CSCI 4830 / 5722
Computer Vision

University of Colorado **Boulder**

CSCI 4830 / 5722
Computer Vision

Dr. Ioana Fleming
Spring 2019
Lecture 20

University of Colorado **Boulder**

Reminders

Submissions:

- Homework 4: Wed 3/20 at 11 pm

Readings:

- Szeliski:
  - chapter 11 (Stereo correspondence)
- P&F:
  - chapter 7 (Stereopsis)

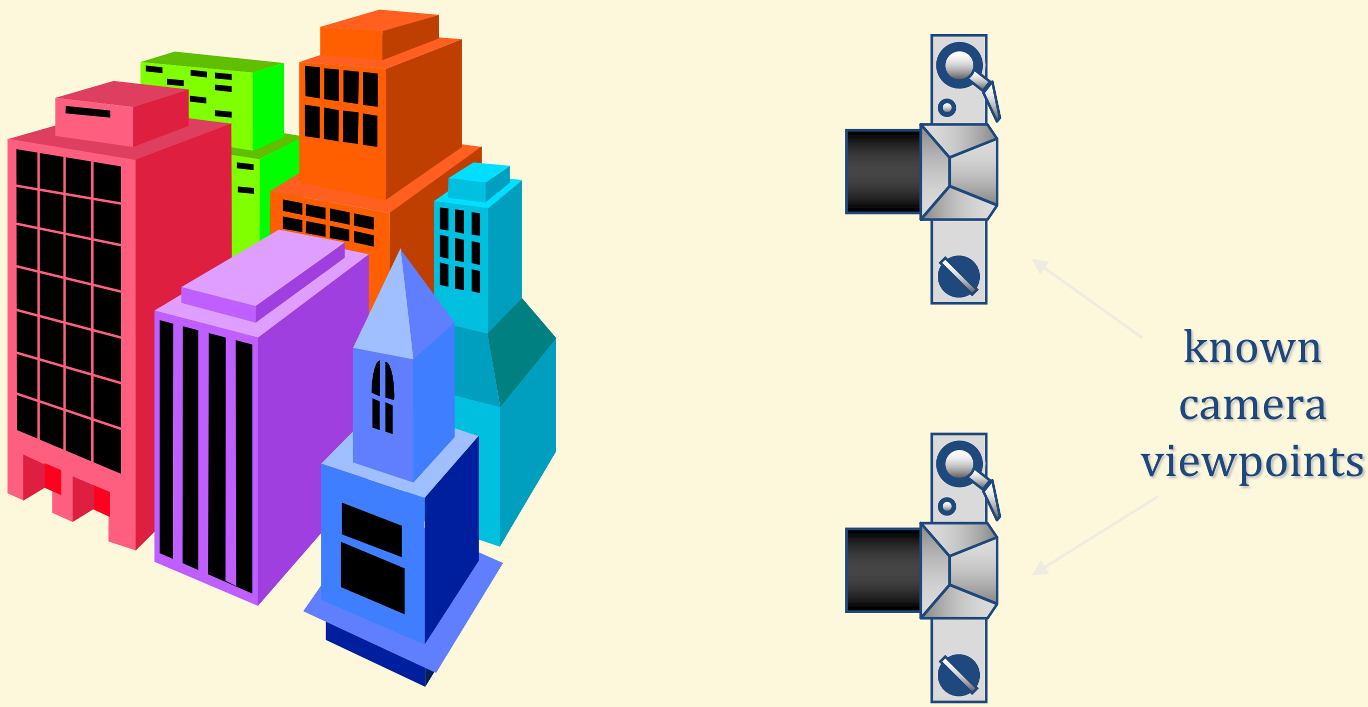
Today

- Epipolar geometry
- Rectification
- Finding correspondences

…with a lot of slides stolen from Steve Seitz, Fei Fei Li, Alexei Efros

Stereo Reconstruction

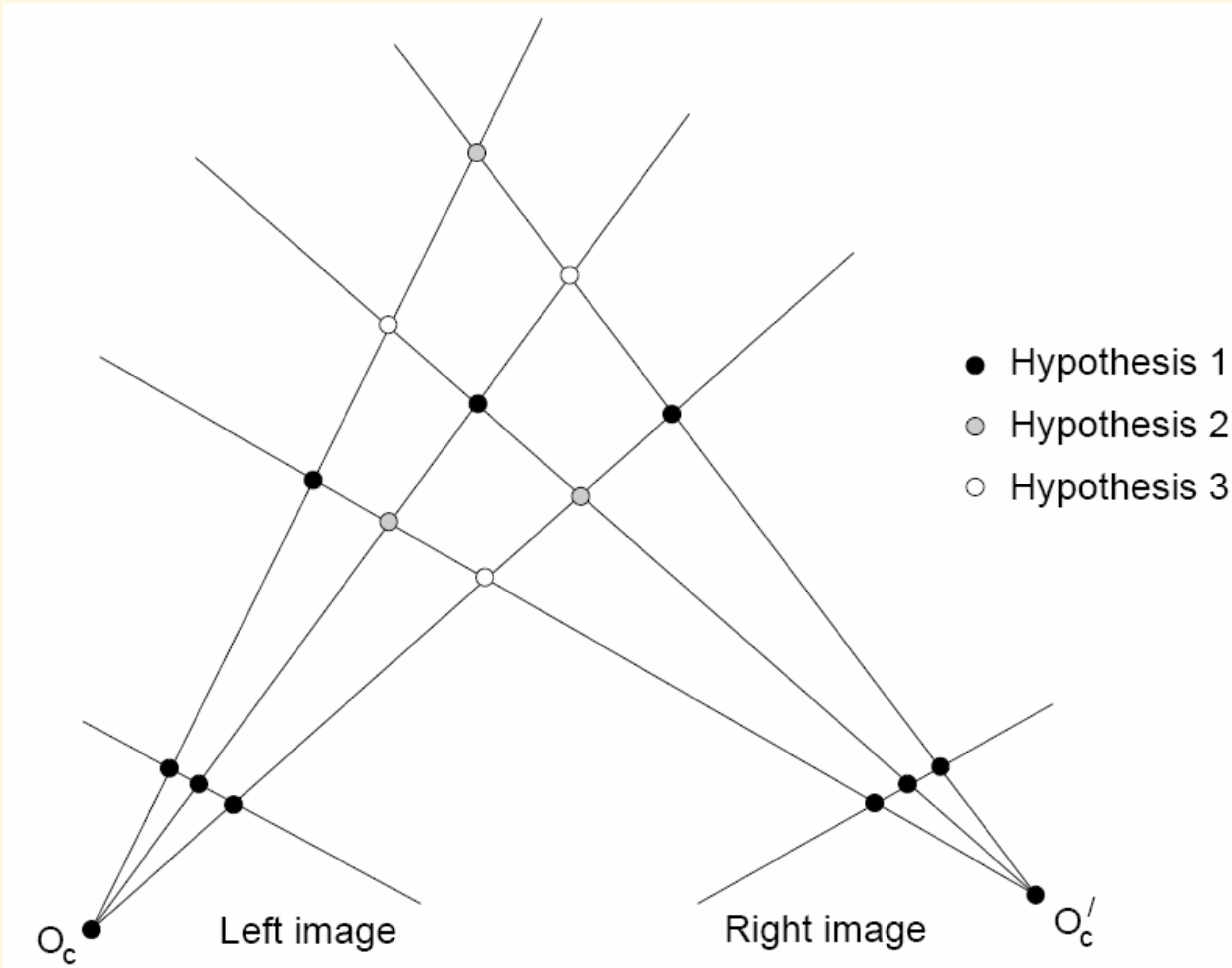
- The Stereo Problem
  - Shape from two (or more) images
  - Biological motivation



Stereopsis – two parts

1. The **correspondence problem** = fusion of features observed by two(or more) eyes
   – establishing correct correspondences is hard
2. The **reconstruction problem** = computing the three-dimensional preimage.
   – pretty straight-forward, for just one feature
   – harder for multiple features

Stereopsis – solving the correspondence problem



Multiple matching hypotheses, but which one is correct?

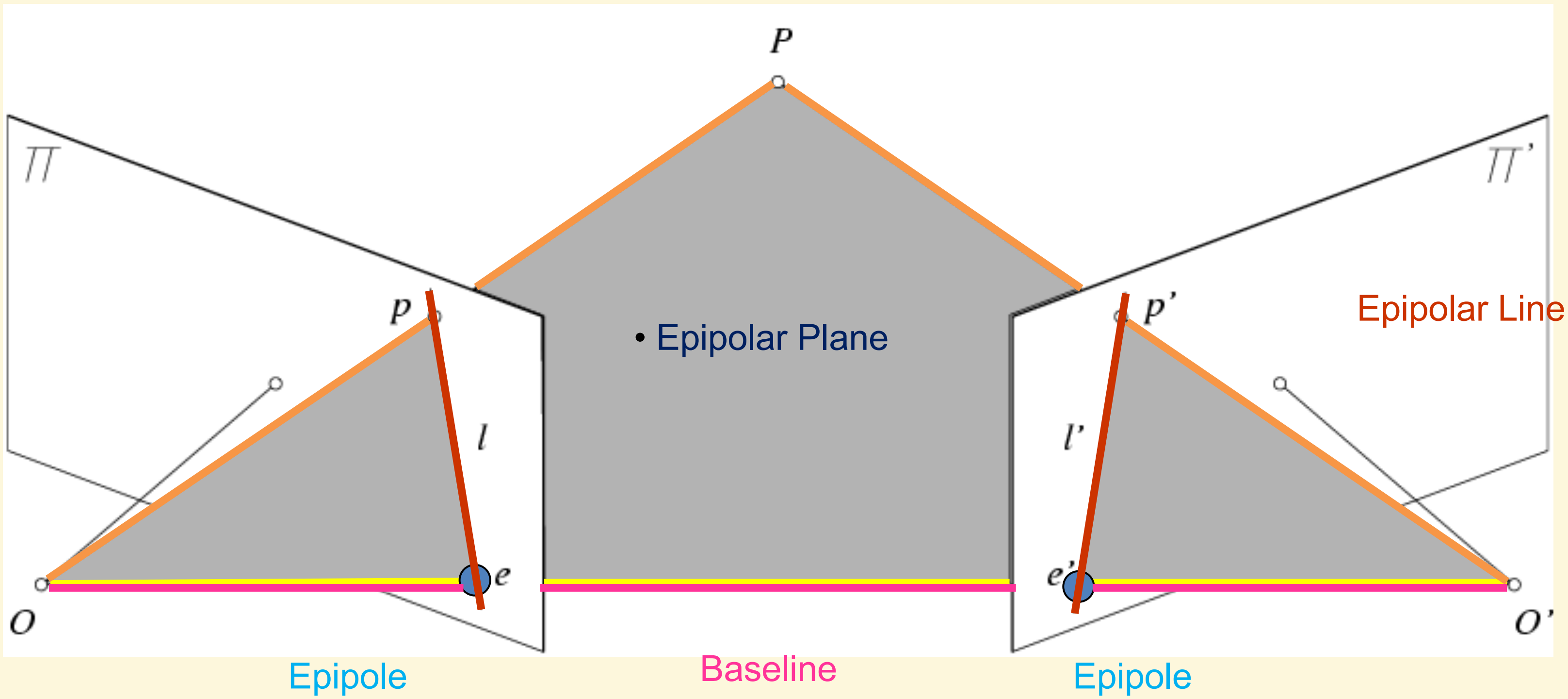
Why is the correspondence problem difficult?

- Some points in each image will have no corresponding point in the other image, because:

- The cameras may have different fields of view
- Due to occlusion
- A stereo system must be able to determine the image parts that should not be matched.

- In the above picture, the part with green and red are the parts that show the different viewpoint of the cameras
- The task is to find points, that can be seen for both cameras
- Occlusion is both visible at the right edge of the box

Epipolar geometry



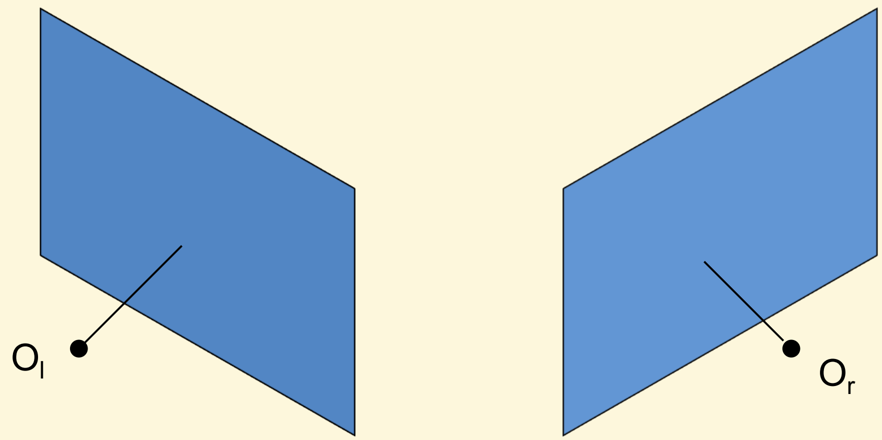
Epipolar constraint

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

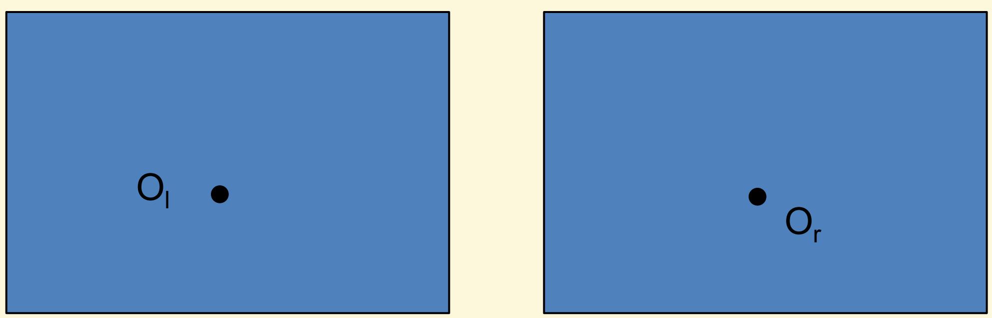
Image from Andrew Zisserman

What do the epipolar lines look like?

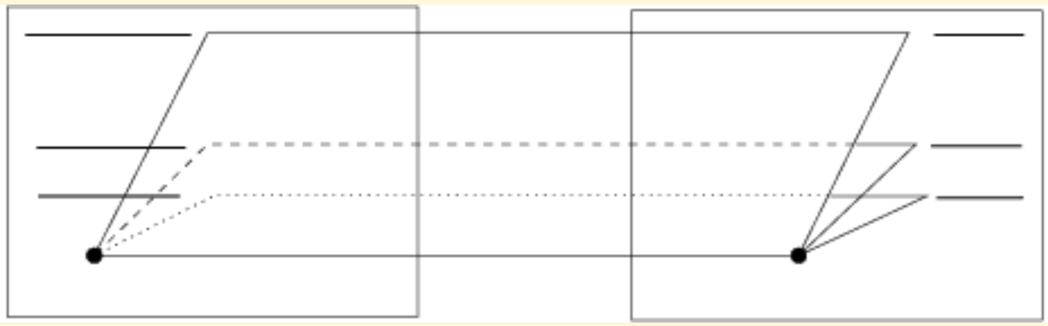
1.



2.



Example: parallel cameras



Where are the epipoles?

