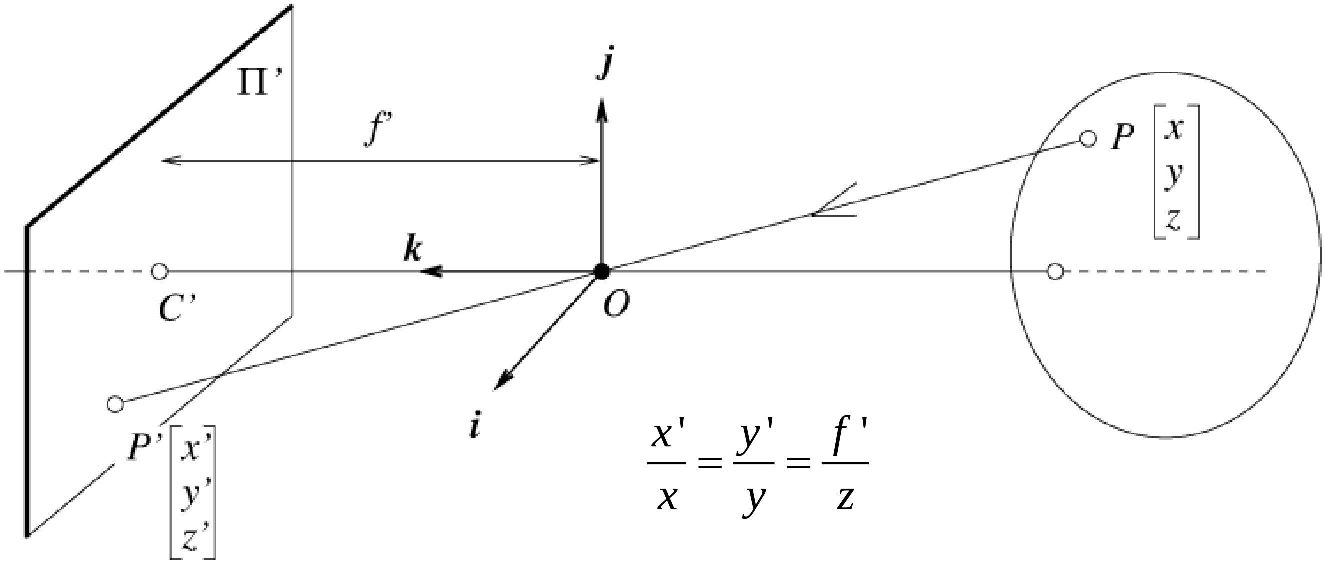
Depth Estimation from Stereo

Jianping Fan
Dept of Computer Science
UNC-Charlotte

Course Website:
http://webpages.uncc.edu/jfan/itcs5152.html

Review: **Perspective Projection**

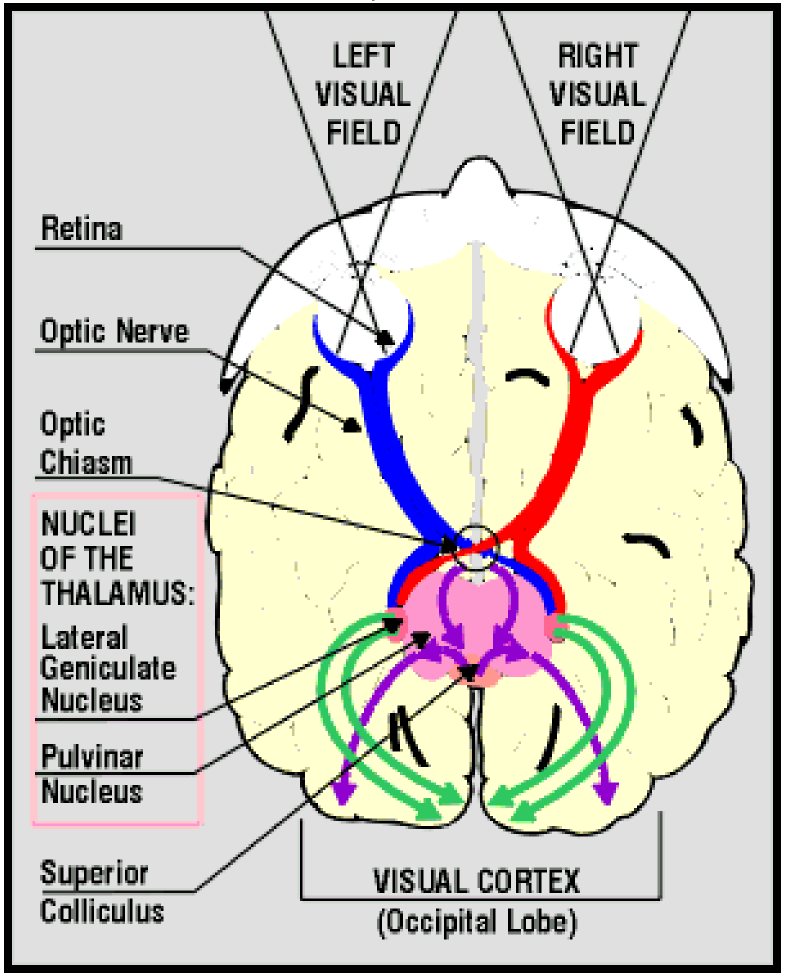


$$\frac{x'}{x} = \frac{y'}{y} = \frac{f'}{z}$$

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

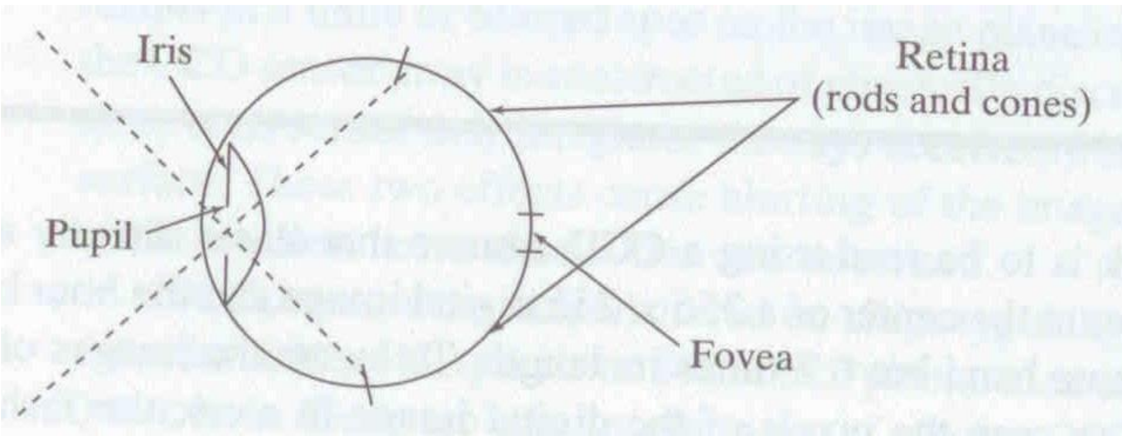
$$y' = f' \frac{y}{z}$$

Human visual pathway



Human eye

Rough analogy with human visual system:



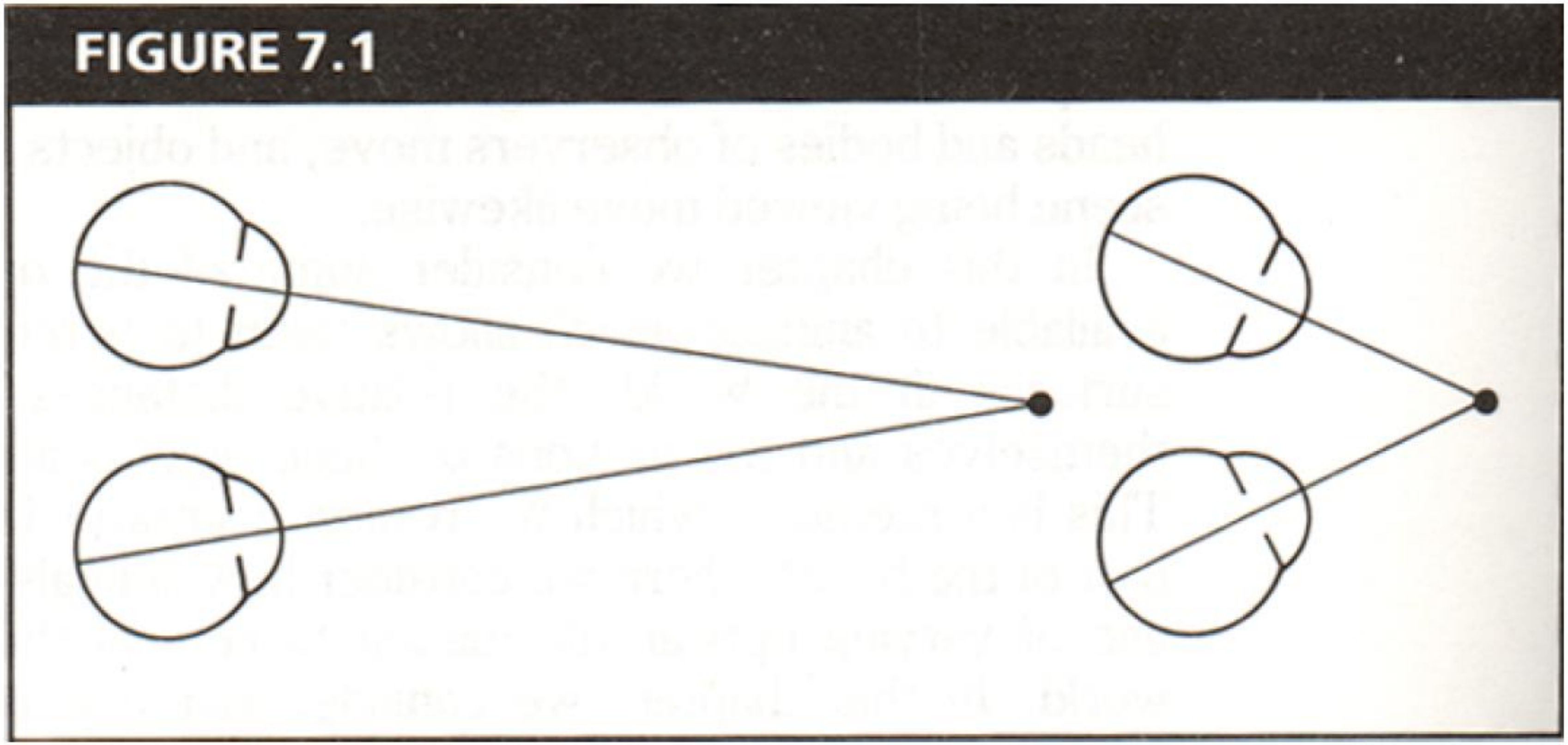
Pupil/Iris – control amount of light passing through lens

Retina - contains sensor cells, where image is formed

Fovea – highest concentration of cones

Fig from Shapiro and Stockman

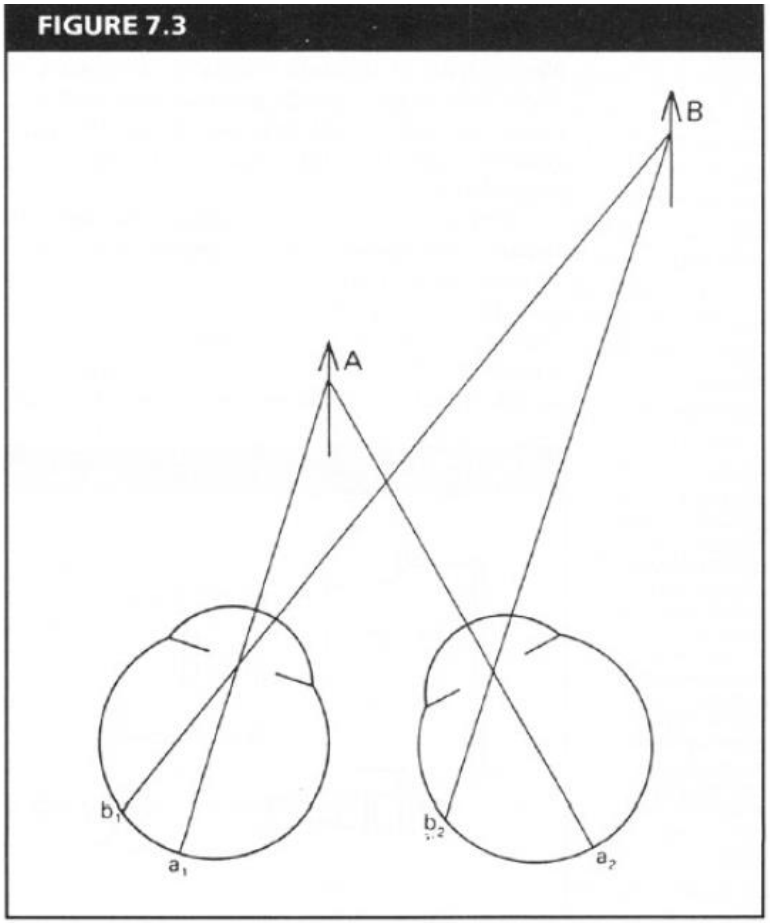
Human stereopsis: disparity



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

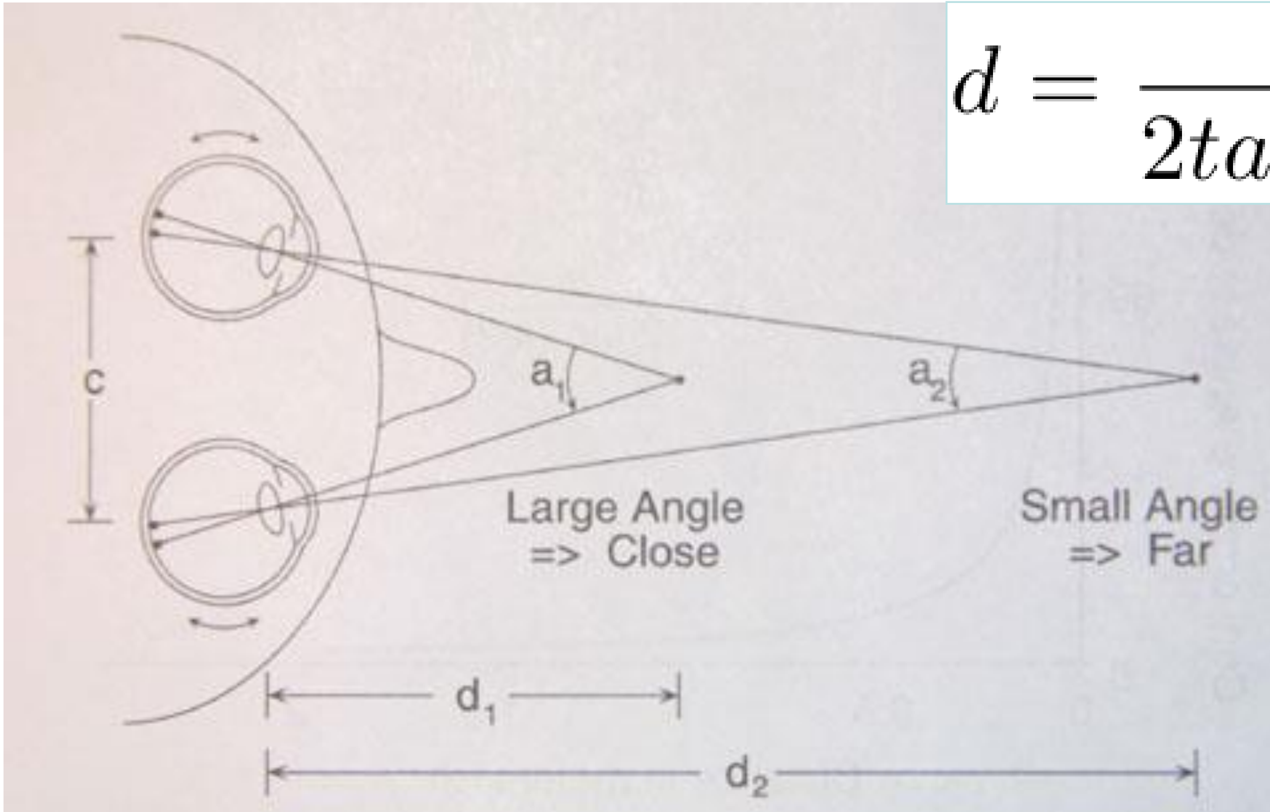


From Bruce and Green, Visual Perception, Physiology, Psychology and Ecology

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

Adapted from David Forsyth, UC Berkeley

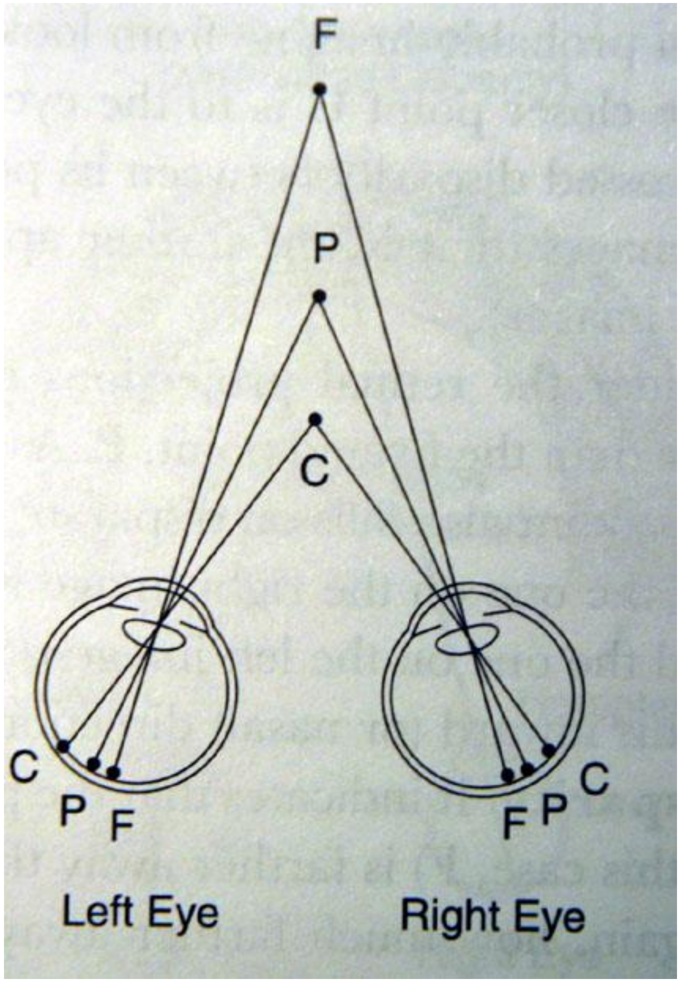
Depth from Convergence



$$d = \frac{c}{2tan(a/2)}$$

Human performance: up to 6-8 feet

Depth from binocular disparity



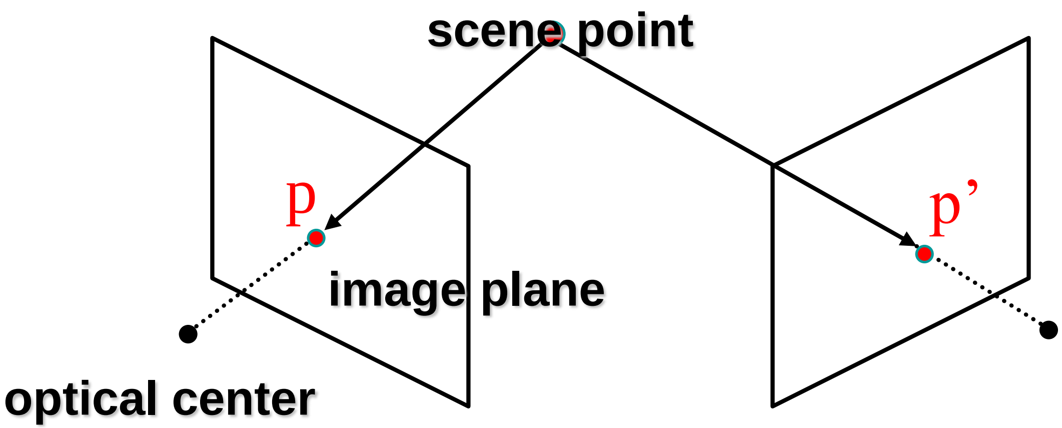
Sign and magnitude of disparity

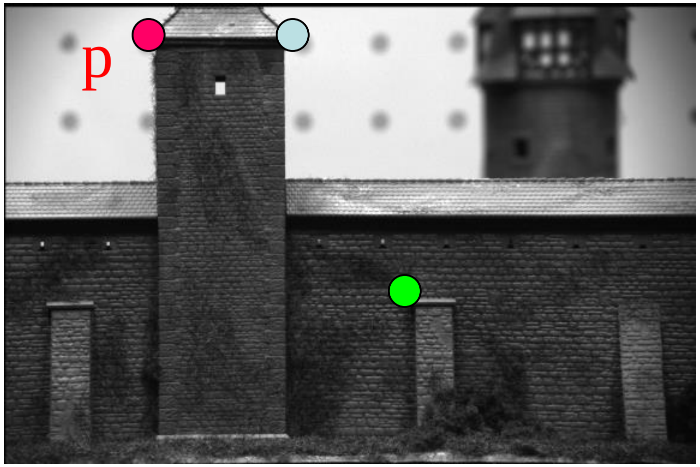
P: converging point

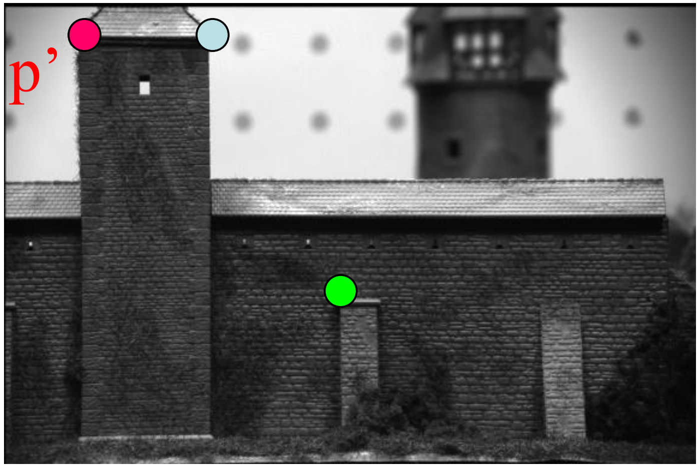
C: object nearer projects to the outside of the P, disparity = +

F: object farther projects to the inside of the P, disparity = -

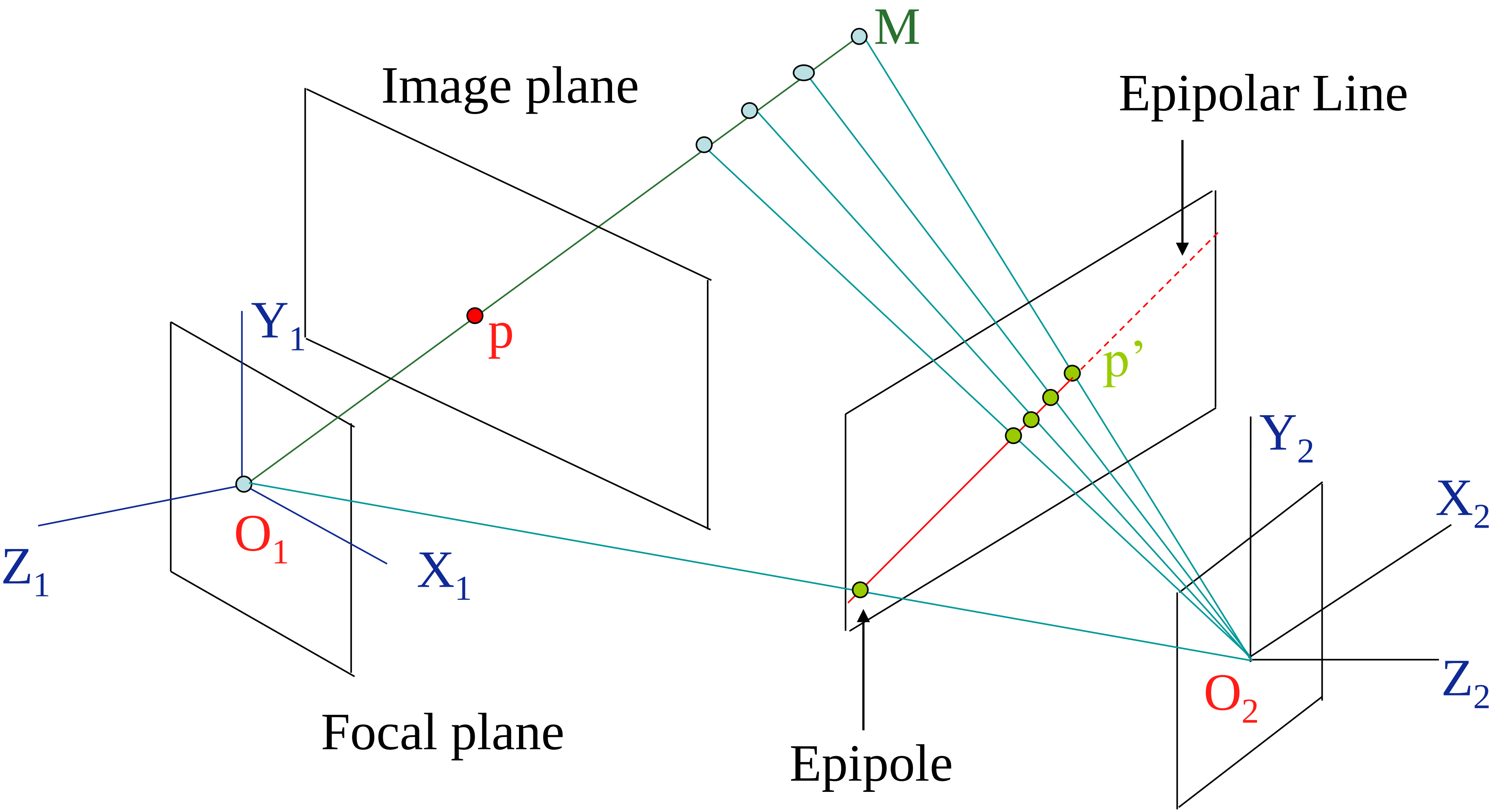
Stereo



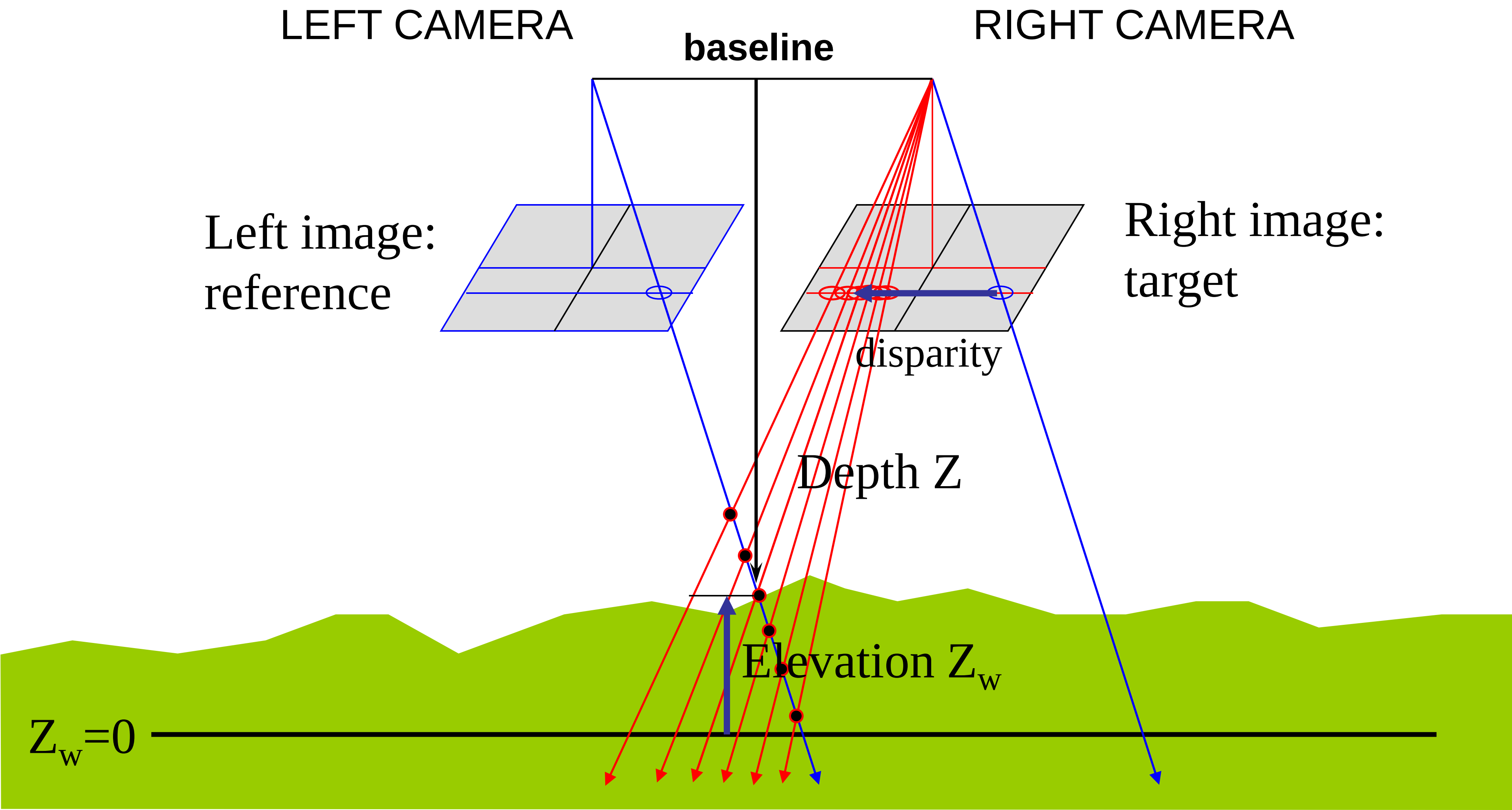




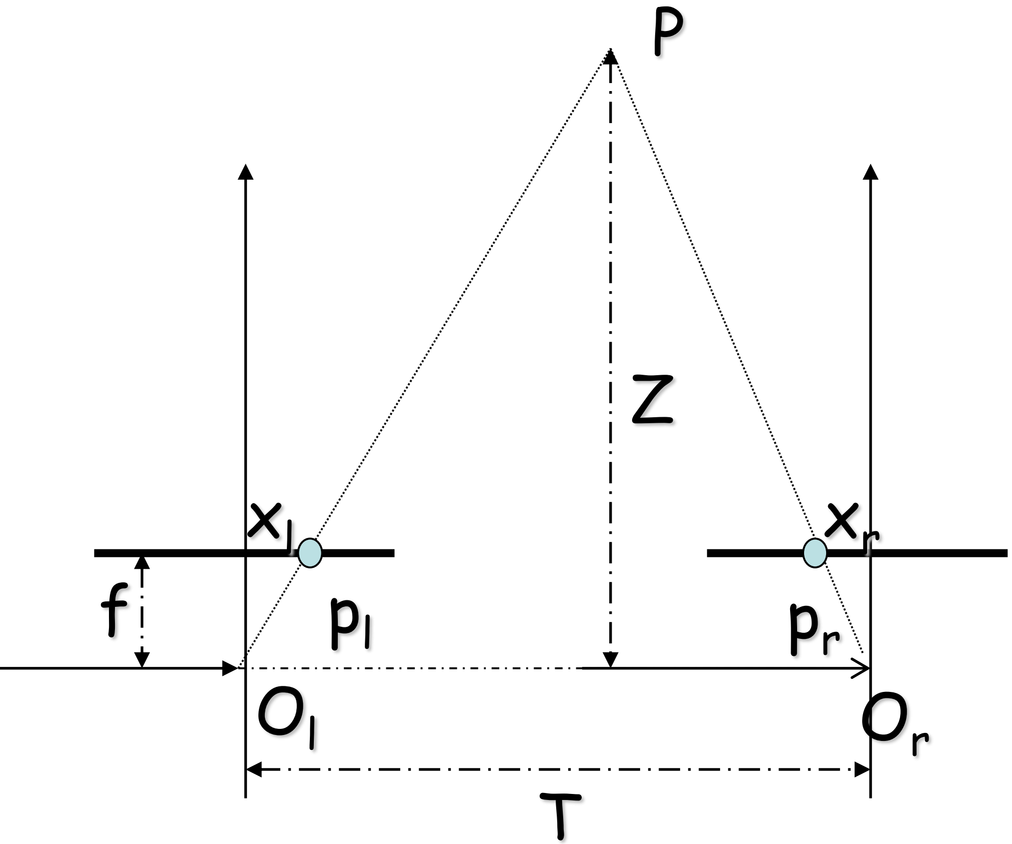
Stereo Constraints



A Simple Stereo System



Parallel Cameras



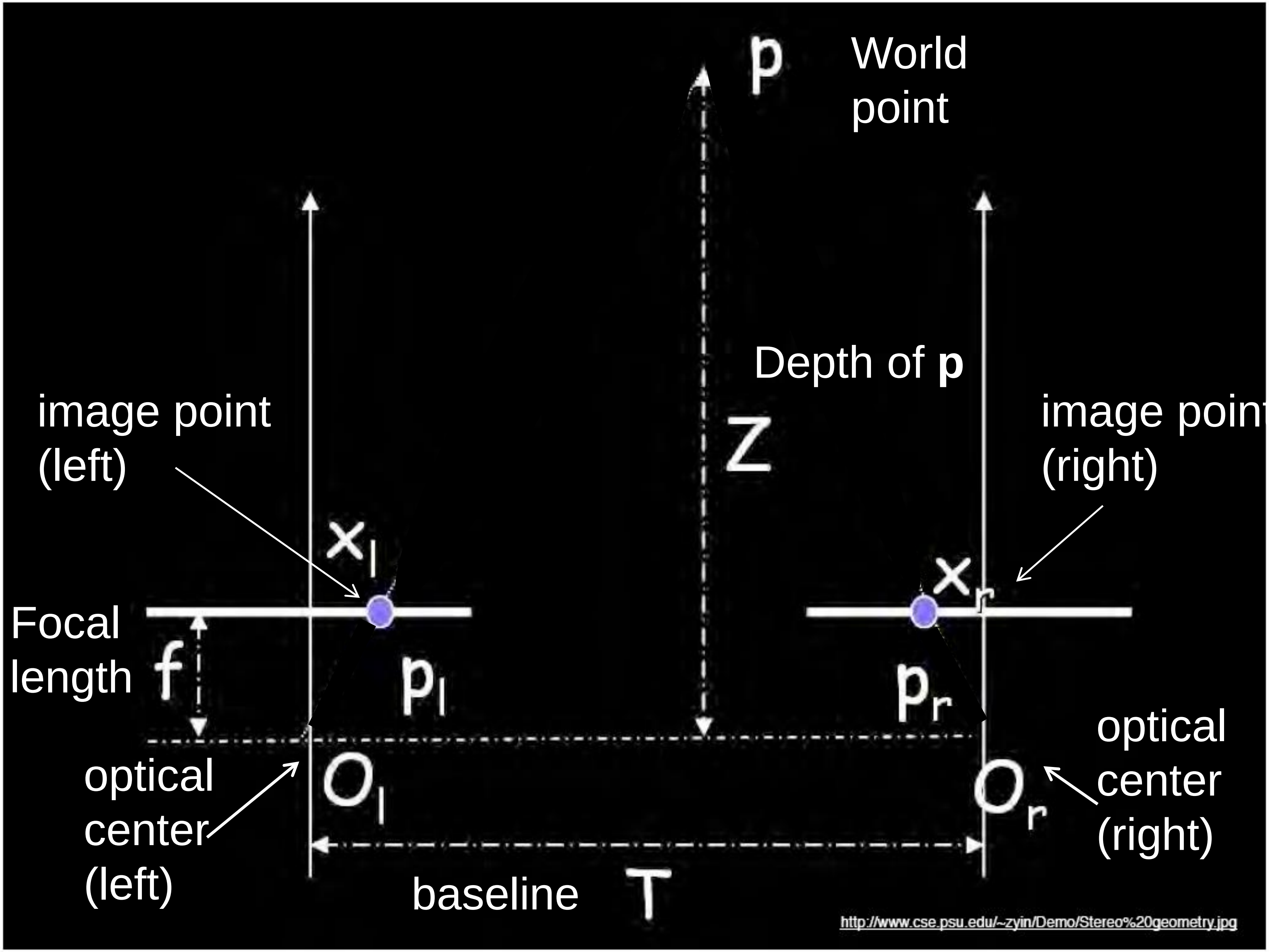
$$\frac{T + x_r - x_l}{Z - f} = \frac{T}{Z}$$

