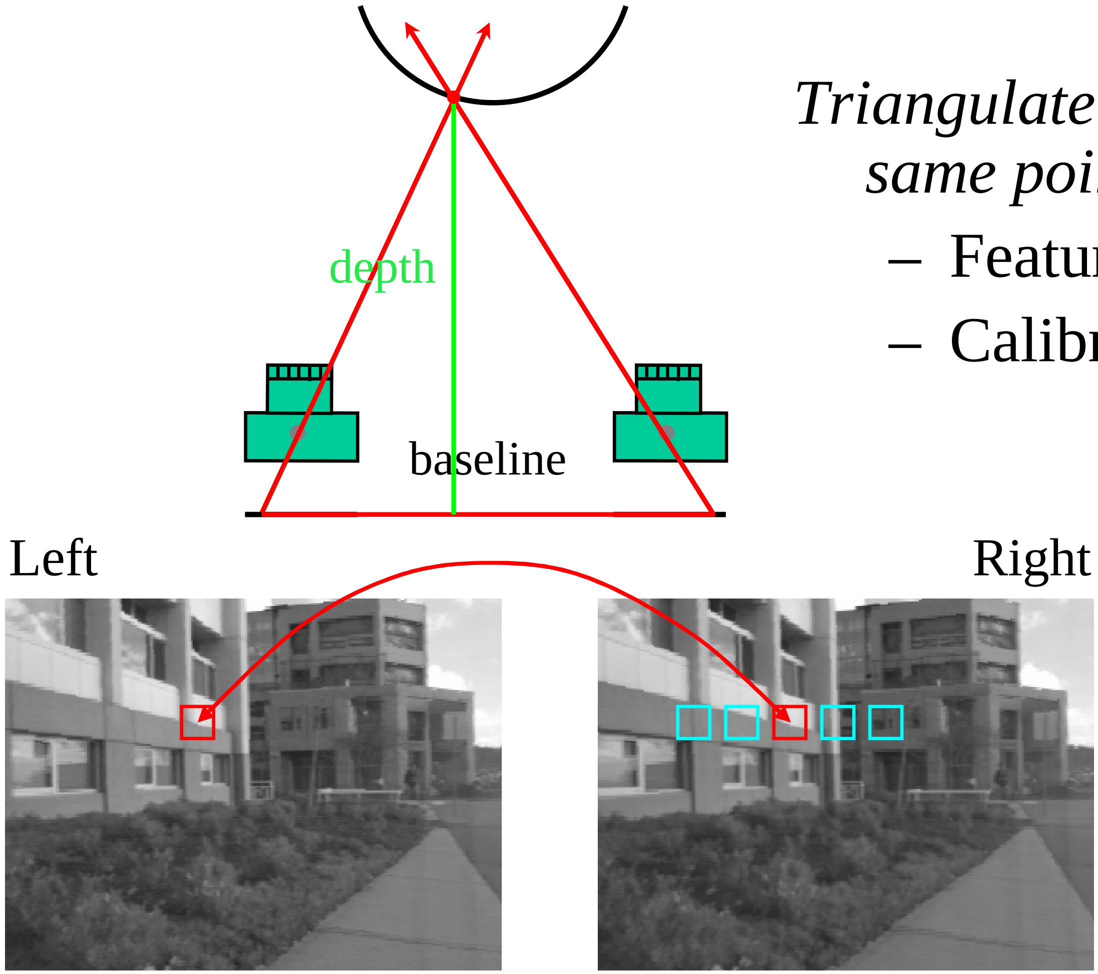
Stereo Vision
Reading: Chapter 11

- Stereo matching computes depth from two or more images
- **Subproblems:**
  - Calibrating camera positions.
  - Finding all corresponding points (hardest part)
  - Computing depth or surfaces.

Slide credits for this chapter: David Jacobs, Frank Dellaert, Octavia Camps, Steve Seitz

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

Stereo vision

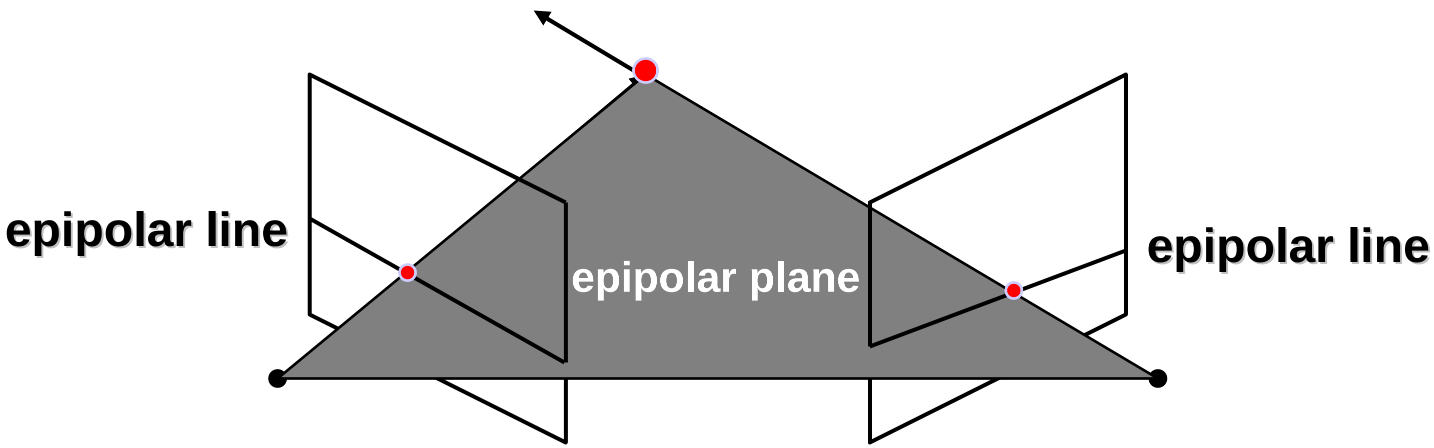


Triangulate on two images of the same point to recover depth.

- Feature matching across views
- Calibrated cameras

Matching correlation windows across scan lines

The epipolar constraint



- Epipolar Constraint
  - Matching points lie along corresponding epipolar lines
  - Reduces correspondence problem to 1D search along *conjugate epipolar lines*
  - Greatly reduces cost and ambiguity of matching

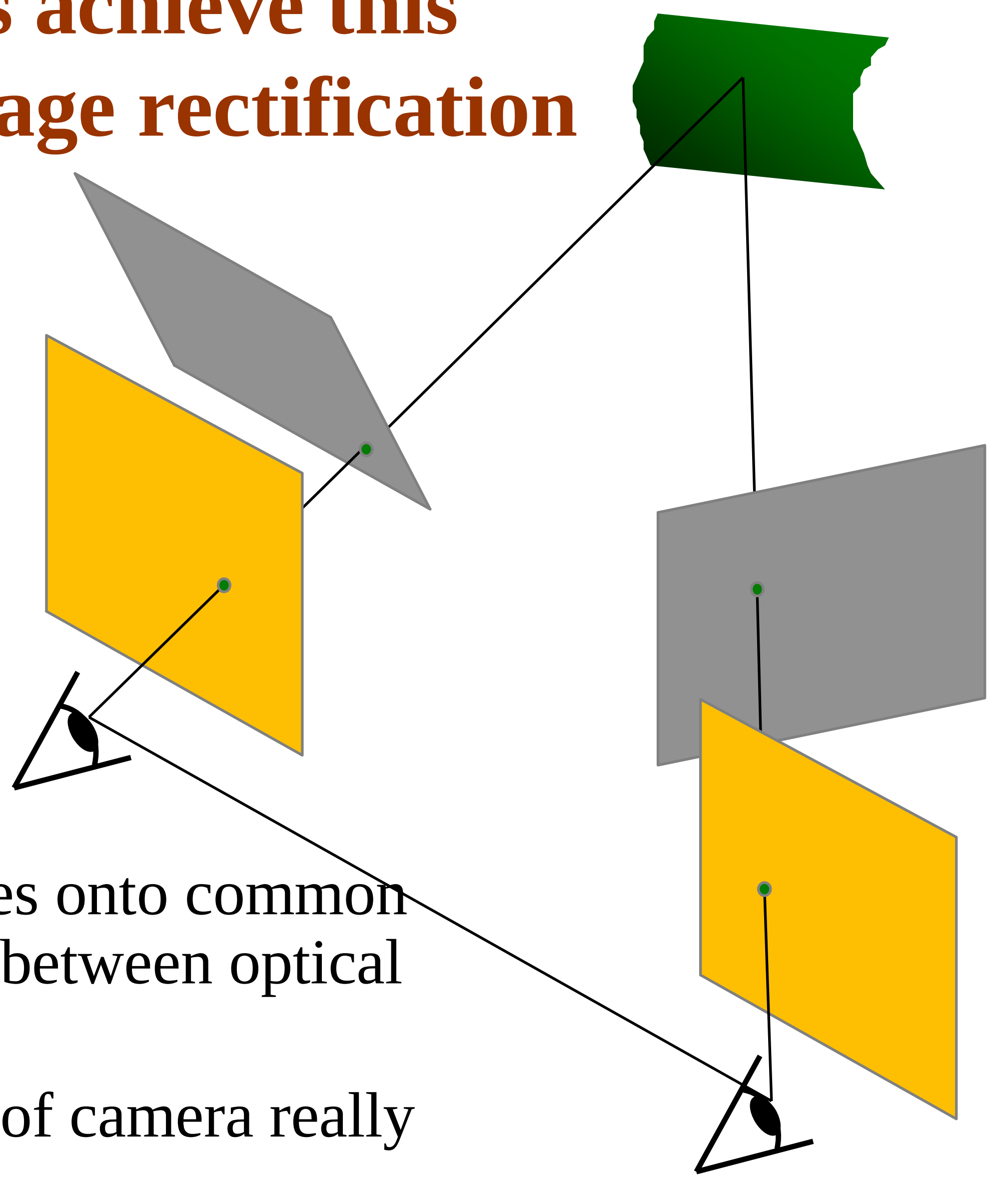
Slide credit: Steve Seitz

Simplest Case: Rectified Images

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

- We will assume images have been *rectified* so that epipolar lines correspond to scan lines
  - Simplifies algorithms
  - Improves efficiency

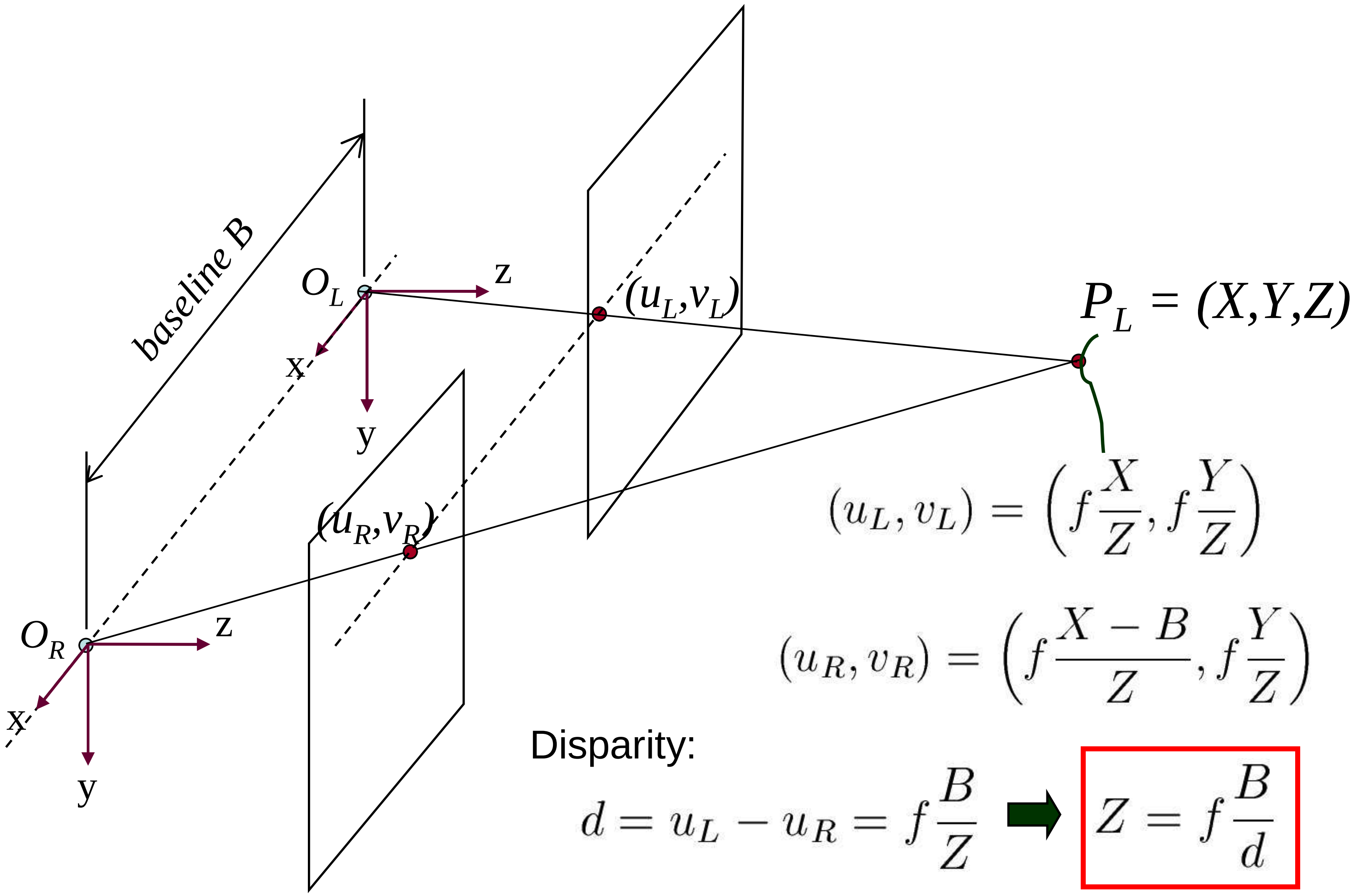
We can always achieve this geometry with image rectification