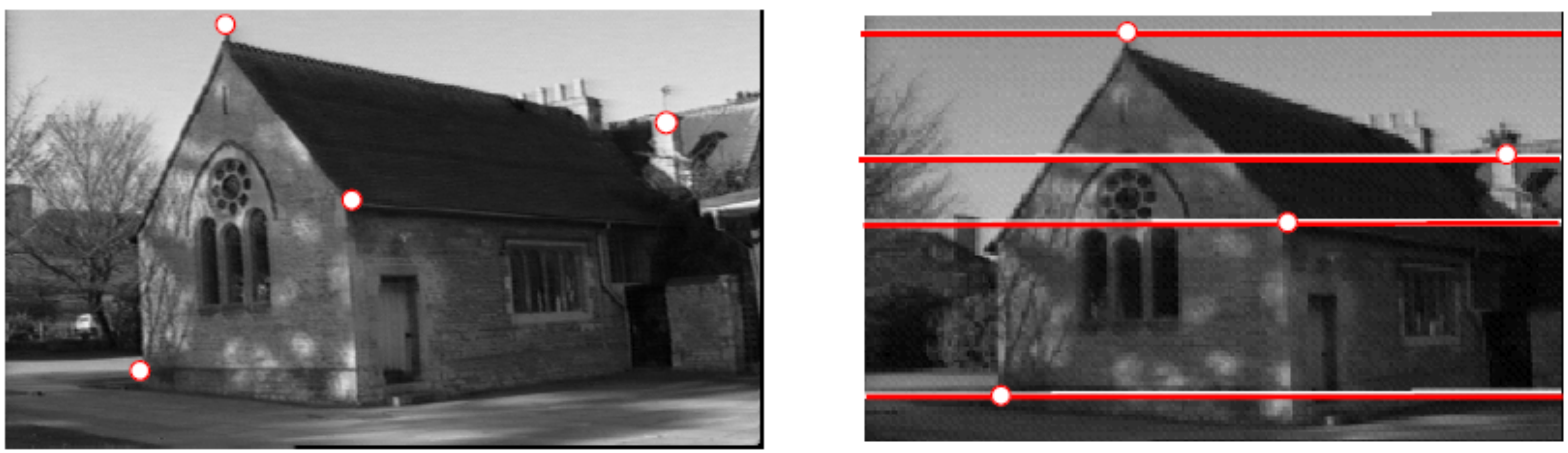


Figure from Hartley & Zisserman

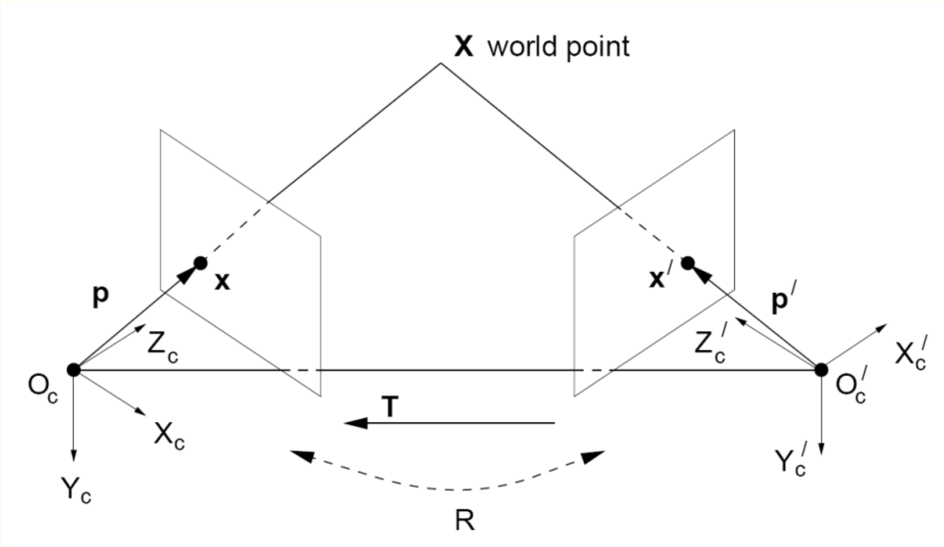
Essential matrix

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

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

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

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

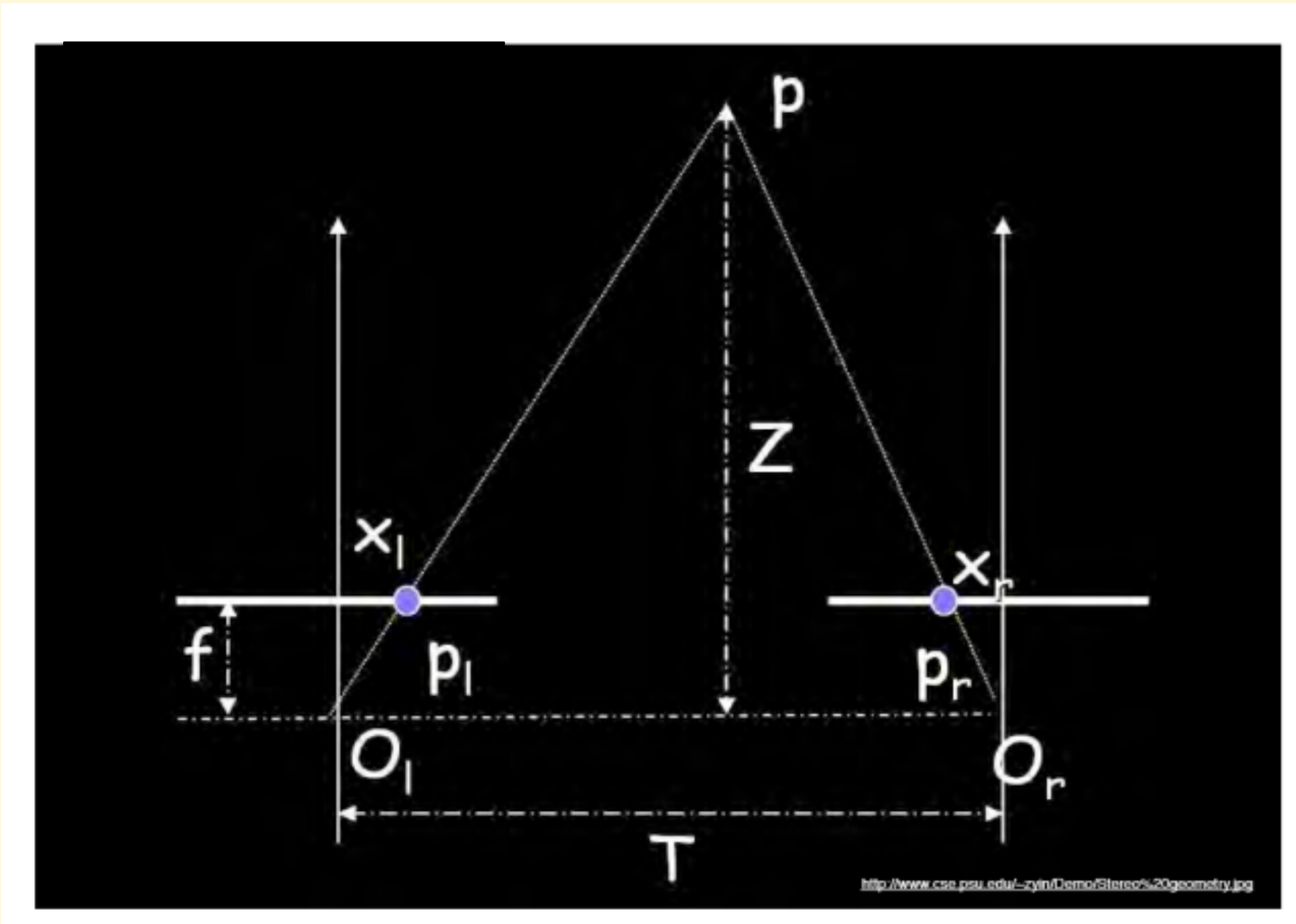


E is called the **essential matrix**, and it relates corresponding image points between both cameras, given the rotation and translation.

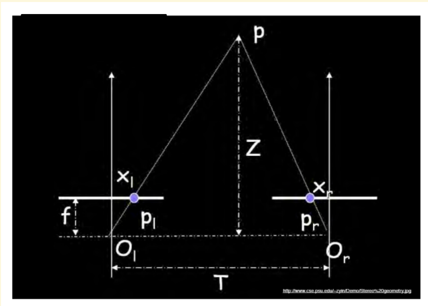
If we observe a point in one image, its position in other image is constrained to lie on line defined by above.

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

Essential matrix example: parallel cameras



Essential matrix example: parallel cameras



$$\mathbf{R} =$$

$$\mathbf{T} =$$

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

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

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

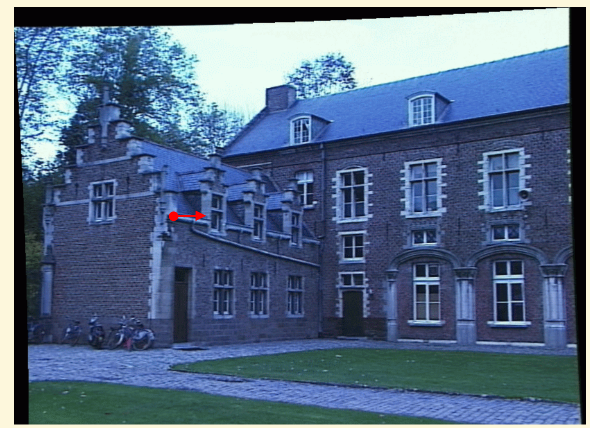
$$\mathbf{p'}^{\mathrm{T}}\mathbf{E}\mathbf{p} = 0$$

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 $I'(x',y')$

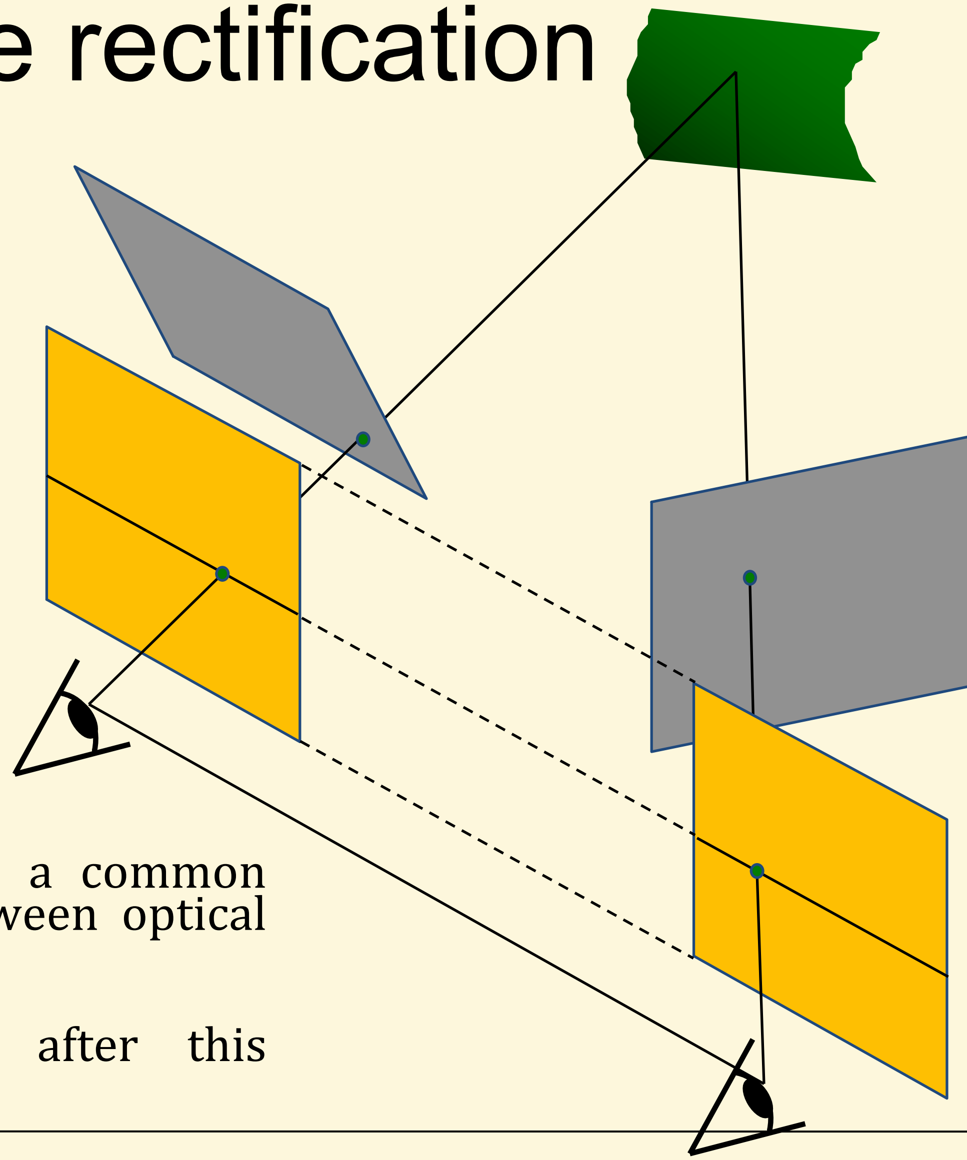


$$(x',y')=(x+D(x,y),y)$$

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.

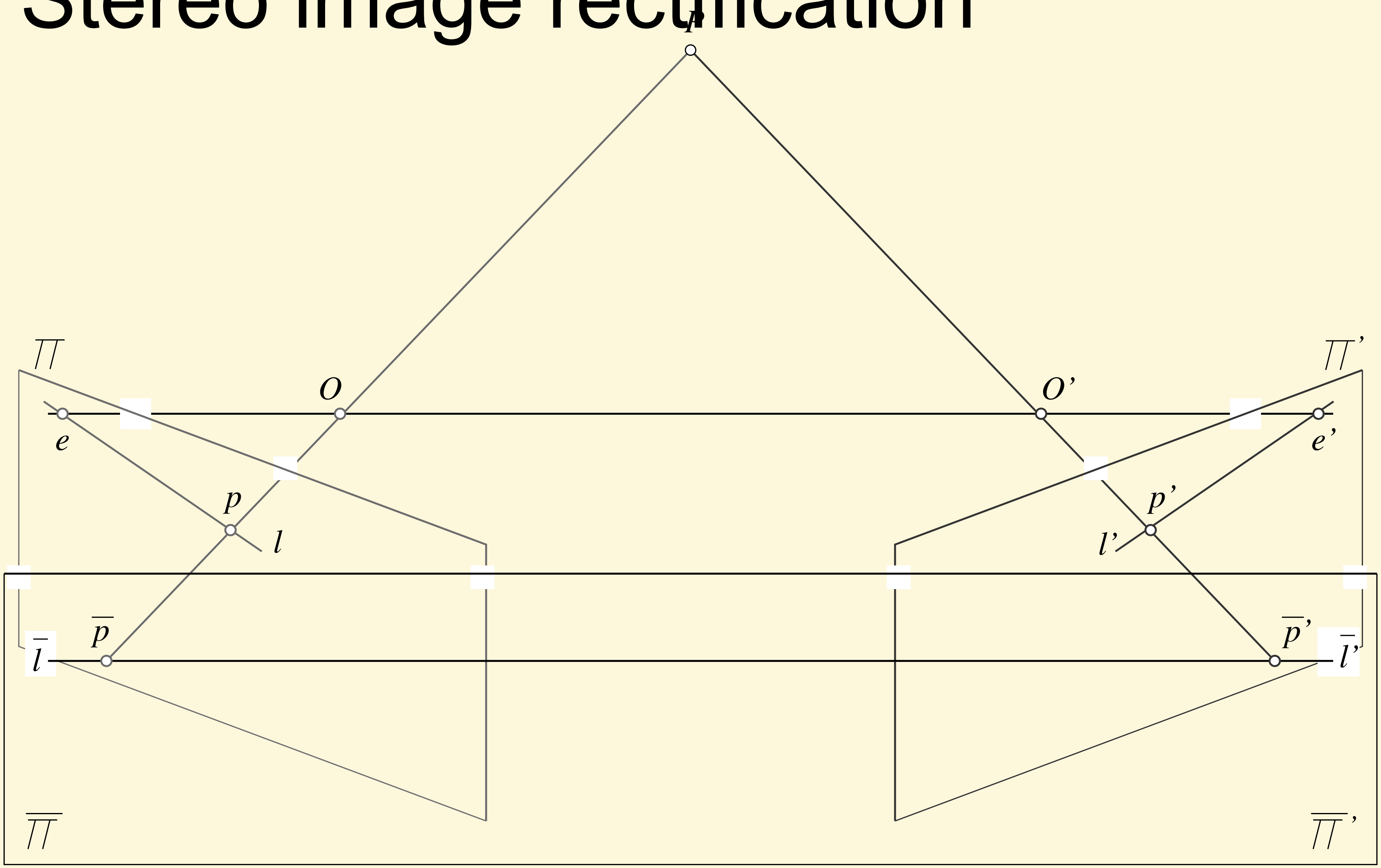


Re-project image planes onto a common plane parallel to the line between optical centers.

