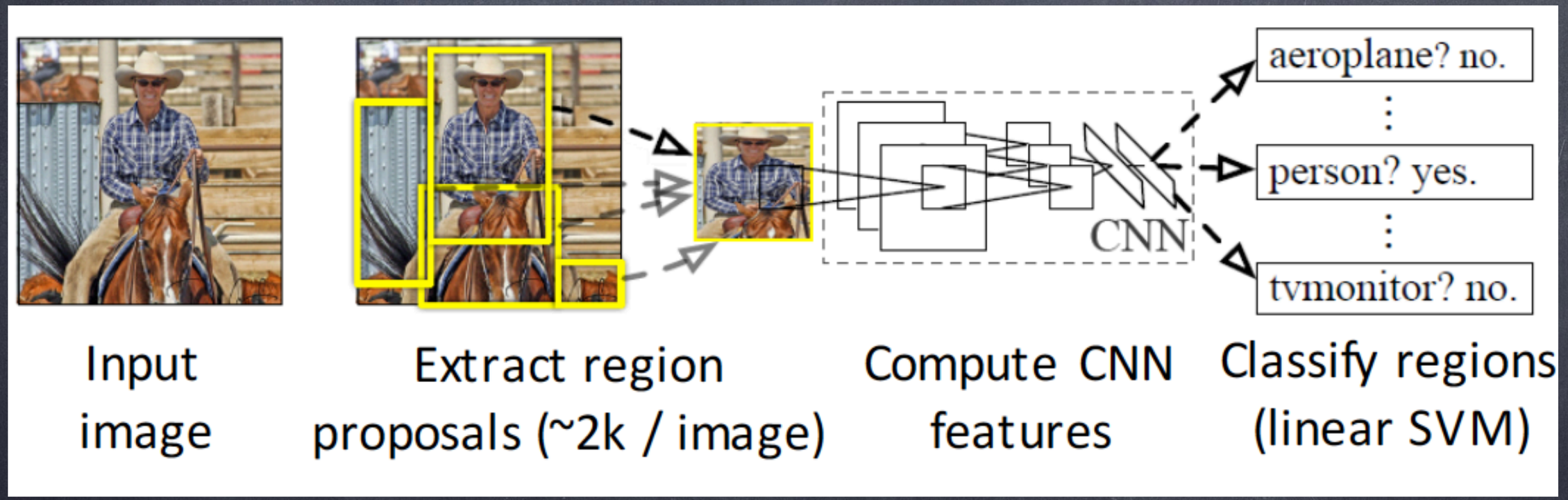
Computer Vision

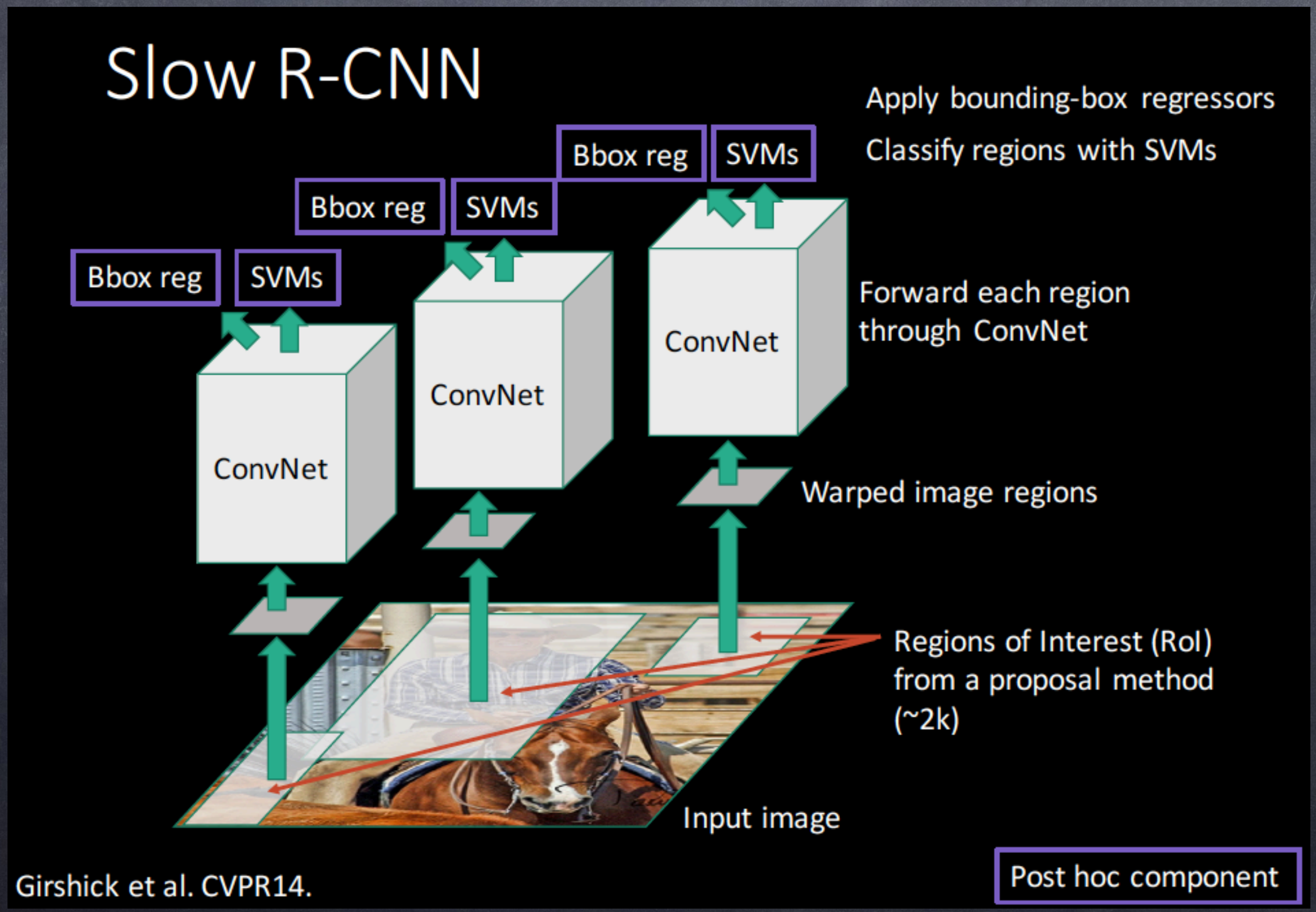
How do we use CNNs for Object Detection and Recognition?

Face Detection and Recognition

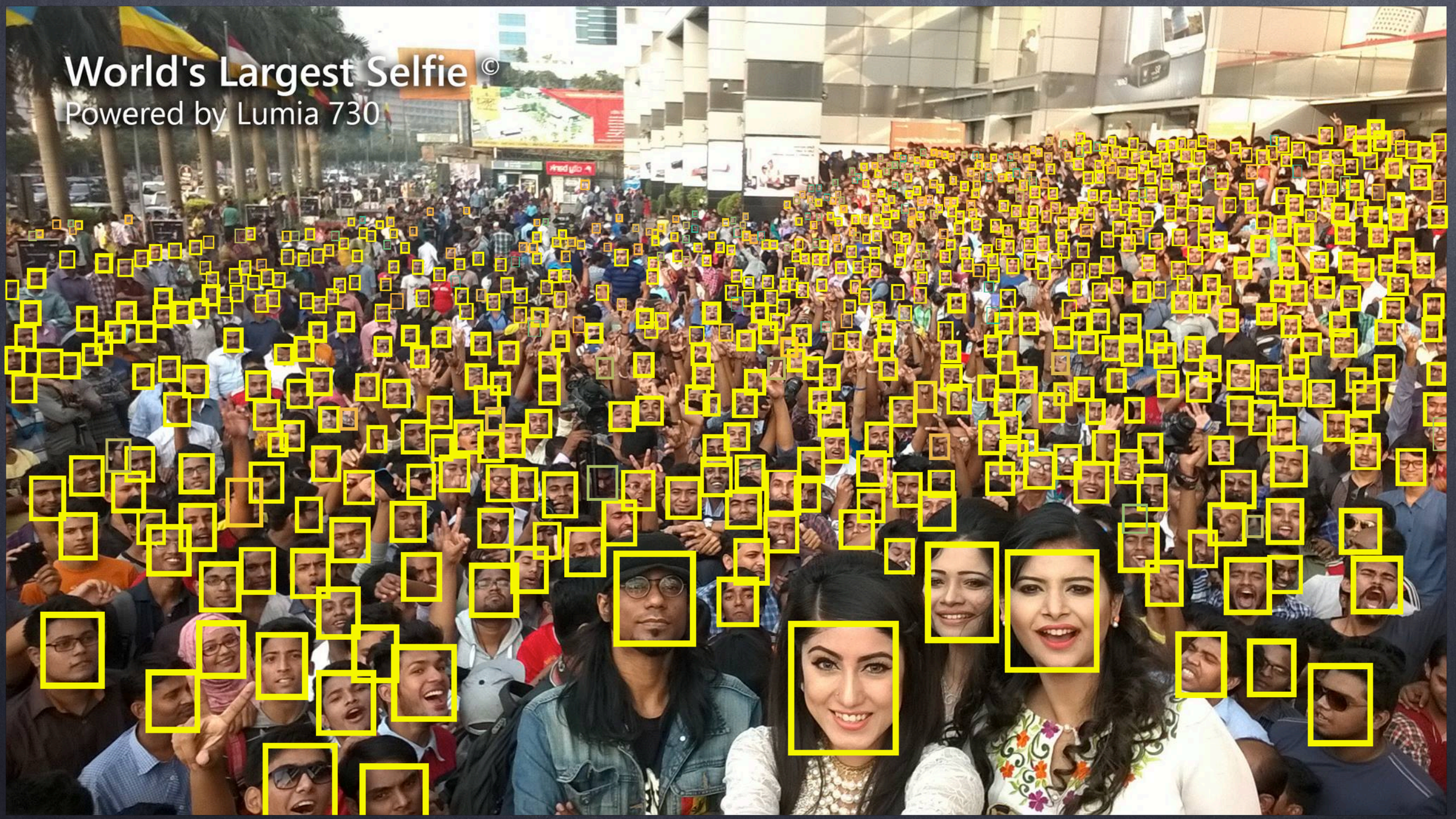
R-CNN (Original)



R-CNN (Original)



Face Detection using DNN



TinyFace Detector

Once you have detected faces ... then apply CNN

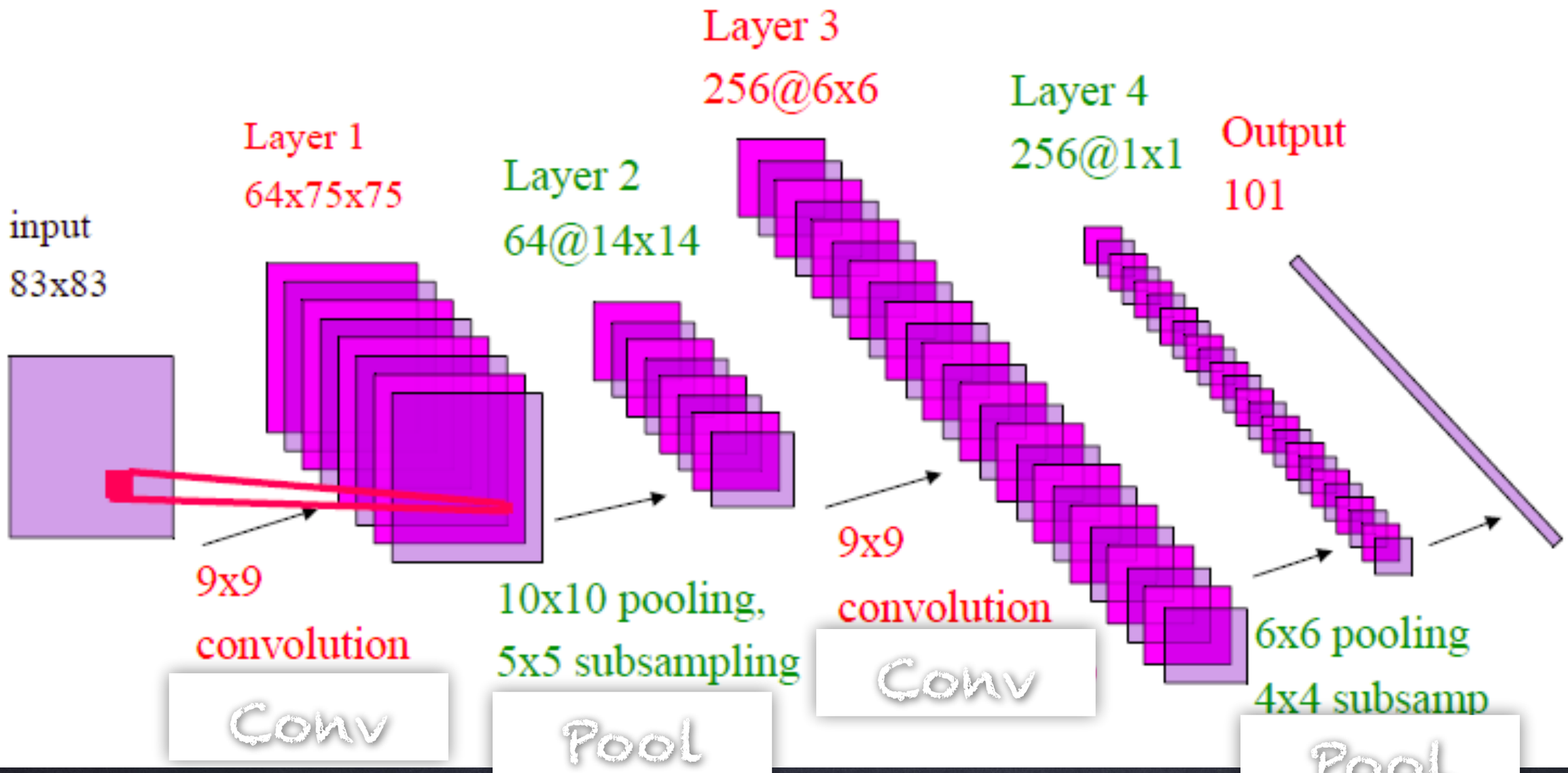


Image credit: Y. LeCun

Let us now move to next topic

- We will now dive down to 3D computer vision

What do you mean by Mosaic?

- Also popularly known as panorama or stitching

Image stitching

geometrical registration

photometric registration

- Stitching = alignment + blending

Image stitching

geometrical registration

photometric registration

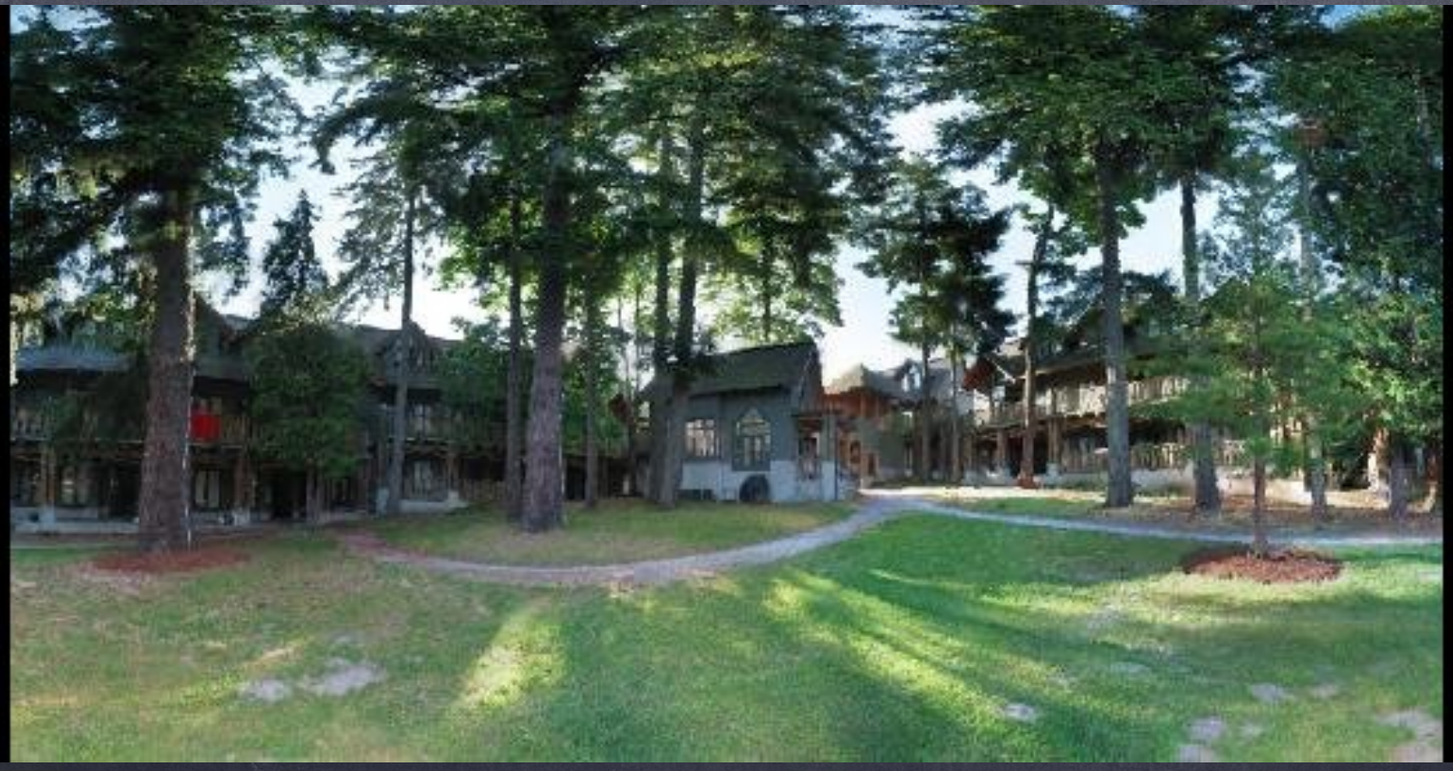
=

Why panorama?

- Are you getting the whole picture?
  - Compact Camera FOV = 50 x 35°
  -

Why panorama?

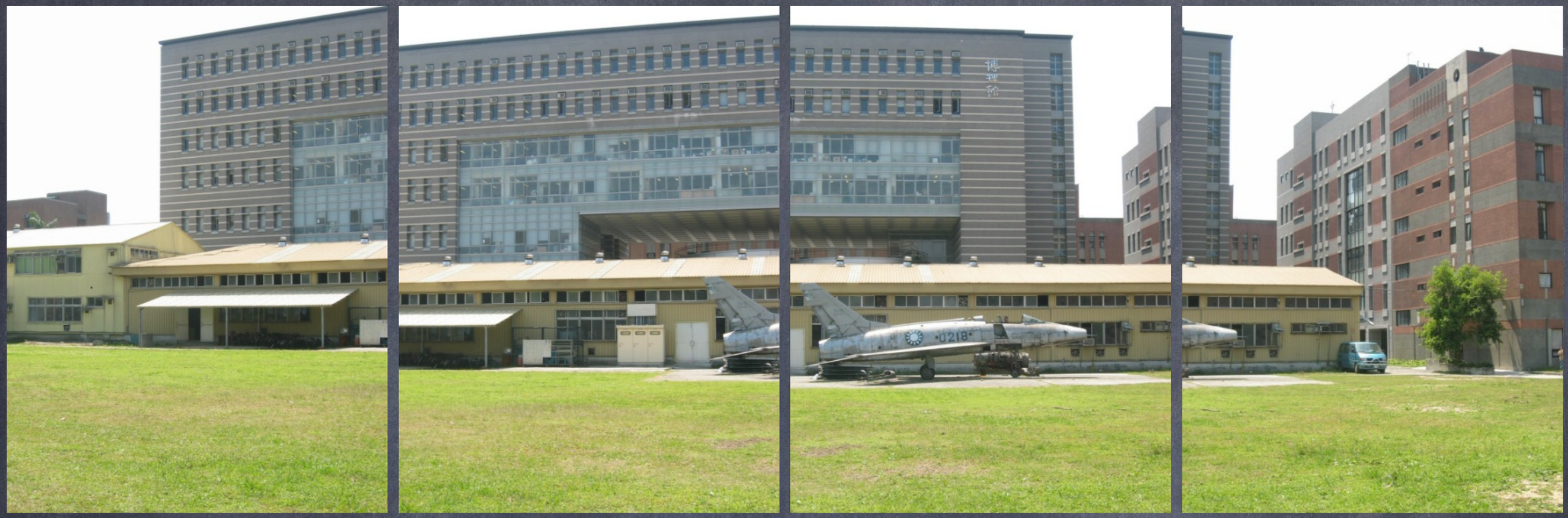
- Are you getting the whole picture?
  - Compact Camera FOV = 50 x 35°
  - Human FOV = 200 x 135°
  -



Why panorama?

- Are you getting the whole picture?
  - Compact Camera FOV = 50 x 35°
  - Human FOV = 200 x 135°
  - Panoramic Mosaic = 360 x 180°

Panorama creation



How to do it?

Basic Procedure

- Take a sequence of images from the same position
  - Rotate the camera about its optical center
- Compute transformation between second image and first
- Shift the second image to overlap with the first
- Blend the two together to create a mosaic
- If there are more images, repeat

Keyword 1: Transformation

Projective transformations

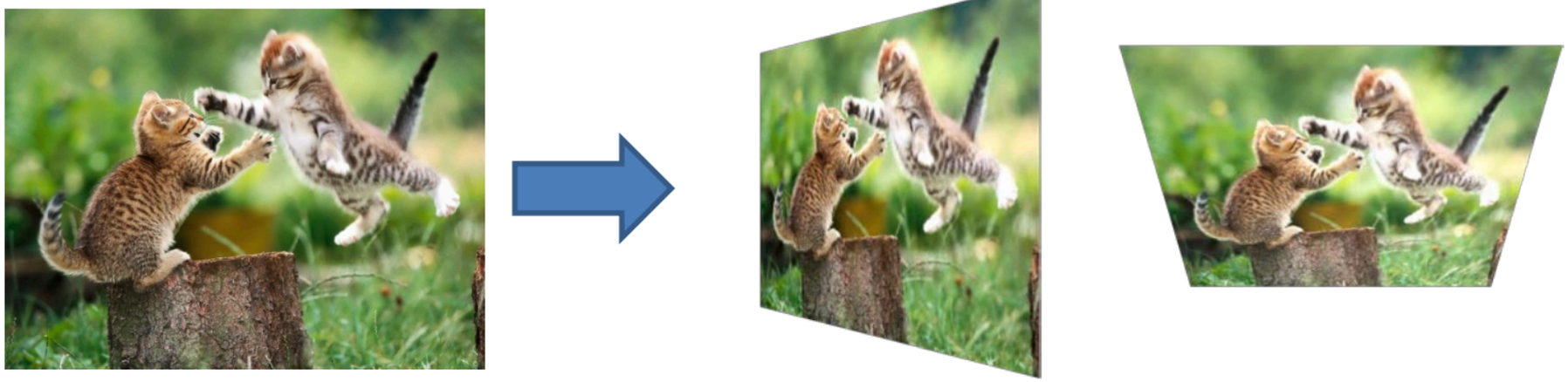
- (aka *homographies*)

homogeneous coordinates

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

$$x' = u/w$$

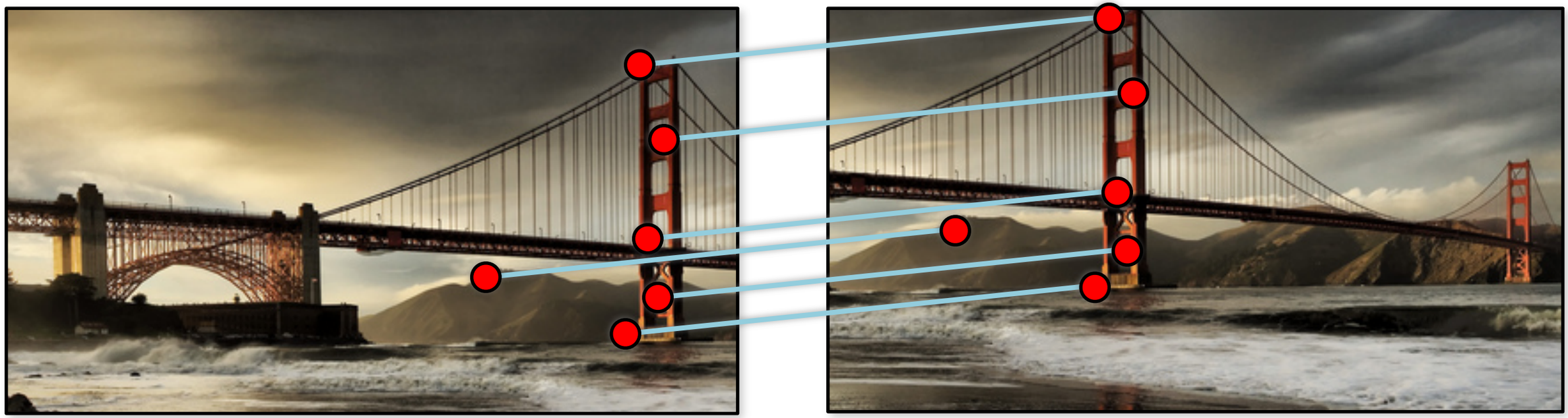
$$y' = v/w$$



Computing transformations

?

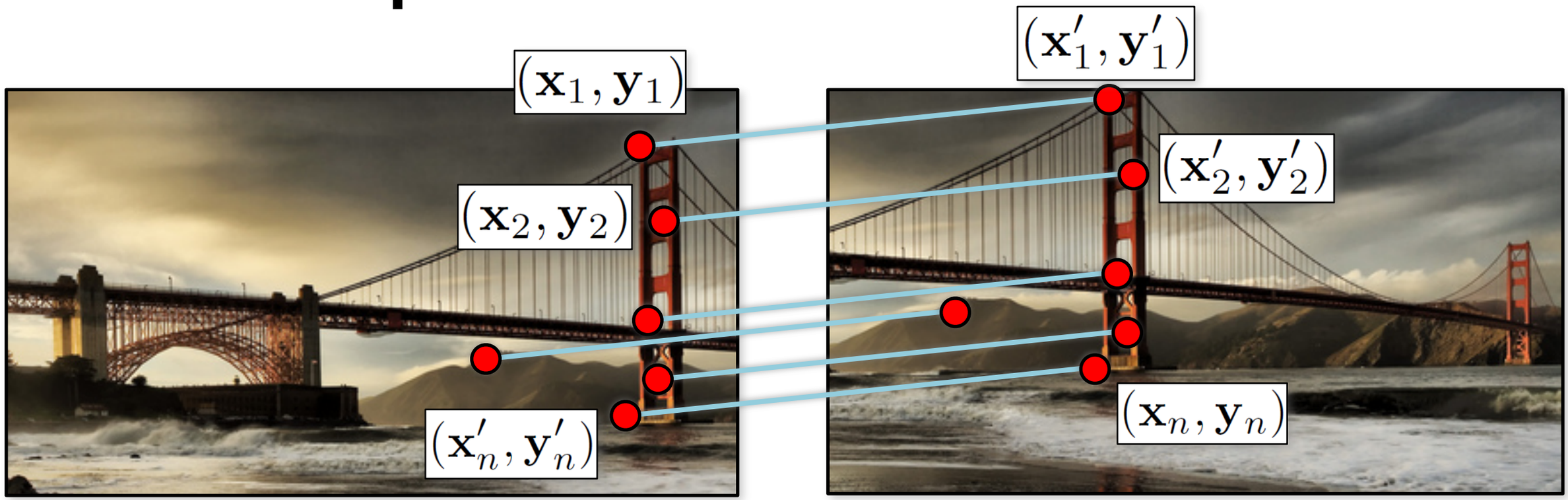
Simple case: translations



$(\mathbf{x}_t, \mathbf{y}_t)$

How do we solve for $(\mathbf{x}_t, \mathbf{y}_t)$?