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

(Seitz)

Basic Stereo Derivations



$$(u_L, v_L) = \left(f\frac{X}{Z}, f\frac{Y}{Z}\right)$$

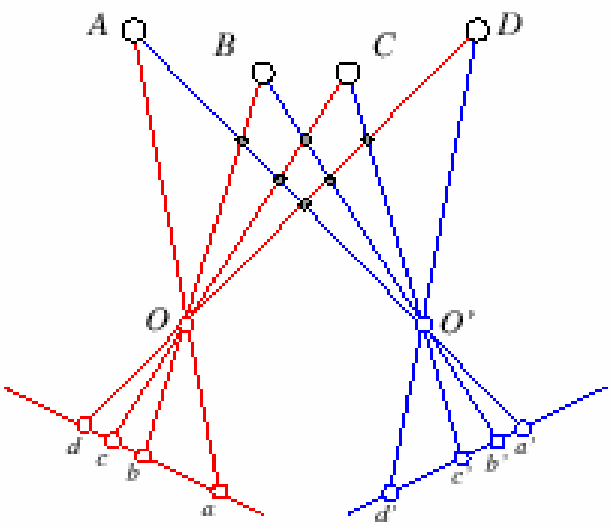
$$(u_R, v_R) = \left(f\frac{X-B}{Z}, f\frac{Y}{Z}\right)$$

Disparity:

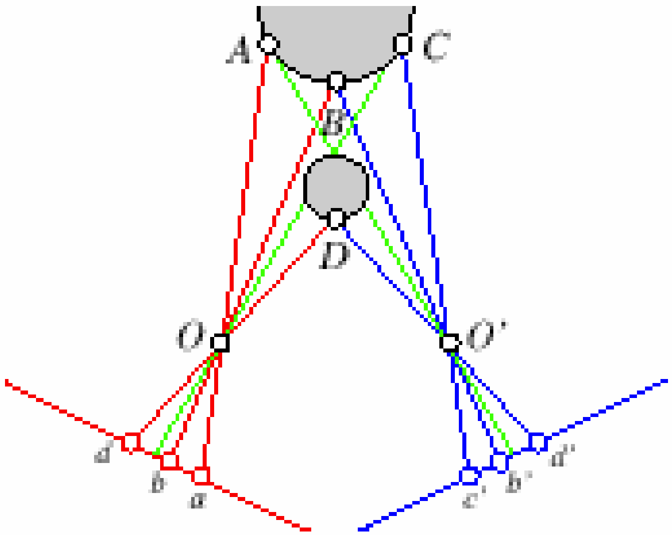
$$d = u_L - u_R = f\frac{B}{Z} \Rightarrow \boxed{Z = f\frac{B}{d}}$$

Correspondence

- It is fundamentally ambiguous, even with stereo constraints



Ordering constraint…

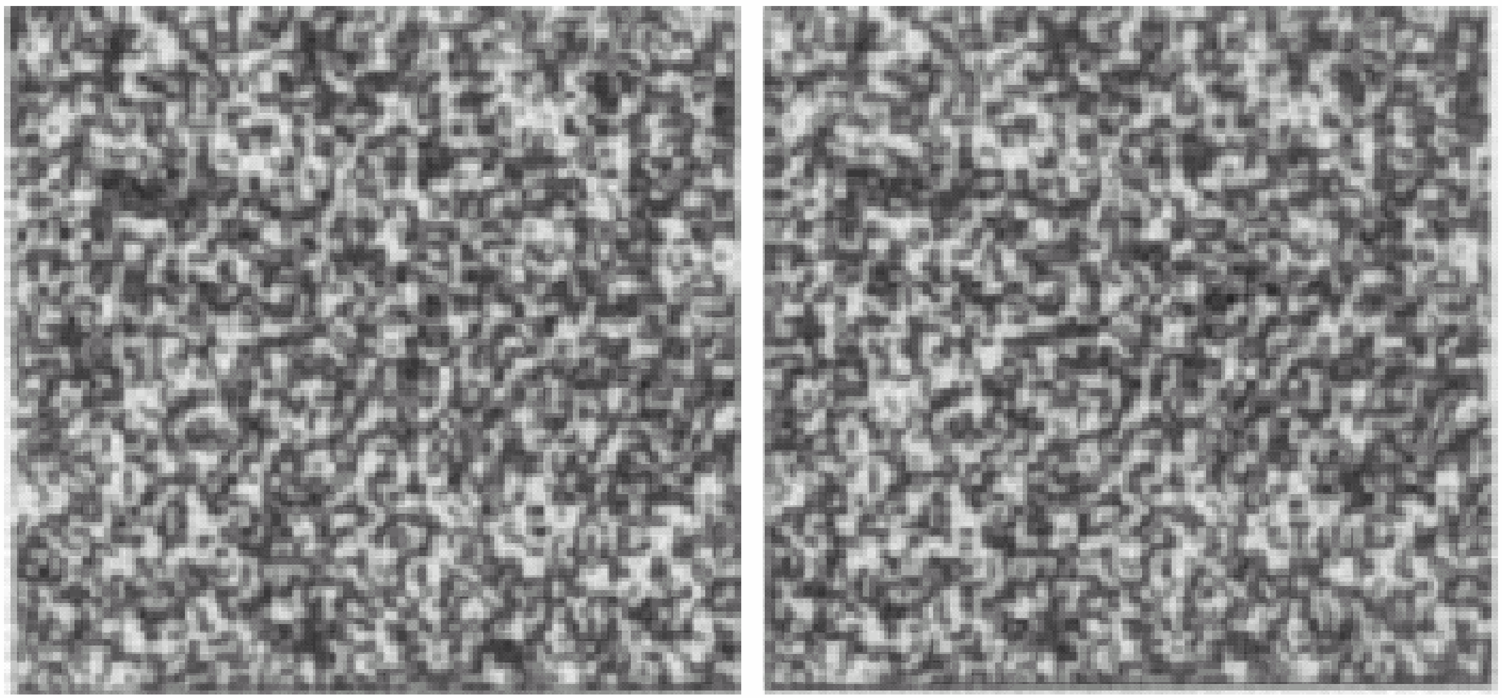


…and its failure

Correspondence: What should we match?

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

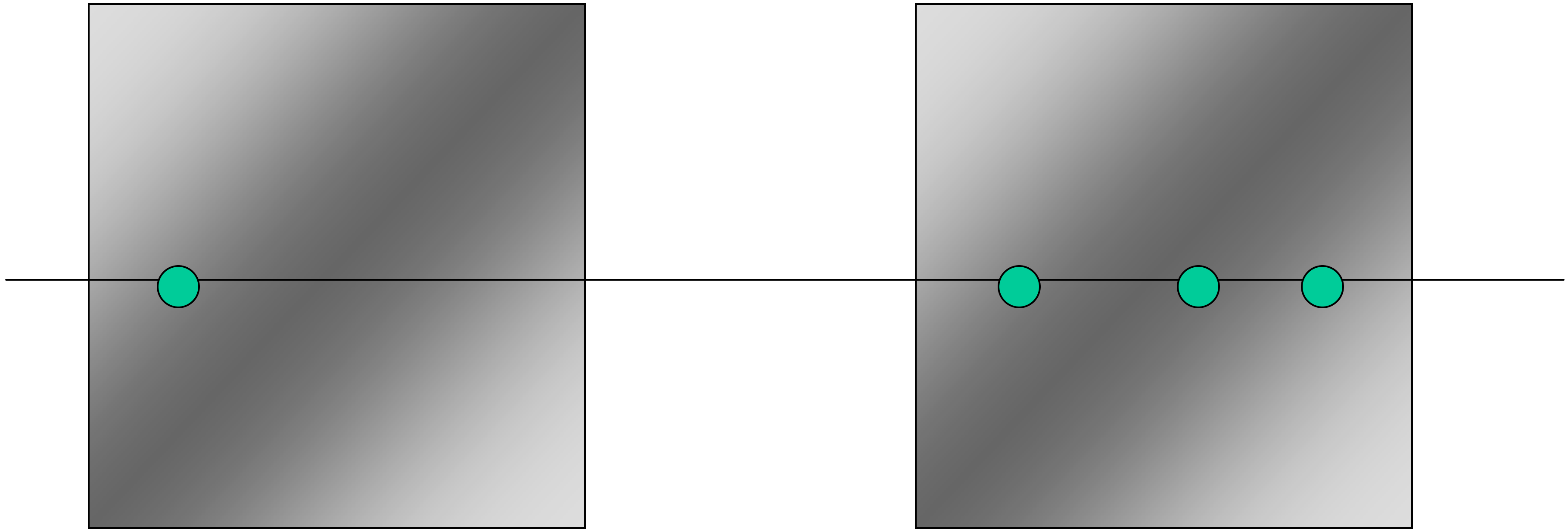
Random dot stereograms



Julesz: showed that recognition is not needed for stereo.

Correspondence: Epipolar constraint.

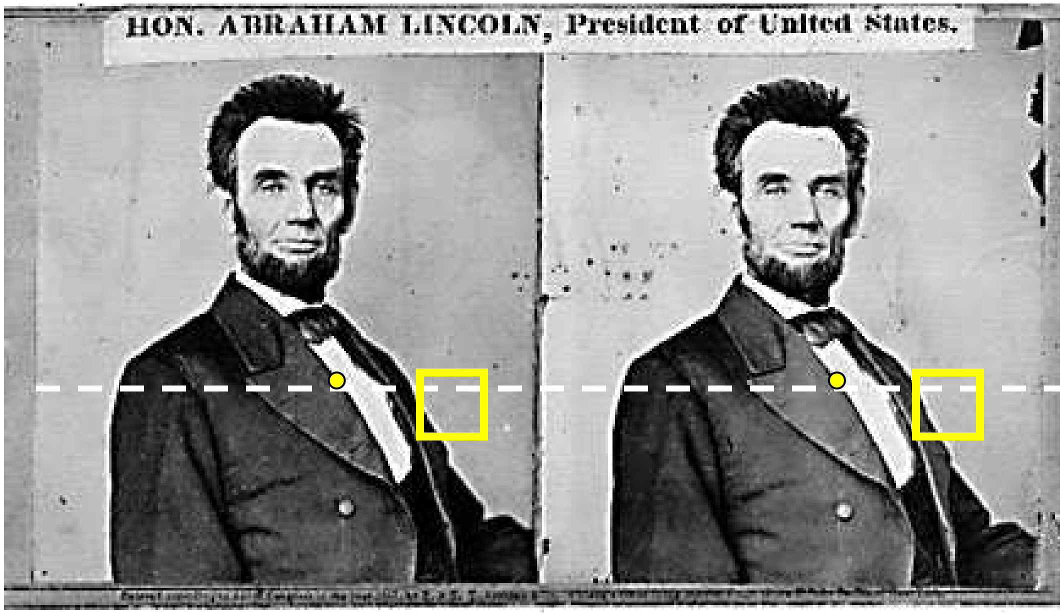
The epipolar constraint helps, but much ambiguity remains.