Pixel motion is horizontal after this transformation

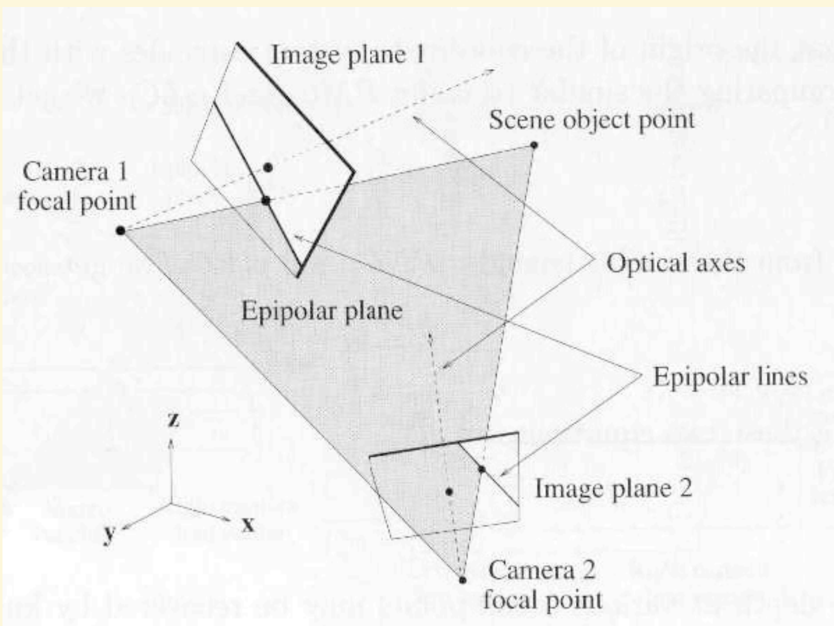
Slide credit: Li Zhang

Stereo image rectification

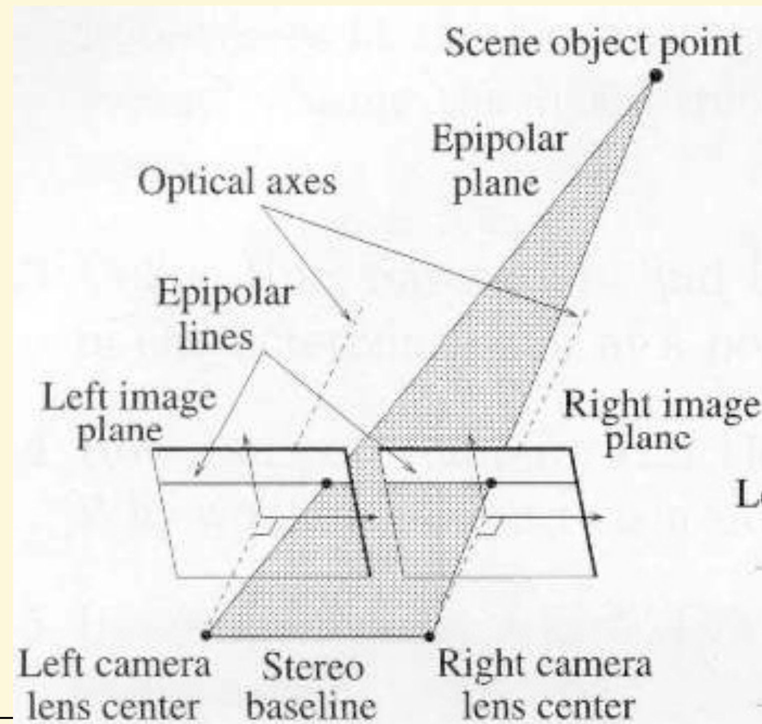


Rectification

Epipolar lines are at arbitrary locations and orientations

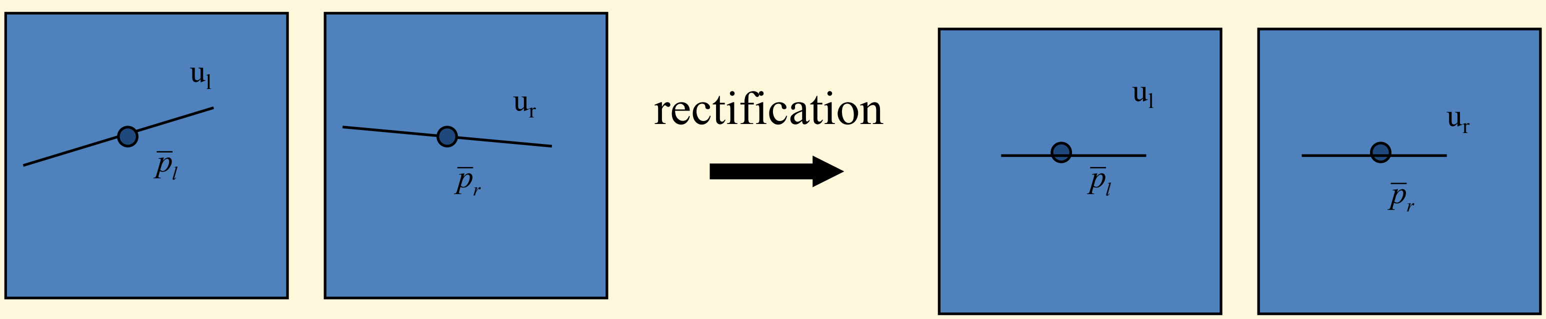


Epipolar lines are colinear and parallel to baseline



Rectification (cont'd)

- Rectification is a transformation which makes pairs of conjugate epipolar lines become collinear and parallel to the horizontal axis (i.e., baseline)
- Searching for corresponding points becomes much simpler for the case of rectified images



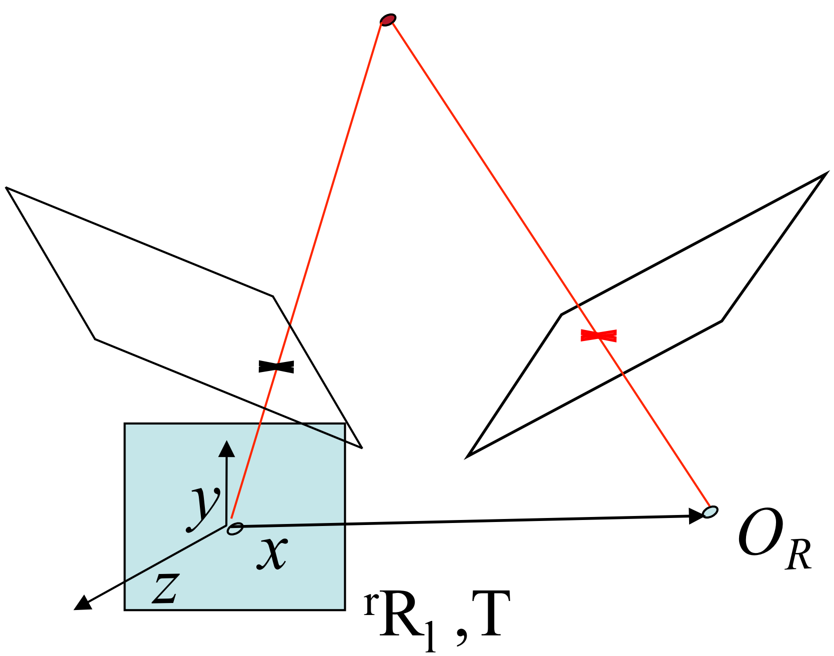
Rectification: How?

- New coordinate system for rectified image planes:
  - X axis: baseline (translation between two optical centers)

    x = T/||T||
  - y axis is orthogonal to x ([0,0,1] is the z axis of the current left camera)

y = [0,0,1] cross x/|| [0,0,1] cross x ||

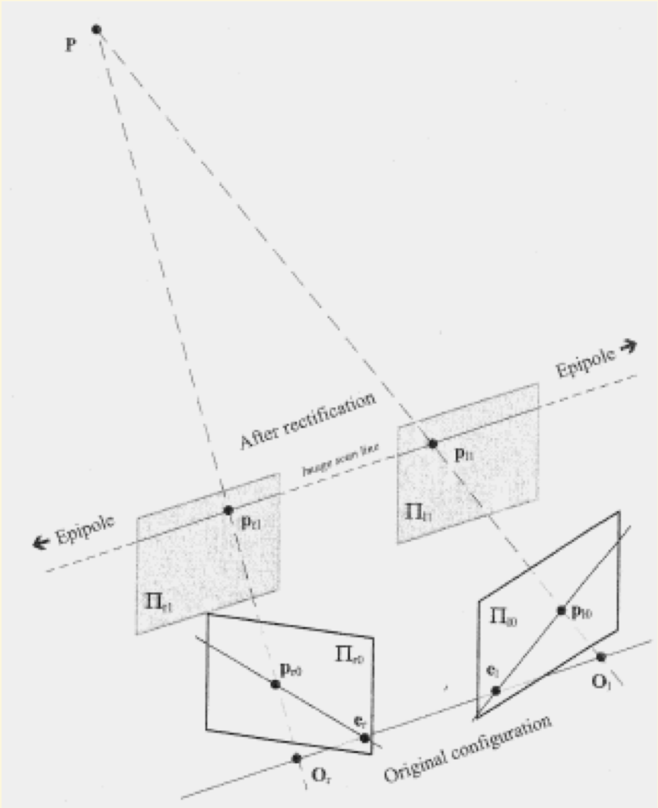
  - z = x cross y



Rectification (cont'd)

Main steps:

(assuming knowledge of the extrinsic/intrinsic stereo parameters):



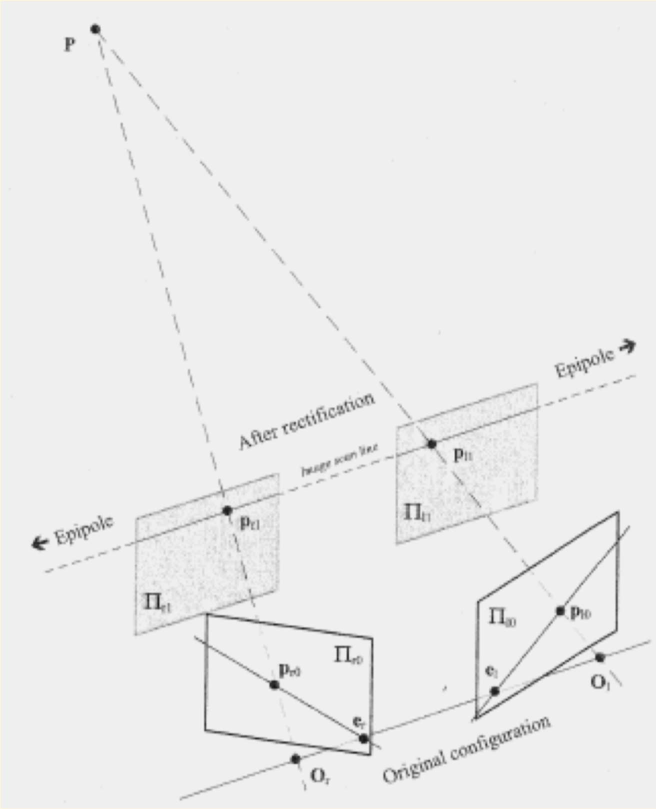
(1) Rotate the left camera so that the epipolar lines become parallel to the horizontal axis (i.e., epipole is mapped to infinity).

Rectification (cont'd)

(2) Apply the same rotation to the right camera to recover the original geometry.

(3) Align right camera with left camera using rotation R.

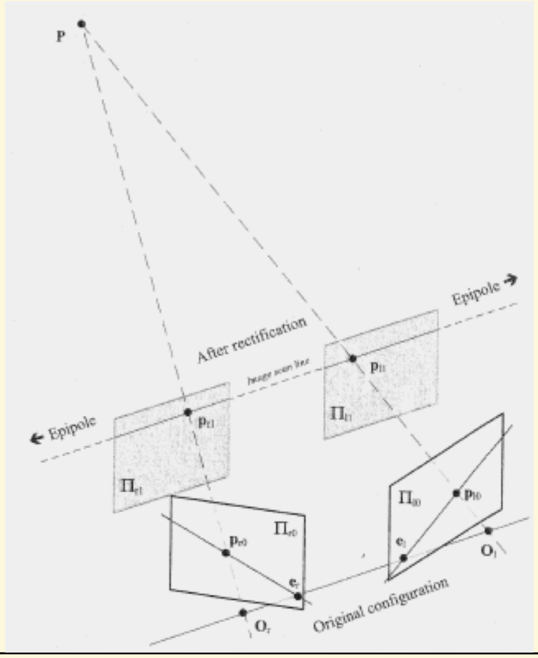
(4) Adjust scale in both camera frames.

