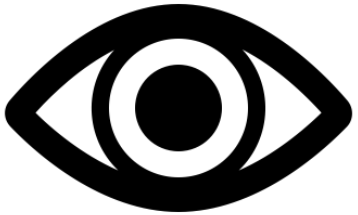


Advanced Image Processing

Recovering Depth from Stereo

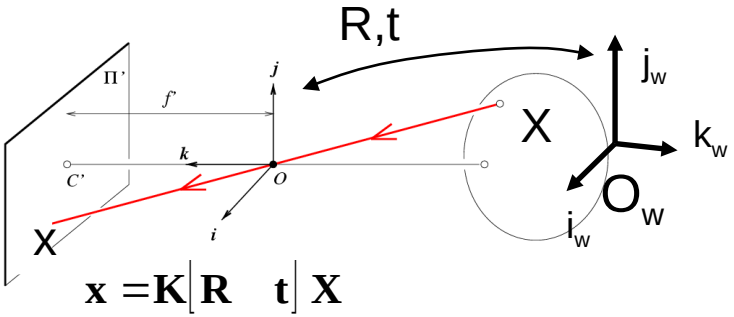
SYDE 671

Computer vision as world measurement

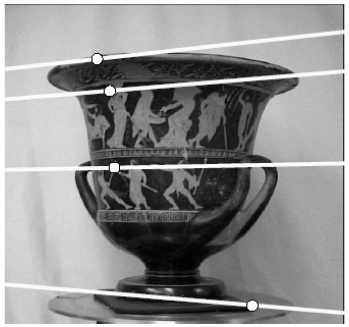
Two cameras, simultaneous views

Single moving camera and static scene

Multiple view geometry

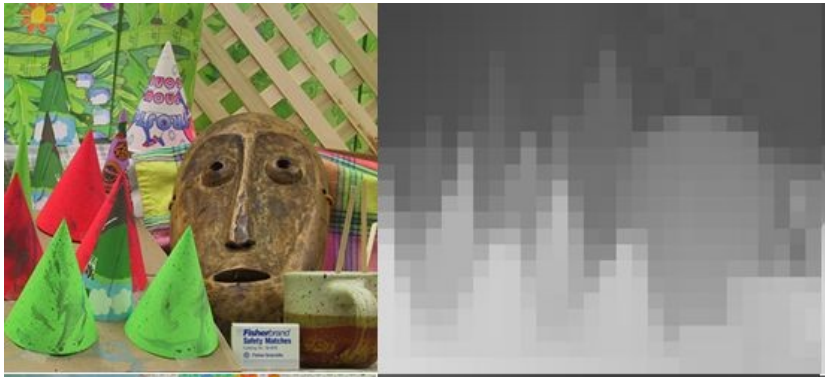


Camera calibration



Hartley and Zisserman

Epipolar geometry



Dense depth map estimation

Fundamental matrix

The matrix F is called

- the “Essential Matrix”
  - when image intrinsic parameters are known
- the “Fundamental Matrix”
  - when image intrinsics are unknown (uncalibrated case)

Can solve for F from point correspondences

Each *(x, x’)* pair gives one linear equation in entries of F

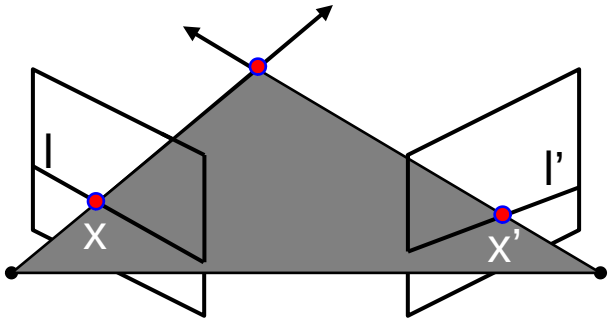
$$x' F x = 0$$

F has 9 entries, but really only 7 degrees of freedom.

With 8 points it is simple to solve for F, but it is also possible with 7. See Marc Pollefey’s notes for a nice tutorial

Fundamental matrix

Let *x* be a point in left image, *x'* in right image



Epipolar relation

x maps to epipolar line *l'*

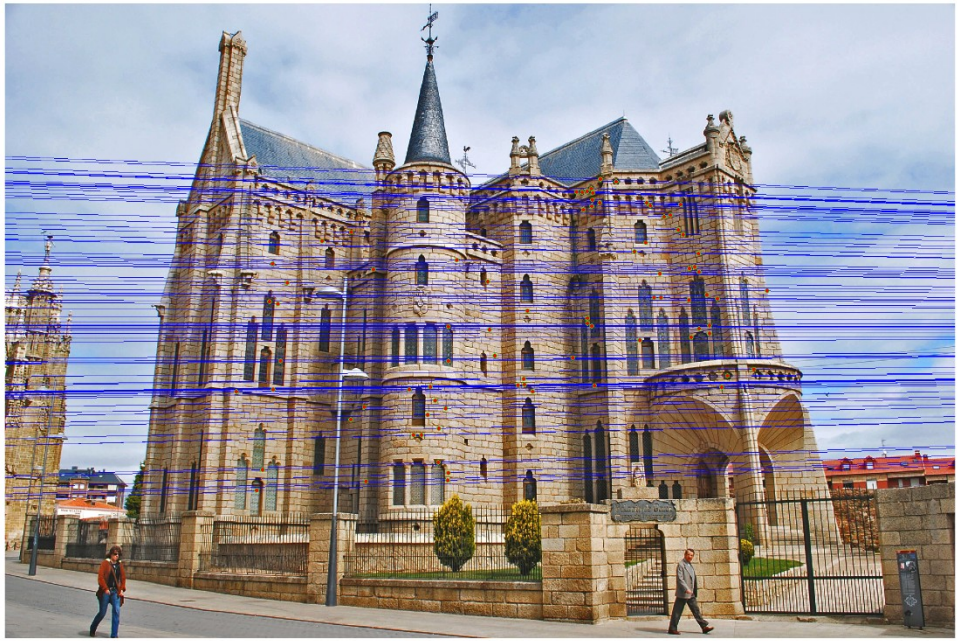
x' maps to epipolar line *l*

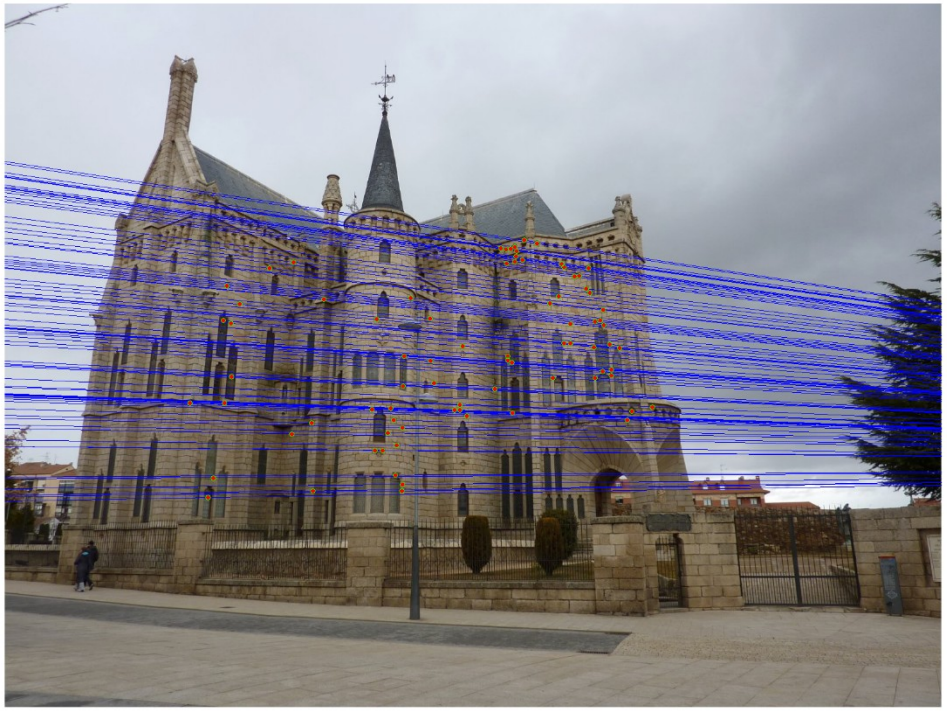
Epipolar mapping described by a 3x3 matrix *F:*

$$l' = Fx$$
$$l = F^T x'$$

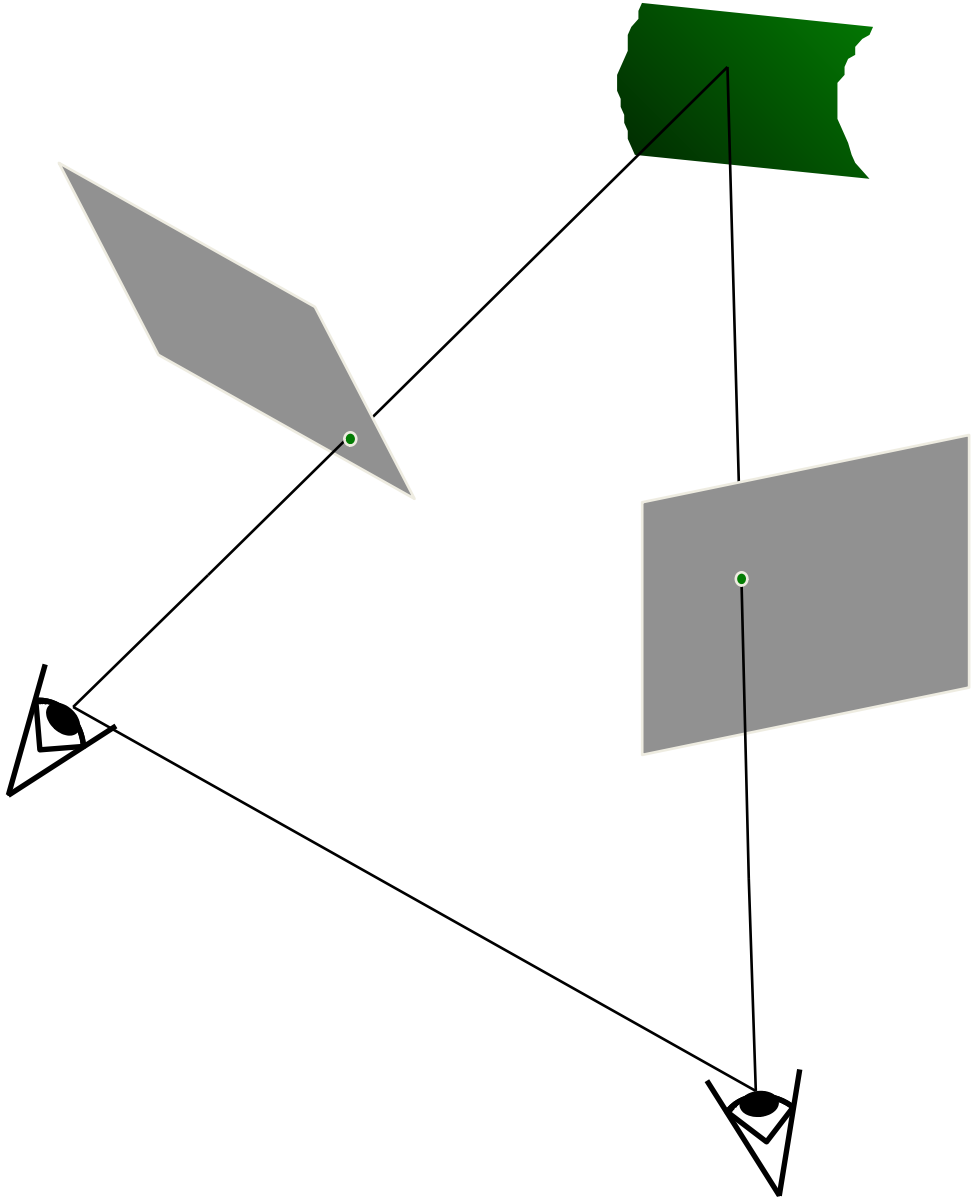
It follows that: $x' F x = 0$

Epipolar lines





Stereo image rectification

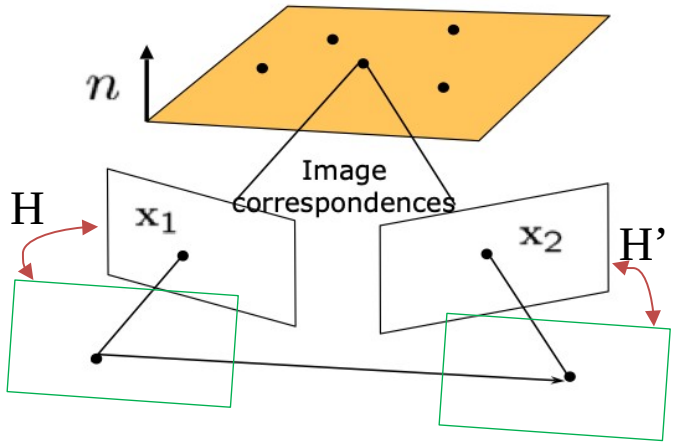


Stereo image rectification

A homography is a mapping from one plane to another.

