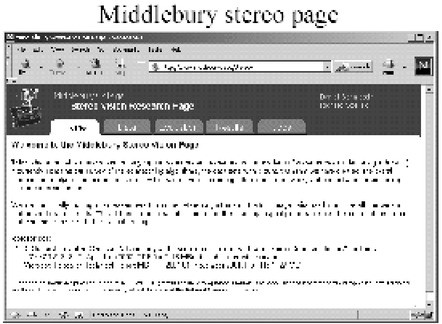
Announcements

- PS3 Due Thursday
- PS4 Available today, due 4/17.
- Quiz 2 4/24.

Information on Stereo

- Forsyth and Ponce Chapter 11
- For DP algorithm in lecture and problem set see: **``A Maximum Likelihood Stereo Algorithm'', by Cox, Hingorani, Rao, and Maggs, from the journal Computer Vision and Image Understanding, 63, 3, pp. 542-567.**
  - On Reserve in CS Library 3rd Floor AV Williams.
- Many slides taken from Octavia Camps and Steve Seitz

Middlebury stereo page



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

Main Points

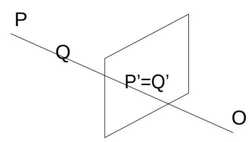
- Stereo allows depth by triangulation
- Two parts:
  - Finding corresponding points.
  - Computing depth (easy part).
- Constraints:
  - Geometry, epipolar constraint.
  - Photometric: Brightness constancy, only partly true.
  - Ordering: only partly true.
  - Smoothness of objects: only partly true.

Main Points (continued)

- Algorithms:
  - What you compare: points, regions, features.
- How you optimize.
  - Local greedy matches.
  - 1D search.
  - 2D search.

Why Stereo Vision?

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



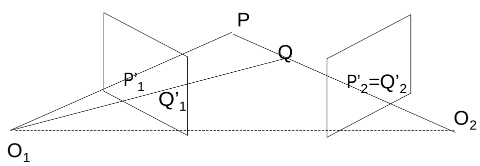
- 3D Points on the same viewing line have the same 2D image:
  - 2D imaging results in depth information loss

(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

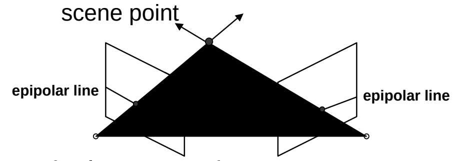
- Finding matching points in the images
- Then using them to compute depth.

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

Suppose image planes are in z = 1 plane.

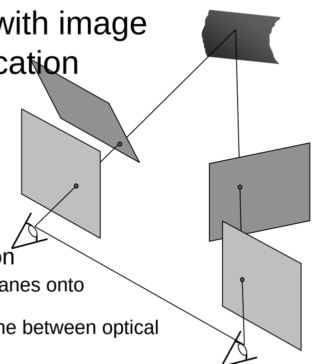
Focal points are on y = 0, z = 0 line.

Any plane containing focal points has form:

Ax + By + Cz + D = 0, with A = 0, D=0, since any point with y = 0 and z = 0 satisfies this equation.

So all planes through focal points have equation By + Cz = 0. If we look at where these intersect the image planes (z=1) it's at: By + C = 0. These are horizontal lines.

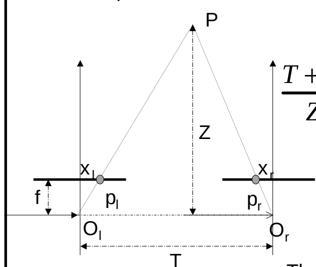
We can always achieve this geometry with image rectification



- 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*



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

Disparity: $d = x_l - x_r$

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

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)

Consider a simple example:

We have cameras with focal points at (-10,0,0) (0,0,0), focal lengths of 1 and image planes at the z=1 plane.

The world contains a 40x40 square in the z=100 plane, and it's lower left corner at (0,0,100).

The background is in the z=200 plane, with vertical stripes. For example, one stripe has sides x=-5, x=5, with z=200.

