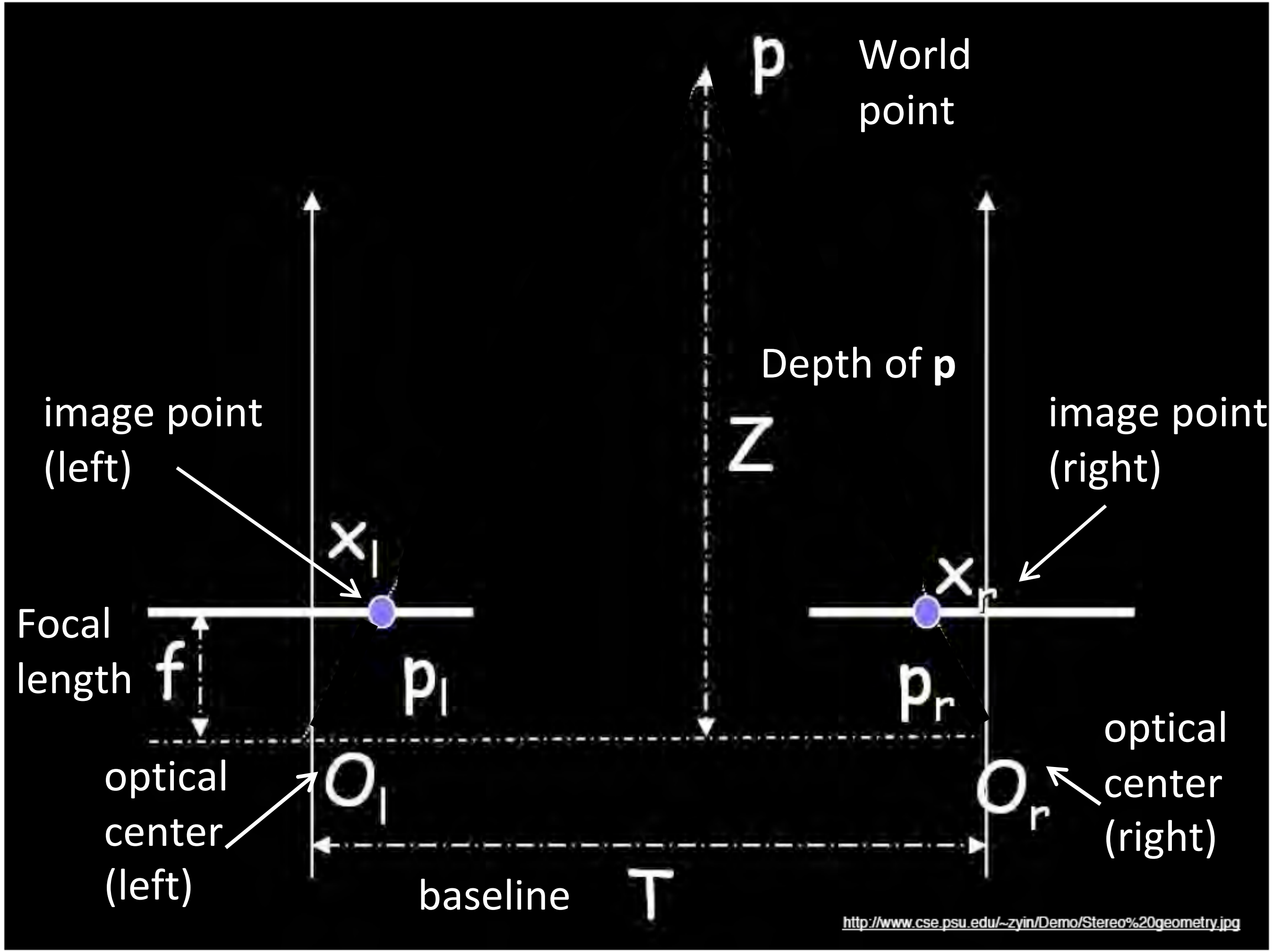
- First, assuming parallel optical axes, known camera parameters (i.e., calibrated cameras):