- Stereo rectification goal is to find two homographies (one for each for each view) that can be used to “warp” the images into new images where epipolar lines are horizontal.

- Horizontal epipolar lines means the epipoles are at infinity.



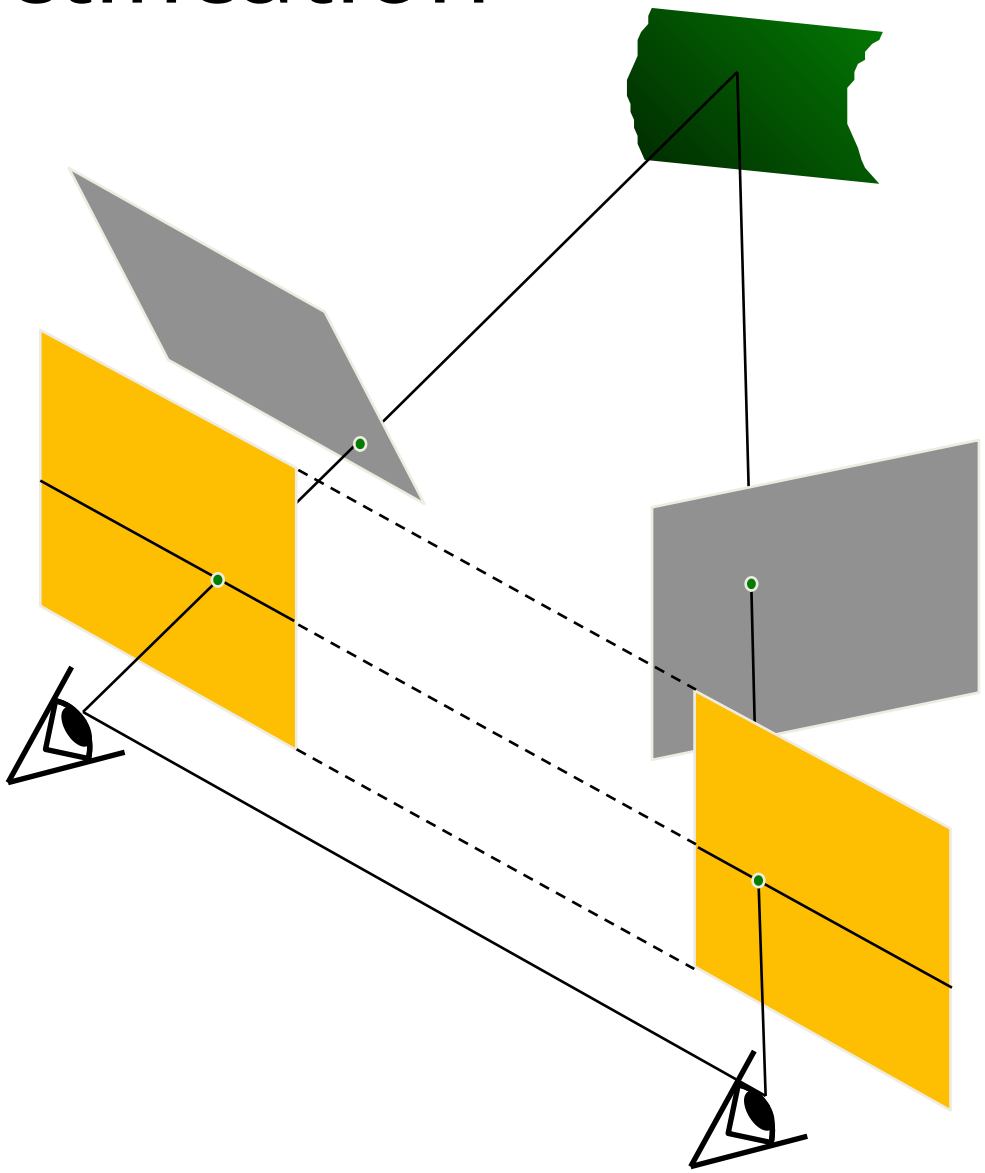
Stereo Rectification algorithm

1. Identify a seed set of image-to-image matches x <- x' between the two images. Seven points at least are needed, though more are preferable.
2. Compute the fundamental matrix F and find the epipoles e and e' in the two images.
3. Select a projective transformation H' that maps the epipole e' to the point at infinity $(1,0,0)^T$. The method of section 11.12.1 gives good results.
4. Find the matching projective transformation H that minimizes the least-squares distance $\sum_i d(Hx_i, H'x_i')$. The method used is a linear method described in section 11.12.2.
5. Resample the first image according to the projective transformation H and the second image according to the projective transformation H'.

Multiple View Geometry in Computer Vision Hartley and Zisserman

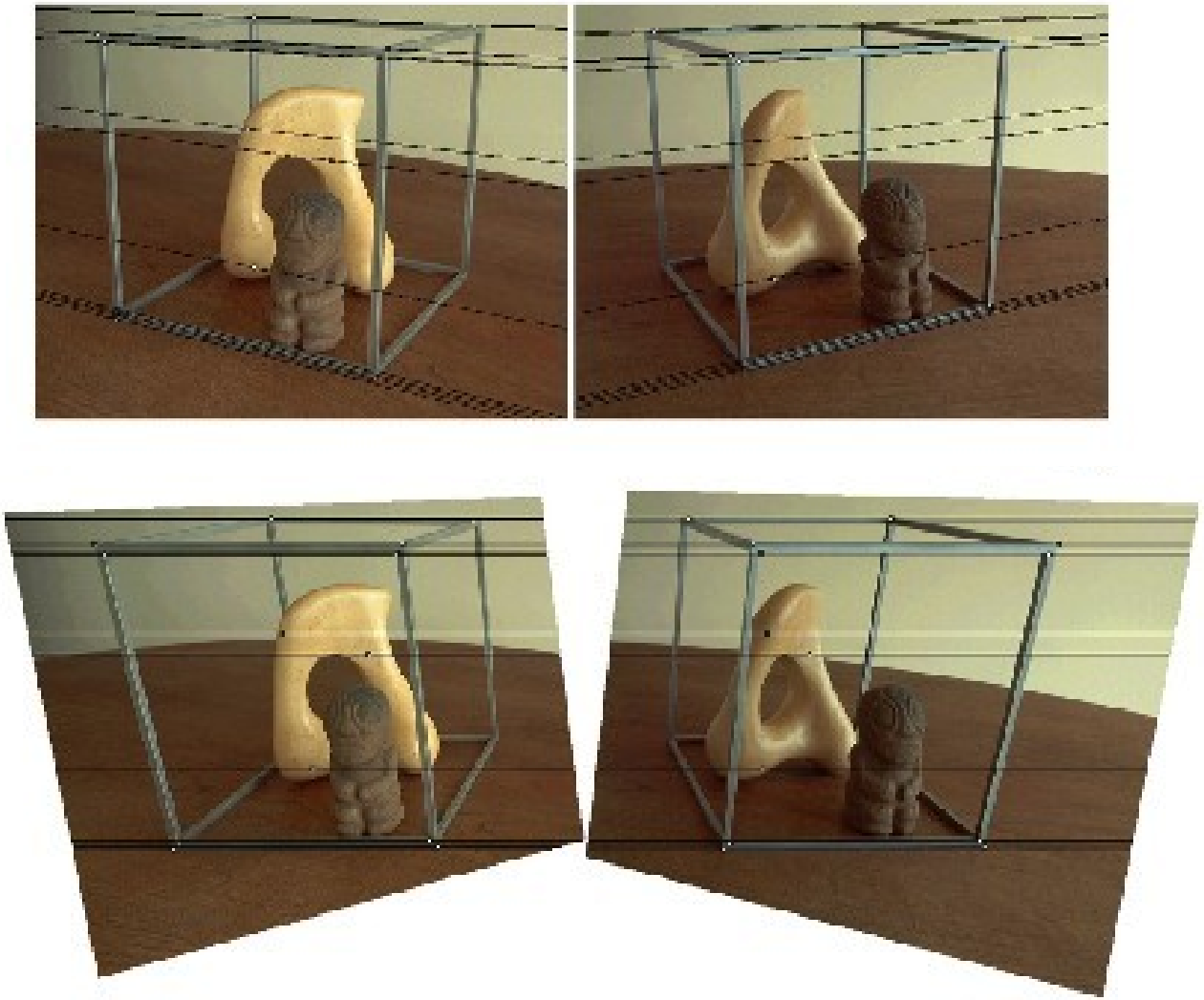
Stereo image rectification

- Reproject image planes onto a common plane parallel to the line between camera centers
- Pixel motion is horizontal after this transformation
- Two homographies (3x3 transform), one for each input image reprojection

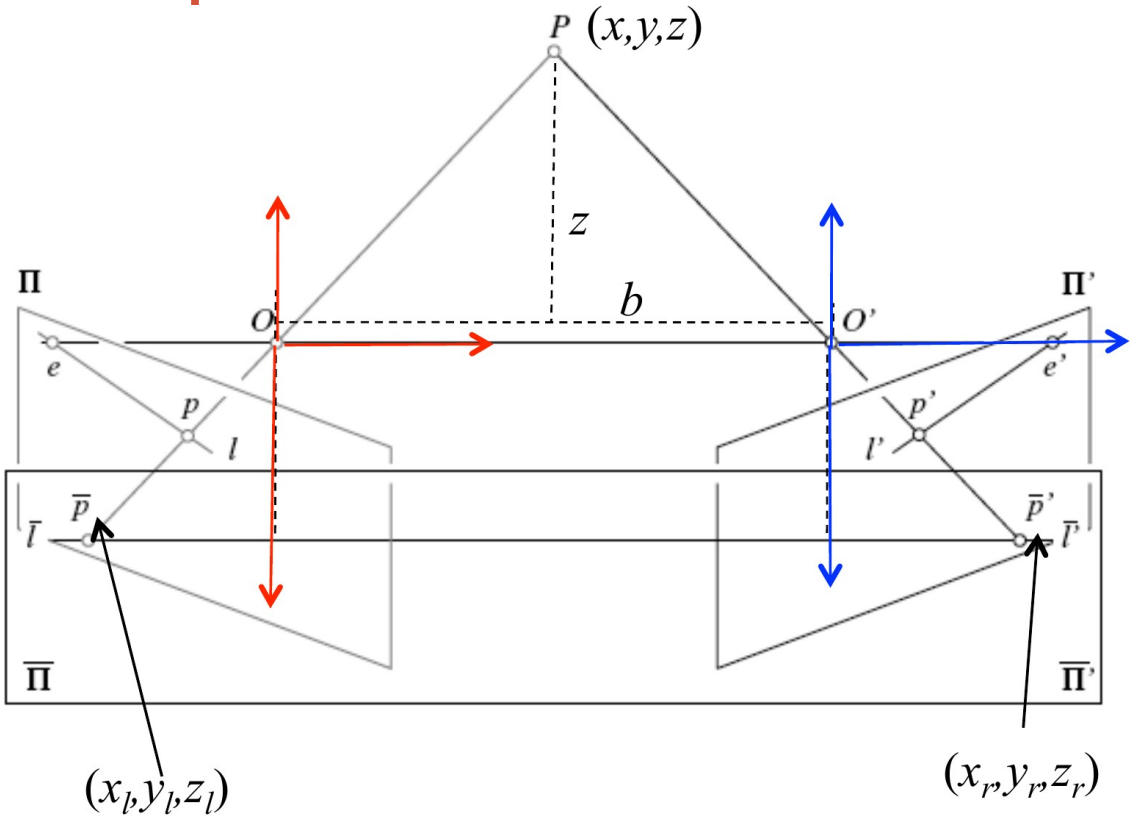


➢ C. Loop and Z. Zhang. Computing Rectifying Homographies for Stereo Vision (extra reading material).