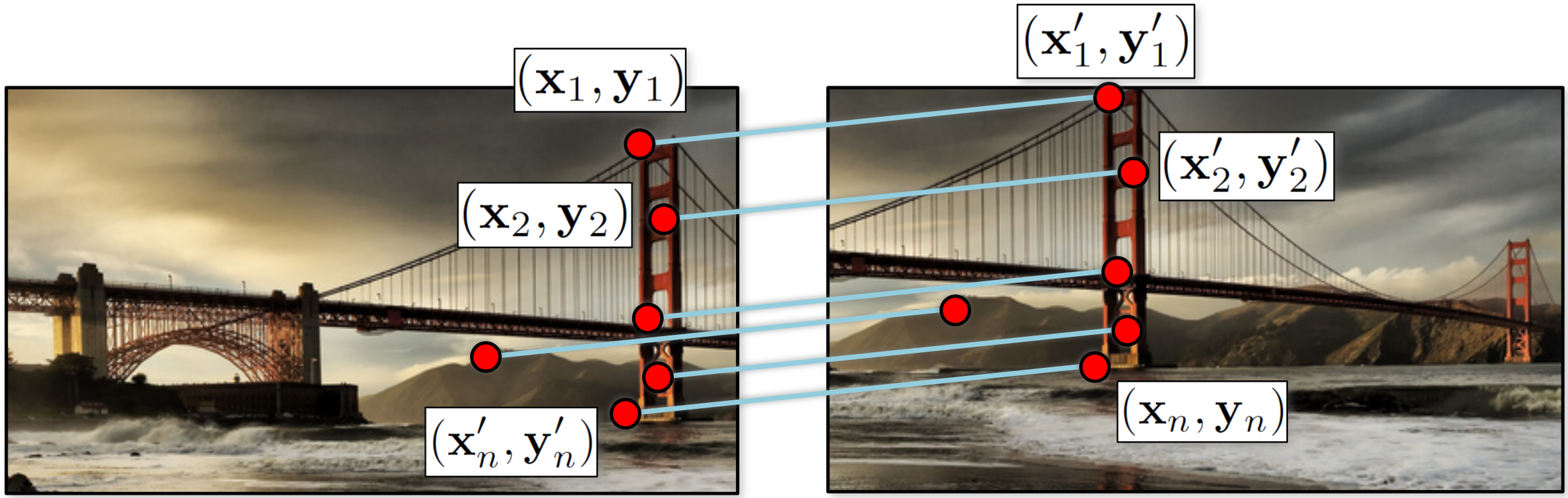
Simple case: translations



Displacement of match *i* $(\mathbf{x}'_i - \mathbf{x}_i, \mathbf{y}'_i - \mathbf{y}_i)$

$$(\mathbf{x}_t, \mathbf{y}_t) = \left(\frac{1}{n} \sum_{i=1}^{n} \mathbf{x}'_i - \mathbf{x}_i, \frac{1}{n} \sum_{i=1}^{n} \mathbf{y}'_i - \mathbf{y}_i \right)$$

Simple case: translations

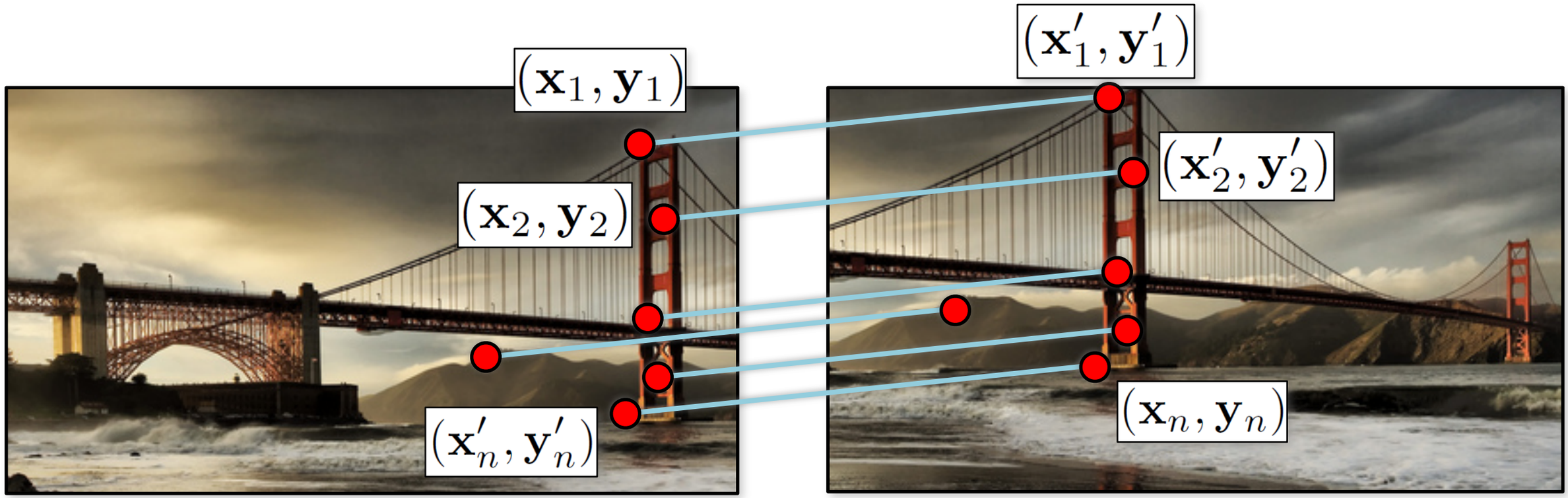


$$\mathbf{x}_i + \mathbf{x_t} = \mathbf{x}'_i$$

$$\mathbf{y}_i + \mathbf{y_t} = \mathbf{y}'_i$$

- System of linear equations
  - What are the knowns? Unknowns?
  - How many unknowns? How many equations (per match)?

Simple case: translations



$$\mathbf{x}_i + \mathbf{x_t} = \mathbf{x}'_i$$

$$\mathbf{y}_i + \mathbf{y_t} = \mathbf{y}'_i$$

- Problem: more equations than unknowns
  - "Overdetermined" system of equations
  - We will find the *least squares* solution

Least squares formulation

- For each point $(\mathbf{x}_i, \mathbf{y}_i)$

$$\begin{aligned} \mathbf{x}_i + \mathbf{x_t} &= \mathbf{x}'_i \\ \mathbf{y}_i + \mathbf{y_t} &= \mathbf{y}'_i \end{aligned}$$

- we define the *residuals* as

$$\begin{aligned} r_{\mathbf{x}_i}(\mathbf{x}_t) &= (\mathbf{x}_i + \mathbf{x}_t) - \mathbf{x}'_i \\ r_{\mathbf{y}_i}(\mathbf{y}_t) &= (\mathbf{y}_i + \mathbf{y}_t) - \mathbf{y}'_i \end{aligned}$$

Least squares formulation

- Goal: minimize sum of squared residuals

$$C(\mathbf{x}_t, \mathbf{y}_t) = \sum_{i=1}^{n} \left(r_{\mathbf{x}_i}(\mathbf{x}_t)^2 + r_{\mathbf{y}_i}(\mathbf{y}_t)^2 \right)$$

- “Least squares” solution

Least squares

$$\mathbf{At} = \mathbf{b}$$

- Find $\mathbf{t}$ that minimizes

$$||\mathbf{At} - \mathbf{b}||^2$$

- To solve, form the *normal equations*

$$\mathbf{A}^{\mathrm{T}}\mathbf{At} = \mathbf{A}^{\mathrm{T}}\mathbf{b}$$

$$\mathbf{t} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{b}$$

Least squares formulation

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ \vdots & \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \end{bmatrix} = \begin{bmatrix} x'_1 - x_1 \\ y'_1 - y_1 \\ x'_2 - x_2 \\ y'_2 - y_2 \\ \vdots \\ x'_n - x_n \\ y'_n - y_n \end{bmatrix}$$

$$\underset{2n \times 2}{\mathbf{A}} \quad \underset{2 \times 1}{\mathbf{t}} = \underset{2n \times 1}{\mathbf{b}}$$

What about affine

Affine transformations

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- How many unknowns?
- How many equations per match?
- How many matches do we need?

Affine transformations

- Residuals:

$$r_{x_i}(a,b,c,d,e,f) = (ax_i + by_i + c) - x'_i$$
$$r_{y_i}(a,b,c,d,e,f) = (dx_i + ey_i + f) - y'_i$$

- Cost function:

$$C(a,b,c,d,e,f) = \sum_{i=1}^{n} \left(r_{x_i}(a,b,c,d,e,f)^2 + r_{y_i}(a,b,c,d,e,f)^2 \right)$$

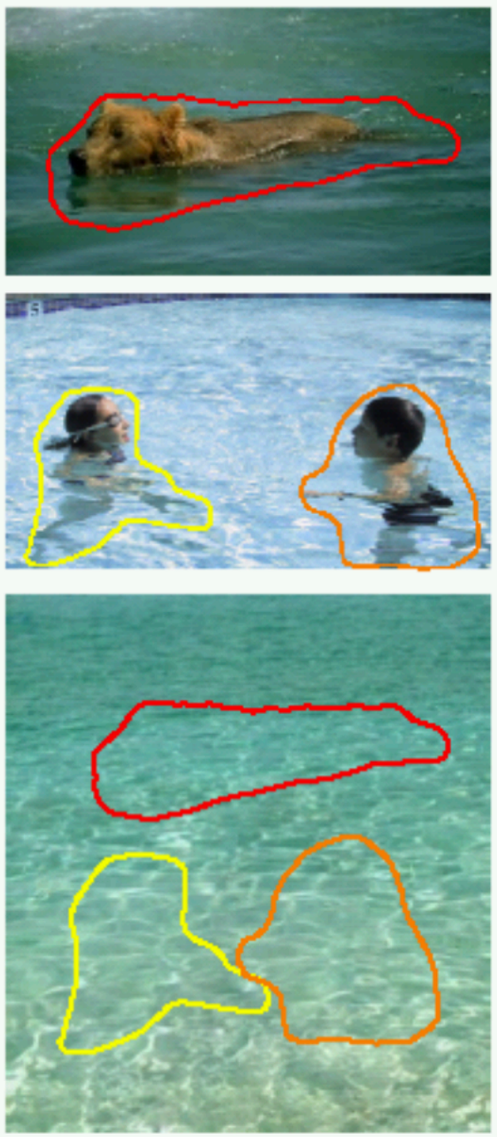
Affine transformations

- Matrix form

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \vdots \\ x'_n \\ y'_n \end{bmatrix}$$

$$\underset{2n \times 6}{\mathbf{A}} \qquad \underset{6 \times 1}{\mathbf{t}} = \underset{2n \times 1}{\mathbf{b}}$$

Keyword 2: Blending



sources/destinations

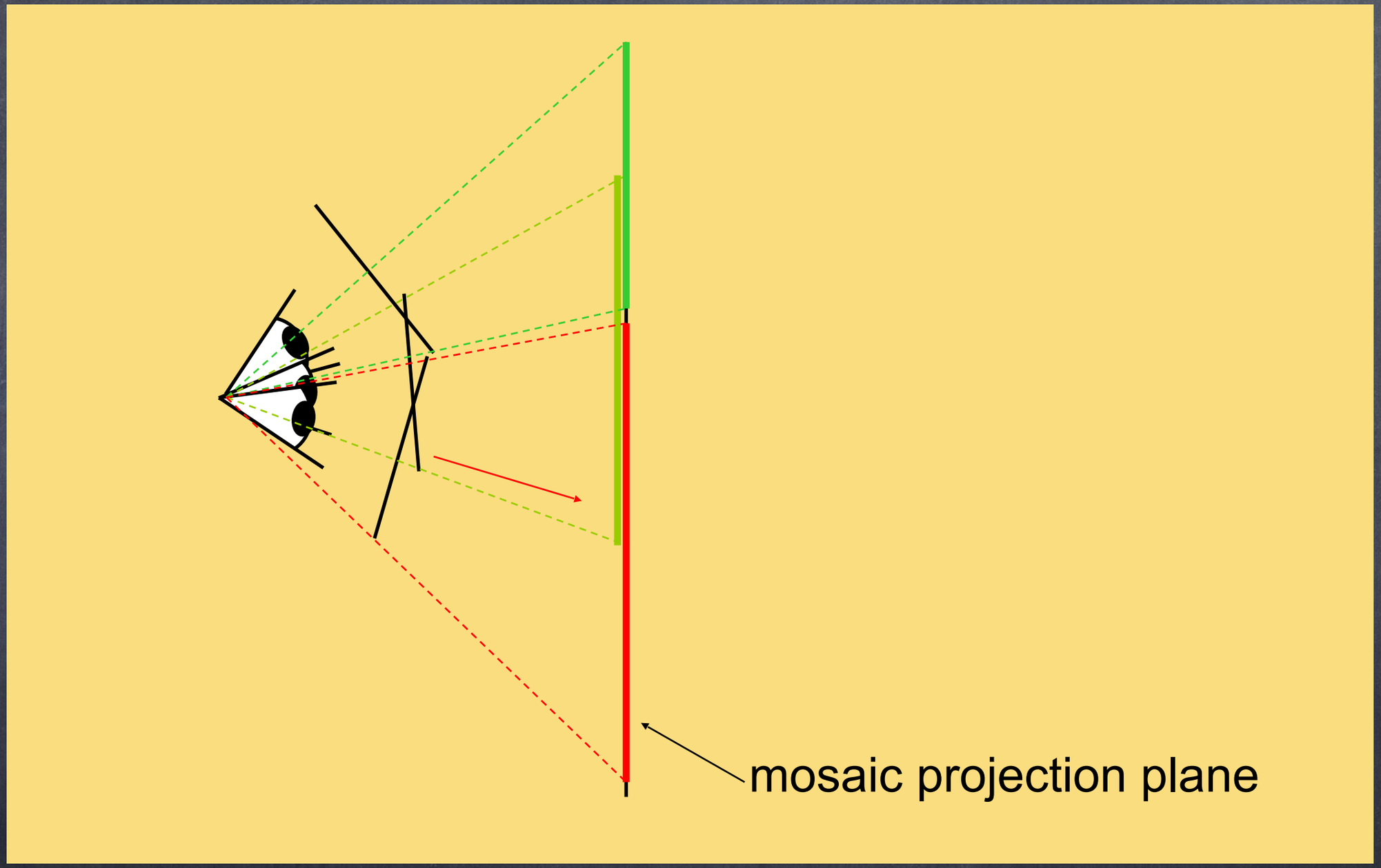
cloning

seamless cloning

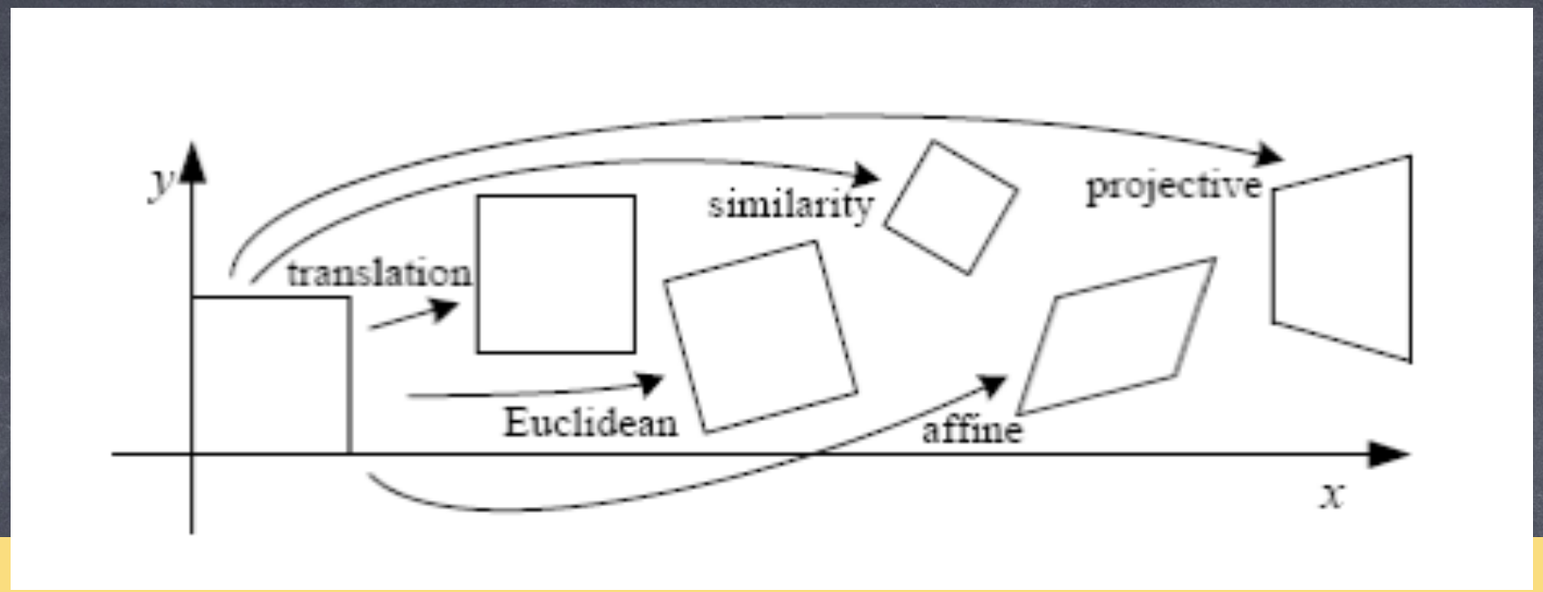
How Blending Works?

Recap

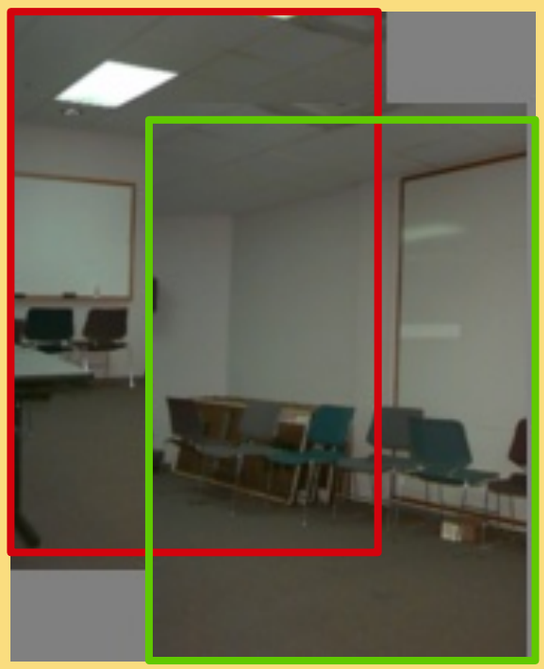
Recap



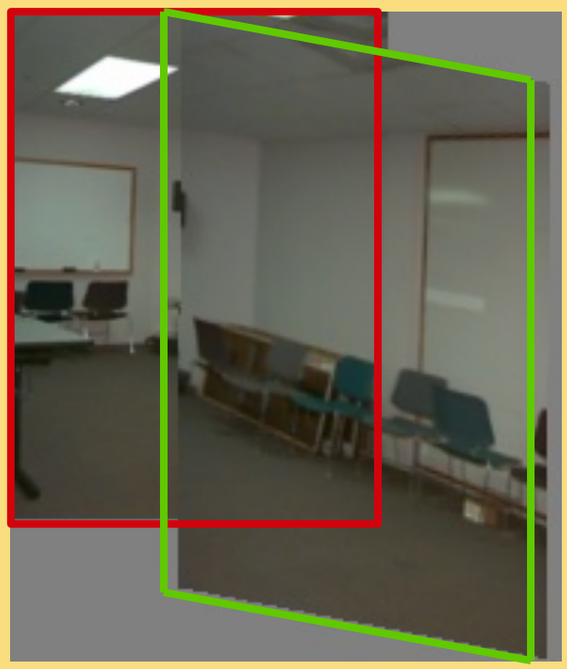
- The images are reprojected onto a common plane
- The mosaic is formed on this plane
- Mosaic is a *synthetic wide-angle camera*



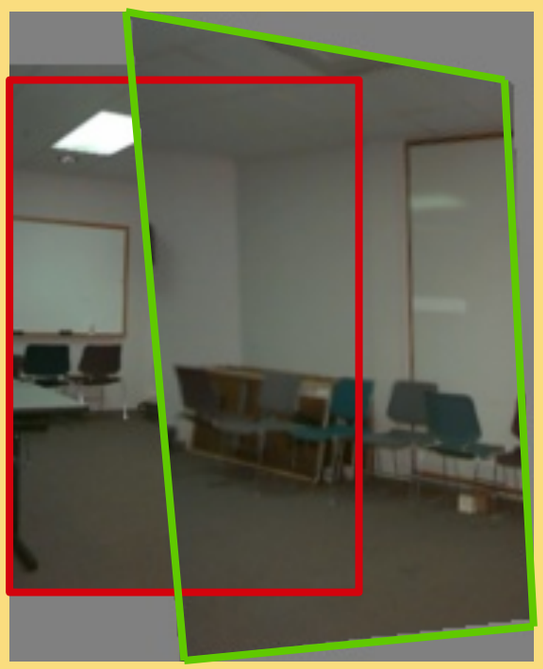
Translation **Affine** **Perspective** **Rotation**



