KECE471 Computer Vision

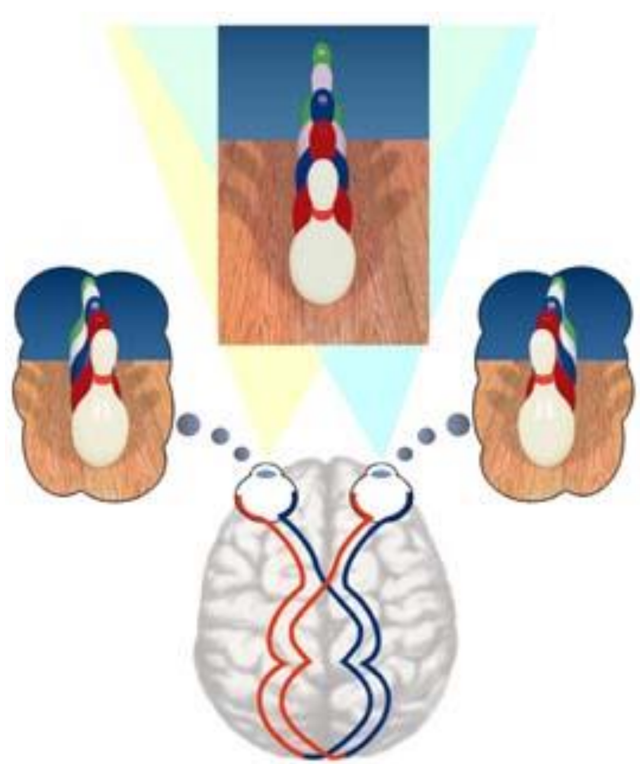
Stereo

Chang-Su Kim

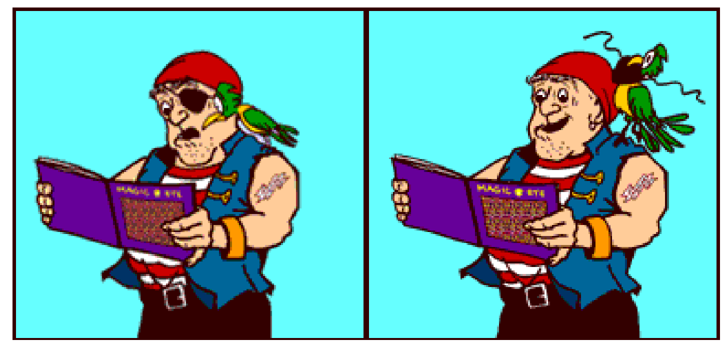
Chapter 11, Computer Vision by Forsyth and Ponce
Note: Most contents were copied from the lecture notes of Prof. Kyeong Mu Lee in SNU

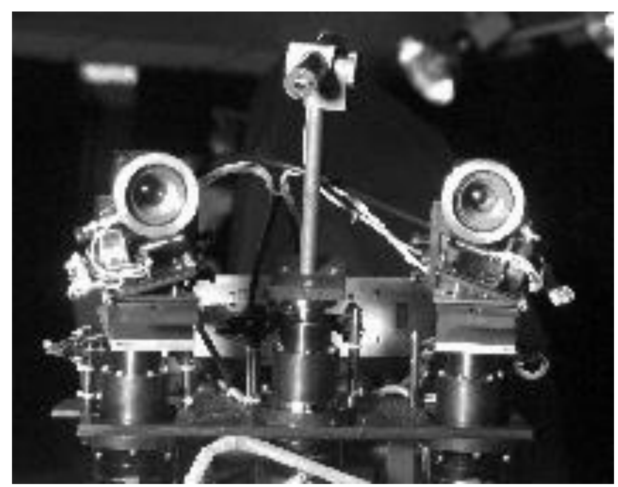
Stereo

- Inferring depth information using two cameras like a human
- Two eyes perceives three-dimension