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

Disparity: $d = x_l - x_r$

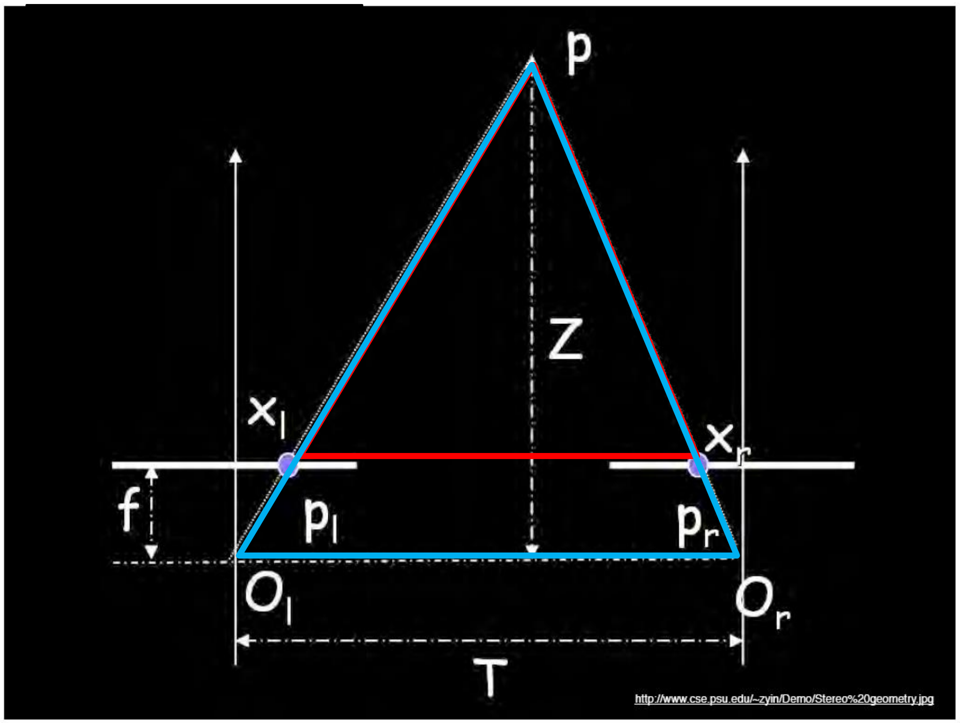
$$\boxed{Z = f\frac{T}{d}}$$

T is the stereo baseline


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)
optical center (right)
O_l
O_r
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?**



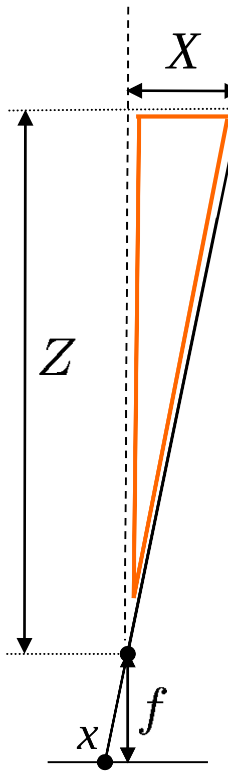
Similar triangles (p_l, P, p_r) and (O_l, P, O_r):

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

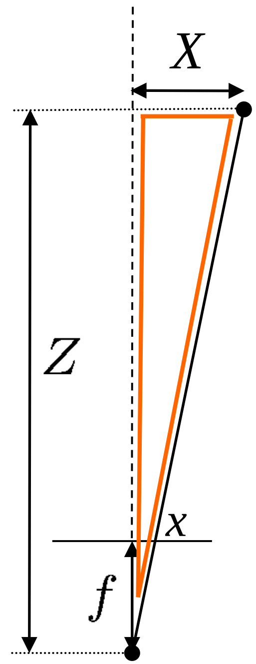
$$Z = f \frac{T}{x_l - x_r}$$

disparity

Perspective projection



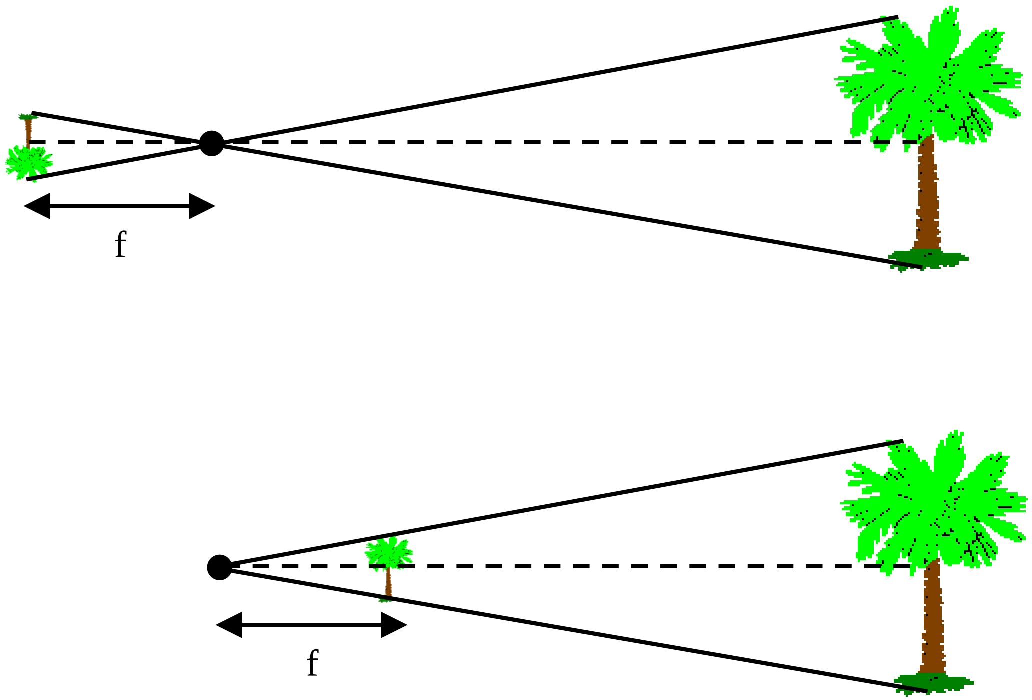
$$x = -f\frac{X}{Z}$$
$$y = -f\frac{Y}{Z}$$



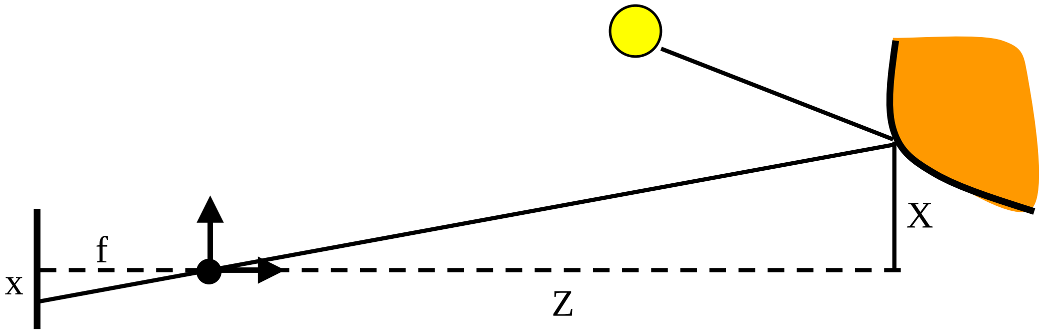
$$x = f\frac{X}{Z}$$
$$y = f\frac{Y}{Z}$$

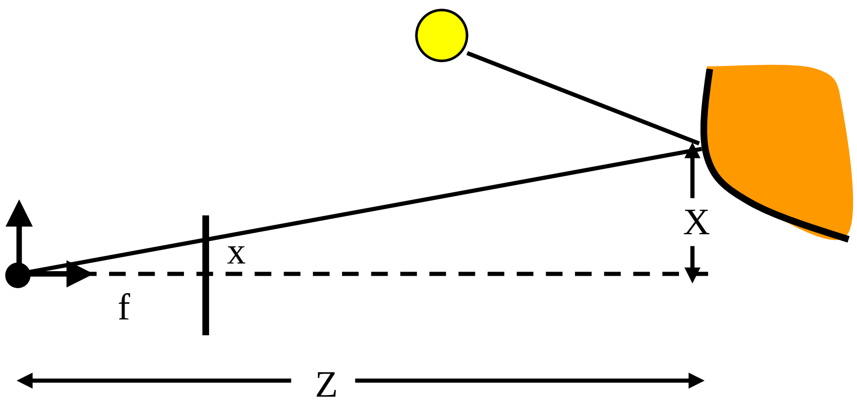
M. Pollefeys, http://www.cs.unc.edu/Research/vision/comp256fall03/

Perspective projection

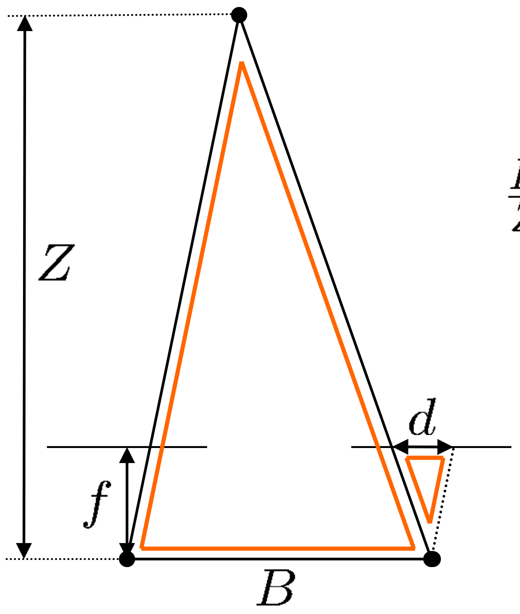


Perspective projection

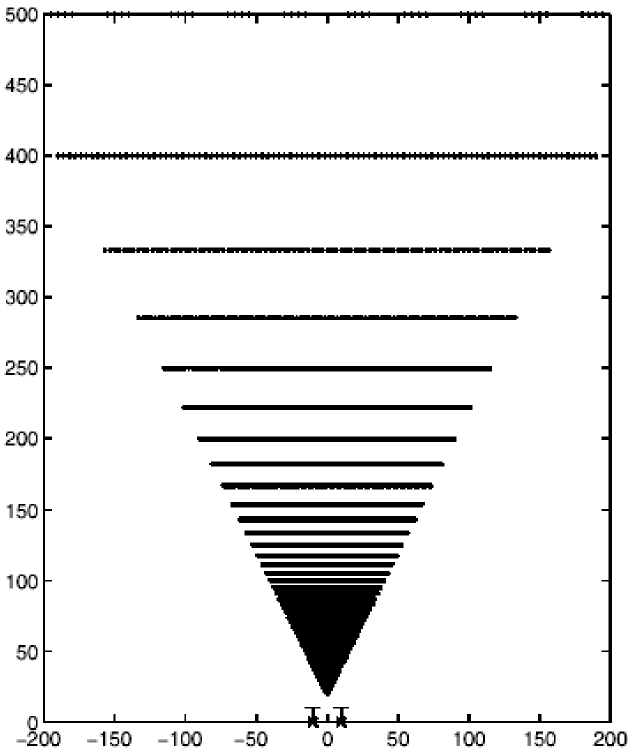




Standard stereo geometry



$$\frac{B}{Z} = \frac{d}{f}$$

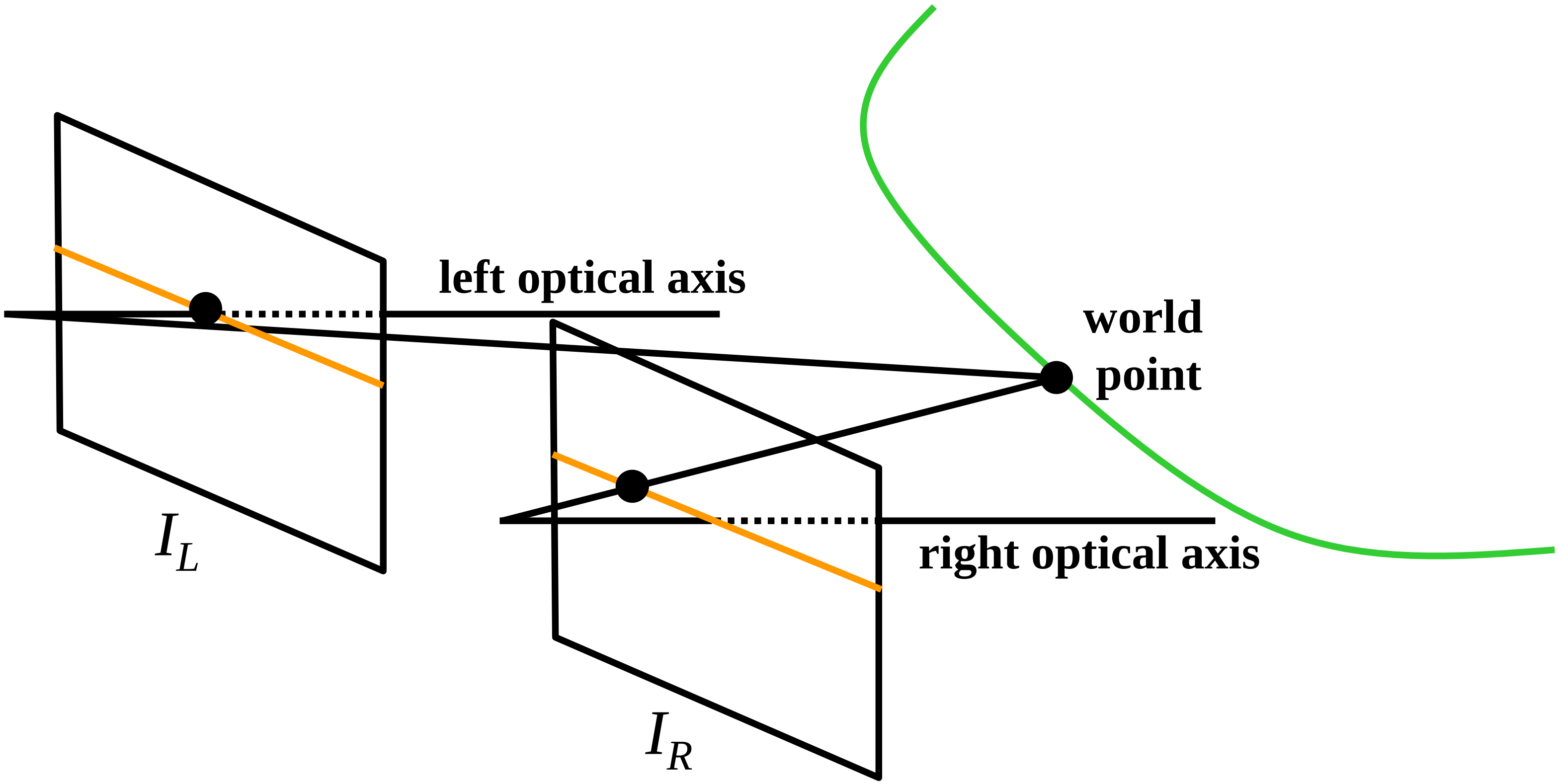


$$d = x_L - x_R = f\frac{X_L}{Z} - f\frac{X_R}{Z} = f\frac{X_L - X_R}{Z} = f\frac{B}{Z}$$

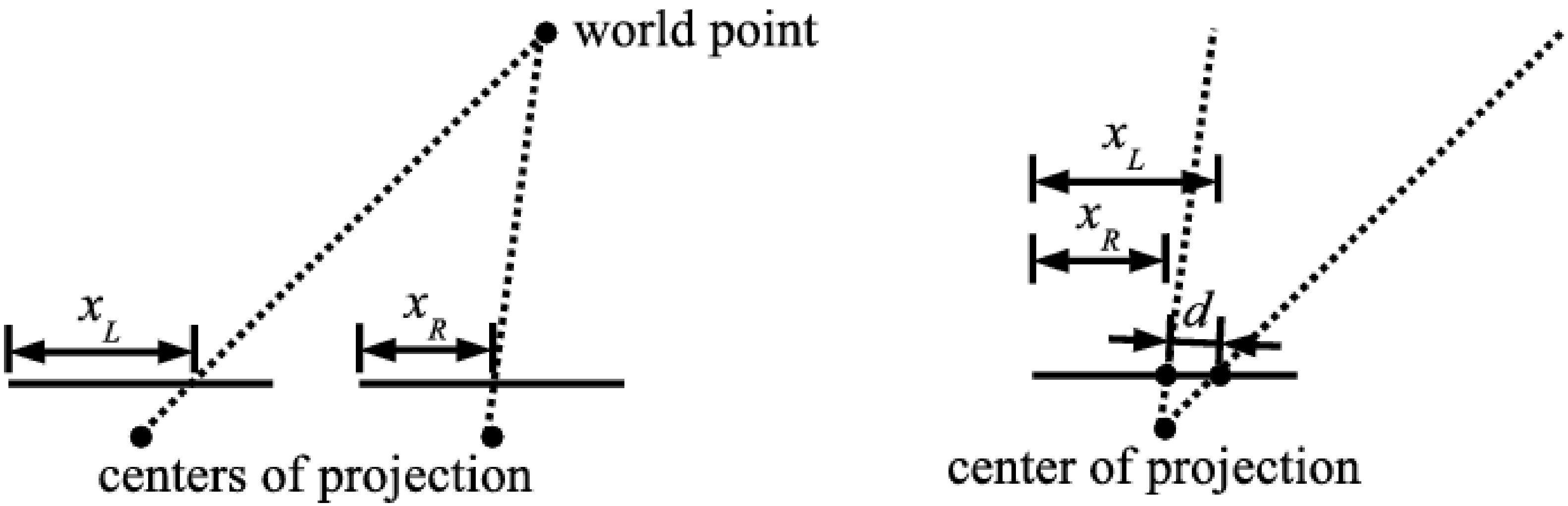
- **disparity is inversely proportional to depth**
- **stereo vision is less useful for distant objects**

M. Pollefeys, http://www.cs.unc.edu/Research/vision/comp256fall03/

Rectified geometry



Rectified geometry

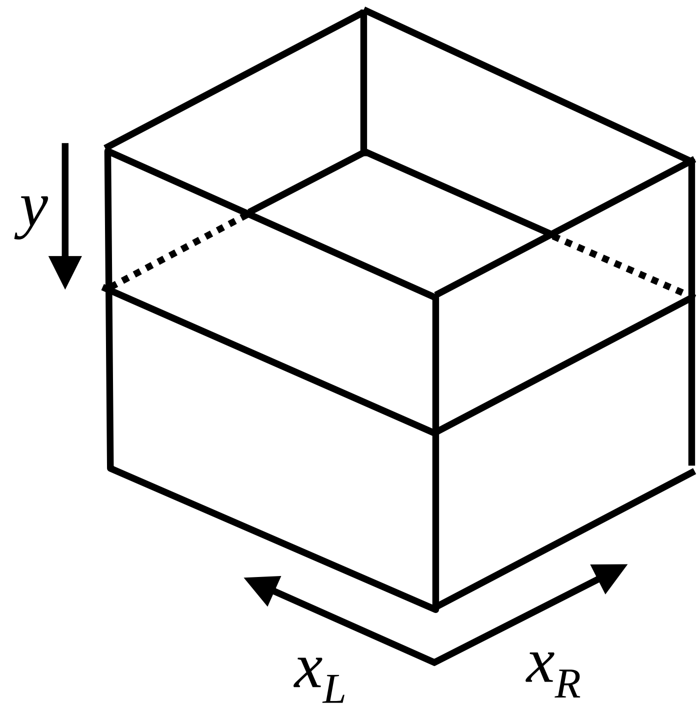


two cameras **overlapped (for display)**

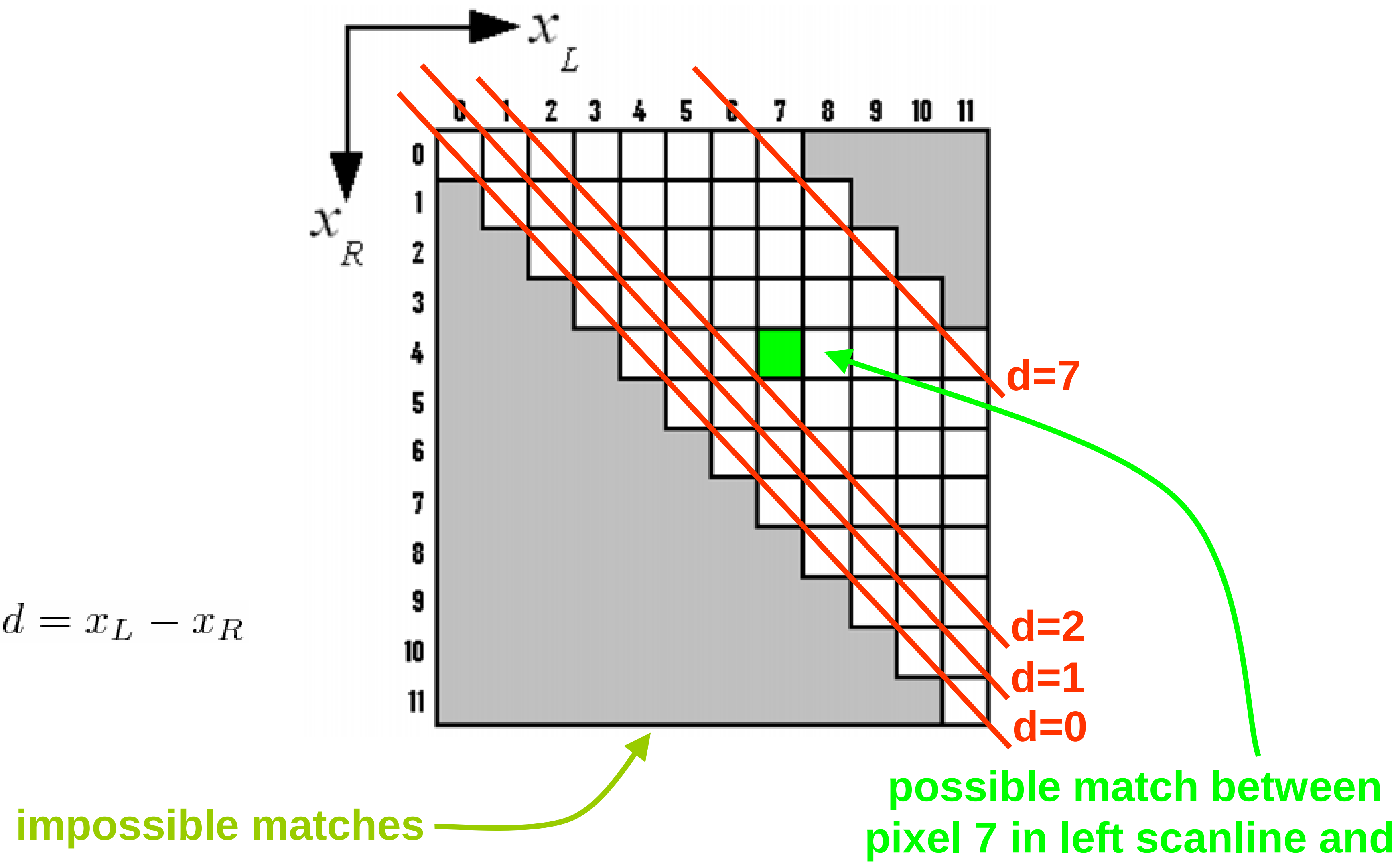
$$d = x_1 - x_2 = f\,(X_1 - X_2)\,/\,Z = f\,b\,/\,Z$$

disparity (d) **baseline** (b) **depth** (Z)

Matching space

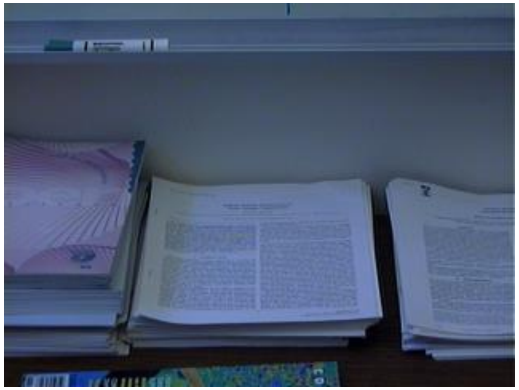


Matching space



$$d = x_L - x_R$$

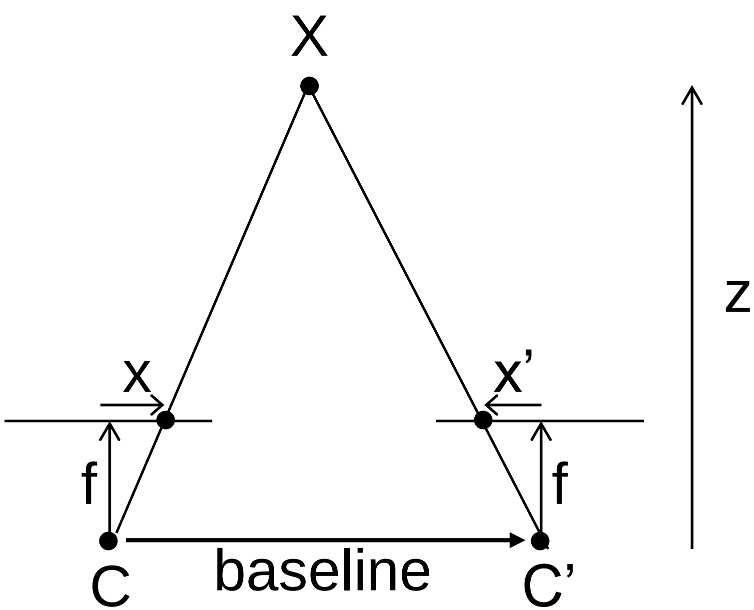
Depth from disparity



input image (1 of 2)

depth map
[Szeliski & Kang '95]

3D rendering



$$disparity = x - x' = \frac{baseline * f}{z}$$

Depth from disparity

image I(x,y)

Disparity map D(x,y)