Rectification example



Depth estimation from rectified images

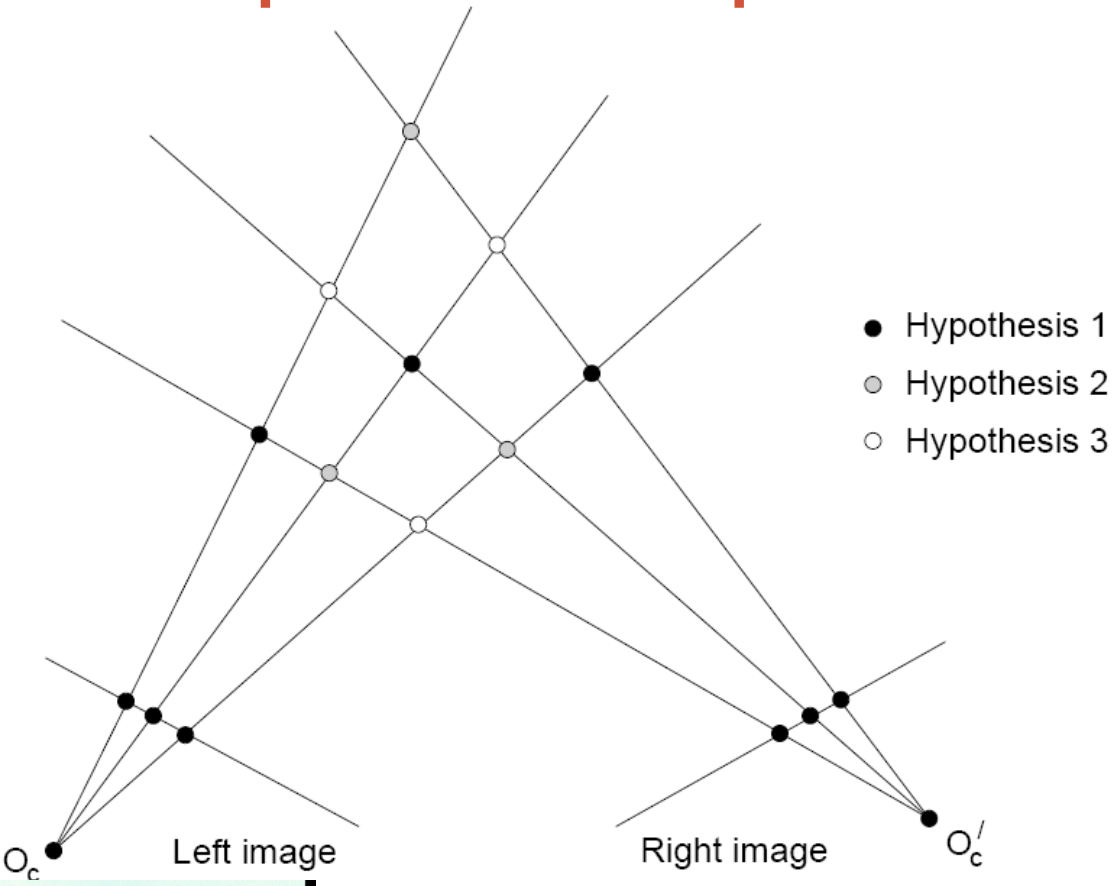


$$\frac{T + x_l - x_r}{Z - f} = \frac{T}{Z}$$

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

disparity

Correspondence problem



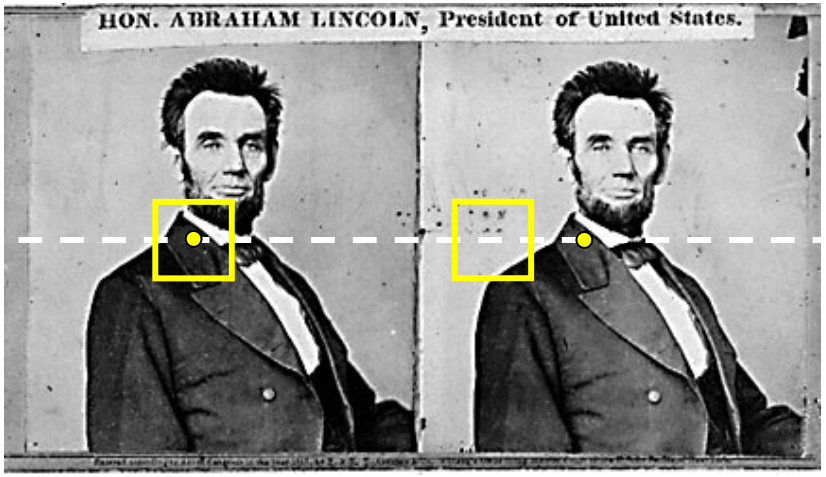
Multiple match hypotheses satisfy epipolar constraint, but which is correct?

Figure from Gee & Cipolla 1999

Dense correspondence search

Assumptions:

1. Most scene points visible from both views
2. Image regions for the matches are similar in appearance



For each epipolar line:

For each pixel / window in the left image:

Compare with every pixel / window on same epipolar line in right image
Pick position with minimum match cost (e.g., SSD, normalized correlation)

Adapted from Li Zhang

Correspondence search process

- Define a square window in the image with window size m x m

$$W_m(x,y) = \{u,v \mid x - \tfrac{m}{2} \le u \le x + \tfrac{m}{2}, y - \tfrac{m}{2} \le v \le y + \tfrac{m}{2}\}$$

- SSD (Sum of Squared distances): $C_r(x,y,d) = \sum_{(u,v)\in W_m(x,y)} [I_L(u,v) - I_R(u-d,v)]^2$

- Exposure time and illumination variation may cause the left and right intensity windows to look different -> need to normalize them:

$$\bar{I} = \frac{1}{|W_m(x,y)|} \sum_{(u,v)\in W_m(x,y)} I(u,v) \qquad \text{Average pixel}$$

$$\|I\|_{W_m(x,y)} = \sqrt{\sum_{(u,v)\in W_m(x,y)} [I(u,v)]^2} \qquad \text{Window magnitude}$$

$$\hat{I}(x,y) = \frac{I(x,y) - \bar{I}}{\|I - \bar{I}\|_{W_m(x,y)}} \qquad \text{Normalized pixel}$$

NSSD (Normalized SSD): $C_{\text{SSD}}(d) = \sum_{(u,v)\in W_m(x,y)} [\hat{I}_L(u,v) - \hat{I}_R(u-d,v)]^2$

A match is found where SSD or NSSD are the smallest.

Correspondence search process

Many more metrics:

- Normalized Correlation
- Cross Correlation
- Zero Mean Normalized Cross-Correlation (commonly used):

$$\mathrm{ZNCC}(x,y) = \frac{\sum_{j=1}^{N}\sum_{i=1}^{M}[I(x+i,y+j)-\mu(I_c(x,y))]\cdot[T(i,j)-\mu(T)]}{\sqrt{\sum_{j=1}^{N}\sum_{i=1}^{M}[I(x+i,y+j)-\mu(I_c(x,y))]^2}\cdot\sqrt{\sum_{j=1}^{N}\sum_{i=1}^{M}[T(i,j)-\mu(T)]^2}}$$

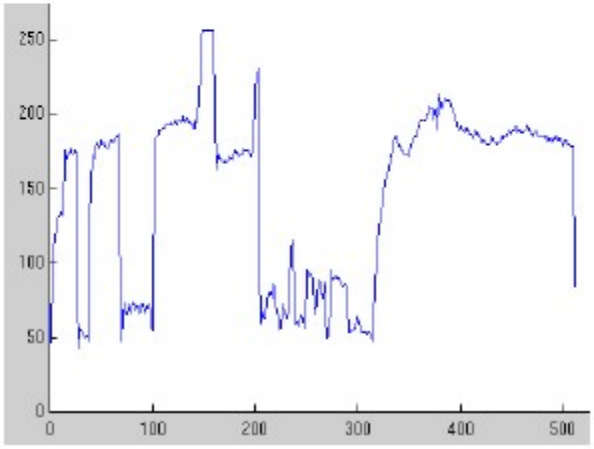
- patches are similar when ZNCC = 1
- patches are dissimilar when ZNCC = -1

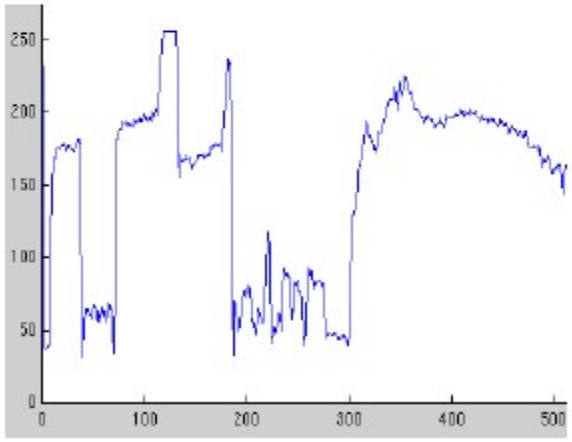
```
def zncc(img1, img2, u1, v1, u2, v2, n):
    stdDeviation1 = getStandardDeviation(img1, u1, v1, n)
    stdDeviation2 = getStandardDeviation(img2, u2, v2, n)
    avg1 = getAverage(img1, u1, v1, n)
    avg2 = getAverage(img2, u2, v2, n)

    s = 0
    for i in range(-n, n + 1):
        for j in range(-n, n + 1):
            s += (img1[u1 + i][v1 + j] - avg1) * (img2[u2 + i][v2 + j] - avg2)
    return float(s) / ((2 * n + 1) ** 2 * stdDeviation1 * stdDeviation2)
```

Correspondence problem

Intensity profiles



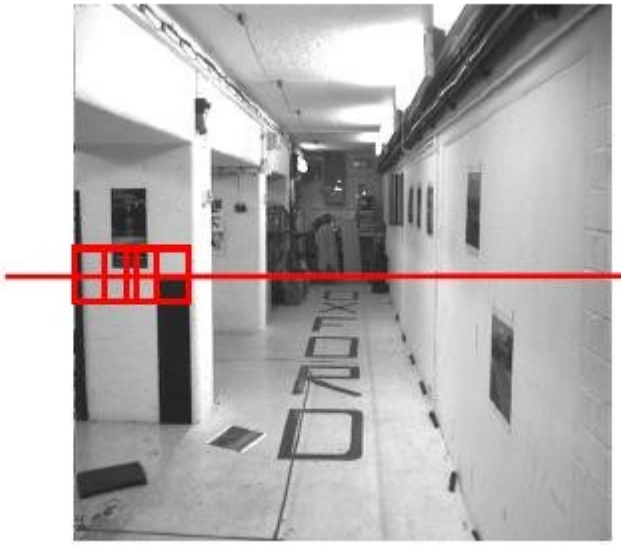


- Clear correspondence between intensities, but also noise and ambiguity

Parallel camera example: epipolar lines are corresponding image scanlines

Source: Andrew Zisserma

Correspondence problem



Neighborhoods of corresponding points are similar in intensity patterns.