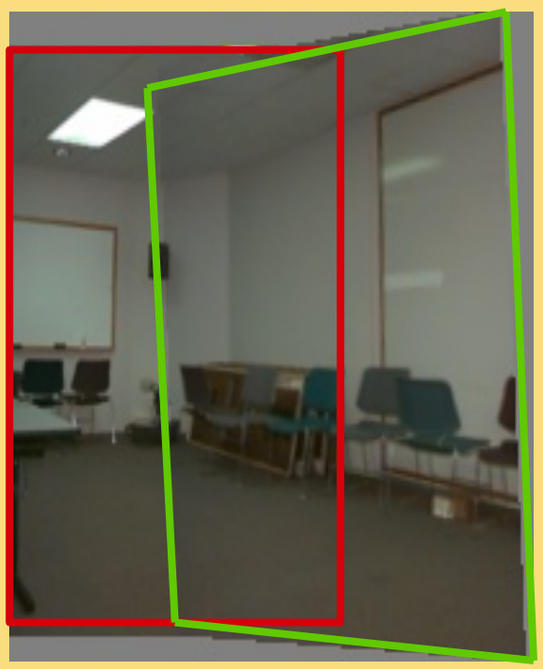
2 unknowns



6 unknowns



8 unknowns



3 unknowns

Full-view Panorama

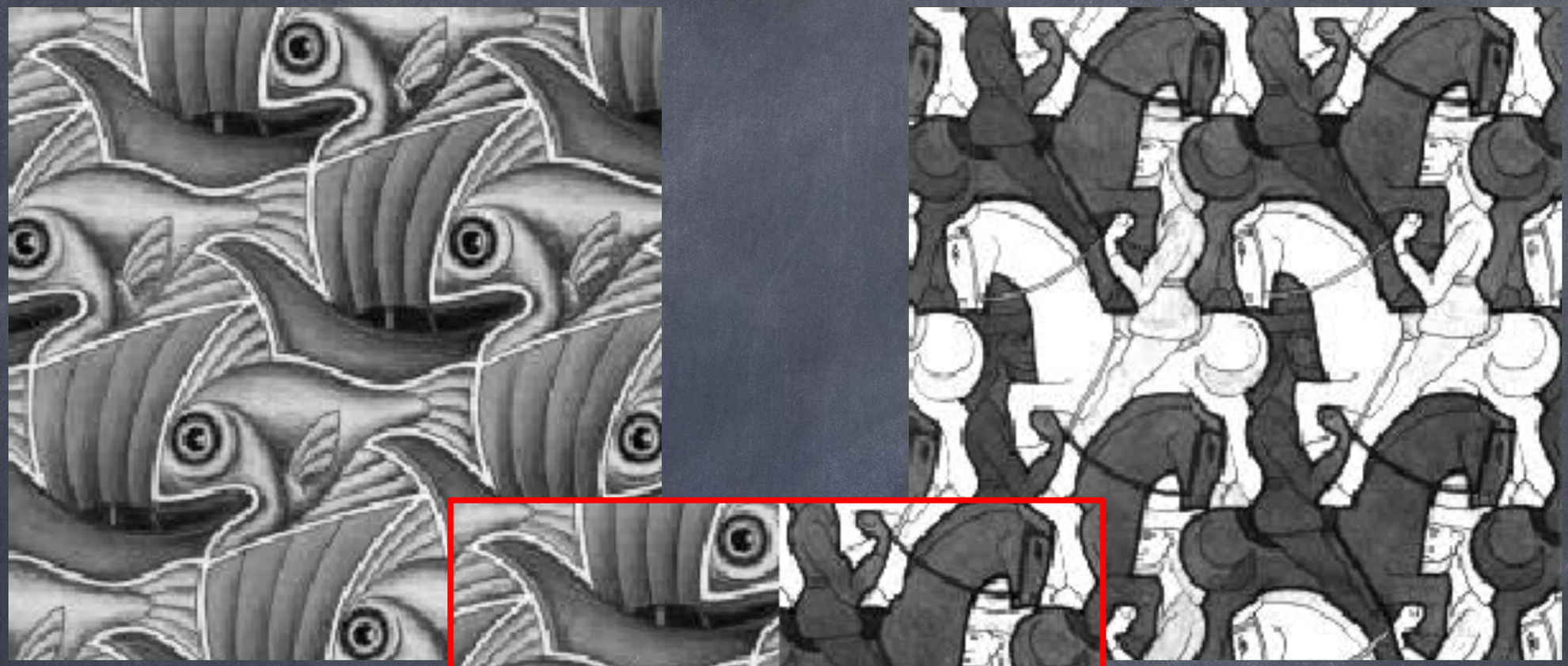

+

+

+

+


Image Blending



Feathering

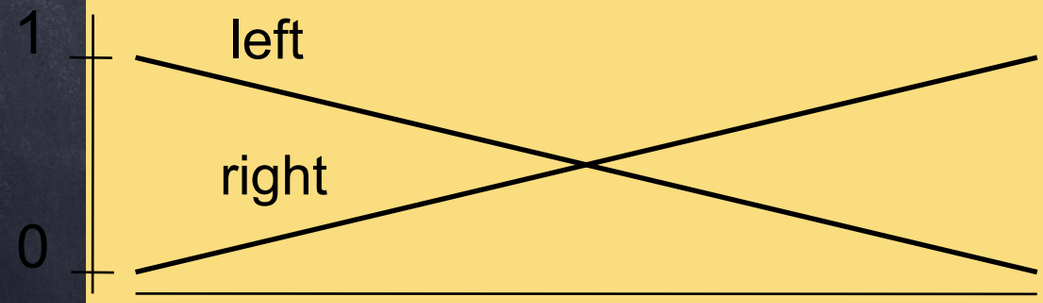
1
0

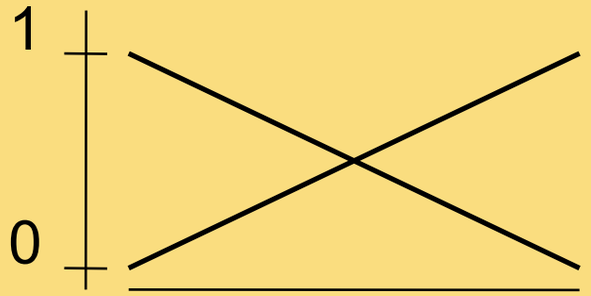
+

1
0

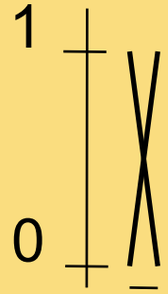
=

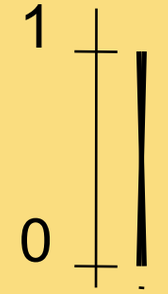
Effect of window (ramp-width) size



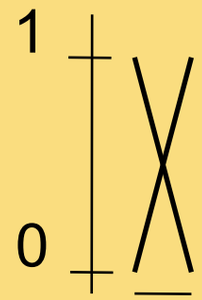


Effect of window size





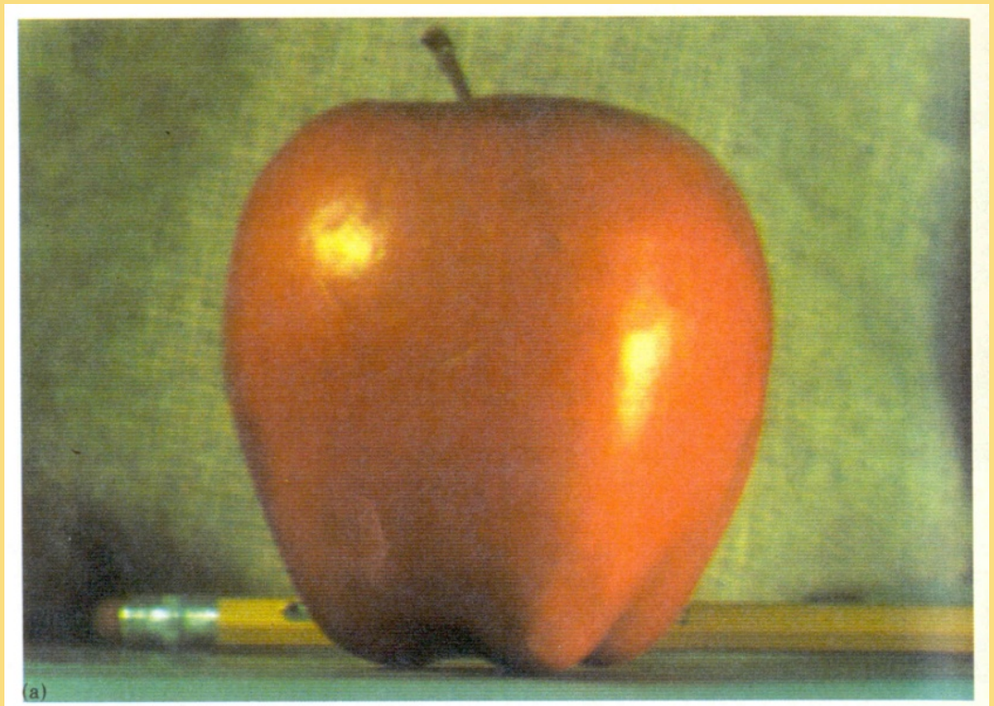
Good window size

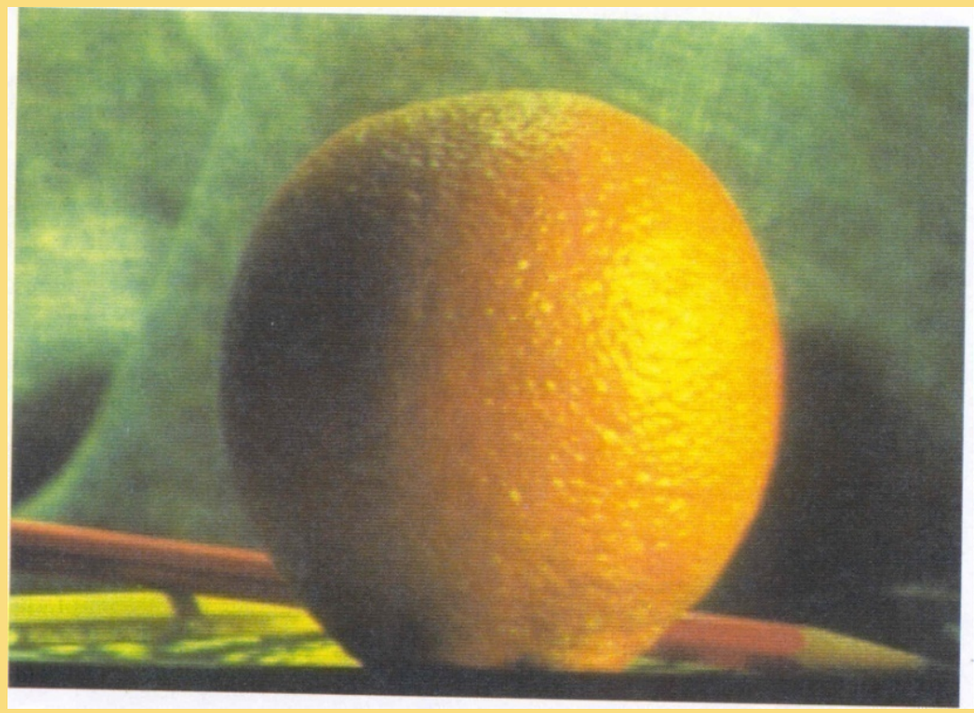


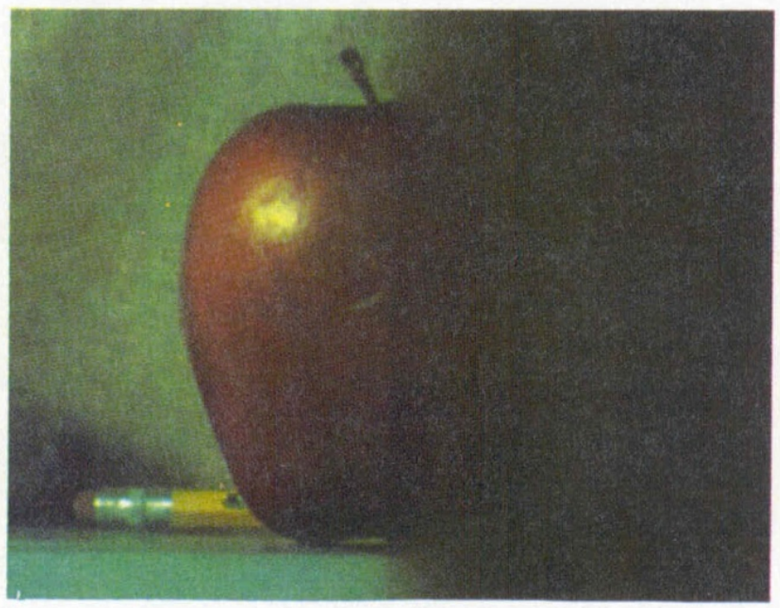
“Optimal” window: smooth but not ghosted

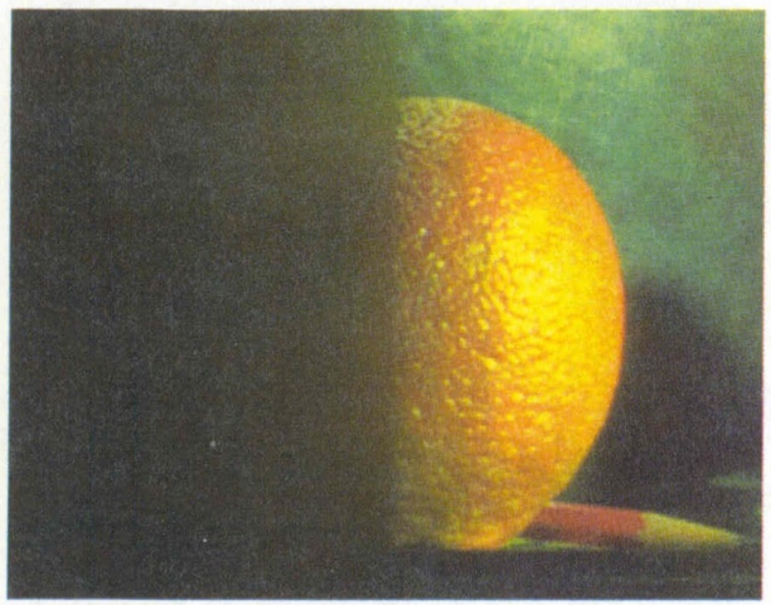
- Doesn’t always work...

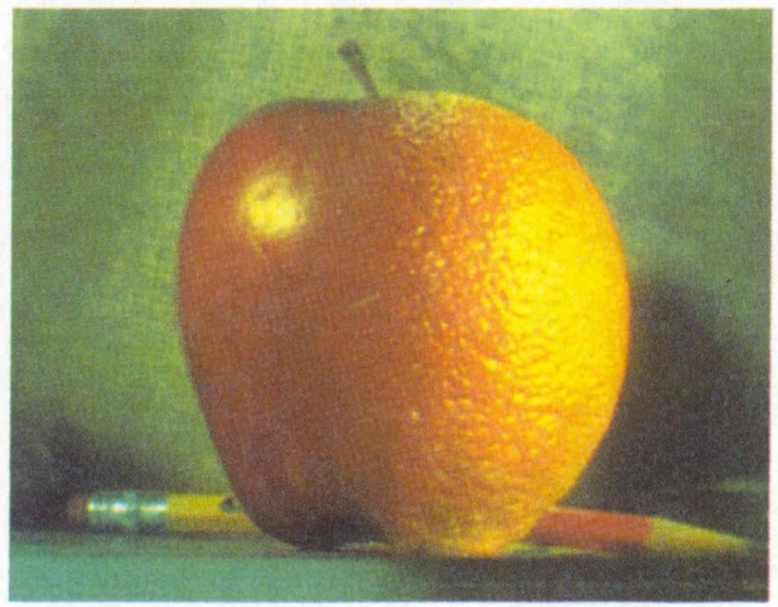
Pyramid blending






(d)


(h)

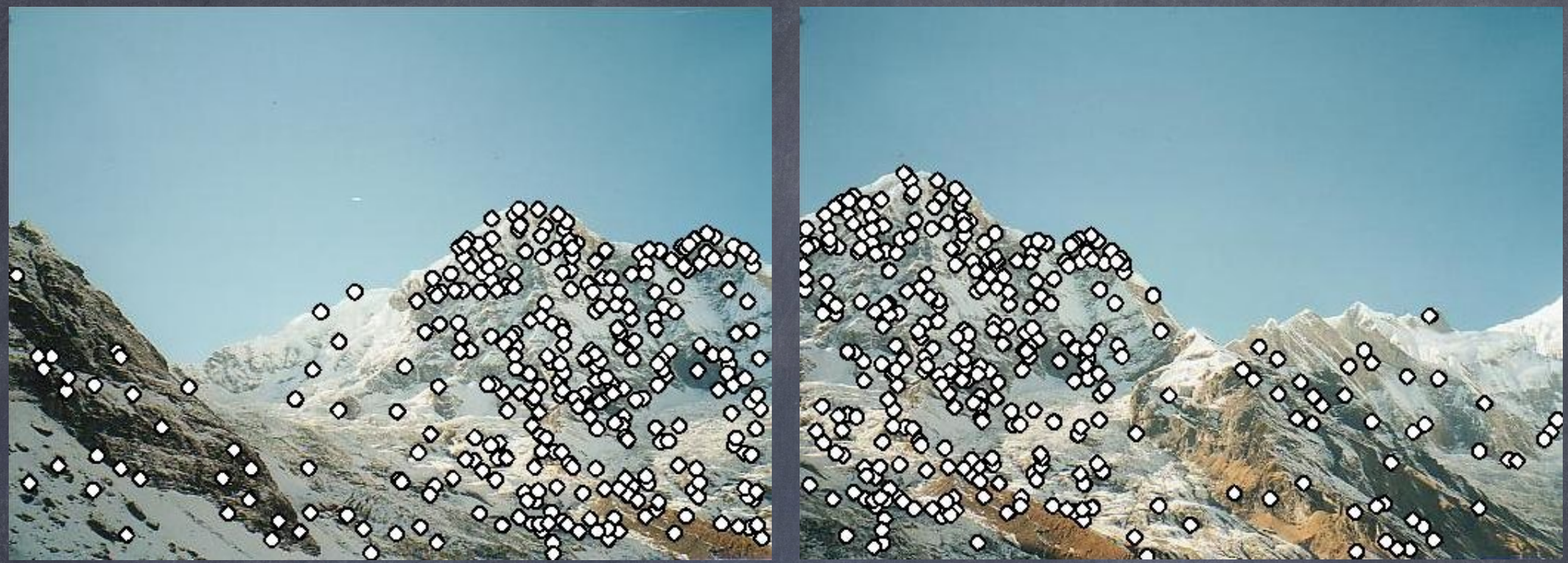

(l)

Create a Laplacian pyramid, blend each level

- Burt, P. J. and Adelson, E. H., A multiresolution spline with applications to image mosaics, ACM Transactions on Graphics, 42(4), October 1983, 217-236.

Feature-based alignment

Feature-based alignment

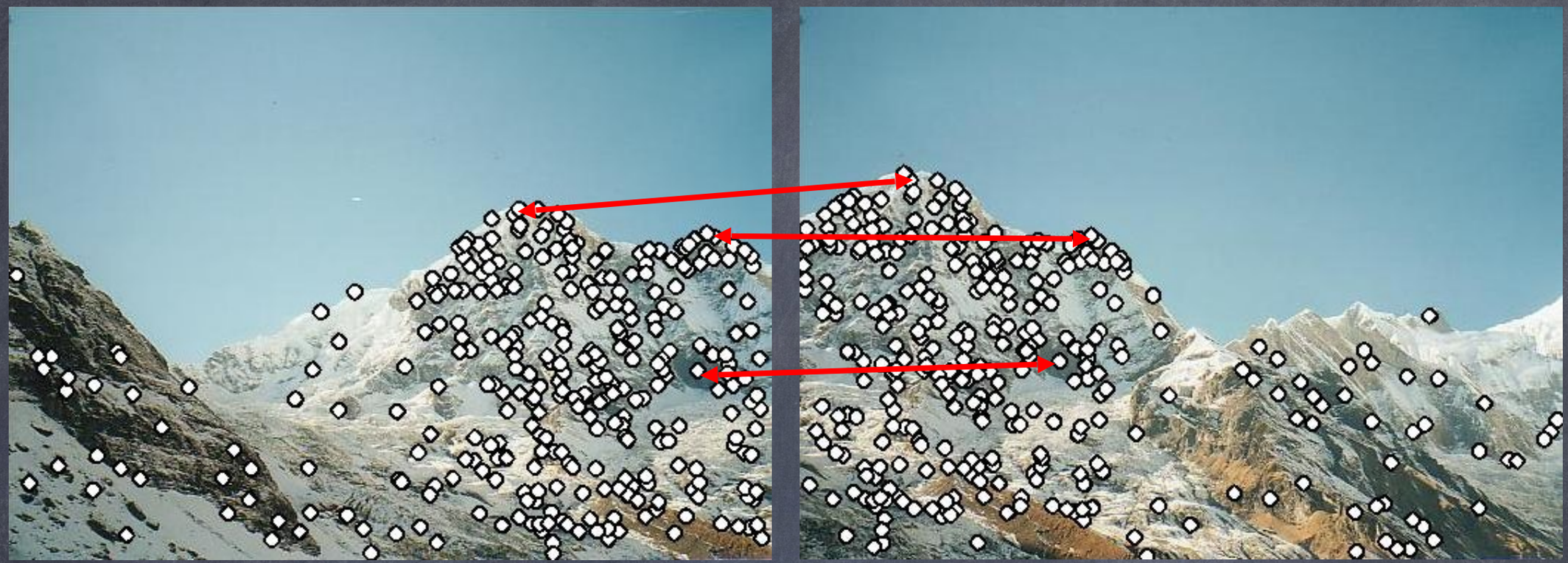


- Extract features

Feature-based alignment

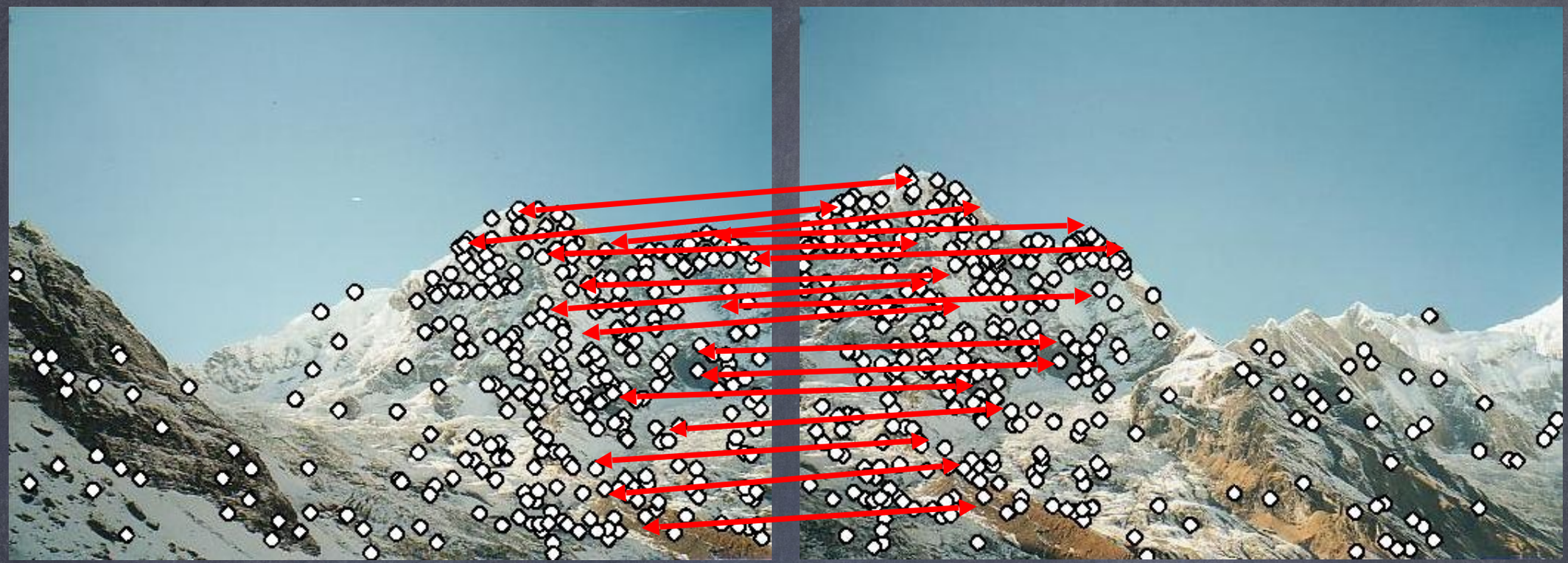
- Extract features
- Compute *putative matches*

Feature-based alignment



- Extract features
- Compute *putative matches*
- Loop

Feature-based alignment

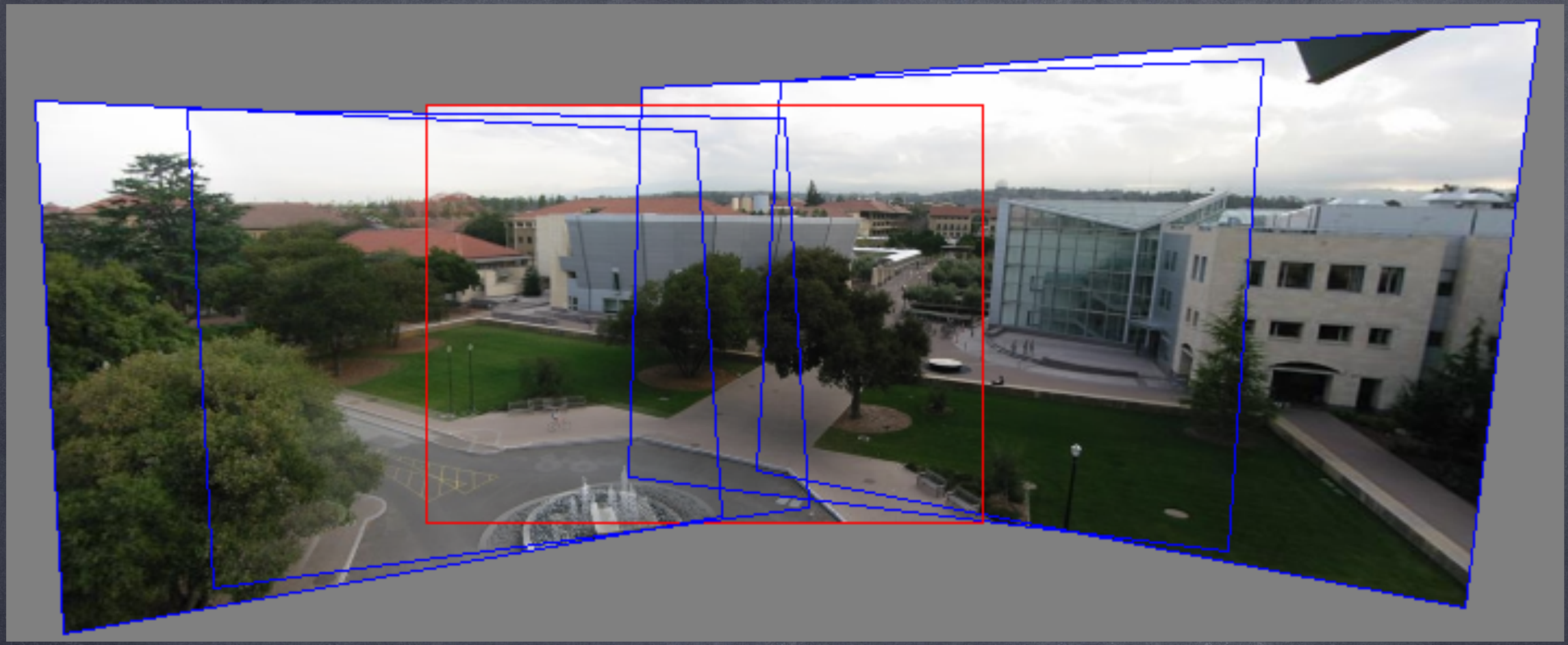


- Extract features
- Compute *putative matches*
- Loop

Feature-based alignment

- Extract features
- Compute *putative matches*
- Loop

Panoramas in Nutshell

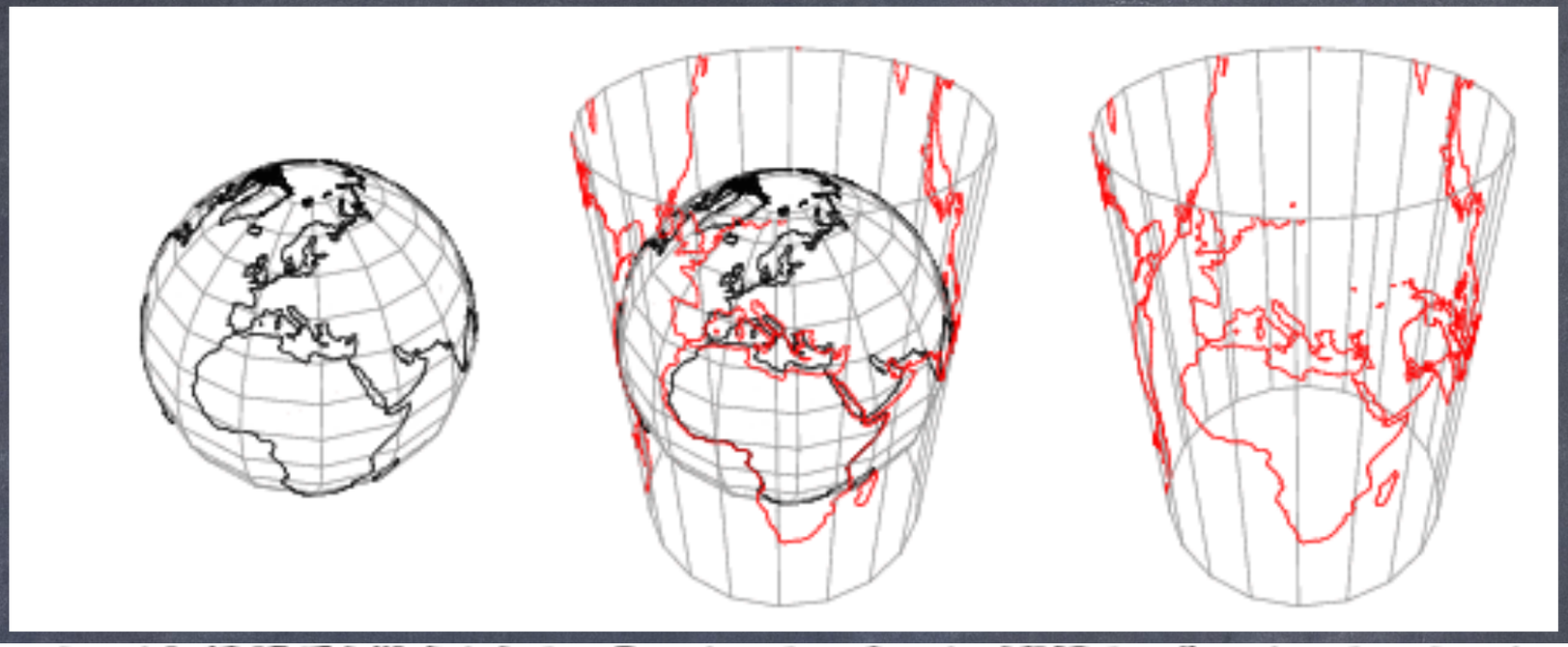


1. Pick one image (red)
2. Warp the other images towards it (usually, one by one)
3. blend

Other types of mosaics

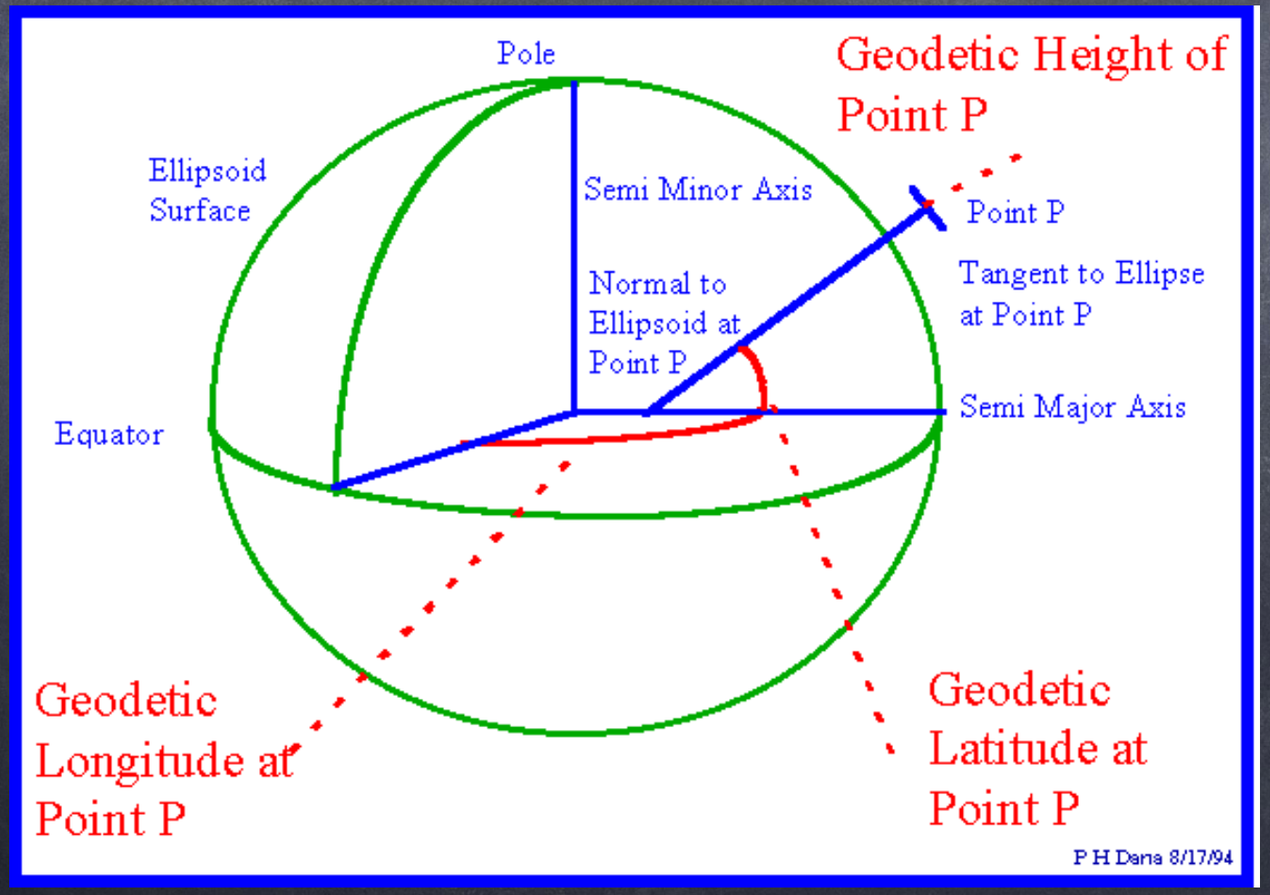
Can mosaic onto *any* surface if you know the geometry

Cylindrical projection



Ellipsoid (Global) Coordinate Systems

- Global coordinates based upon "spherical" coordinates modified to account for imperfect shape of earth.