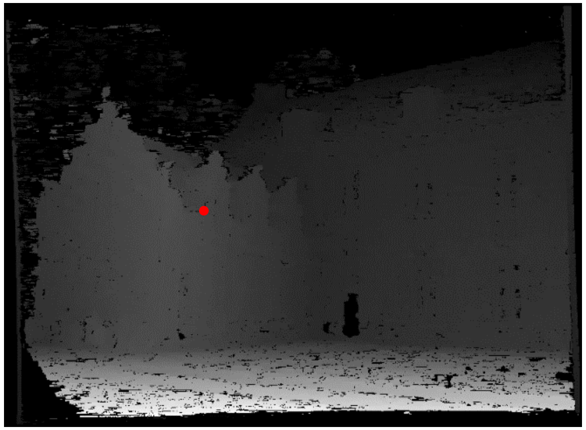
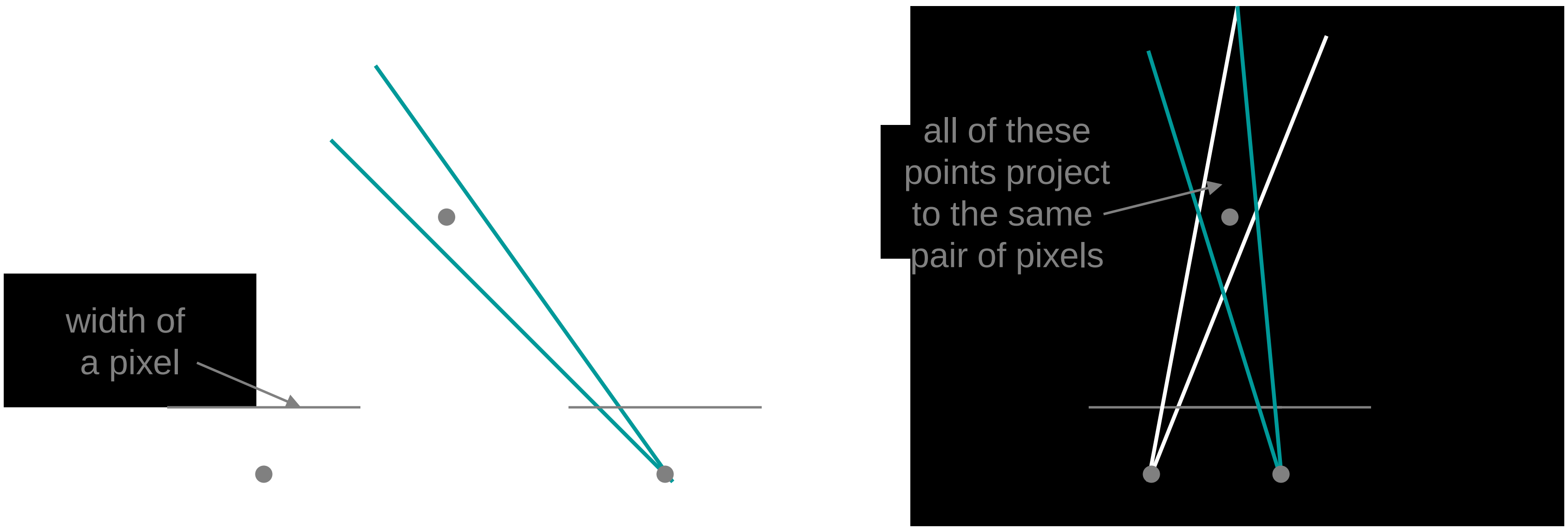


image I´(x´,y´)

(x´,y´)=(x+D(x,y), y)

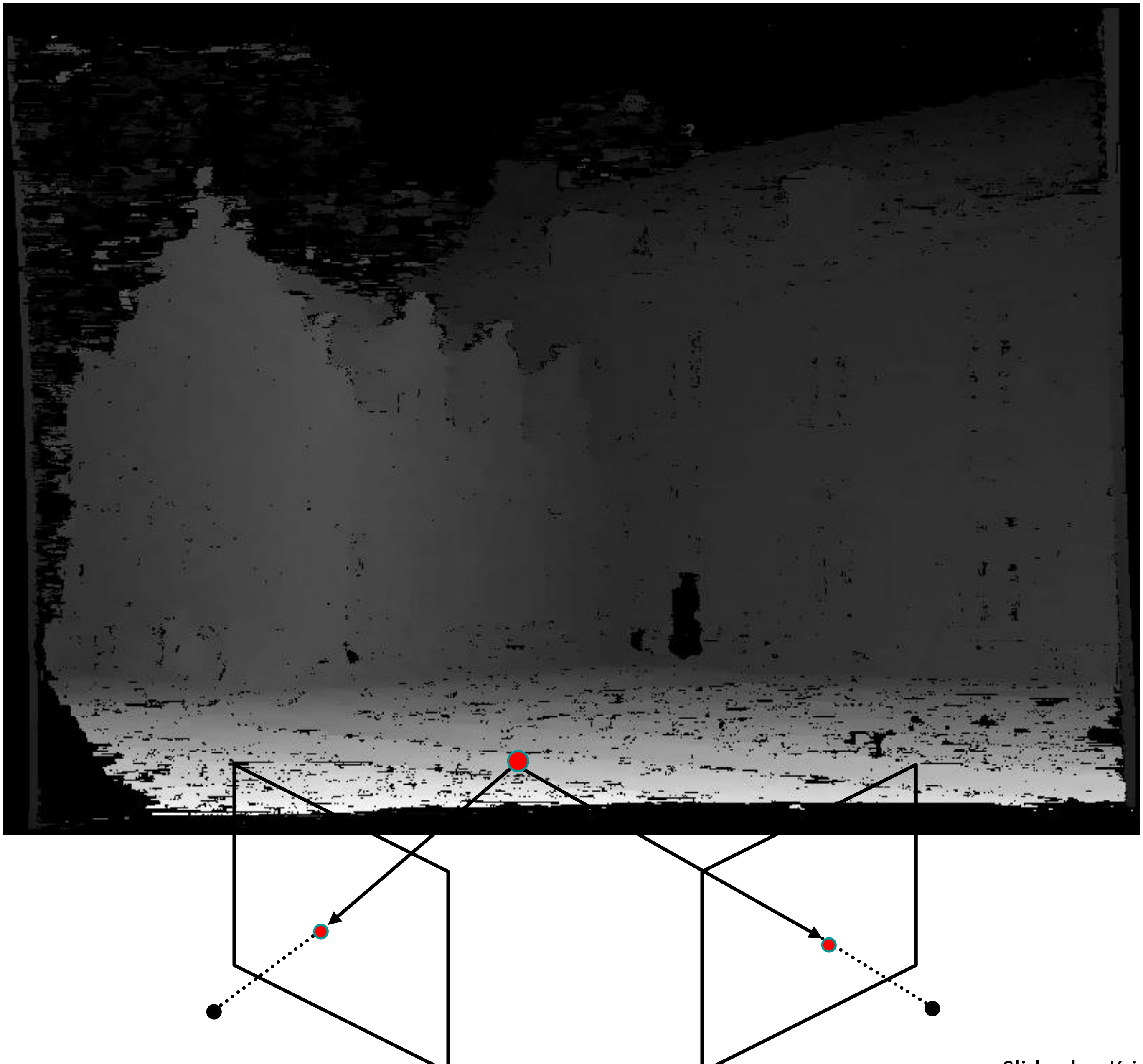
So if we could find the **corresponding points** in two images, we could **estimate relative depth**…

Choosing the stereo baseline



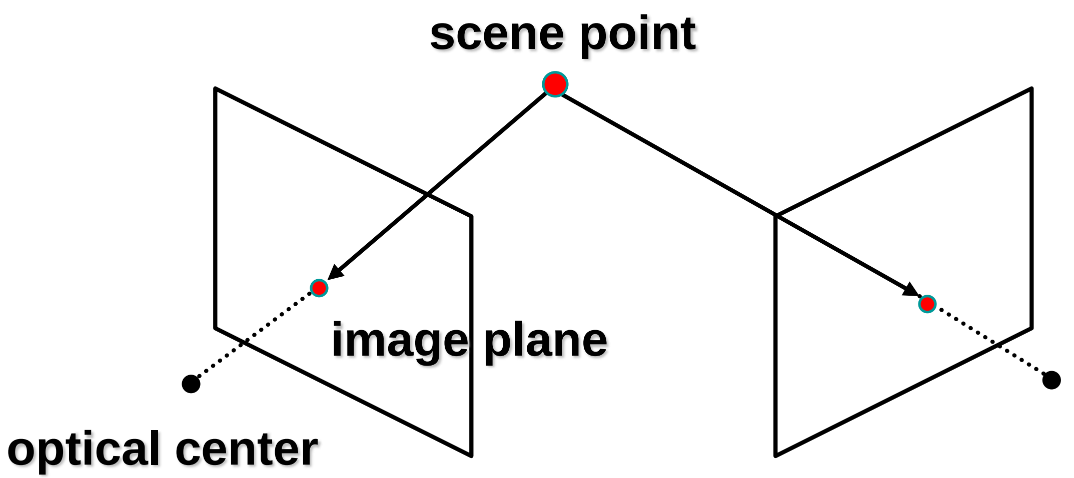
Large Baseline **Small Baseline**

- What's the optimal baseline?
  - Too small: large depth error
  - Too large: difficult search problem

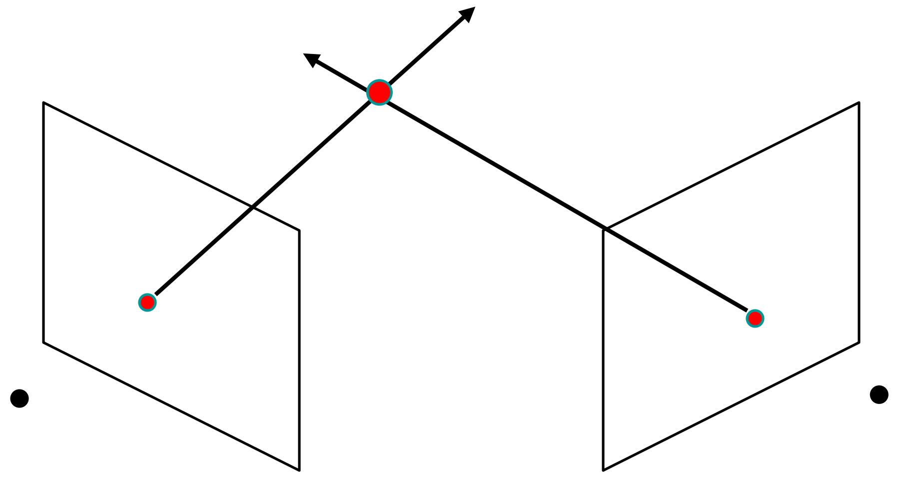


Slides by Kristen Grauman

Stereo



Stereo

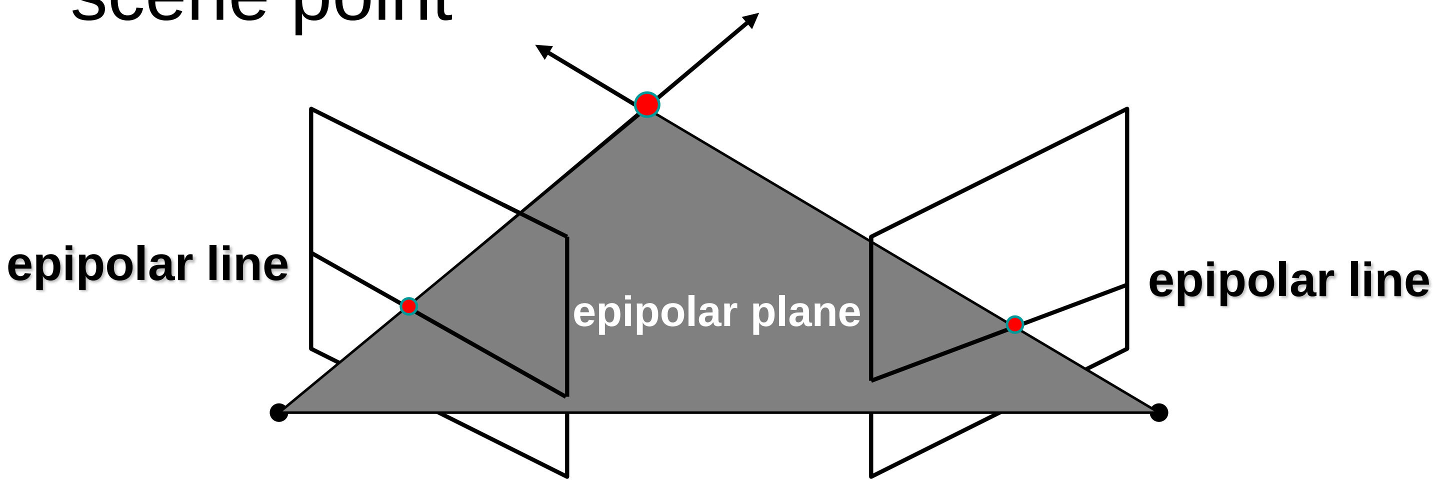


Basic Principle: Triangulation

- Gives reconstruction as intersection of two rays
- Requires
  - calibration
  - ***point correspondence***

Stereo correspondence

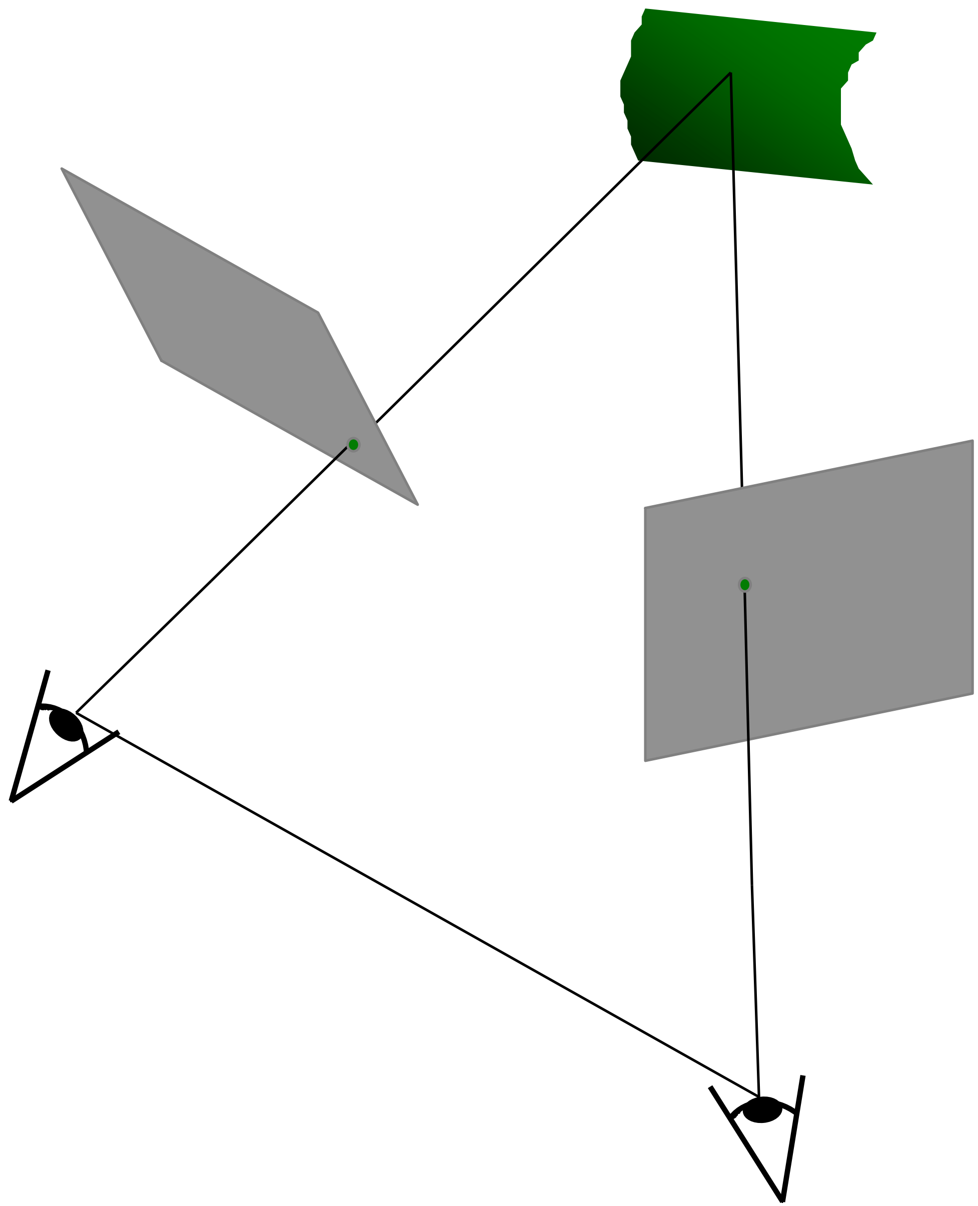
- Determine Pixel Correspondence
  - Pairs of points that correspond to same scene point



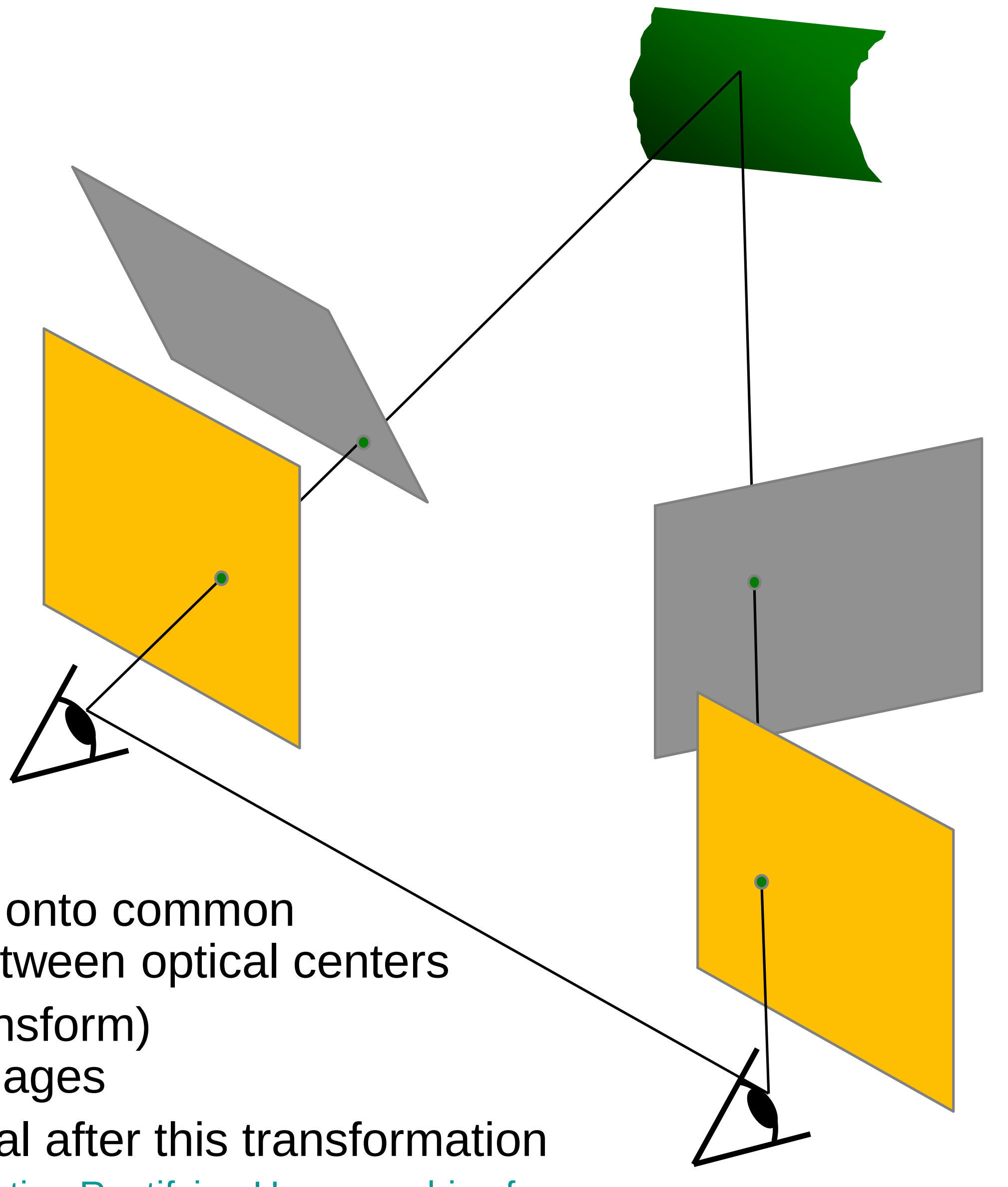
Epipolar Constraint

- Reduces correspondence problem to 1D search along *conjugate epipolar lines*

Stereo image rectification



Stereo image rectification

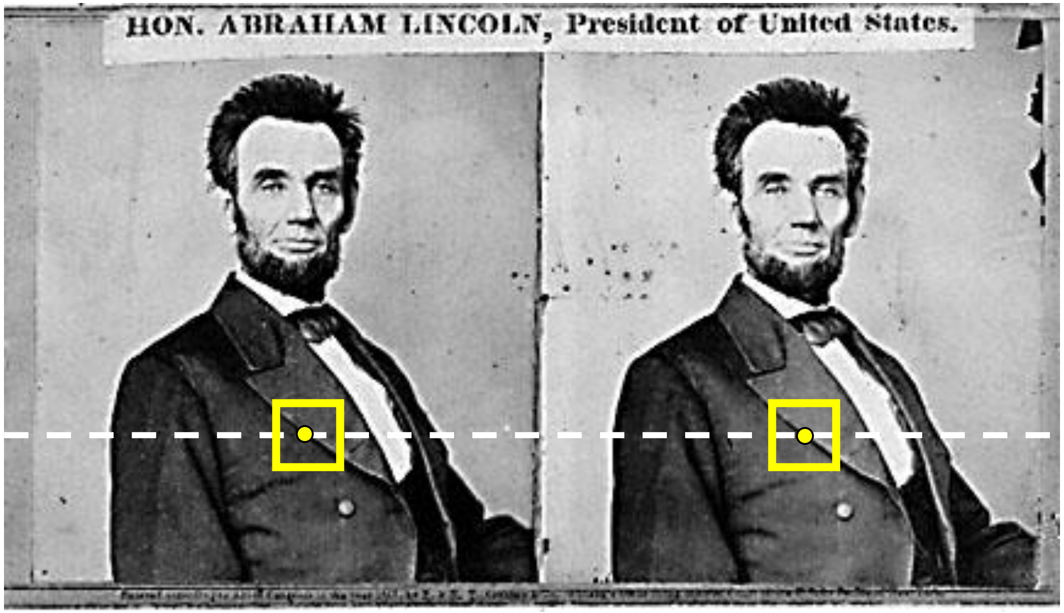


- Image Reprojection
  - reproject image planes onto common plane parallel to line between optical centers
  - a homography (3x3 transform) applied to both input images
  - pixel motion is horizontal after this transformation
  - C. Loop and Z. Zhang. Computing Rectifying Homographies for Stereo Vision. IEEE Conf. Computer Vision and Pattern Recognition, 1999.

Stereo matching algorithms

- Match Pixels in Conjugate Epipolar Lines
  - Assume brightness constancy
  - This is a tough problem
  - Numerous approaches
    - A good survey and evaluation: http://www.middlebury.edu/stereo/

Your basic stereo algorithm



For each epipolar line

For each pixel in the left image

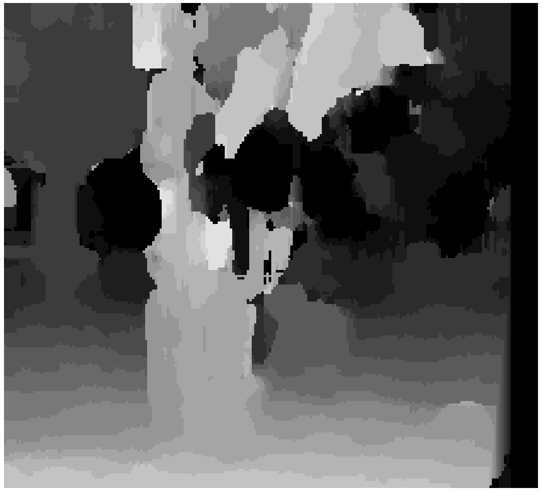
- compare with every pixel on same epipolar line in right image
- pick pixel with minimum match cost

Improvement: match ***windows***

- This should look familar...
- Can use Lukas-Kanade or discrete search (latter more common)

Window size

W = 3



W = 20

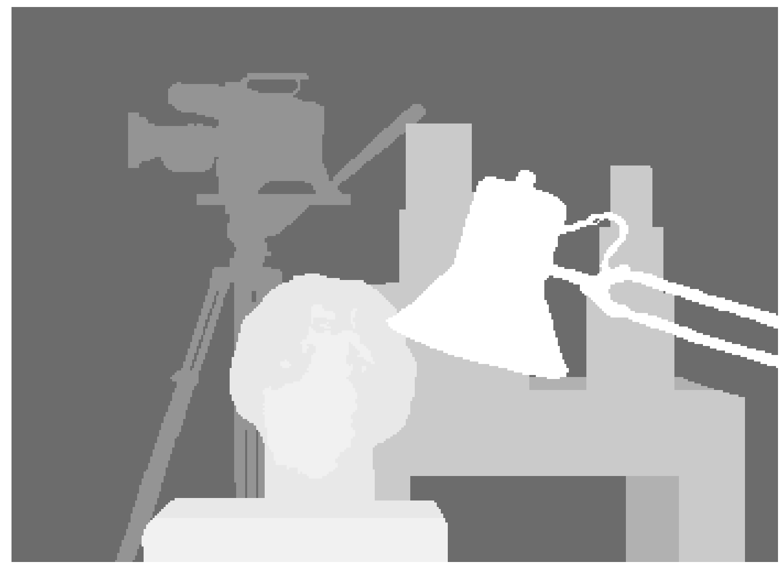
Effect of window size

- Smaller window
  - +
  - •
- Larger window
  - +
  - •

Stereo results

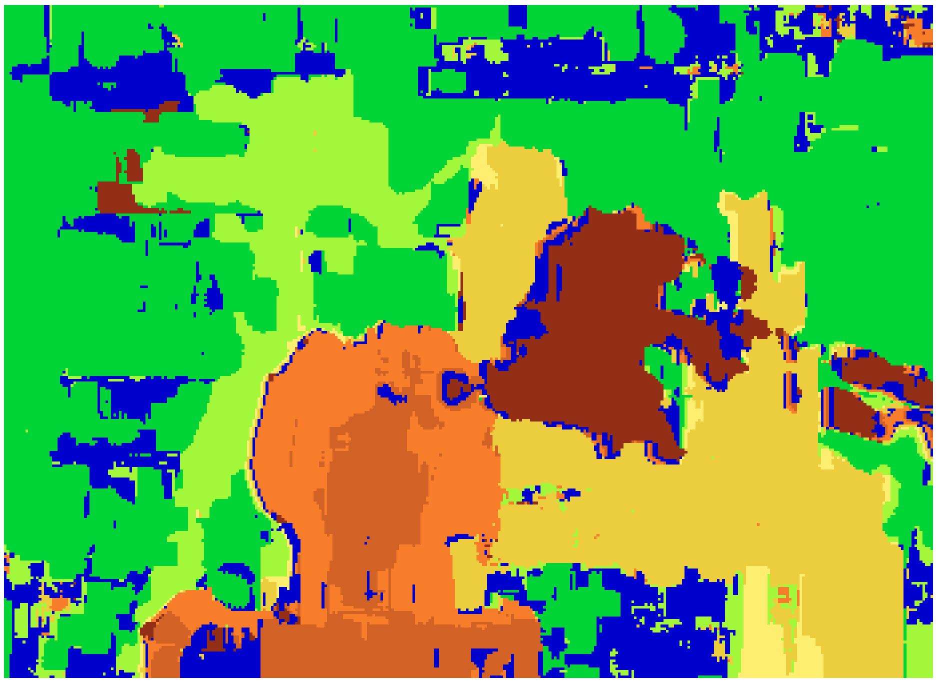
- – Data from University of Tsukuba
- – Similar results on other images without ground truth

Scene



Ground truth

Results with window search



Window-based matching
(best window size)

Ground truth

Better methods exist...

State of the art method

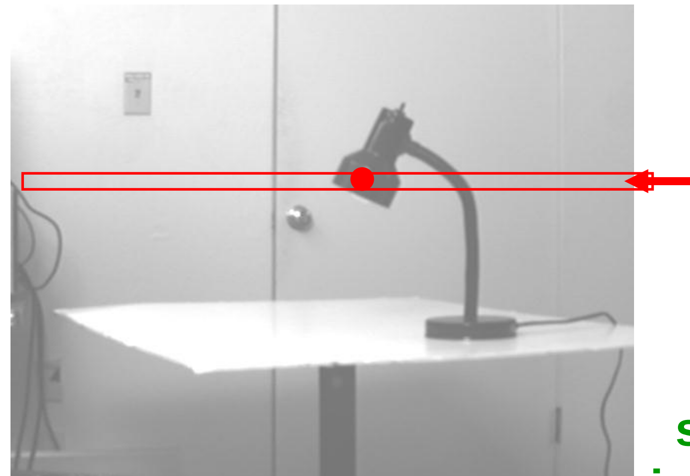
Ground truth

Boykov et al., [Fast Approximate Energy Minimization via Graph Cuts](), International Conference on Computer Vision, September 1999.