Source: Andrew

Cross Correlation-based window matching

Cross Correlation-based window matching

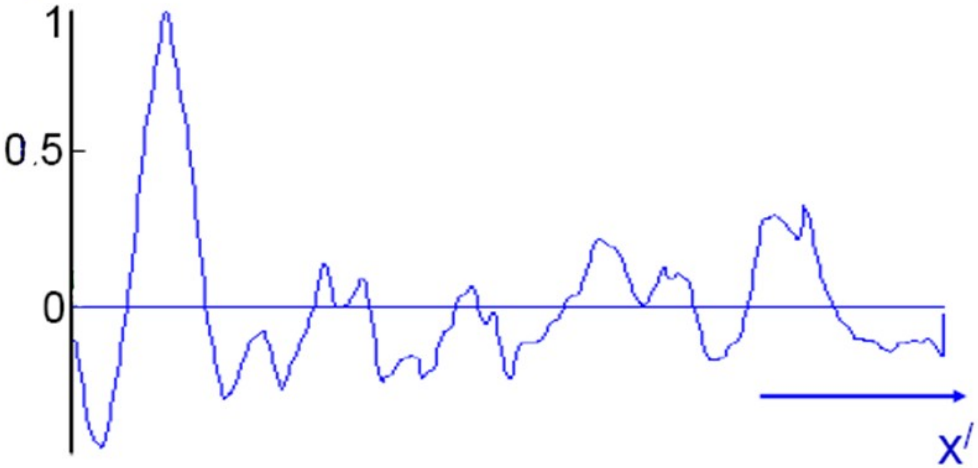
left image band (x)

right image band (x')

Cross Correlation-based window matching

left image band (x)

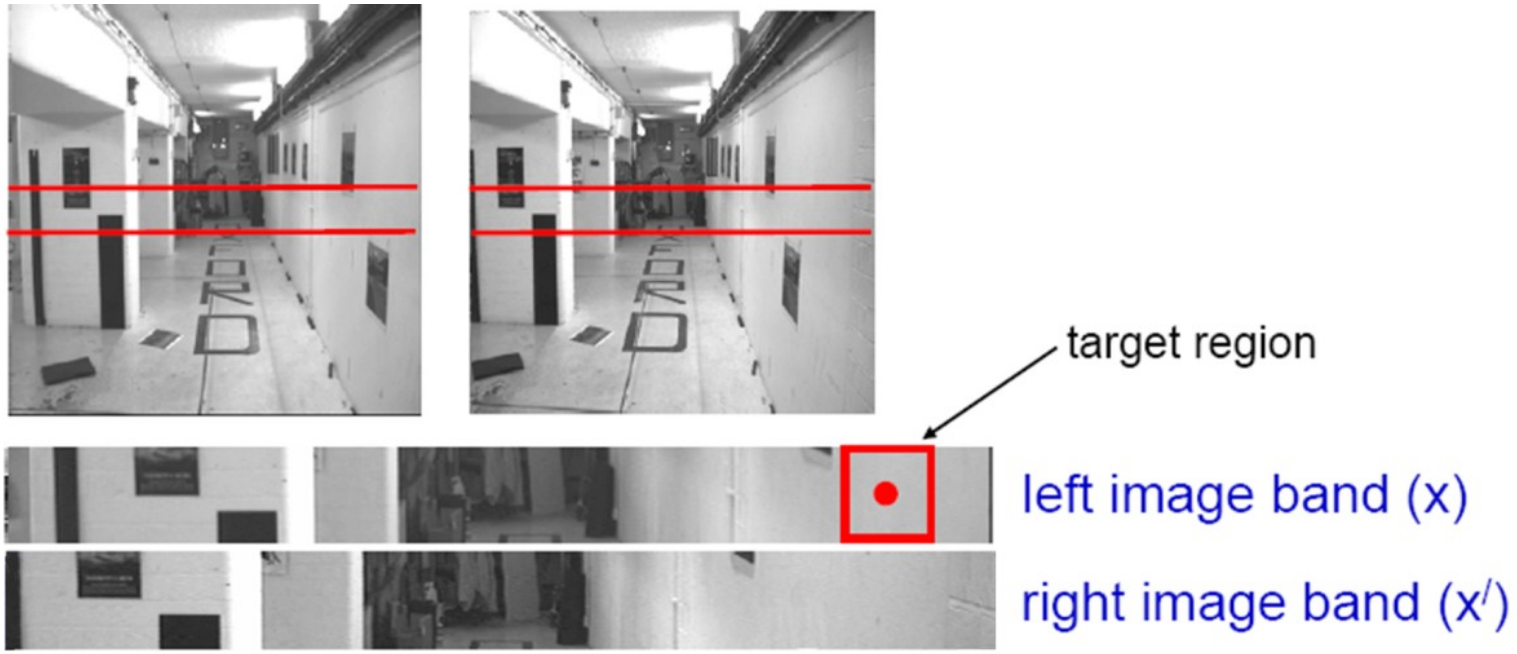
right image band (x')



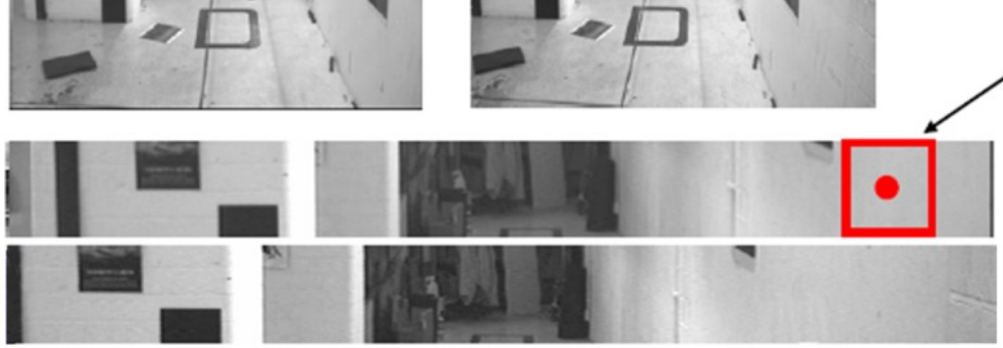
cross correlation

disparity = $x' - x$

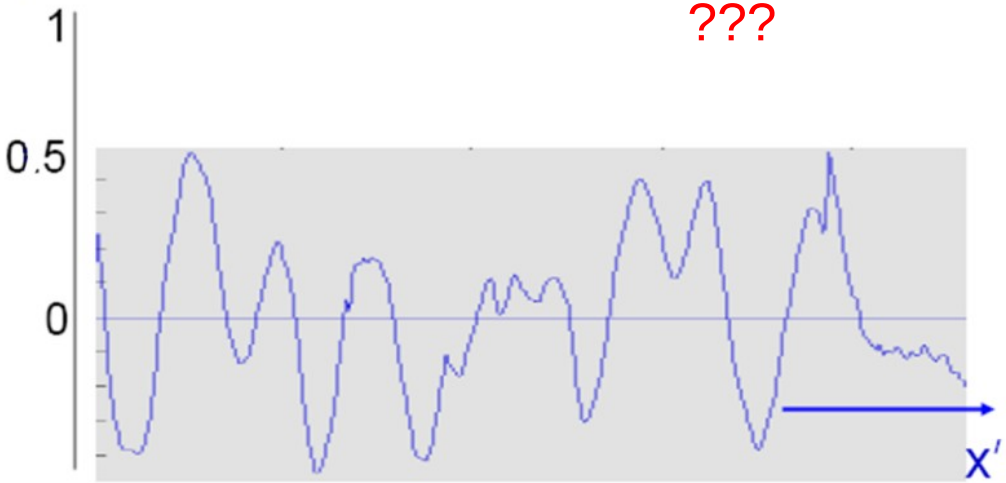
Cross Correlation-based window matching



Cross Correlation-based window matching



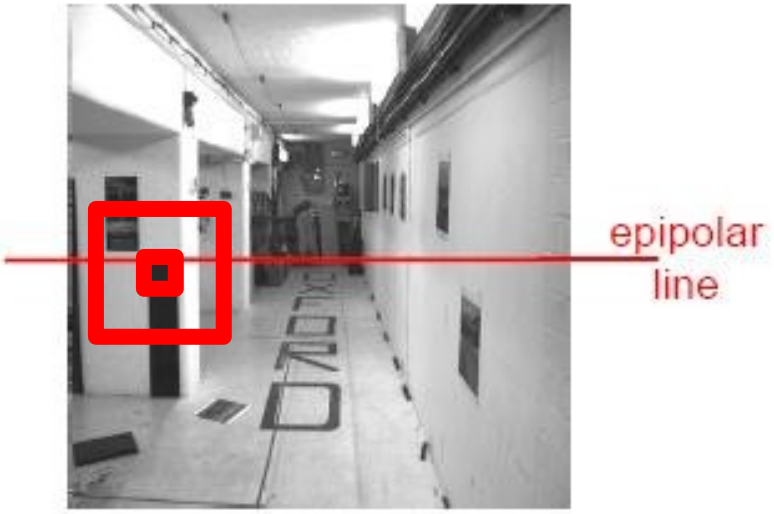
???



cross correlation

Textureless regions are non-distinct; high ambiguity for matches.

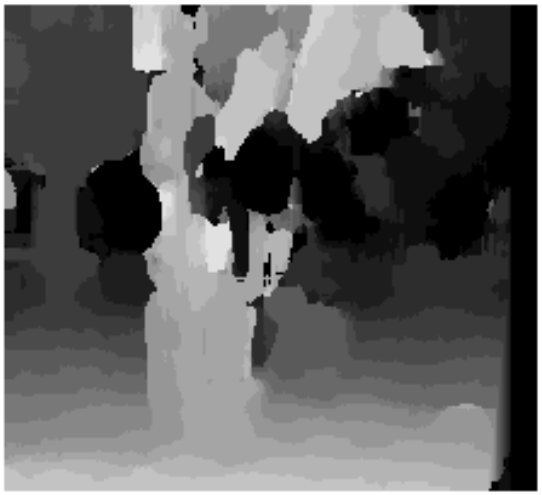
Effect of window size



Source: Andrew Zisserman

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

Correspondence constraints

- Beyond the hard constraint of epipolar geometry, there are “soft” constraints to help identify corresponding points
  - Similarity (we saw patch similarity)
  - Uniqueness
  - Disparity gradient – depth doesn’t change too quickly.
  - Ordering

Uniqueness constraint

- Up to one match in right image for every point in left image

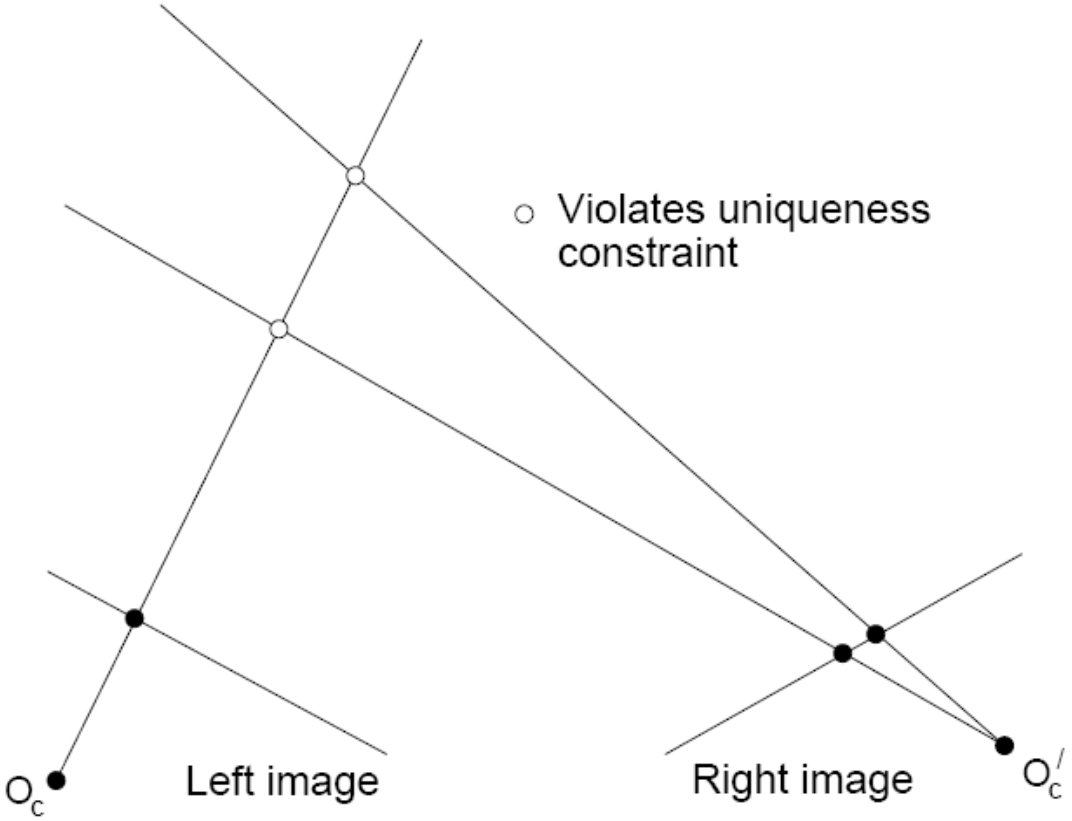
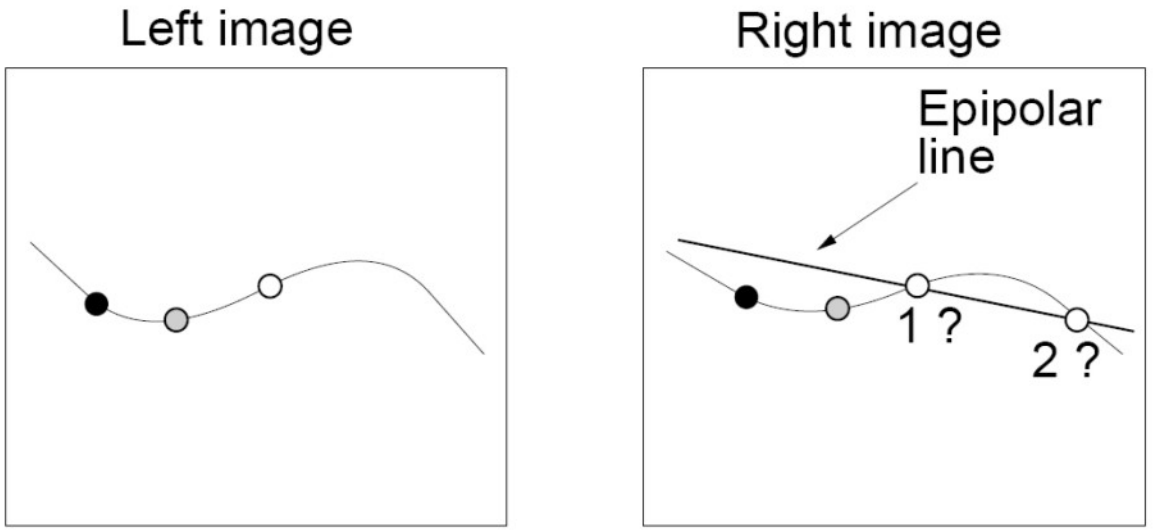