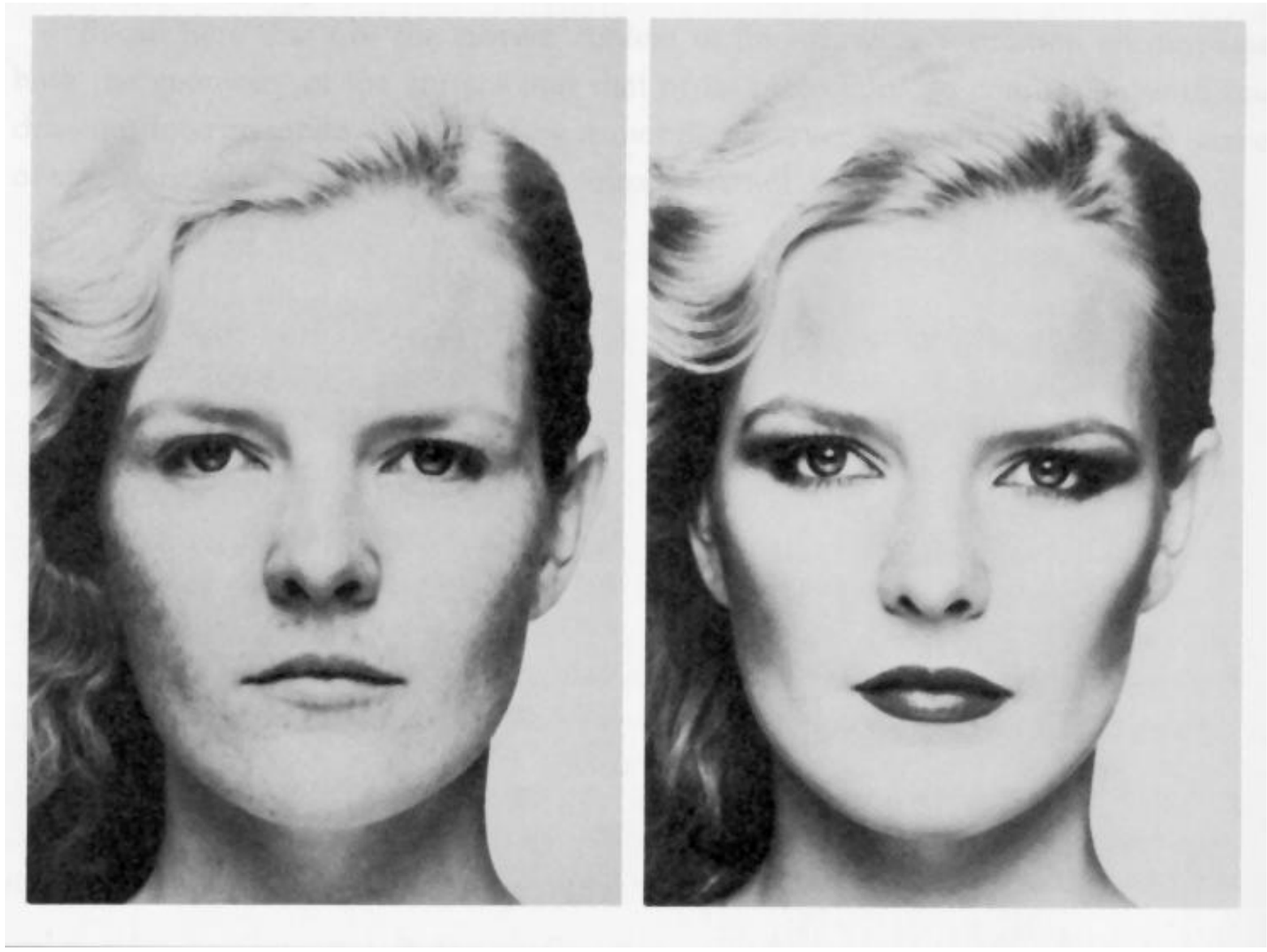
From Panorama's to Stereo's

Recovering 3D from Images

- How can we automatically compute 3D geometry from images?
  - What cues in the image provide 3D information?

Visual Cues for 3D

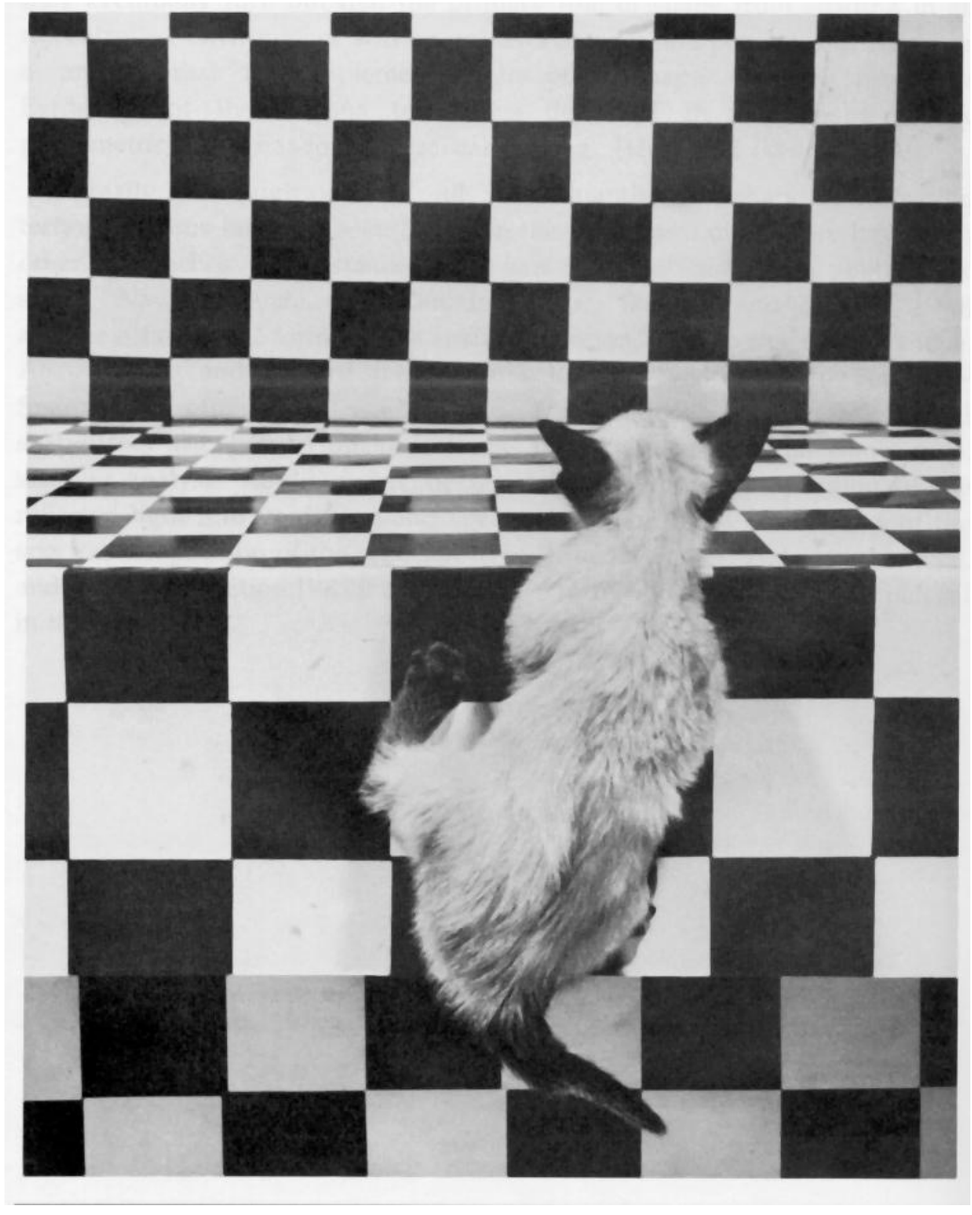
- Shading



Merle Norman Cosmetics, Los Angeles

Visual Cues for 3D

- Shading
- Texture



***The Visual Cliff*, by William Vandivert, 1960**

Visual Cues for 3D

- Shading
- Texture
- Focus

From *The Art of Photography*, Canon

Visual Cues for 3D

- Shading
- Texture
- Focus
- Motion

Visual Cues for 3D

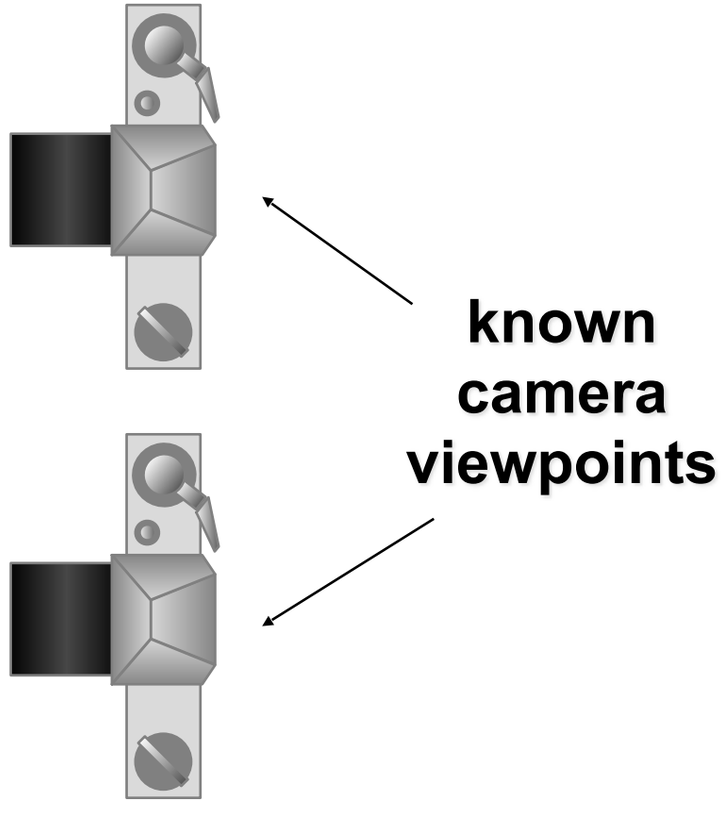
- Shading
- Texture
- Focus
- Motion
- Others:
  - Highlights
  - Shadows
  - Silhouettes
  - Inter-reflections
  - Symmetry
  - Light Polarization
  - ...

Shape From X

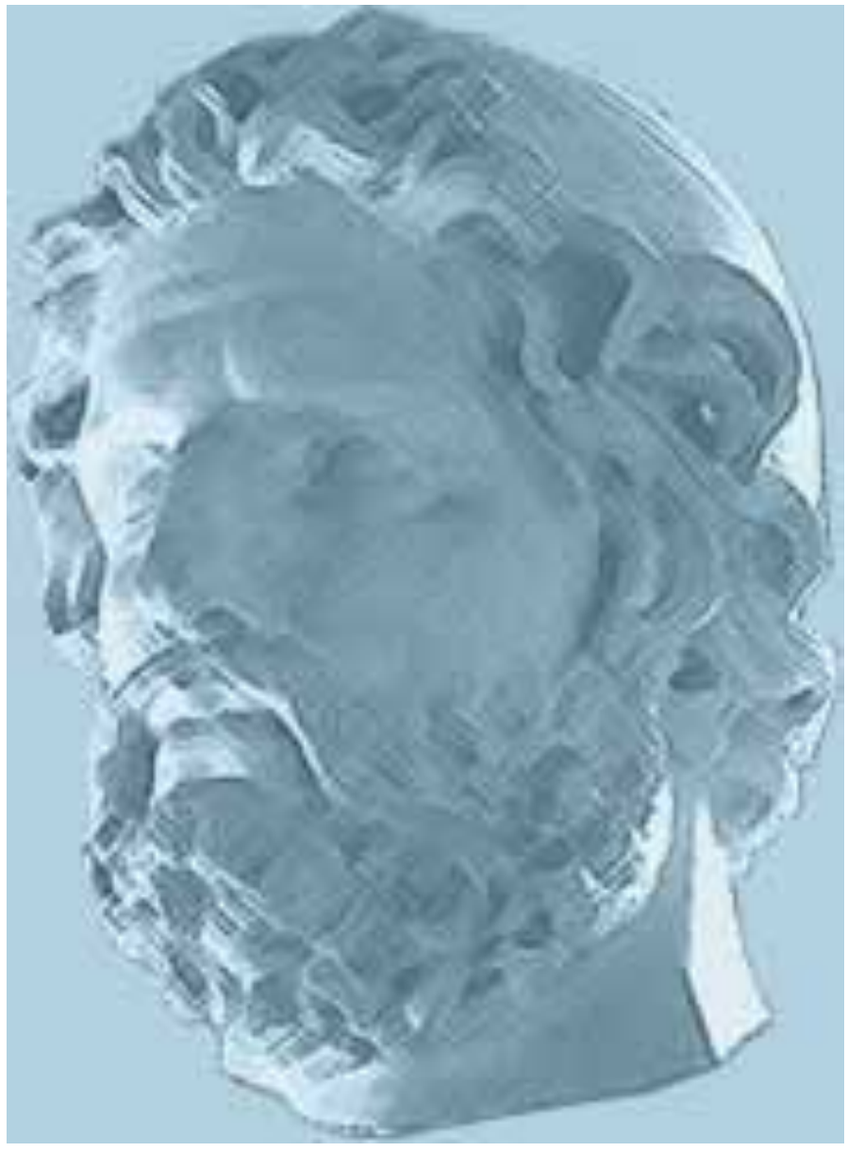
- X = shading, texture, focus, motion, ...

Stereo Reconstruction

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

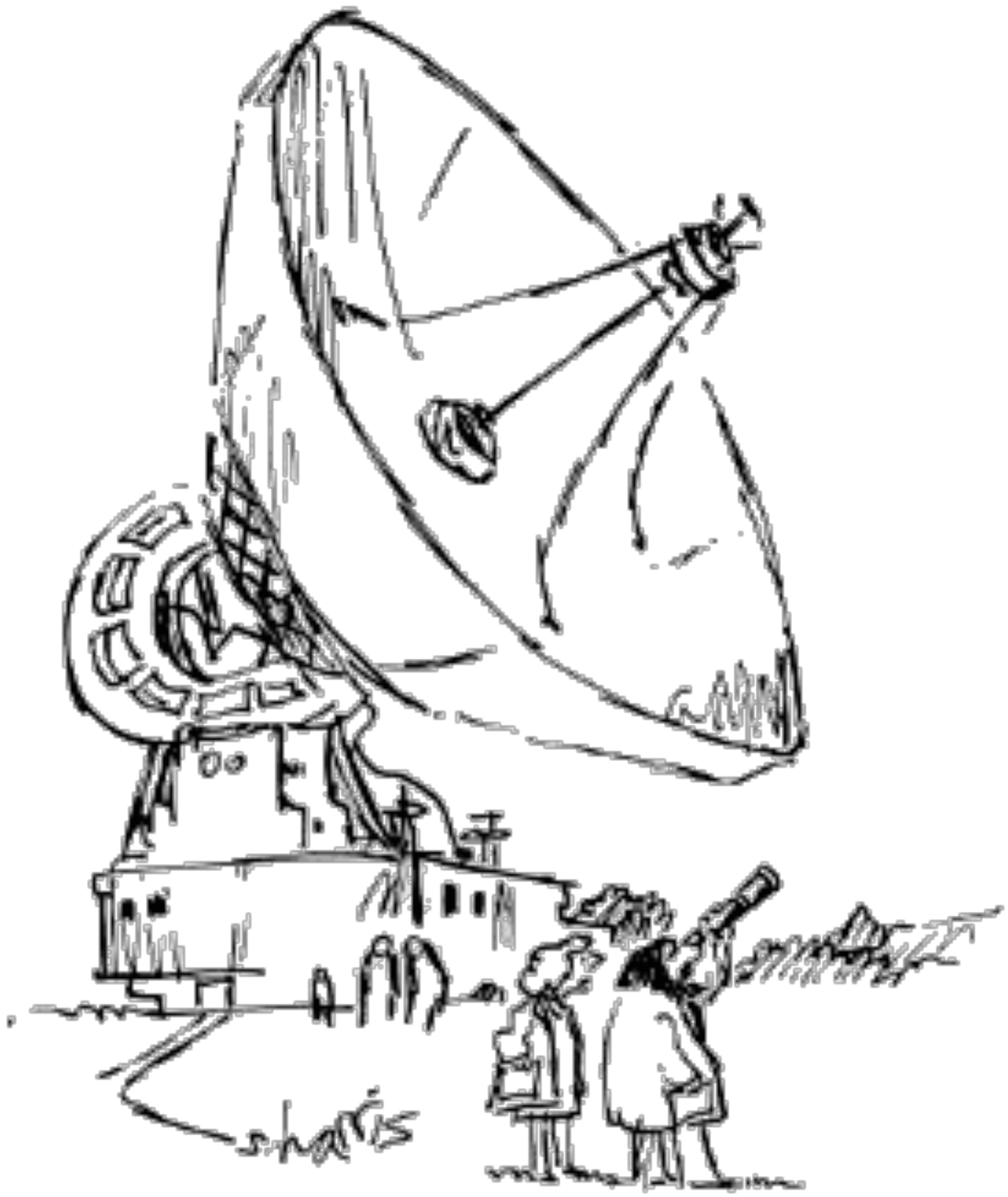


Why do we have two eyes?



Cyclope **vs.** **Odysseus**

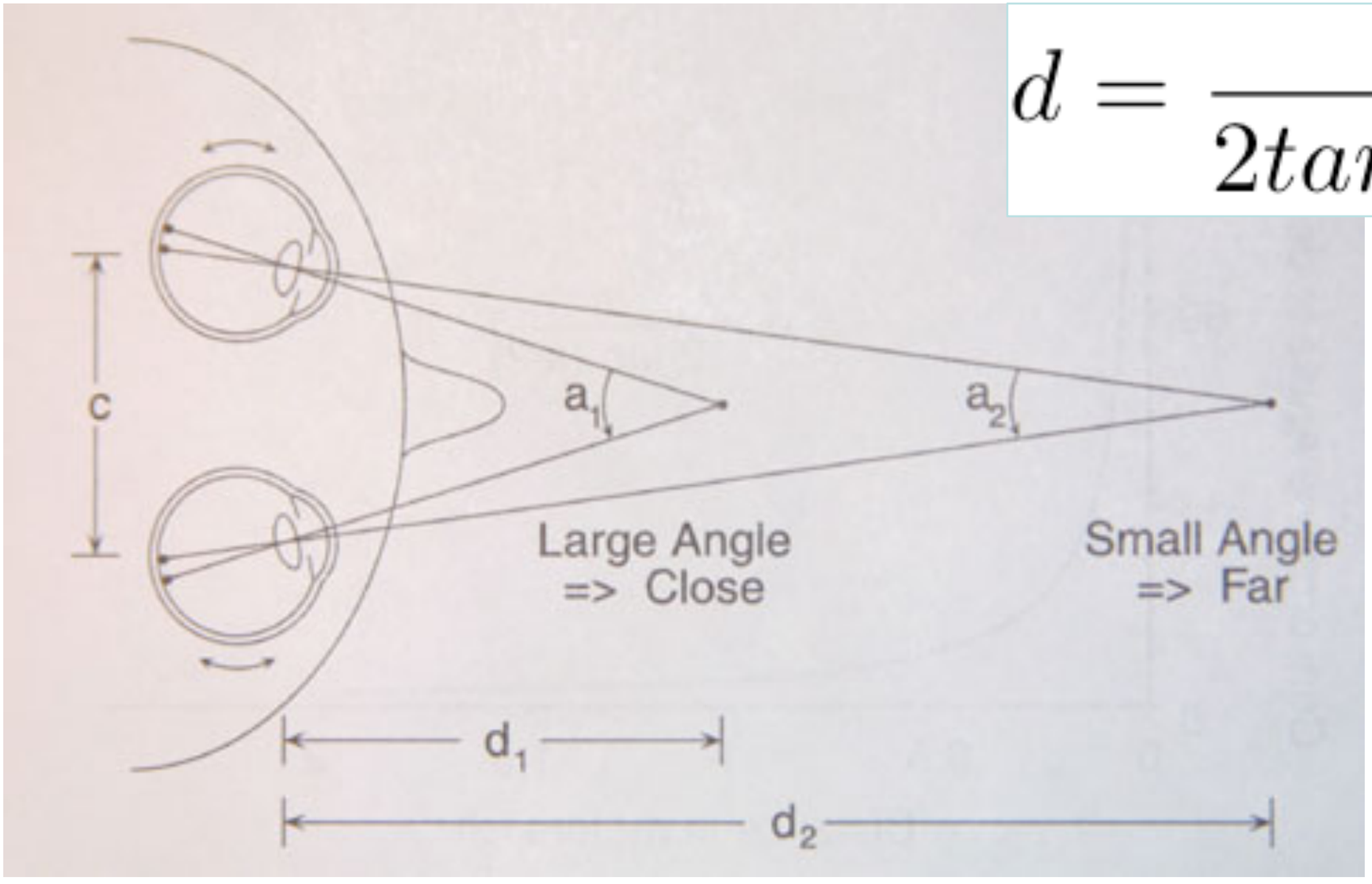
1. Two is better than one



"Just checking."

2. Depth from Convergence

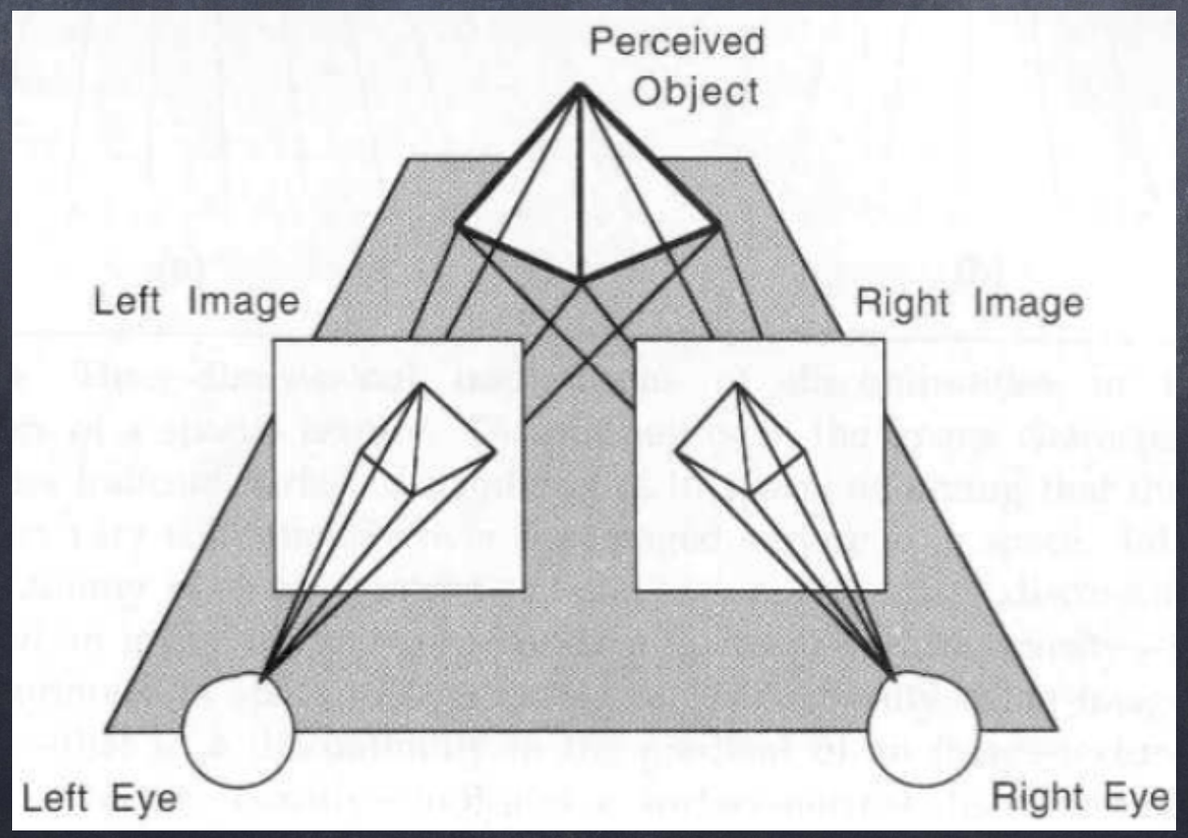
$$d = \frac{c}{2tan(a/2)}$$



Human performance: up to 6-8 feet

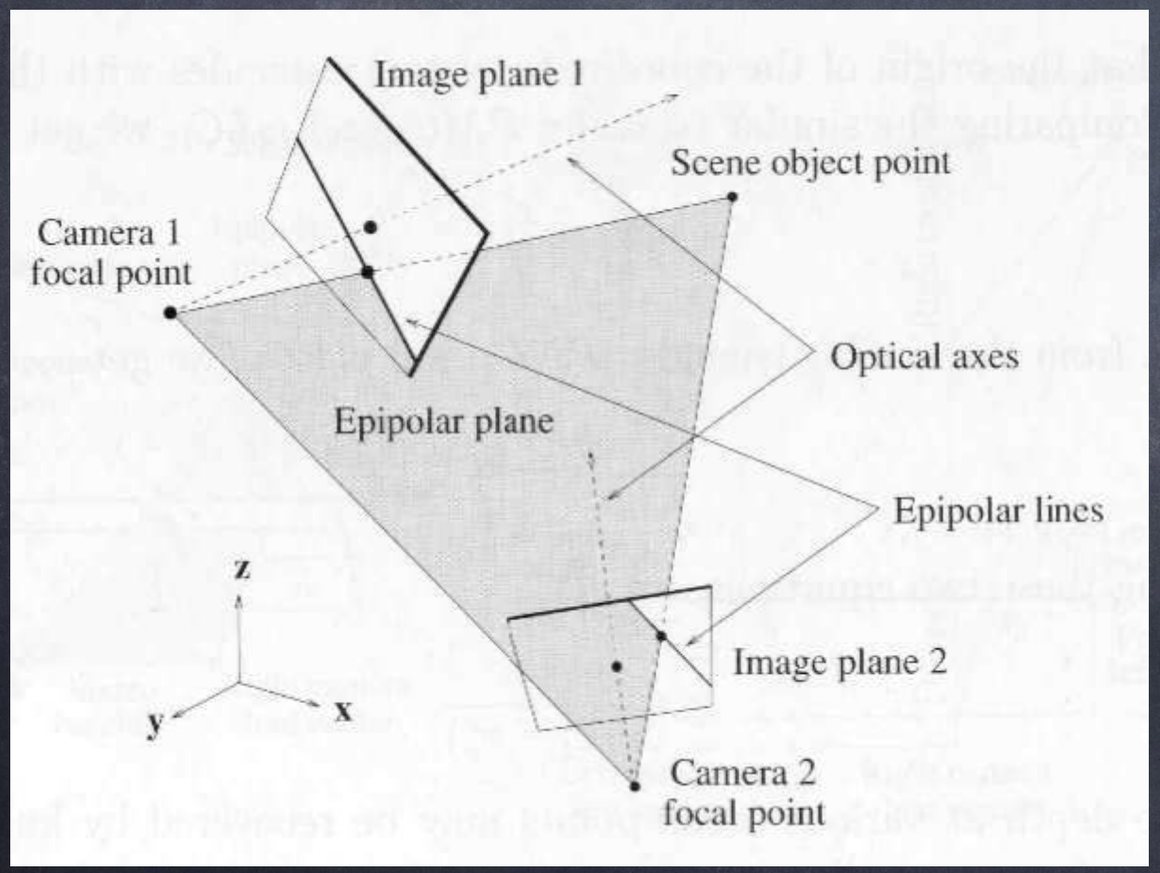
What is the goal of stereo vision?