Correspondence: Photometric constraint

- Same world point has same intensity in both images.
  - True for Lambertian surfaces
    - A Lambertian surface has a brightness that is independent of viewing angle
  - Violations:
    - Noise
    - Specularity
    - Non-Lambertian materials
    - Pixels that contain multiple surfaces

Pixel matching



For each epipolar line

For each pixel in the left image

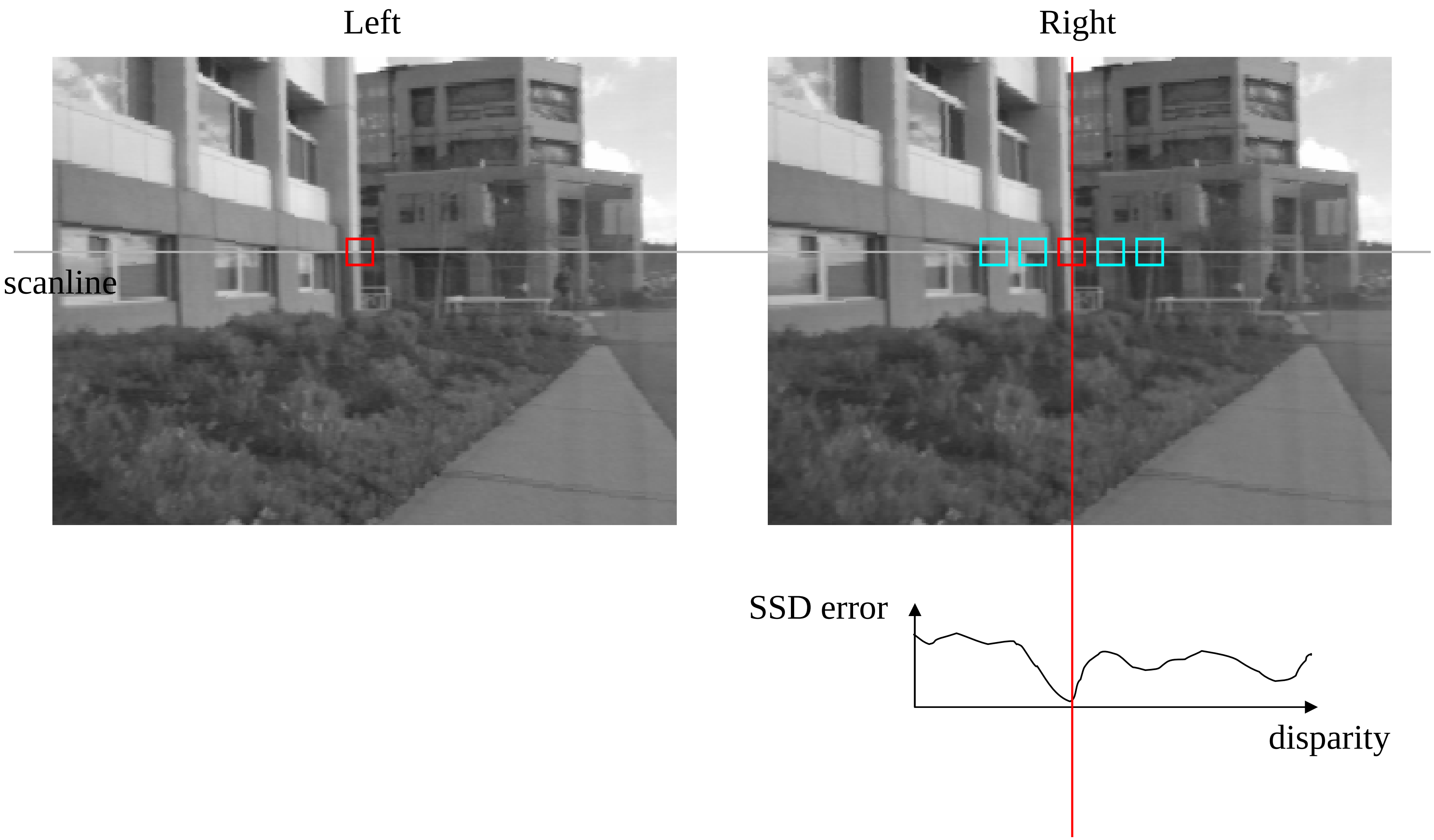
- compare with every pixel on same epipolar line in right image
- pick pixel with minimum match cost

This leaves too much ambiguity, so:

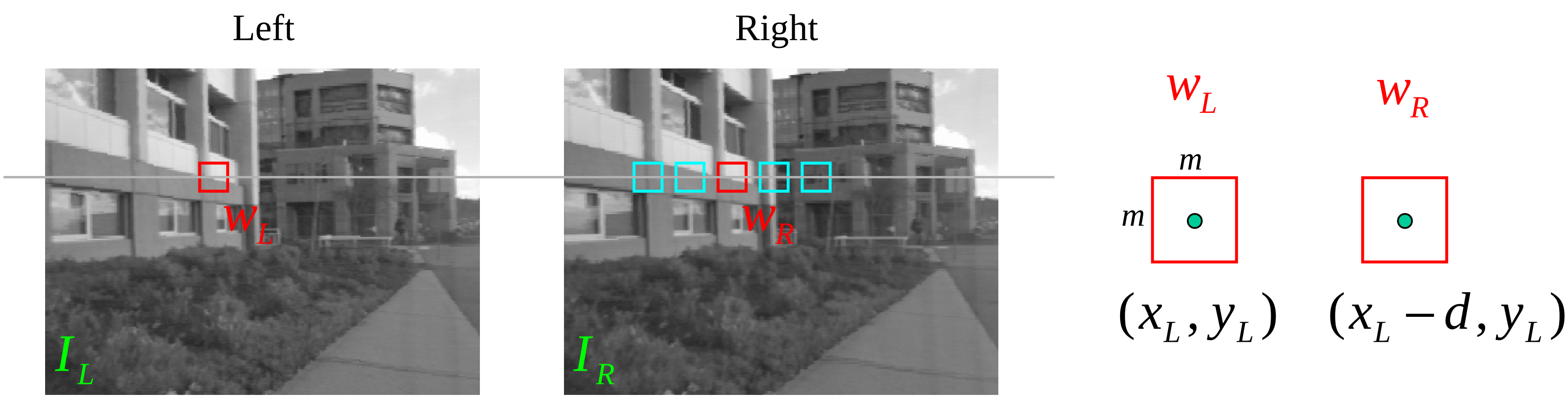
Improvement: match ***windows***

(Seitz)

Correspondence Using Correlation



Sum of Squared (Pixel) Differences



w_L and w_R are corresponding m by m windows of pixels.

We define the window function :

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

The SSD cost measures the intensity difference as a function of disparity :

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

Image Normalization

- Even when the cameras are identical models, there can be differences in gain and sensitivity.
- For these reason and more, it is a good idea to normalize the pixels in each window:

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

Average pixel

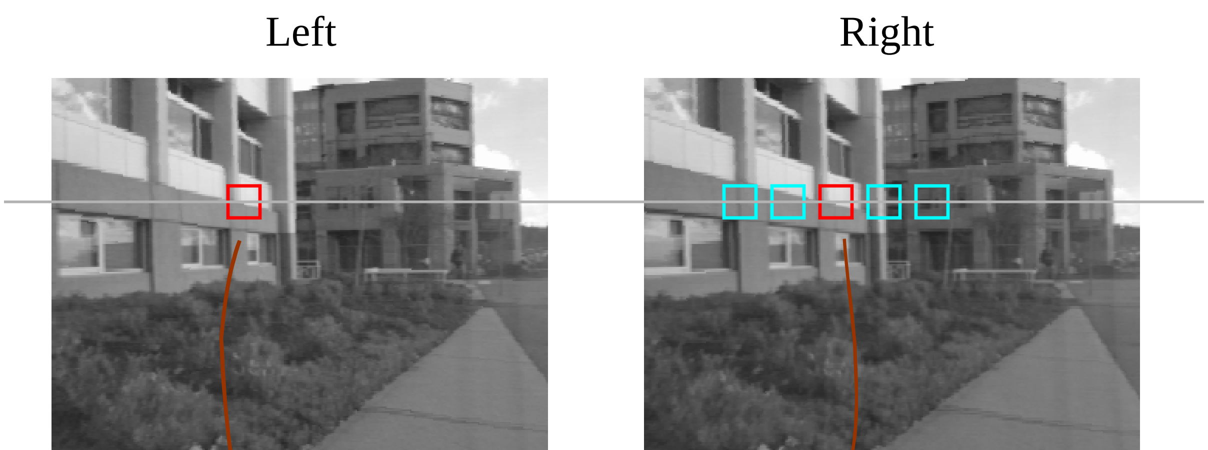
$$\|I\|_{W_m(x,y)} = \sqrt{\sum_{(u,v)\in W_m(x,y)} [I(u,v)]^2}$$

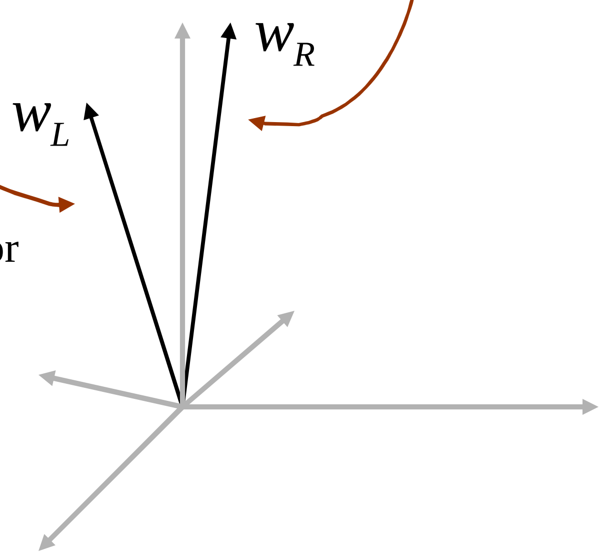
Window magnitude

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

Normalized pixel

Images as Vectors





Each window is a vector in an m^2 dimensional vector space. Normalization makes them unit length.