In the left image the square has corners at (.1,0), (.5,0), (.1, .4) , (.5, .4). In the right image, it's at (0,0), (.4,0), (0,.4), (.4,.4). The baseline is 10, the disparity is .1, so distance is 10/.1 = 100.

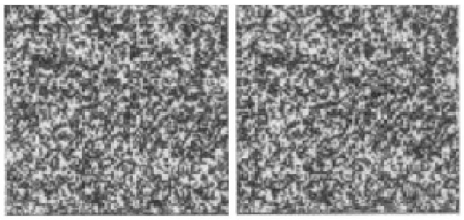
In the left image, the stripe is bounded by the lines x = .025, x = .05. In the right image, it's -.025, .025. So in the left image, the stripe is partly blocked by the square, in the right image it's fully to the left of the square. For the stripe, disparity is .05, so distance is 10/.05 = 200.

Notice that a line segment with ends at (-10,0,200), (0,0,100) projects in the left image to (0,0),(.1,0) and in the right to (-.05,0) (0,0). The line gets shorter in the right image due to foreshortening.

Correspondence: What should we match?

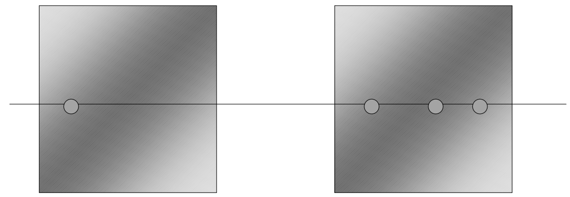
- Objects?
- Edges?
- Pixels?
- Collections of pixels?

Random dot stereograms



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

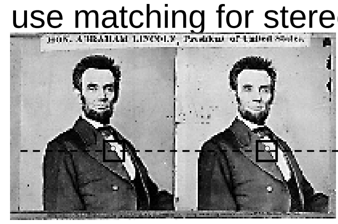
Correspondence: Epipolar constraint.



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

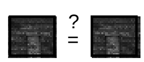
For each pixel in the left image

- 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:



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

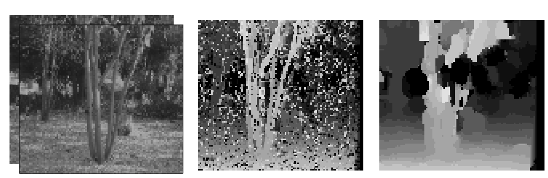
$$C_{fg} = \sum_{[i,j] \in R} f(i,j)g(i,j)$$

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*

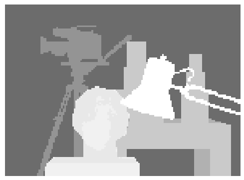
- T. Kanade and M. Okutomi, *A Stereo Matching Algorithm with an Adaptive Window: Theory and Experiment*,, Proc. International Conference on Robotics and Automation, 1991.
- D. Scharstein and R. Szeliski. Stereo matching with nonlinear diffusion. International Journal of Computer Vision, 28(2):155-174,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

(Seitz)

Stereo results

– Data from University of Tsukuba

Scene

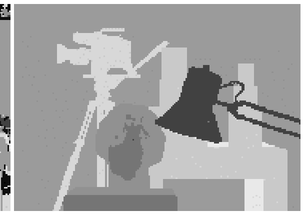


Ground truth

(Seitz)

Results with window correlation

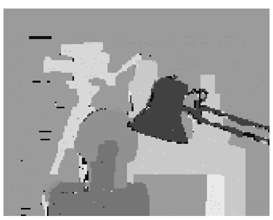
Window-based matching
(best window size)



Ground truth

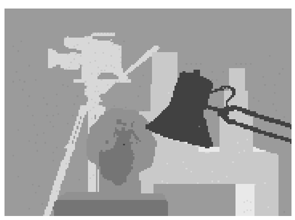
(Seitz)

Results with better method



State of the art method

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



Ground truth

(Seitz)

Ordering constraint

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

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 (a la Cox et al.).*
- How well does this work? See problem set.

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.

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.