Figure from Gee & Cipolla 1999

Disparity gradient constraint

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



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

Ordering constraint

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

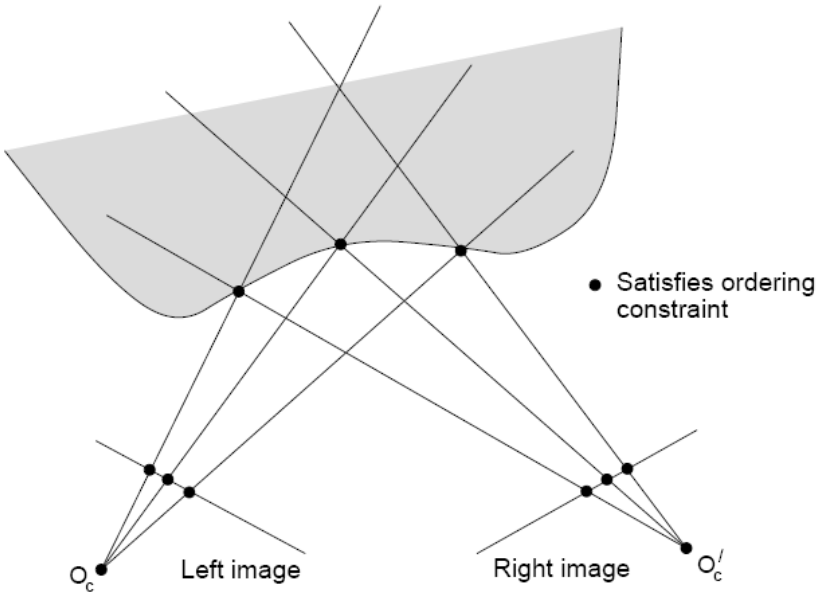
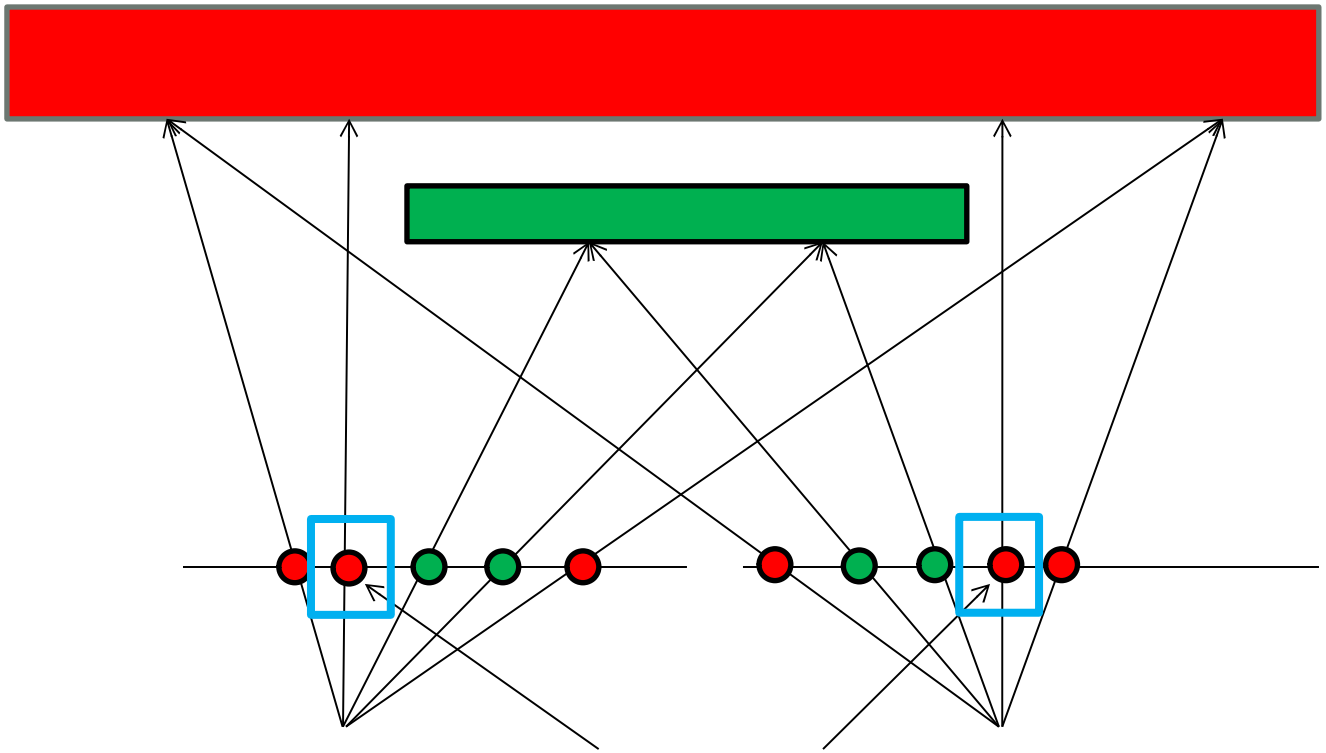


Figure from Gee & Cipolla 1999

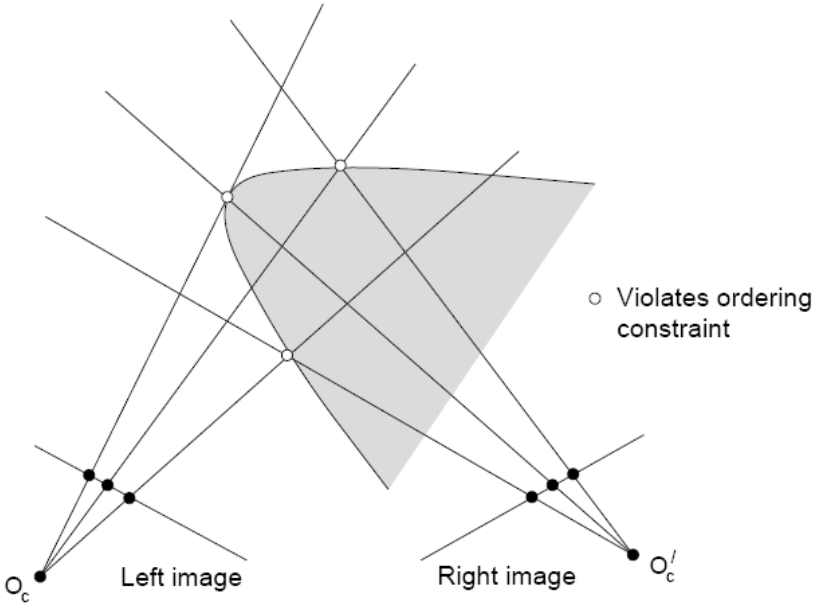
Problem: Occlusion

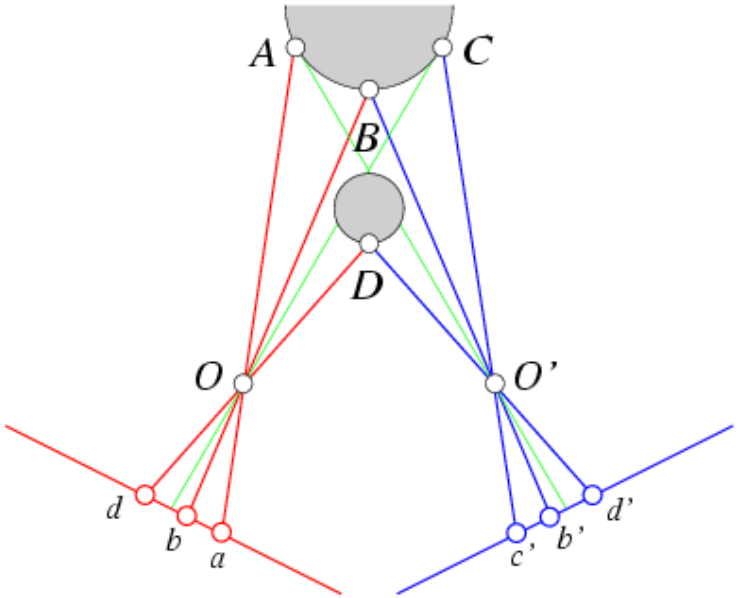
- Uniqueness says “up to one match” per pixel
- When is there no match?



Ordering constraint limitations

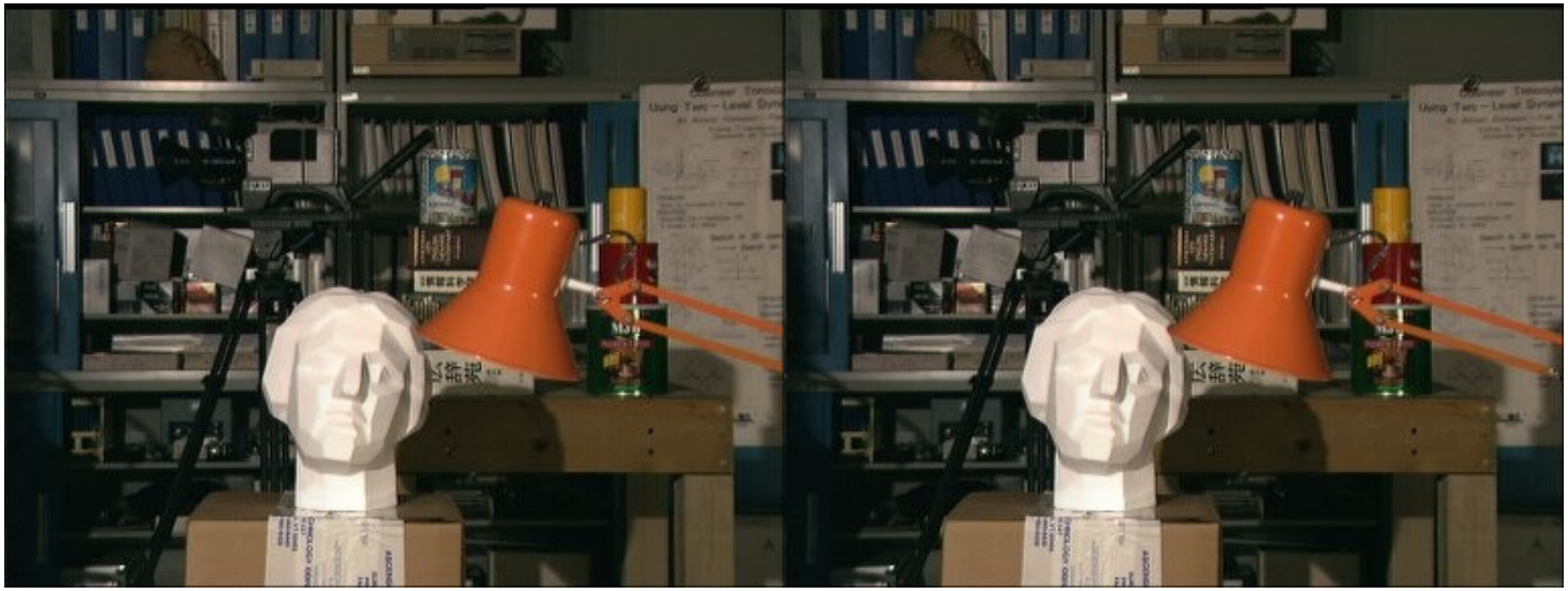
- Won't always hold, e.g. consider transparent object, or an occluding surface