Simplest Case: Parallel images



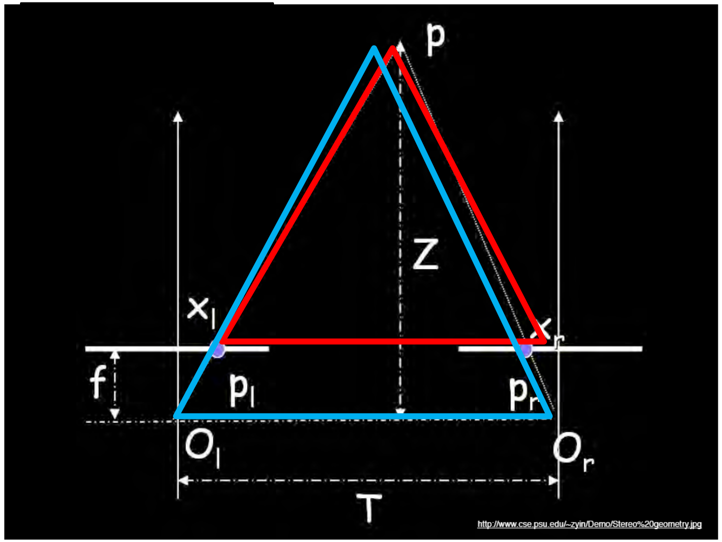
- Image planes of cameras are parallel to each other and to the baseline
- Camera centers are at same height
- Focal lengths are the same
- Then epipolar lines fall along the horizontal scan lines of the images


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 → $x_r - x_l$

Depth from disparity

image I(x,y)

Disparity map D(x,y)

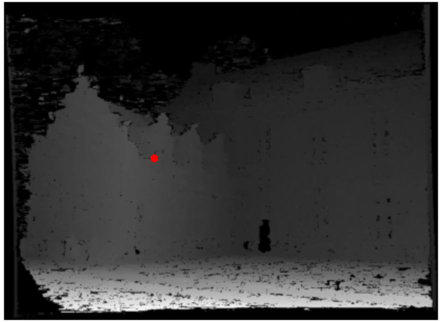
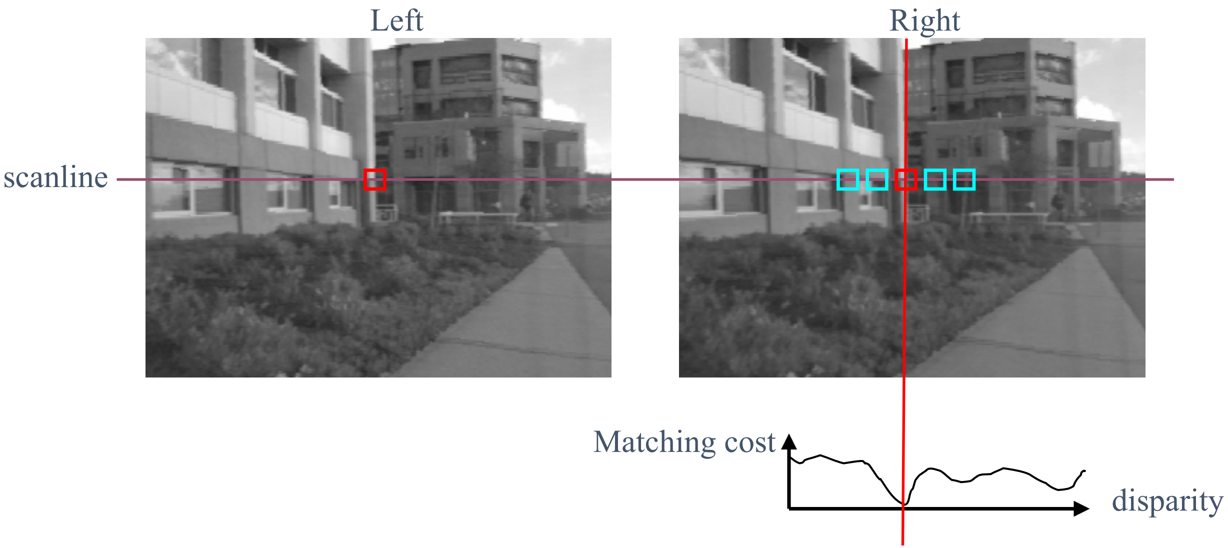