- The recovery of the 3D structure of a scene using two or more images of the 3D scene, each acquired from a different viewpoint in space.
- The term **binocular vision** is used when two cameras are employed.

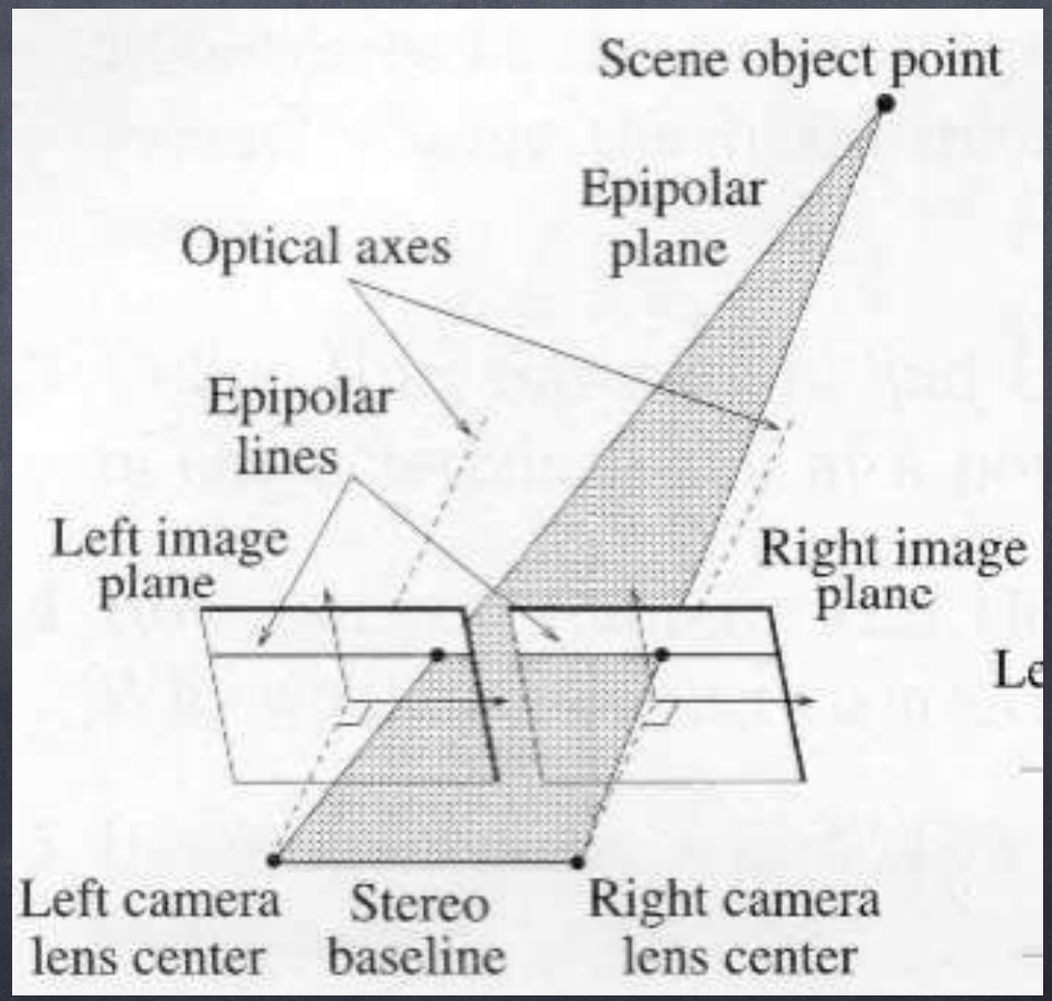


Stereo setup

Cameras in arbitrary position and orientation:



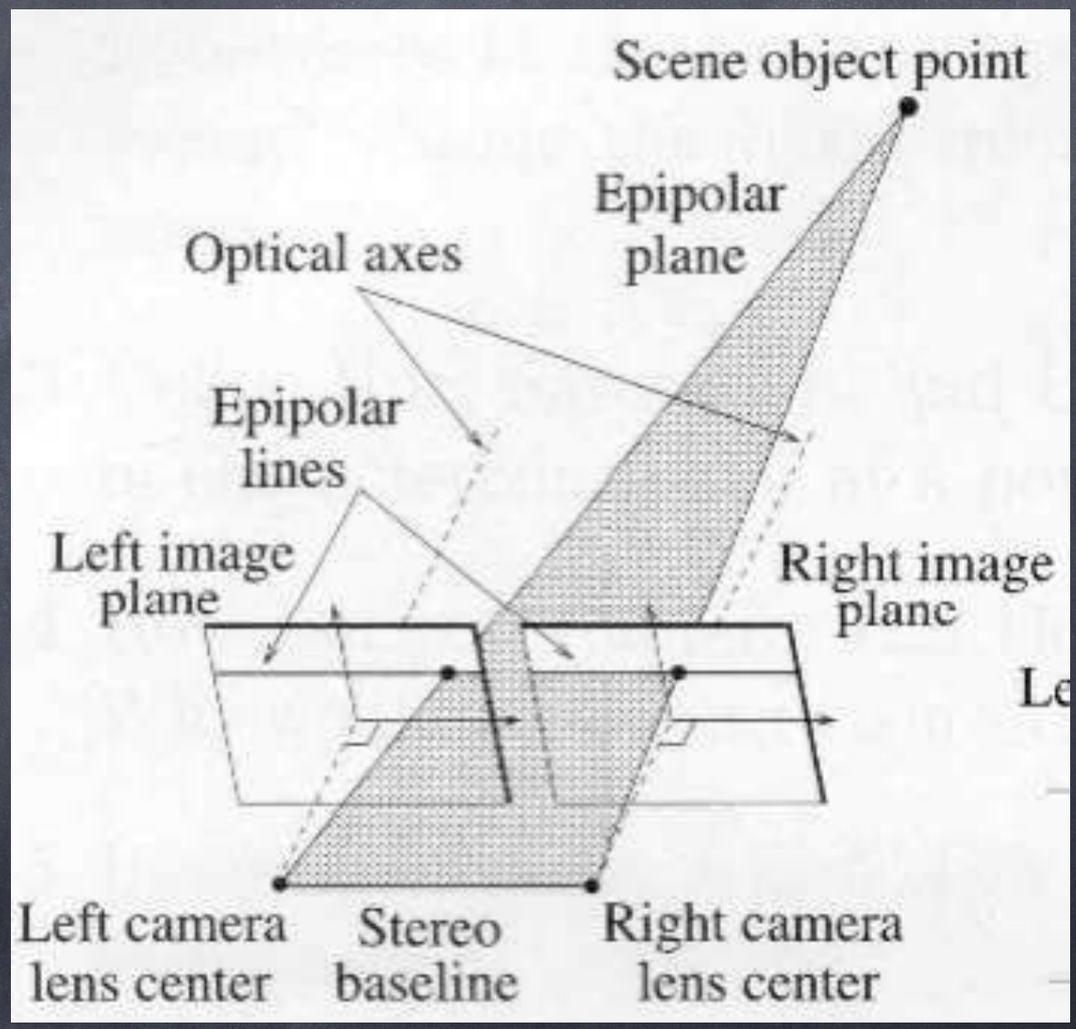
More convenient setup:



Stereo terminology

- **Fixation point:** the point of intersection of the optical axes.

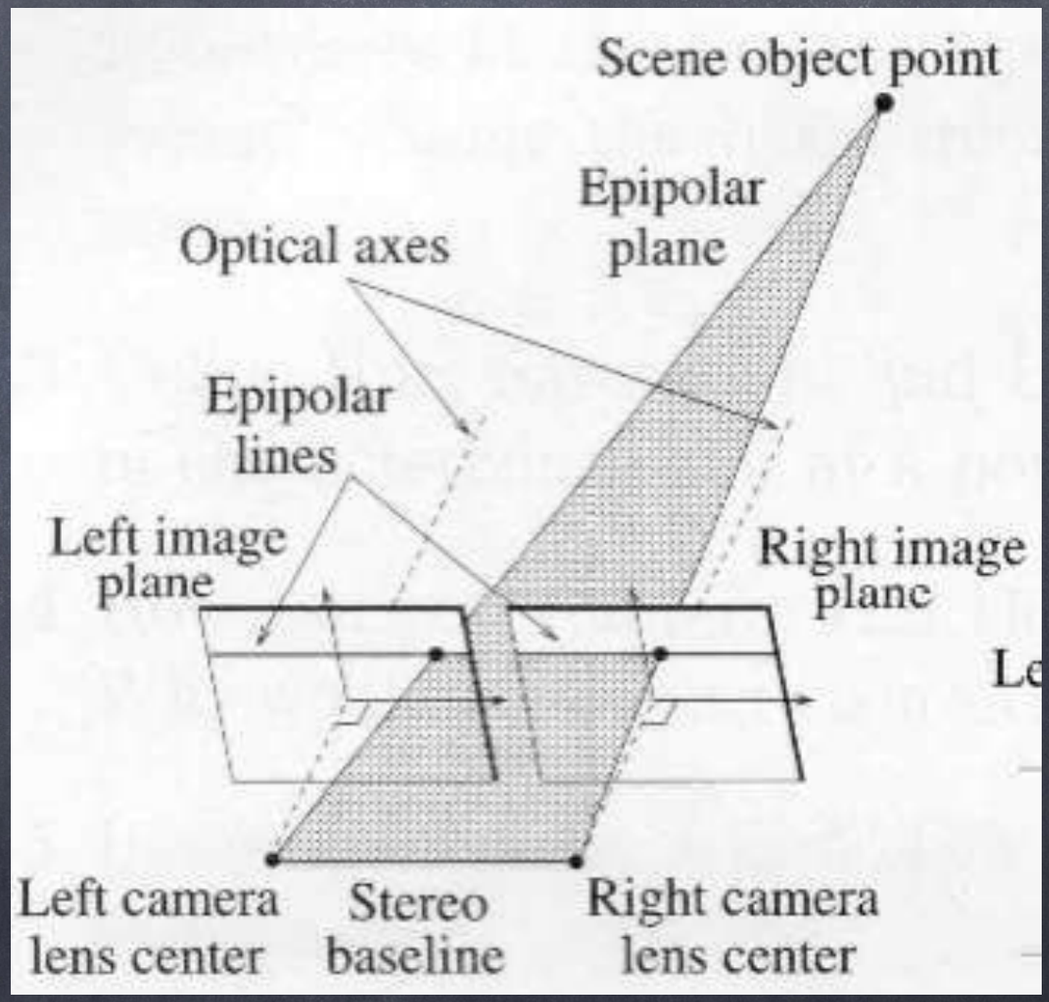
- **Baseline:** the distance between the centers of projection.



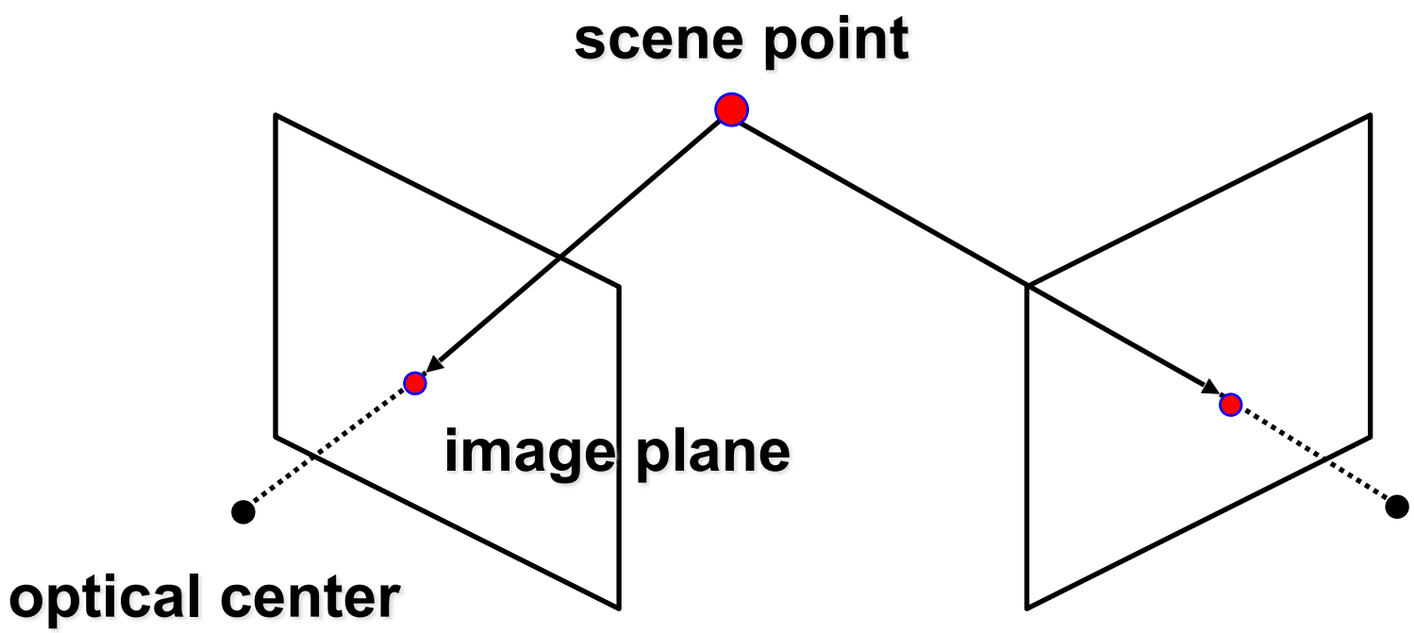
Stereo terminology (cont'd)

- **Disparity:** the distance between corresponding points when the two images are superimposed.

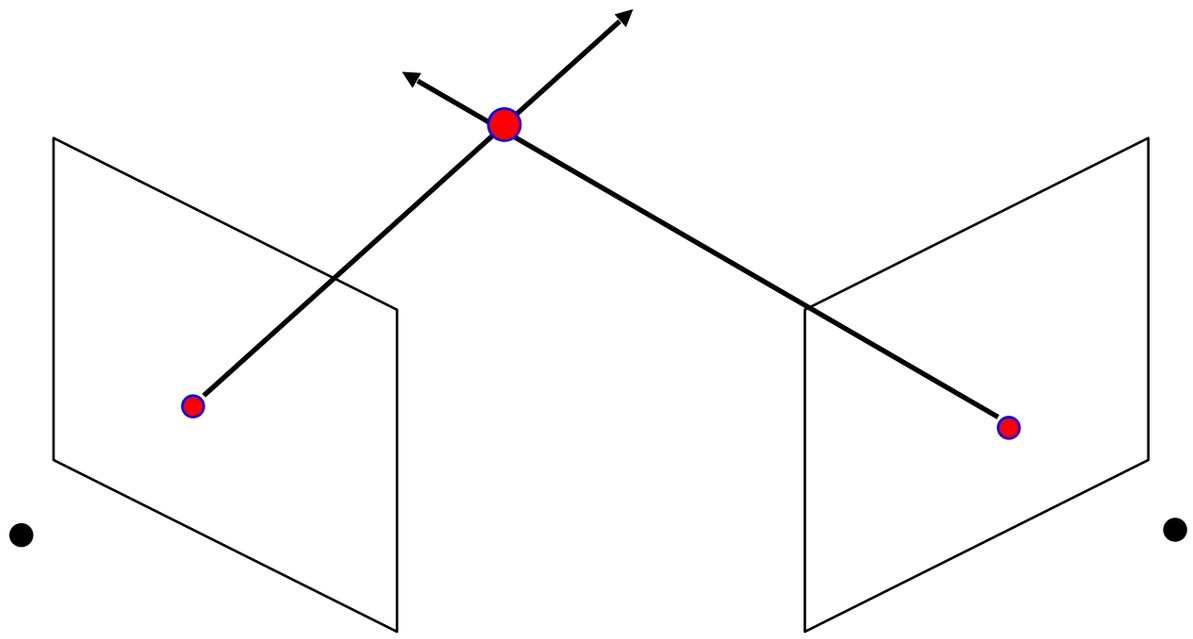
- **Disparity map:** the disparities of all points form the disparity map.



Stereo



Stereo



Basic Principle: Triangulation

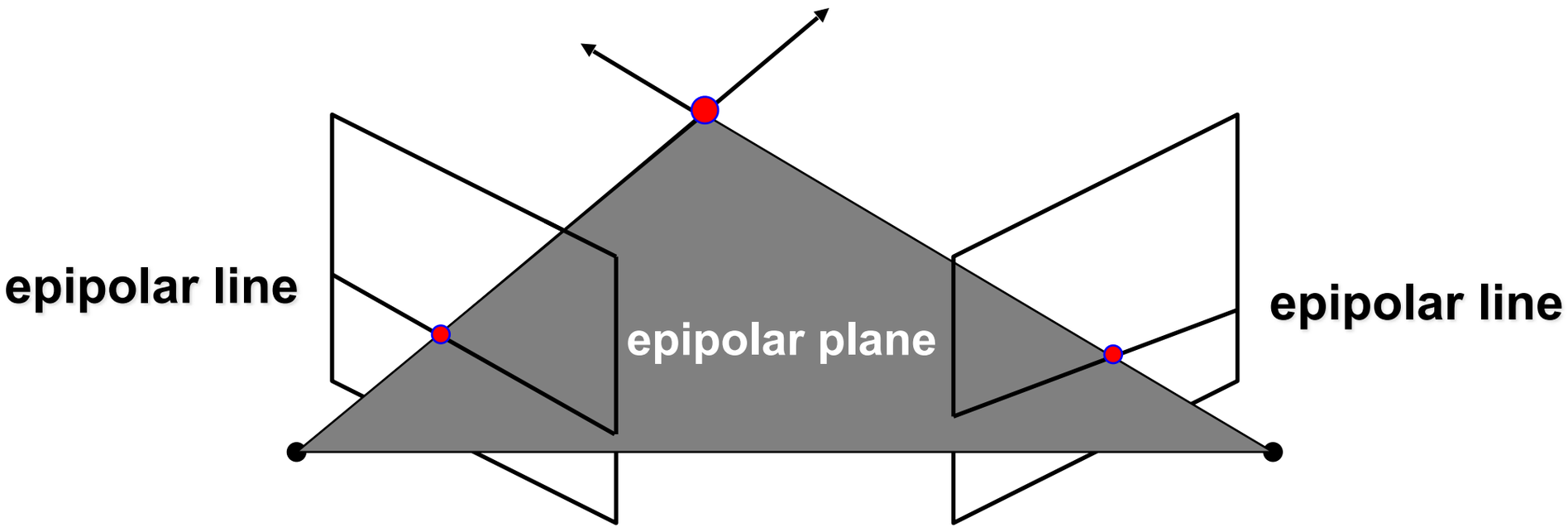
- Gives reconstruction as intersection of two rays

- Requires
  - calibration
  - ***point correspondence***

Stereo correspondence

Determine Pixel Correspondence

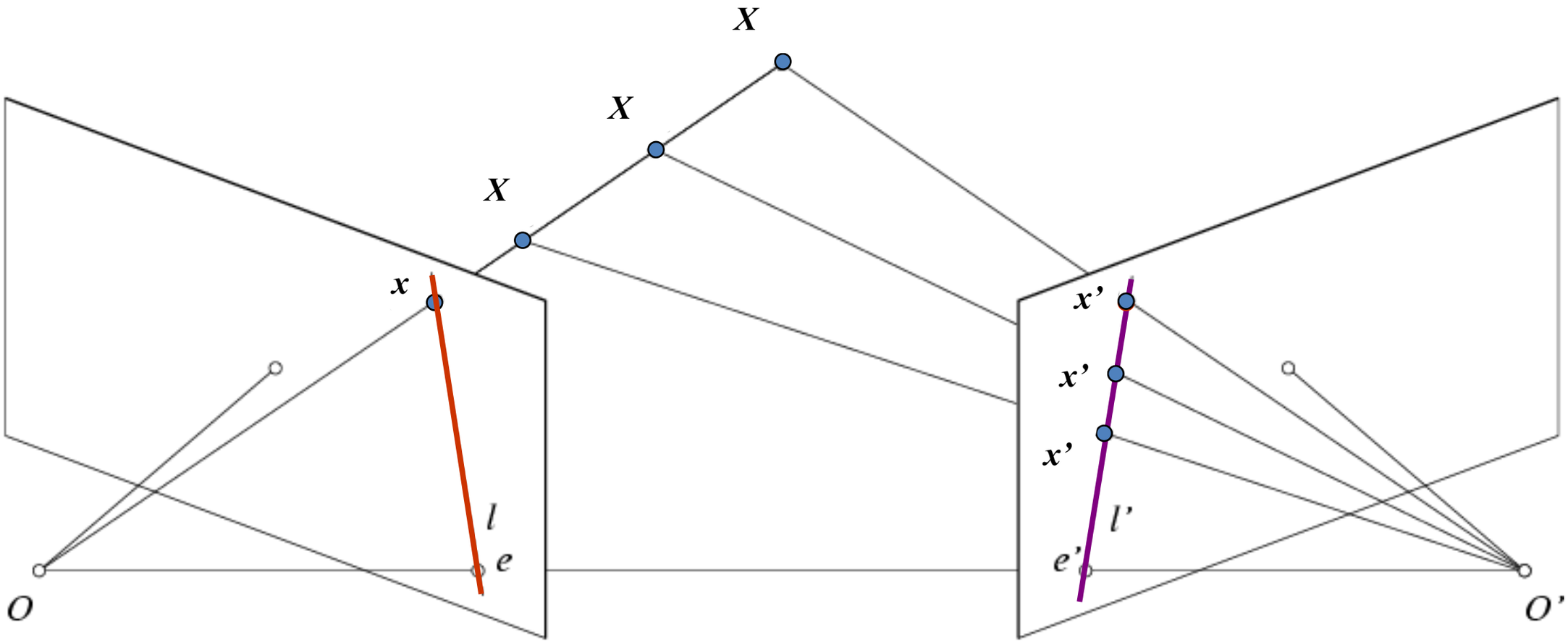
- Pairs of points that correspond to same scene point



Epipolar Constraint

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

Epipolar constraint

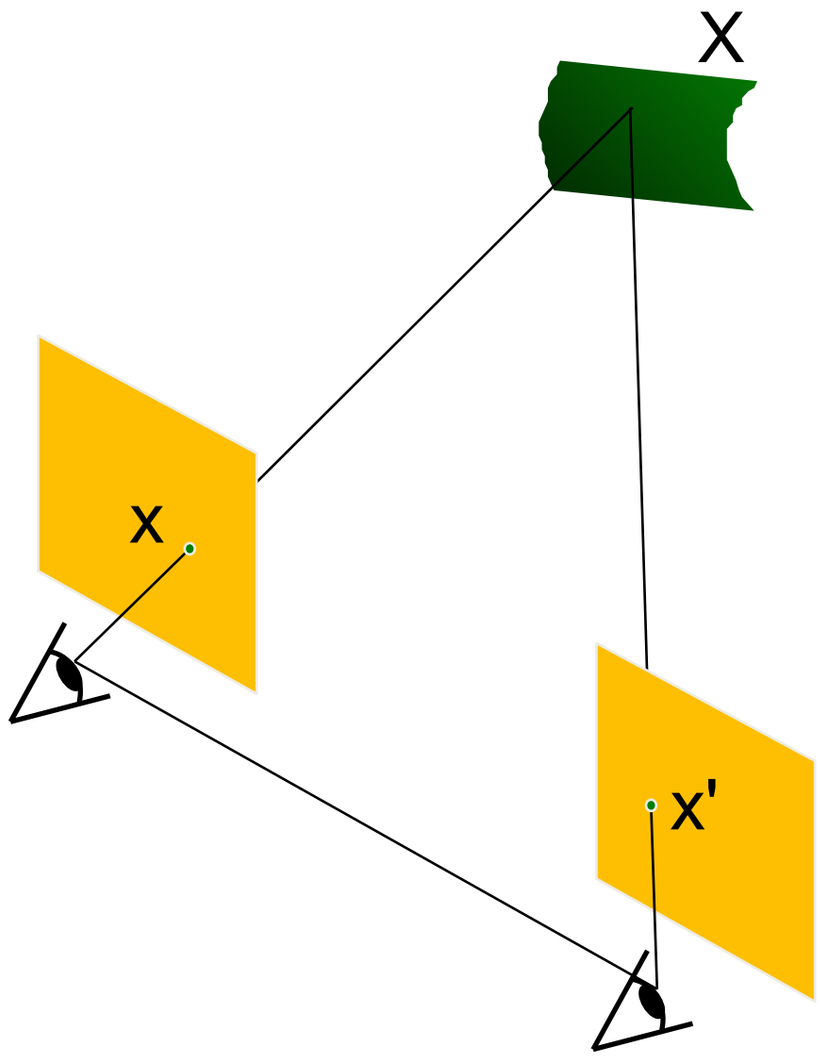


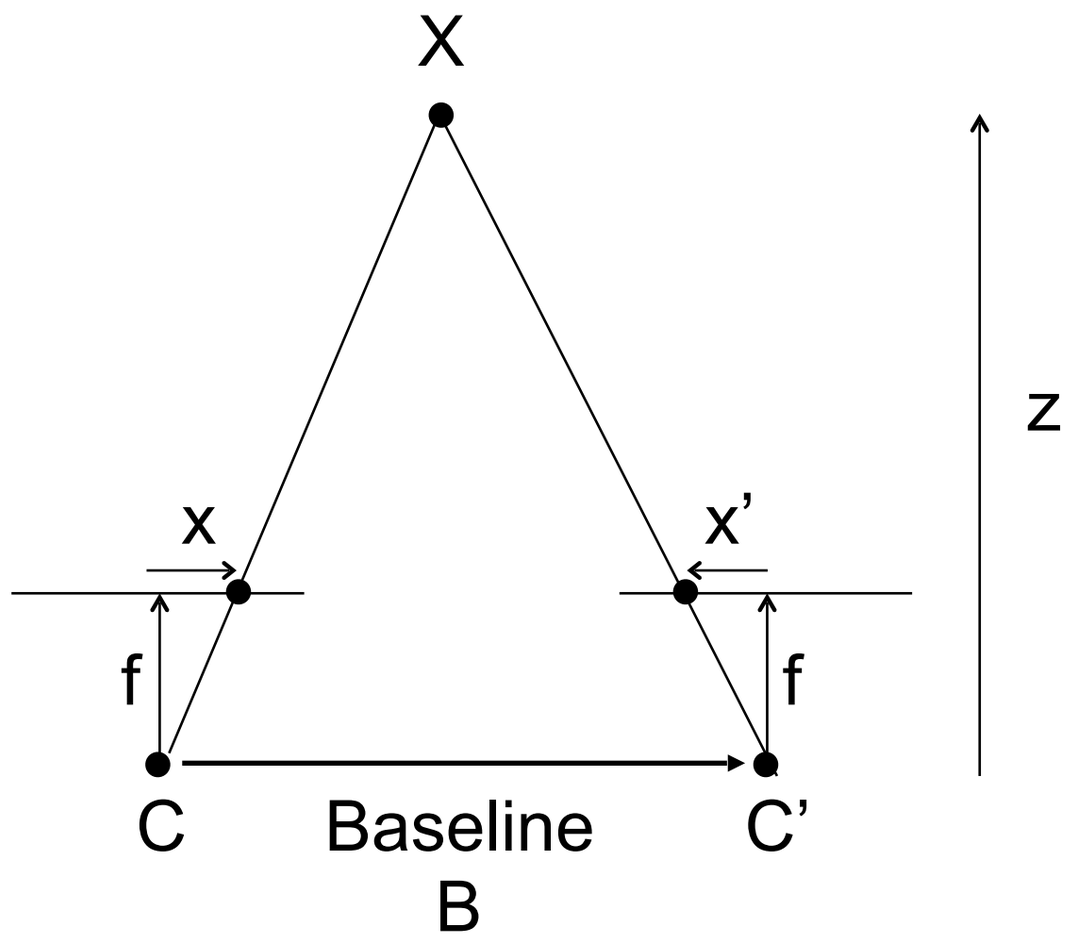
Potential matches for *x* have to lie on the corresponding line *l'*.