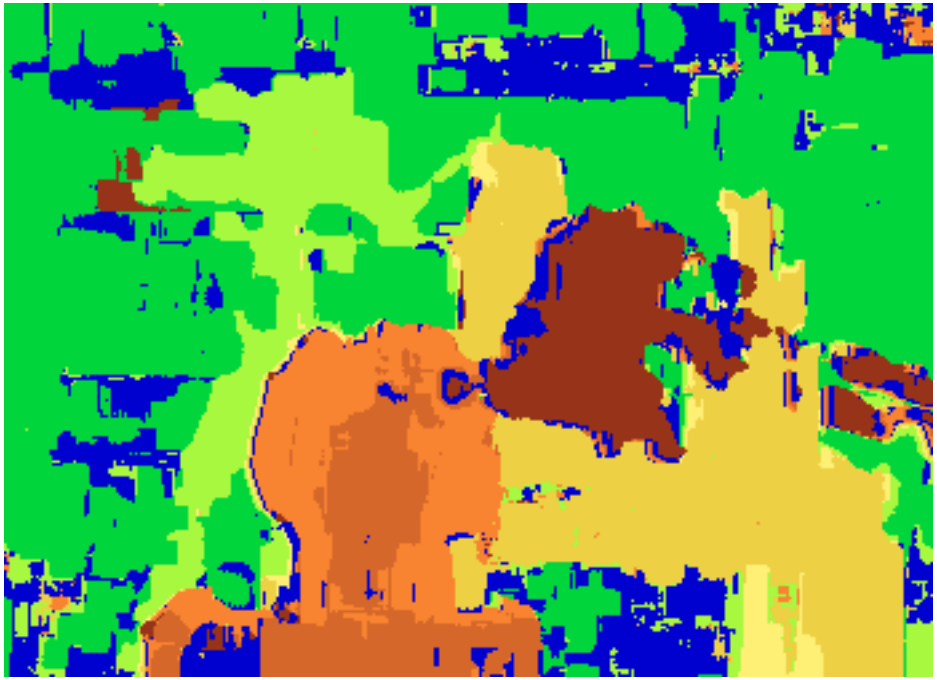


Figures from Forsyth & Ponce

Stereo – Tsukuba test scene (now old)



Results with window search



Window-based matching
(best window size)

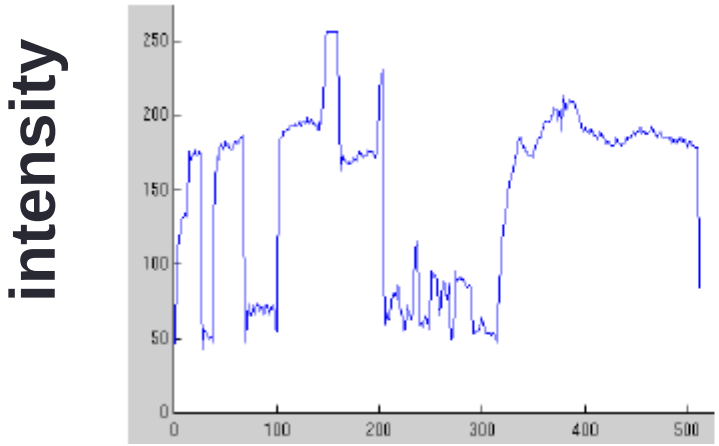
‘Ground truth’

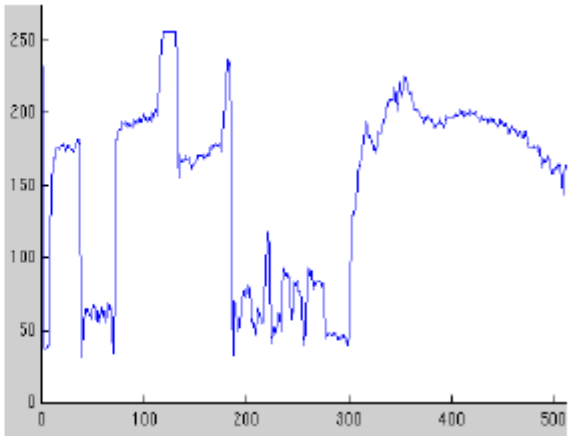
Better solutions

- Beyond individual correspondences to estimate disparities, instead optimize correspondence assignments jointly:
  - Scanline stereo
  - Energy minimization: Full 2D grid (graph cuts)

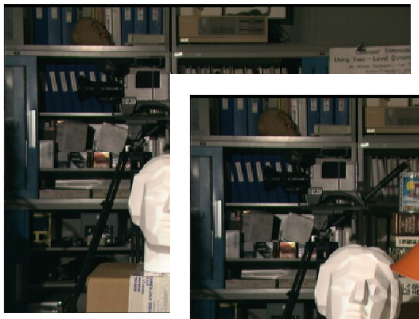
Scanline stereo

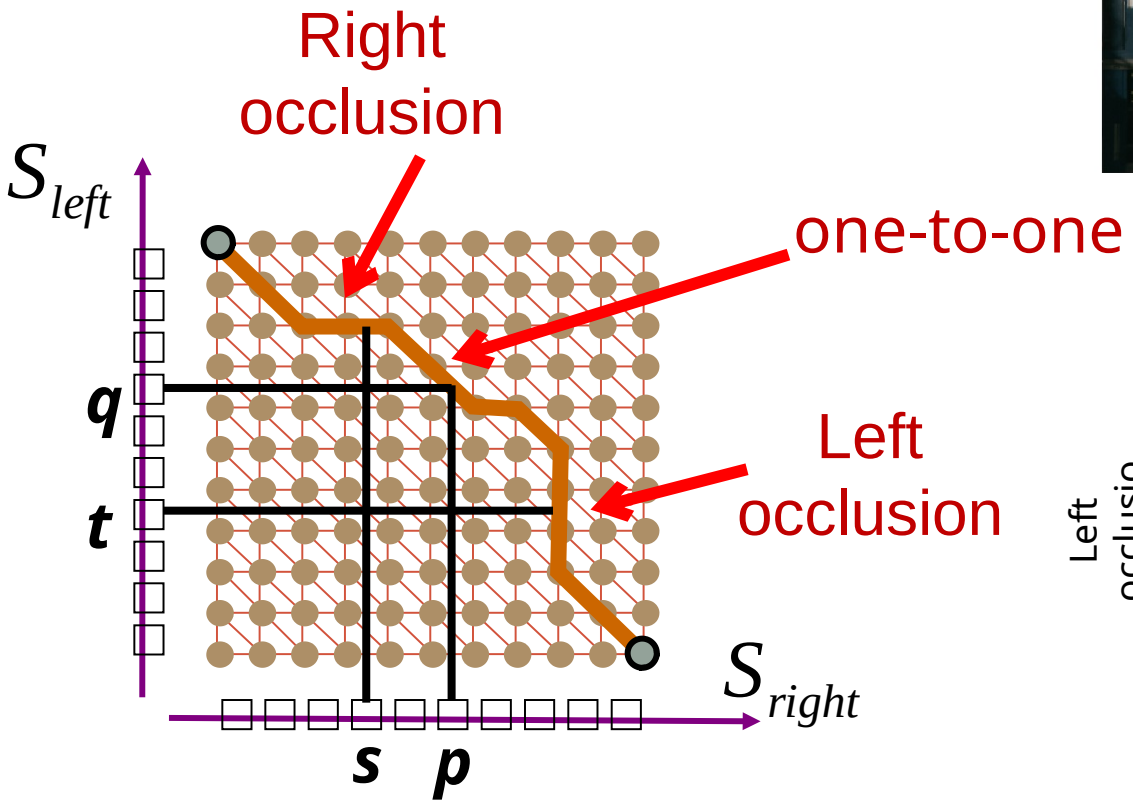
- Try to coherently match pixels on the entire scanline
- Different scanlines are still optimized independently

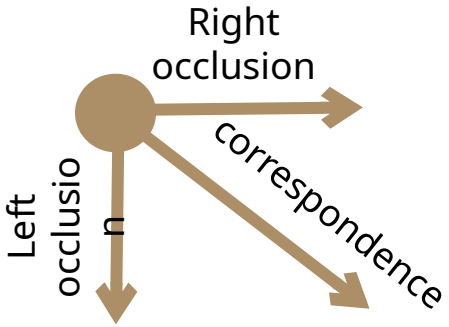




“Shortest paths” for scan-line stereo







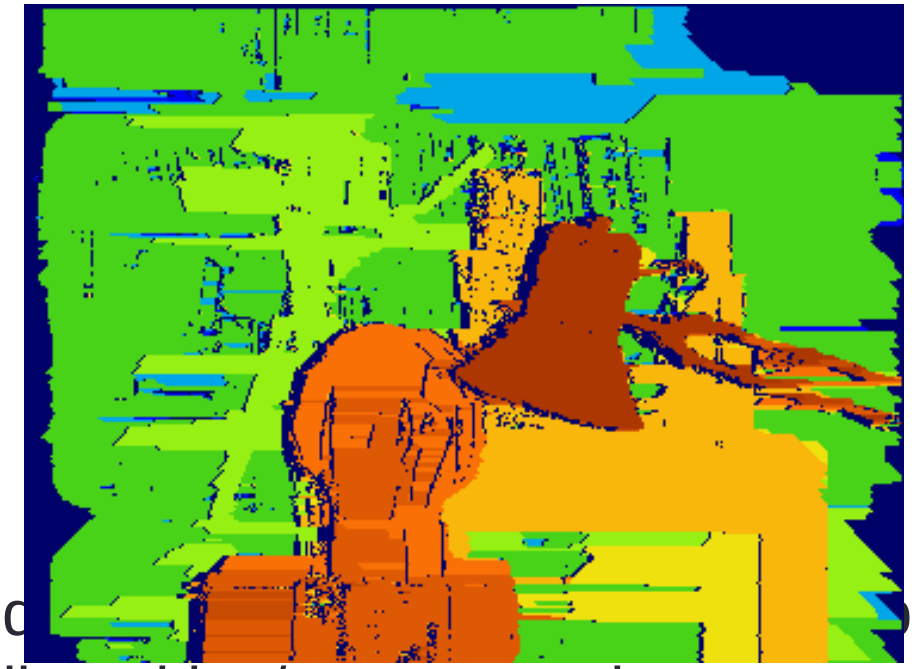
Can be implemented with dynamic programming