image I´(x´,y´)

$$(x',y')=(x+D(x,y),\ y)$$

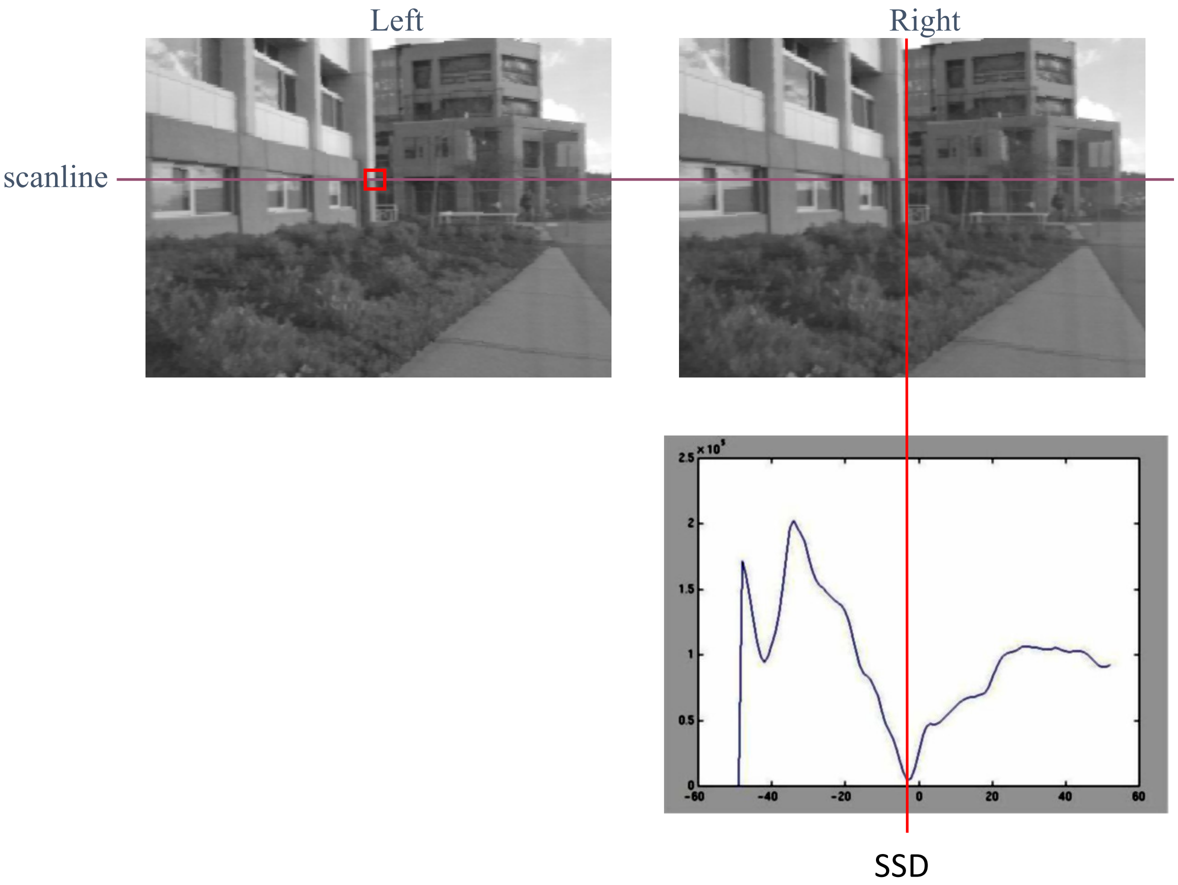
So if we could find the **corresponding points** in two images, we could **estimate relative depth**…

