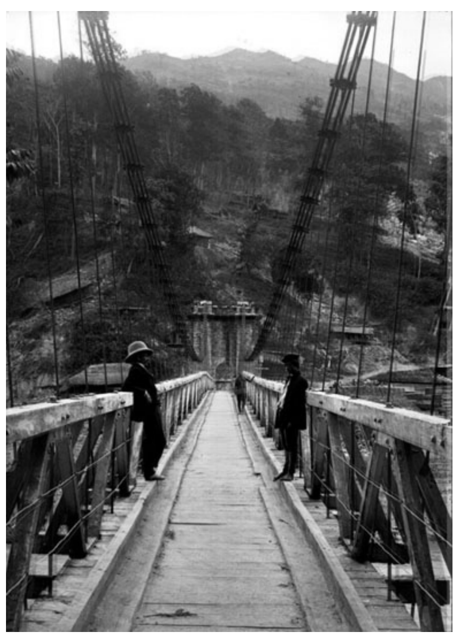
**Public Library, Stereoscopic Looking Room, Chicago, by Phillips, 1923**



**Teesta suspension bridge-Darjeeling, India**

Mark Twain at Pool Table", no date, UCR Museum of Photography

Woman getting eye exam during immigration procedure at Ellis Island, c. 1905 - 1920 , UCR Museum of Phography

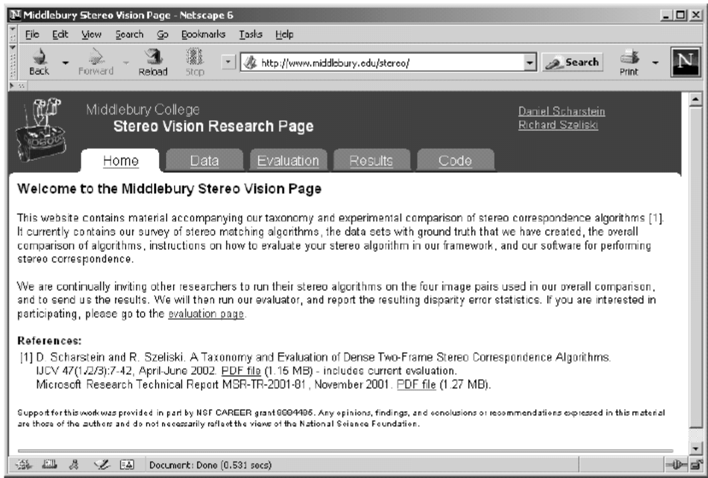
## Announcements

- Problem set 5 is on the class web page. (I made minor changes today).

## Information on Stereo

- Trucco and Verri Read Chapter 7 from the start through 7.3.2, also 7.3.7 and 7.4, 7.4.1. The rest is optional.
- Many slides taken from Octavia Camps and Steve Seitz

# Middlebury stereo page




http://www.middlebury.edu/stereo/

# Main Points

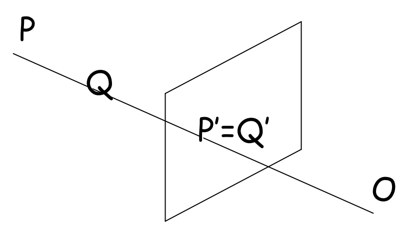
- Cameras with known position.
- Stereo allows depth by triangulation
- Two parts:
  - Finding corresponding points.
  - Computing depth (easy part).
- Constraints:
  - Geometry, epipolar constraint.
  - Photometric: Brightness constancy, only partly true.
  - Ordering: only partly true.
  - Smoothness of objects: only partly true.

# Matching

- Cost function:
  - What you compare: points, regions, features.
  - How you compare: eg., SSD, correlation.
- How you optimize.
  - Local greedy matches.
  - 1D search.
  - 2D search.

# Why Stereo Vision?

- 2D images project 3D points into 2D:



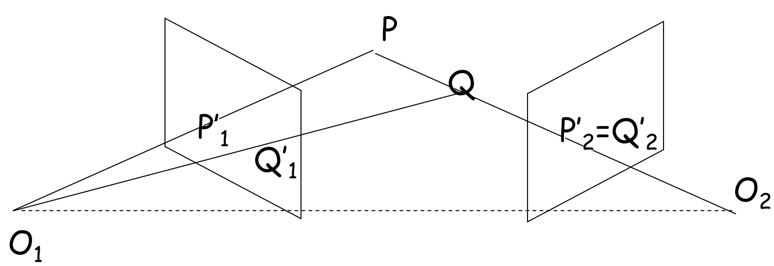

- 3D Points on the same viewing line have the same 2D image:
  - 2D imaging results in depth information loss

(Camps)

# Stereo

- Assumes (two) cameras.
- Known positions.
- Recover depth.