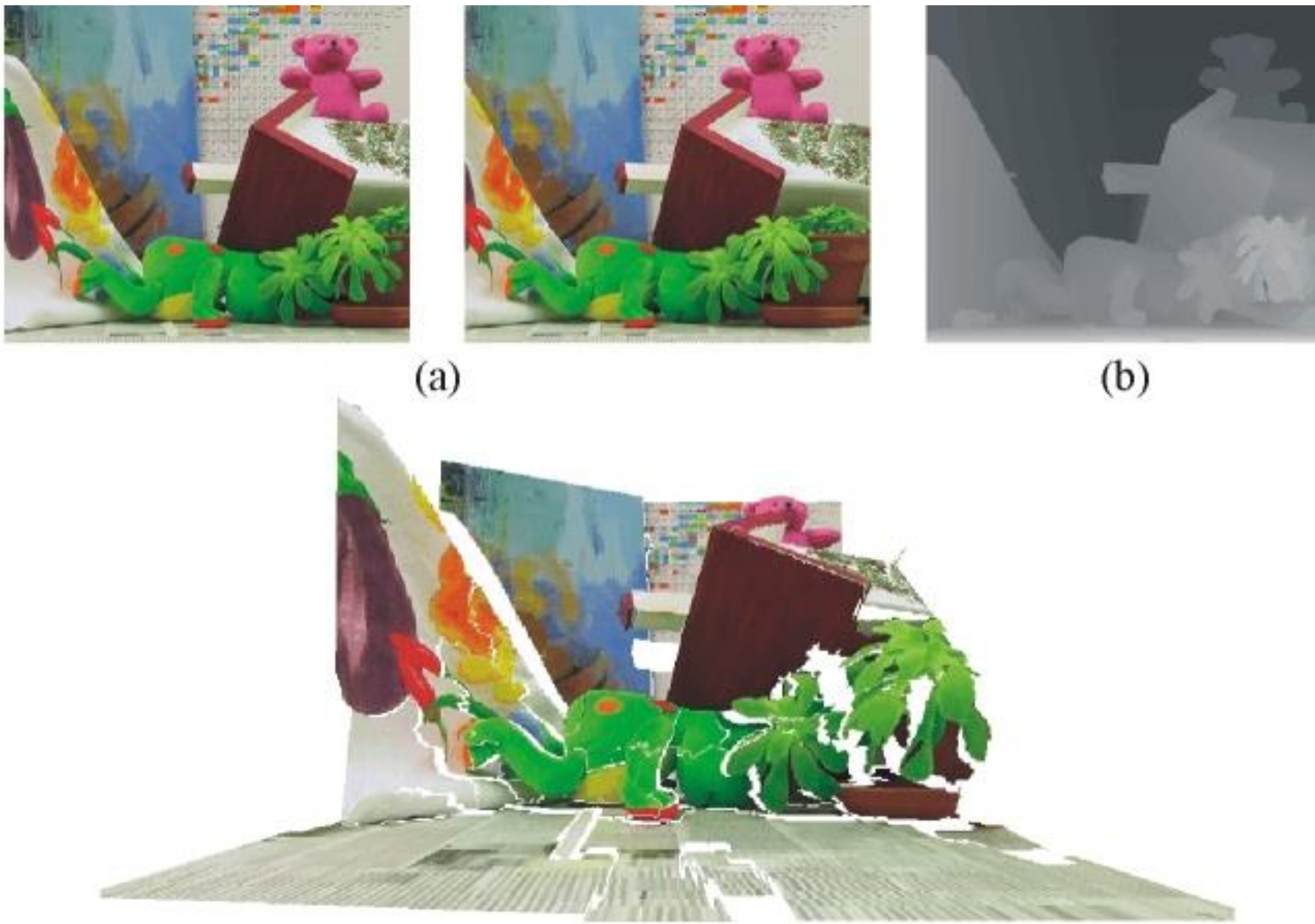
Human eyes





Robot eyes

Stereo



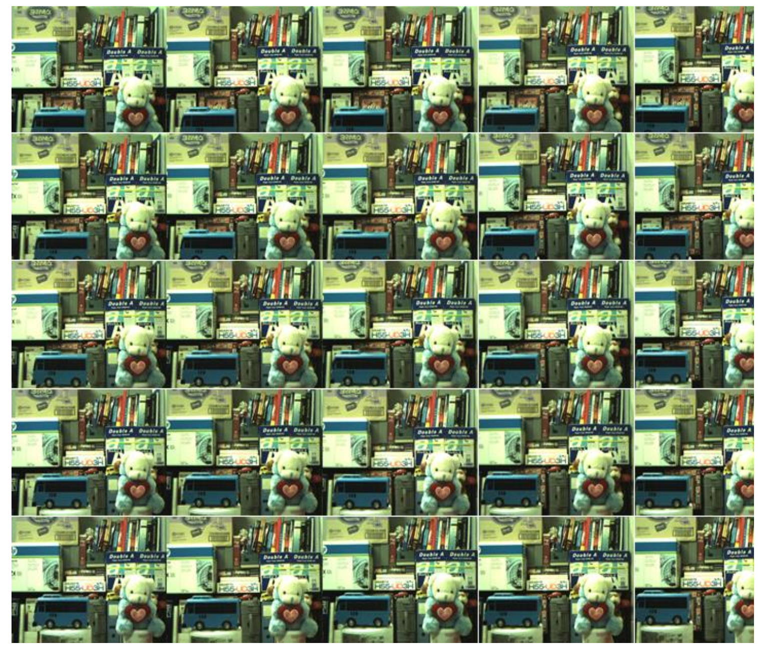
Public Library, Stereoscopic Looking Room, Chicago, by Phillips, 1923


Teesta suspension bridge-Darjeeling, India

Stereo

- Inferring depth information using two eyes or cameras
- Two eyes perceive 3rd dimension

(a)



(b)

Applications

[Matthies,Szeliski,Kanade'88]

Applications

input image

317 images (hemisphere)

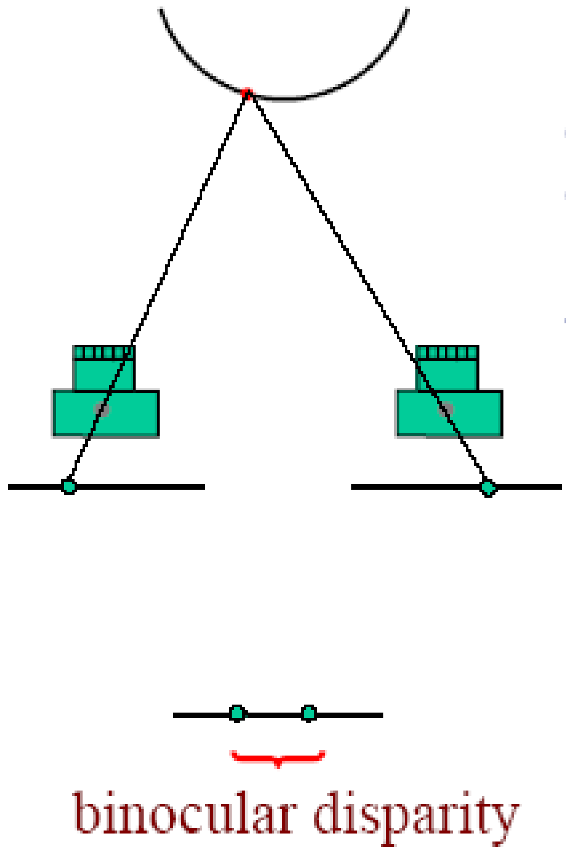
ground truth model

[Goesele, Curless, Seitz, 2006]