Correspondence search

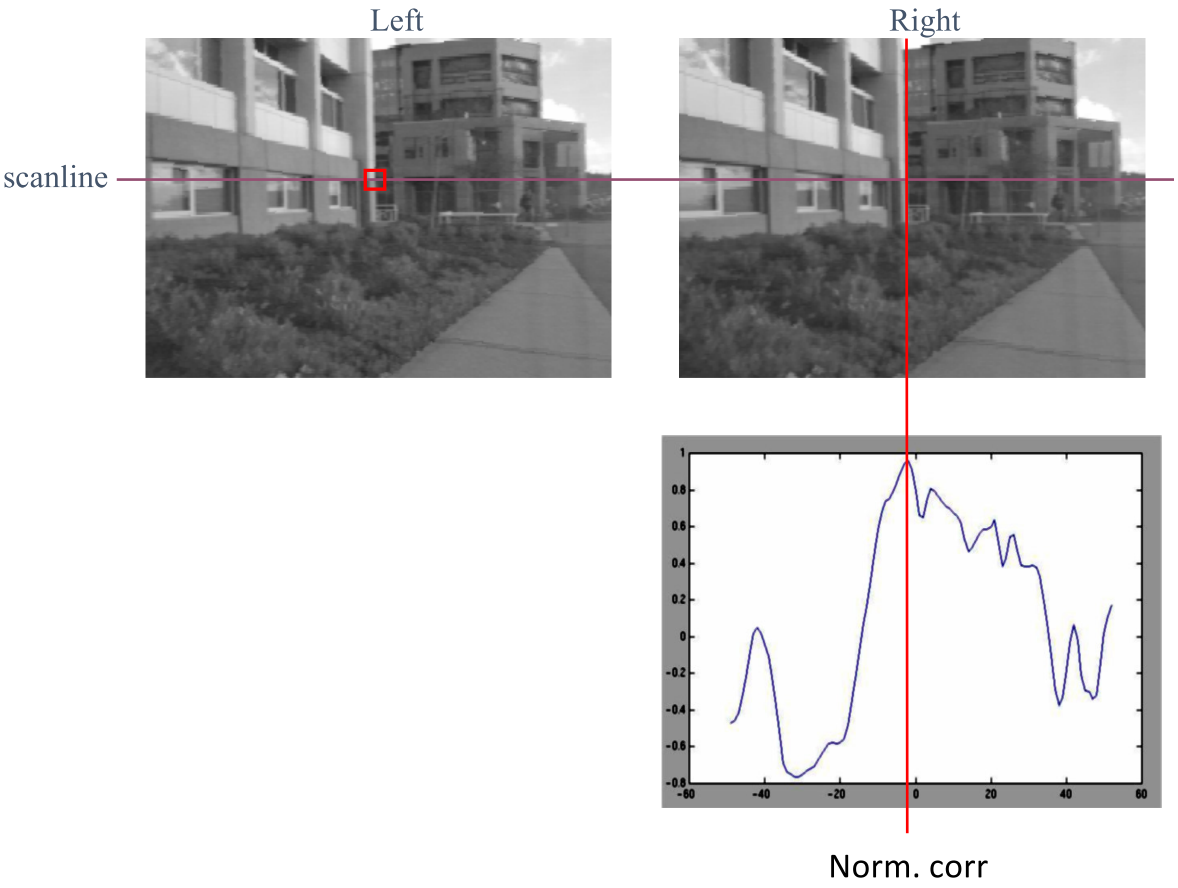


- Slide a window along the right scanline and compare contents of that window with the reference window in the left image
- Matching cost: SSD or normalized correlation

