Main Points

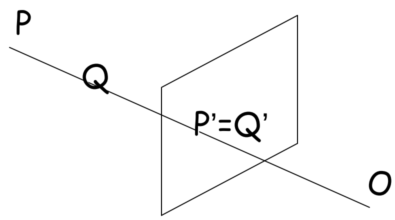
- Cameras with known position.
- Stereo allows depth by triangulation
- Two parts:
  - Finding corresponding points.
  - Computing depth (easy part).
- Constraints:
  - Geometry, epipolar constraint.
  - Photometric: Brightness constancy, only partly true.
  - Ordering: only partly true.
  - Smoothness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



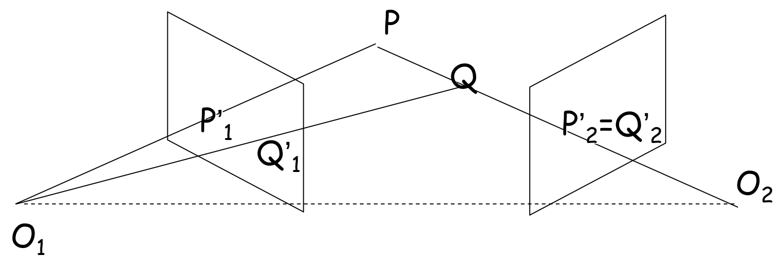
- 3D Points on the same viewing line have the same 2D image:
  - 2D imaging results in depth information loss

(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

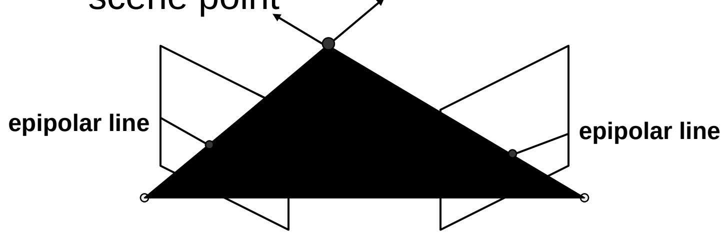
- Finding matching points in the images
- Then using them to compute depth.

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

Suppose image planes are in z = 1 plane.

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

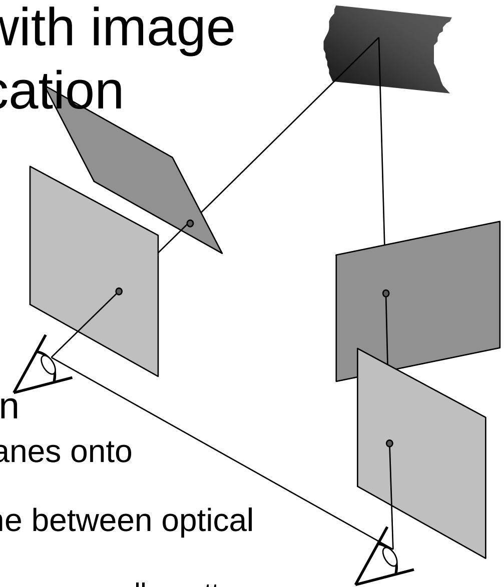
Any plane containing focal points has form:

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

Specifically, we could say focal points are (0,0,0) (10,0,0). Then (0,0,0)*(A,B,C) + D = 0, so D = 0. (10,0,0)*(A,B,C) + D = 10A = 0 so A = 0.

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.

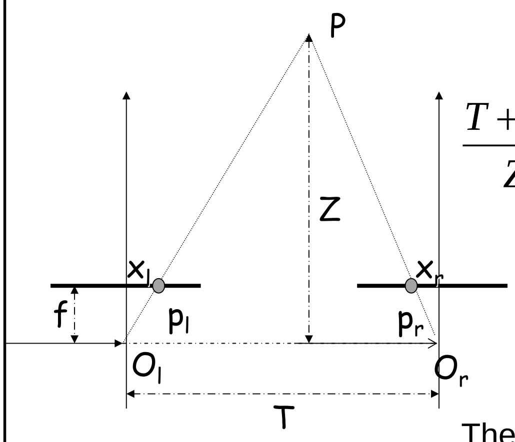
We can always achieve this geometry with image rectification



- 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*



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

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

Disparity: $d = x_l - x_r$

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

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.