# Recovering Depth Information:




Depth can be recovered with two images and triangulation.

(Camps)

## So Stereo has two steps

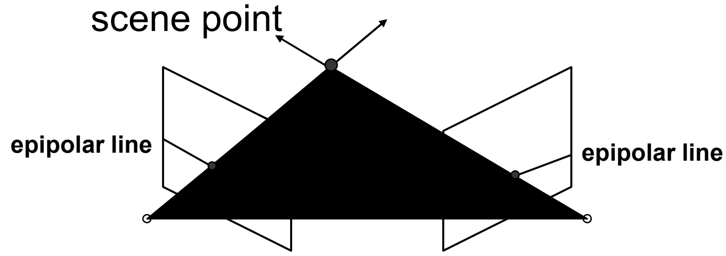
- Finding matching points in the images
- Then using them to compute depth.

## Epipolar Constraint

- Most powerful correspondence constraint.
- Simplifies discussion of depth recovery.

# 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*

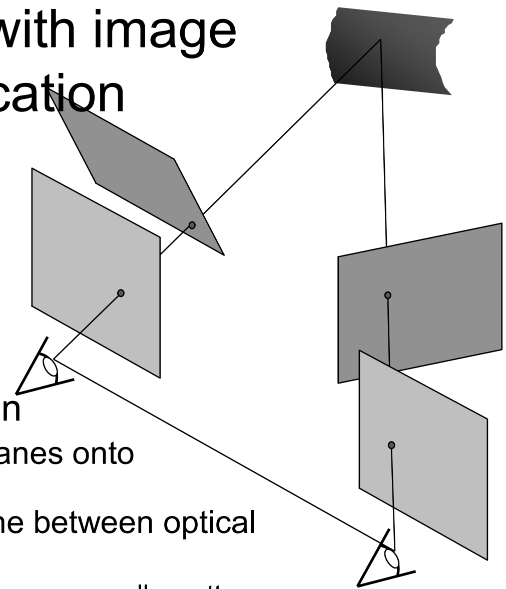
(Seitz)

# Simplest Case

- Image planes of cameras are parallel.
- Focal points are at same height.
- Focal lengths same.
- Then, epipolar lines are horizontal scan lines.

*blackboard*

# We can always achieve this geometry with image rectification

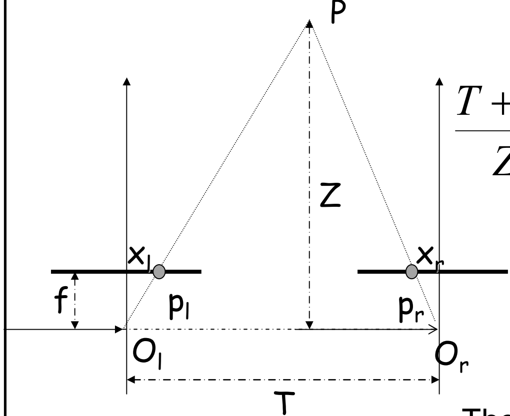


- Image Reprojection
  - reproject image planes onto common
    plane parallel to line between optical centers
- Notice, only focal point of camera really matters

(Seitz)

---

Let’s discuss reconstruction with this geometry before correspondence, because it’s much easier. *blackboard*




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

Disparity: $d = x_r - x_l$

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

Then given Z, we can compute X and Y.

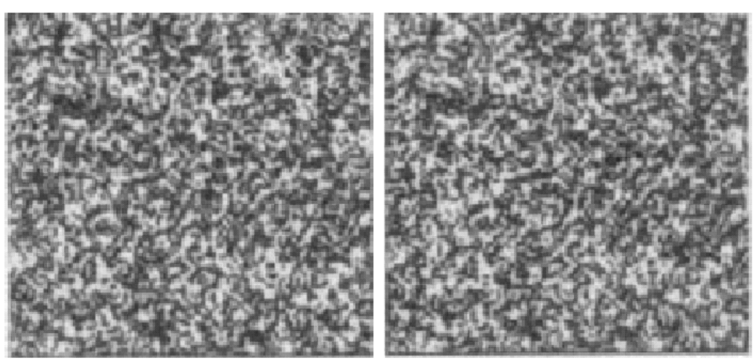
T is the stereo baseline
d measures the difference in retinal position between corresponding points

(Camps)