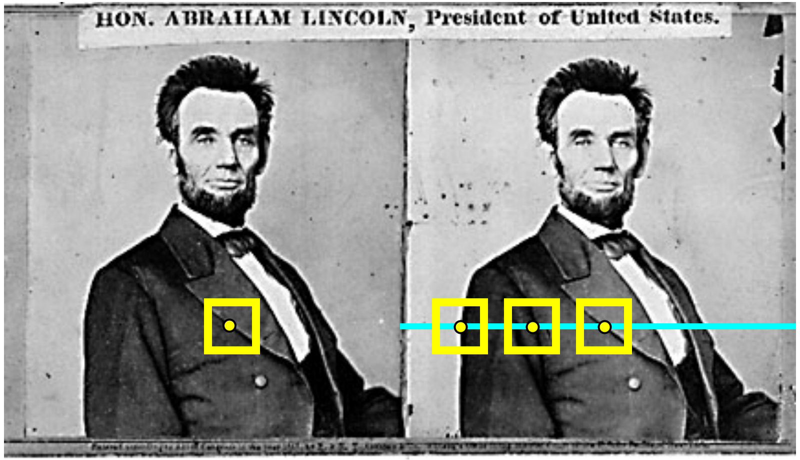
Correspondence search



Correspondence search

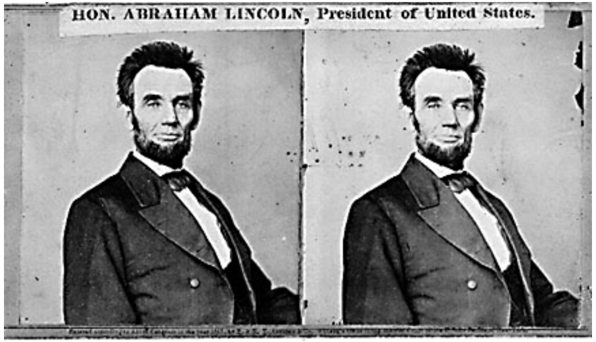


Basic stereo matching algorithm



- 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 $\text{depth}(x) = B*f/(x-x')$

Failures of correspondence search



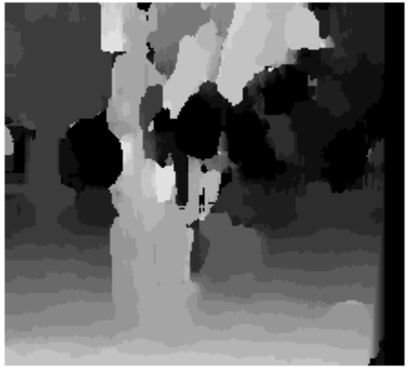
Textureless surfaces

Occlusions, repetition

Non-Lambertian surfaces, specularities

Effect of window size

W = 3



W = 20

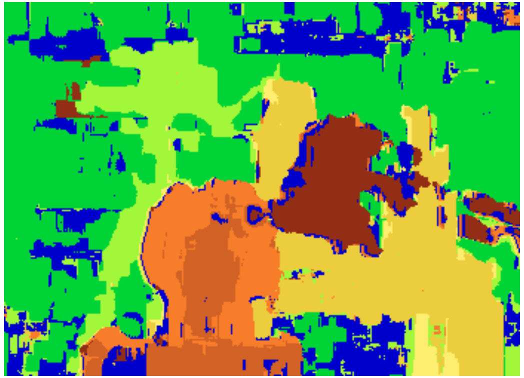
- Smaller window
  - \+ More detail
  - • More noise

- Larger window
  - \+ Smoother disparity maps
  - • Less detail

Results with window search

Data

Window-based matching



Ground truth

Better methods exist...