Ohta & Kanade ’85, Cox et al. ’96, Intille & Bobick, ‘01

Slide credit: Y. Boykov

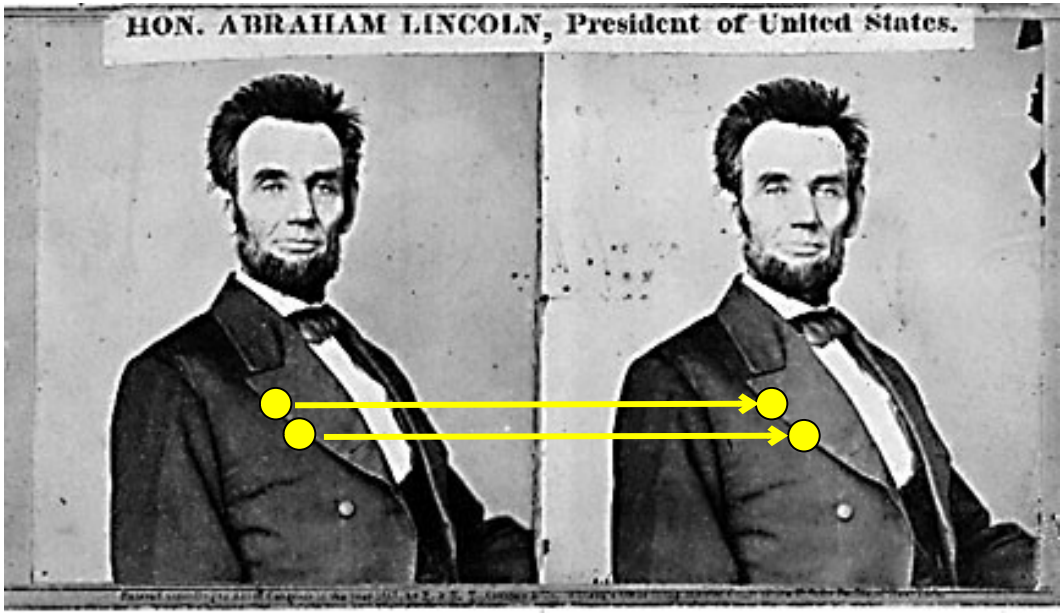
Coherent stereo on 2D grid

- Scanline stereo generates streaking artifacts



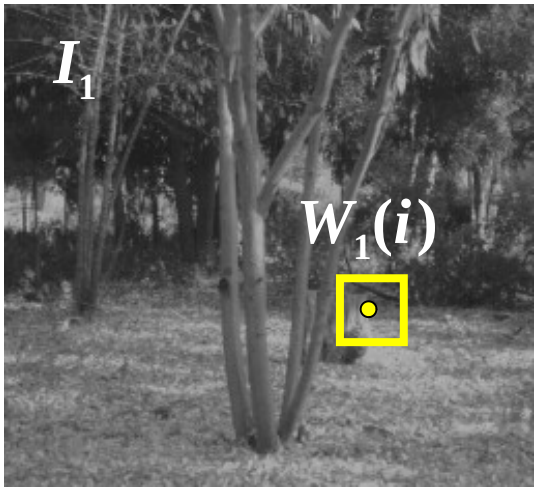
- Can’t use c atially coherent disparities/ correspondences on a 2D grid

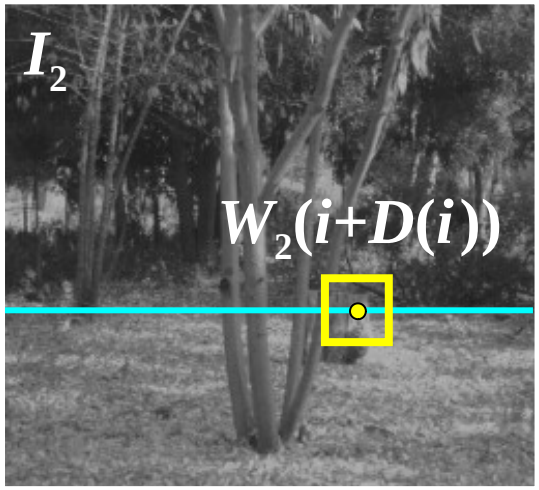
Stereo as energy minimization

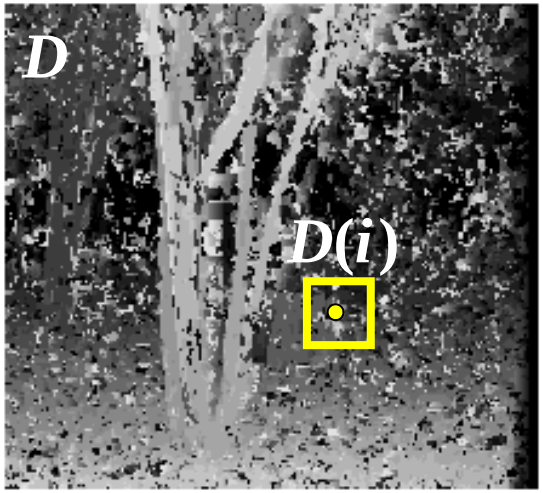


- What defines a good stereo correspondence?
  1. Match quality
     - Want each pixel to find a good match in the other image
  2. Smoothness
     - If two pixels are adjacent, they should (usually) move about the same amount

Stereo matching as energy minimization







$$E = \alpha E_{\text{data}}(I_1, I_2, D) + \beta E_{\text{smooth}}(D)$$

$$E_{\text{data}} = \sum_i \left(W_1(i) - W_2(i + D(i))\right)^2 \qquad E_{\text{smooth}} = \sum_{\text{neighbors } i,j} \rho\left(D(i) - D(j)\right)$$

Energy functions of this form can be minimized using *graph cuts.*

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

Source: Steve Seitz

Better results…

Graph cut method

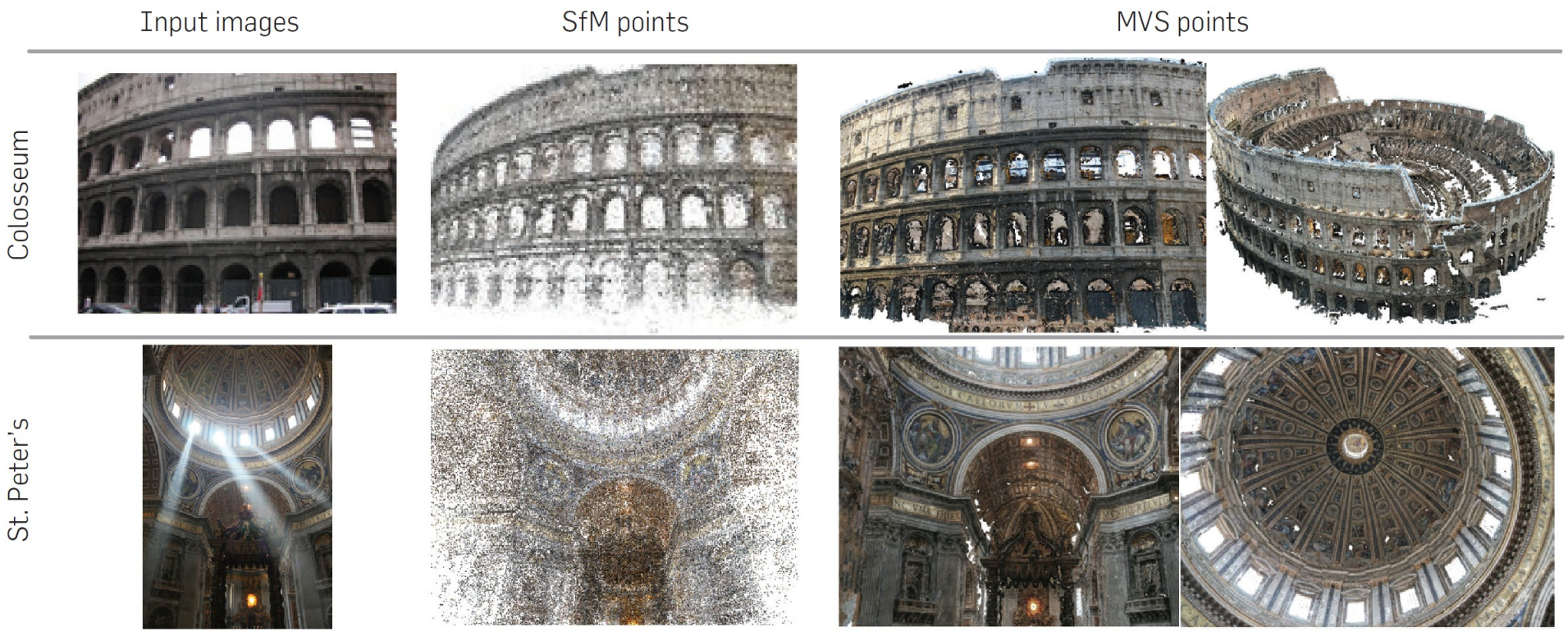
Ground truth

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

Summary of Challenges

- Low-contrast ‘textureless’ image regions
- Occlusions
- Violations of brightness constancy
  - Specular reflections
- Really large baselines
  - appearance change
- Camera calibration errors

SIFT + Fundamental Matrix + RANSAC + dense correspondence



Building Rome in a Day
By Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M. Seitz, Richard Szeliski
Communications of the ACM, Vol. 54 No. 10, Pages 105-112

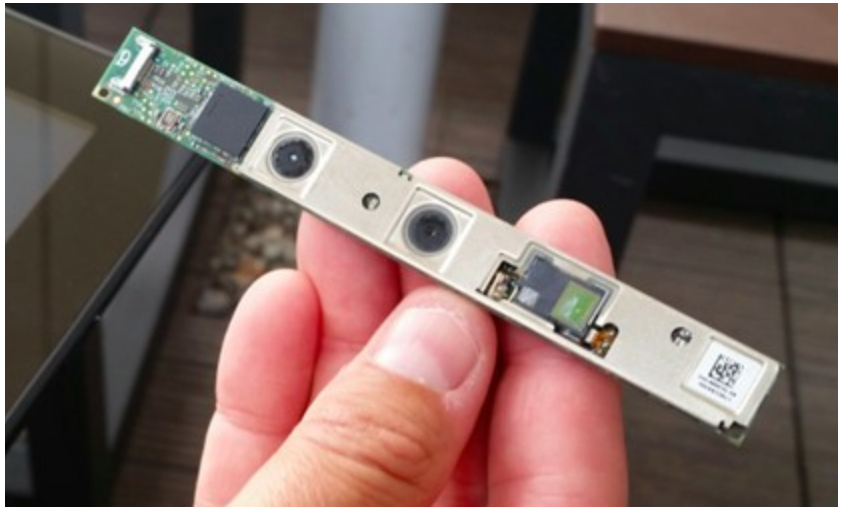
SIFT + Fundamental Matrix + RANSAC + dense correspondence

https://youtu.be/NdeD4cjLI0c

Depth Cameras

Many technologies:

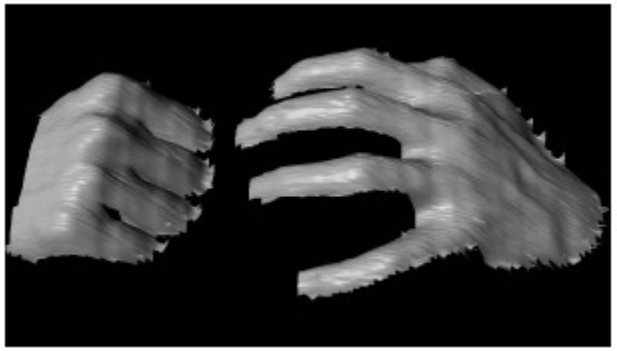
- Precalibrated stereo rig
- Structured light
- Infrared cameras
- Time of Flight
- combinations of the above
- etc.



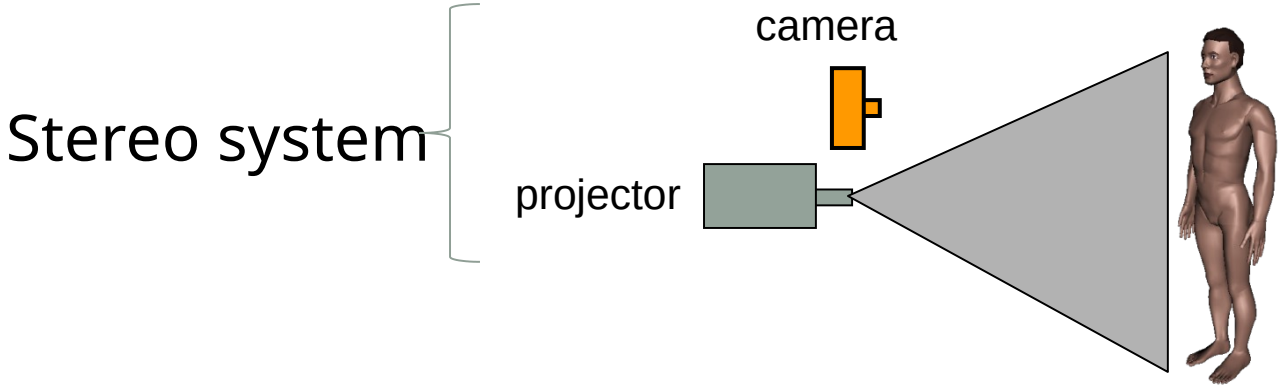
Intel laptop depth camera

Active stereo with structured light





- Project “structured” light patterns onto the object
  - Simplifies the correspondence problem
  - Allows us to use only one camera