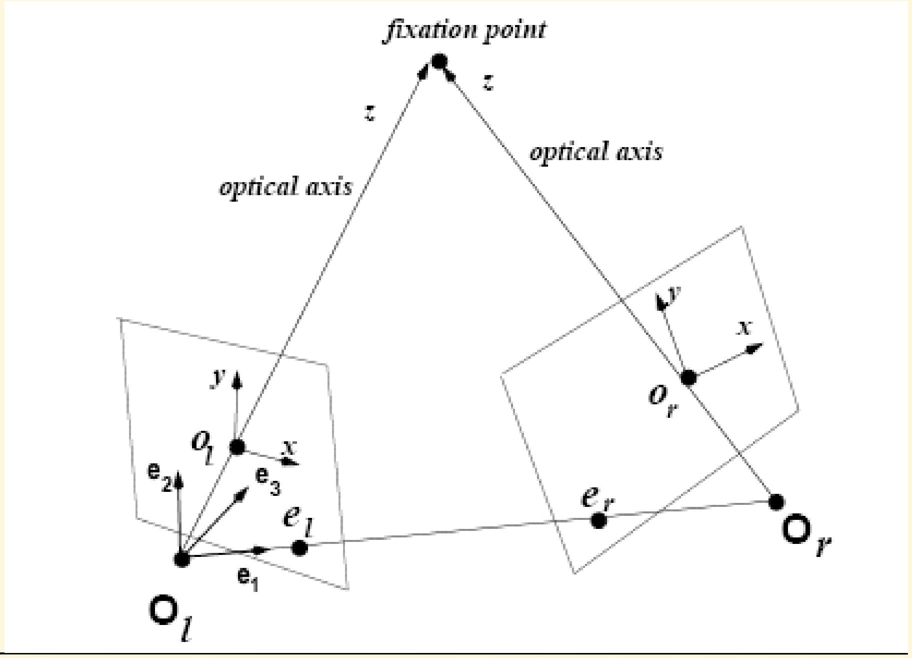


Rectification (cont'd)

Consider step (1) only (i.e., other steps are easy):

- Build coordinate system (e_1, e_2, e_3) centered at O_l.

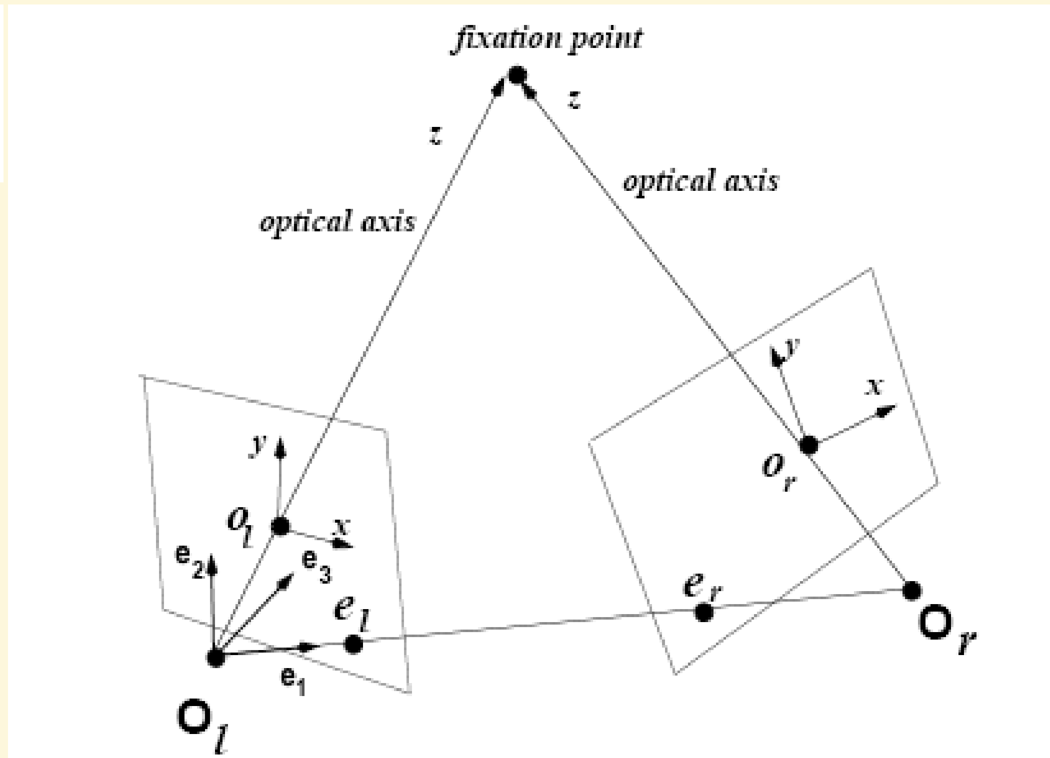




Rectification (cont'd)

(1.1) e_1 is a unit vector along the vector *T (baseline)*

$$e_1 = \frac{T}{\|T\|} = \frac{[T_x, T_y, T_z]^T}{\sqrt{T_x^{\ 2} + T_y^{\ 2} + T_z^{\ 2}}}$$

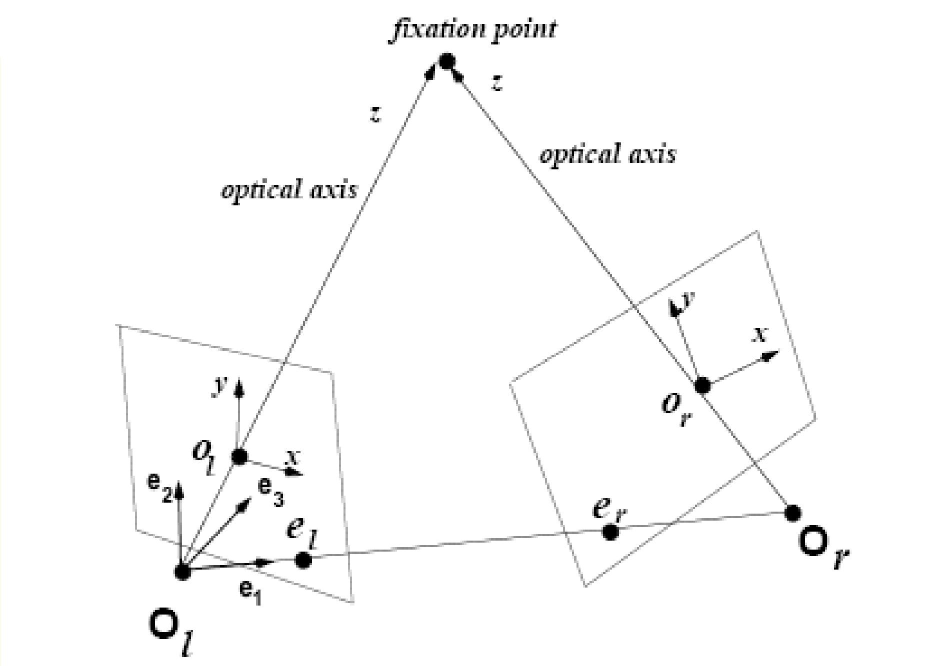


Rectification (cont'd)

(1.2) e_2 must be perpendicular to e_1 (i.e., cross product of e_1 with the optical axis)

$$e_2 = \frac{e_1 \; x \; [0,0,1]^T}{\|e_1 \; x \; z\|} = \frac{1}{\sqrt{{T_x}^2 + {T_y}^2}} [-T_y, T_x, 0]^T$$

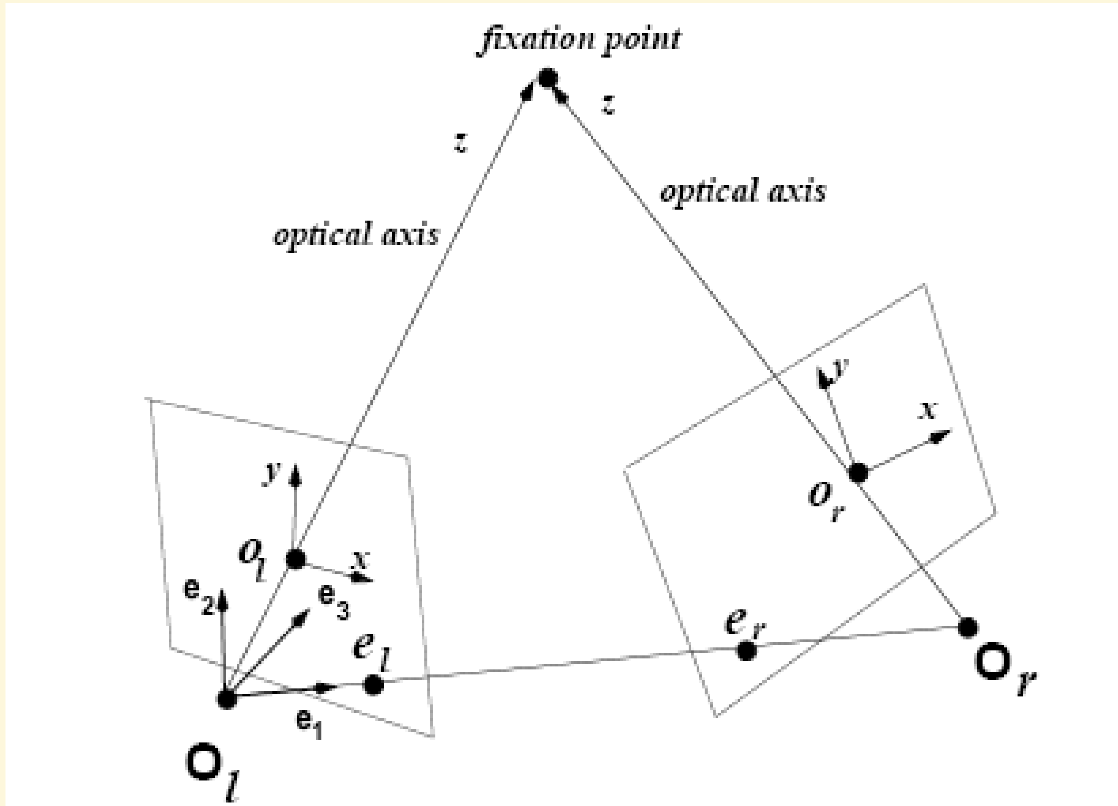
$z=[0,0,1]^T$



Rectification (cont'd)

(1.3) choose e_3 as the cross product of e_1 and e_2

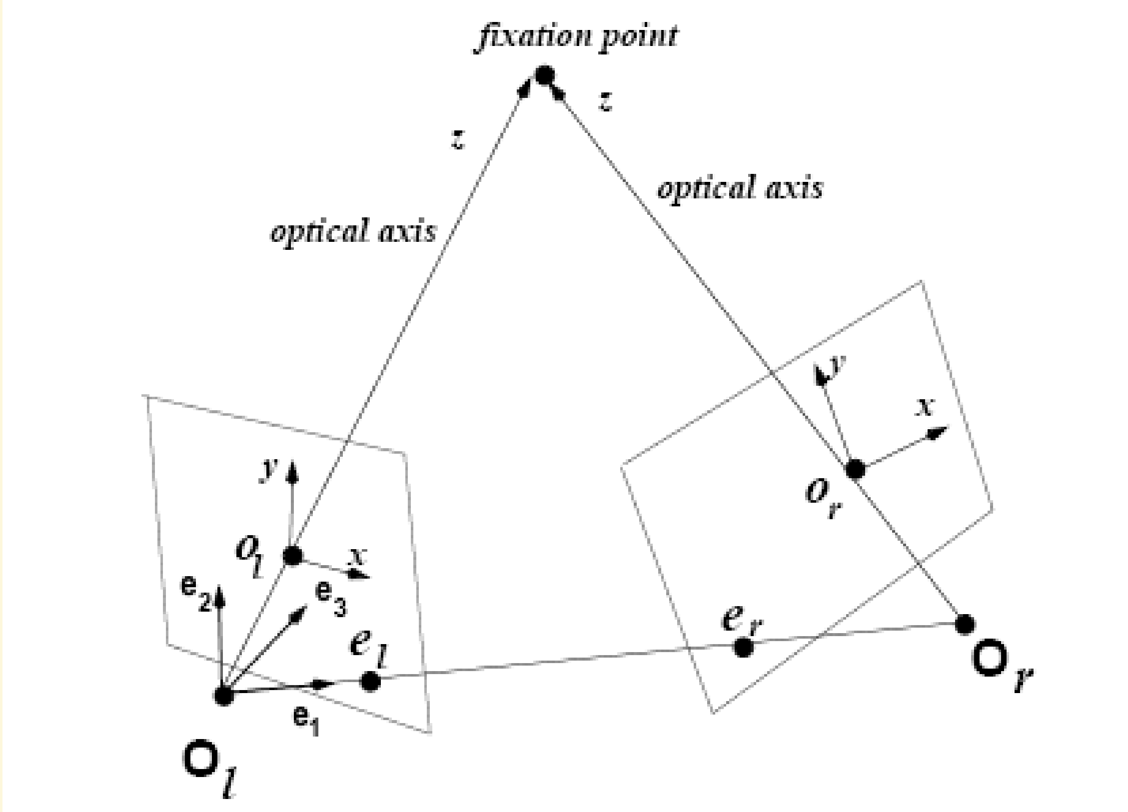
$$e_3 = e_1 \times e_2$$



Rectification (cont'd)

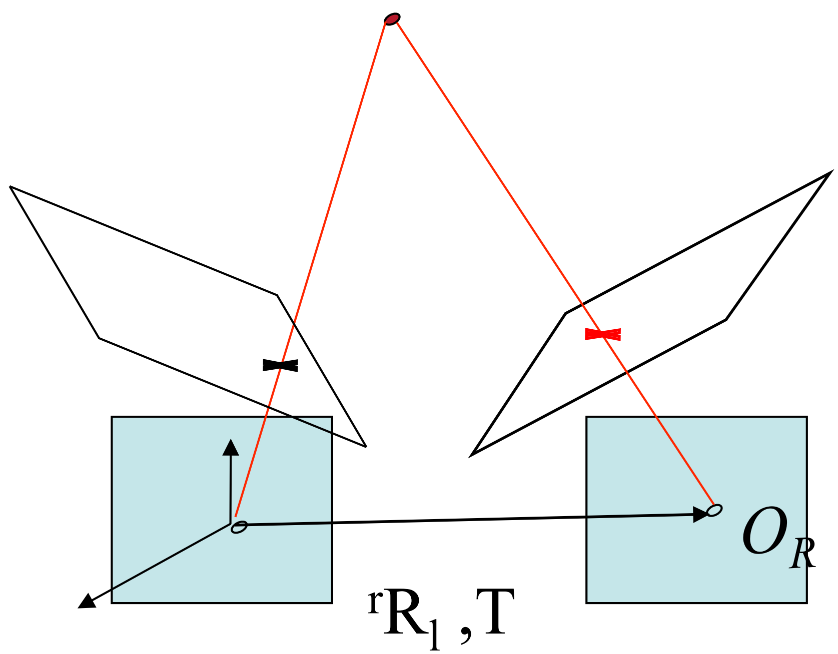
The rotation matrix that maps the left epipole to infinity is the transformation that aligns (e_1, e_2, e_3) with (i ,j, k):

$$R_{rect} = \begin{bmatrix} e_1^T \\ e_2^T \\ e_3^T \end{bmatrix}$$

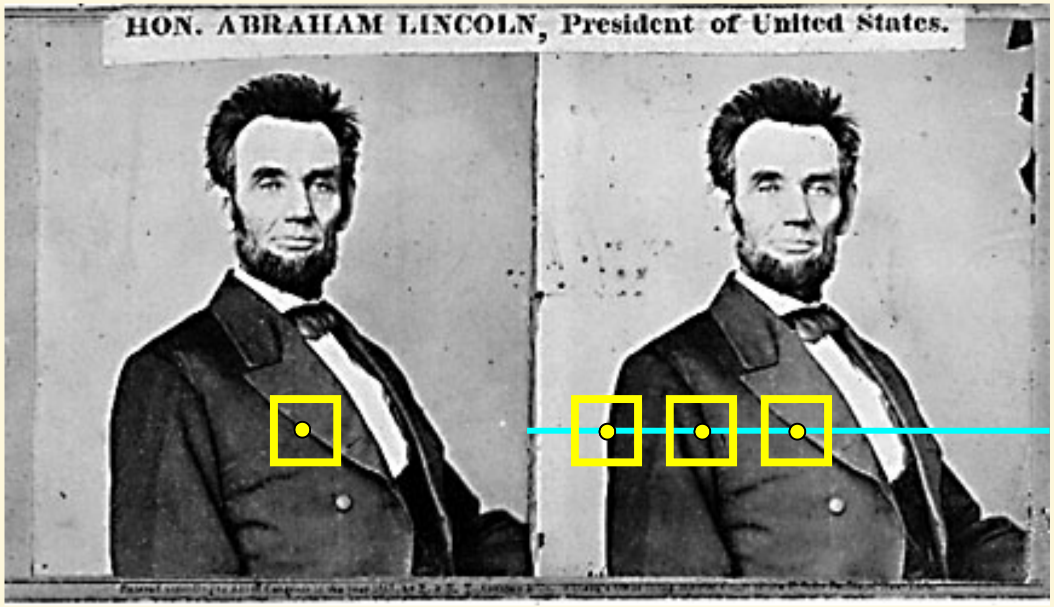


Rectification: How?