# Correspondence: What should we match?

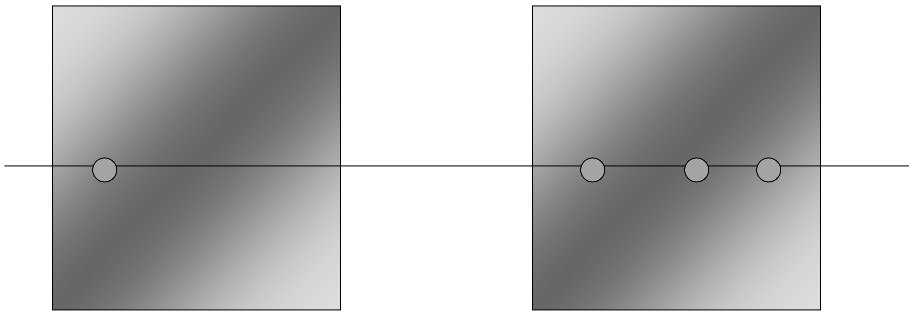
- Objects?
- Regions?
- Edges?
- Pixels?
- Collections of pixels?

# Random dot stereograms



Julesz: had huge impact because it showed that recognition not needed for stereo.

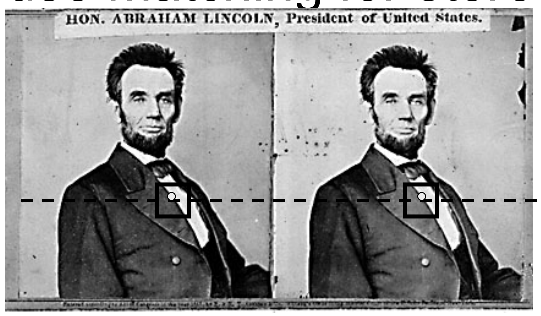
# Correspondence: Epipolar constraint.



# Correspondence: Photometric constraint

- Same world point has same intensity in both images.
  - Lambertian fronto-parallel
  - Issues:
    - Noise
    - Specularity
    - Foreshortening

## Using these constraints we can use matching for stereo




For each epipolar line

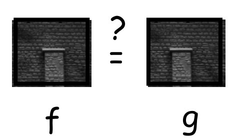
For each pixel in the left image

- compare with every pixel on same epipolar line in right image
- pick pixel with minimum match cost
- This will never work, so:

Improvement: match ***windows***

(Seitz)

## Comparing Windows:




$$SSD = \sum_{[i,j]\in R} (f(i,j) - g(i,j))^2$$

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

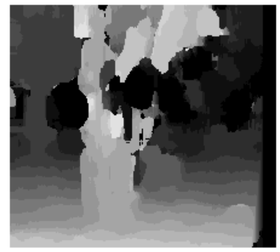
} Most popular

For each window, match to closest window on epipolar line in other image.

(Camps)

# Window size

W = 3



W = 20

- Effect of window size

Better results with *adaptive window*

- T. Kanade and M. Okutomi, *A Stereo Matching Algorithm with an Adaptive Window: Theory and Experiment*,, Proc. International Conference on Robotics and Automation, 1991.
- D. Scharstein and R. Szeliski. Stereo matching with nonlinear diffusion. International Journal of Computer Vision, 28(2):155-174, July 1998

(Seitz)

# Stereo results

– Data from University of Tsukuba

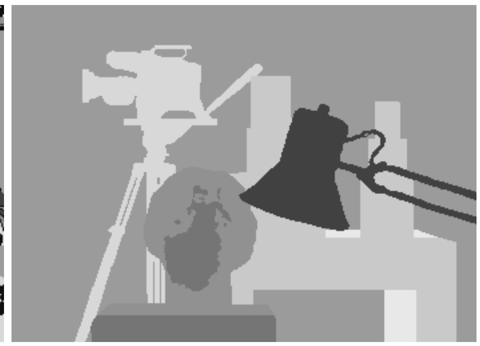
Scene

Ground truth

(Seitz)

## Results with window correlation

Window-based matching
(best window size)



Ground truth

(Seitz)

## Ordering constraint

- Usually, order of points in two images is same.
- *blackboard*

## Uniqueness

• One pixel cannot match more than one pixel.

## Occlusions

• This means some points must go unmatched

# This enables Dynamic Programming

- If we match pixel i in image 1 to pixel j in image 2, no matches that follow will affect which are the best preceding matches.
- *Example with pixels.*

First of all, we can represent a matching with a disparity map. Since disparity is non-negative, we'll use -1 to indicate an occlusion. So a 1D disparity map for the left image could be: [-1 1 1 -1 2 2 0]

This means the first pixel is occluded, the second has a disparity of 1, etc…. Notice that whenever there is an occlusion, the disparity will generally increase by one because we are advancing one pixel in the left image, without advancing in the right image (unless there's been an occlusion at the same time in the right image). When the disparity decreases, this means there's been an occlusion in the right image.