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

Depth from Stereo

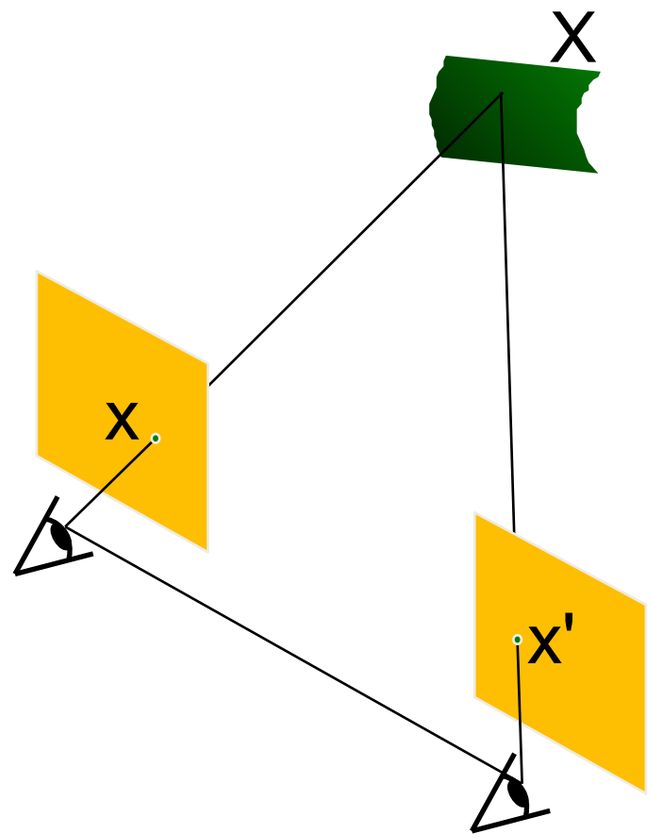
- Goal: recover depth by finding image coordinate x’ that corresponds to x





Depth from Stereo

- Goal: recover depth by finding image coordinate x’ that corresponds to x
- Sub-Problems
  1. Calibration: How do we recover the relation of the cameras (if not already known)?
  2. Correspondence: How do we search for the matching point x’?



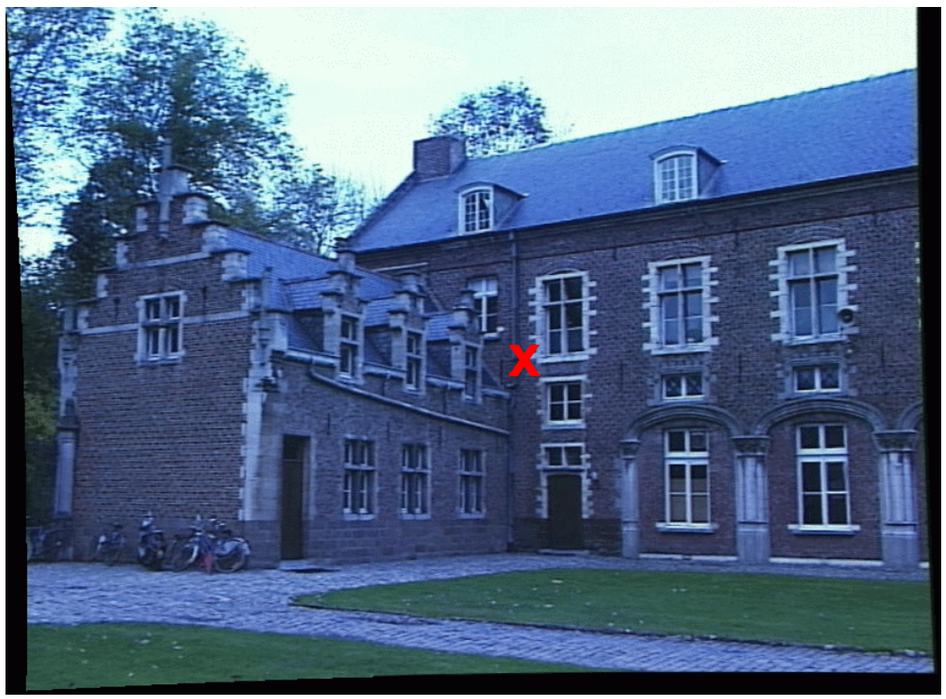
The correspondence problem

- Finding pairs of matched points such that each point in the pair is the projection of the same 3D point.

left camera

right camera

Correspondence Problem



- We have two images taken from cameras with different intrinsic and extrinsic parameters

- How do we match a point in the first image to a point in the second? How can we constrain our search?

Binocular stereo

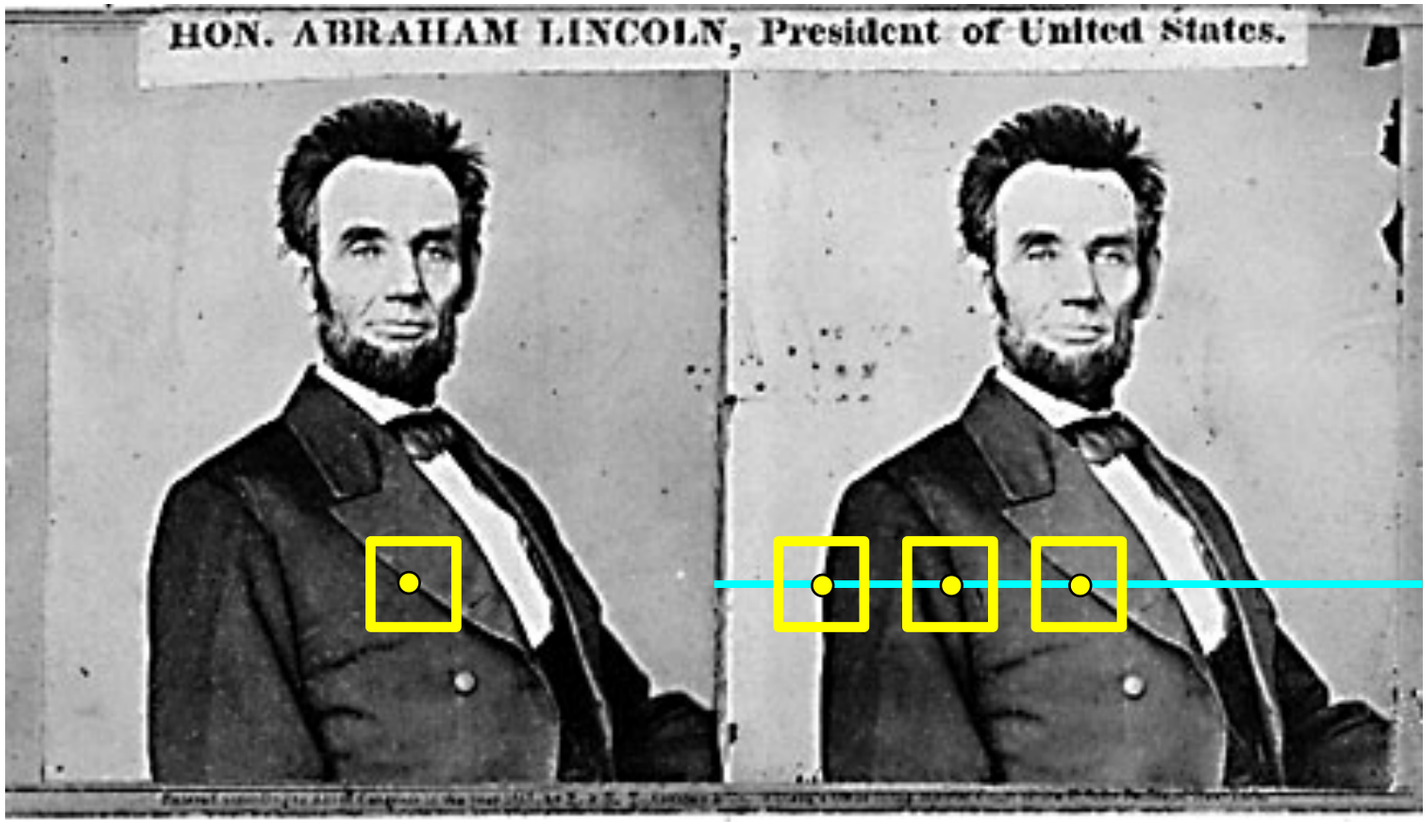
- Given a calibrated binocular stereo pair, fuse it to produce a depth image

image 1

image 2

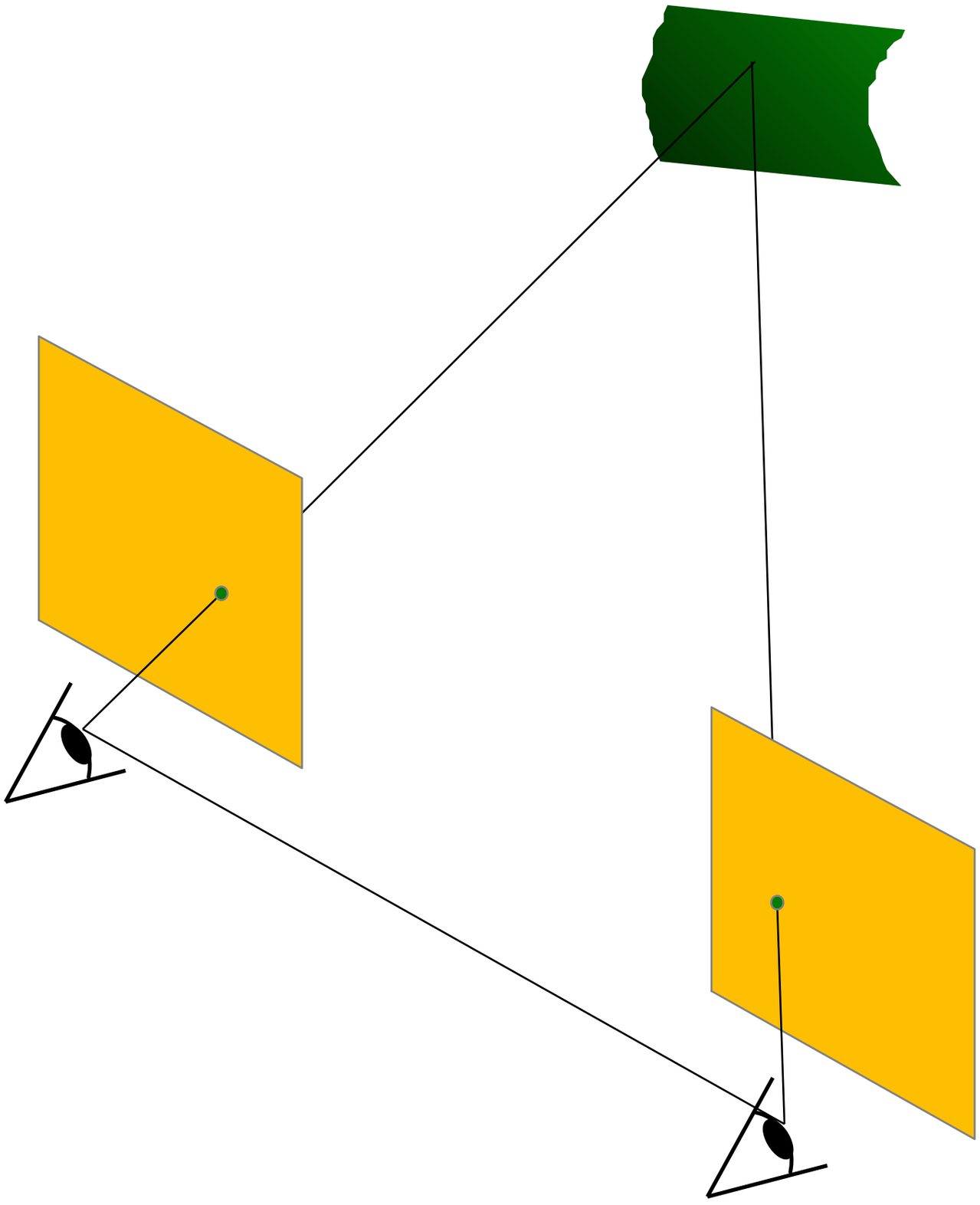
Dense depth map

Basic stereo matching algorithm



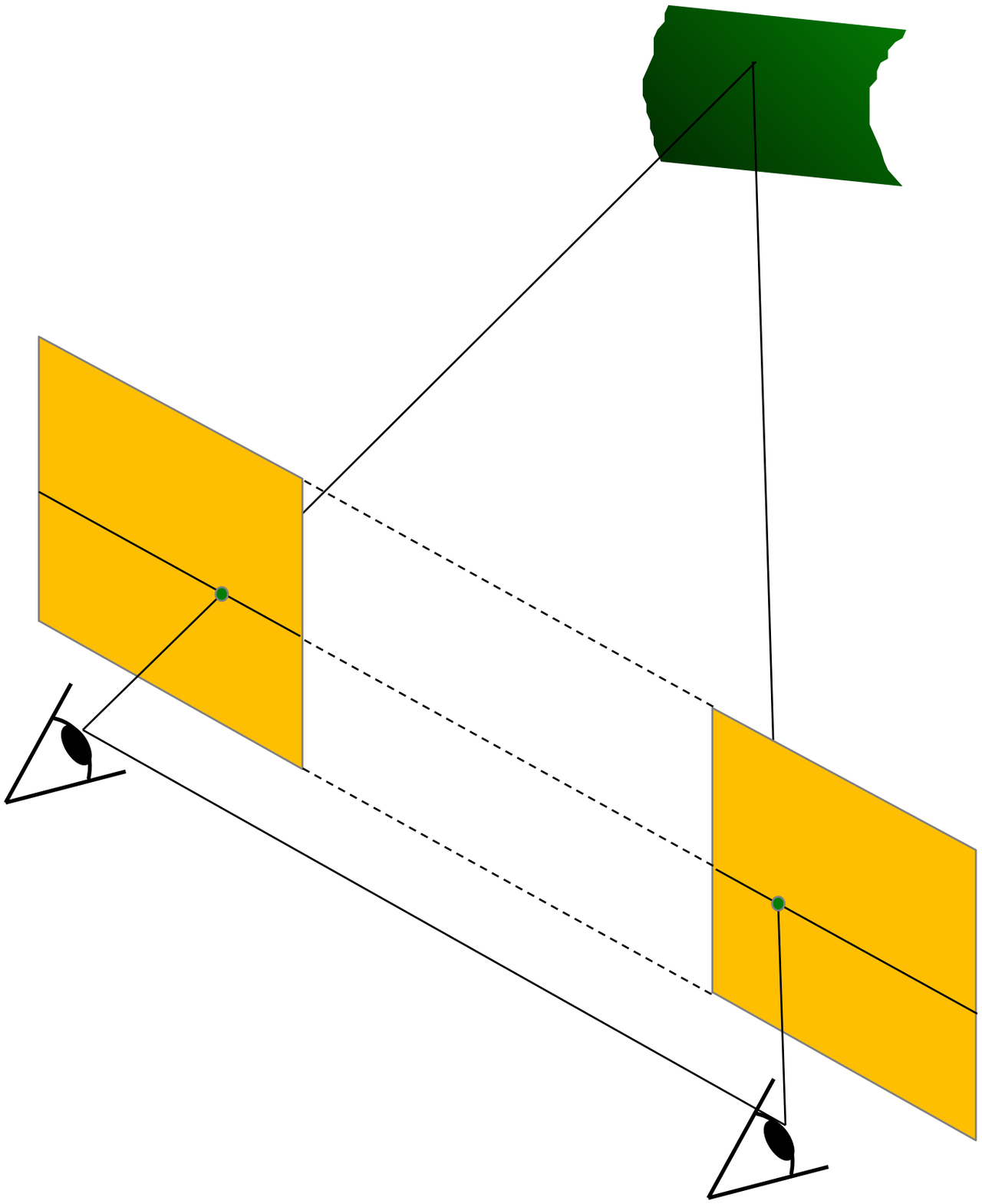
- For each pixel in the first image
  - Find corresponding epipolar line in the right image
  - Examine all pixels on the epipolar line and pick the best match
  - Triangulate the matches to get depth information

Simplest Case: Parallel images



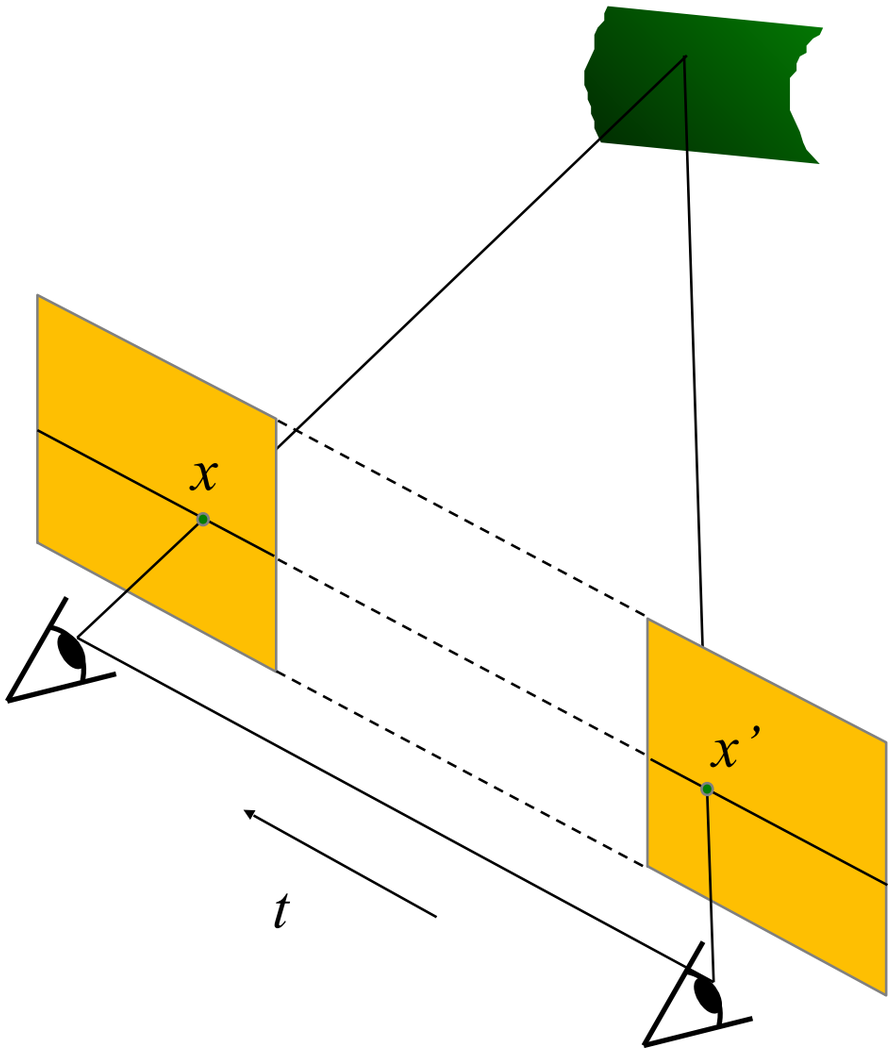
- Image planes of cameras are parallel to each other and to the baseline
- Camera centers are at same height
- Focal lengths are the same

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

Essential matrix for parallel images



Epipolar constraint:

$$x^T E x' = 0, \quad E = [t_\times]R$$

$$R = I \qquad t = (T, 0, 0)$$

$$E = [t_\times]R = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T \\ 0 & T & 0 \end{bmatrix}$$

$$[a_\times] = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}$$

Essential matrix

Let $\vec{p}_w^L$ and $\vec{p}_w^R$ be the left and right image points (written in world coordinates) for some given 3D point $\vec{X}_w$. Then the epipolar constraint states that these two image points and the two nodal points $\vec{d}_w^L$, $\vec{d}_w^R$ are all coplanar. This constraint can be written as

$$(\vec{p}_w^L - \vec{d}_w^L)^T \left[(\vec{d}_w^L - \vec{d}_w^R) \times (\vec{p}_w^R - \vec{d}_w^R) \right] \; = \; 0, \tag{4}$$

where '$\times$' denotes the cross-product.

We rewrite this by replacing the cross-product by an equivalent matrix-vector product,

$$\vec{T} \times \vec{p} \; = \; [\vec{T}]_\times \vec{p}, \;\; \text{where } [\vec{T}]_\times \; = \; \begin{pmatrix} 0 & -T_3 & T_2 \\ T_3 & 0 & -T_1 \\ -T_2 & T_1 & 0 \end{pmatrix}.$$

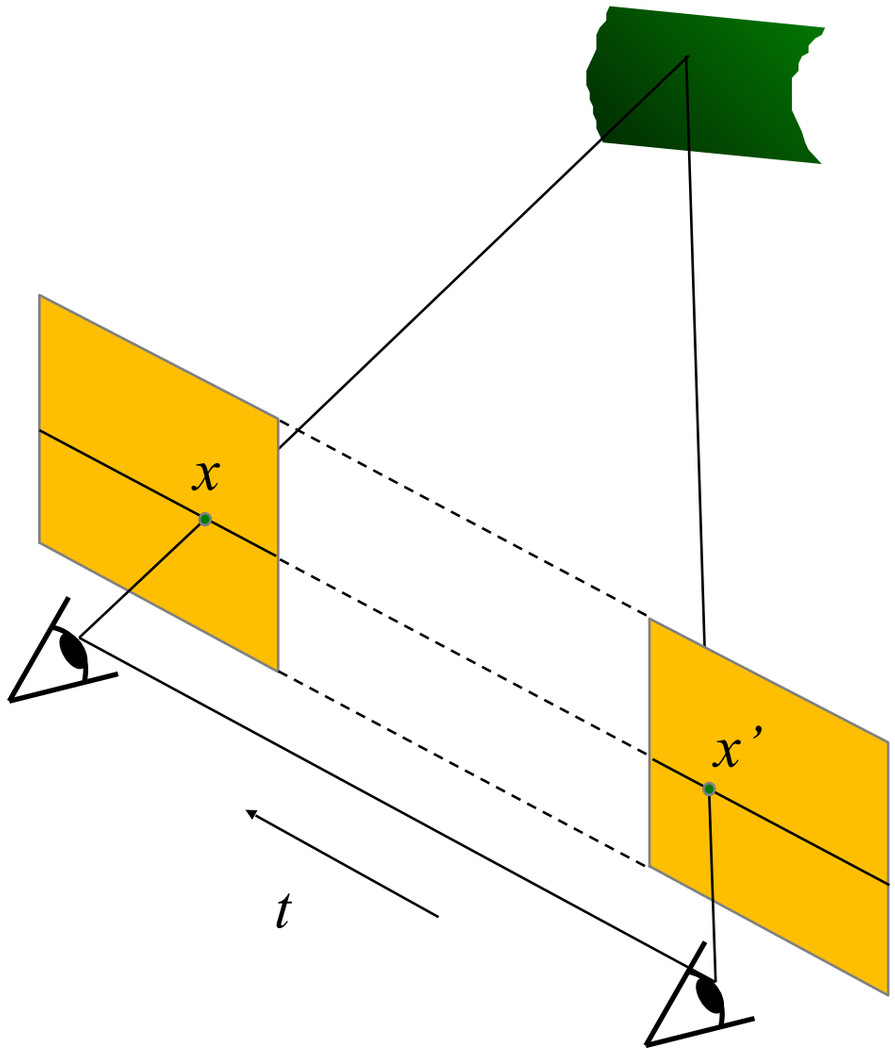
Also, we use (2) and (3) to write $\vec{p}_w^L - \vec{d}_w^L$ in terms of the left camera's coordinates as $\vec{p}_w^L - \vec{d}_w^L \; = \; (R^L)^T \vec{p}_c^L$. Using the analogous expression for the right image point, we find that (4) can be rewritten as

$$(\vec{p}_c^L)^T E \vec{p}_c^R \; = \; 0, \;\; \text{(epipolar constraint)} \tag{5}$$

where E is the 3×3 *essential matrix* (or E-matrix)

$$E \; = \; R^L [\vec{d}_w^L - \vec{d}_w^R]_\times (R^R)^T. \tag{6}$$

Essential matrix for parallel images



Epipolar constraint:

$$x^T E x' = 0, \quad E = [t_\times]R$$

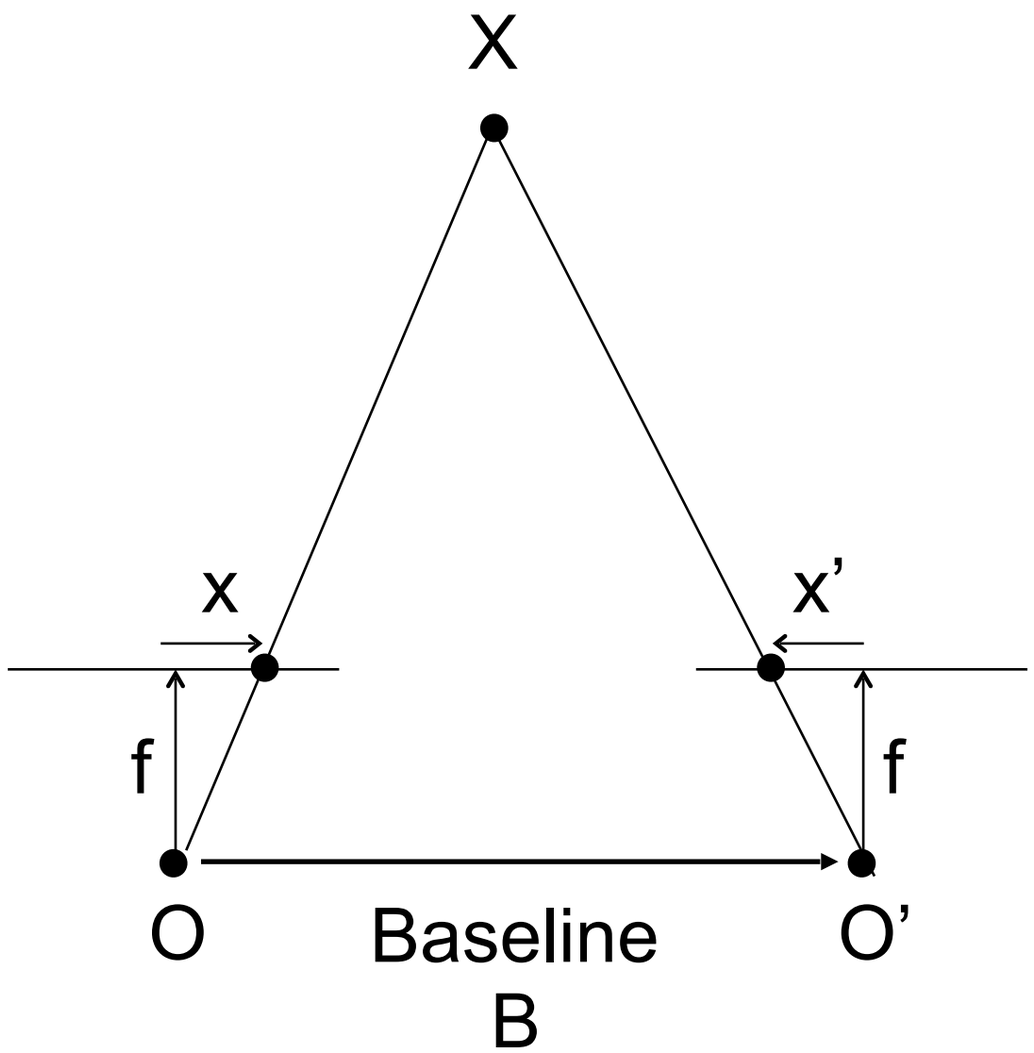
$$R = I \qquad t = (T, 0, 0)$$

$$E = [t_\times]R = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T \\ 0 & T & 0 \end{bmatrix}$$

$$\begin{pmatrix} u & v & 1 \end{pmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T \\ 0 & T & 0 \end{bmatrix} \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = 0 \qquad \begin{pmatrix} u & v & 1 \end{pmatrix} \begin{pmatrix} 0 \\ -T \\ Tv' \end{pmatrix} = 0 \qquad Tv = Tv'$$

The y-coordinates of corresponding points are the same!