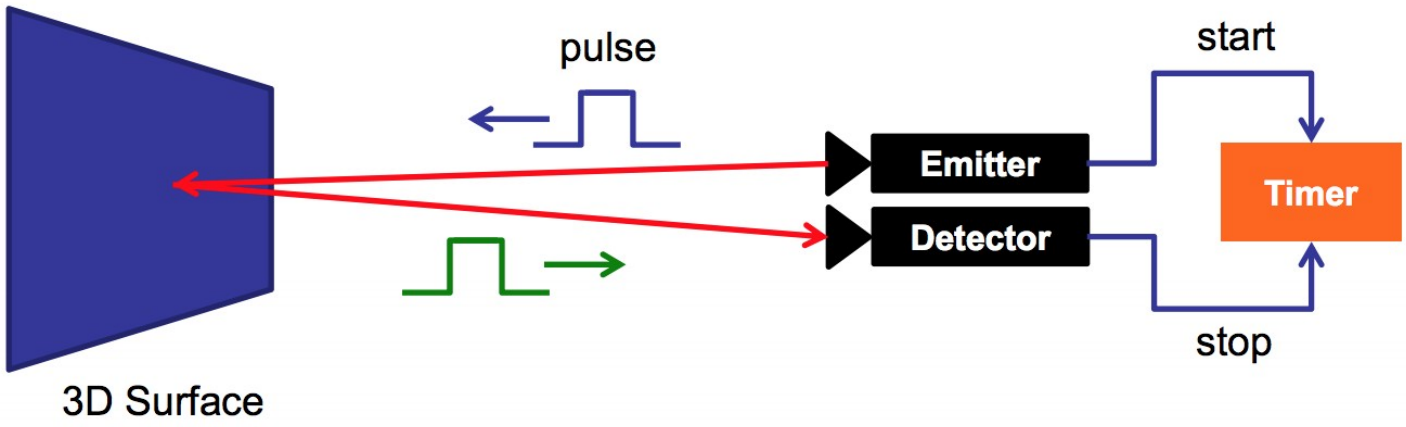
L. Zhang, B. Curless, and S. M. Seitz.
Rapid Shape Acquisition Using Color Structured Light and Multi-pass Dynamic Programming.
3DPVT 2002

Kinect: Structured infrared light

http://bbzippo.wordpress.com/2010/11/28/kinect-in-infrared/

Time of Flight (Kinect V2)

- Depth cameras in HoloLens use *time of flight*
  - Emit light of a known wavelength, and time how long it takes for it to come back

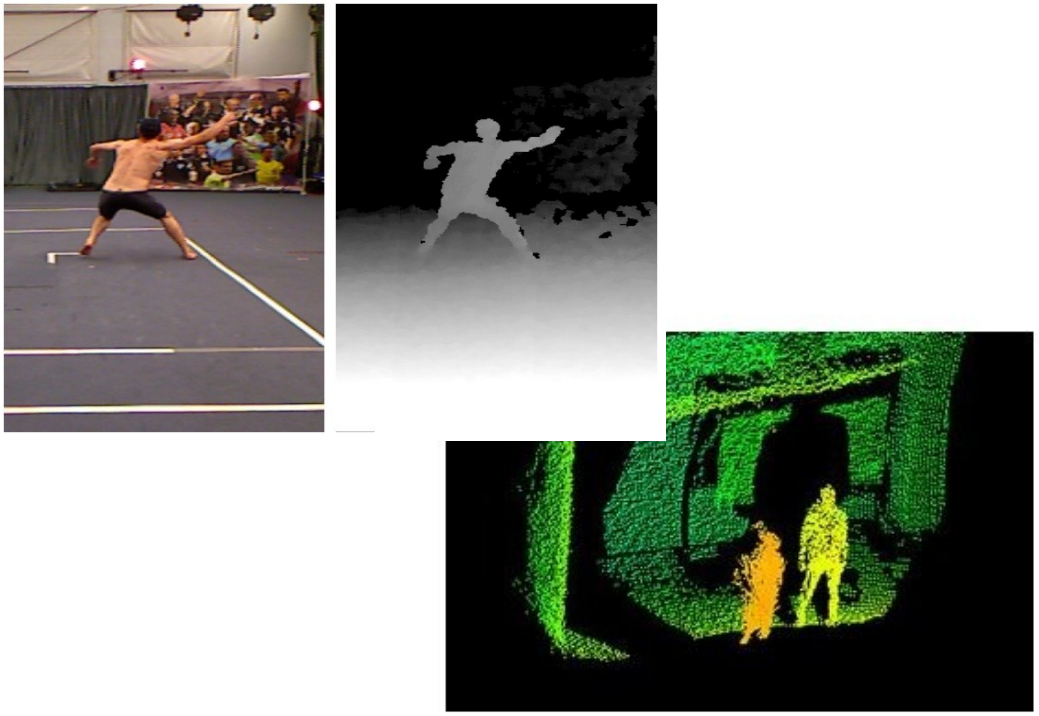


Integrating depth cameras

With the above methods we get a single “static” shot.

What can we do with them?

- make measurements.
- Integrate results: combine multiple scans together



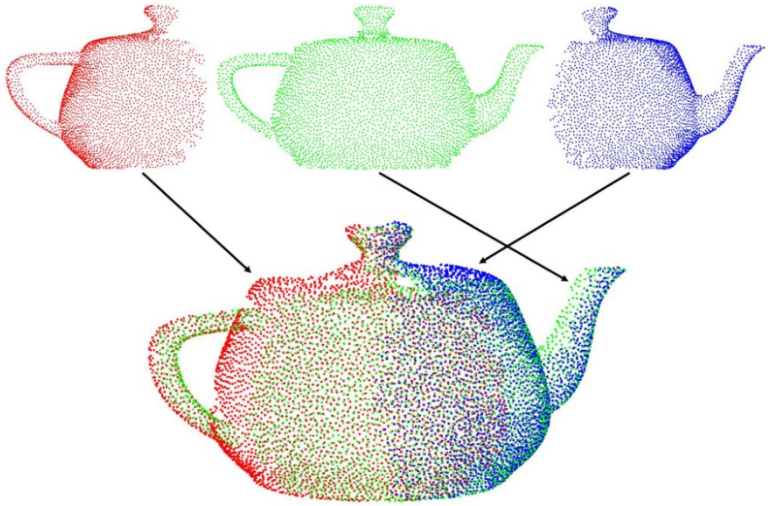
Optex Depth Camera Based on Canesta Solution

Integrating depth cameras

ICP: Iterative Closest Point.

- Problem: We don’t have correspondances in 3D like we did in 2D to find the transformations relating two independent views.

applications in Vision and Robotics: match point clouds



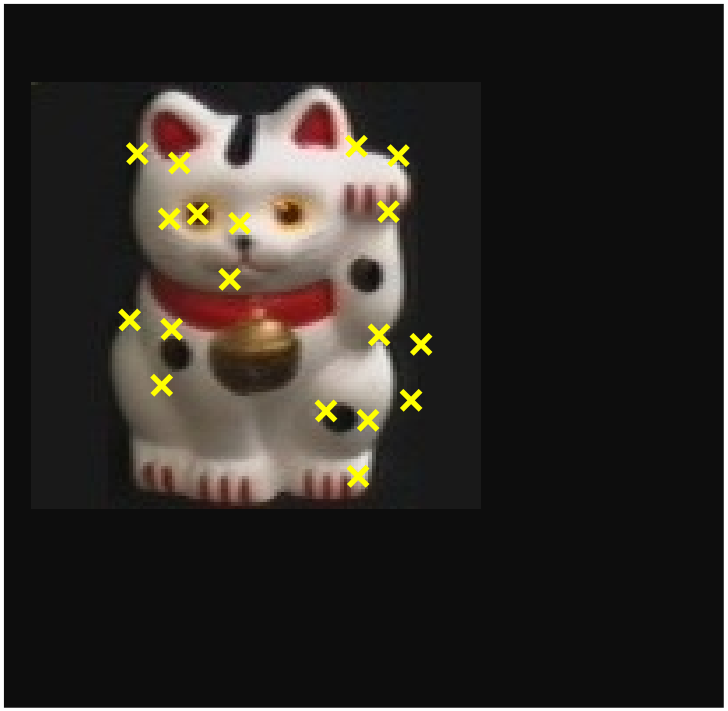
Iterative Closest Points (ICP) Algorithm

Goal:

Estimate transform between two dense point sets S_1 and S_2

1. **Initialize** transformation
   - Compute difference in mean positions, subtract
   - Compute difference in scales, normalize
2. **Assign** each point in S_1 to its nearest neighbor in S_2
3. **Estimate** transformation parameters T
   - Least squares or robust least squares, e.g., rigid transform
4. **Transform** the points in S_1 using estimated parameters T
5. **Repeat** steps 2-4 until change is very small (convergence)

Example: solving for translation



(t_x, t_y)

Problem: no initial guesses for correspondence

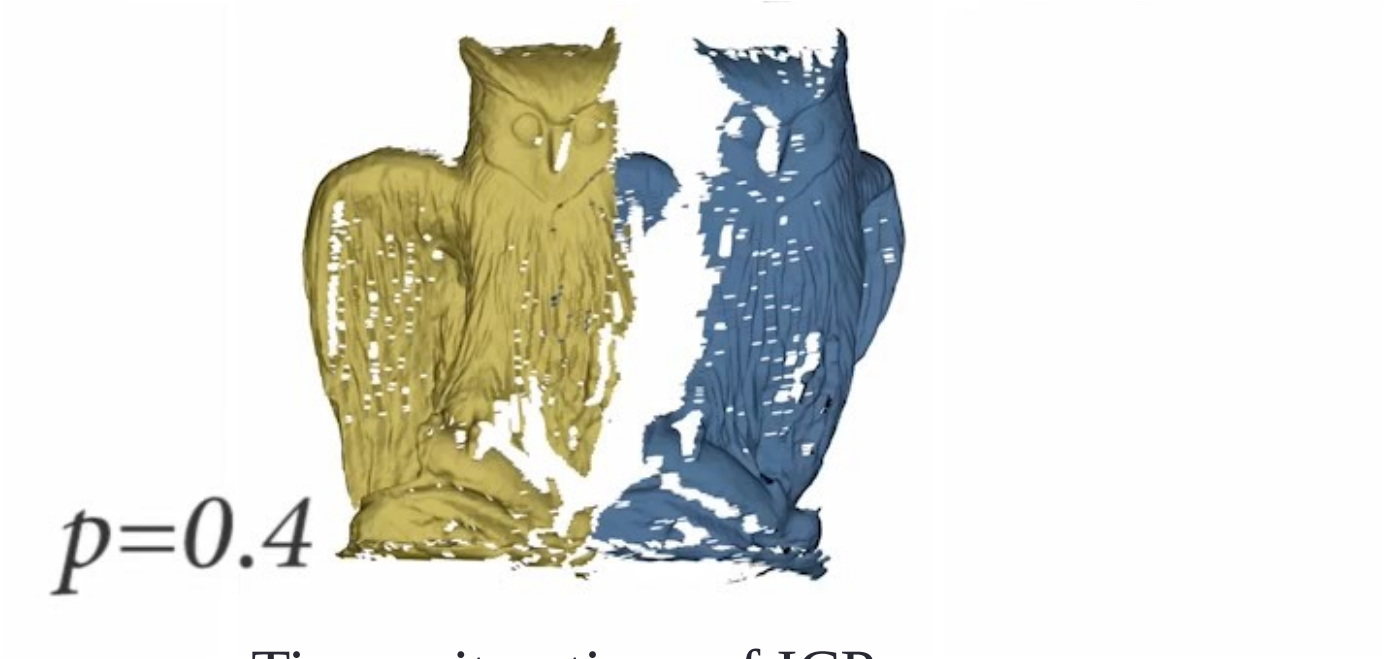
ICP solution

1. Initialize *t* by mean point translation
2. Find nearest neighbors for each point
3. Compute transform using matches
4. Move points using transform
5. Repeat steps 2-4 until convergence

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

ICP demonstration

https://www.youtube.com/watch?v=ii2vHBwlmo8



Time = iterations of ICP

Very useful and powerful library: https://pointclouds.org/

Bouaziz et al.

Image + Depth Camera + Variant of ICP

BundleFusion: Real-time Globally Consistent
3D Reconstruction using Online Surface Re-integration

Angela Dai[1] *Matthias Nießner*[1]
Michael Zollhöfer[2] *Shahram Izadi*[3]
Christian Theobalt[2]

[1]Stanford University
[2]Max Planck Institute for Informatics
[3]Microsoft Research

(*contains audio*)

https://www.youtube.com/watch?v=keIirXrRb1k