"Unwrap" image to form vector, using raster scan order

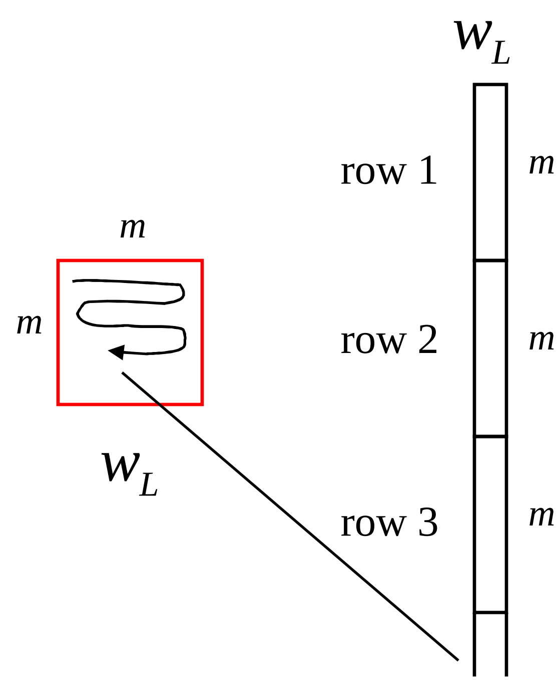
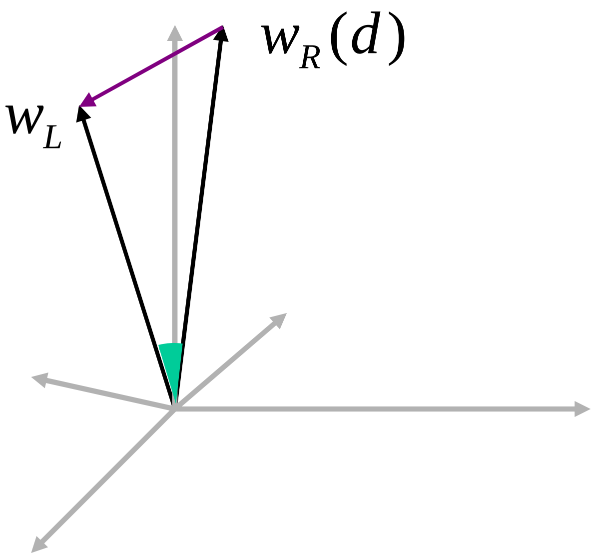


Image Metrics



(Normalized) Sum of Squared Differences

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

$$= \left\| w_L - w_R(d) \right\|^2$$

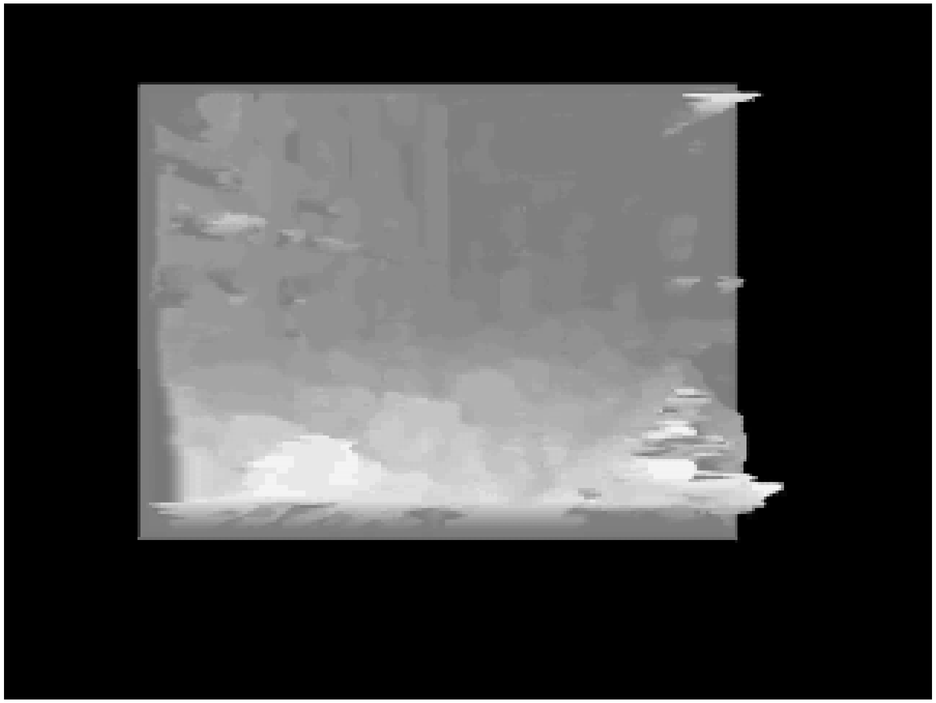
Normalized Correlation

$$C_{\text{NC}}(d) = \sum_{(u,v)\in W_m(x,y)} \hat{I}_L(u,v)\hat{I}_R(u-d,v)$$

$$= w_L \cdot w_R(d) = \cos\theta$$

$$d^* = \arg\min_d \left\| w_L - w_R(d) \right\|^2 = \arg\max_d \, w_L \cdot w_R(d)$$

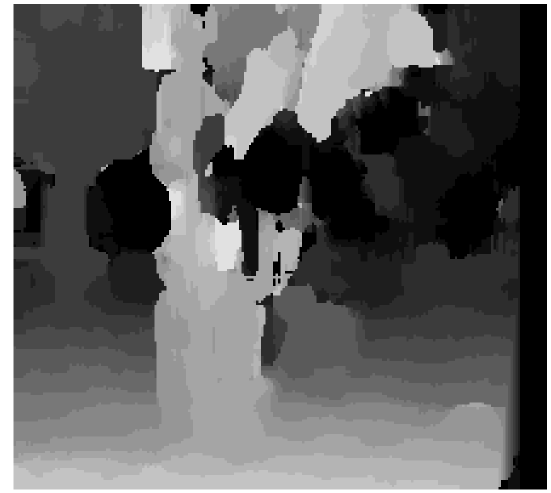
Stereo Results



Images courtesy of Point Grey Research

Window size

W = 3



W = 20

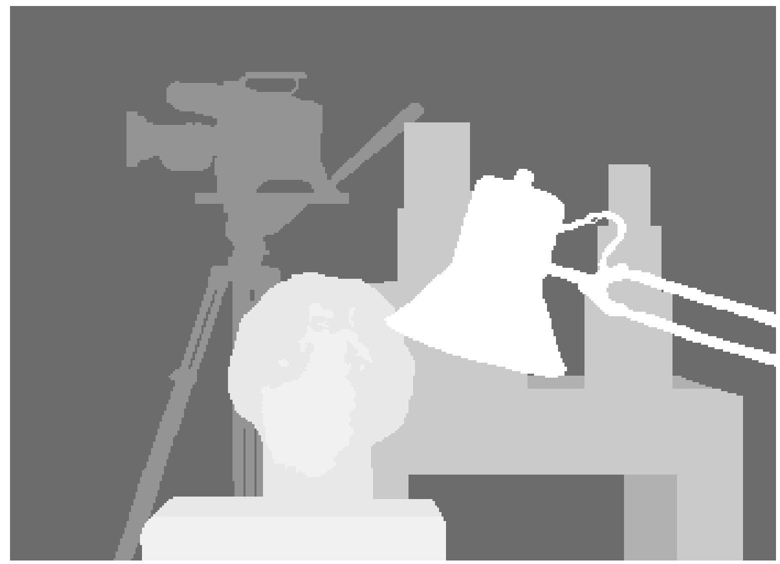
- Effect of window size
- Some approaches have been developed to use an adaptive window size (try multiple sizes and select best match)

(Seitz)

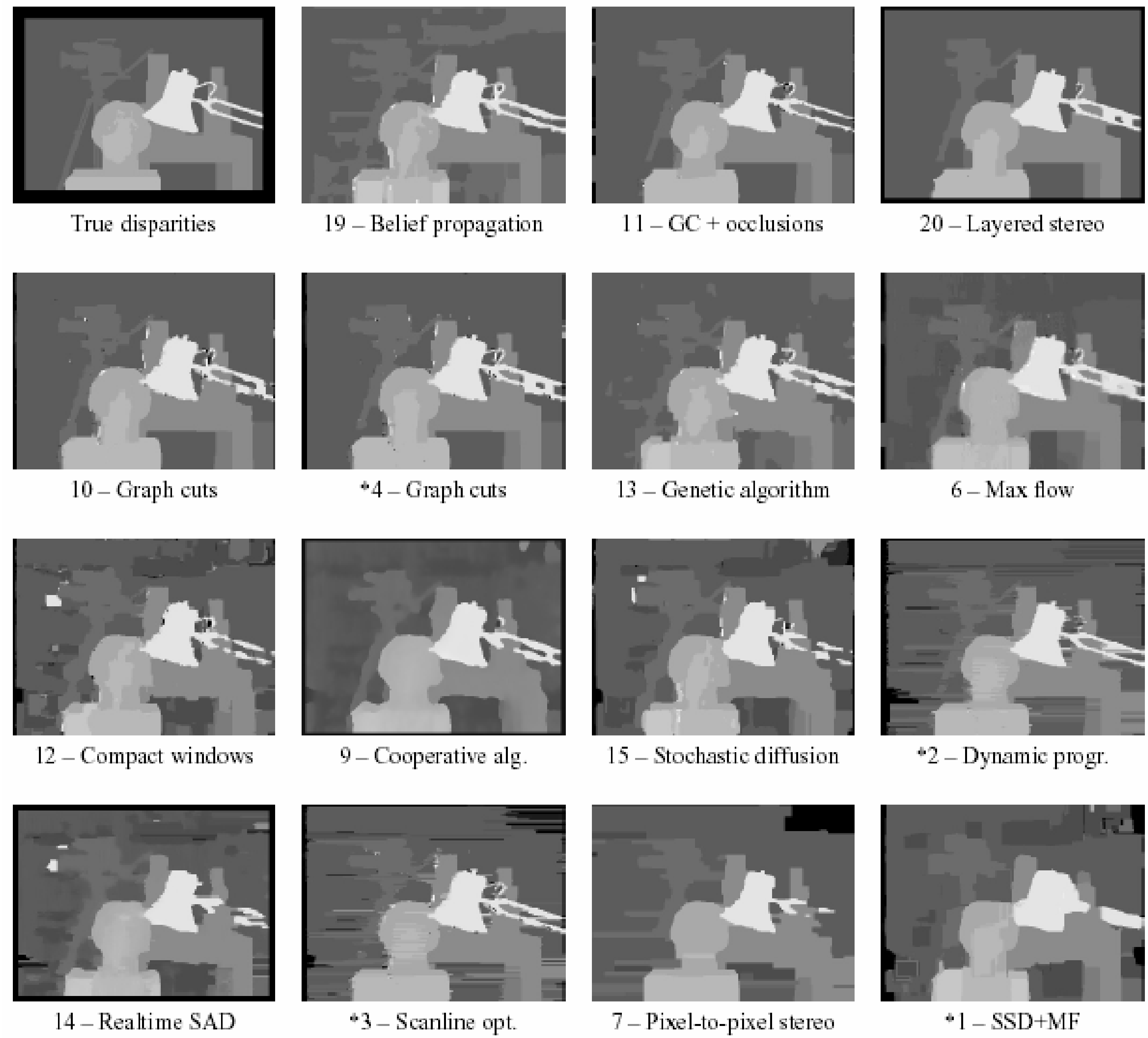
Stereo testing and comparisons

D. Scharstein and R. Szeliski. "A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms," *International Journal of Computer Vision*, **47** (2002), pp. 7-42.

Scene

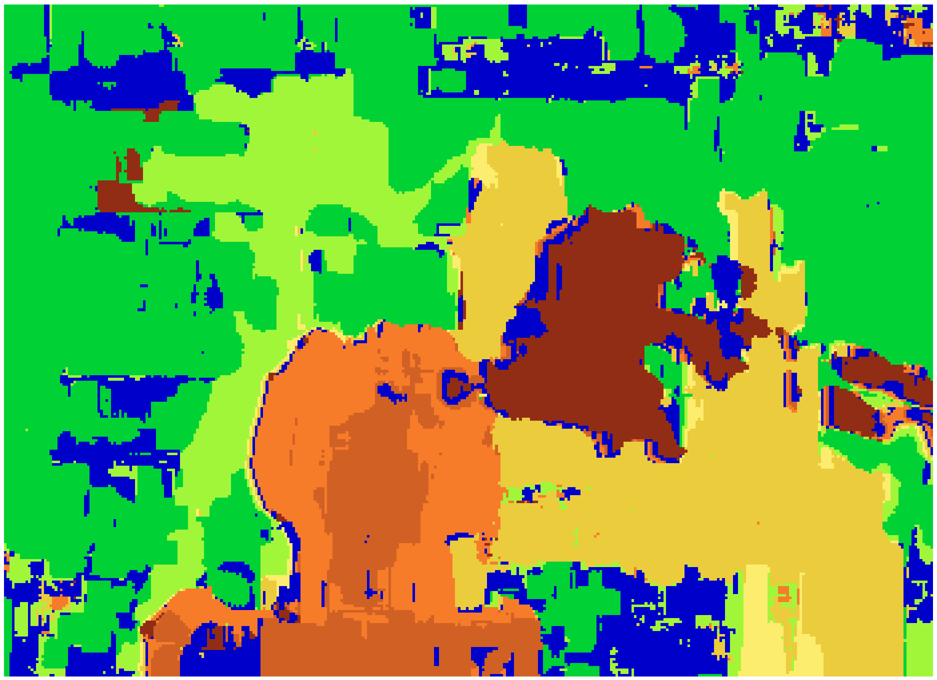


Ground truth



Scharstein and Szeliski

Results with window correlation



Window-based matching
(best window size)