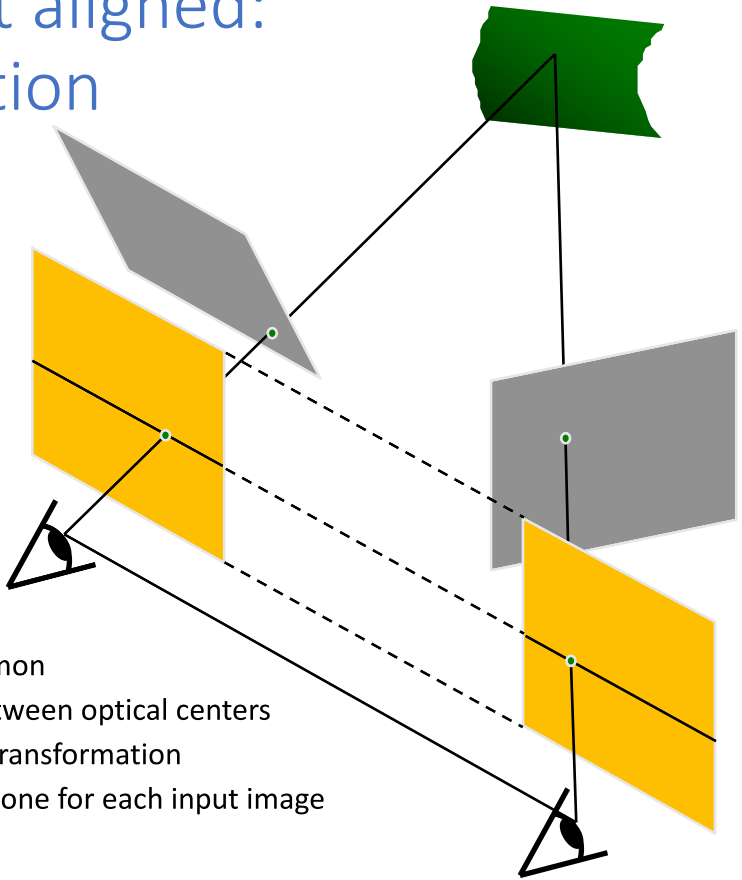
Graph cuts

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/

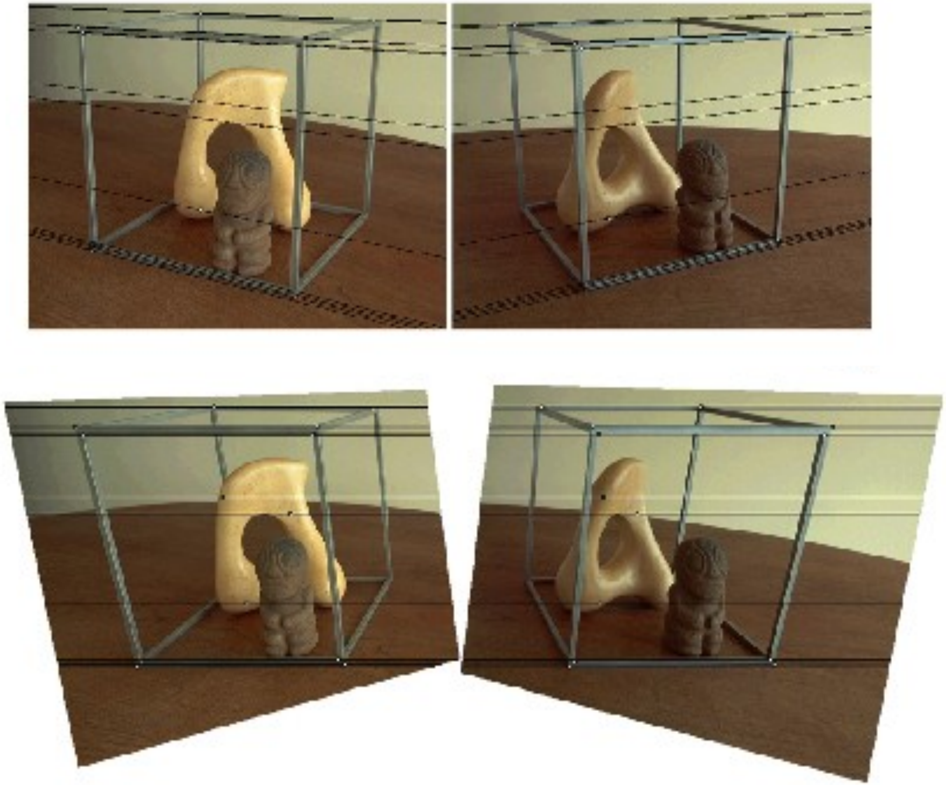
When cameras are not aligned: Stereo image rectification



- 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?