Binocular Stereo

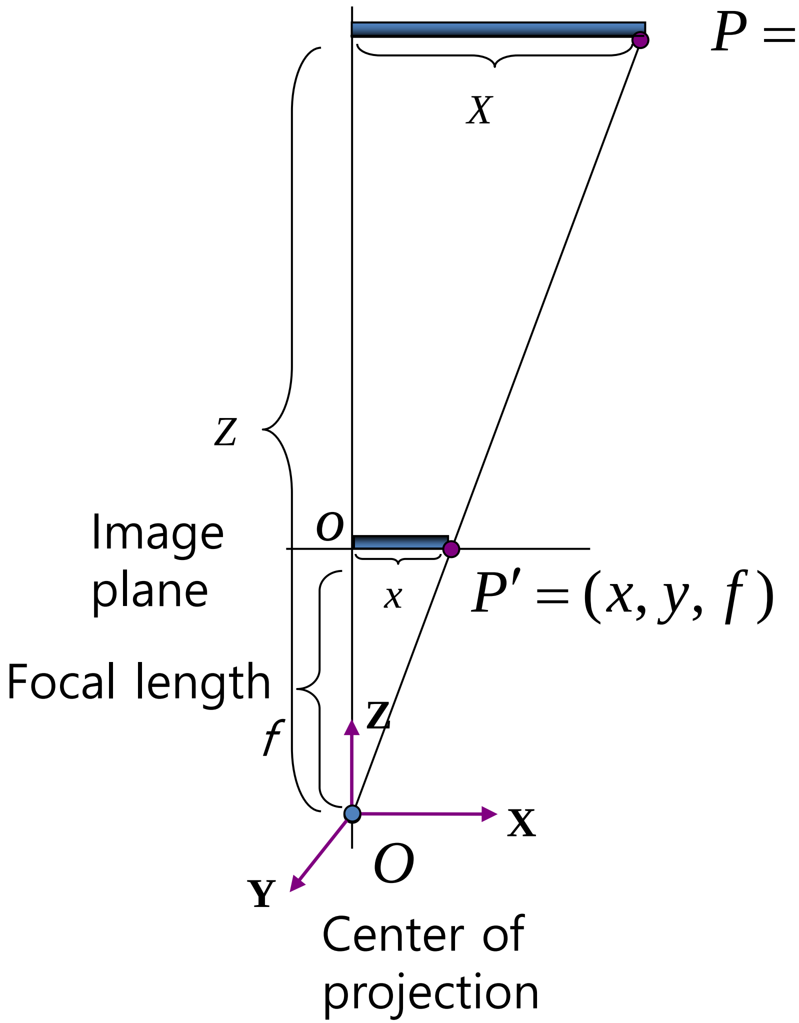
From known geometry of the cameras and estimated disparity, recover depth in the scene

Left



Right

Pinhole Camera Model



- 3D to 2D projection:

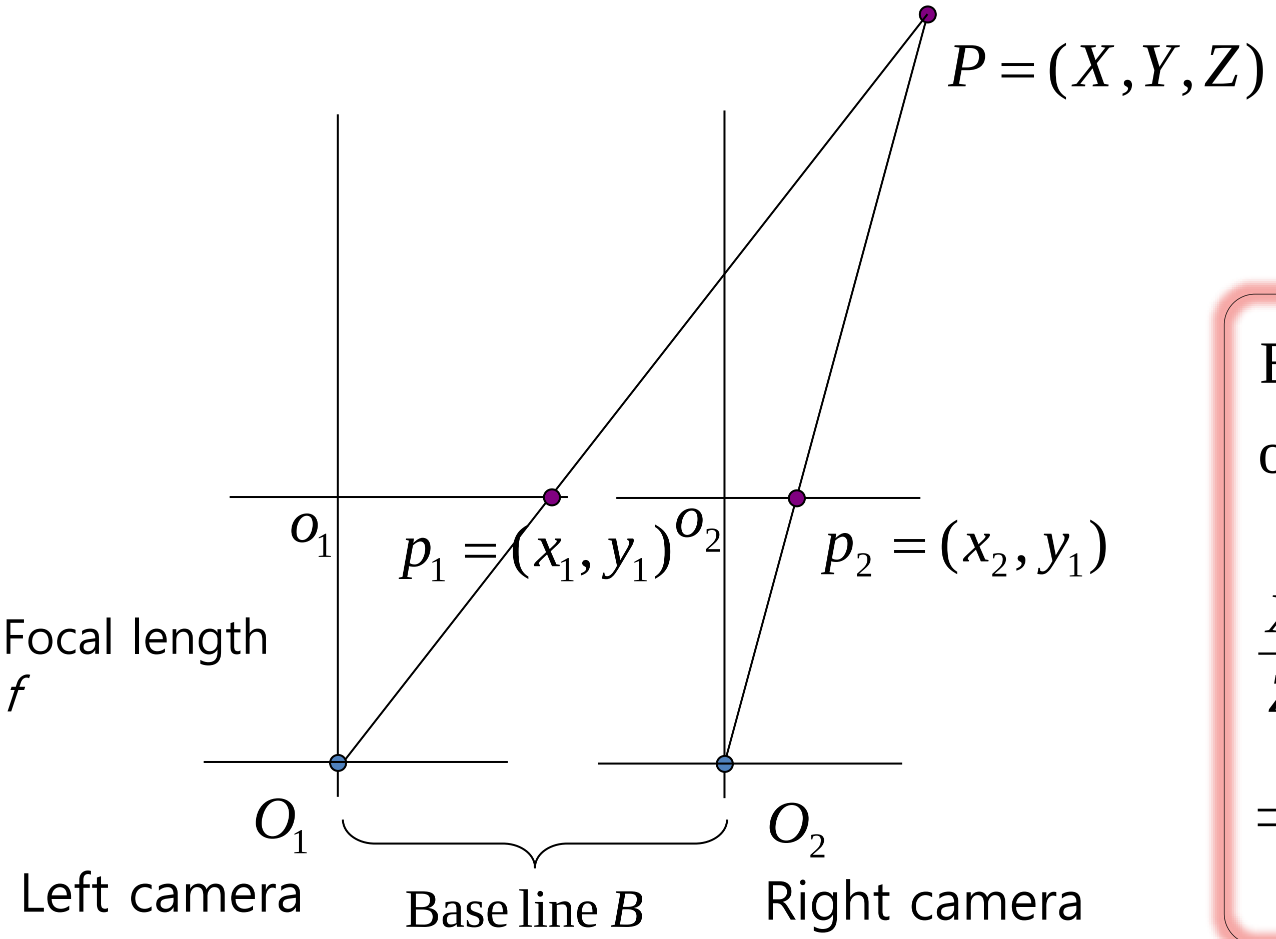
$$\frac{x}{X} = \frac{f}{Z} \Rightarrow x = f\,\frac{X}{Z}$$