Ground truth

(Seitz)

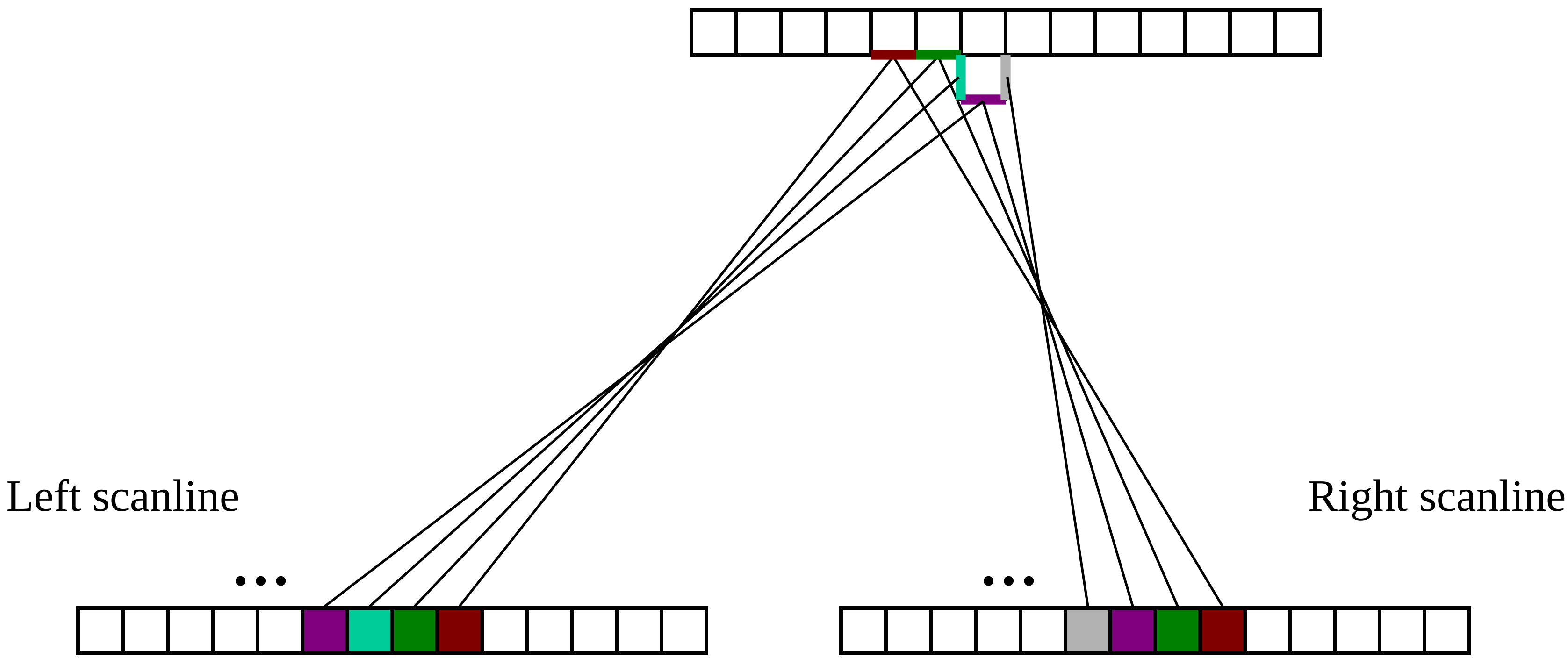
Results with better method

State of the art method: Graph cuts

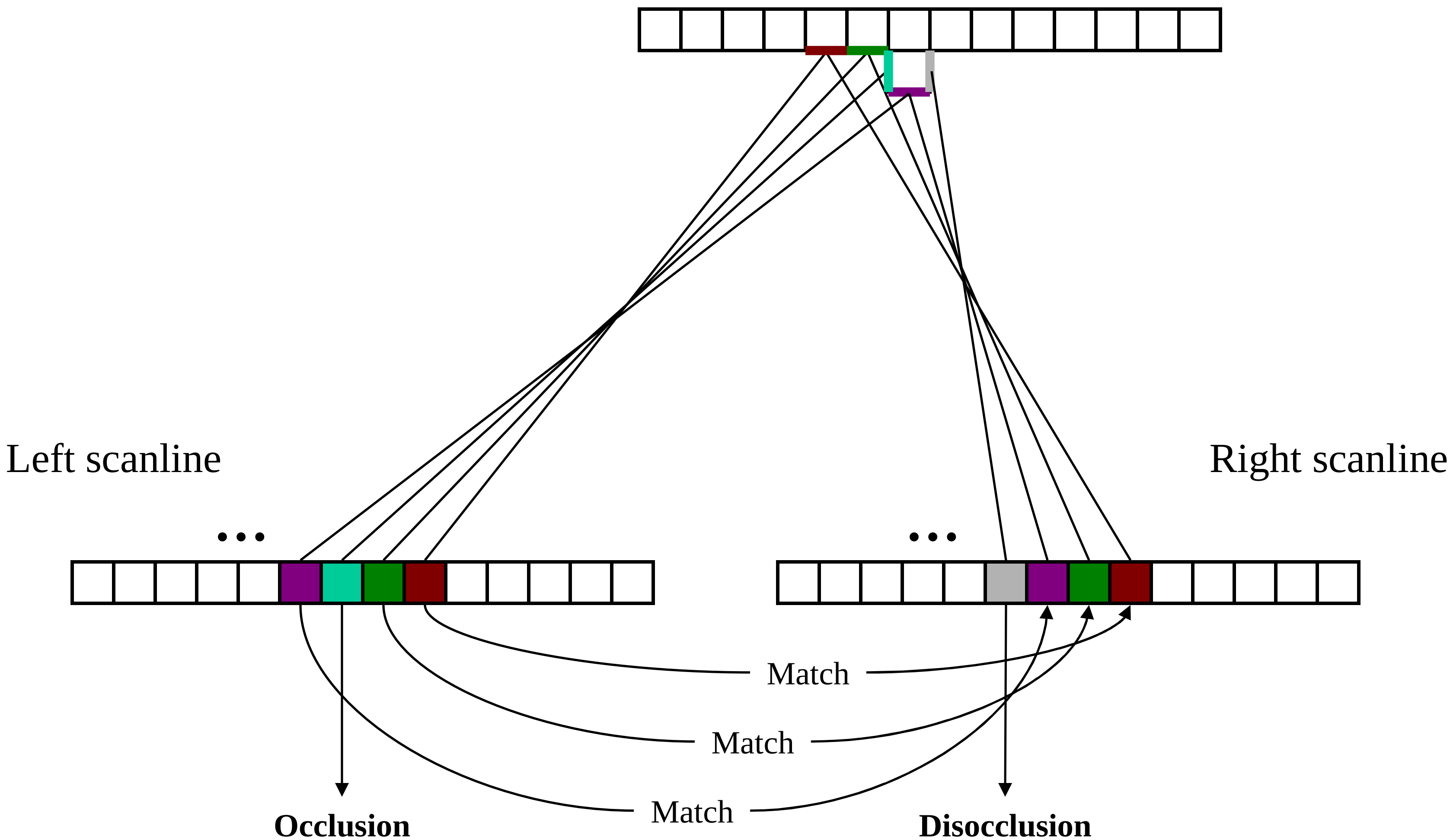
Ground truth

(Seitz)

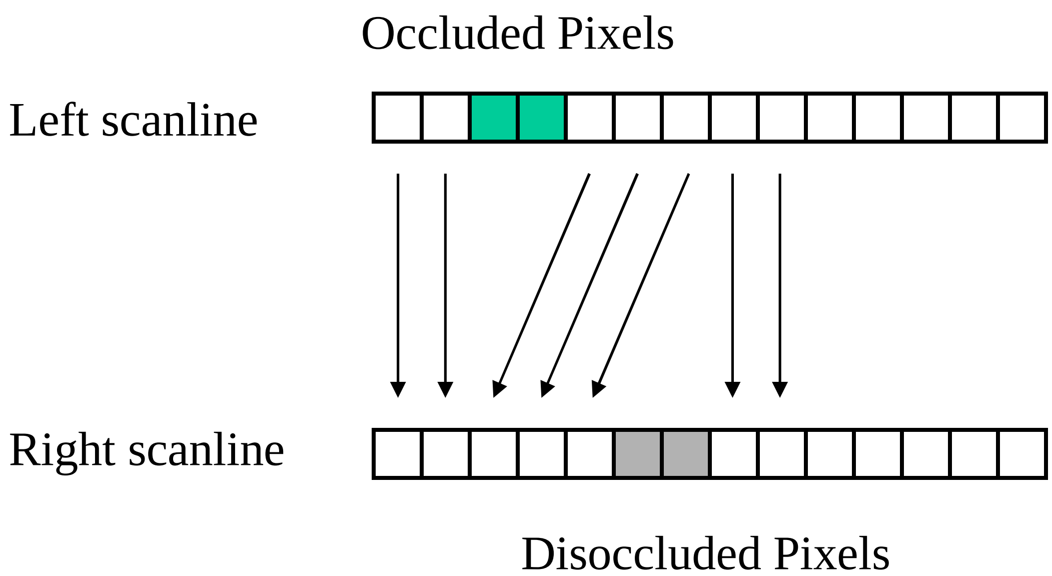
Stereo Correspondences



Stereo Correspondences



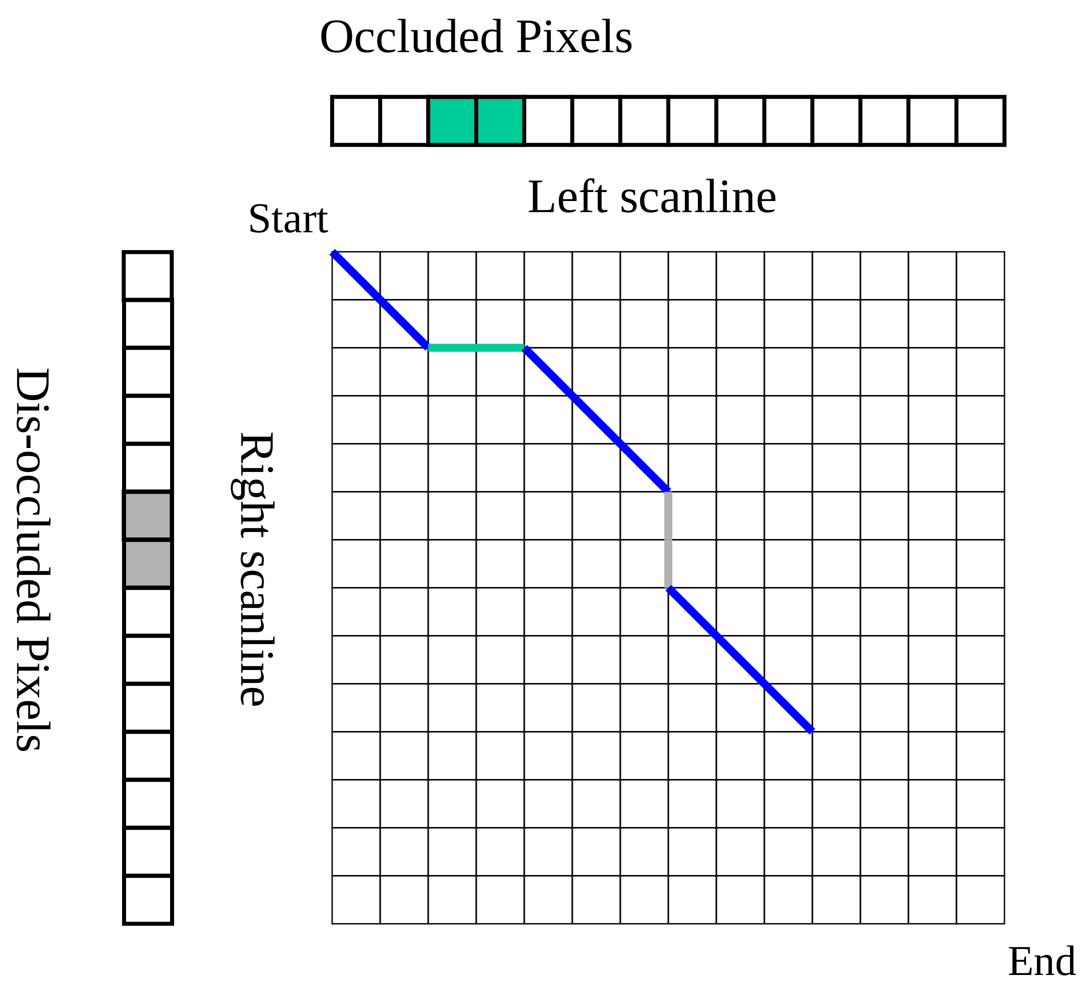
Search Over Correspondences



Three cases:

– Sequential – add cost of match (small if intensities agree)
– Occluded – add cost of no match (large cost)
– Disoccluded – add cost of no match (large cost)

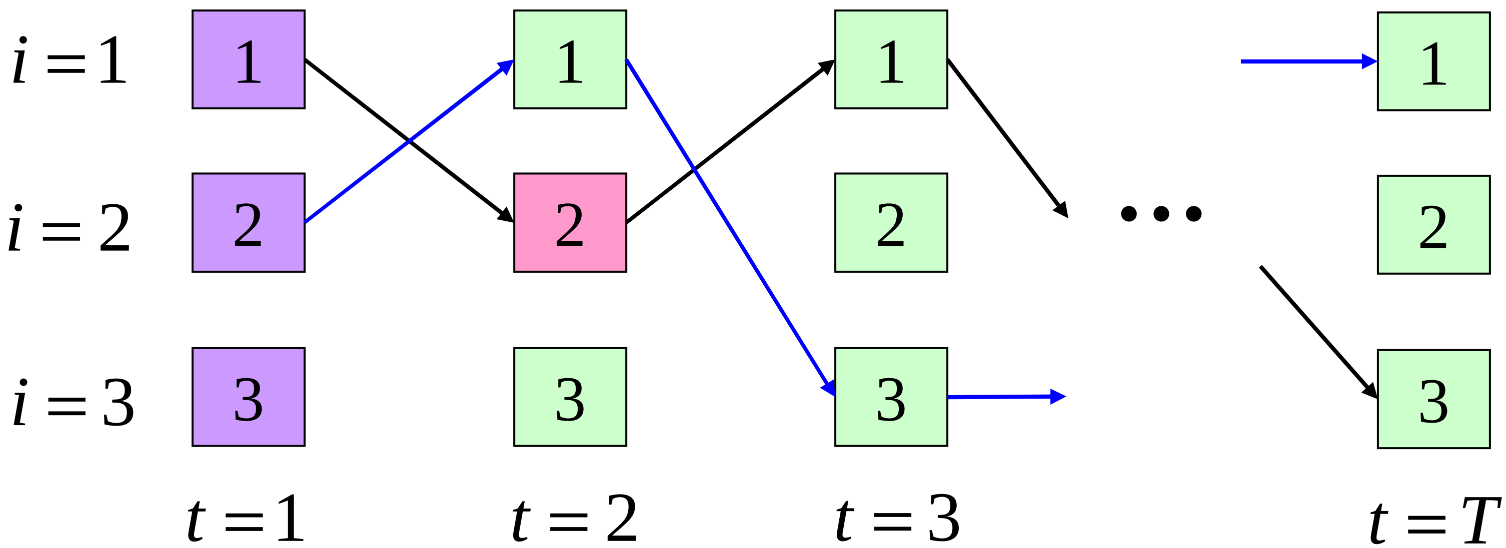
Stereo Matching with Dynamic Programming



Dynamic programming yields the optimal path through grid. This is the best set of matches that satisfy the ordering constraint

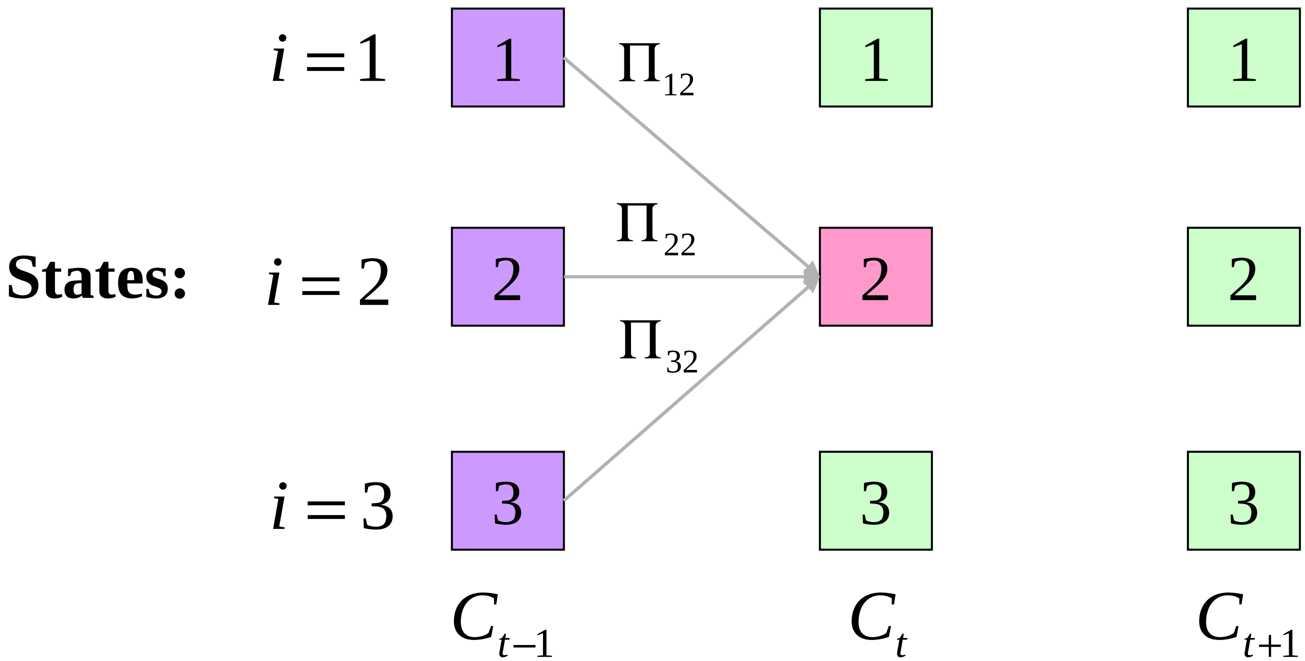
Dynamic Programming

- Efficient algorithm for solving sequential decision (optimal path) problems.



How many paths through this trellis? 3^T

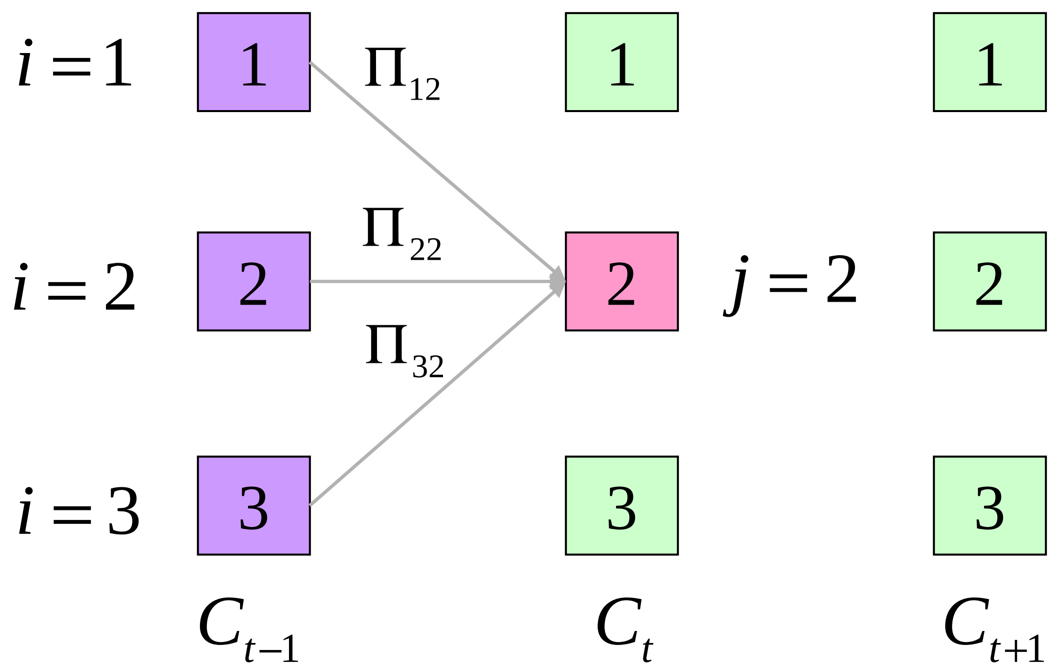
Dynamic Programming



Suppose cost can be decomposed into stages:

$$\Pi_{ij} = \text{Cost of going from state } i \text{ to state } j$$

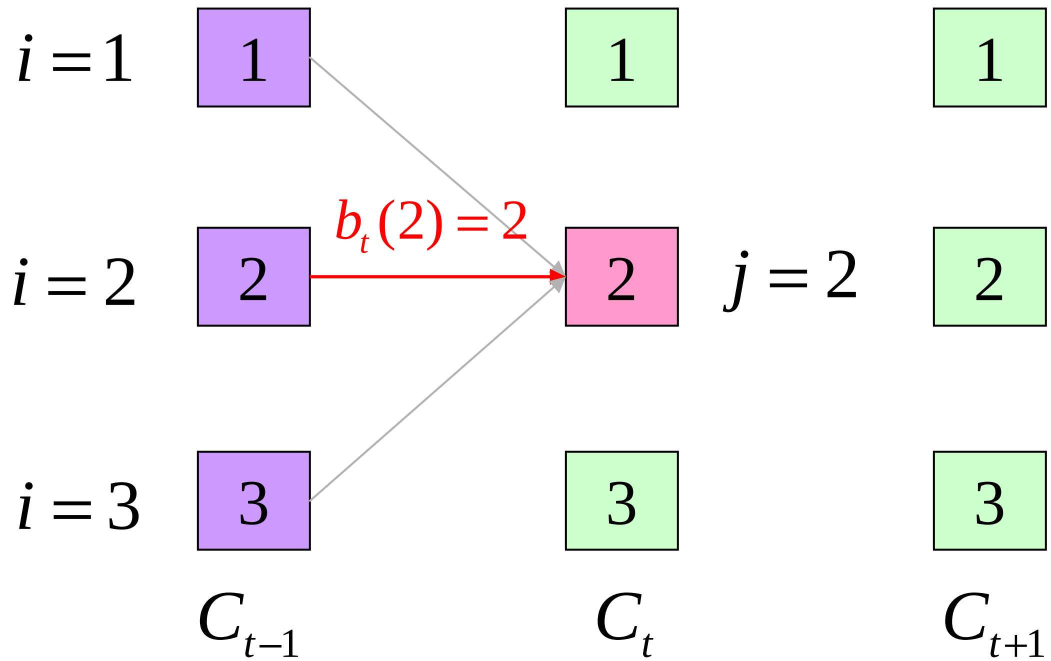
Dynamic Programming



Principle of Optimality for an n-stage assignment problem:

$$C_t(j) = \min_i(\Pi_{ij} + C_{t-1}(i))$$

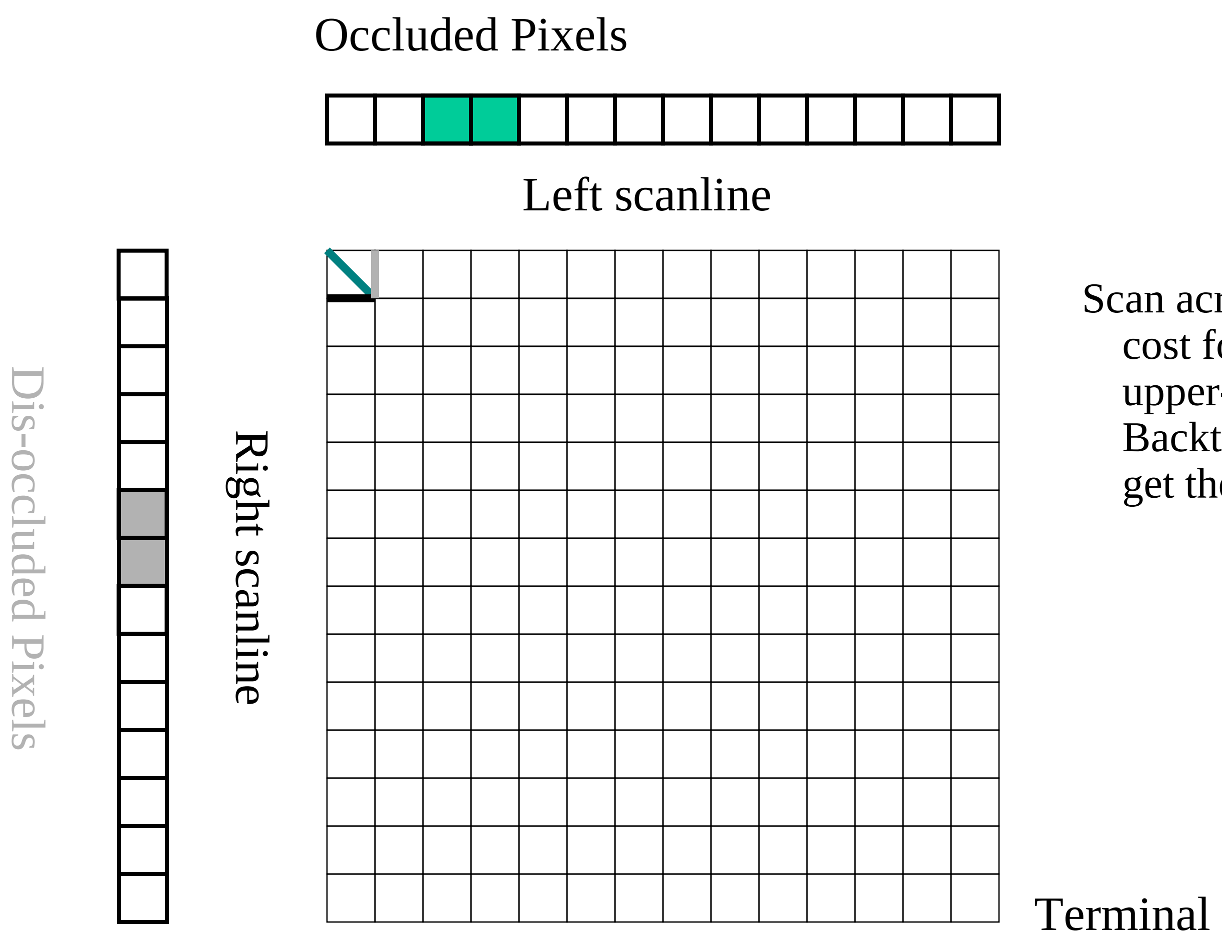
Dynamic Programming



$$C_t(j) = \min_i(\Pi_{ij} + C_{t-1}(i))$$

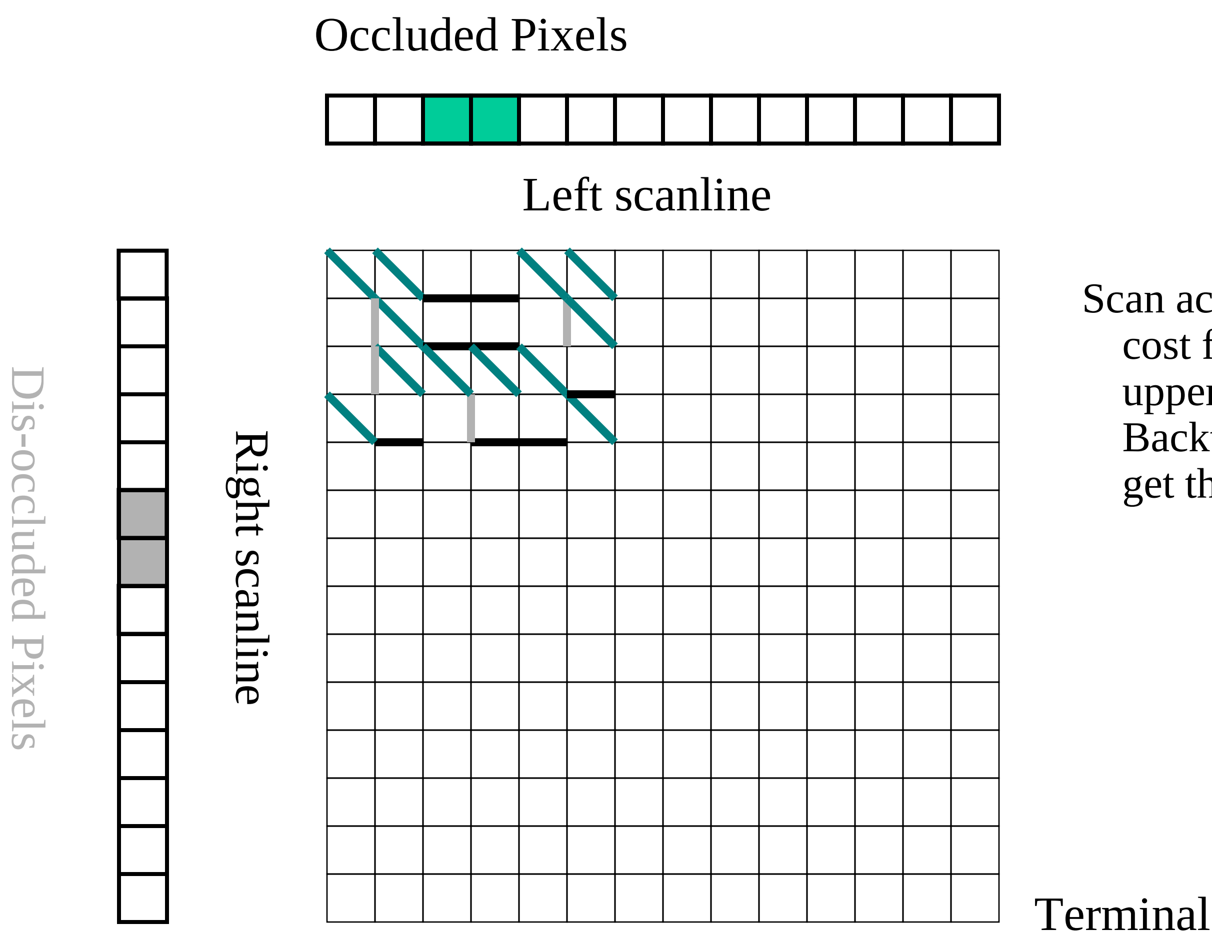
$$b_t(j) = \arg\min_i(\Pi_{ij} + C_{t-1}(i))$$

Stereo Matching with Dynamic Programming



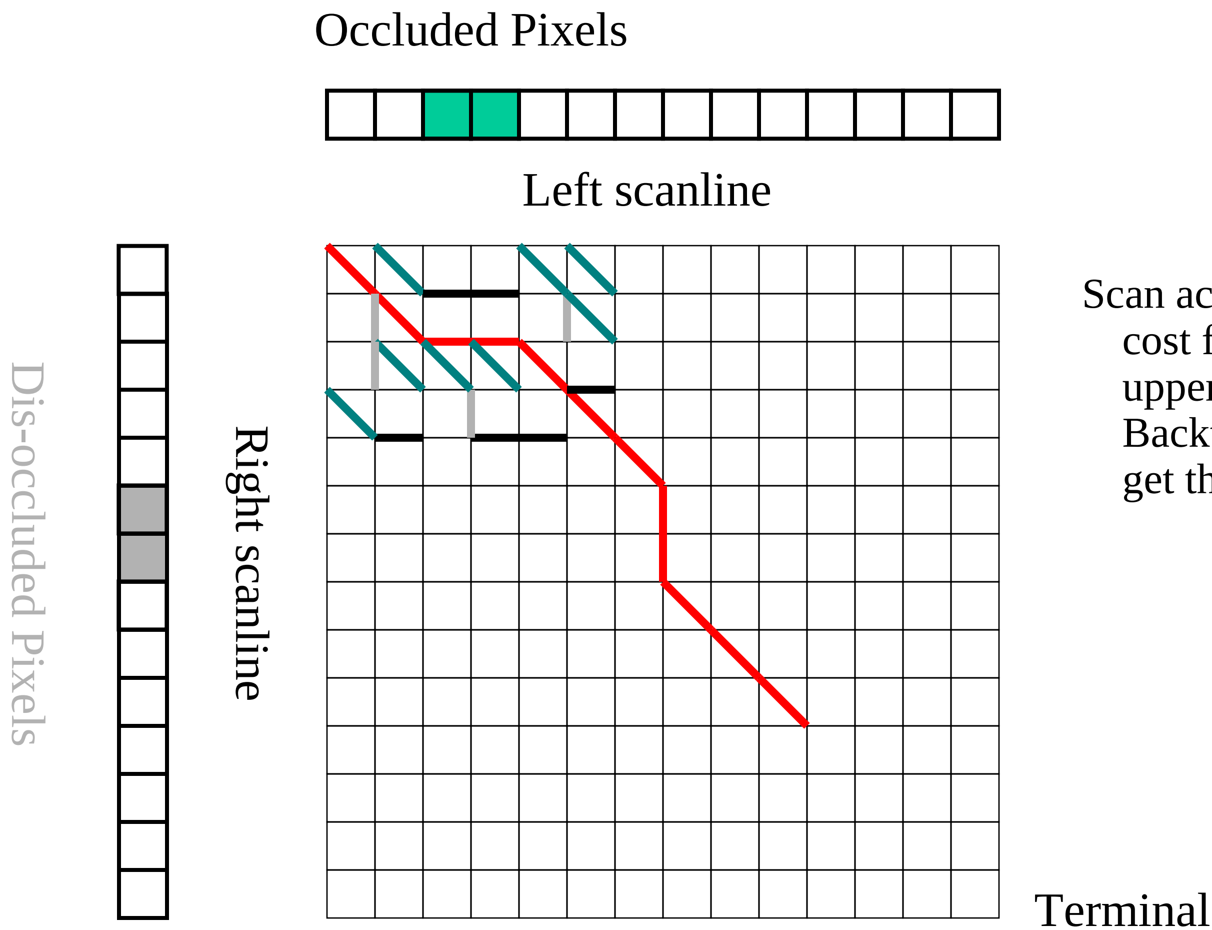
Scan across grid computing optimal cost for each node given its upper-left neighbors. Backtrack from the terminal to get the optimal path.