Some stereo problems:

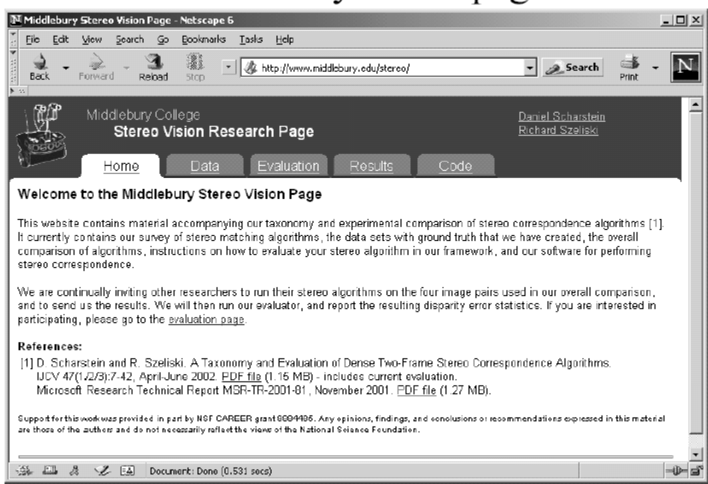
1)Suppose we have two cameras that are side-by-side. The left camera sees a point at (1,2). Give an equation for the line that contains the point. Give an equation for a projection of this line into the other image, ie., the epipolar line.

2)Suppose we have a camera with focal point of (0,0,0) and image plane of x+z = 1. We want to rectify this so that the image plane is z = 1. Give equations for this.

3)Suppose we have an object that is moving towards us in a straight line of x=1, y = 1 at constant speed. Give an equation for the curve it traces in the image.

4)Are the epipolar lines always parallel? Prove this or give an example where they are not.

Middlebury stereo page

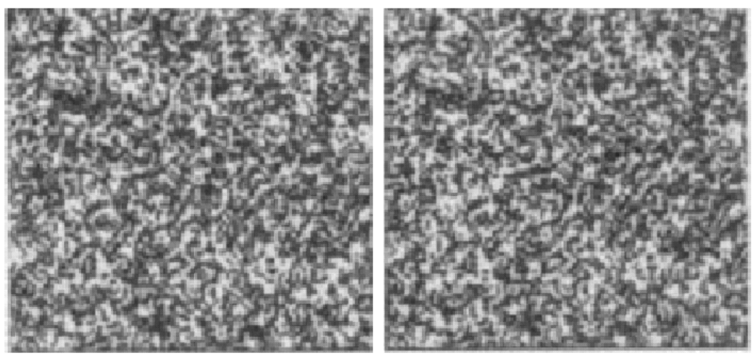


http://www.middlebury.edu/stereo/

Correspondence: What should we match?

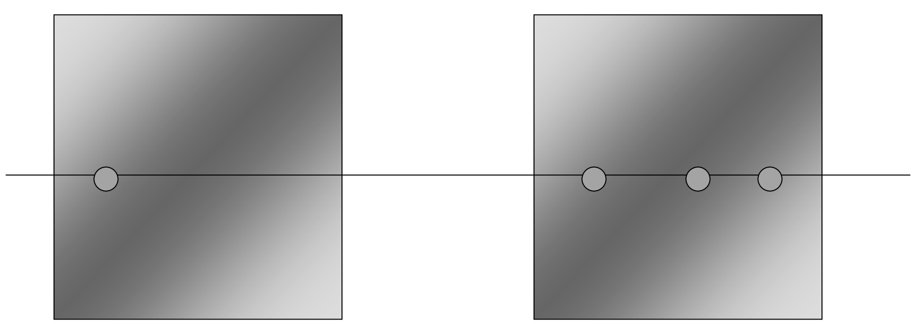
- Objects?
- Regions?
- Edges?
- Pixels?
- Collections of pixels?

Random dot stereograms



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

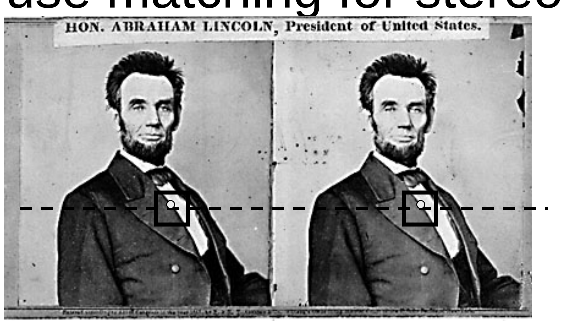
Correspondence: Epipolar constraint.