$$\frac{y}{Y} = \frac{f}{Z} \Rightarrow y = f\,\frac{Y}{Z}$$

Thus

$$(X, Y, Z) \rightarrow (x, y) = (f\,\frac{X}{Z},\ f\,\frac{Y}{Z})$$

Basic Stereo Model

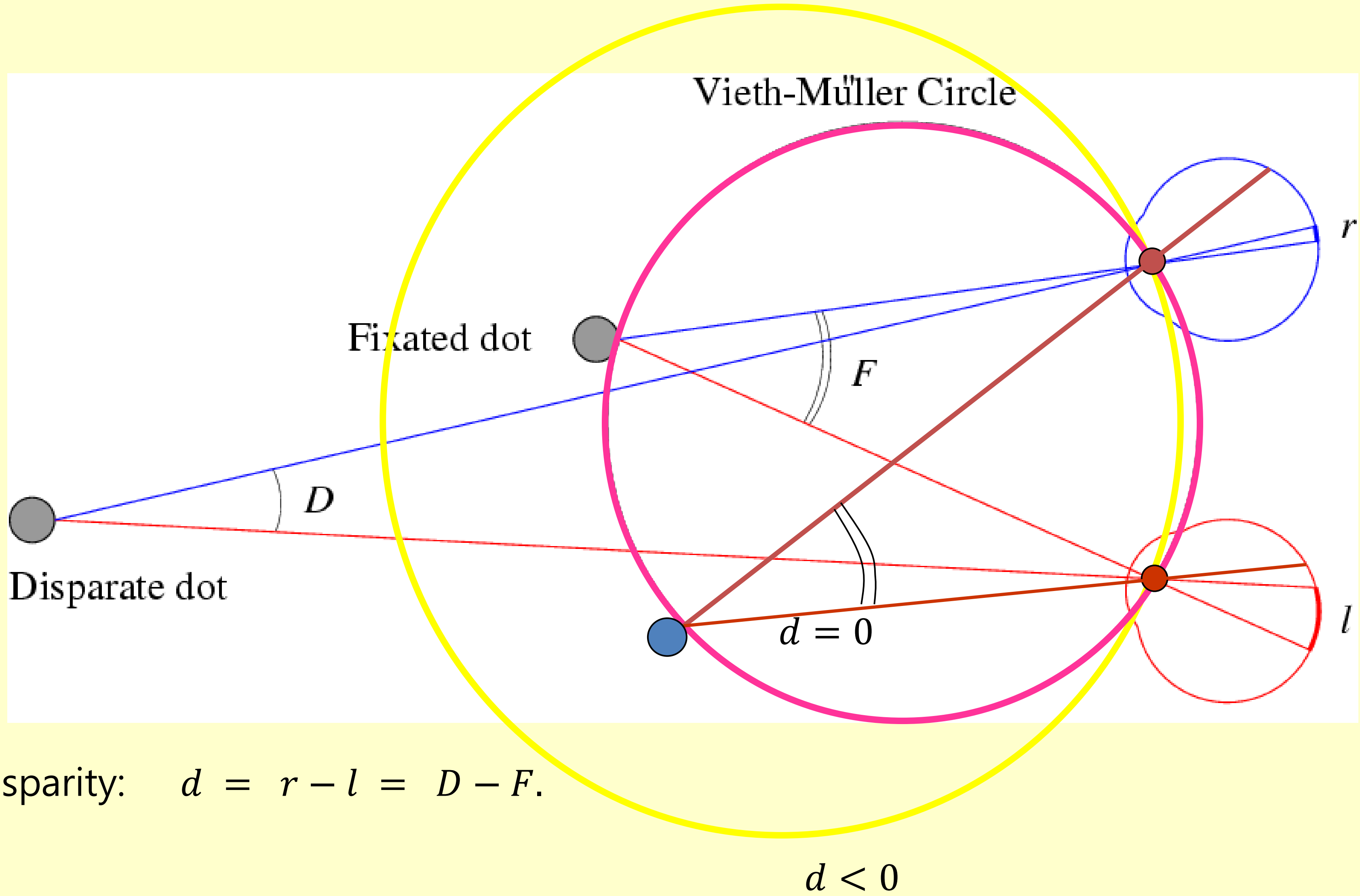


Express Z as a function of x_1, x_2, f, B

$$\frac{X}{Z} = \frac{x_1}{f} \text{ and } \frac{X - B}{Z} = \frac{x_2}{f}$$

$$\Rightarrow Z = \frac{fB}{x_1 - x_2} = \frac{fB}{d(p_1)}$$