Binocular stereo matching

Binocular rectified stereo

left

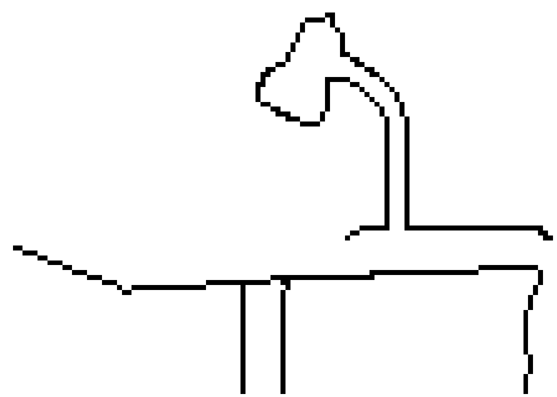


right

epipolar constraint

1D search: look for similar pixel in other image

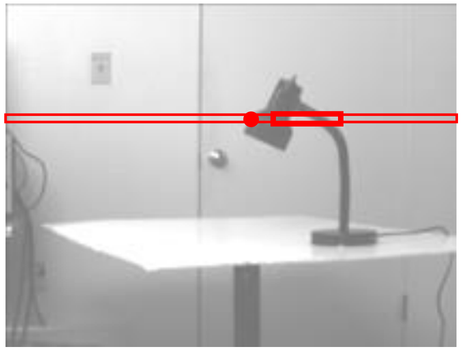
disparity map



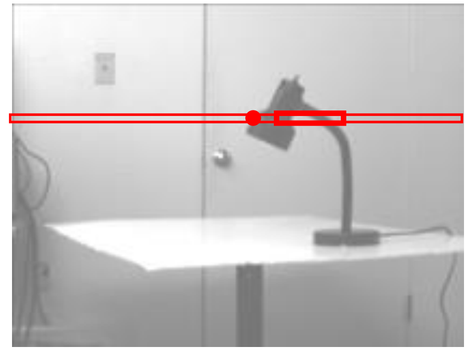
depth discontinuities

S. Birchfield, Clemson Univ., ECE 847, http://www.ces.clemson.edu/~stb/ece847

Disparity function

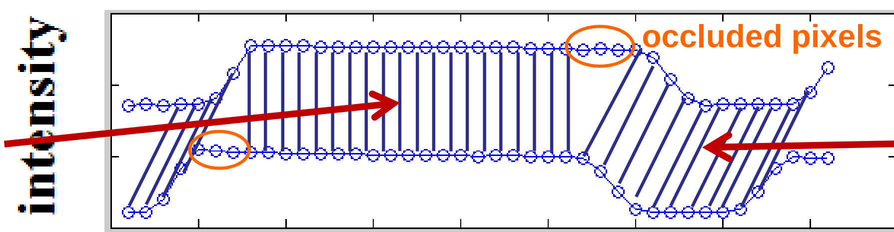


left

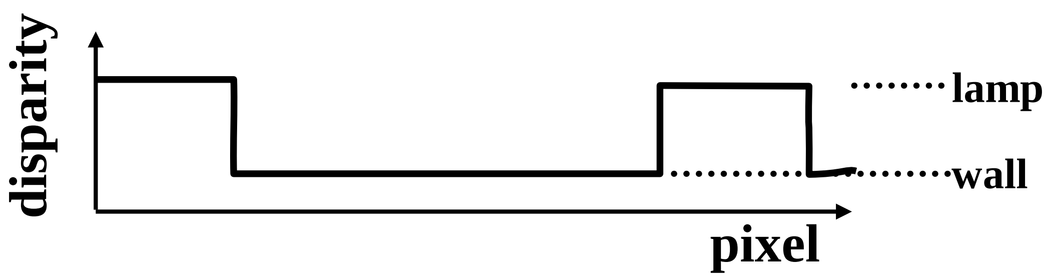


right

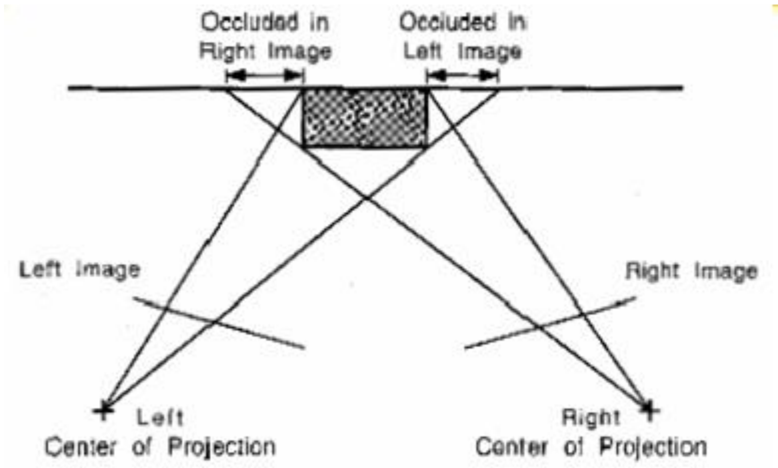
smaller slope =
smaller disparity =
farther from camera



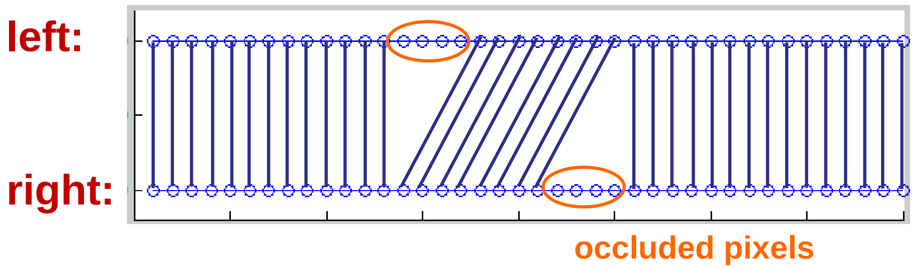
higher slope =
larger disparity =
closer to camera



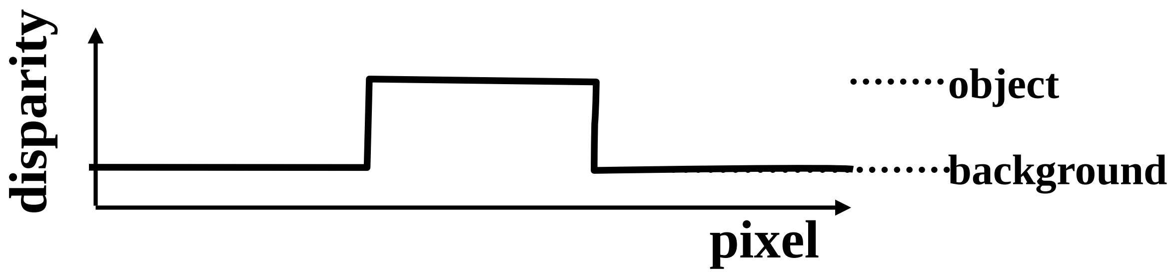
Occlusions



Occluded in Right Image
Occluded in Left Image
Left Image
Right Image
Left Center of Projection
Right Center of Projection



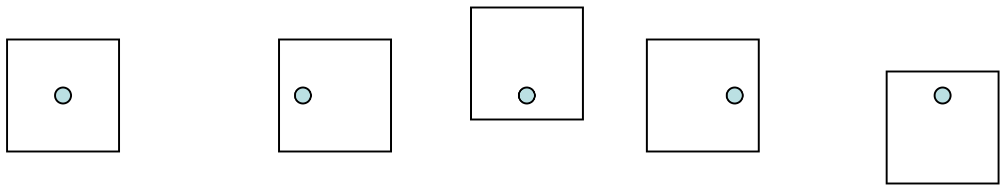
left:
right:
occluded pixels



disparity
object
background
pixel

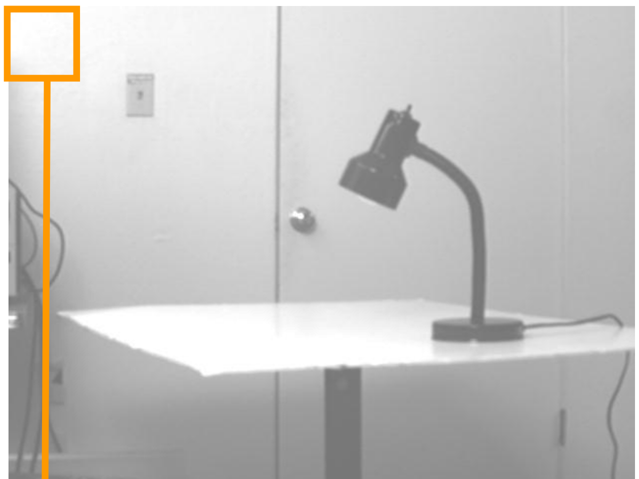
Matching a pixel

- Pixel's value is not unique
  - Only 256 values but ~100,000 pixels!
  - Also, noise affects value
- Solution: use more than one pixel
- Assume neighbors have similar disparity
  - Correlation window around pixel



  - Can use any similarity measure

Block matching

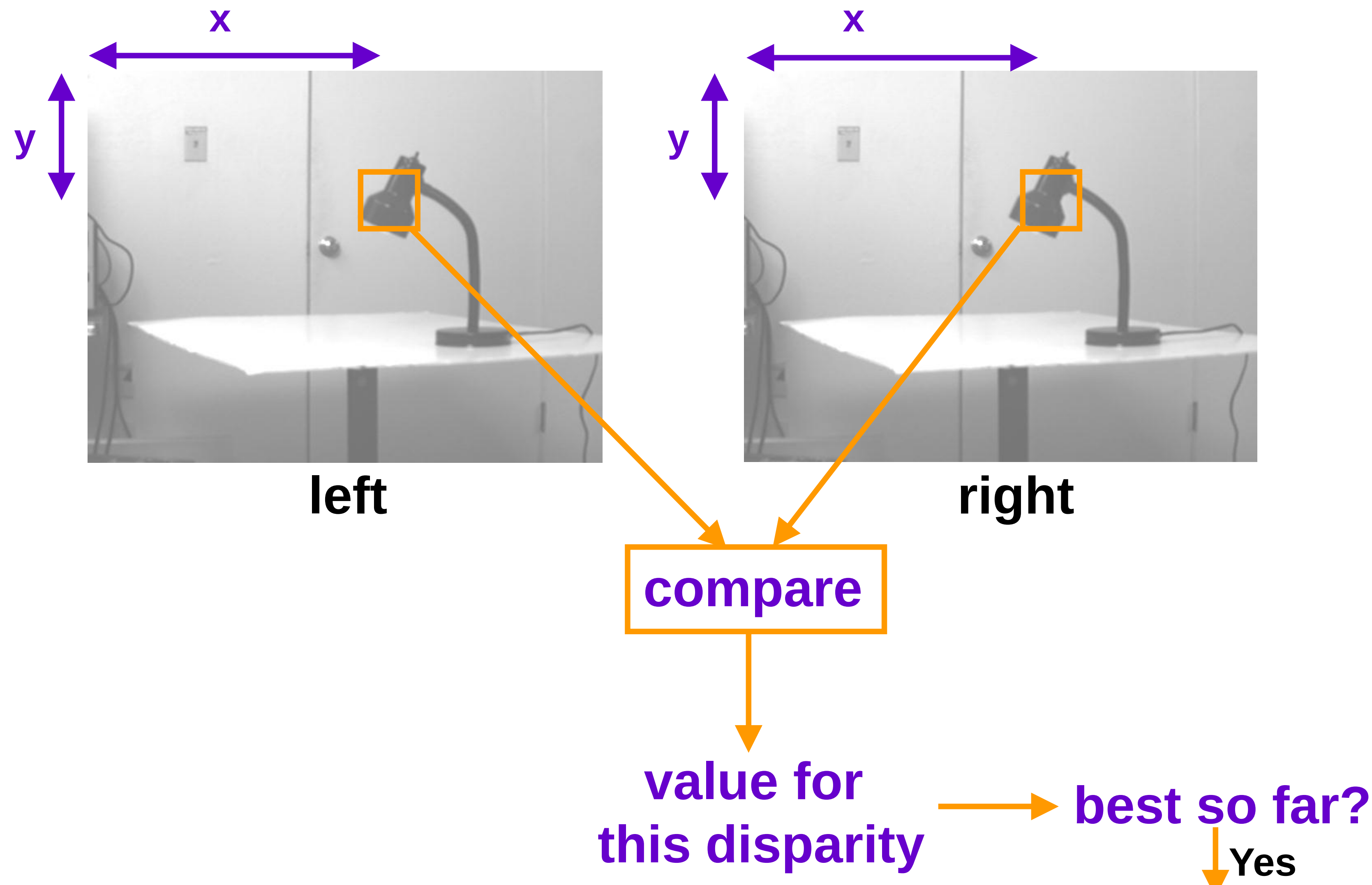


left

disparity map

- compute best disparity for each pixel
- store result in disparity map

Block matching (cont.)



Note: Window only moves left. Why?

Block matching

$$d_L(x,y) = \arg \min_{0 \le d \le d_{\max}} dissim\left(I_L(x,y), I_R(x-d,y)\right)$$

disparity **dissimilarity**

```
Function disparity_map = BlockMatch1(img_left, img_right; min_disp, max_disp)
  for y = 0 to height-1
    for x = 0 to width-1
      ghat = infinity
      for d = min_disp to max_disp
        g = 0
        for j = -w to w
          for i = -w to w
            g = g + dissimilarity(img_left(x+i, y+j), img_right(x+i-d, y+j))
        if g < ghat,
          ghat = g
          dhat = d
      disparity_map(x, y) = dhat
```

5 nested for loops!!!!!

Block matching

$$d_L(x,y) = \arg\min_{0 \le d \le d_{\max}} dissim\left(I_L(x,y), I_R(x-d,y)\right)$$

disparity

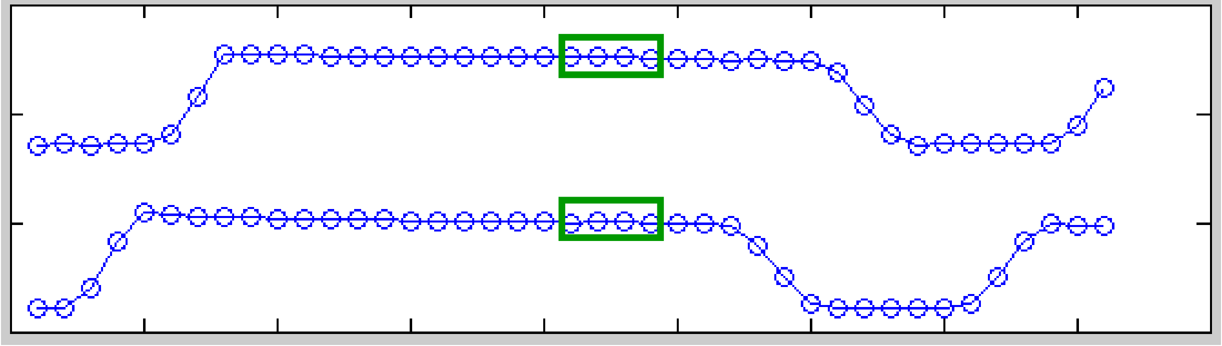
dissimilarity

$\text{BlockMatch1}(I_L, I_R, d_{\min}, d_{\max})$

for $(x,y) \in I_L$ **do**
    $\hat{g} \leftarrow \infty$
    for $d \leftarrow d_{\min}$ **to** $d_{\max}$ **do**
        $g \leftarrow 0$
        for $(\tilde{x},\tilde{y}) \in \mathcal{W}$ **do**
            $g \leftarrow g + dissim(I_L(x+\tilde{x}, y+\tilde{y}), I_R(x+\tilde{x}-d, y+\tilde{y})$
        if $g < \hat{g}$ **then**
            $\hat{g} \leftarrow g$
            $\hat{d} \leftarrow d$
    $d_L(x,y) \leftarrow \hat{d}$
return d_L

5 nested for loops!!!!!

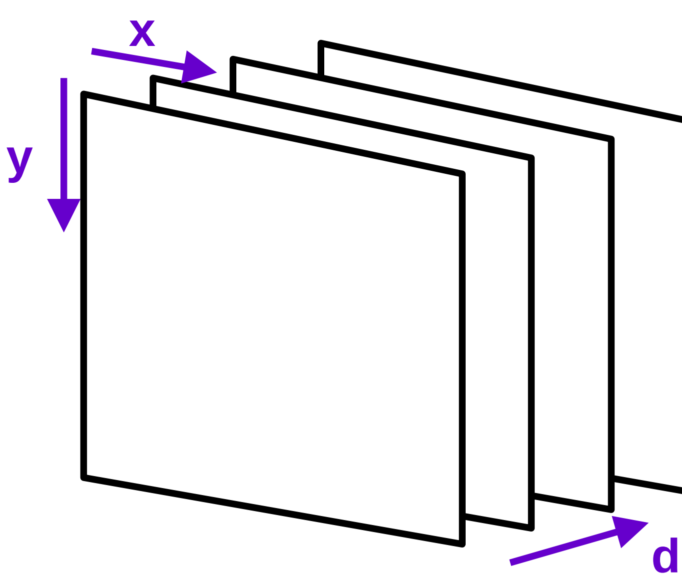
Eliminating redundant computations



for same disparity, overlapping windows recompute the same dissimilarities for many pixels

Block matching: another view

- Alternatively,
  - precompute
    $\Delta(x,y,d) = dissim(\ I_L(x,y),\ I_R(x-d,y)\ )$
    for all *x, y, d*
  - then for each *(x,y)* select the best *d*



More efficient block matching

```
Function dbar = ComputeDbar(img_left, img_right; min_disp, max_disp)
  for d=min_disp:max_disp,
    // compare pixels
    for y=0:height-1,
     for x=0:width-1,
       dbar(x, y, d) = dissimilarity(img_left(x, y), img_right(x-d, y)
    // convolve with 2D box filter to sum over window
    tmp = convolve dbar(:, :, d) with 1D kernel [1 ... 1]
    dbar(:, :, d) = convolve tmp with 1D kernel [1 ... 1]^T
```

} separable

```
Function disparity_map = BlockMatch2(img_left, img_right; min_disp, max_disp)
dbar = ComputeDbar(img_left, img_right; min_disp, max_disp)
  for y=0:height-1,
   for x=0:width-1,
     disparity_map(x, y) = arg min of dbar(x, y, :)
```

Key idea: **Summation over window is convolution with box filter, which is separable**

(only 3 nested for loops!!!)

Running sum improves efficiency even more

More efficient block matching

$\textsc{BlockMatch2}(I_L, I_R, d_{\min}, d_{\max})$

$\Delta \leftarrow \textsc{ComputeSummedDissimilarities}(I_L, I_R, d_{\min}, d_{\max})$
for $(x, y) \in I_L$ **do**
  $d_L(x, y) \leftarrow \arg\min_d \Delta(x, y, d)$
return d_L

$\textsc{ComputeSummedDissimilarities}(I_L, I_R, d_{\min}, d_{\max})$

for $d \leftarrow d_{\min}$ **to** $d_{\max}$ **do**
  for $(x, y) \in I_L$ **do**
    $\Delta(x, y, d) \leftarrow dissim(I_L(x, y), I_R(x - d, y))$
  $\Delta(:, :, d) \leftarrow \textsc{Convolve}(\Delta(:, :, d), \mathbf{1}_{w \times w})$
return Δ

separable

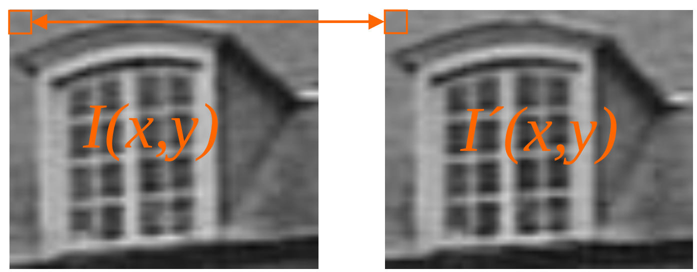
Key idea: **Summation over window is convolution with box filter, which is separable**

(only 3 nested for loops!!!)

Running sum improves efficiency even more

Comparing image regions

Compare intensities pixel-by-pixel



Dissimilarity measures

Sum of Square Differences

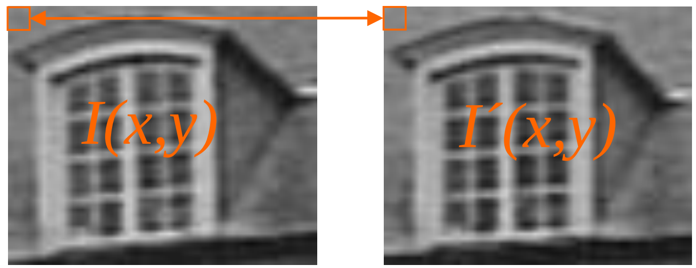
$$SSD = \iint_{W} \left[I'(x,y) - I(x,y) \right]^2 dx dy$$

Note: **SAD is fast approximation (replace square with absolute value)**

M. Pollefeys, http://www.cs.unc.edu/Research/vision/comp256fall03/

Comparing image regions

Compare intensities pixel-by-pixel



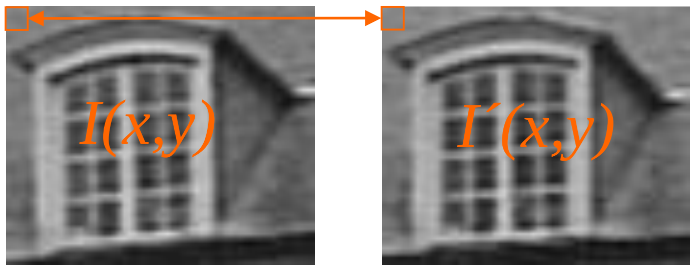
Dissimilarity measures

If energy does not change much, then
minimizing SSD equals maximizing cross-correlation

M. Pollefeys, http://www.cs.unc.edu/Research/vision/comp256fall03/

Comparing image regions

Compare intensities pixel-by-pixel



Similarity measures

Zero-mean Normalized Cross Correlation

$$NCC = \frac{N(I',I)}{\sqrt{N(I',I')N(I,I)}}$$

$$N(A,B) = \iint_W \left(A(x,y) - \overline{A}\right)\left(B(x,y) - \overline{B}\right)dxdy$$

M. Pollefeys, http://www.cs.unc.edu/Research/vision/comp256fall03/

Dissimilarity measures

Most common:

$$D(\mathbf{x}_L, \mathbf{x}_R) = [I_L(x_L, y_L) - I_R(x_R, y_R)]^2 \qquad \text{SSD}$$

$$D(\mathbf{x}_L, \mathbf{x}_R) = |I_L(x_L, y_L) - I_R(x_R, y_R)| \qquad \text{SAD}$$

$$D(\mathbf{x}_L, \mathbf{x}_R) = -I_L(x_L, y_L) I_R(x_R, y_R) \qquad \text{cross correlation}$$

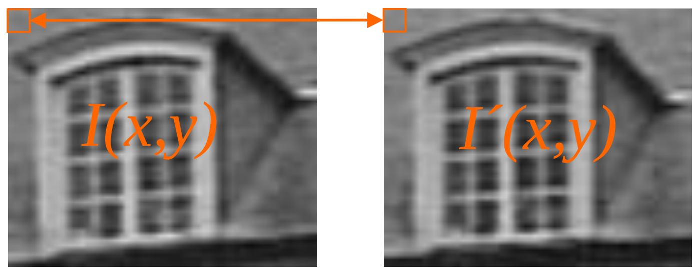
Connection between SSD and cross correlation:

$$\begin{aligned} D(\mathbf{x}_L, \mathbf{x}_R) &= [I_L(x_L, y_L) - I_R(x_R, y_R)]^2 \\ &= [I_L(x_L, y_L)]^2 + [I_R(x_R, y_R)]^2 - 2 I_L(x_L, y_L) I_R(x_R, y_R) \\ &\propto -I_L(x_L, y_L) I_R(x_R, y_R) \end{aligned}$$

Also normalized correlation, rank, census, sampling-insensitive ...

Comparing image regions

Compare intensities pixel-by-pixel



Similarity measures

Census

$$C_I(i,j) = (I(x+i, y+j) > I(x,y))$$

125	126	125
127	128	130
129	132	135

→

0	0	0
0		1
1	1	1

→ [00001111]

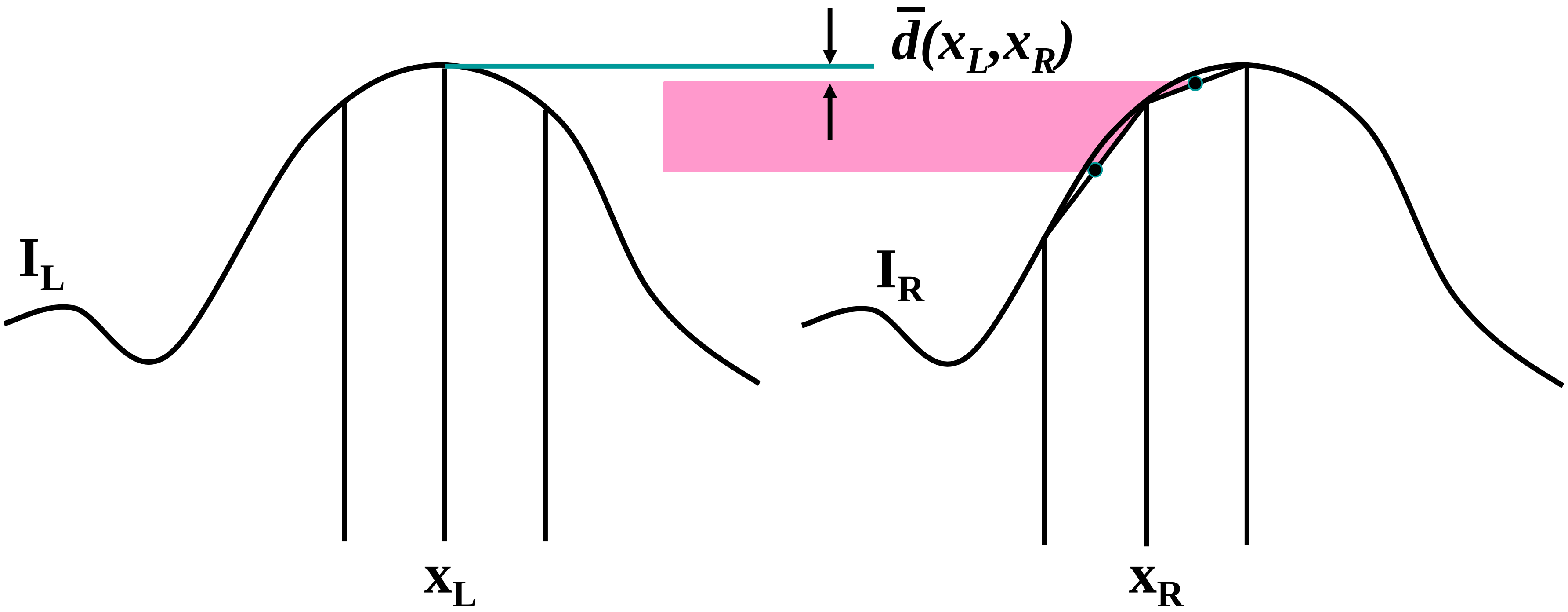
only compare bit signature

using XOR, SAD, or Hamming distance (all equivalent)

(Real-time chip from TZYX based on Census)

M. Pollefeys, http://www.cs.unc.edu/Research/vision/comp256fall03/

Sampling-Insensitive Pixel Dissimilarity



Our dissimilarity measure: $d(x_L,x_R) = \min\{\bar{d}(x_L,x_R), \bar{d}(x_R,x_L)\}$

[Birchfield & Tomasi 1998]

Dissimilarity Measure Theorems

Given: **An interval A such that**
$[x_L - ½ , x_L + ½] \subseteq A$, and
$[x_R - ½ , x_R + ½] \subseteq A$

Theorem 1:

$$\text{If } | x_L - x_R | \leq ½, \text{ then } d(x_L,x_R) = 0$$

(when A is convex or concave)

Theorem 2:

$$| x_L - x_R | \leq ½ \text{ iff } d(x_L,x_R) = 0$$

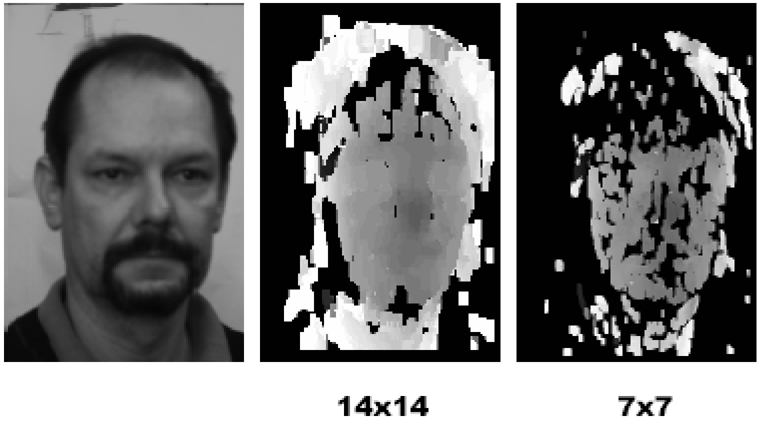
(when A is linear)

[Birchfield & Tomasi 1998]

Aggregation window sizes

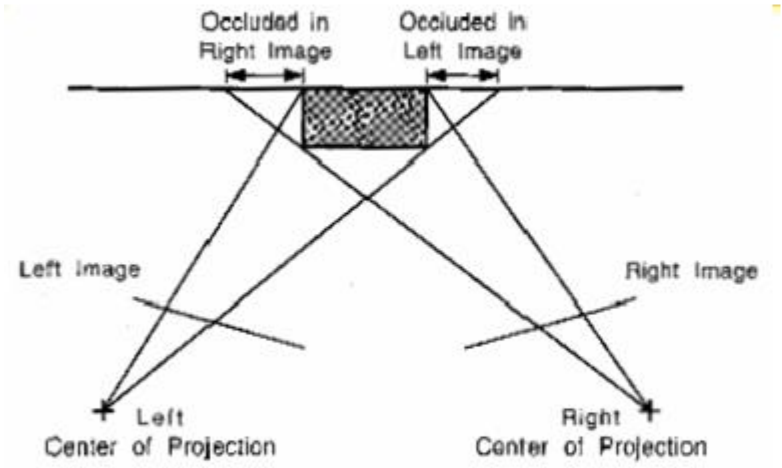
Small windows
- disparities similar
- more ambiguities
- accurate when correct

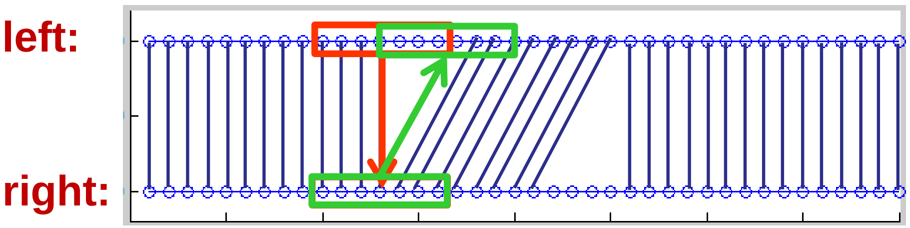
Large windows
- larger disp. variation
- more discriminant
- often more robust
- use shiftable windows to deal with discontinuities



(Illustration from Pascal Fua)

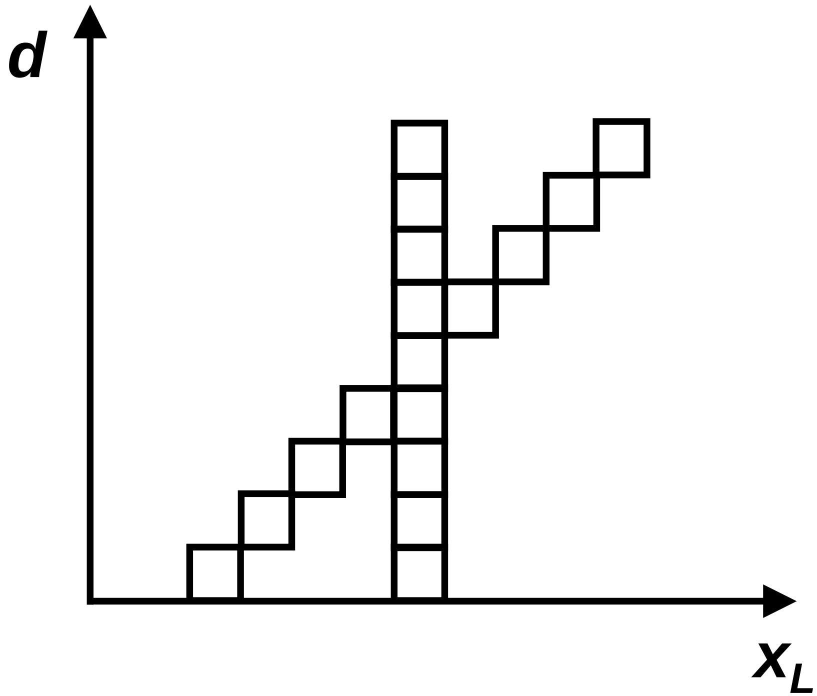
Occlusions





If pixel matches do not agree in both directions, then unreliable

Left-right consistency check



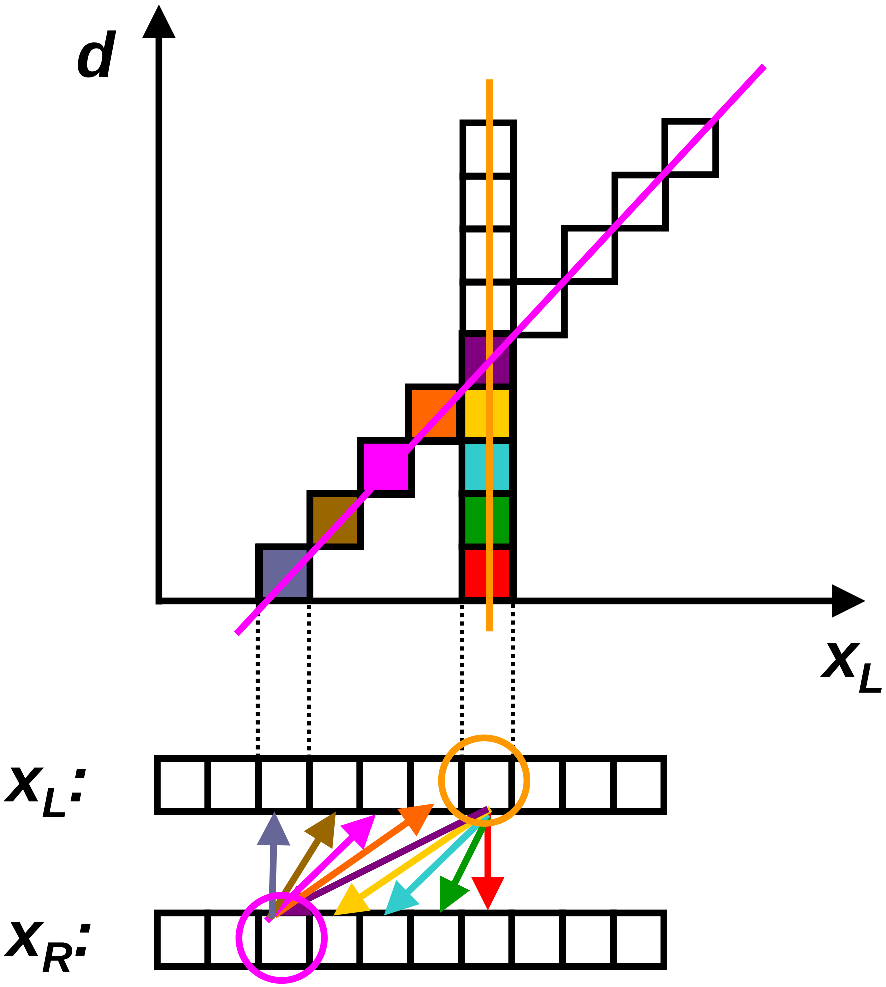
- **Search left-to-right, then right-to-left**
- **Retain disparity only if they agree**

Do minima coincide?