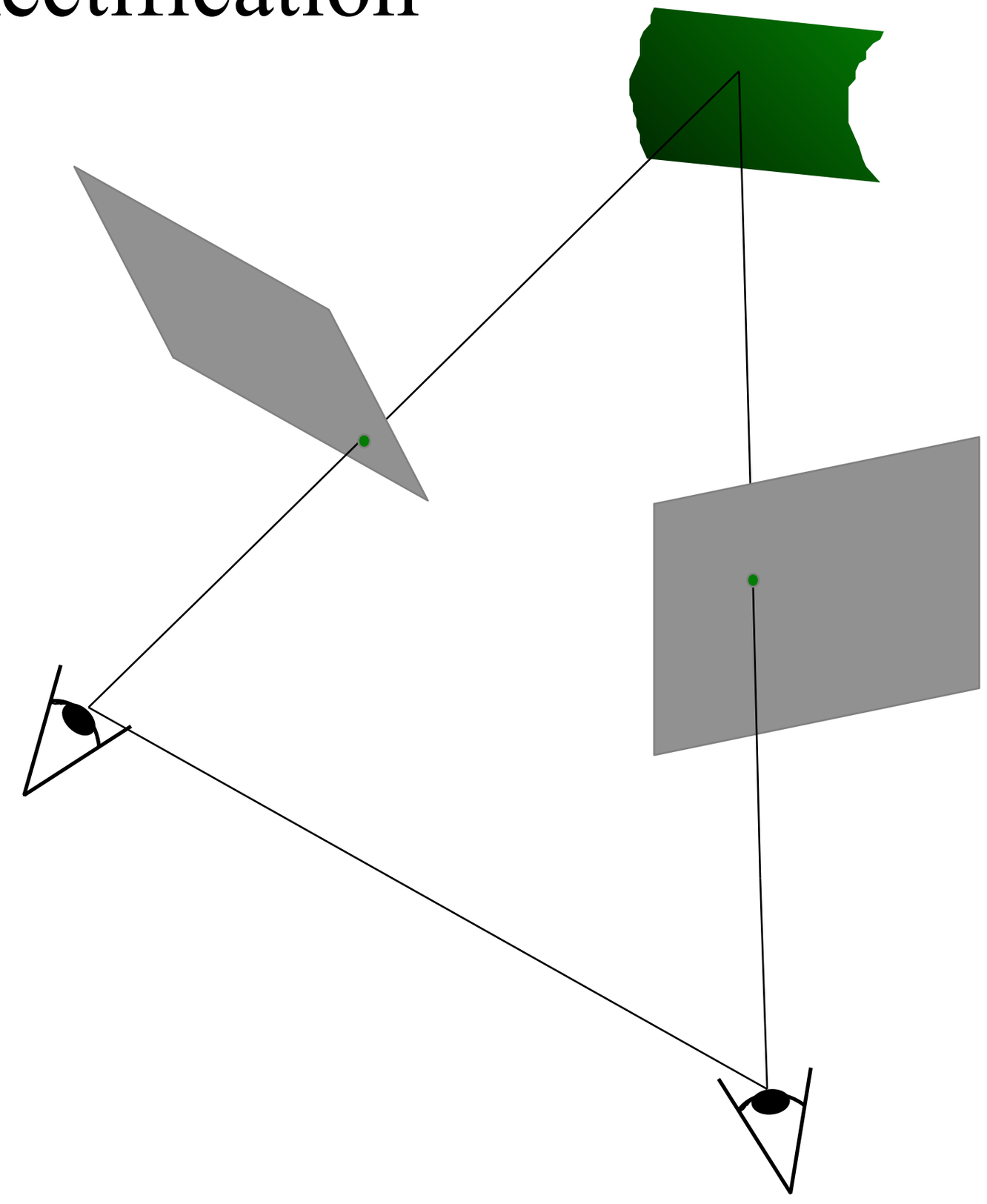
Depth from disparity



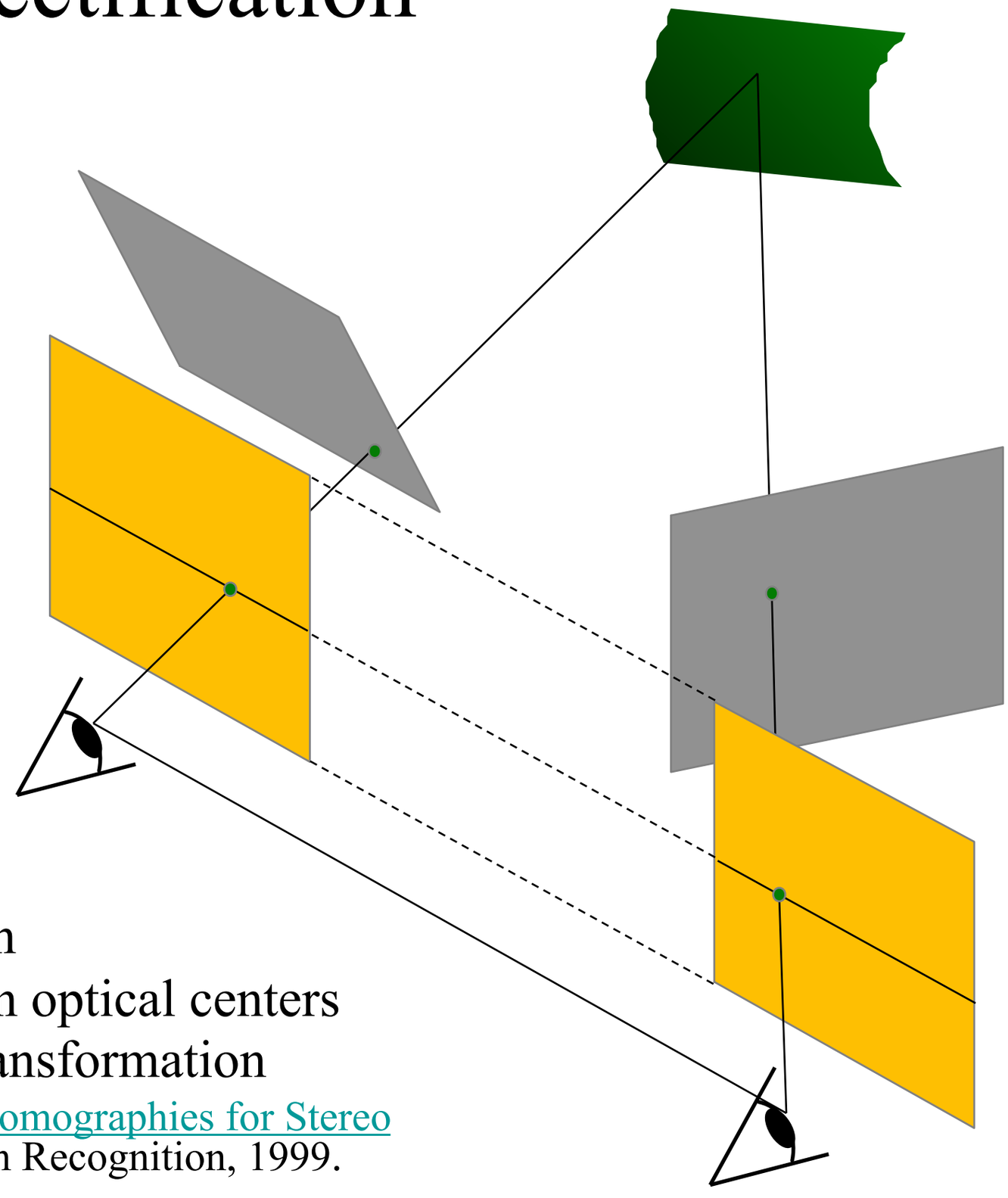
$$disparity \qquad x - x' = \frac{B \cdot f}{z}$$

Disparity is inversely proportional to depth!

Stereo Image Rectification



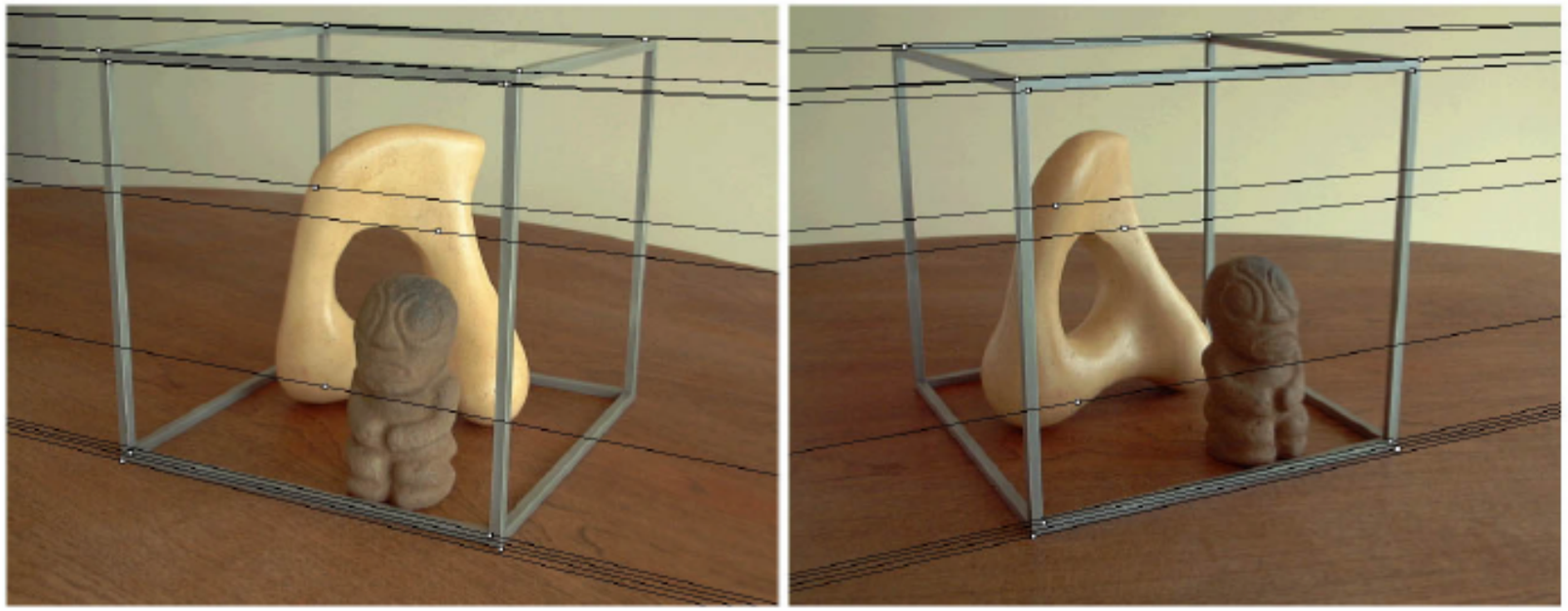
Stereo Image Rectification



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

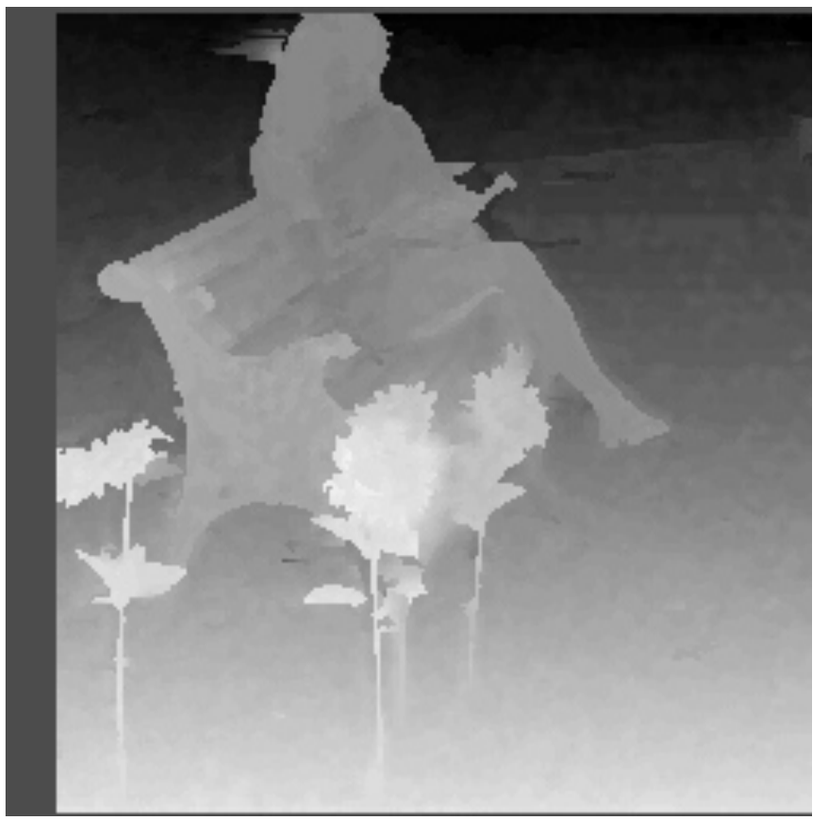
Stereo Rectification



Stereo Example

left image

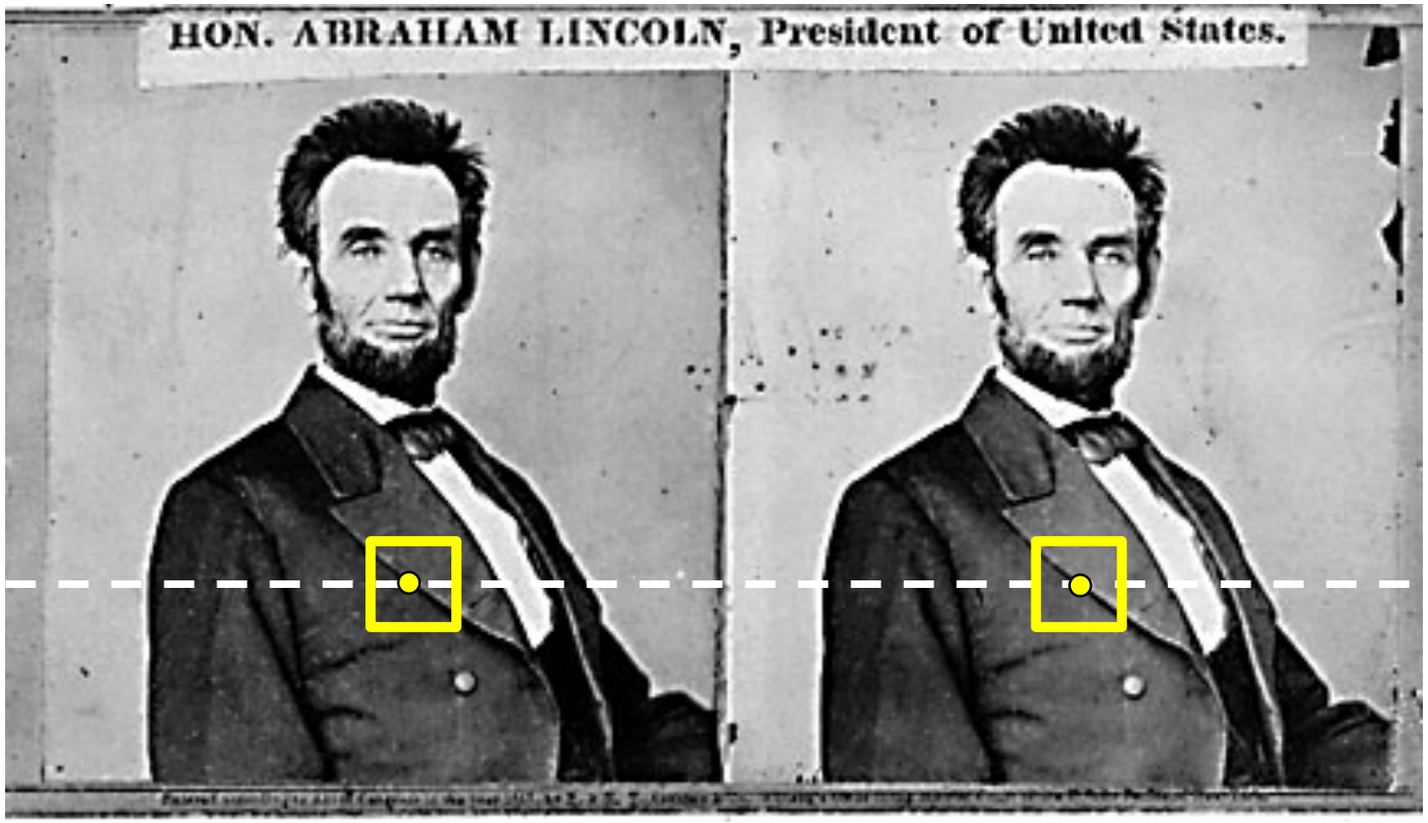
right image



depth map

H. Tao et al. “Global matching criterion and color segmentation based stereo”

Your basic stereo algorithm



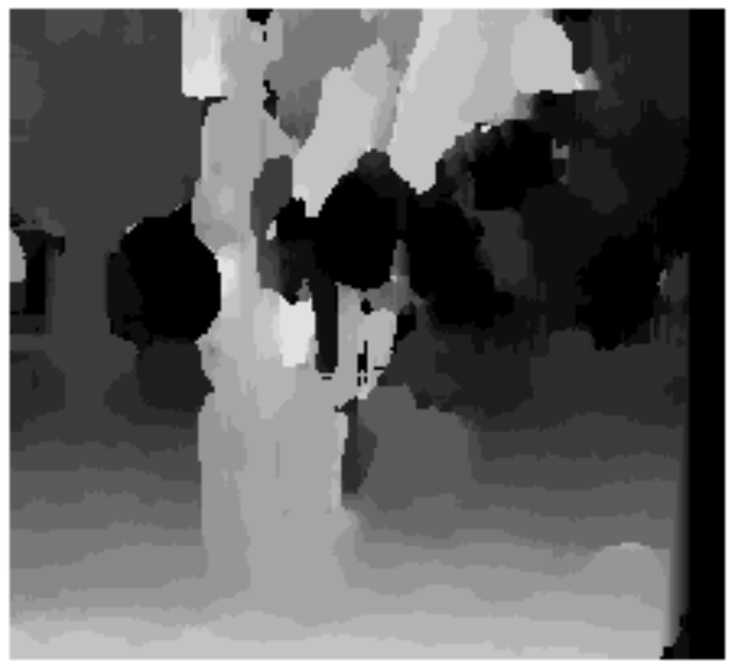
For each epipolar line

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***

Window size



W = 3

W = 20

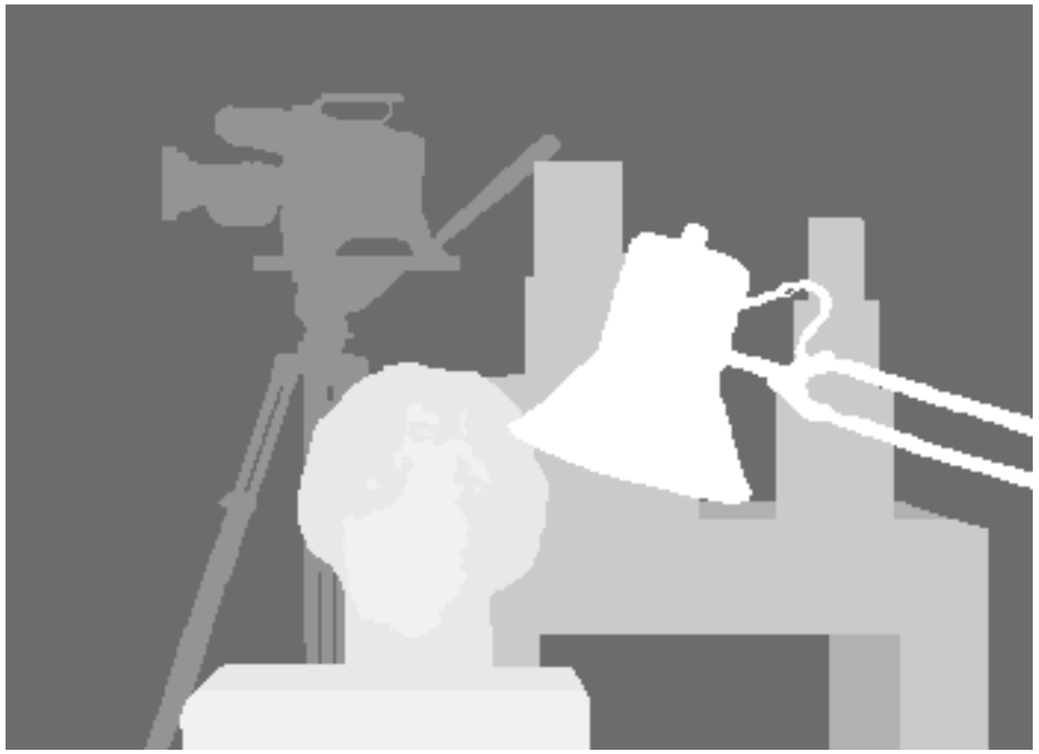
Effect of window size

- Smaller window
  - \+
  - –
- Larger window
  - \+
  - –

Stereo results

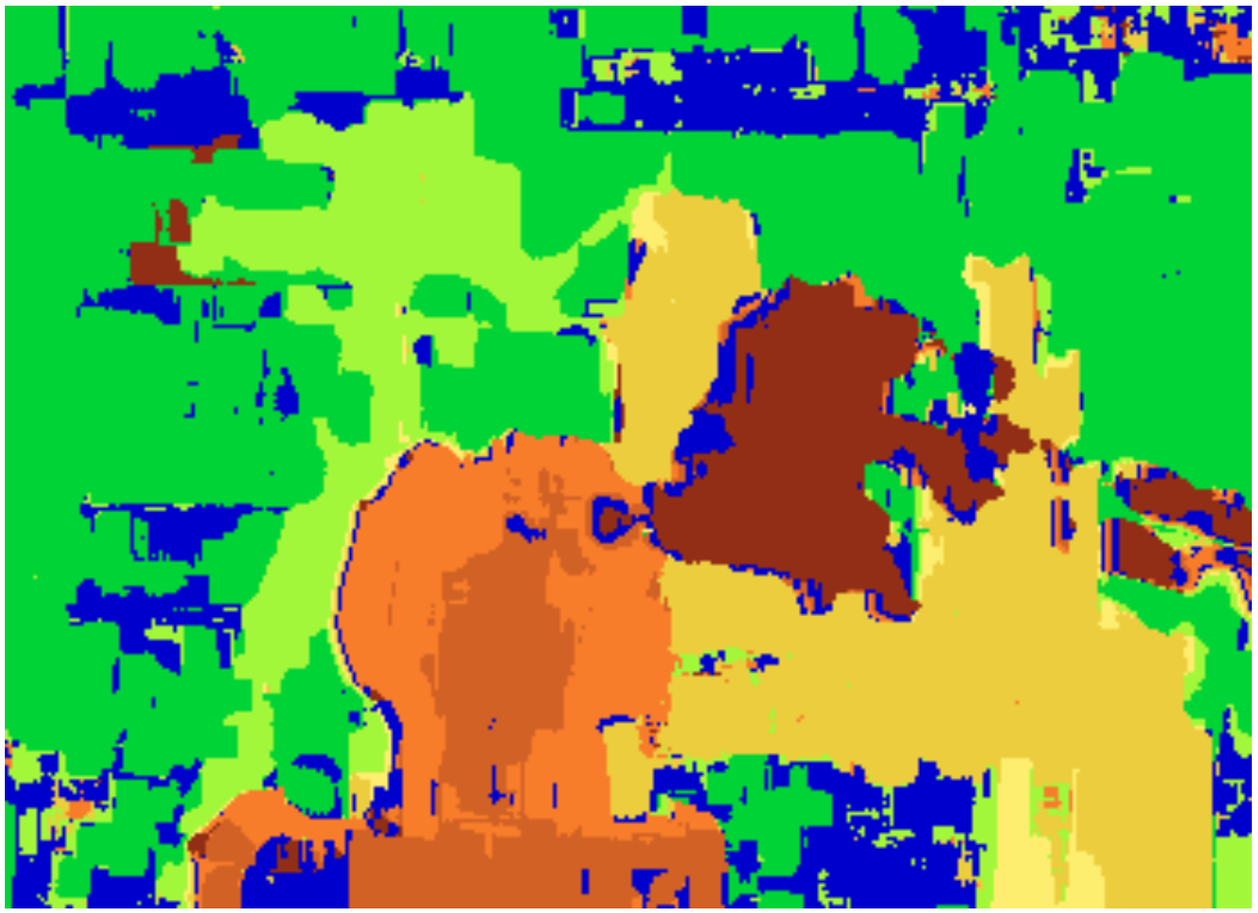
- Data from University of Tsukuba
- Similar results on other images without ground truth

Scene



Ground truth

Results with window search



Window-based matching
(best window size)

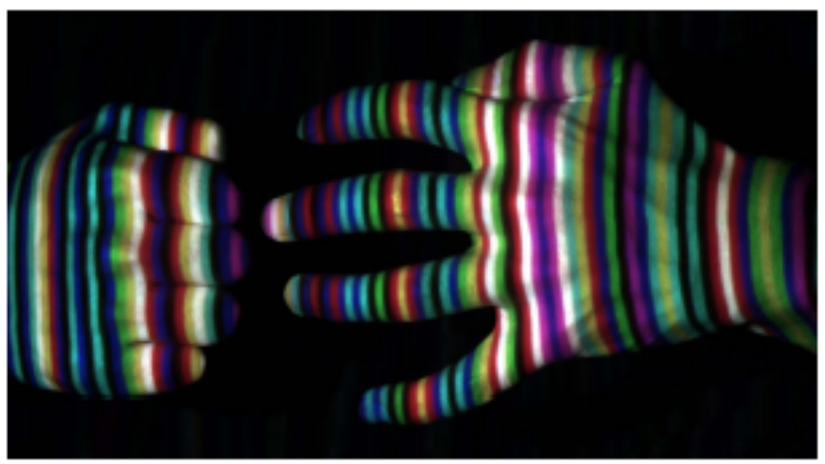
Ground truth

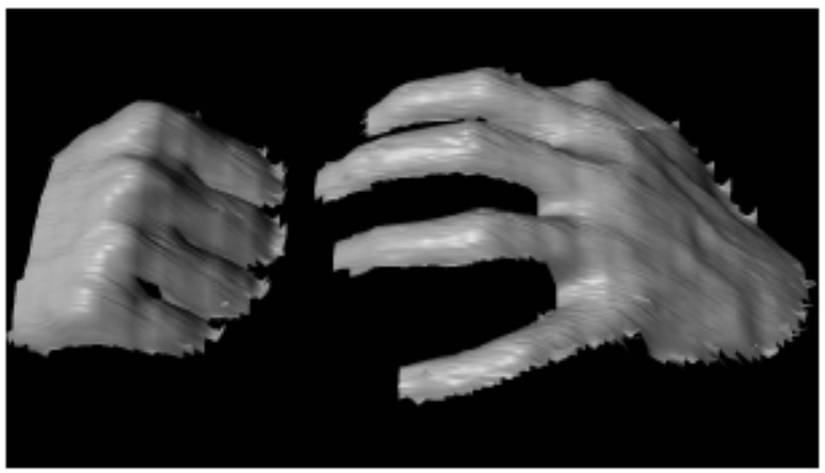
Better methods exist...