Human Stereopsis: Reconstruction



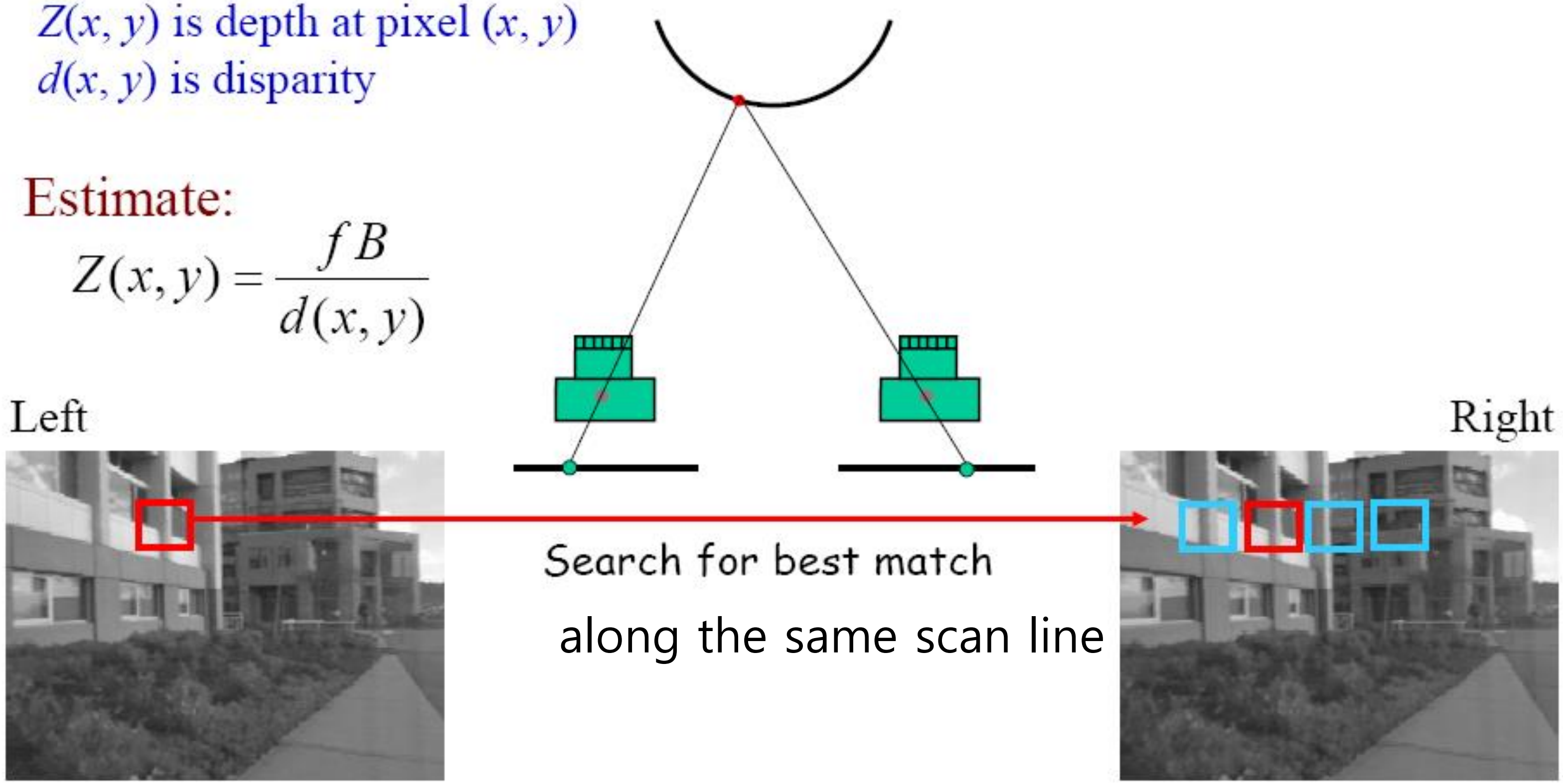
Disparity: $d = r - l = D - F.$

Finding Correspondence

$Z(x, y)$ is depth at pixel (x, y)
$d(x, y)$ is disparity

Estimate:

$$Z(x,y) = \frac{f\,B}{d(x,y)}$$

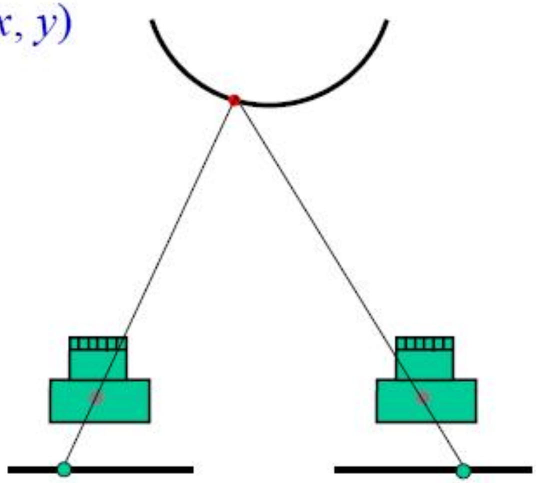


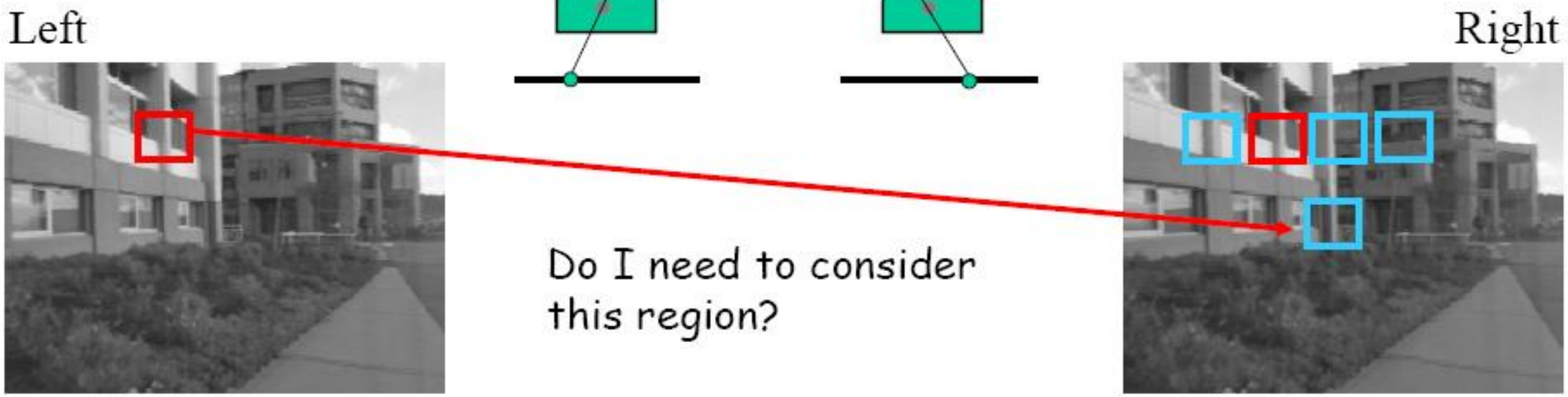
Finding Correspondence

$Z(x, y)$ is depth at pixel (x, y)
$d(x, y)$ is disparity

Estimate:

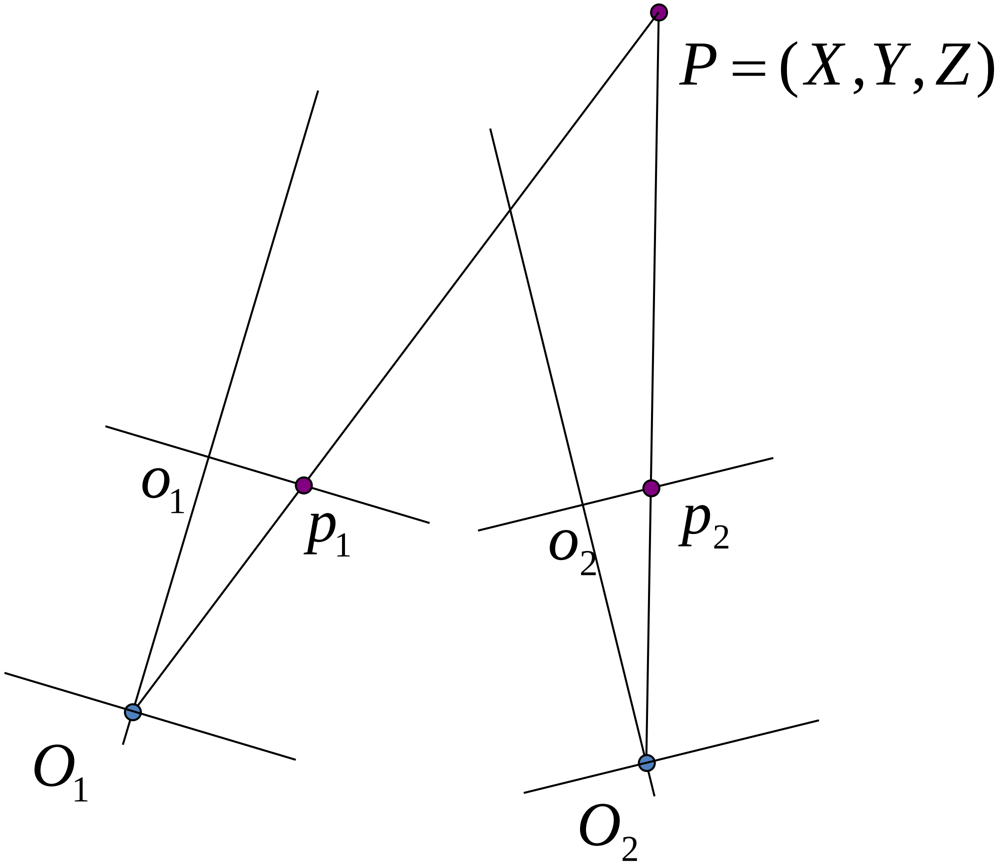
$$Z(x,y) = \frac{f B}{d(x,y)}$$



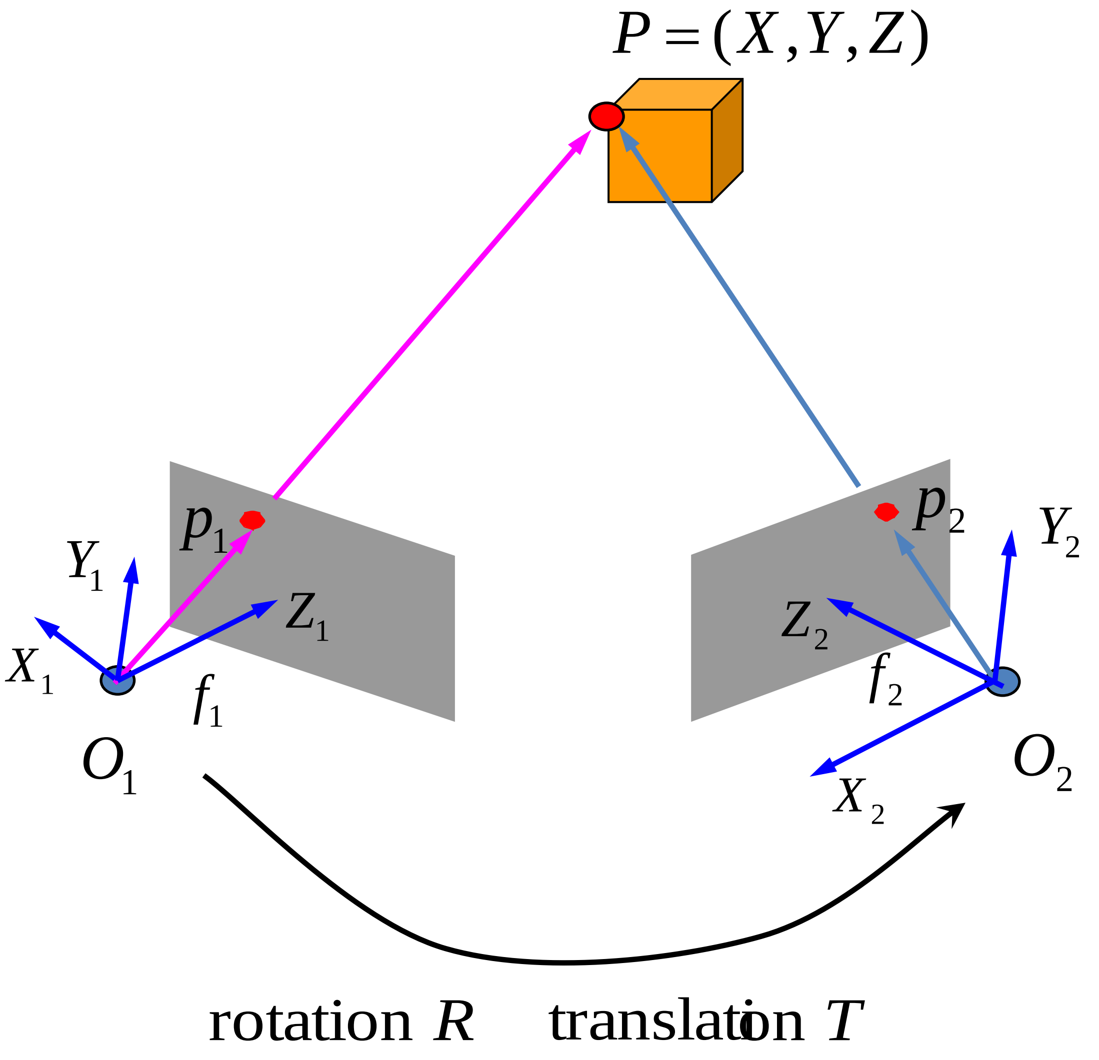


General stereo

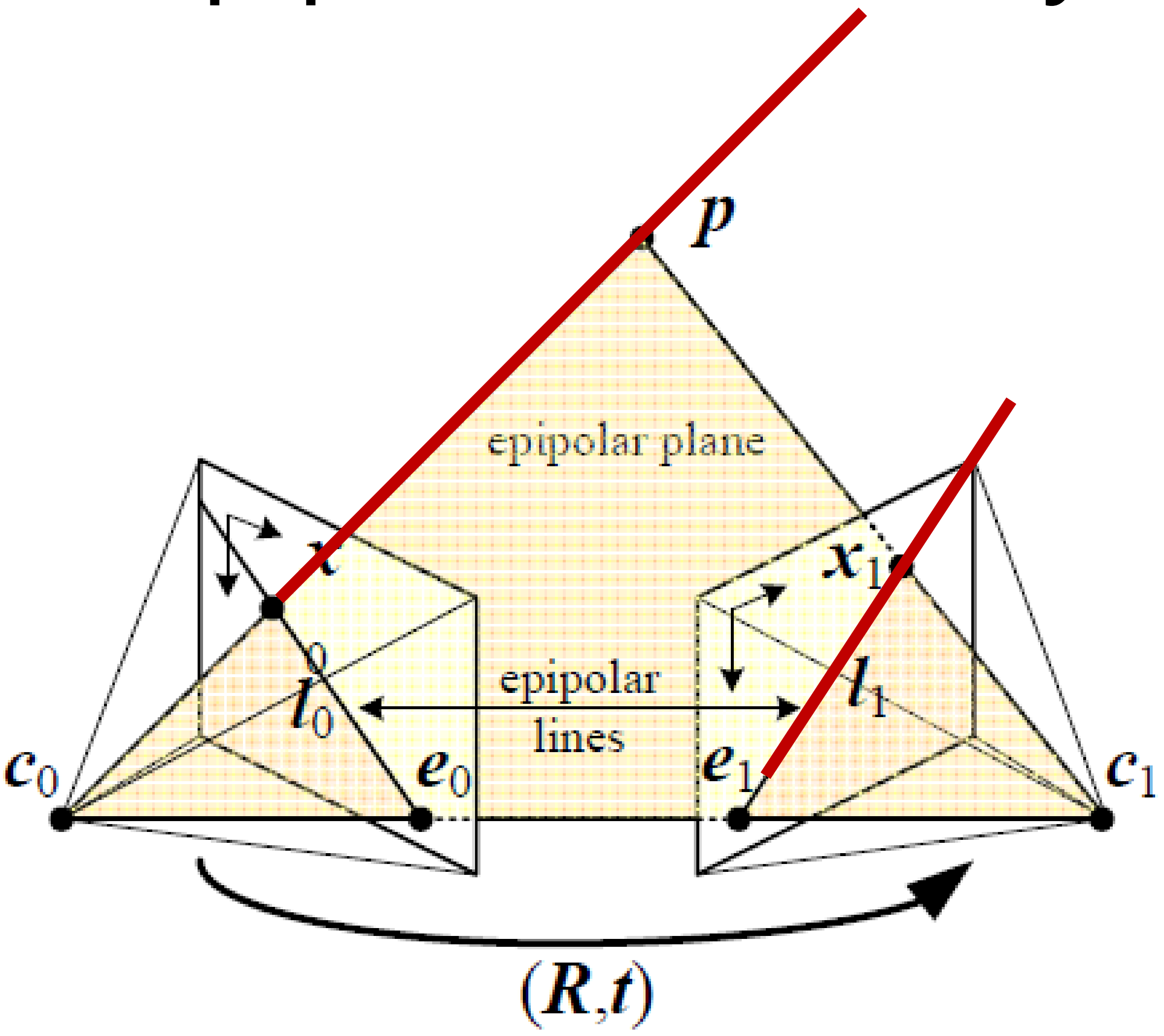
- What if two cameras are not parallel?



Epipolar Geometry

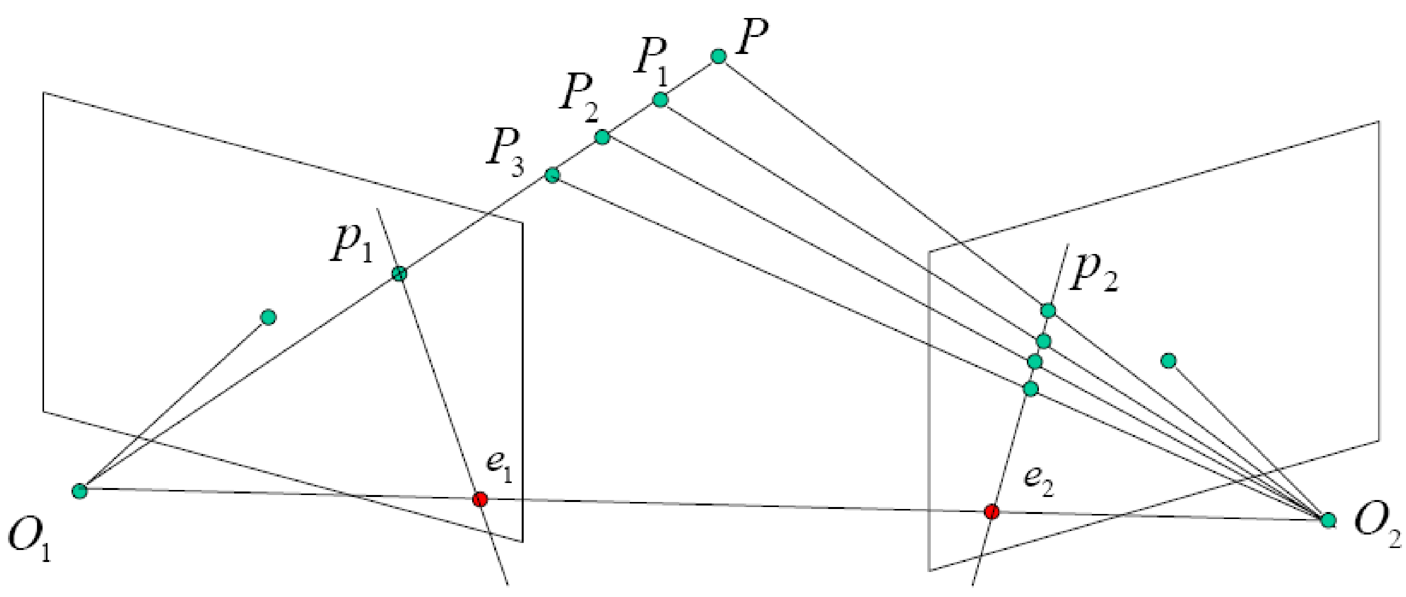


Epipolar Geometry



Epipolar Geometry

- Epipolar Constraint
  - A matching points lies on the associated epipolar line
  - It reduces the correspondence problem to 1D search along the epipolar line
  - It reduces the cost and ambiguity of matching



Rectification

- Simple case
  - Cameras are parallel
  - Focal lengths are the same
  - Two image planes lie on the same plane