Conceptually,

```
dm_L = BlockMatch(img_left, img_right; 0, max_disp)
dm_R = BlockMatch(img_right, img_left; -max_disp, 0)
for y=0:height-1,
for x=0:width-1,
  if dm_L(x, y) != - dm_R(x - dm_L(x, y), y)
    dm_L(x, y) = NOT_MATCHED
```

Left-right consistency check



for pixel (x,y) in left image, choices are

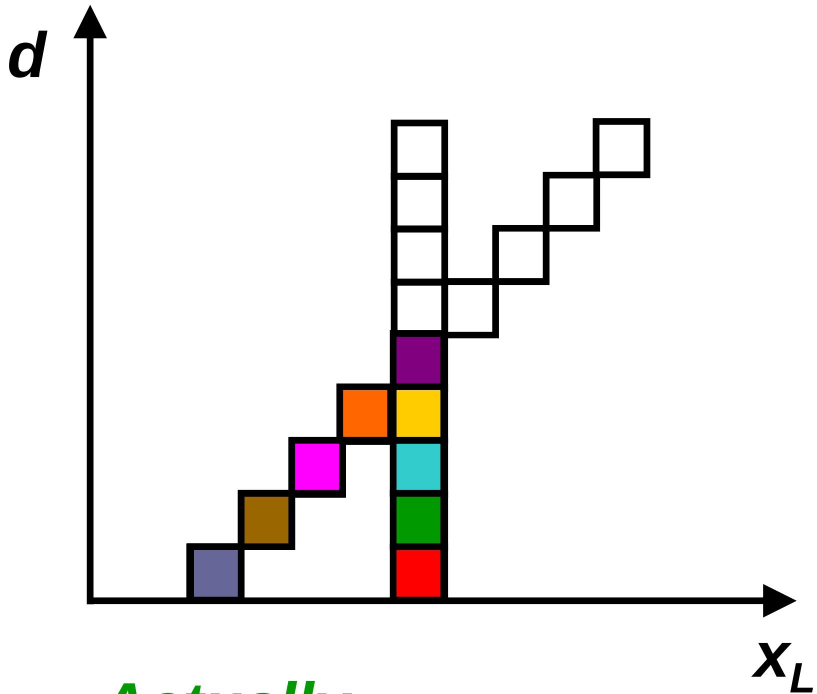
Δ(x,y,0),
Δ(x,y,1),
Δ(x,y,2),
…,
Δ(x,y,max_disp)

for pixel (x,y) in right image, choices are

Δ(x,y,0),
Δ(x+1,y,1),
Δ(x+2,y,2),
…,
Δ(x+max_disp,y,max_disp)

because $x_L = x_R$ + disparity

Left-right consistency check



Actually,

```
Function disparity_map = BlockMatchWithRightLeftCheck(img_left, img_right; max_disp)
△= ComputeDbar(img_left, img_right; 0, max_disp)
for y=0:height-1,
for x=0:width-1,
  // find left answer
  d_left = arg min( △(x,y,0), △(x,y,1), ..., △(x,y,max_disp) )
  d_right = arg min( △(x-d_left,y,0), △(x-d_left+1,y,1), ..., △(x-d_left+max_disp,y,max_disp)
  disp_map(x,y) = (d_left == d_right) ? d_left : NOT_MATCHED
```

With left-right check

inefficient:

$\textsc{BlockMatchWithLeftRightCheck1}(I_L, I_R, d_{\max})$

$d_L \leftarrow \textsc{BlockMatch2}(I_L, I_R, 0, d_{\max})$
$d_R \leftarrow \textsc{BlockMatch2}(I_R, I_L, -d_{\max}, 0)$
for $(x, y) \in I_L$ **do**
    if $d_L(x, y) \neq -d_R(x - d_L(x, y), y)$ **then**
        $d_L(x, y) \leftarrow$ NOT-MATCHED
return d_L

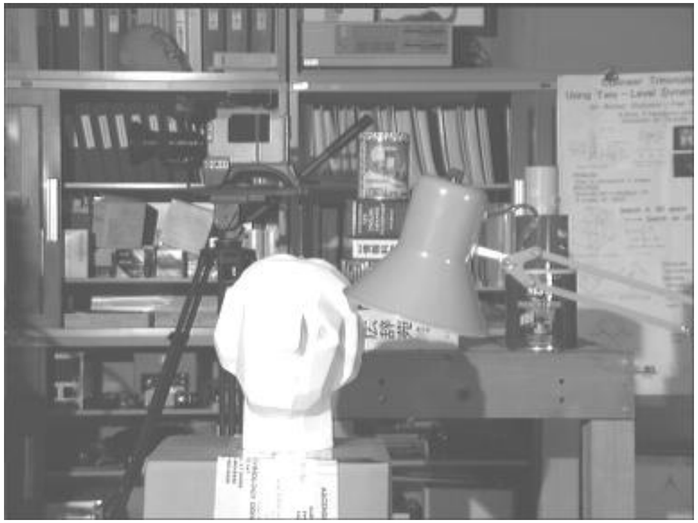
more efficient:

$\textsc{BlockMatchWithLeftRightCheck2}(I_L, I_R, d_{\max})$

$\Delta \leftarrow \textsc{ComputeSummedDissimilarities}(I_L, I_R, 0, d_{\max})$
for $(x, y) \in I_L$ **do**
    $\delta_L \leftarrow \arg\min\{\Delta(x, y, 0), \Delta(x, y, 1), \ldots, \Delta(x, y, d_{\max})\}$
    $\delta_R \leftarrow \arg\min\{\Delta(x - \delta_L, y, 0), \Delta(x - \delta_L + 1, y, 1), \ldots, \Delta(x - \delta_L + d_{\max}, y, d_{\max})\}$
    if $\delta_L == \delta_R$ **then**
        $d_L(x, y) \leftarrow \delta_L$
    else
        $d_L(x, y) \leftarrow$ NOT-MATCHED
return d_L

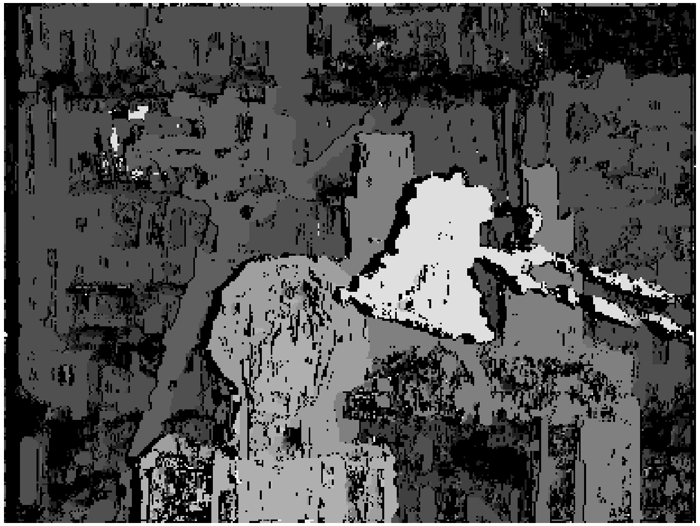
S. Birchfield, Clemson Univ., ECE 847, http://www.ces.clemson.edu/~stb/ece847

Results: correlation



left

disparity map

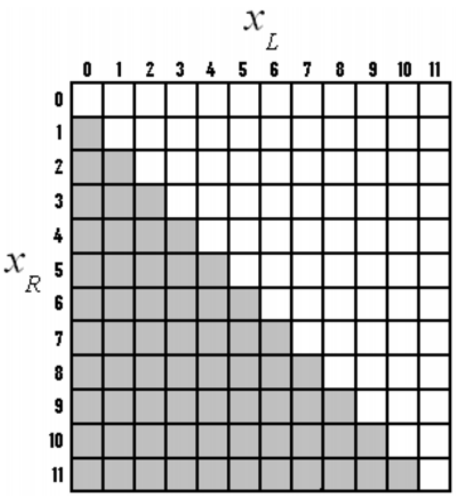


with left-right consistency check

Constraints

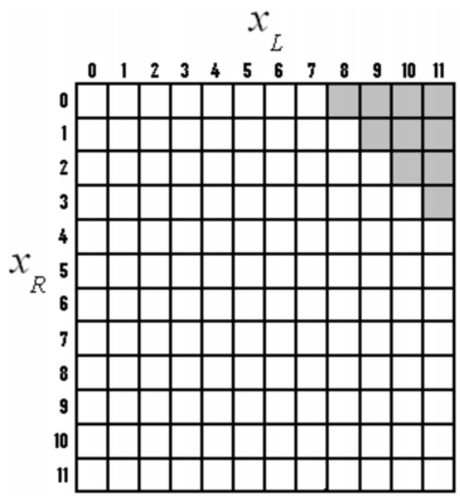
- Epipolar – match must lie on epipolar line
- Piecewise constancy – neighboring pixels should usually have same disparity
- Piecewise continuity – neighboring pixels should usually have similar disparity
- Disparity – impose allowable range of disparities (Panum's fusional area)
- Disparity gradient – restricts slope of disparity
- Figural continuity – disparity of edges across scanlines
- Uniqueness – each pixel has no more than one match (violated by windows and mirrors)
- Ordering – disparity function is monotonic (precludes thin poles)

Stereo constraints

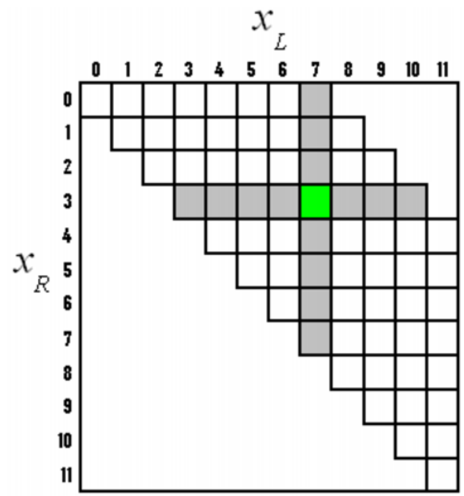


cheirality

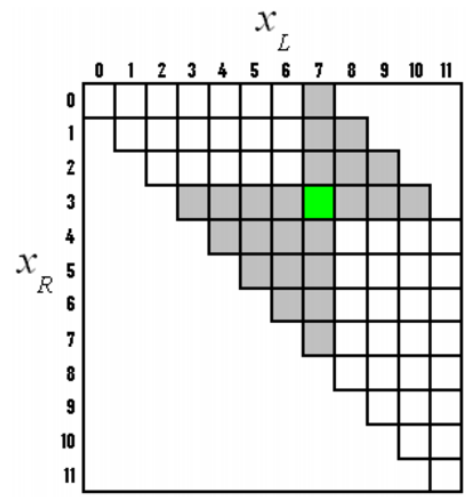
$$d = x_L - x_R \geq 0$$



maximum disparity



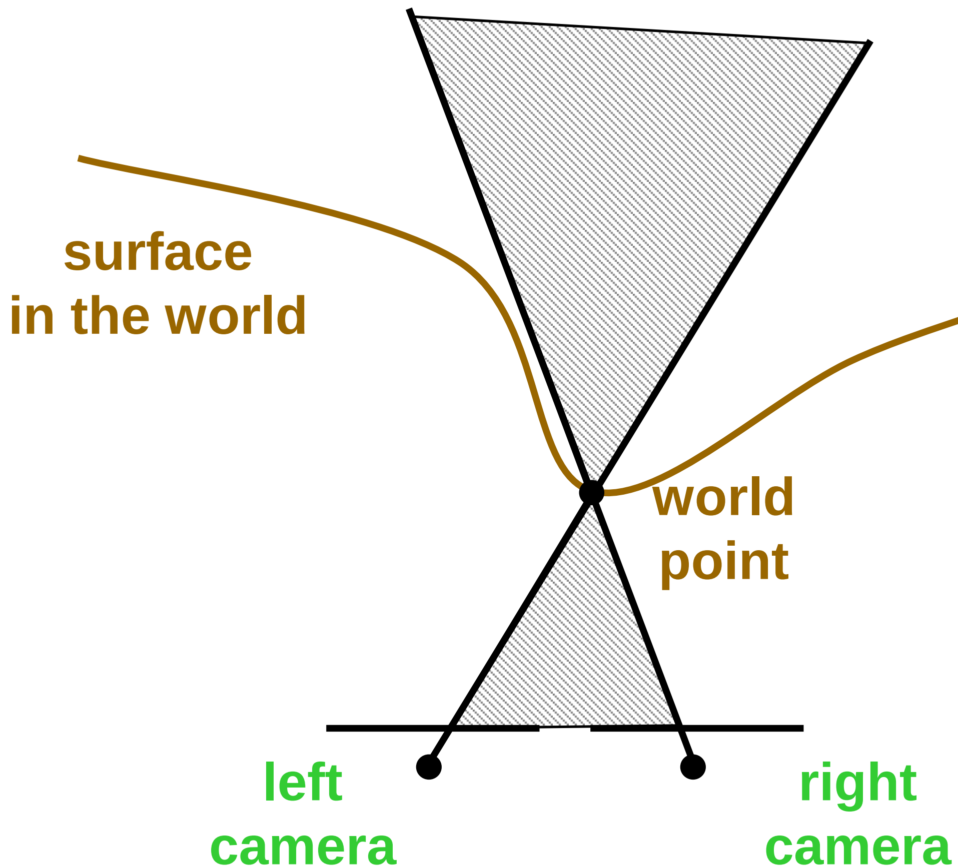
uniqueness



ordering (monotonicity)

When are these violated?

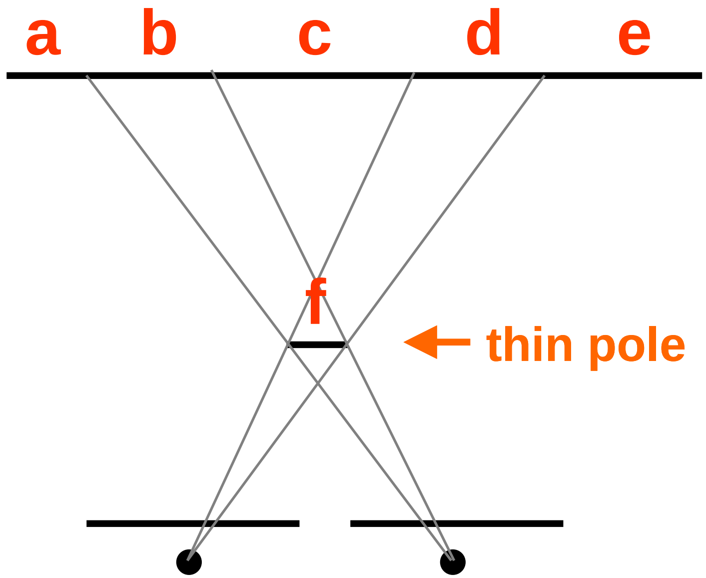
Forbidden zone

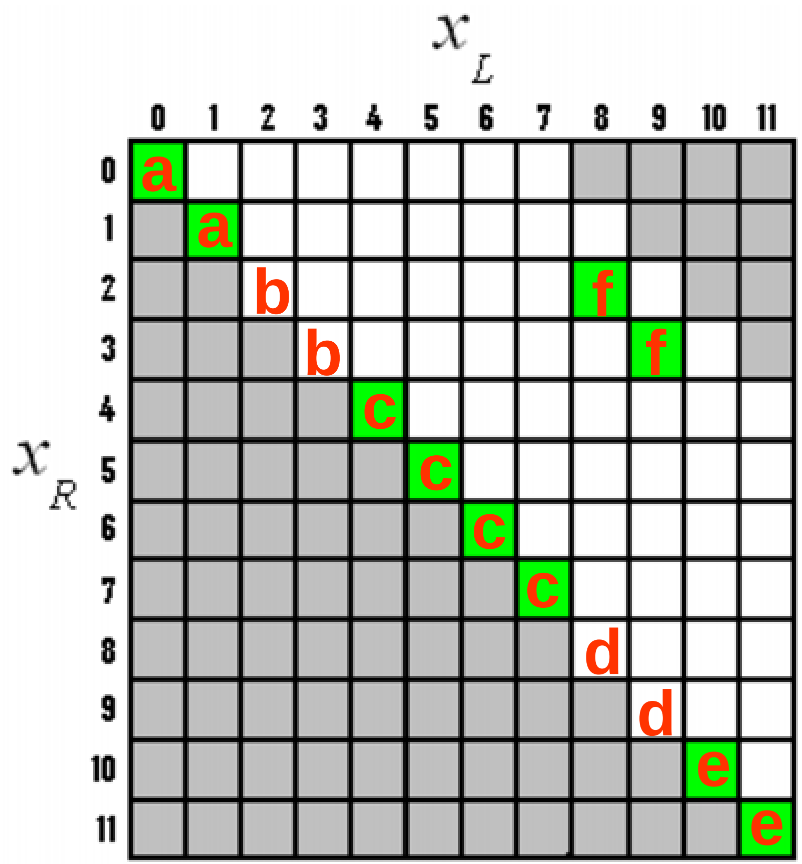


no matches
are possible in the
hourglass-shaped region
(if surface is continuous)

(Related to ordering constraint)

Violation of ordering constraint





Disparity gradient

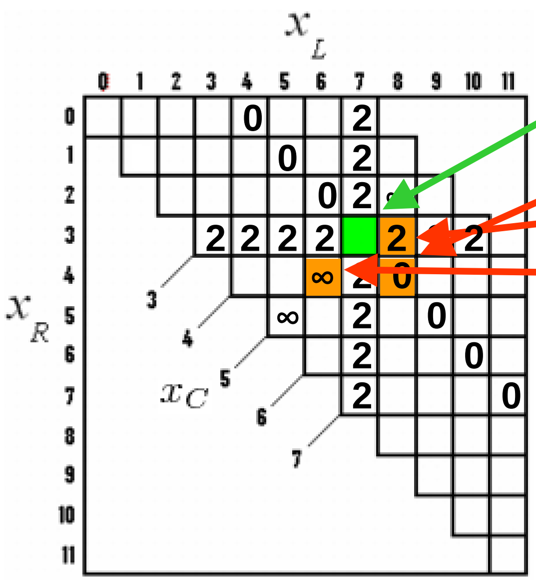
$$x_C = \frac{1}{2}(x_L + x_R)$$ ← **Cyclopean coordinate**

***x_1* in *I_L* matches *x'_1* in *I_R*:** $d_1 = x_1 - x'_1$

***x_2* in *I_L* matches *x'_2* in *I_R*:** $d_2 = x_2 - x'_2$

disparity gradient:

$$\left|\frac{\partial d}{\partial x_c}\right| = \frac{d_2 - d_1}{\frac{1}{2}(x_2 + x'_2) - \frac{1}{2}(x_1 + x'_1)} = \frac{2(d_2 - d_1)}{x_2 + x'_2 - x_1 - x'_1}$$



d_1 = 7-3 = 4 x_c = (7+3)/2 = 5

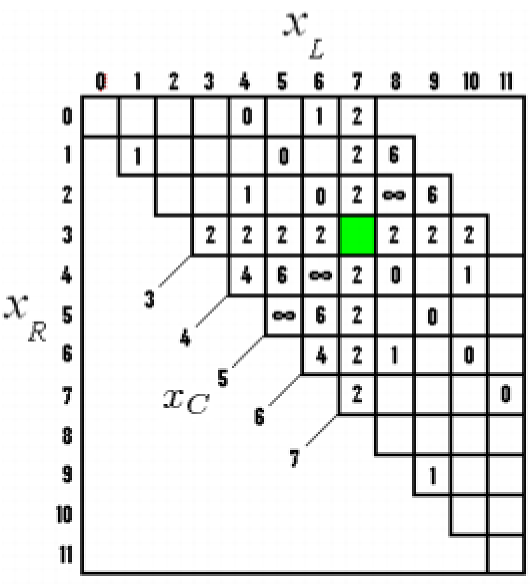
d_2 = 8-4 = 4 x_c = (8+4)/2 = 6   d.g. = 0

d_2 = 8-3 = 5 x_c = (8+3)/2 = 5.5   d.g. = 2

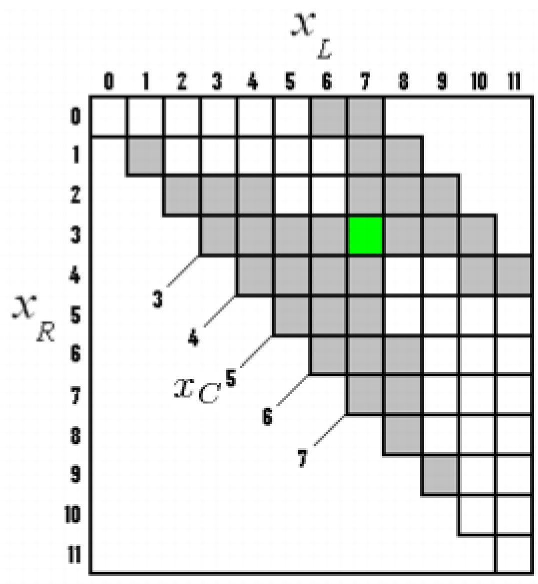
d_2 = 6-4 = 2 x_c = (6+4)/2 = 5   d.g. = ∞

…

Disparity gradient constraint

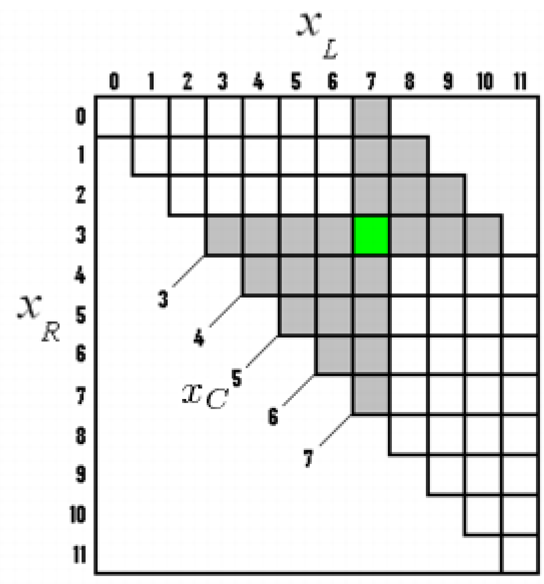


$$\left|\frac{\partial d}{\partial x_c}\right|$$



$$\left|\frac{\partial d}{\partial x_c}\right| \leq 1$$

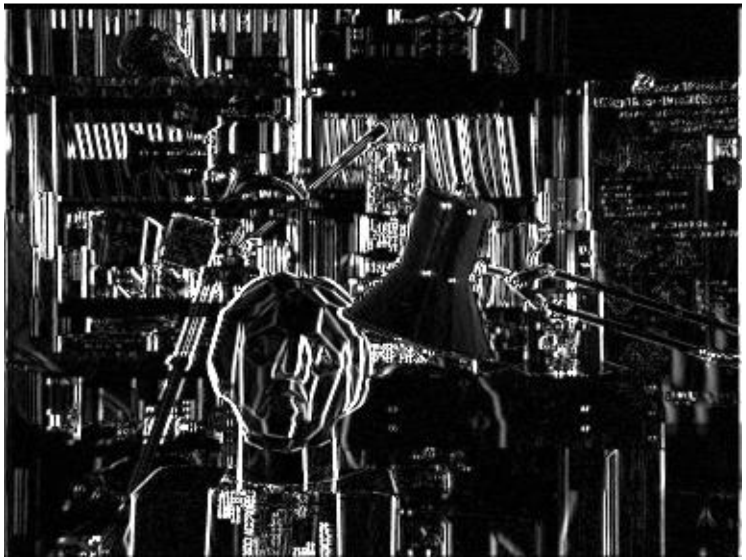
(human visual system imposes this)



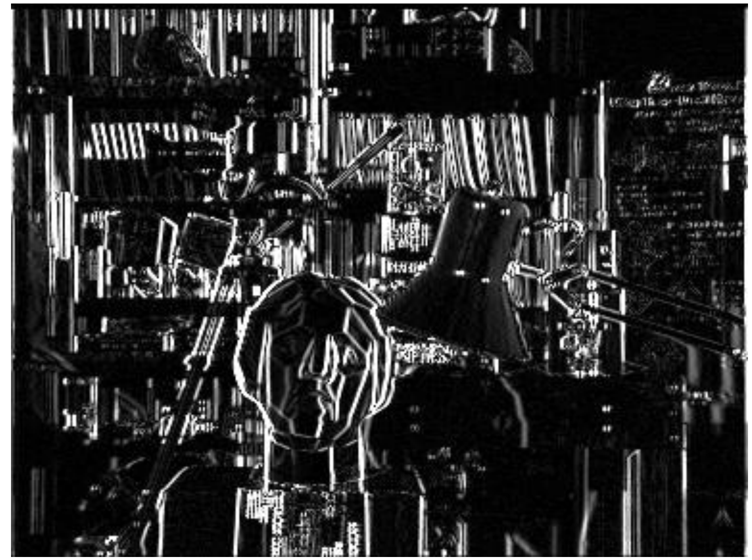
$$\left|\frac{\partial d}{\partial x_c}\right| \leq 2$$

(same as ordering constraint)

Figural continuity constraint



right

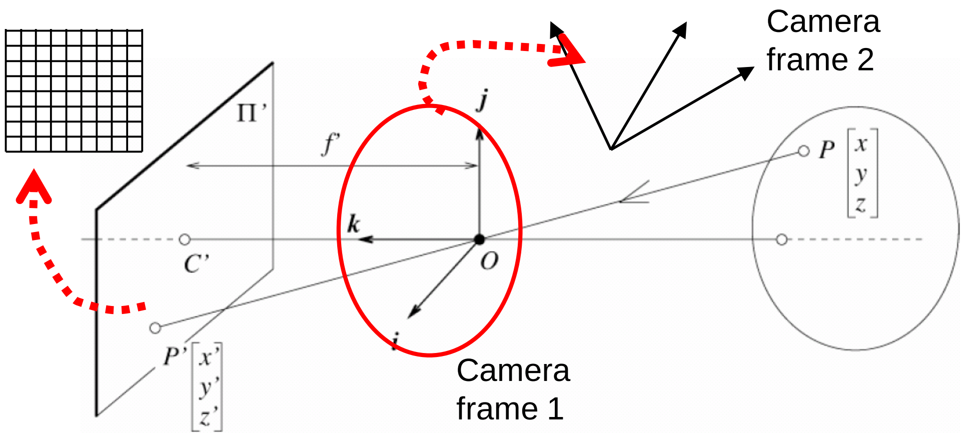


left

[University of Tsukuba]

Epipolar Geometry

Camera parameters



Extrinsic parameters:
Camera frame 1 ←→ Camera frame 2

Intrinsic parameters:
Image coordinates relative to camera ←→ Pixel coordinates

- *Extrinsic* params: rotation matrix and translation vector
- *Intrinsic* params: focal length, pixel sizes (mm), image center point, radial distortion parameters

We'll assume for now that these parameters are given and fixed.

Camera calibration

- From world coordinate to image coordinate

Viewport projection

Perspective projection

View transformation

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \Leftrightarrow \begin{vmatrix} s_x & a & u_0 \\ 0 & -s_y & v_0 \\ 0 & 0 & 1 \end{vmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0}_3^{\mathbf{T}} & 1 \end{bmatrix} \begin{bmatrix} X_S \\ Y_S \\ Z_S \\ 1 \end{bmatrix}$$

$$x = w(x_s; p)$$

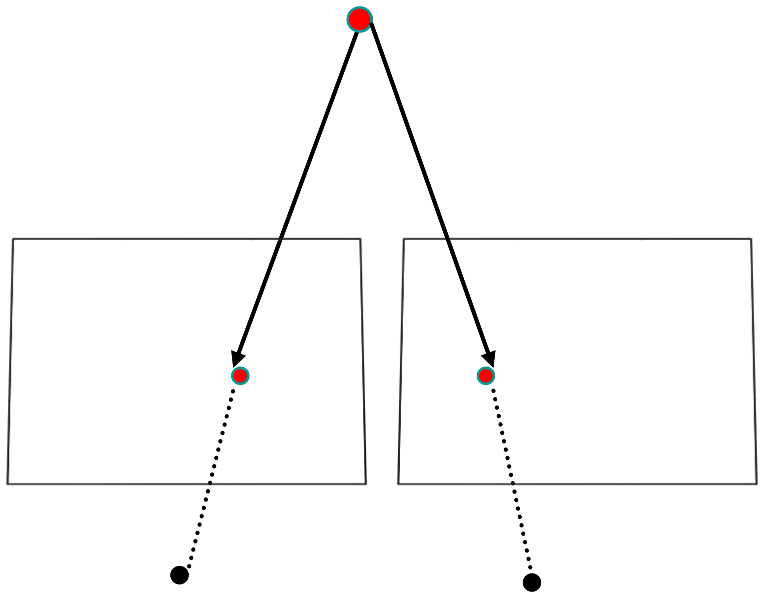
2D projections

3D points

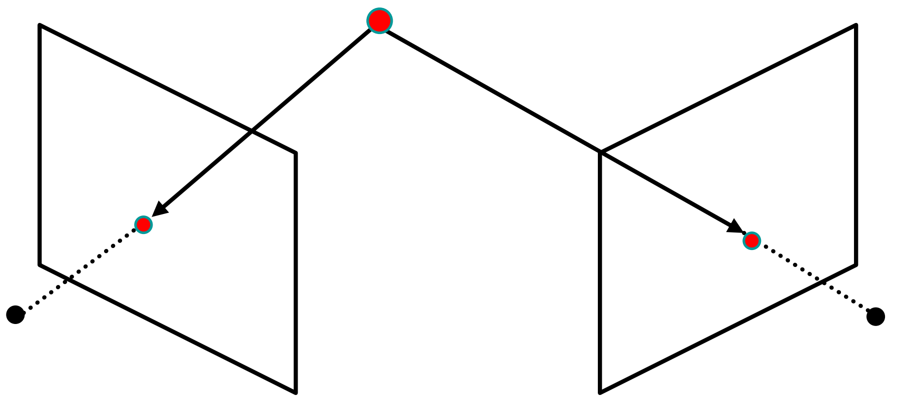
Camera parameters

General case, with calibrated cameras

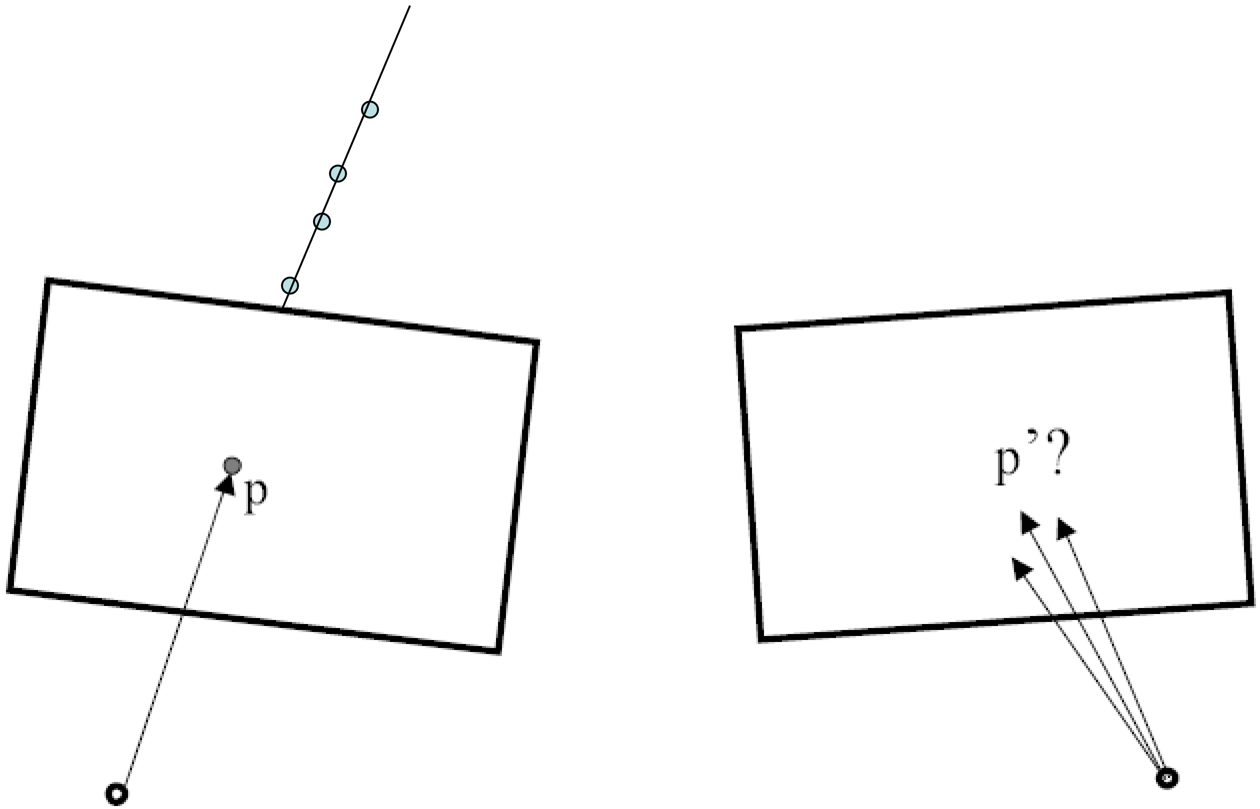
- The two cameras need not have parallel optical axes.