- Both the left and right camera can now be rotated to be parallel to this frame
  - ${}^{l}R_u$ = [x y z] is the rotation from unverged to verged frame for left camera
  - ${}^{r}R_u = {}^{r}R_l \, {}^{l}R_u$

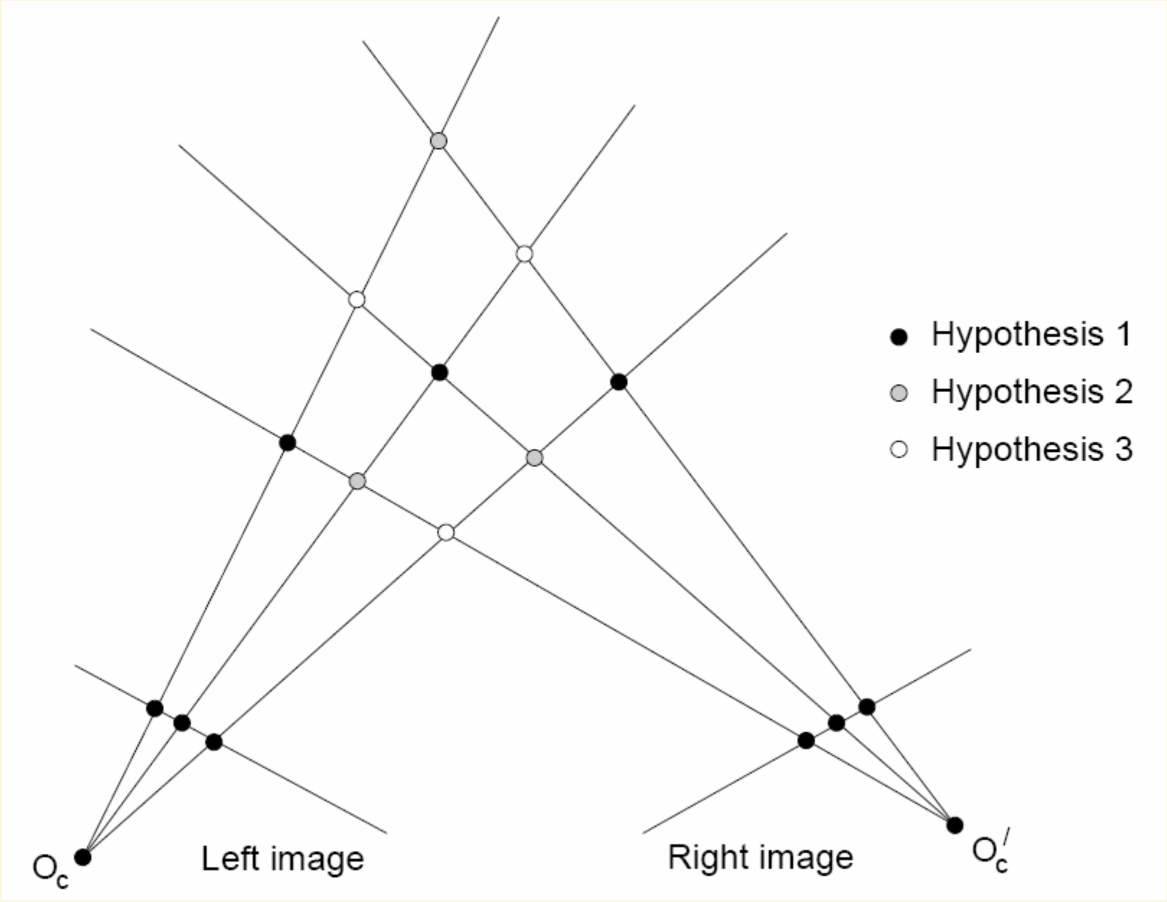


Basic stereo matching algorithm



- 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) = 1/(x-x')

Correspondence problem

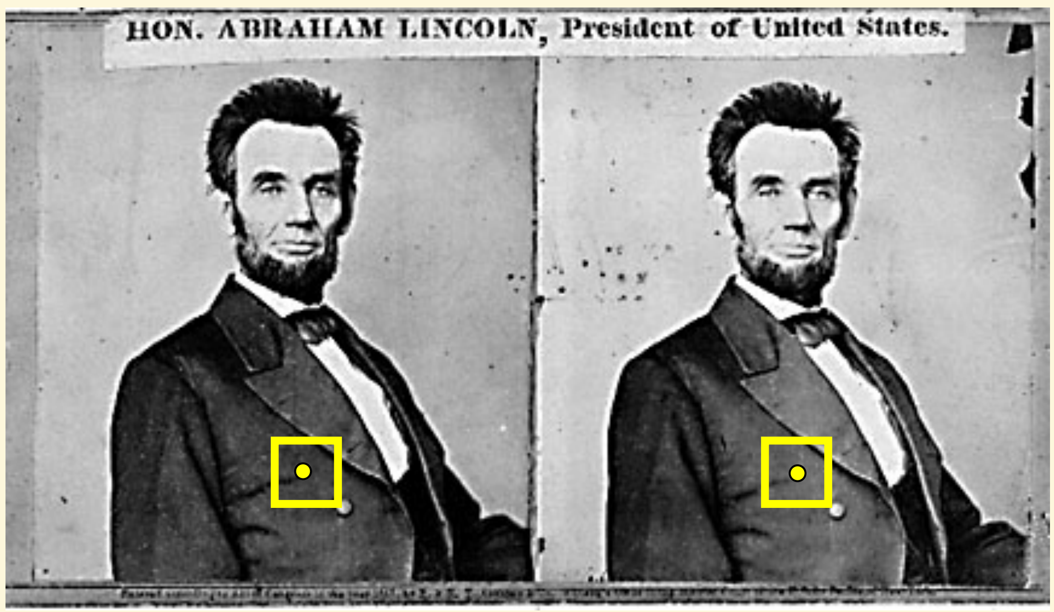


Correspondence problem

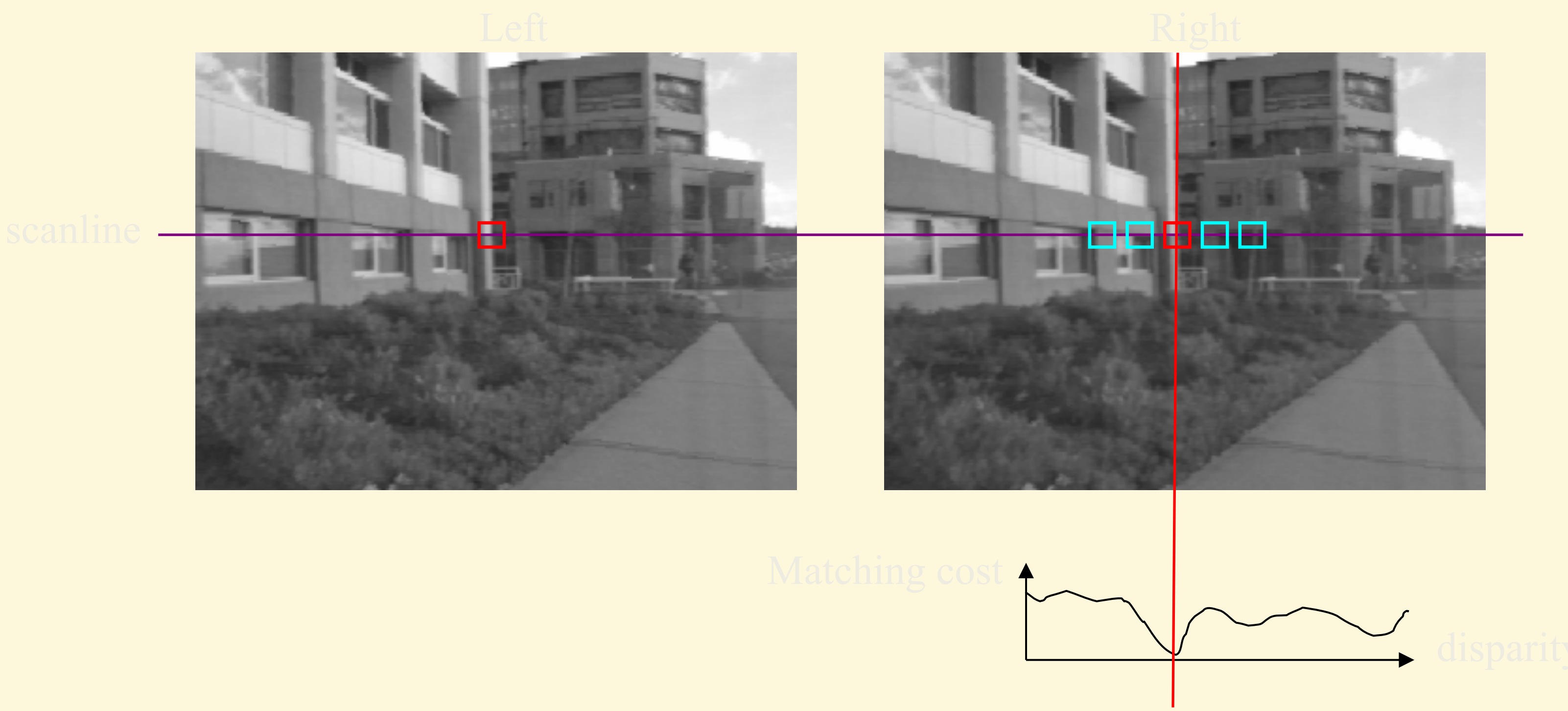
- Beyond the hard constraint of epipolar geometry, there are “soft” constraints to help identify corresponding points
  - Similarity
  - Uniqueness
  - Ordering
  - Disparity gradient

Correspondence problem

- Let's make some assumptions to simplify the matching problem
  - The baseline is relatively small (compared to the depth of scene points)
  - Then most scene points are visible in both views
  - Also, matching regions are similar in appearance



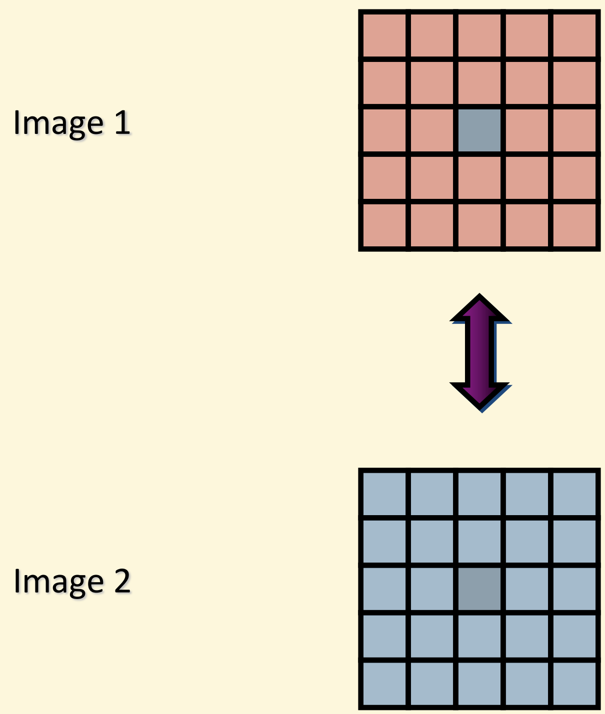
Correspondence search with similarity constraint



- 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

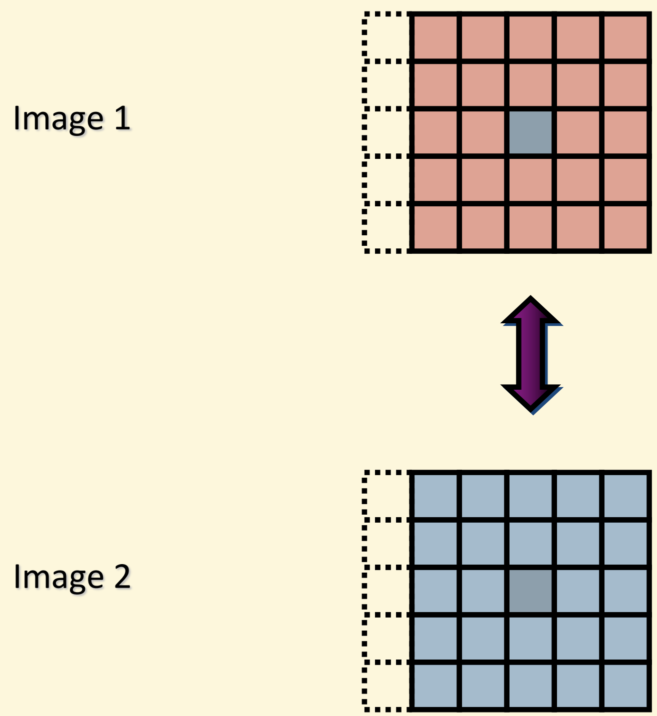
Incremental computation

- Given SSD of a window, at some disparity



Incremental computation

- Want: SSD at next location



Incremental computation

- Subtract contributions from leftmost column, add contributions from rightmost column

Image 1

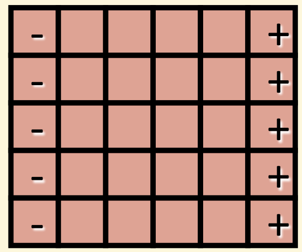
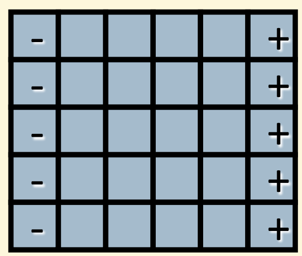