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:  ? = 

f g

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

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

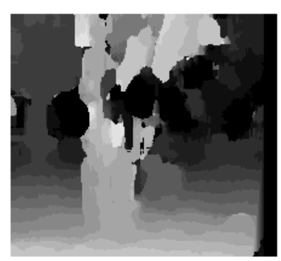
} Most popular

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

(Camps)

Window size

W = 3



W = 20

- Effect of window size

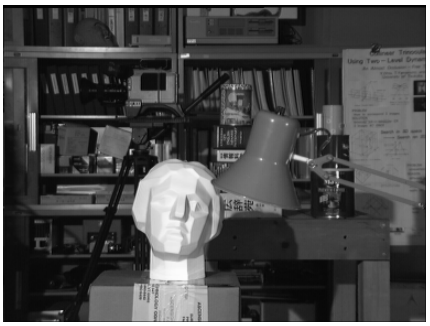
Better results with *adaptive window*

- 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

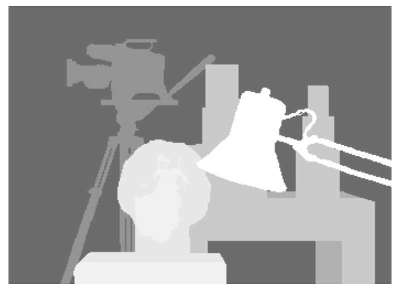
(Seitz)

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.

We can match the images one row at a time.

We will assume that we can do two possible things. One is to match two pixels, the second is to allow a pixel to go unmatched. We will create a graph in which nodes represent choices about matching, and edges represent the cost of matching. We will name a node in a way that indicates which pixels have been matched so far. For example, if we reach node N(3,5) this will mean that the first three pixels in image 1, and the first 5 pixels in image 2, have all been taken care of. From N(3,5) we can go to N(4,6). This must mean that we take care of both nodes 4 and 6 in one step, by matching them together. Or we can go to node N(3,6). This means that node 6 in the second image is taken care of by not matching it to anything. That is, node 6 in the second image is occluded. Likewise, we could go to N(4,5).

We need a special start node, S. This is connected to N(0,1), N(1,0) and N(1,1). We need a special end node, E. For example, if there are 9 pixels in each image, E will be connected to N(8,8), N(8,9), N(9,8).

Finally, we use edge weights to encode the cost of these choices.

E(N(i-1,j-1), N(i,j)) = (I1(i)-I2(j))^2.

E(N(i-1,j),N(i,j)) = OCCLUSION_PENALTY

Now when we take a path from S to E we are going through nodes that represent a matching of the images. The cost of the path is the cost of the matching.

Why do we need the ordering constraint to use this?

Suppose our images have the following two rows:
0 1 1 1 0 0 1 1 0
0 1 1 0 0 1 1 0 0
Suppose the occlusion cost is .01. The path from S to N(1,1) will have a cost of 0. From S to N(1,0) will have a cost of .01.
Notice that there will be two shortest paths. Either the second or the fourth pixel in the first image may be occluded.

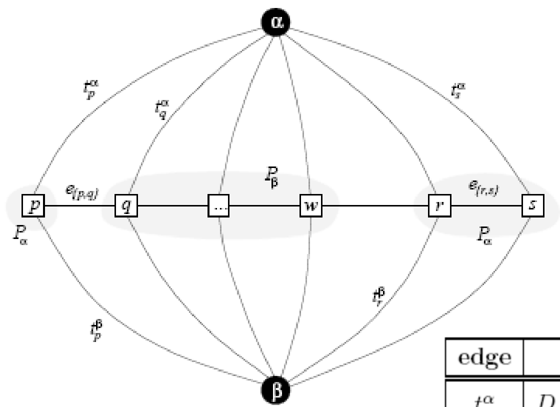
Other constraints

- 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.

α–β swap

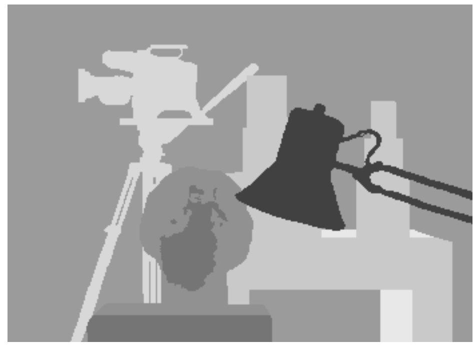


edge	weight	for
t_p^α	$D_p(\alpha) + \sum_{q\in\mathcal{N}_p,\ q\notin\mathcal{P}_{\alpha\beta}} V(\alpha, f_q)$	$p \in \mathcal{P}_{\alpha\beta}$
t_p^β	$D_p(\beta) + \sum_{q\in\mathcal{N}_p,\ q\notin\mathcal{P}_{\alpha\beta}} V(\beta, f_q)$	$p \in \mathcal{P}_{\alpha\beta}$
$e_{\{p,q\}}$	$V(\alpha,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

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.