Vs.

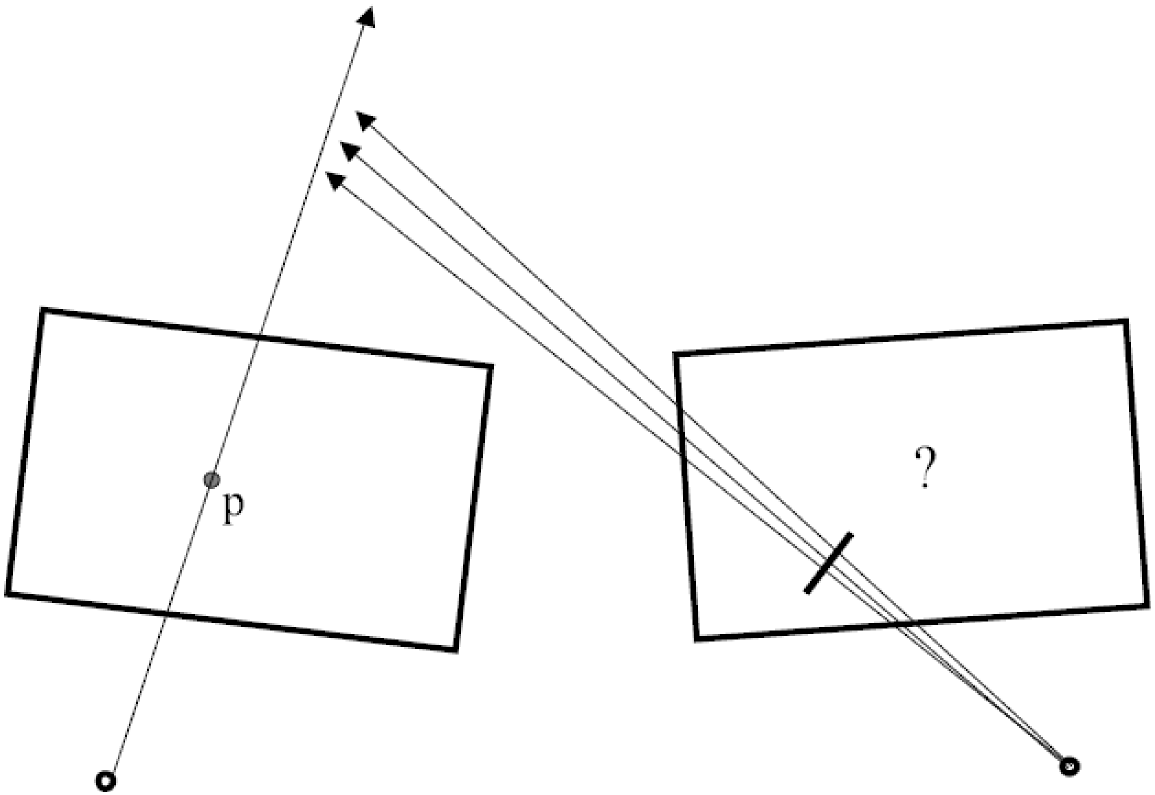


Stereo correspondence constraints

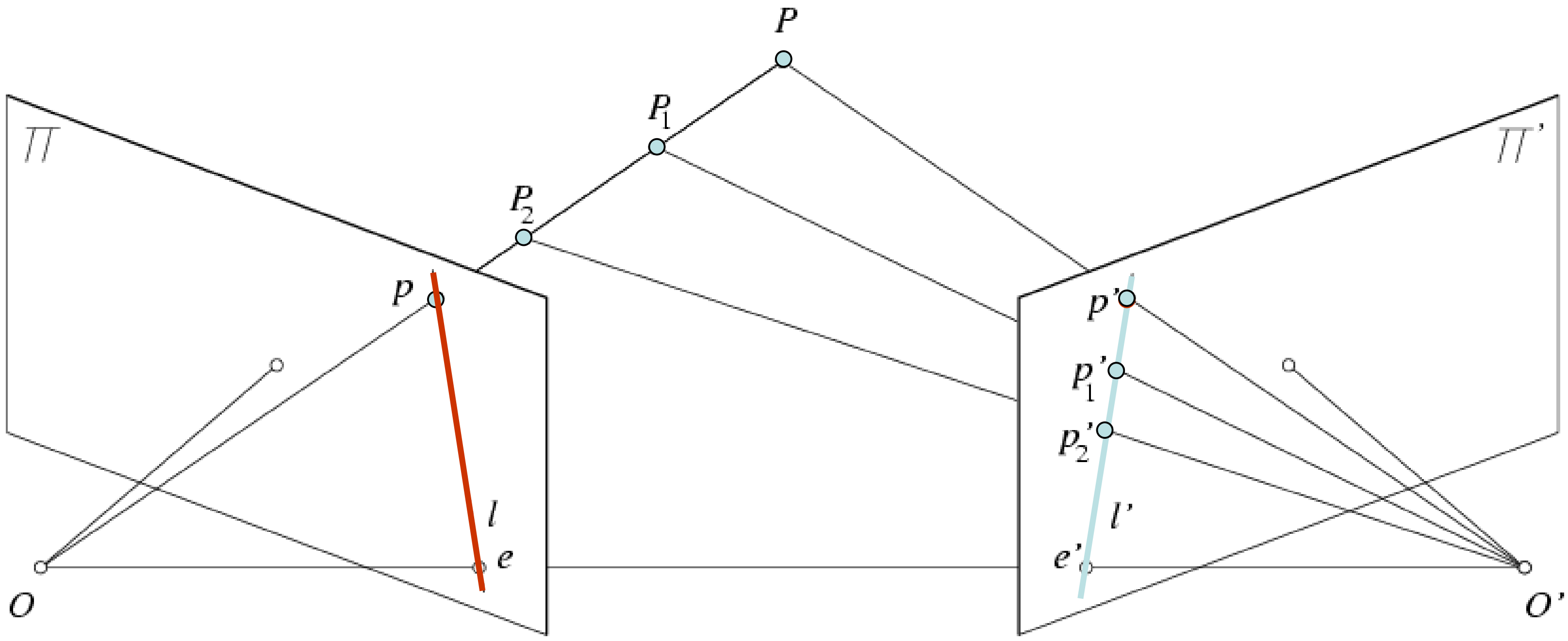


- Given p in left image, where can corresponding point p’ be?

Stereo correspondence constraints



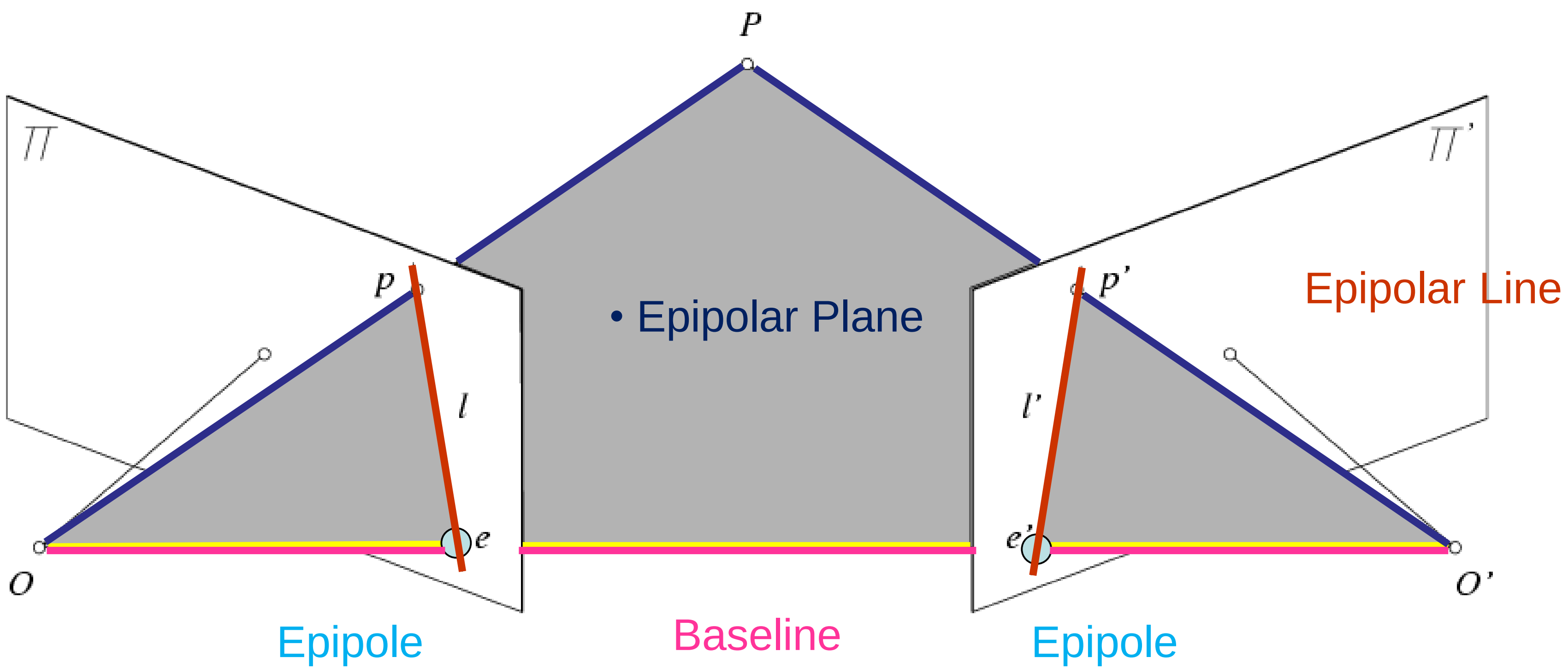
Epipolar constraint



Geometry of two views constrains where the corresponding pixel for some image point in the first view must occur in the second view.

- It must be on the line carved out by a plane connecting the world point and optical centers.

Epipolar Geometry



http://www.ai.sri.com/~luong/research/Meta3DViewer/EpipolarGeo.html

Epipolar Geometry: terms

- **Baseline**: line joining the camera centers
- **Epipole**: point of intersection of baseline with image plane
- **Epipolar plane**: plane containing baseline and world point
- **Epipolar line**: intersection of epipolar plane with the image plane

- All epipolar lines intersect at the epipole
- An epipolar plane intersects the left and right image planes in epipolar lines

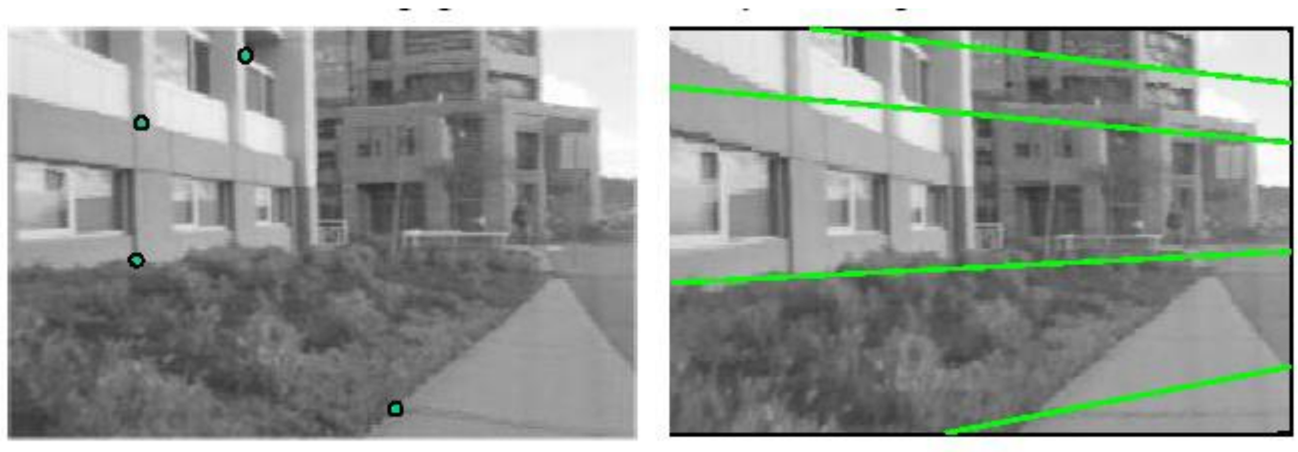
Why is the epipolar constraint useful?

Epipolar Constraint

This is useful because it reduces the correspondence problem to a 1D search along an epipolar line.

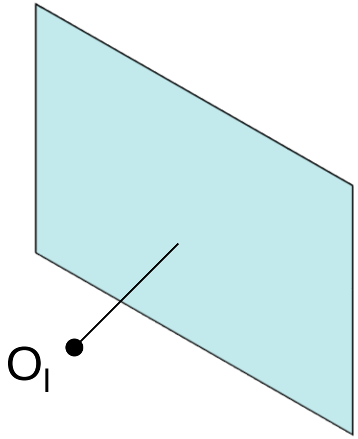
Image from Andrew Zisserman

Example

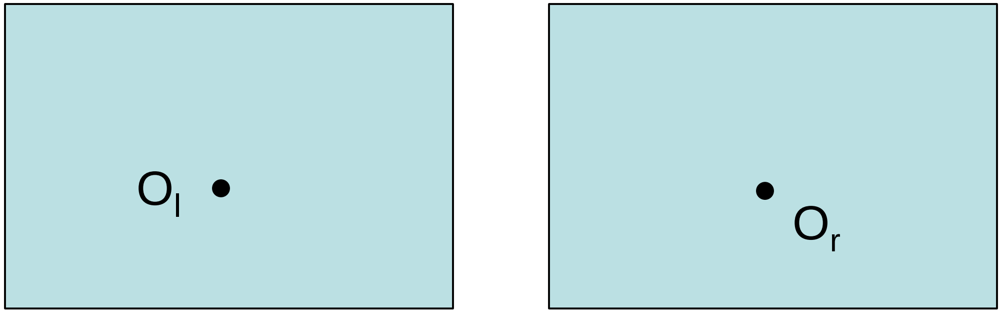


What do the epipolar lines look like?

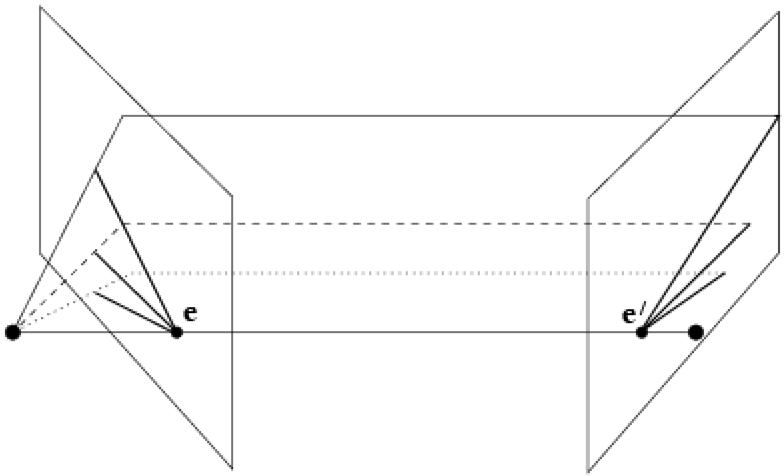
1.



2.



Example: converging cameras



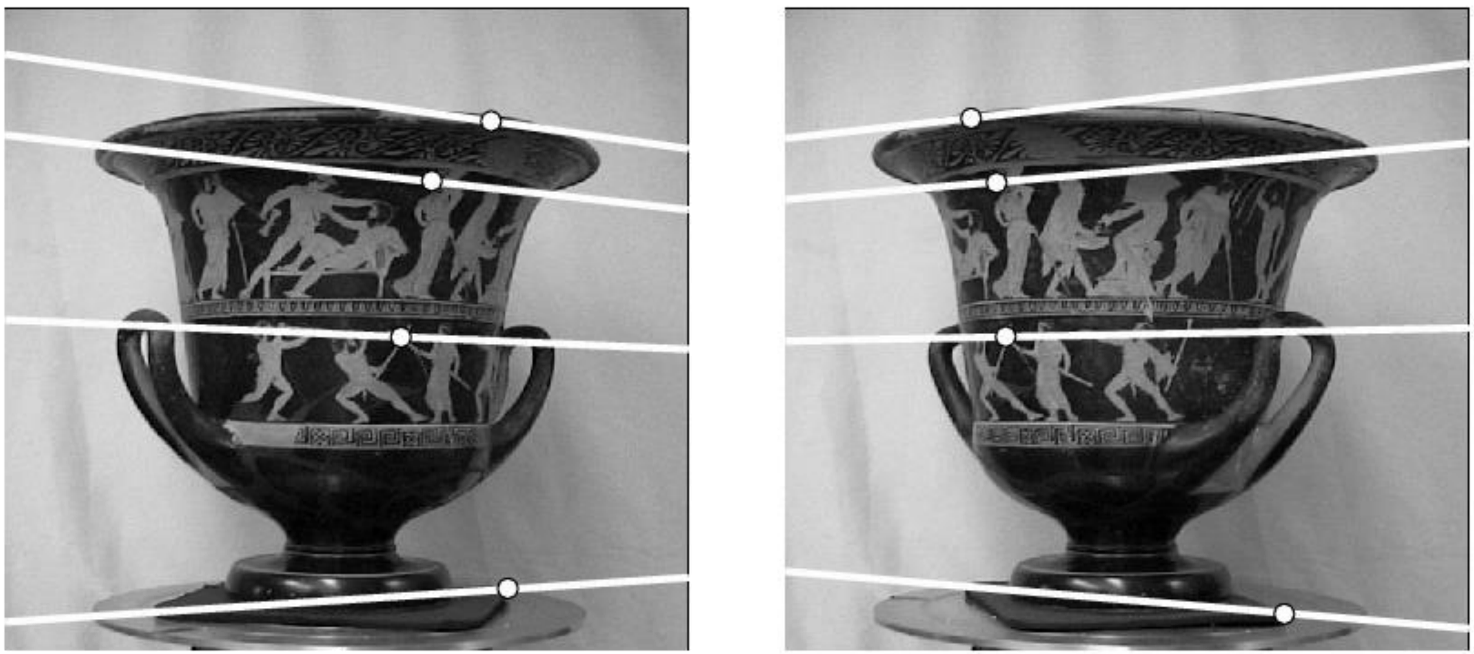
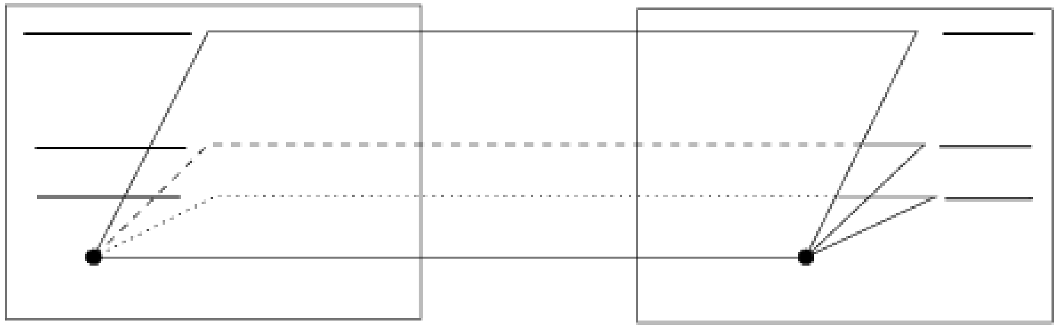


Figure from Hartley & Zisserman

Example: parallel cameras



Where are the epipoles?

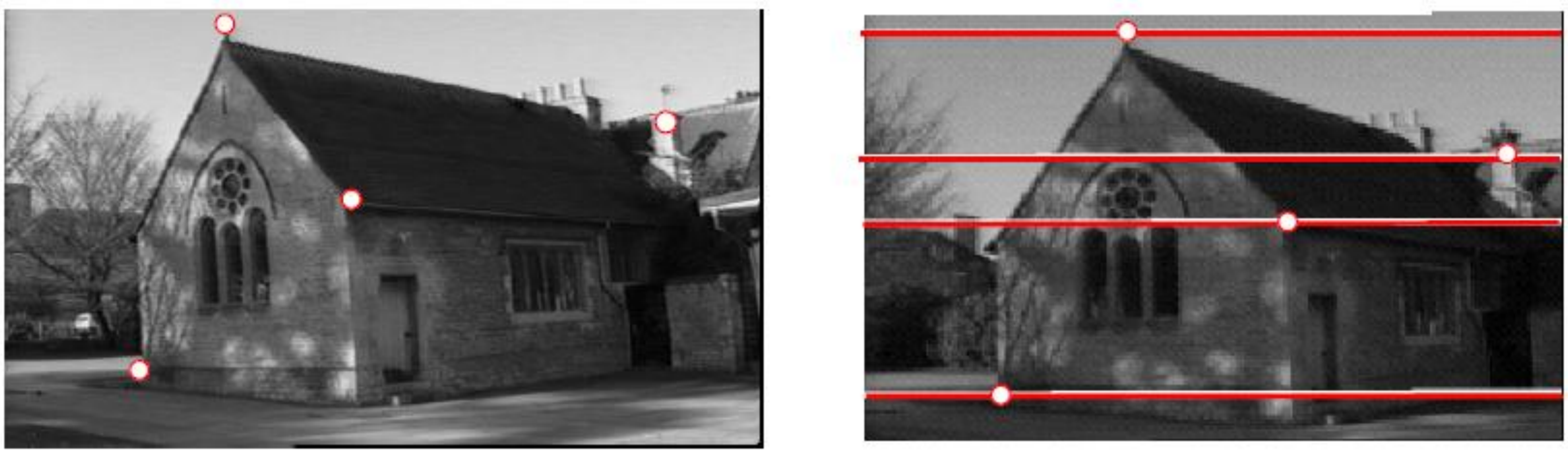
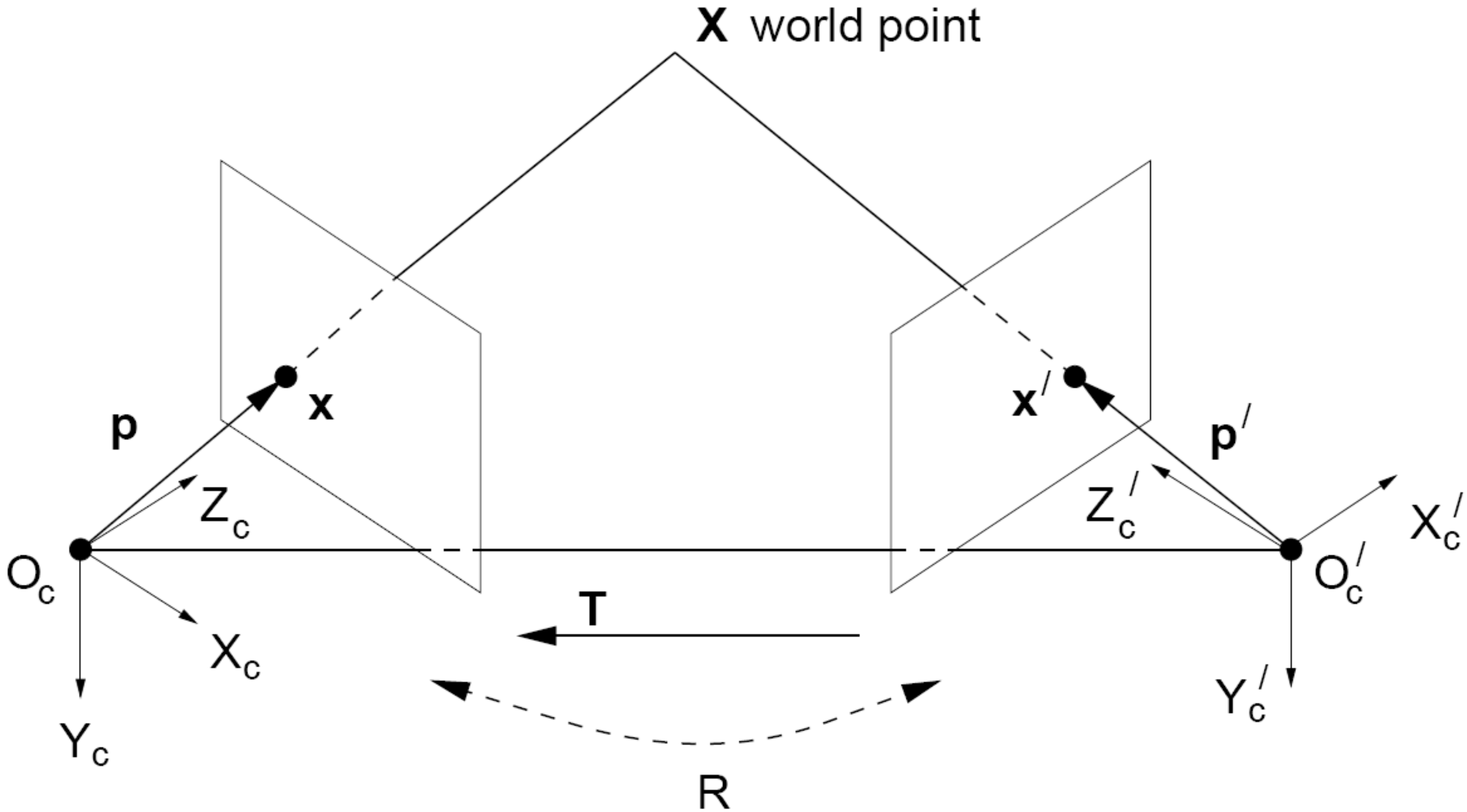


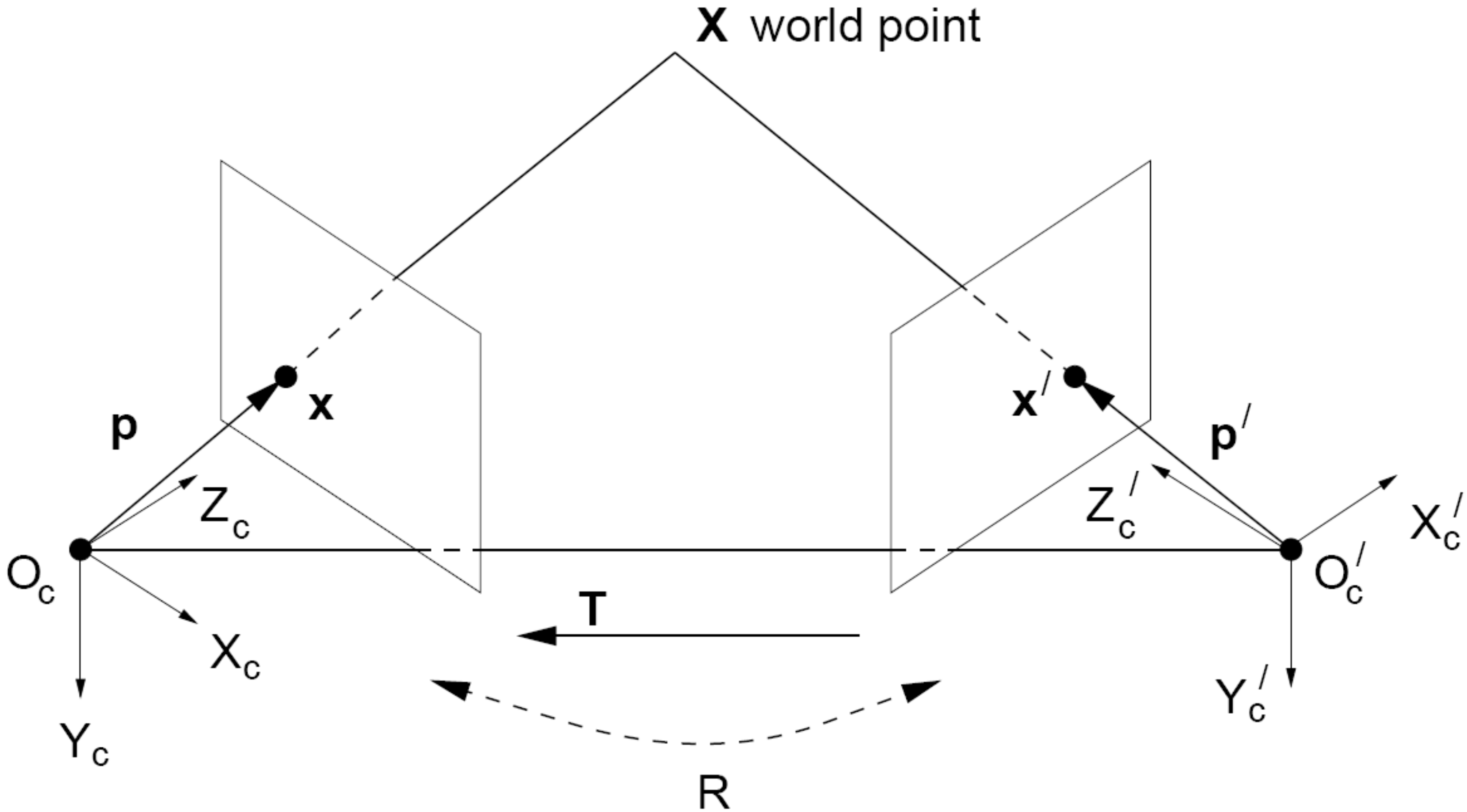
Figure from Hartley & Zisserman

Stereo geometry, with calibrated cameras



Main idea

Stereo geometry, with calibrated cameras

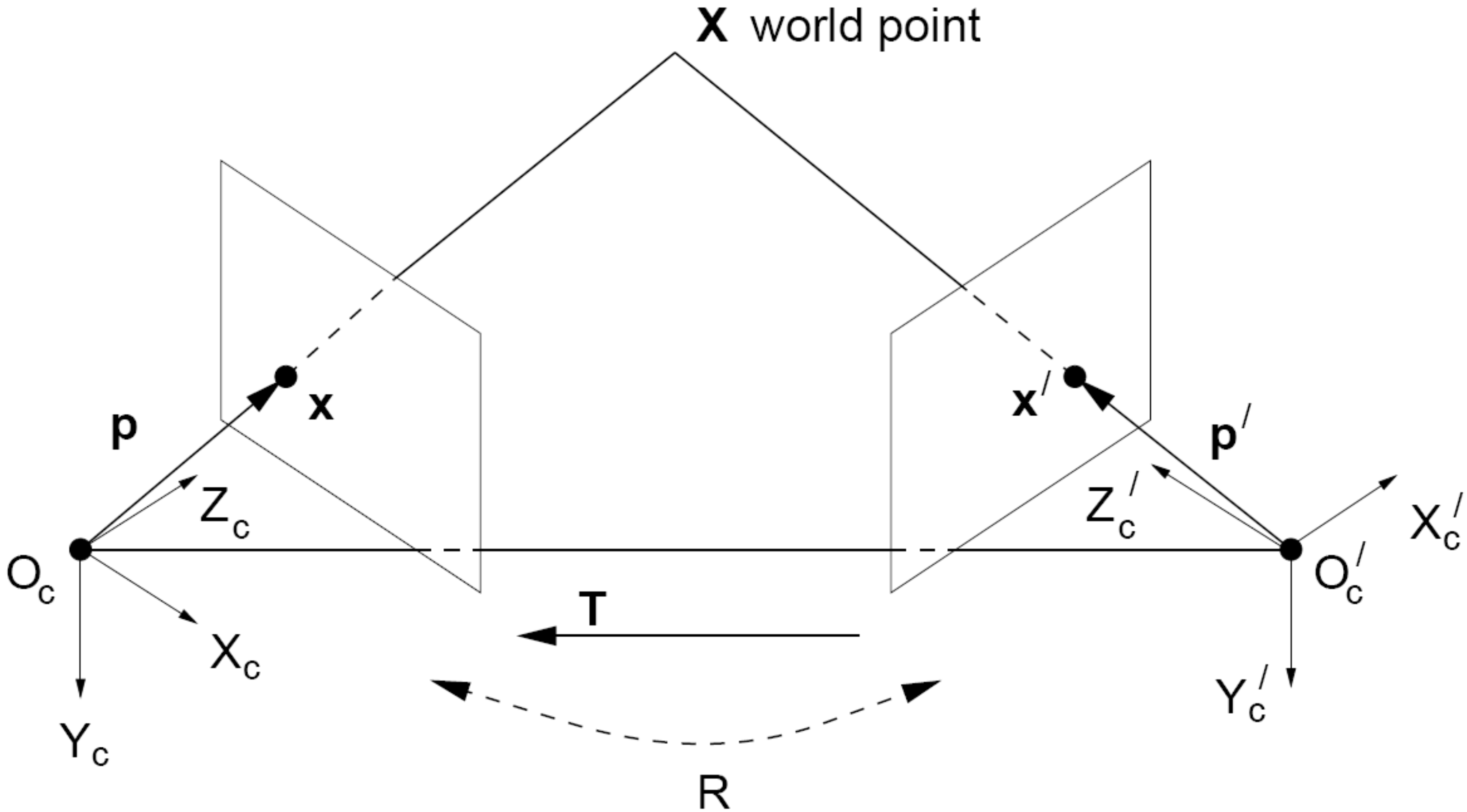


If the stereo rig is calibrated, we know :

how to **rotate** and **translate** camera reference frame 1 to get to camera reference frame 2.