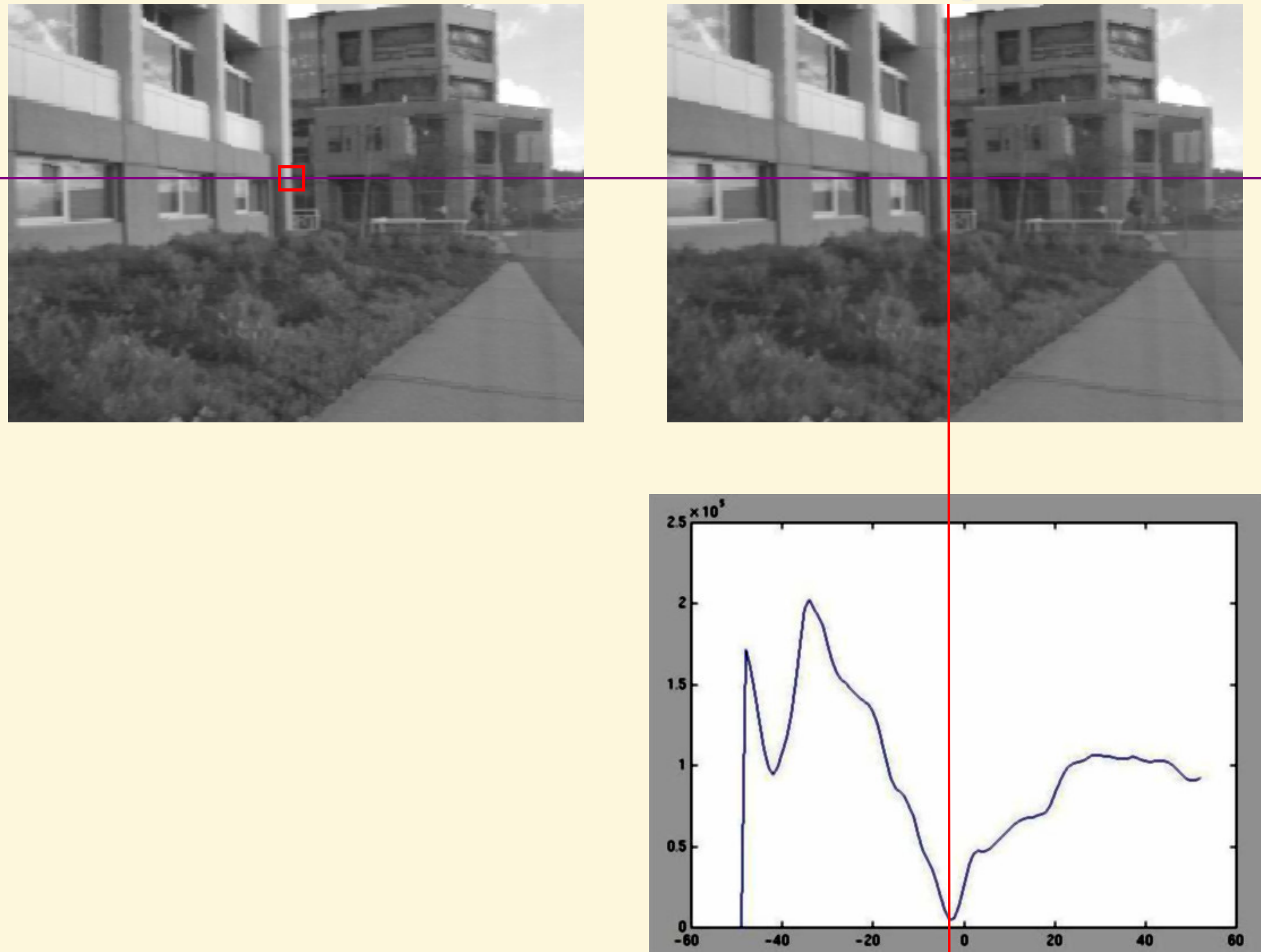
Image 2



Correspondence search with similarity constraint

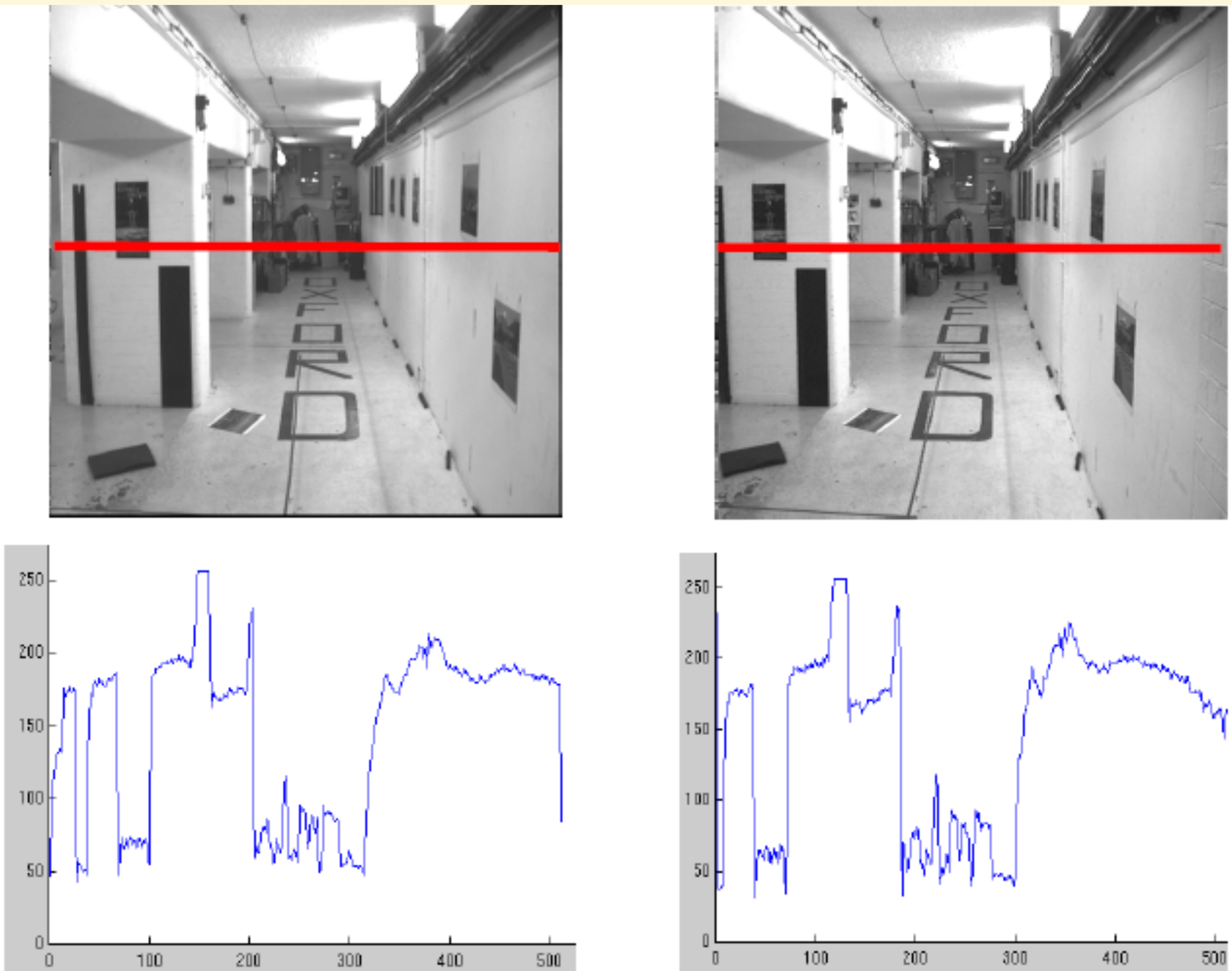
Left　　　　Right

scanline



SSD

Correspondence problem



Intensity profiles

- Clear correspondence between intensities, but also noise and ambiguity

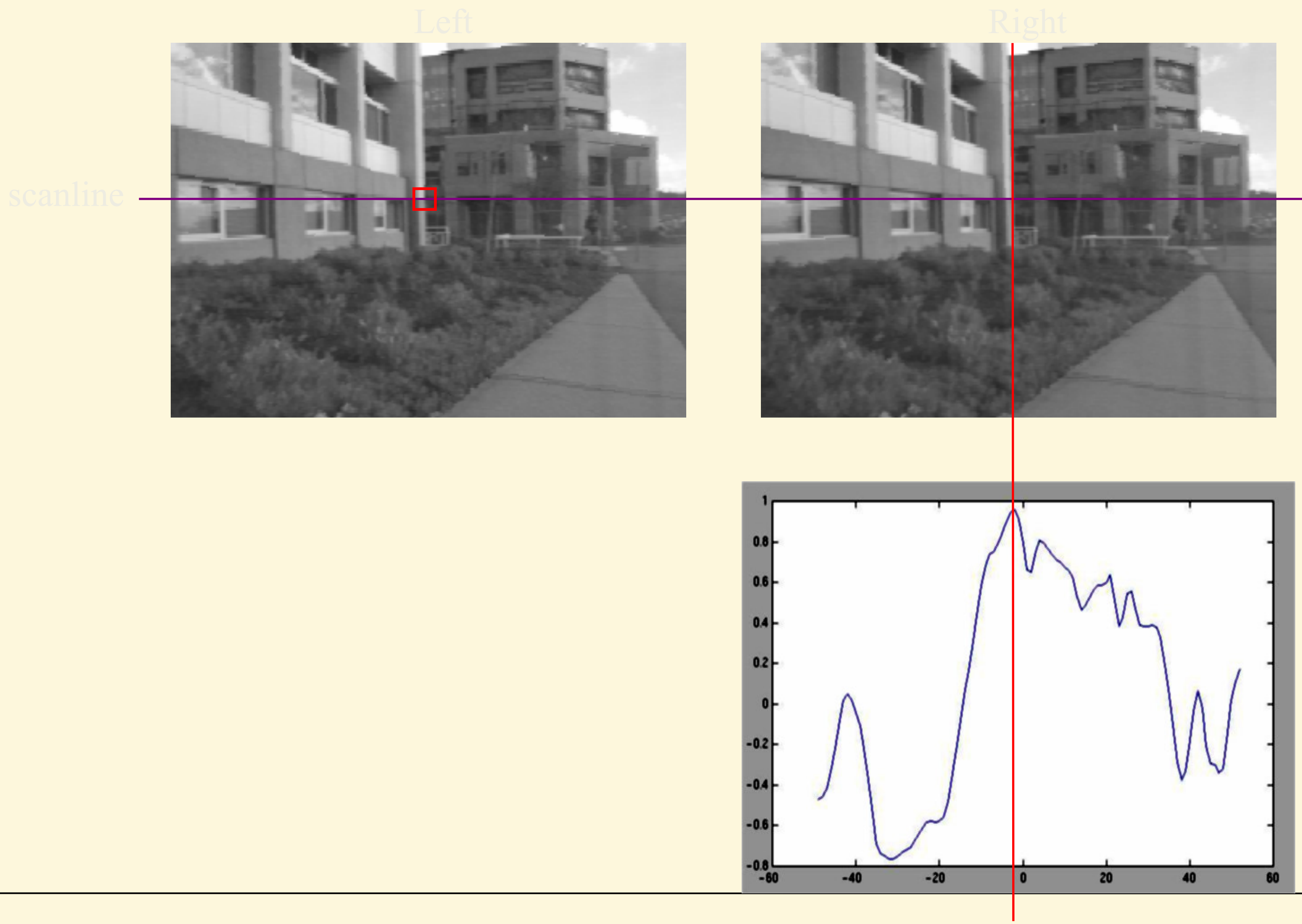
Source: Andrew Zisserman

Correspondence problem

Neighborhoods of corresponding points are similar in intensity patterns.

Source: Andrew Zisserman

Correspondence search with similarity constraint



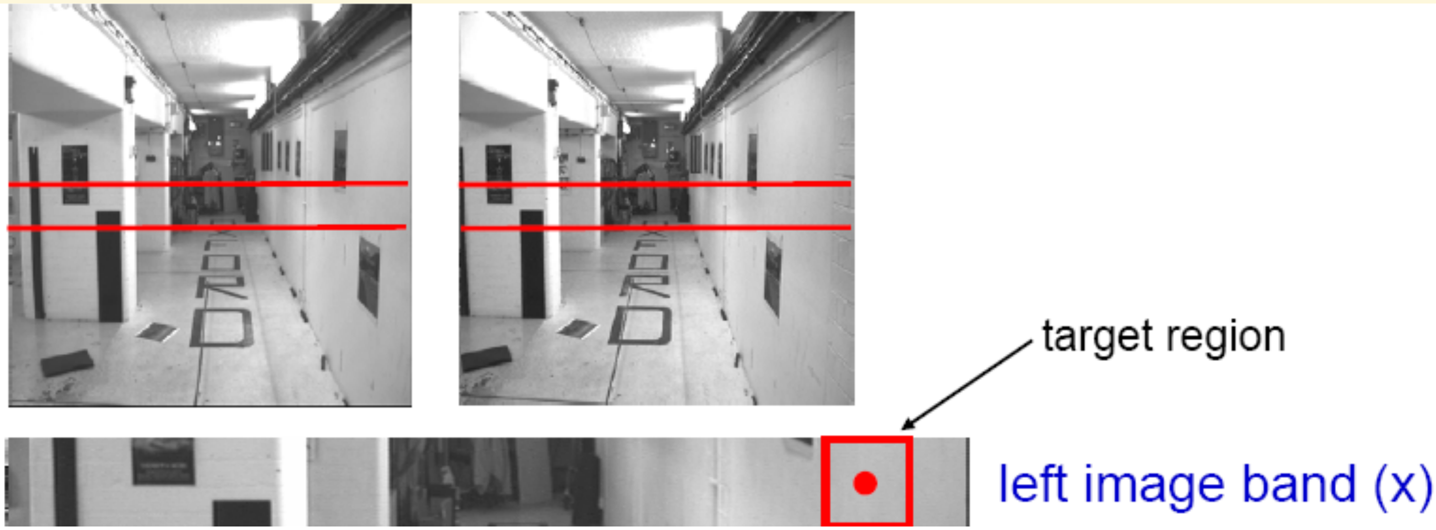
Normalized correlation

Correlation-based window matching

left image band (x)