Next, given two images and a disparity map, we can assign a cost to this hypothesized matching. There are many ways to do this, but let's look at a simple example. When we match two pixels, the cost is the square of the difference in their intensities. For every occluded pixel, we assign a fixed cost. (In the problem set, we scale intensities to range from 0 to 1 and use an occlusion cost of .01.

*See Problem Set 7 for notes on how to find the disparity map with lowest cost using dynamic programming.*

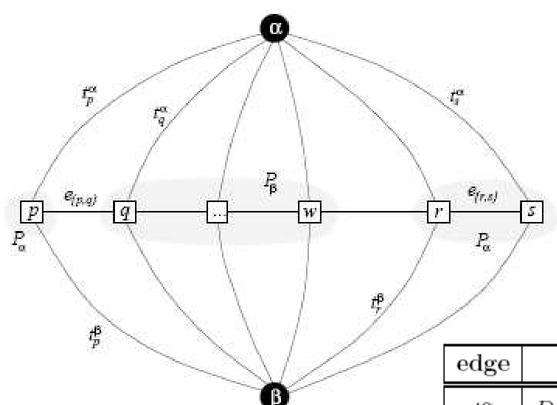
# Other constraints

- Smoothness: disparity usually doesn't change too quickly.
  - Unfortunately, this makes the problem 2D again.
  - Solved with a host of graph algorithms, Markov Random Fields, Belief Propagation, ….
- Occlusion and disparity are connected.

# Correspondence with MRF

- Every pixel is a site.
- Label of a pixel is its disparity.
- Disparity implies two pixels match. Prob. depends on similarity of pixels.
- Disparity at one pixel related to others since nearby pixels have similar disparities. Penalty based on different disparities at neighboring pixels.
- Finding best labeling is NP-hard, but good local algorithms exist.

# $\alpha - \beta$ swap

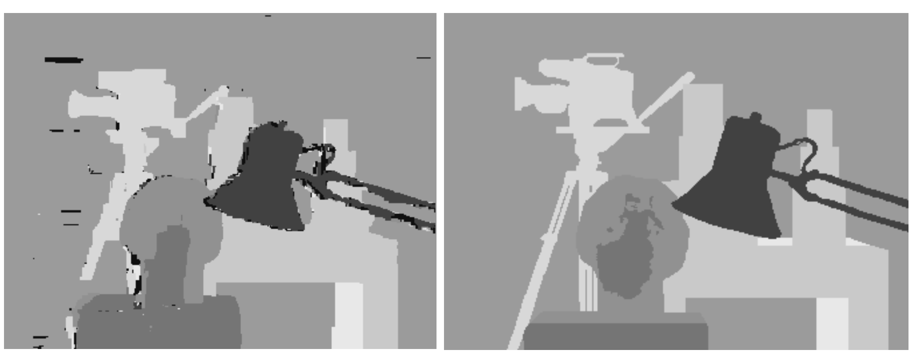



| edge | weight | for |
|---|---|---|
| $t_p^{\alpha}$ | $D_p(\alpha) + \sum_{q \in \mathcal{N}_p, q \notin \mathcal{P}_{\alpha\beta}} V(\alpha, f_q)$ | $p \in \mathcal{P}_{\alpha\beta}$ |
| $t_p^{\beta}$ | $D_p(\beta) + \sum_{q \in \mathcal{N}_p, q \notin \mathcal{P}_{\alpha\beta}} V(\beta, f_q)$ | $p \in \mathcal{P}_{\alpha\beta}$ |
| $e_{\{p,q\}}$ | $V(\alpha, \beta)$ | $\{p,q\} \in \mathcal{N}$, $p,q \in \mathcal{P}_{\alpha\beta}$ |

# Min-Cut gives best swap

- Min-cut
  - requires edge to one label be cut.
  - Cut between neighbors w/ diff. labels.
- Link to each label is cost of applying that label; cut means label is applied.
- Link between pixels = neighborhood cost (0 when same label).
- Keep performing swaps with all pairs of disparities until convergence.

# Results with graph cuts



Graph Cuts    Ground truth

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

(Seitz)

# Summary

- First, we understand constraints that make the problem solvable.
  - Some are hard, like epipolar constraint.
    - Ordering isn't a hard constraint, but most useful when treated like one.
  - Some are soft, like pixel intensities are similar, disparities usually change slowly.
- Then we find optimization method.
  - Which ones we can use depend on which constraints we pick.