Rotation: 3 x 3 matrix **R**; translation: 3 vector **T**.

Stereo geometry, with calibrated cameras



If the stereo rig is calibrated, we know :

how to **rotate** and **translate** camera reference frame 1 to get to camera reference frame 2.

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

An aside: cross product

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

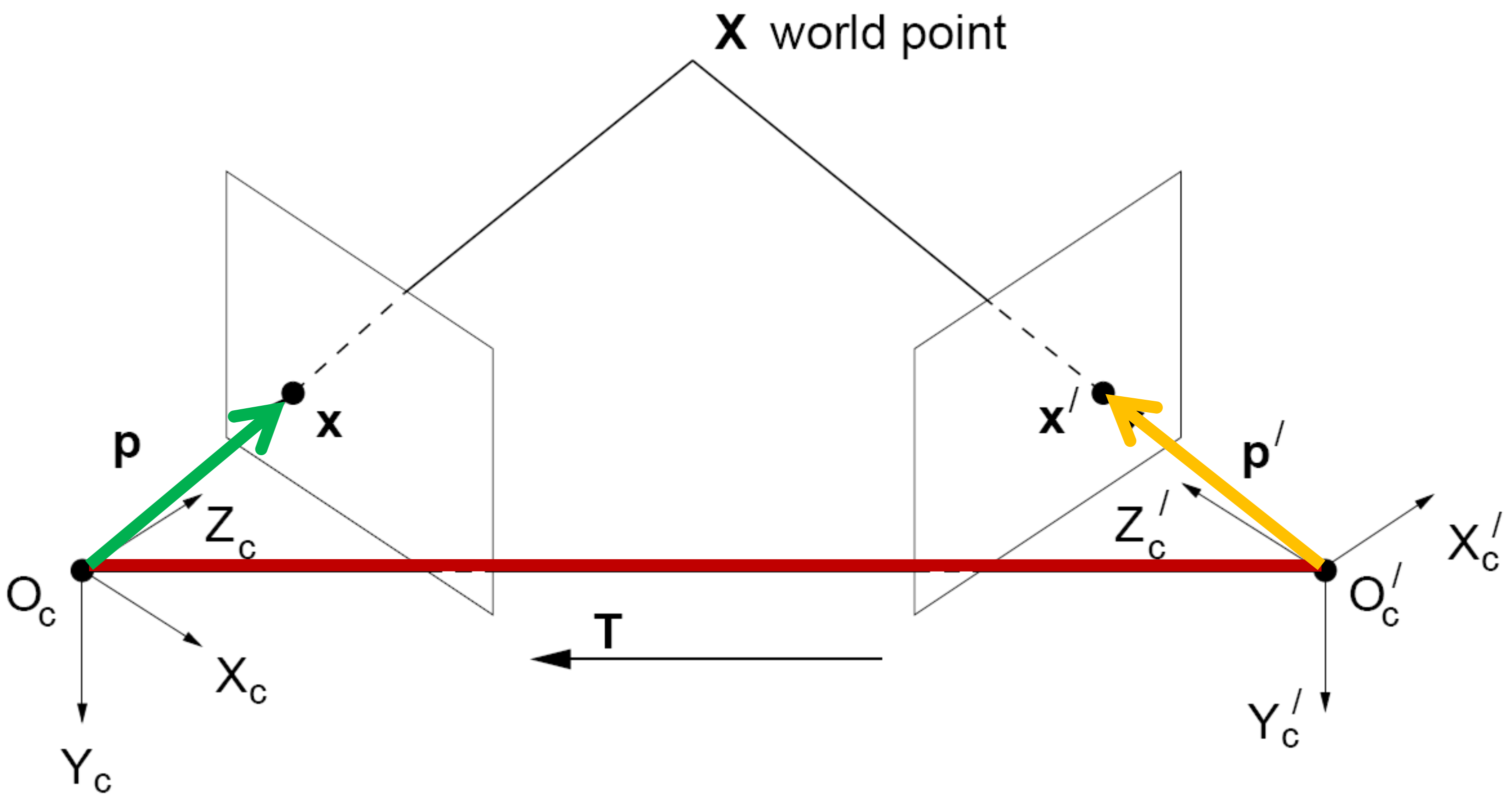
$$\vec{a} \cdot \vec{c} = 0$$

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

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.

From geometry to algebra



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

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

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

$$= 0$$

Another aside:
Matrix form of cross product

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

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

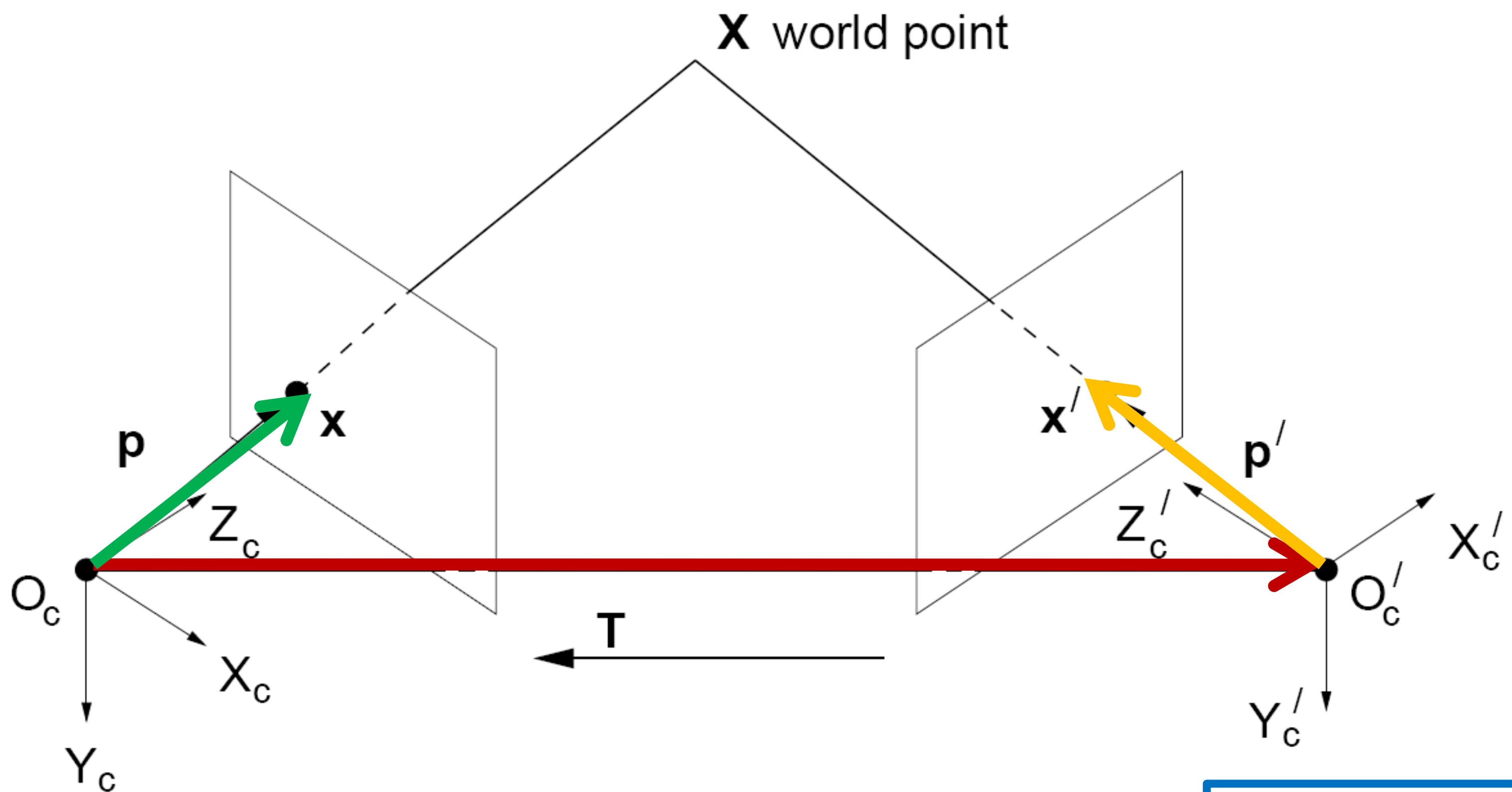
$$\vec{b} \cdot \vec{c} = 0$$

Can be expressed as a matrix multiplication.

$$[a_x] = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$$

$$\boxed{\vec{a} \times \vec{b} = [a_x]\vec{b}}$$

From geometry to algebra



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

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

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

$$\mathbf{X}' \cdot (\mathbf{T} \times \mathbf{X}') \boxed{= \mathbf{X}' \cdot (\mathbf{T} \times \mathbf{R}\mathbf{X}) \\ = 0}$$

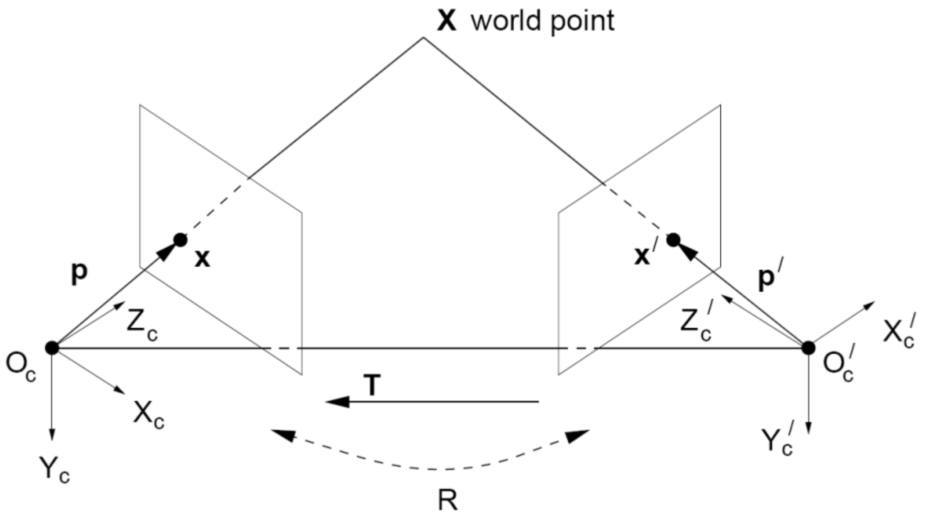
Essential matrix

$$\boxed{\mathbf{X}' \cdot (\mathbf{T} \times \mathbf{R}\mathbf{X}) = 0}$$

$$\mathbf{X}' \cdot ([\mathrm{T}_x]\mathbf{R}\mathbf{X}) = 0$$

Let $\mathbf{E} = [\mathrm{T}_x]\mathbf{R}$

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

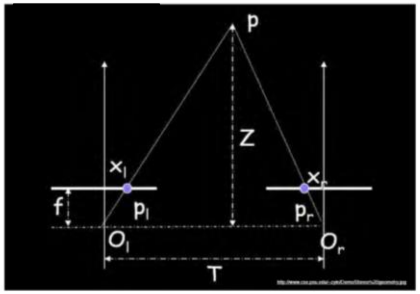


E is called the **essential matrix**, and it relates corresponding image points between both cameras, given the rotation and translation.

If we observe a point in one image, its position in other image is constrained to lie on line defined by above.

Note: these points are in **camera coordinate systems**.

Essential matrix example: parallel cameras



$$\mathbf{R} =$$

$$\mathbf{T} =$$

$$\mathbf{E} = [\mathrm{T}_{\mathbf{x}}]\mathbf{R} =$$

$$\mathbf{p} = [x, y, f]$$

$$\mathbf{p}' = [x', y', f]$$

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

For the parallel cameras, image of any point must lie on same horizontal line in each image plane.

image I(x,y)

Disparity map D(x,y)

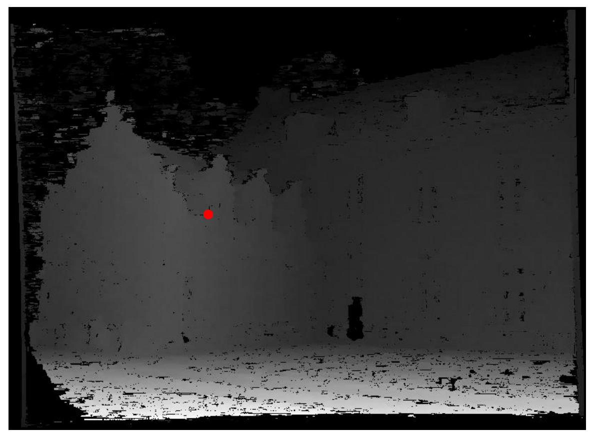


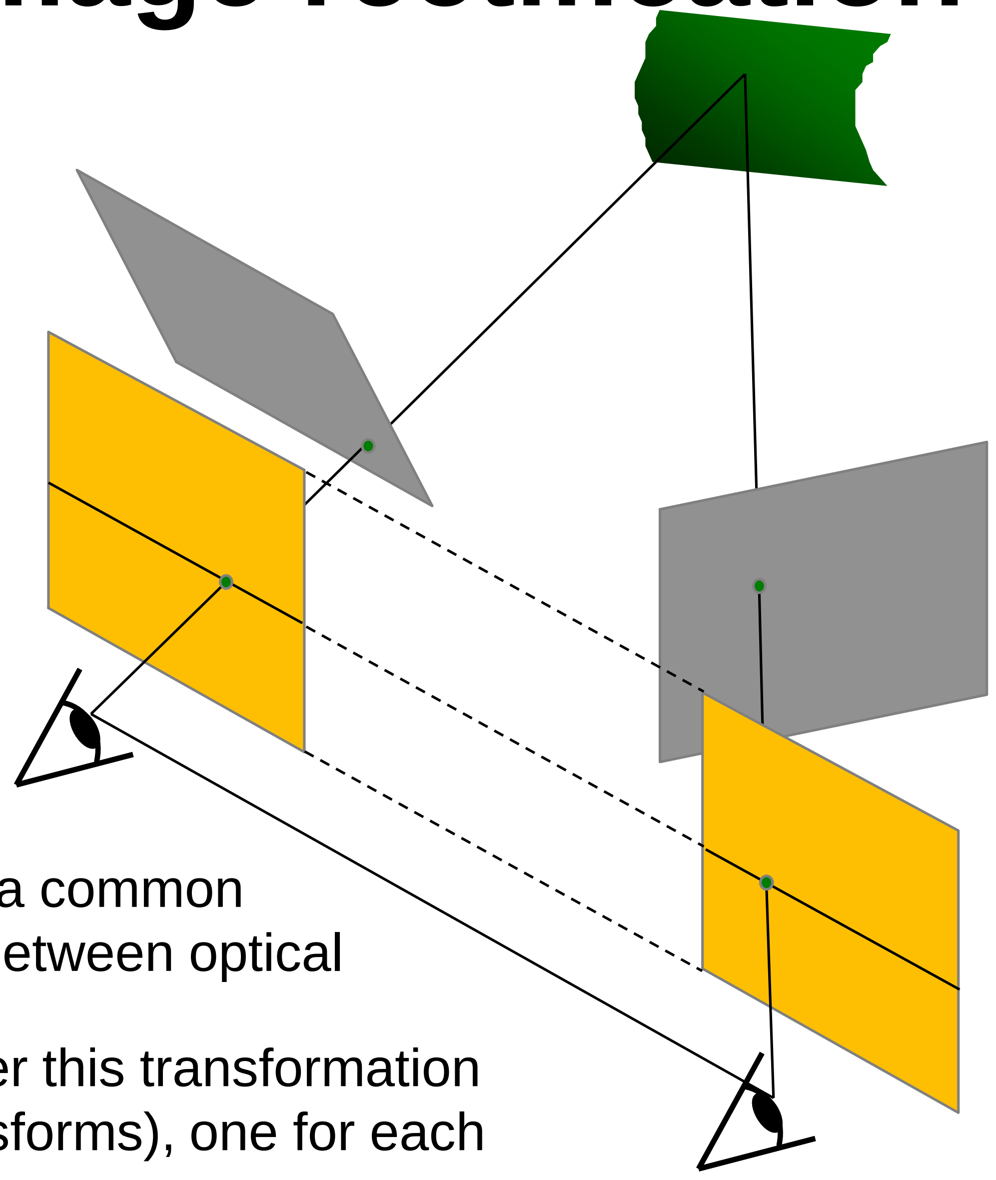
image I´(x´,y´)

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

What about when cameras' optical axes are not parallel?

Stereo image rectification

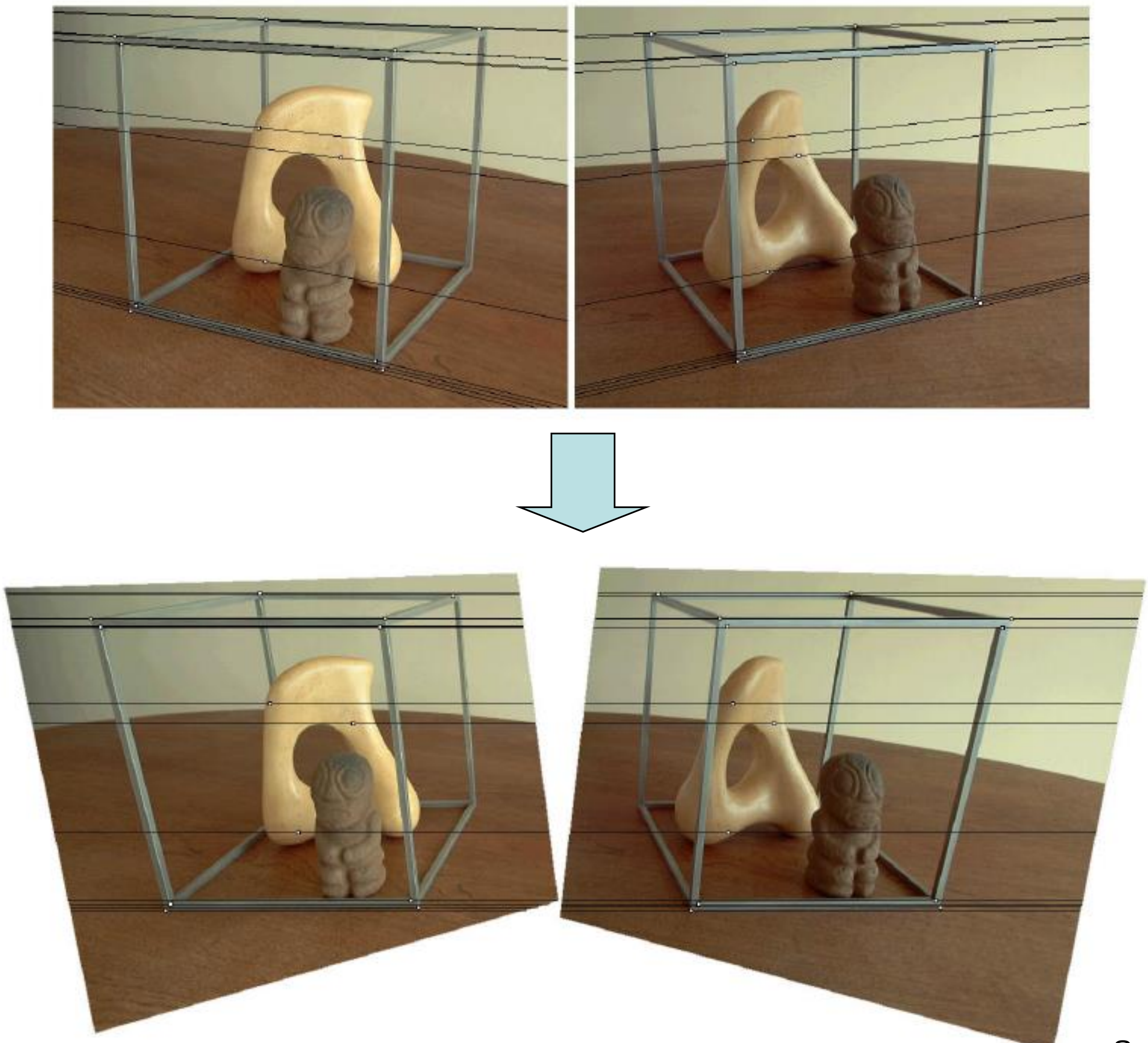
In practice, it is convenient if image scanlines (rows) are the epipolar lines.



- reproject image planes onto a common plane parallel to the line between optical centers
- pixel motion is horizontal after this transformation
- two homographies (3x3 transforms), one for each input image reprojection

Slide credit: Li Zhang

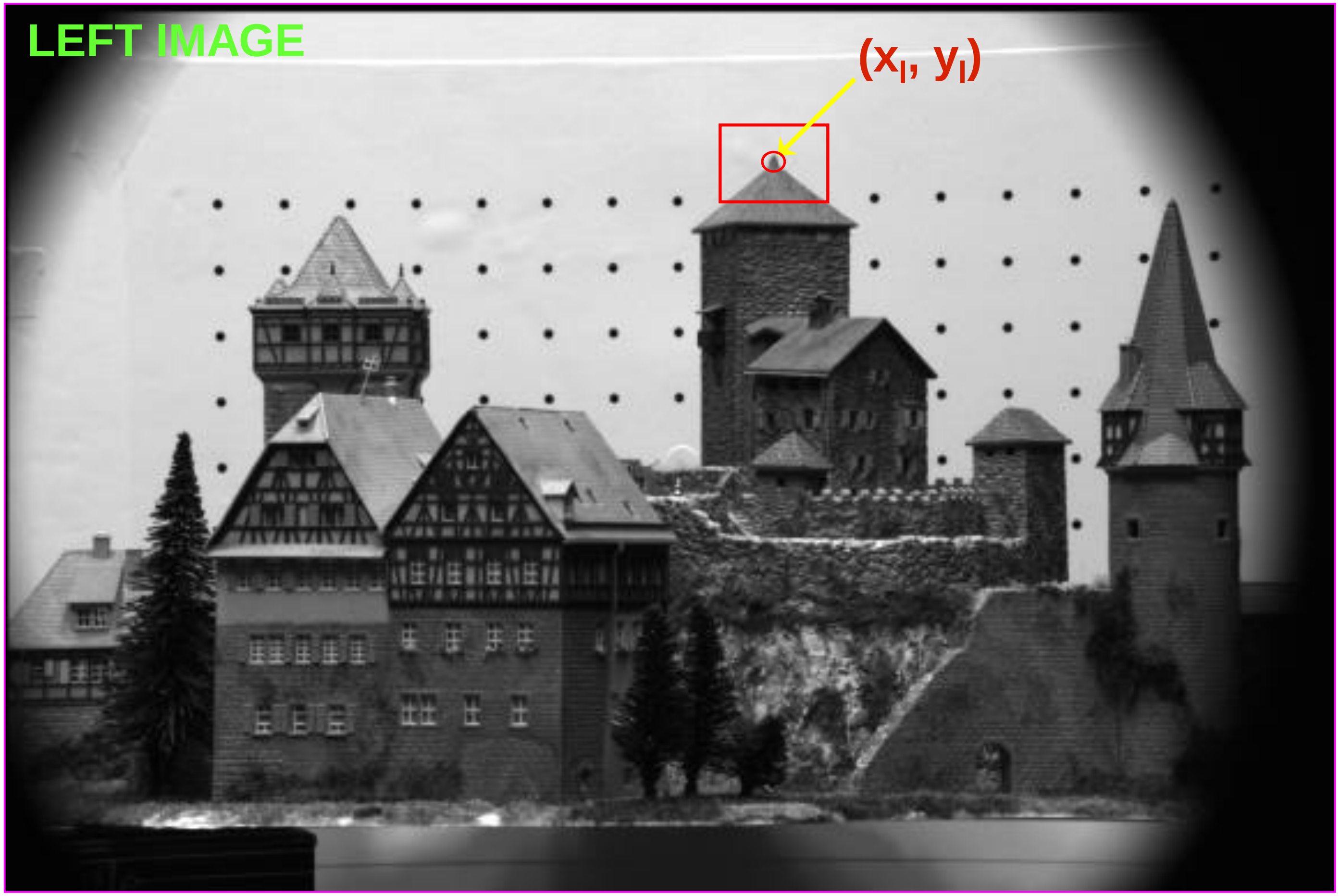
Stereo image rectification:



Source: Alyosha Efros

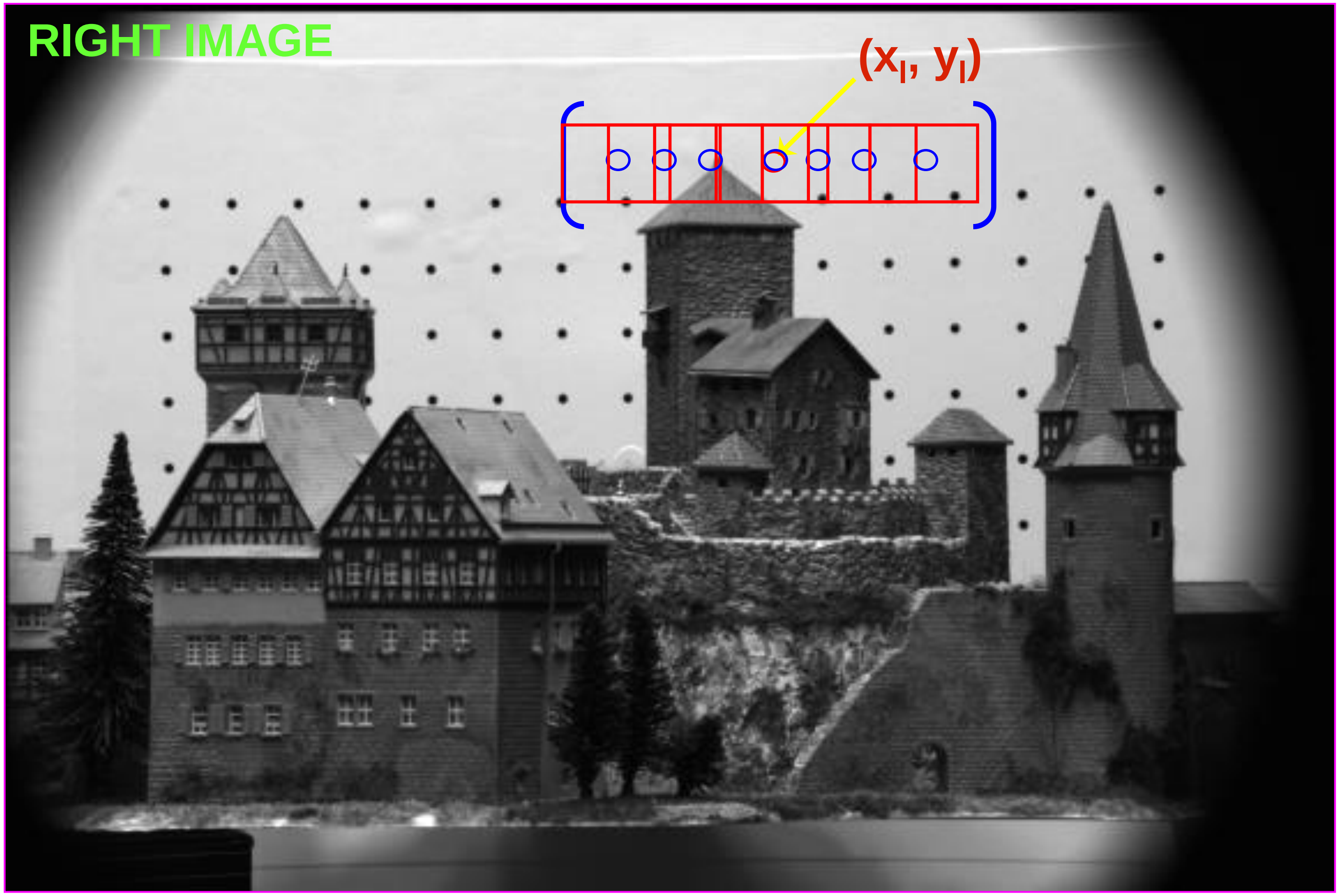
Feature-Based Matching

Correlation Approach



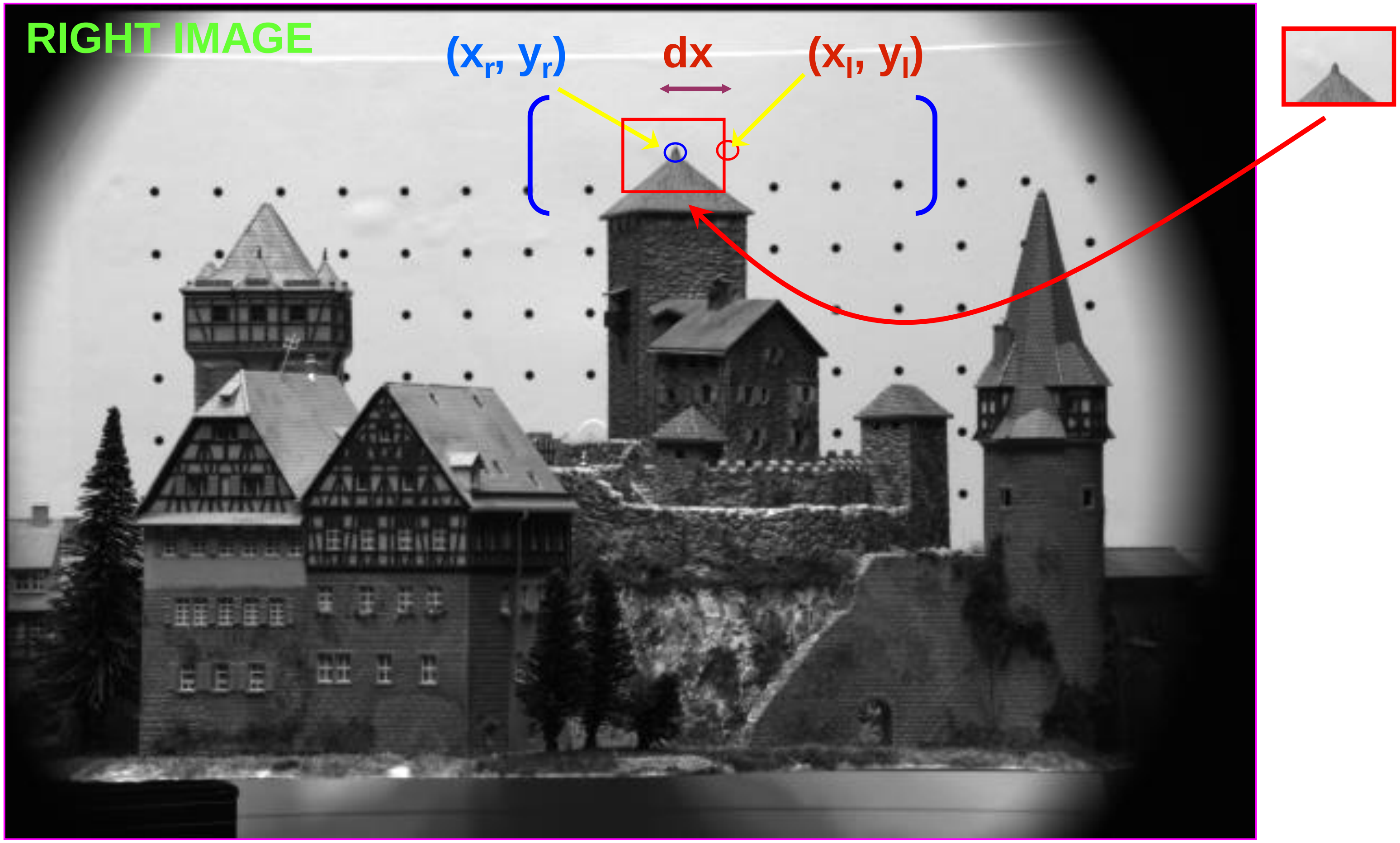
- For Each point (x_l, y_l) in the left image, define a window centered at the point