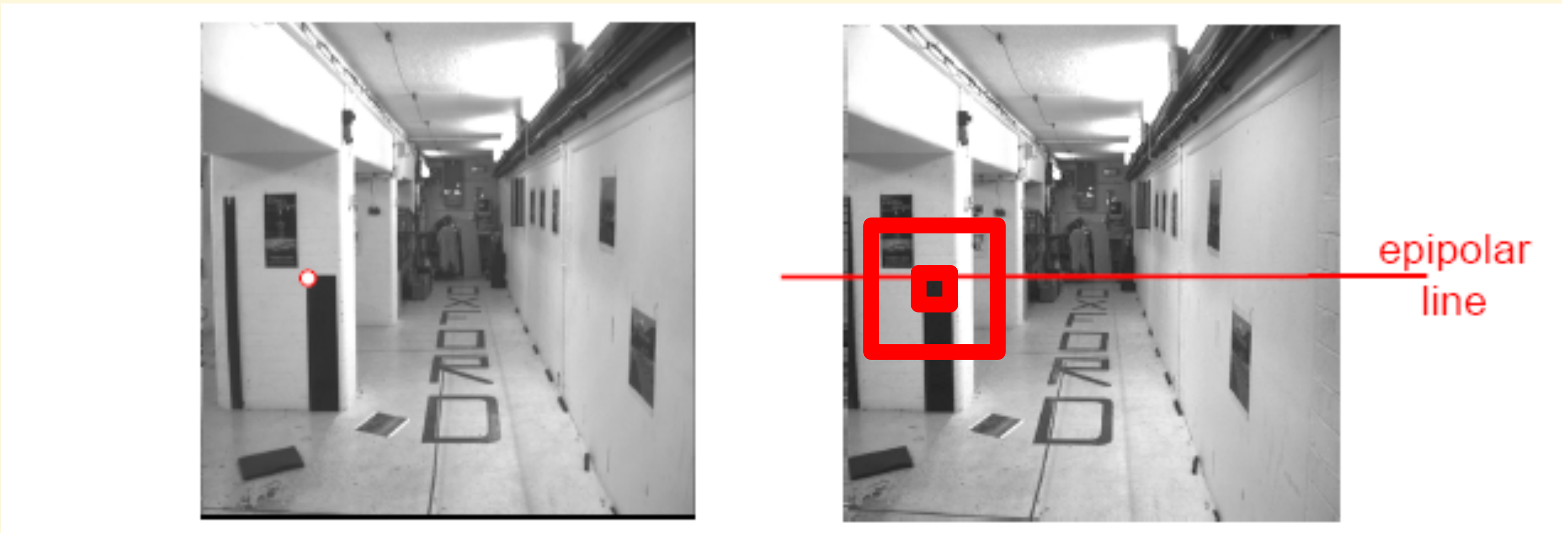
Source: Andrew Zisserman

Textureless regions



Source: Andrew Zisserman

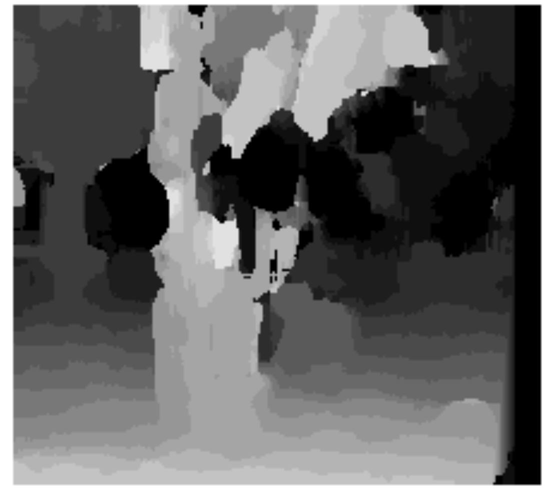
Effect of window size



Source: Andrew Zisserman

Effect of window size

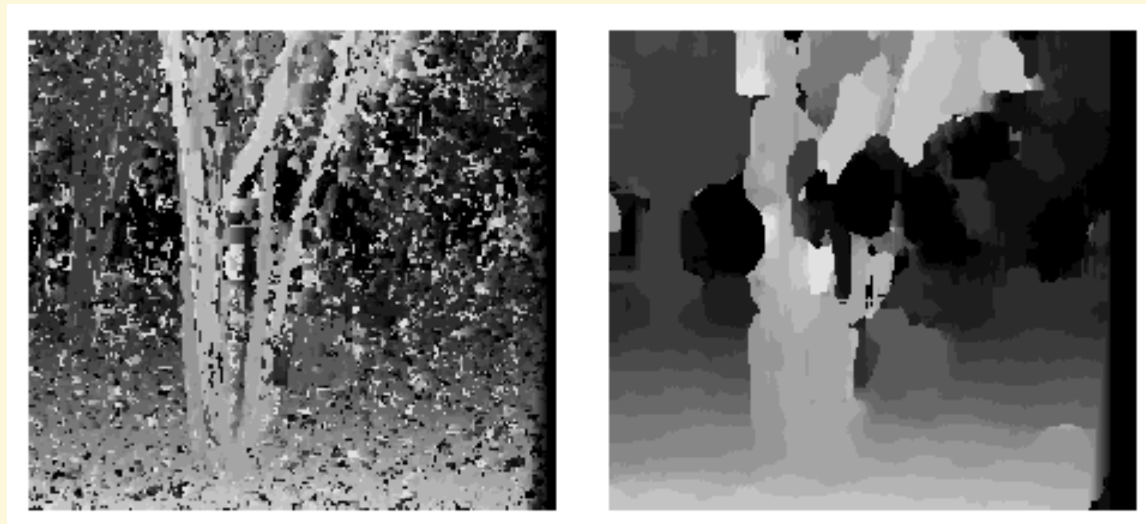
W = 3



W = 20

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

Effect of window size



W = 3 W = 20

Want window large enough to have sufficient intensity variation, yet small enough to contain only pixels with about the same disparity.

Figures from Li Zhang

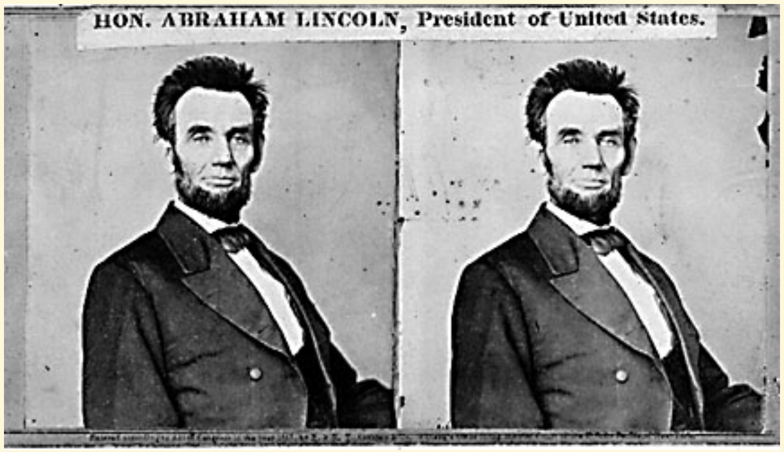
Non-square windows

- Compromise: have a large window, but higher weight near the center
- Example: Gaussian
- Example: Shifted windows (computation cost?)

Problems with window matching

- No guarantee that the matching is one-to-one
- Hard to balance window size and smoothness

Limitations of similarity constraint

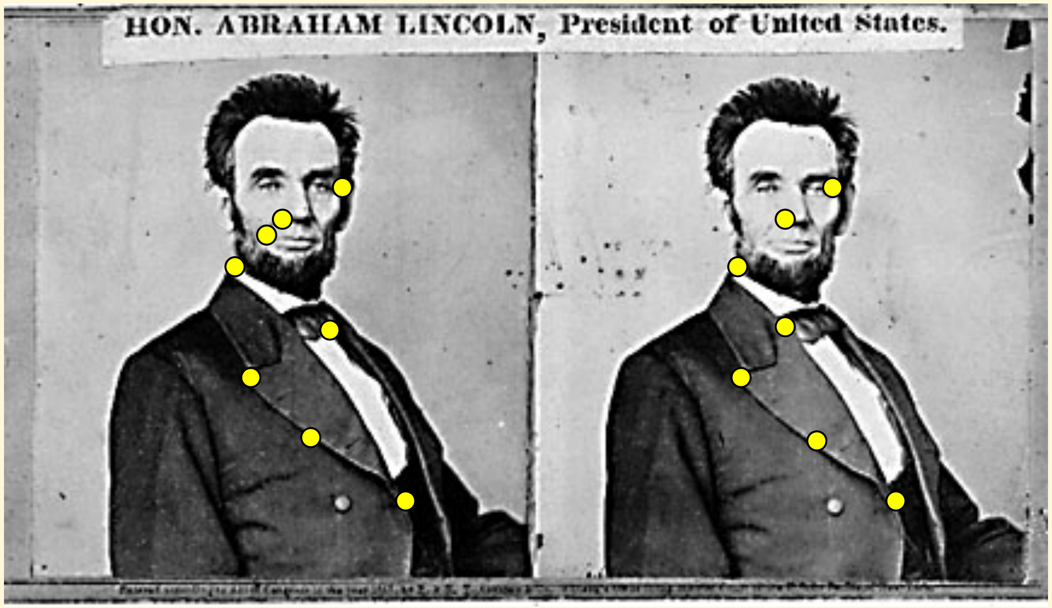


Textureless surfaces

Occlusions, repetition

Non-Lambertian surfaces, specularities

Sparse correspondence search



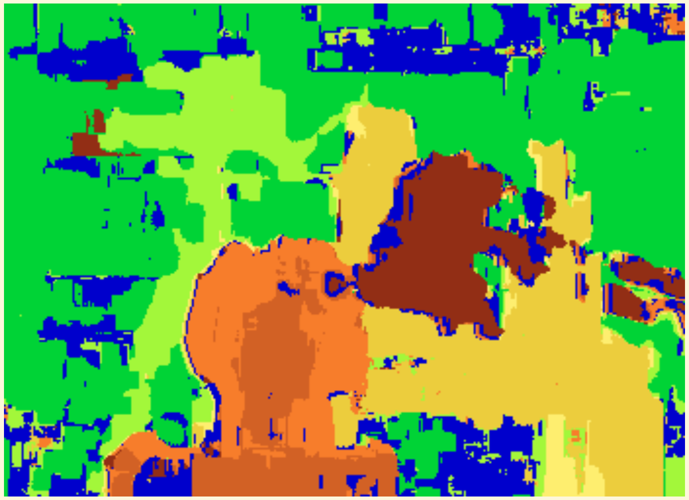
- Restrict search to sparse set of **detected features** (e.g., corners)
- Rather than pixel values (or lists of pixel values) use *feature descriptor* and an associated *feature distance*
- Still narrow search further by epipolar geometry

Tradeoffs between dense and sparse search?

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/

How can we improve window-based matching?

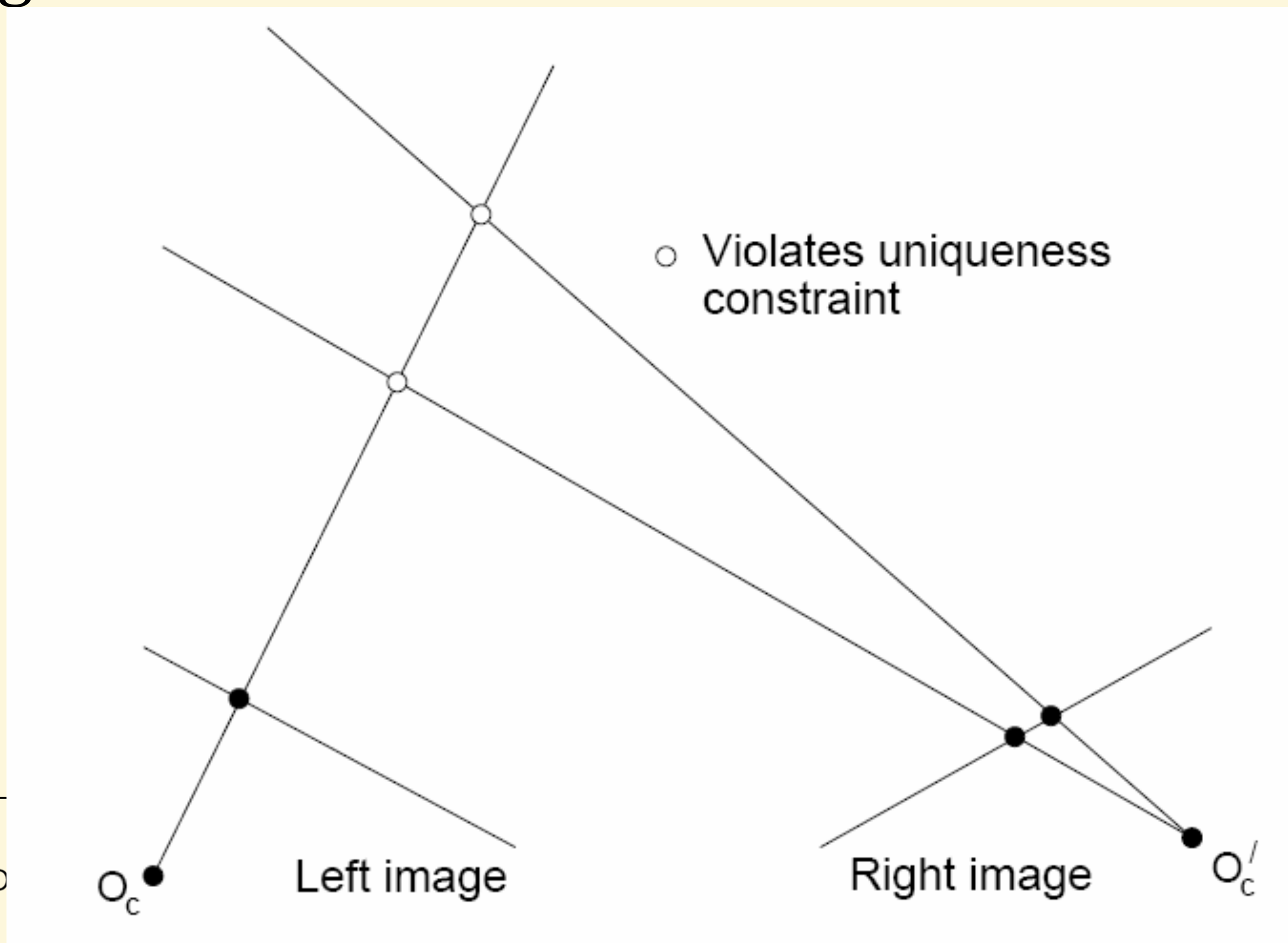
- The similarity constraint is **local** (each reference window is matched independently)
- Need to enforce **non-local** correspondence constraints

Correspondence problem

- Beyond the hard constraint of epipolar geometry, there are “soft” constraints to help identify corresponding points
  - Similarity
  - Uniqueness
  - Disparity gradient
  - Ordering

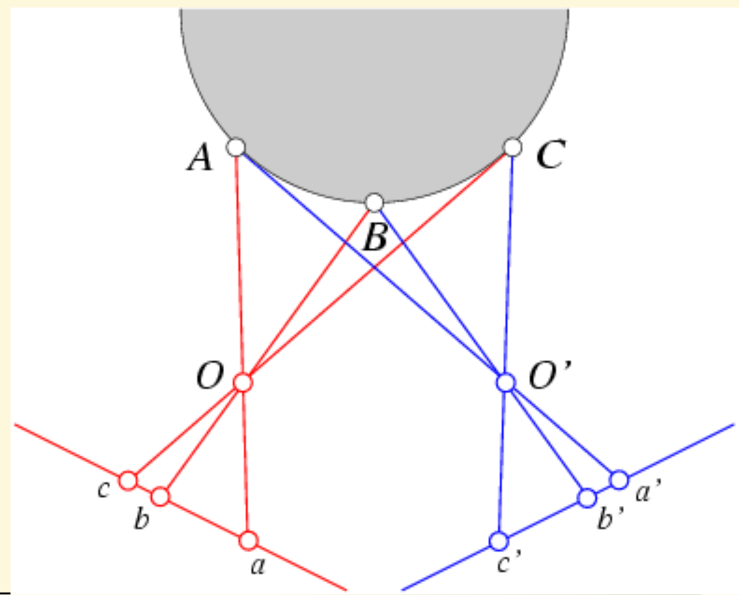
Non-local constraints

- Uniqueness = for any point in one image, there should be at most one matching point in the other image



Non-local constraints

- Uniqueness = for any point in one image, there should be at most one matching point in the other image
- Ordering = corresponding points should be in the same order in both views