Stereo Matching with Dynamic Programming

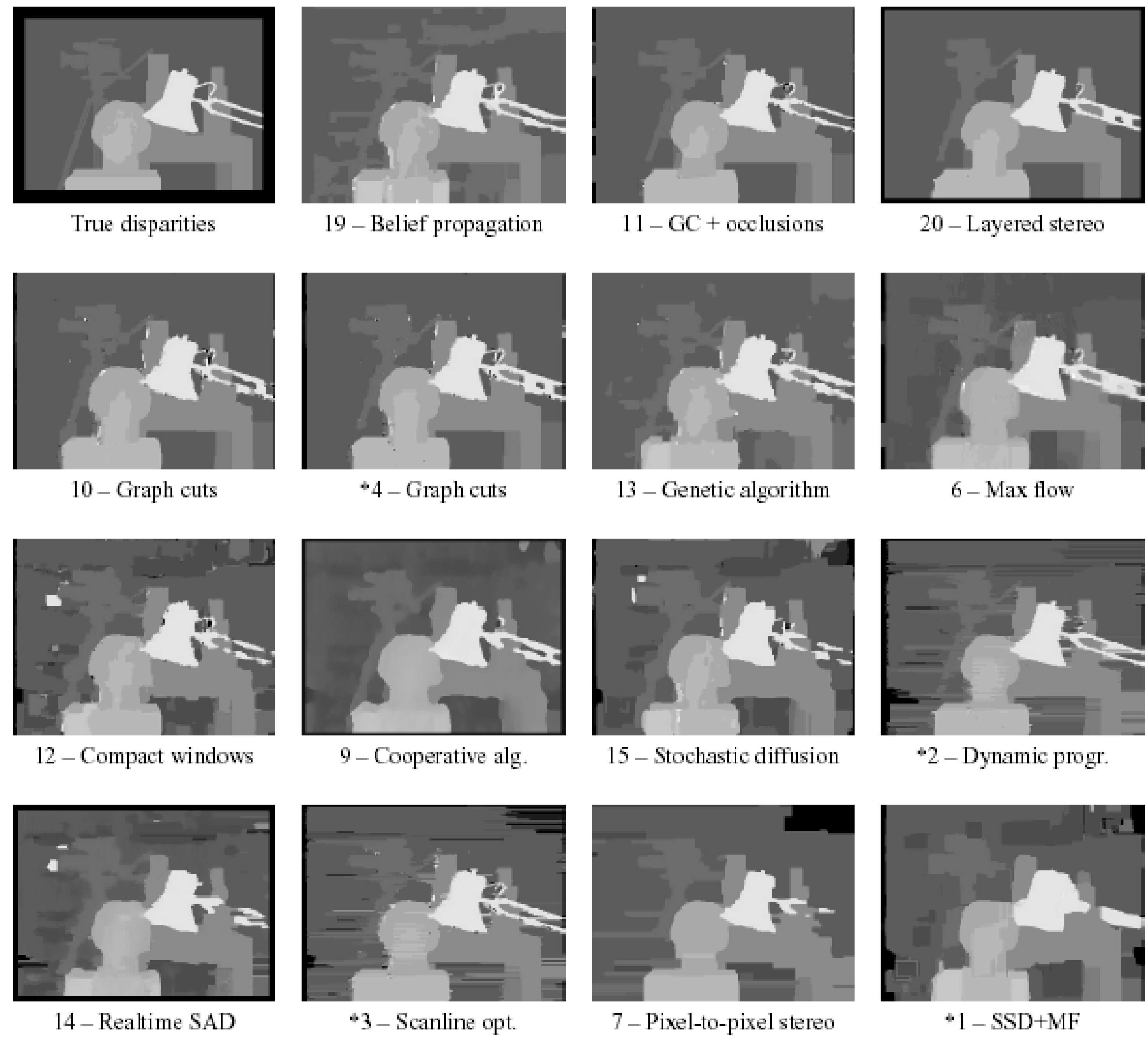


Scan across grid computing optimal cost for each node given its upper-left neighbors.
Backtrack from the terminal to get the optimal path.

Stereo Matching with Dynamic Programming

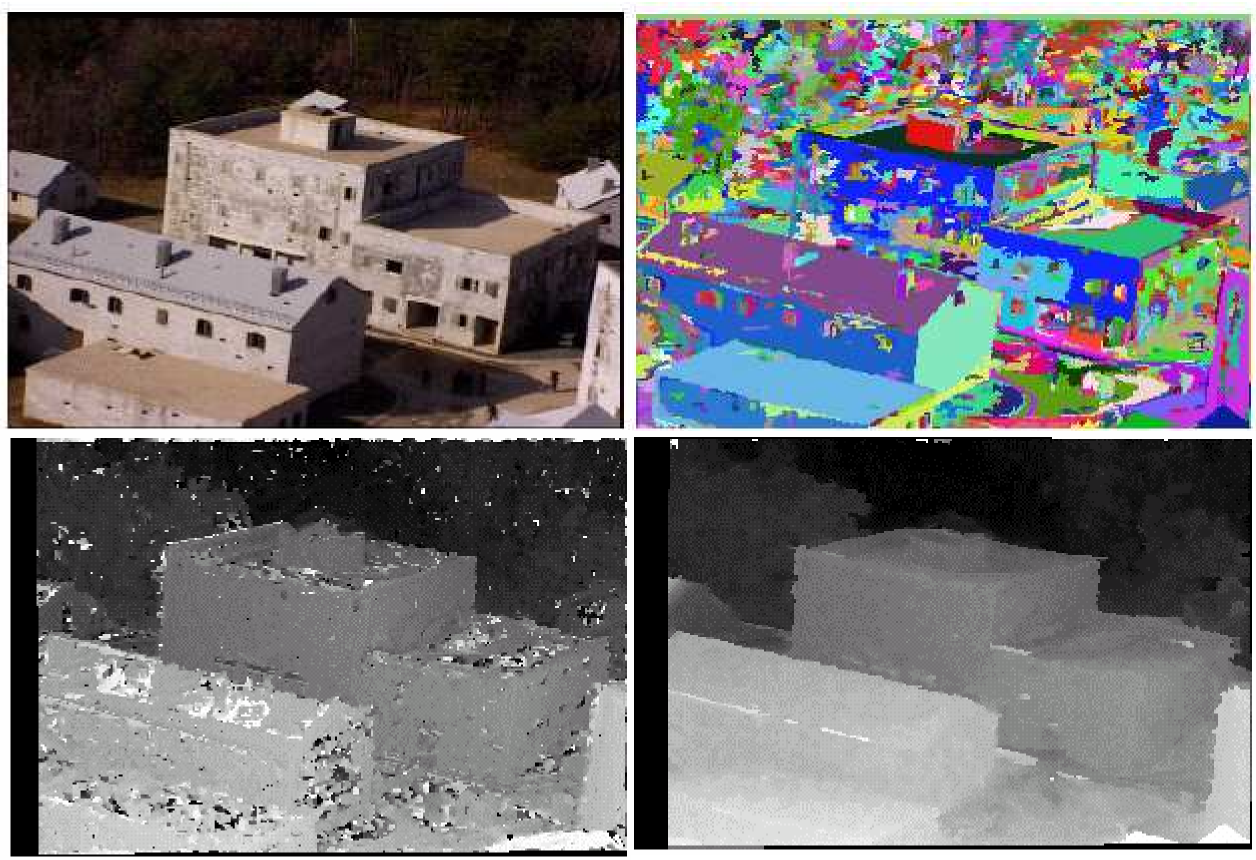


Scan across grid computing optimal cost for each node given its upper-left neighbors. Backtrack from the terminal to get the optimal path.



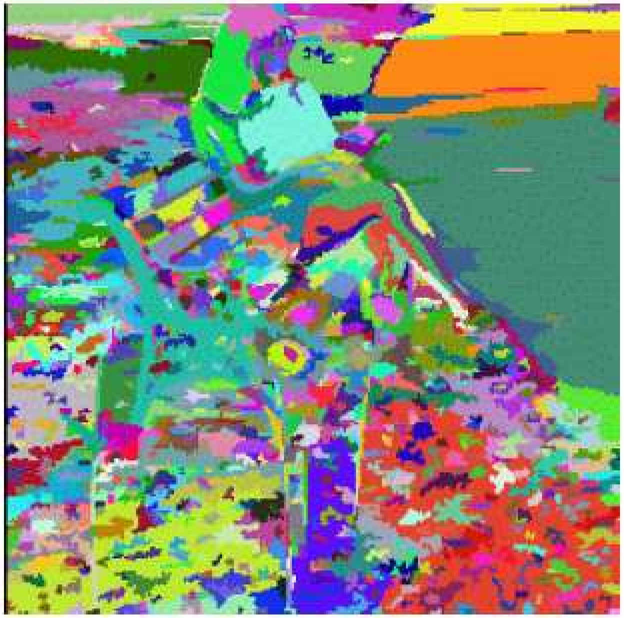
Scharstein and Szeliski

Segmentation-based Stereo

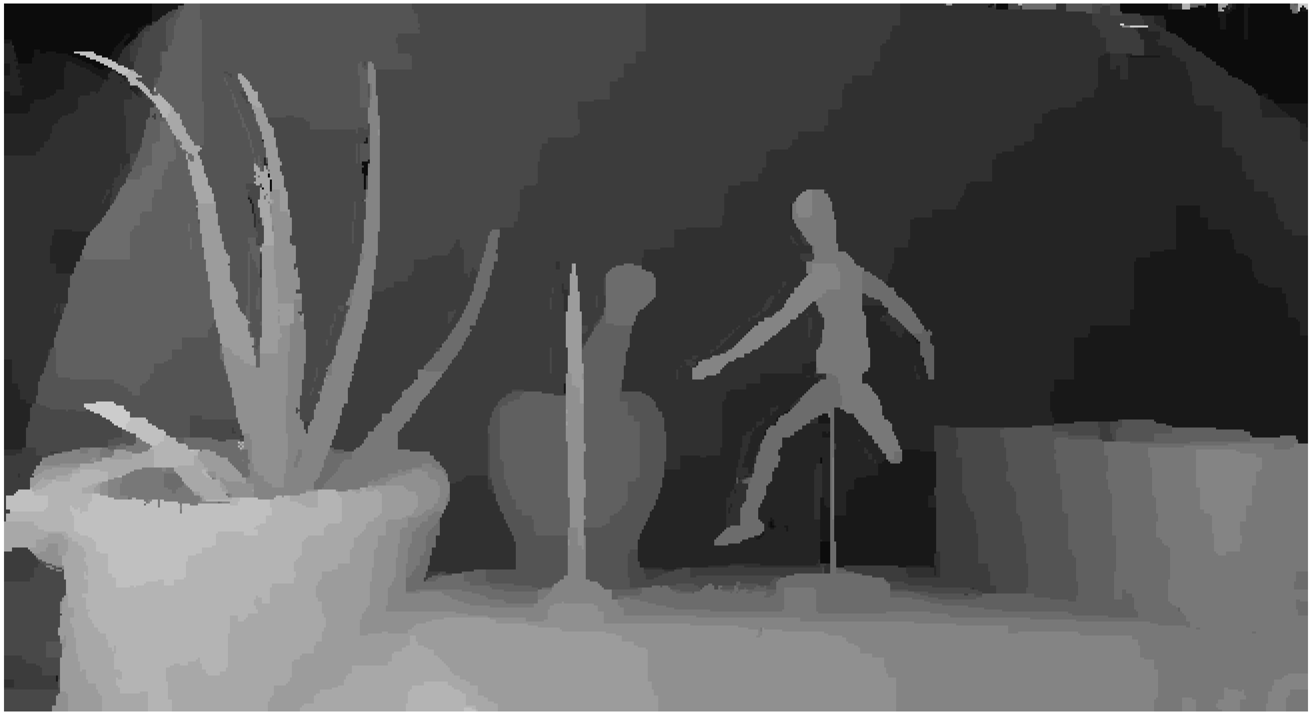


Hai Tao and Harpreet W. Sawhney

Another Example

Result using a good technique



Right Image

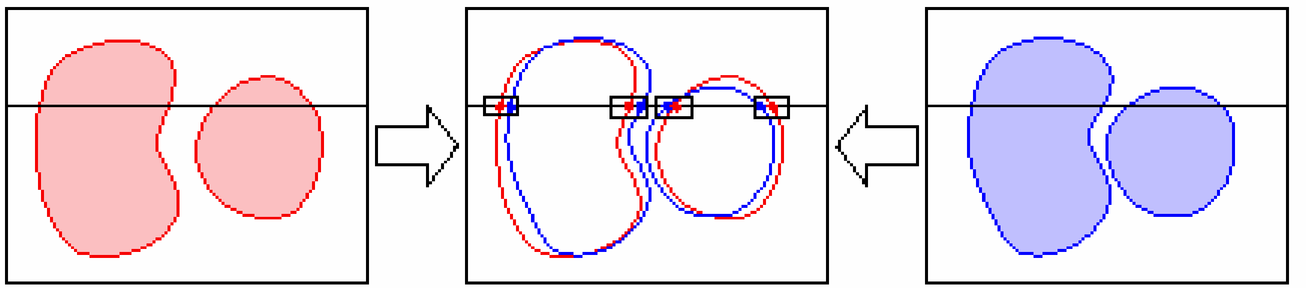
Left Image

Disparity

View Interpolation

Computing Correspondence

- Another approach is to match *edges* rather than windows of pixels:



- V
  - Edges tend to fail in dense texture (outdoors)
  - Correlation tends to fail in smooth featureless areas

Summary of different stereo methods

- **Constraints:**
  - Geometry, epipolar constraint.
  - Photometric: Brightness constancy, only partly true.
  - Ordering: only partly true.
  - Smoothness of objects: only partly true.

- **Algorithms:**
  - What you compare: points, regions, features?

- **How you optimize:**
  - Local greedy matches.
  - 1D search.
  - 2D search.