Correlation Approach



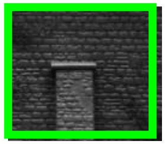
- … search its corresponding point within a search region in the right image

Correlation Approach



- … the disparity (dx, dy) is the displacement when the correlation is maximum

Comparing Windows

 $\stackrel{?}{=}$ 

f g

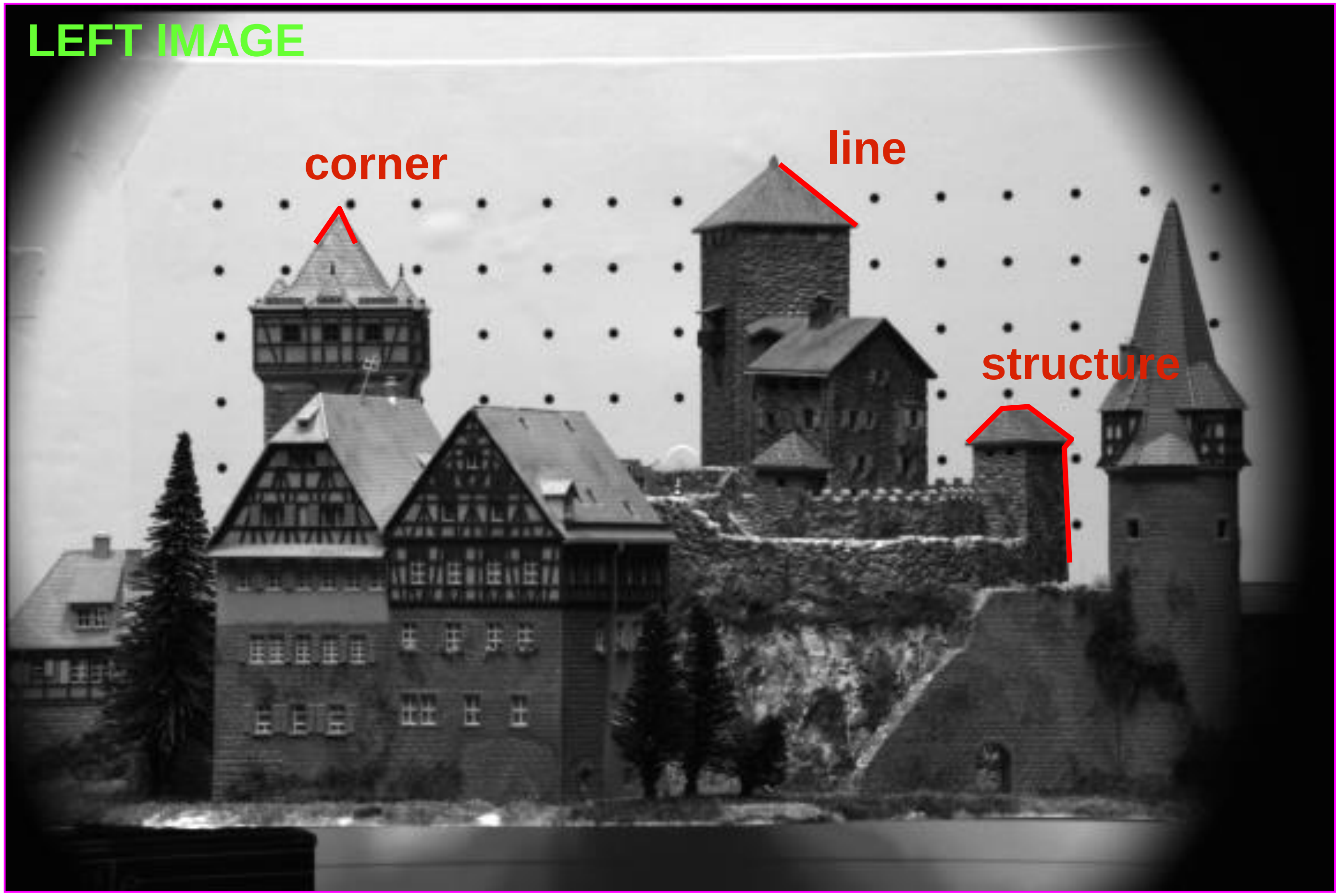
Minimize $$\sum_{[i,j]\in R} (f(i,j) - g(i,j))^2$$ Sum of Squared Differences

Maximize $$C_{fg} = \sum_{[i,j]\in R} f(i,j)g(i,j)$$ Cross correlation

Feature-based correspondence

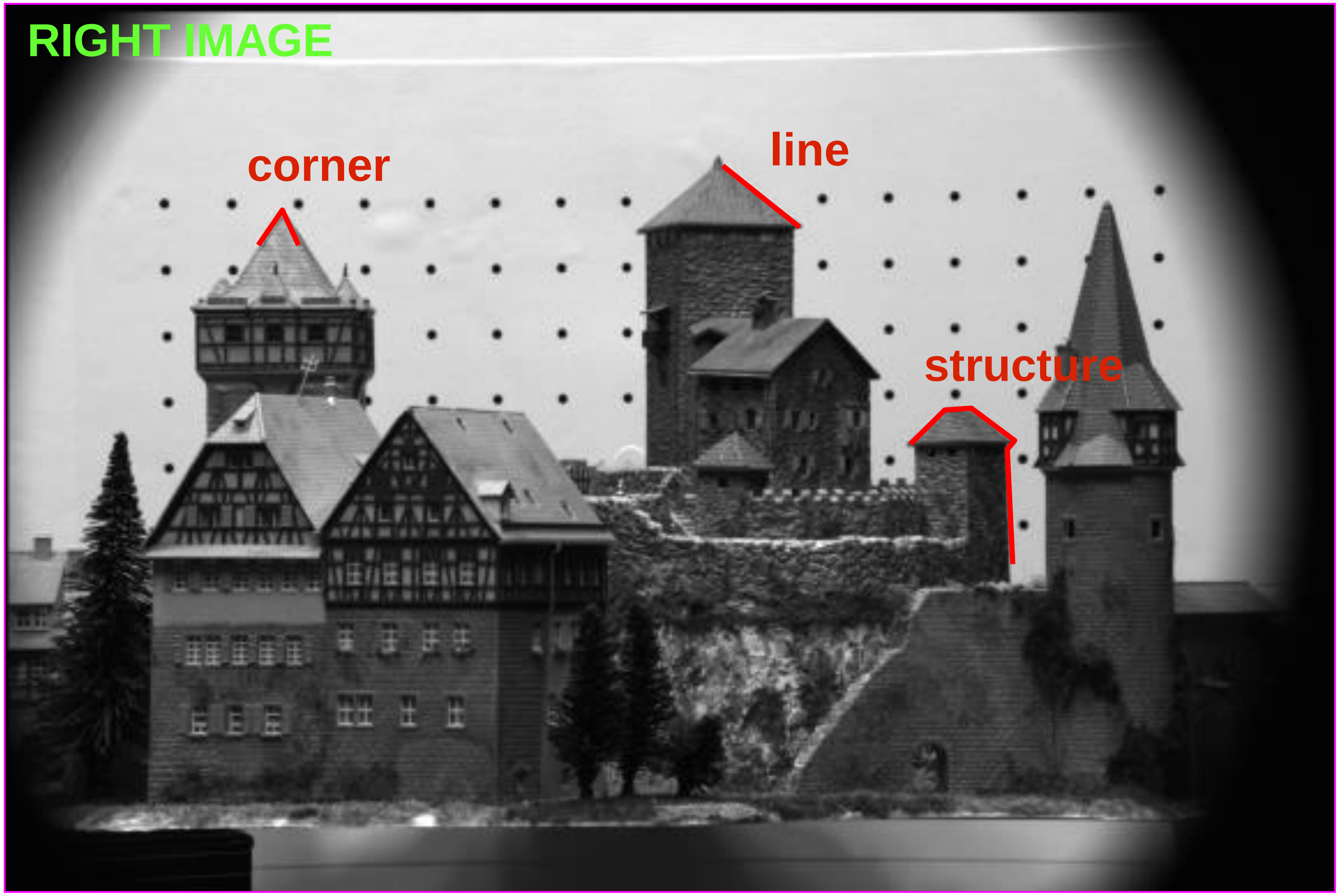
- Features most commonly used:
  - Corners
    - Similarity measured in terms of:
      - surrounding gray values (SSD, Cross-correlation)
      - location
  - Edges, Lines
    - Similarity measured in terms of:
      - orientation
      - contrast
      - coordinates of edge or line’s midpoint
      - length of line

Feature-based Approach



- For each feature in the left image…

Feature-based Approach



- Search in the right image… the disparity (dx, dy) is the displacement when the similarity measure is maximum

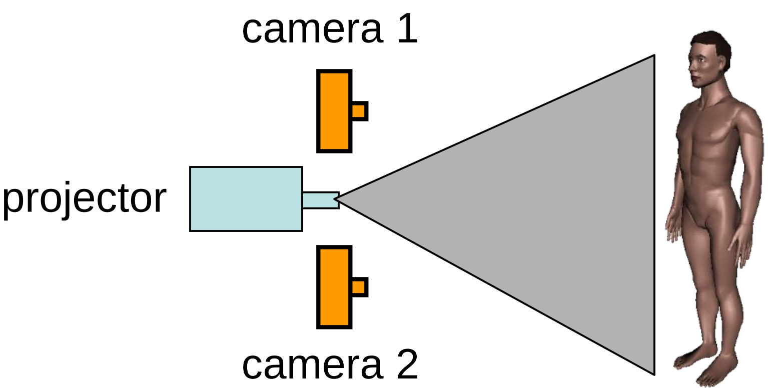
Correspondence Difficulties

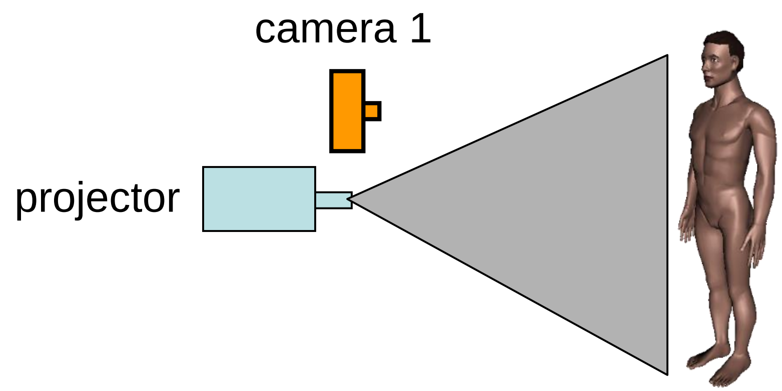
- Why is the correspondence problem difficult?
  - Some points in each image will have no corresponding points in the other image.
    - (1) the cameras might have different fields of view.
    - (2) due to occlusion.
- A stereo system must be able to determine the image parts that should not be matched.

Structure Light

Active stereo with structured light

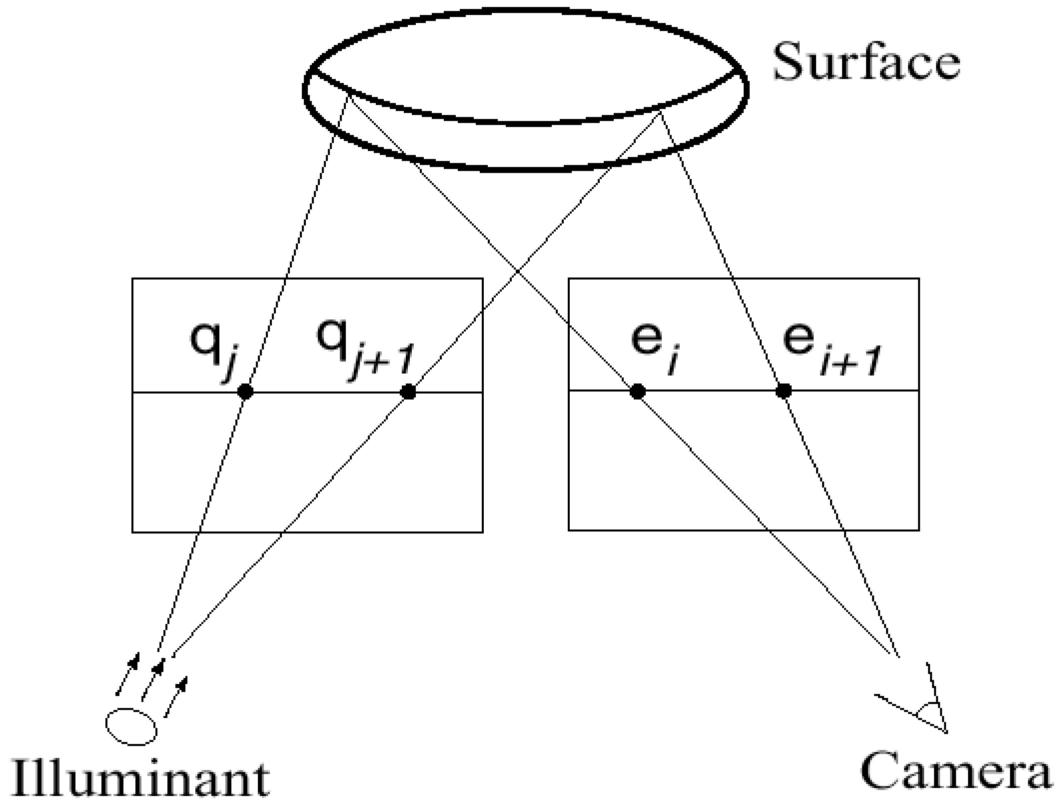
Li Zhang's one-shot stereo



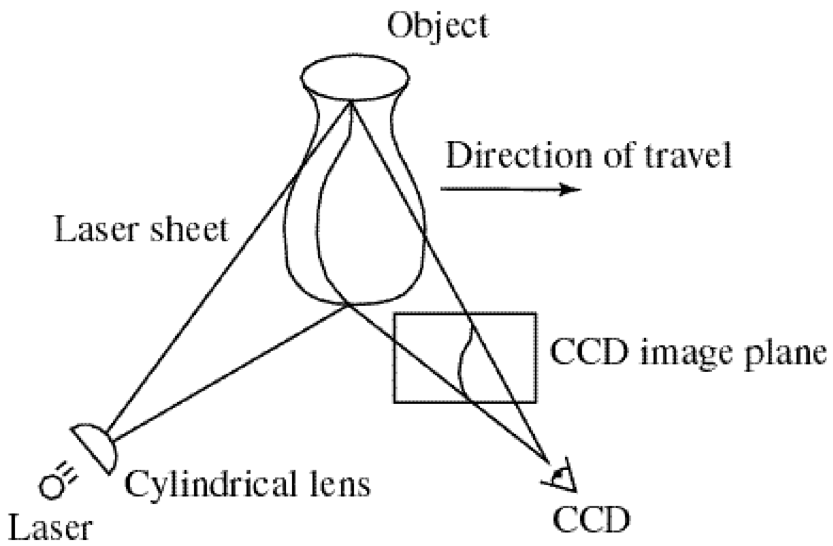


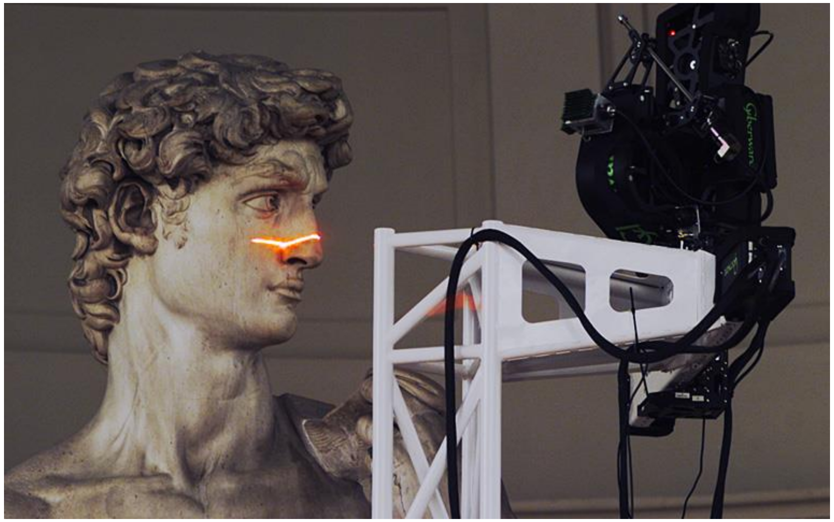
- Project "structured" light patterns onto the object
  - simplifies the correspondence problem

Active stereo with structured light



Laser scanning

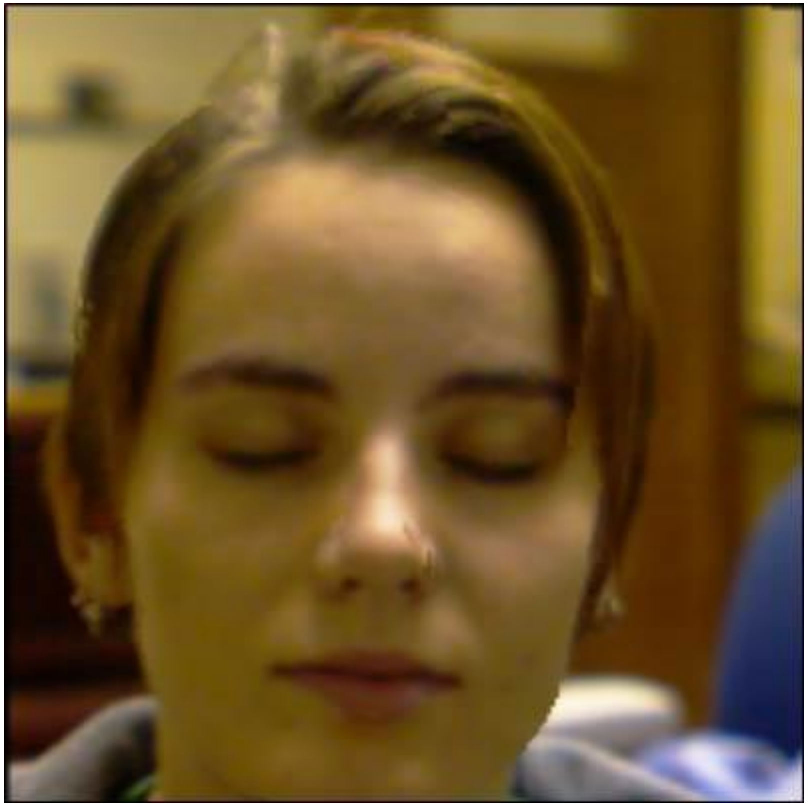




Digital Michelangelo Project
http://graphics.stanford.edu/projects/mich/

- Optical triangulation
  - Project a single stripe of laser light
  - Scan it across the surface of the object
  - This is a very precise version of structured light scanning

Portable 3D laser scanner (this one by Minolta)



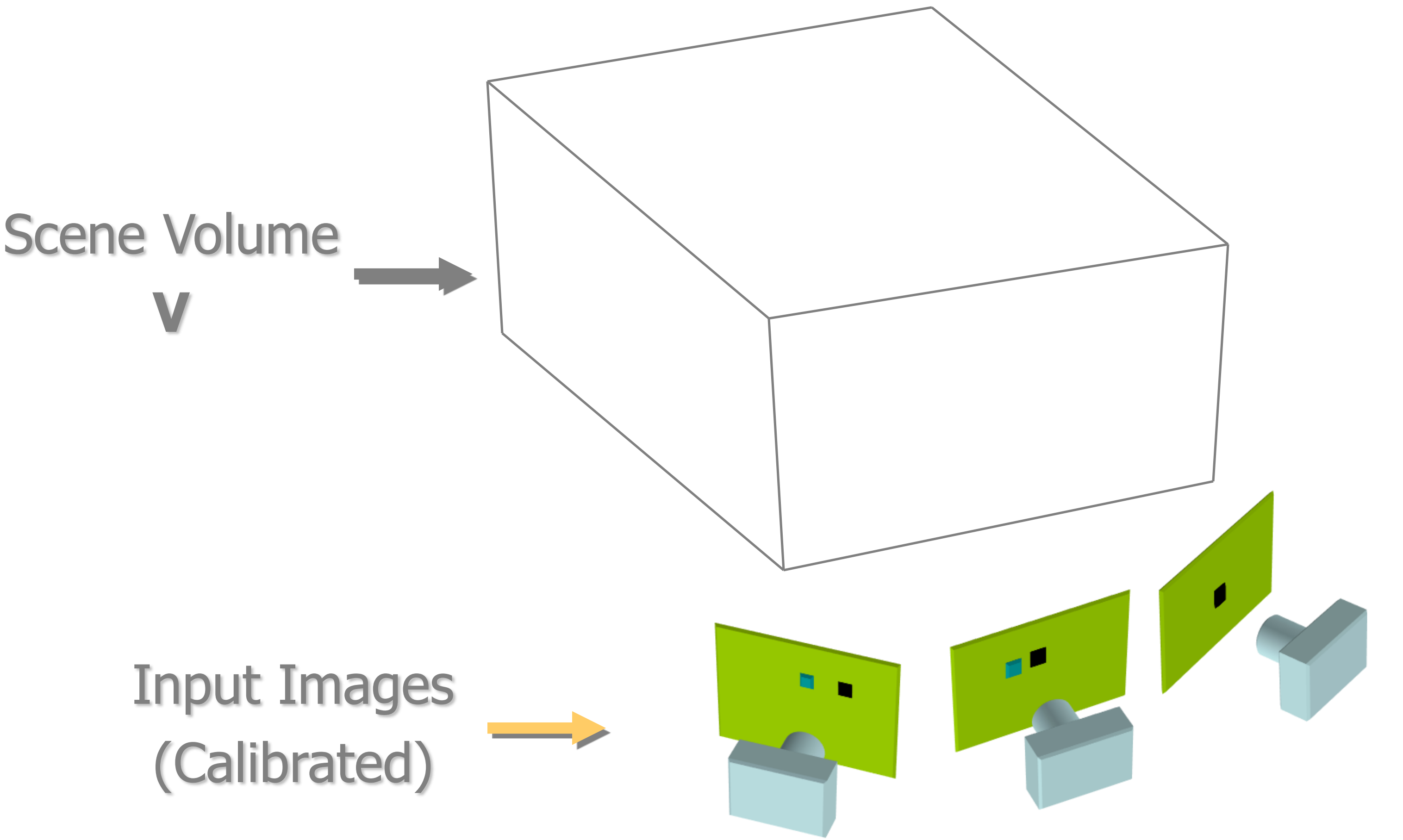
Laser scanned models

The Digital Michelangelo Project, Levoy et al.

Laser scanned models

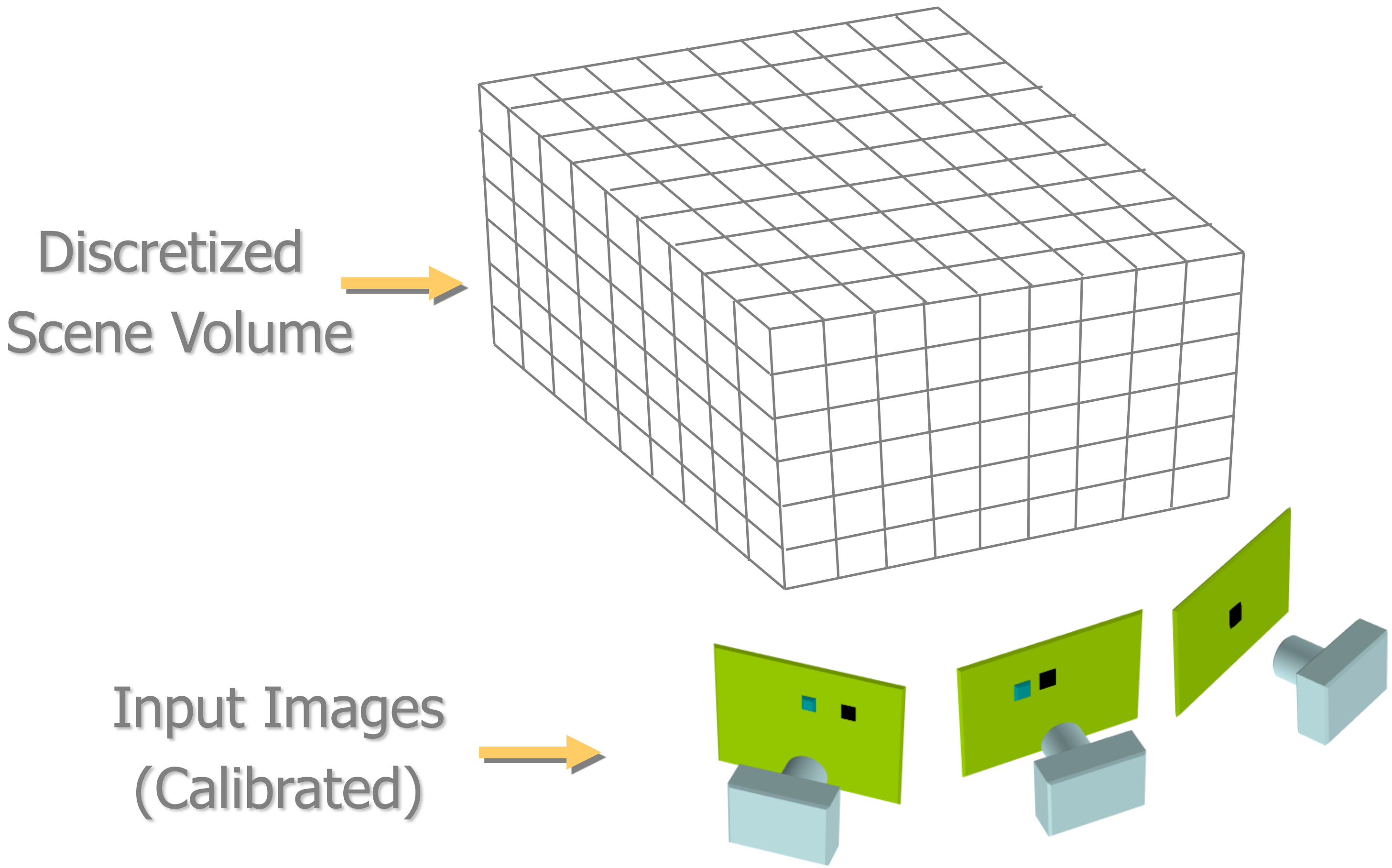
The Digital Michelangelo Project, Levoy et al.

Volumetric Stereo



Goal: **Determine transparency, radiance of points in V**

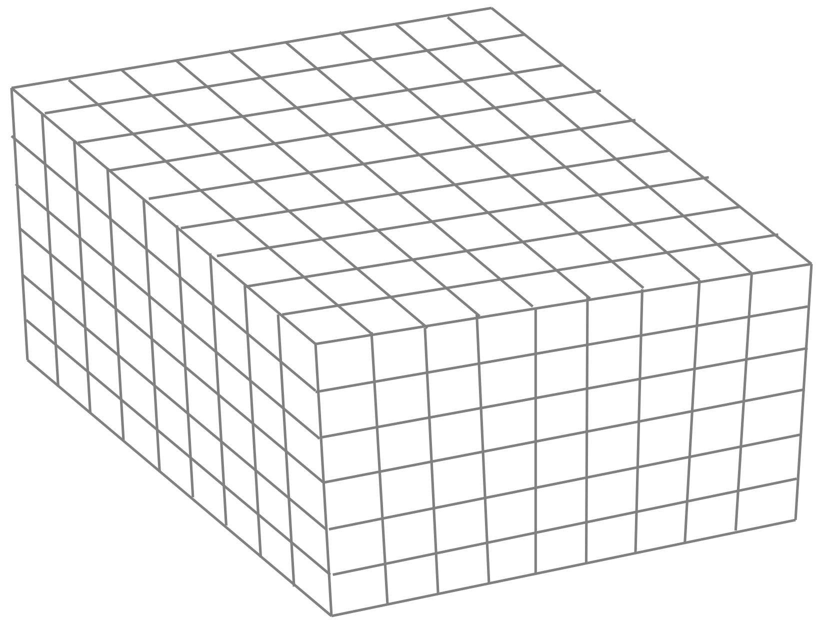
Discrete Formulation: Voxel Coloring

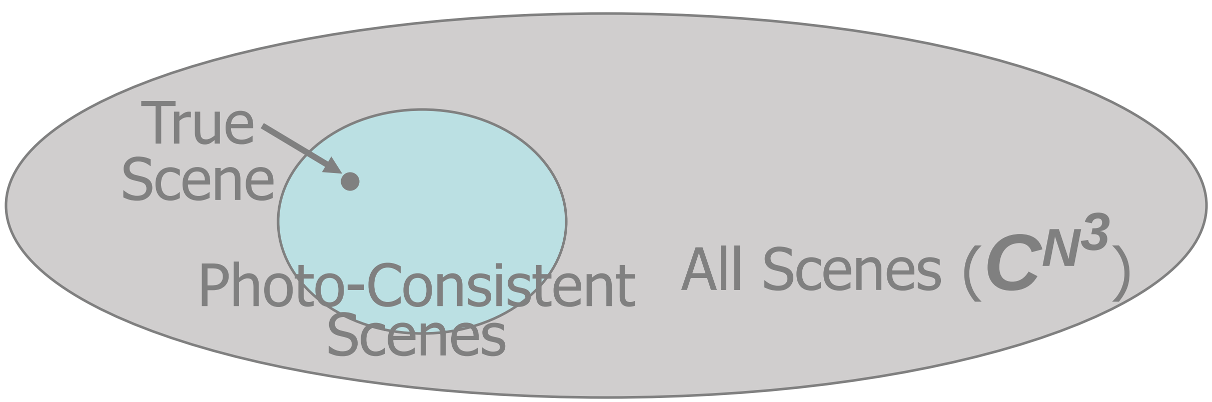


Goal: **Assign RGBA values to voxels in V**
photo-consistent **with images**

Complexity and Computability

Discretized
Scene Volume
N^3 voxels
C colors





Stereo vision

Two cameras, simultaneous views

Single moving camera and static scene