Ordering constraint

- Points on **same surface** (opaque object) will be in same order in both views

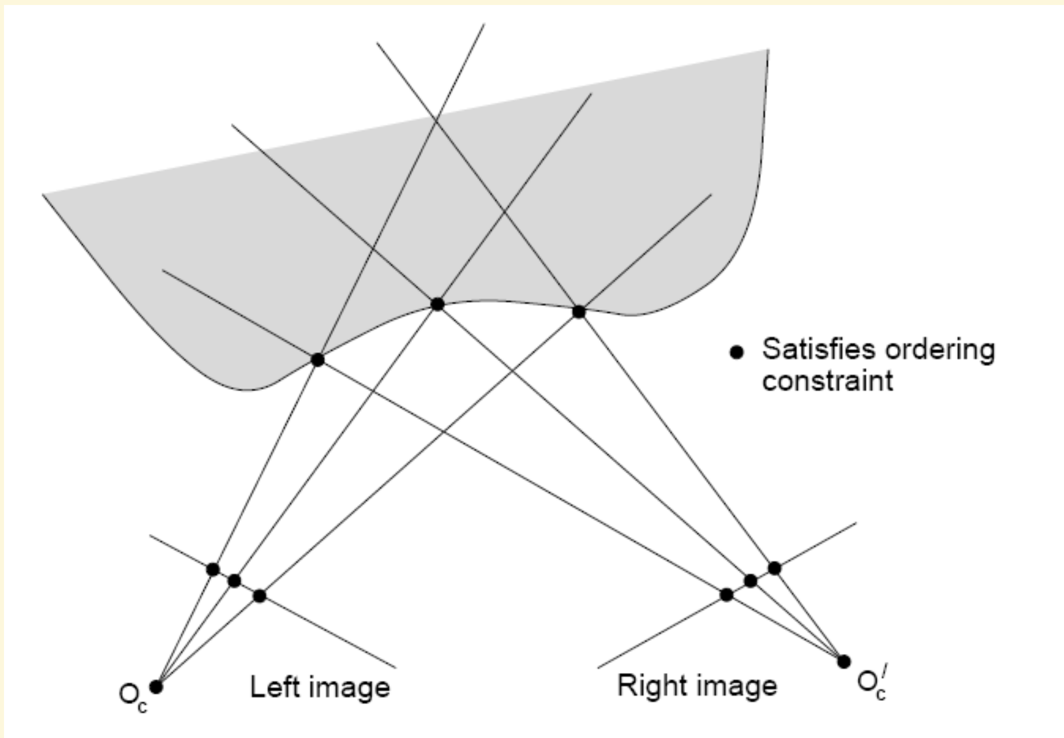
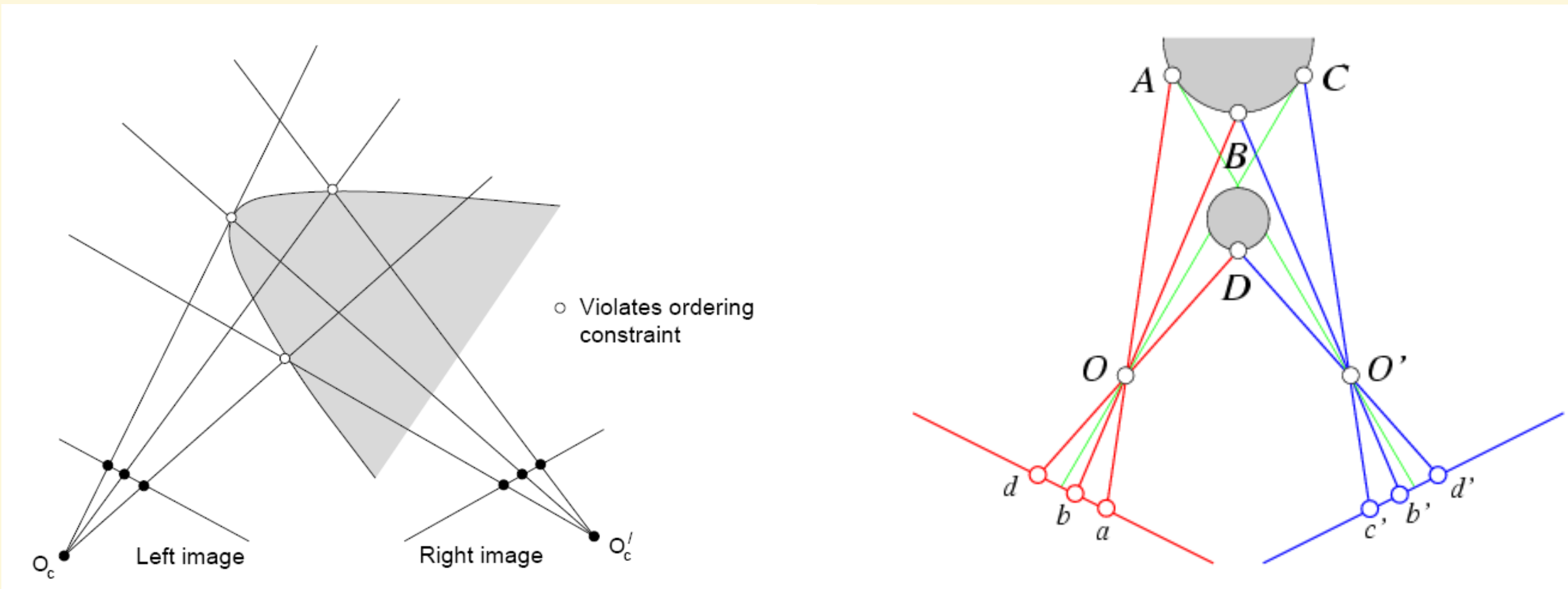


Figure from Gee & Cipolla 1999

Ordering constraint

- Won't always hold, e.g. consider transparent object, or an occluding surface



Figures from Forsyth & Ponce

Non-local constraints

- Uniqueness
  - For any point in one image, there should be at most one matching point in the other image
- Ordering
  - Corresponding points should be in the same order in both views
- Smoothness
  - We expect disparity values to change slowly (for the most part)

Disparity gradient constraint

Assume piecewise continuous surface, so want disparity estimates to be locally smooth

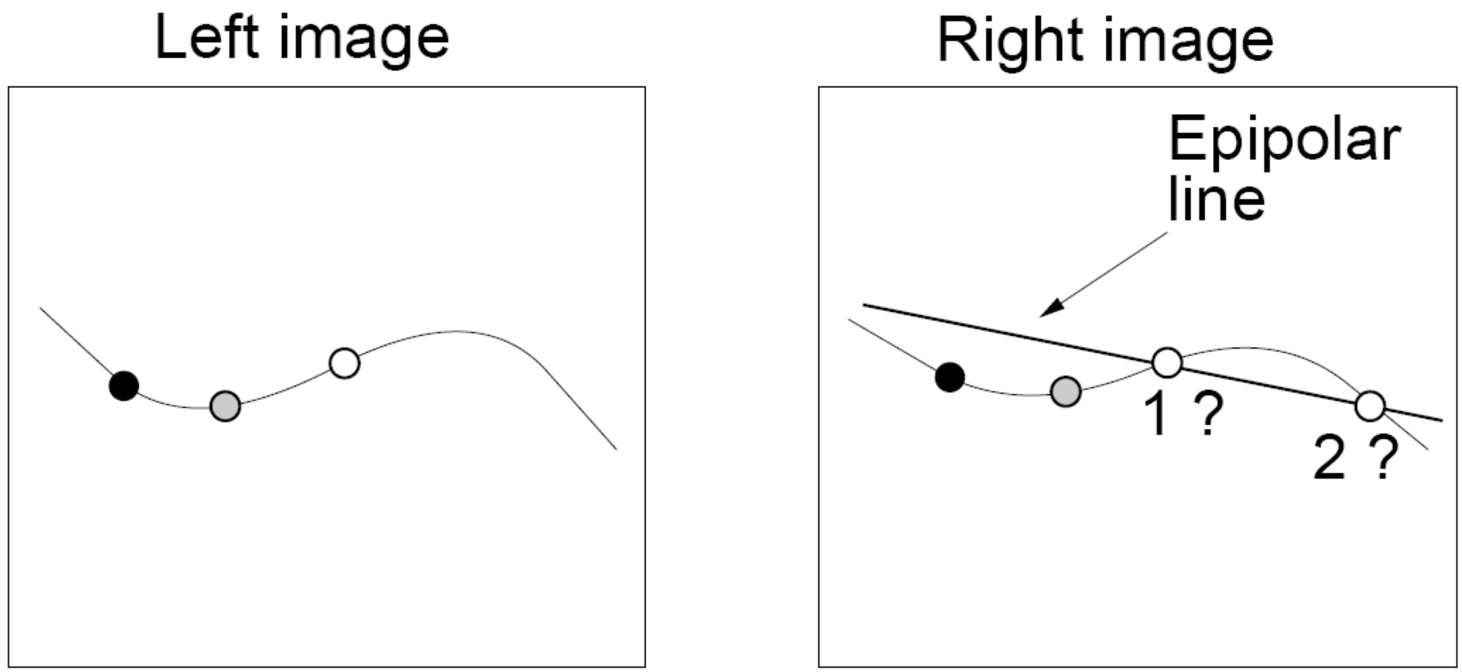
(a) each point in the left image corresponds to a unique point in the right image and vice versa;

(b) if we were given one image painted on a rubber sheet, we could, without tearing the sheet or gluing it to itself, deform it so that we obtained the other image.

Disparity gradient = disparity / distance

Figure from Gee & Cipolla 1999

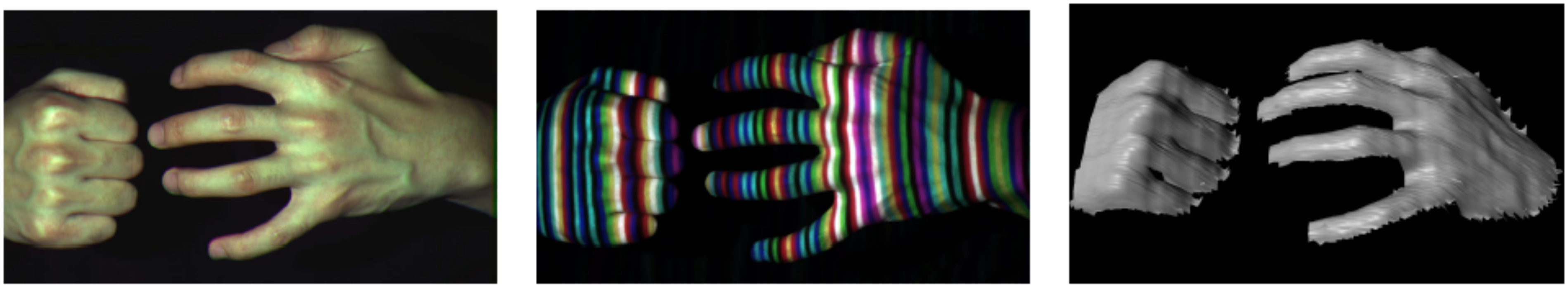
Disparity gradient constraint



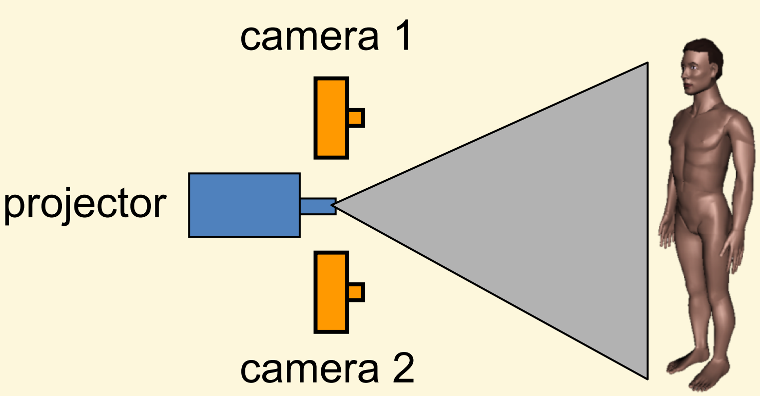
Given matches ● and ● , point ○ in the left image must match point 1 in the right image. Point 2 would exceed the disparity gradient limit.

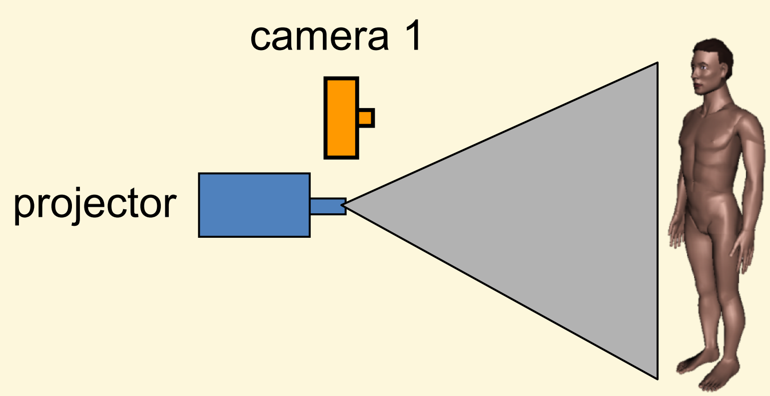
Figure from Gee & Cipolla 1999

Active stereo with structured light



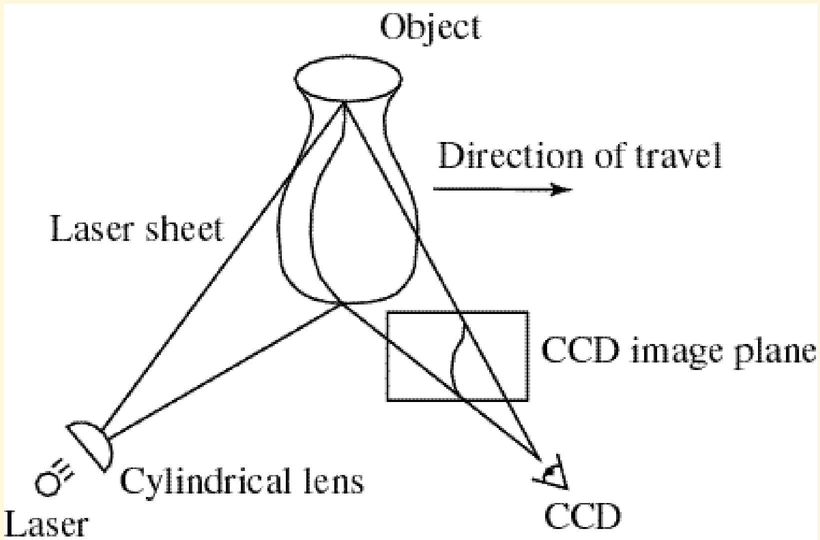
Li Zhang's one-shot stereo

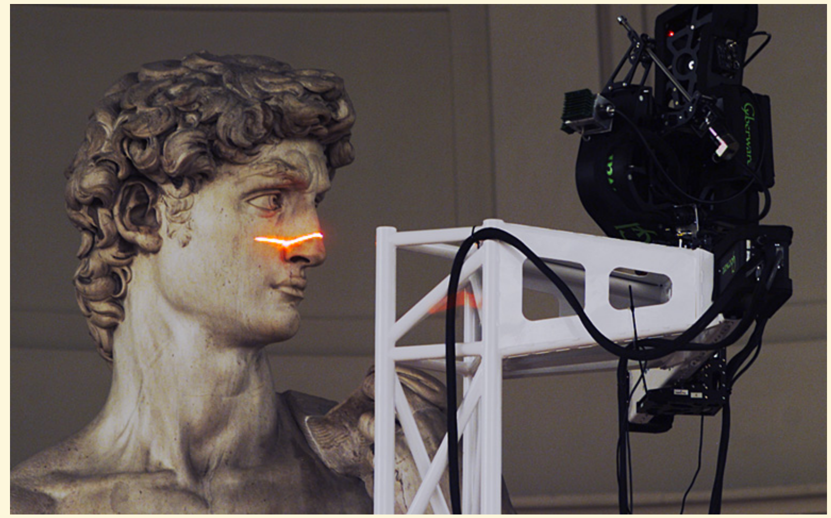




- Project "structured" light patterns onto the object
  - simplifies the correspondence problem

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