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

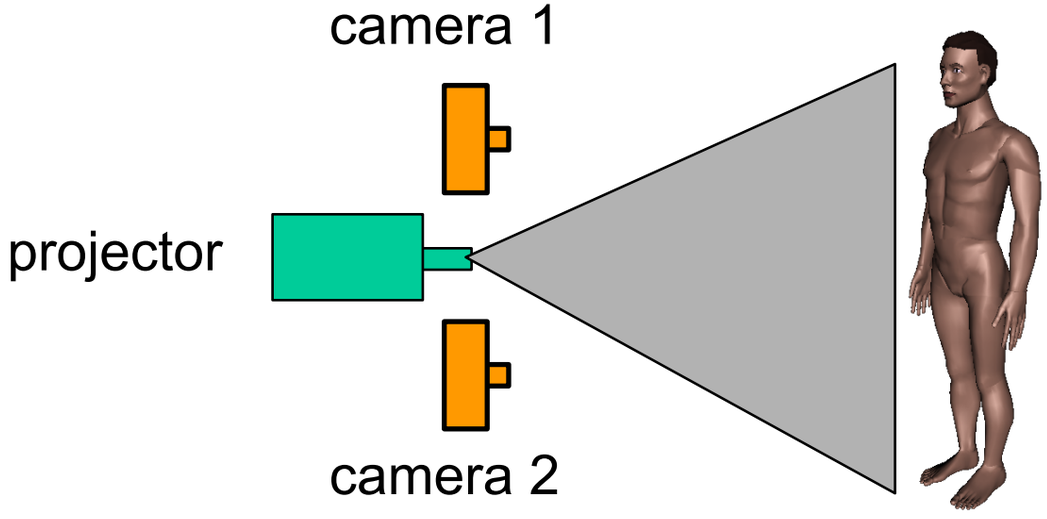
Ground truth

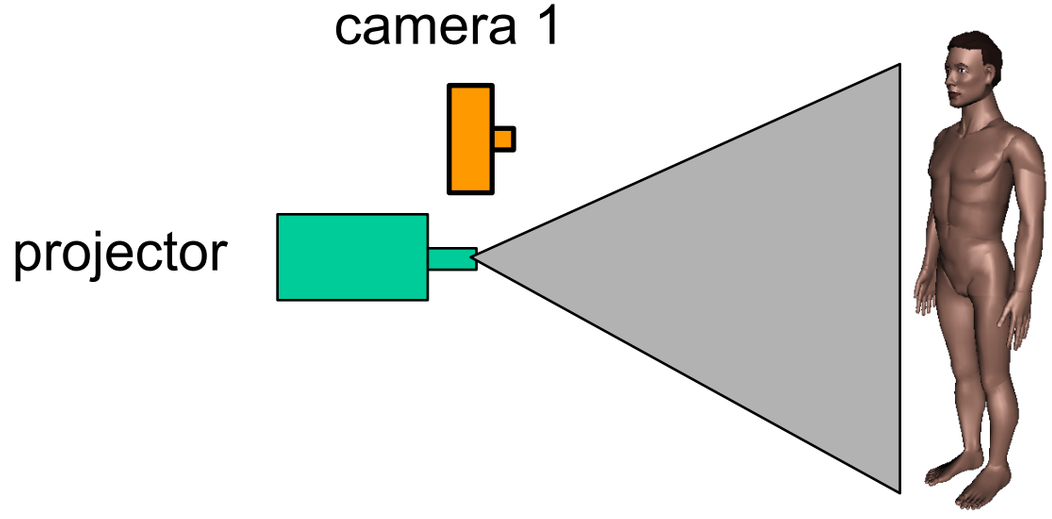
Active stereo with structured light





Li Zhang's one-shot stereo

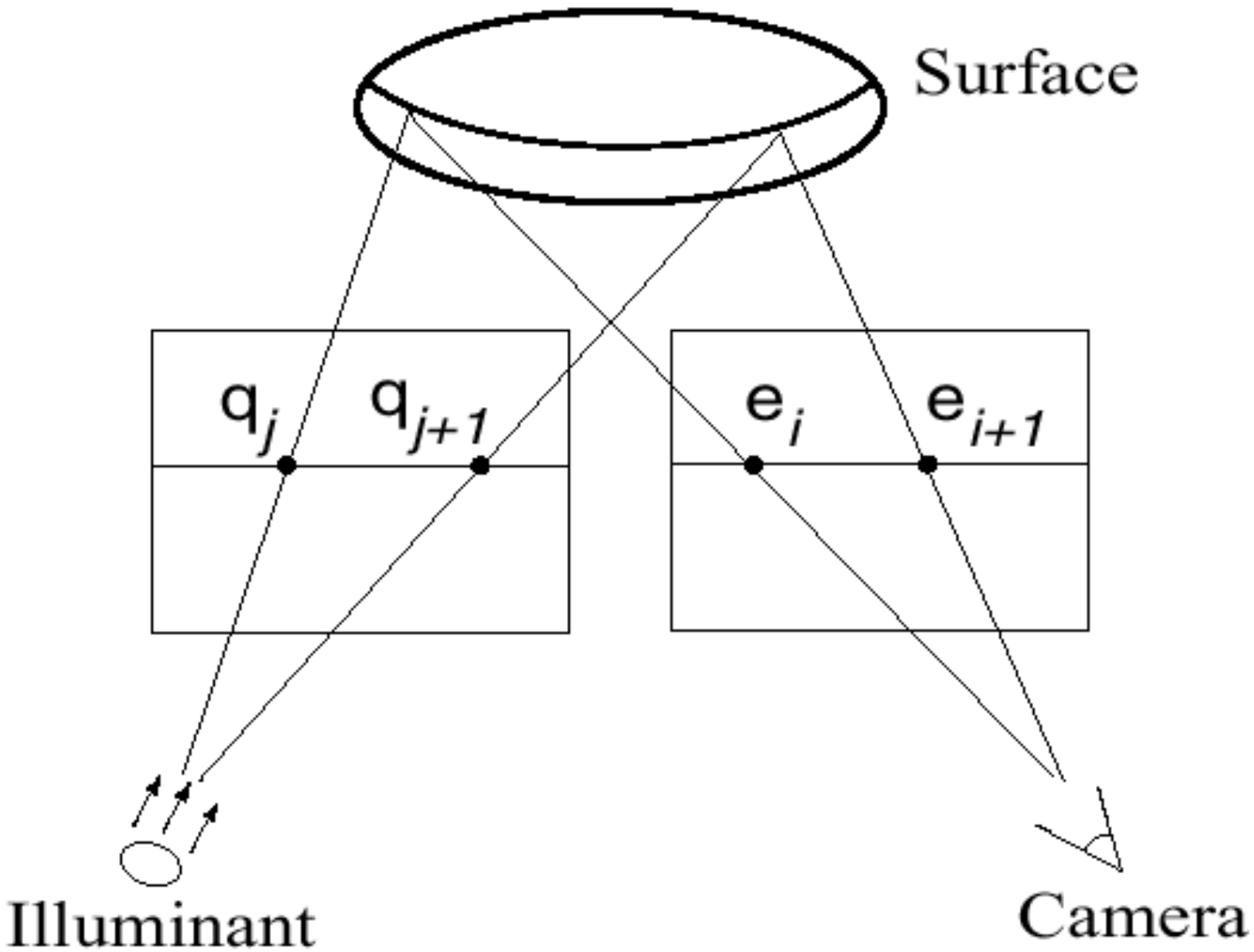




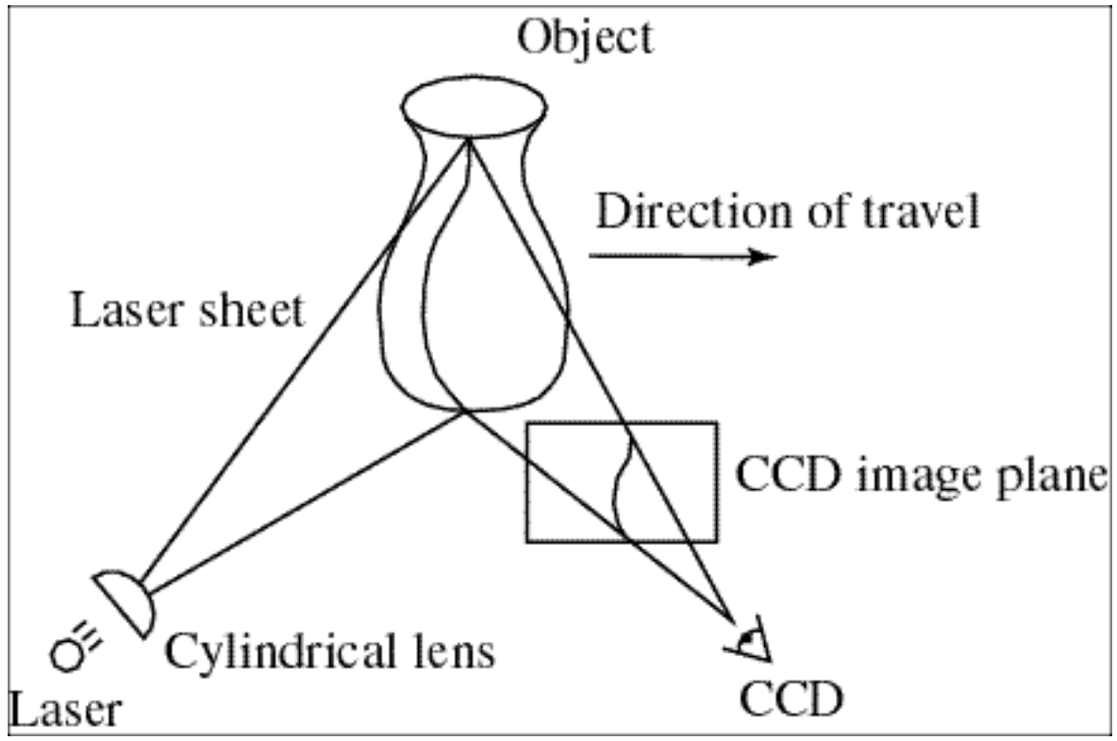
Project "structured" light patterns onto the object

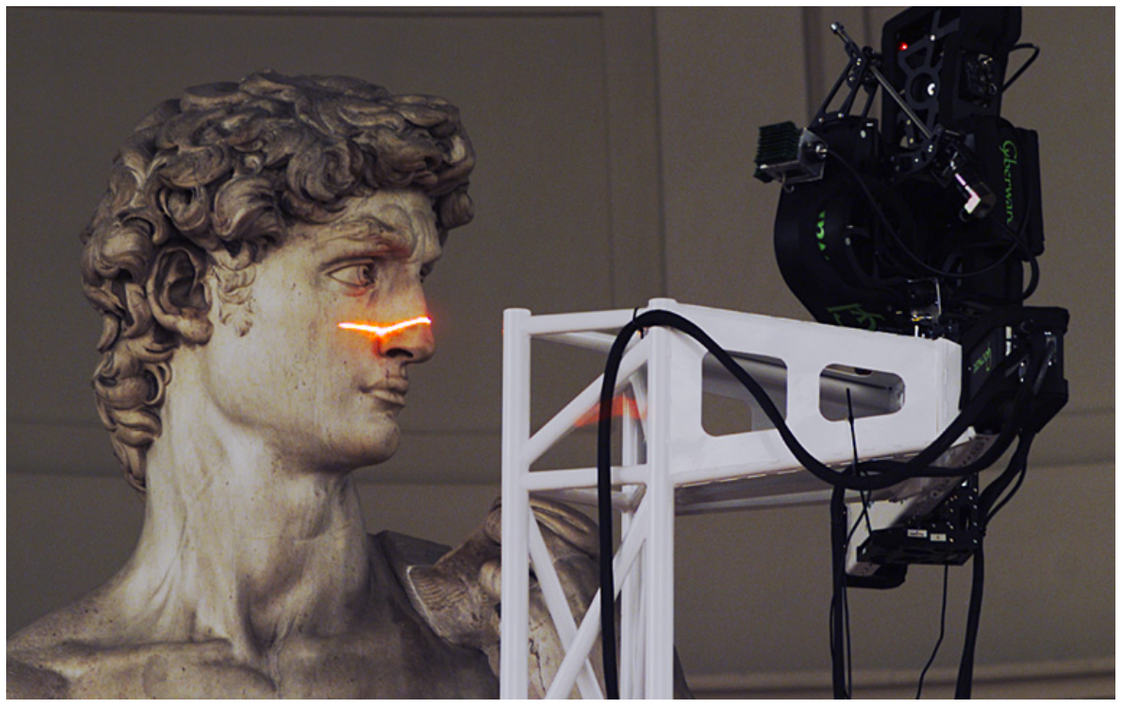
- simplifies the correspondence problem

Active stereo with structured light



Laser scanning

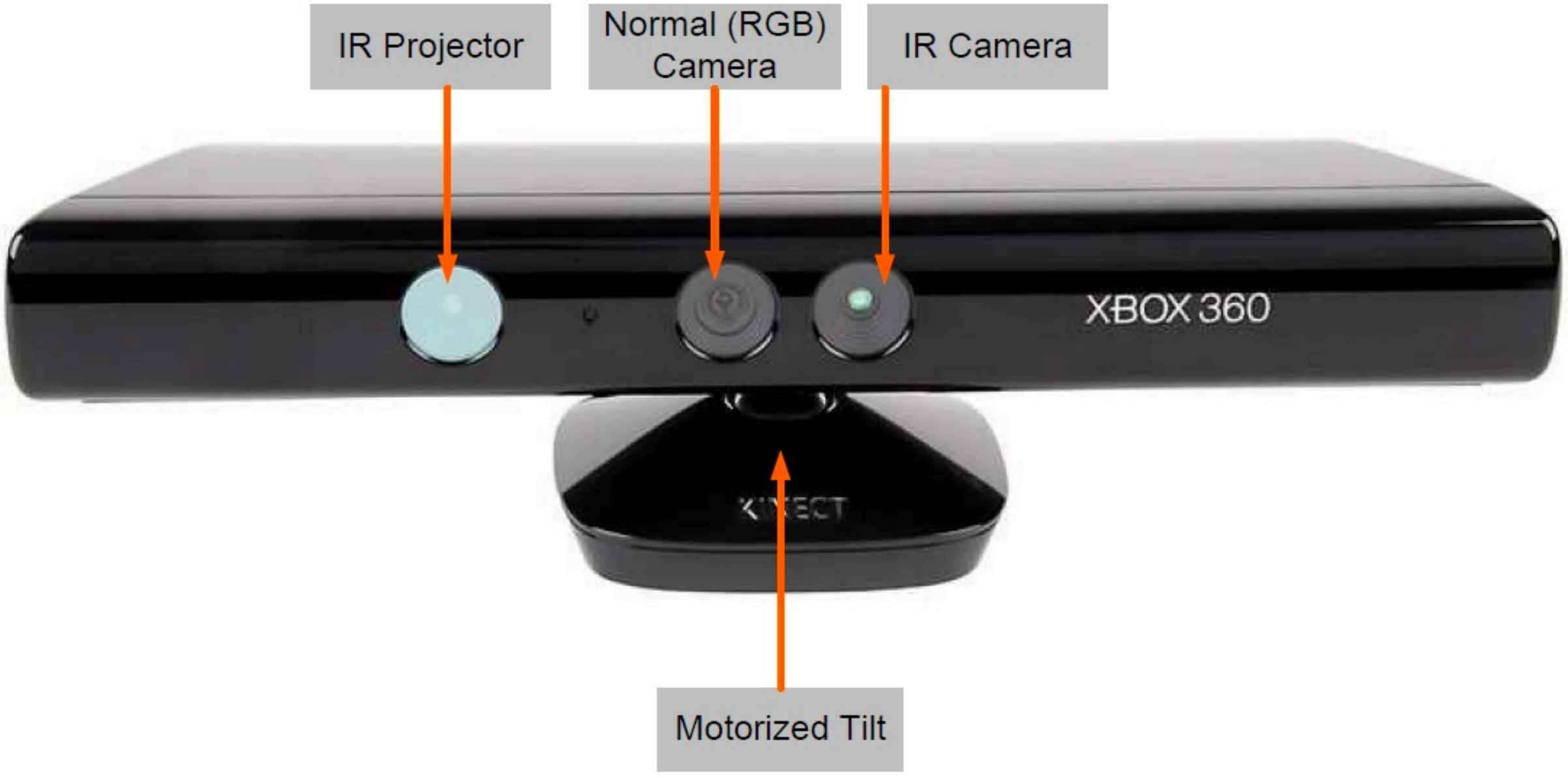


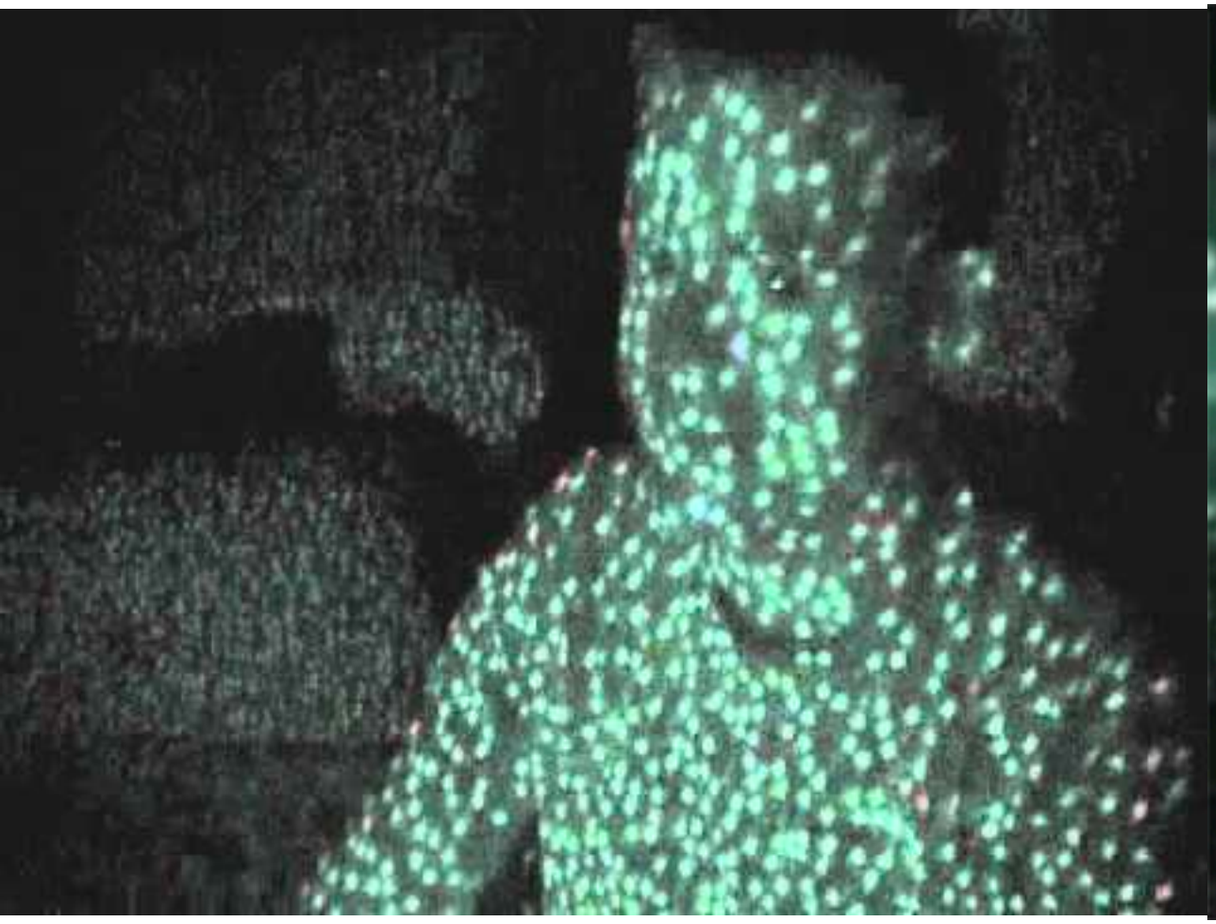


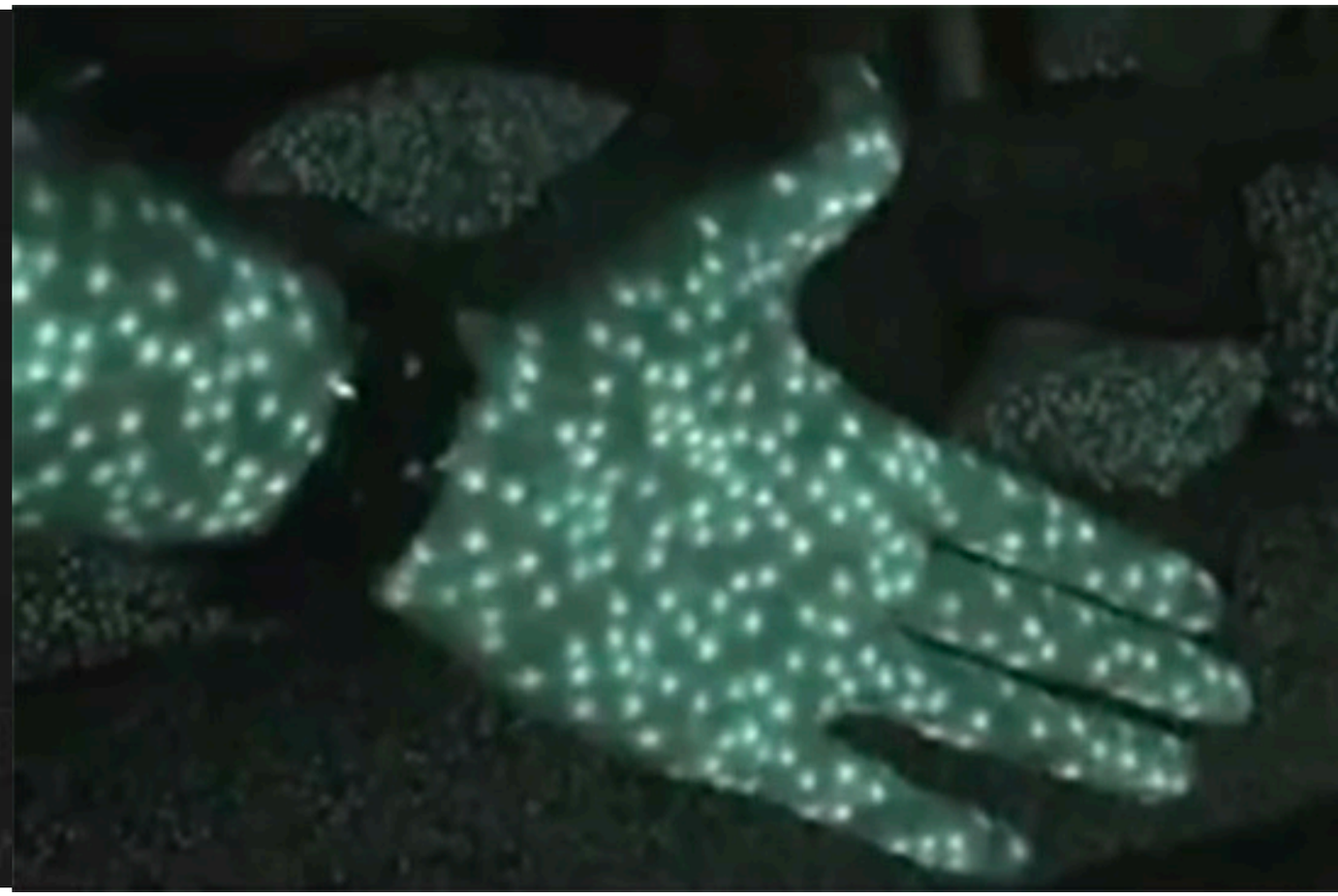
Digital Michelangelo Project
http://graphics.stanford.edu/projects/mich/

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


IR Projector
Normal (RGB) Camera
IR Camera
XBOX 360
Motorized Tilt





Next Lecture

- Tracking - last lecture