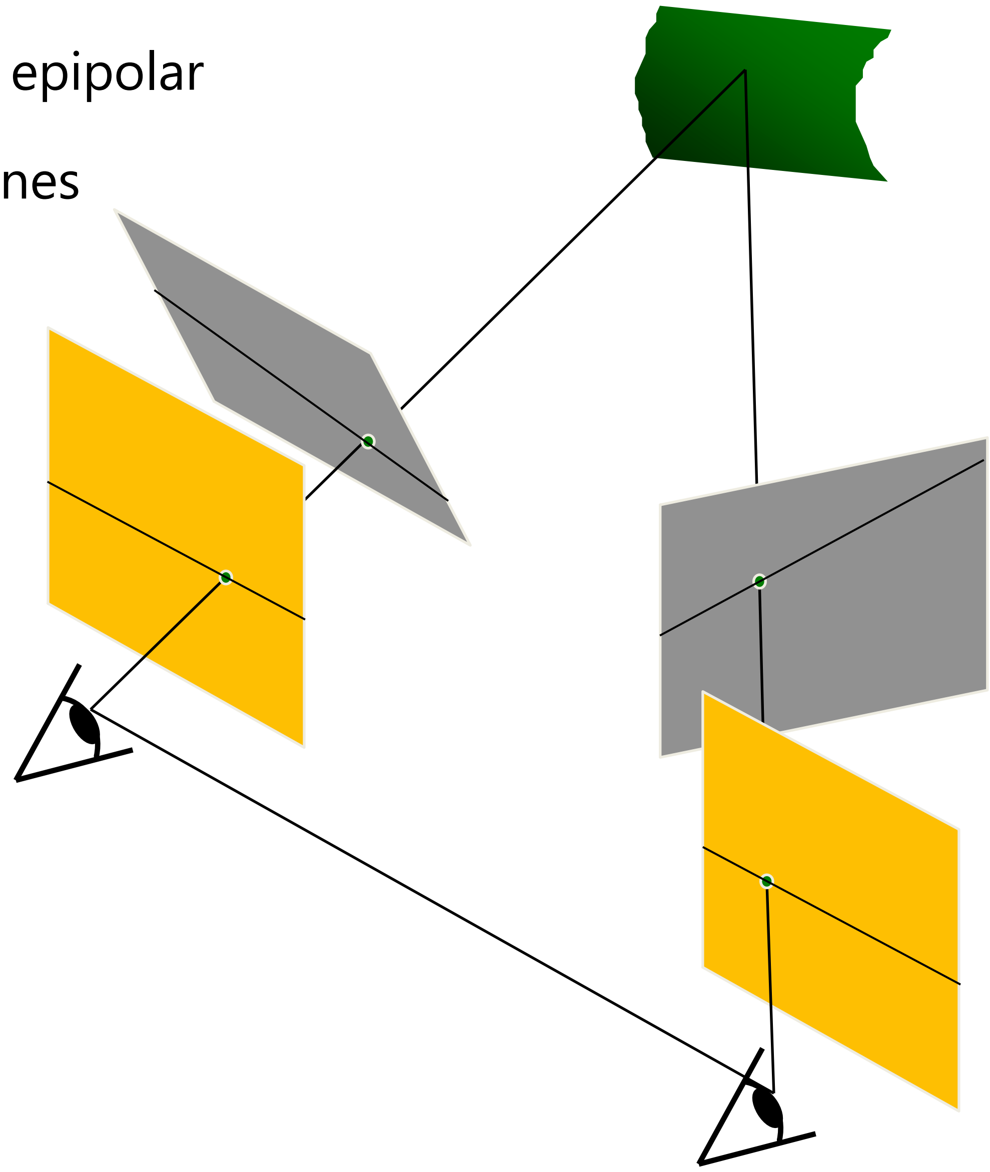
- Then, epipolar lines correspond to scan lines

- Rectification is a procedure to convert images so that the assumptions are satisfied
  - It simplifies algorithms
  - It improves efficiency

[KM Lee, Lecture Notes]

Rectification

- Reproject (warp) images so that epipolar lines are aligned with the scan lines



Rectification

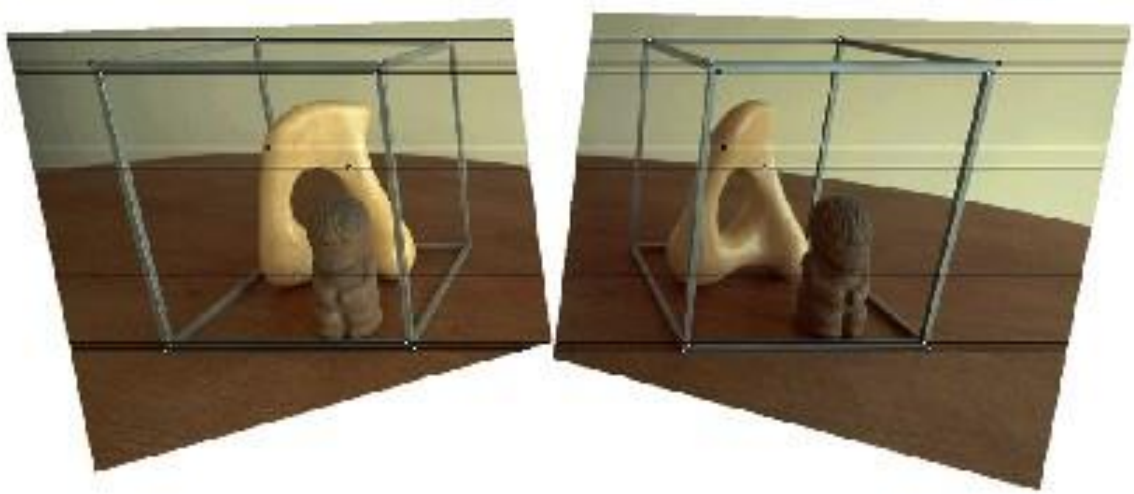
(a) Original image pair overlayed with several epipolar lines.

(b) Image pair transformed by the specialized projective mapping $\mathbf{H}_p$ and $\mathbf{H}'_p$. Note that the epipolar lines are now parallel to each other in each image.

[Loop and Zhang, CVPR'99]

Rectification

(c) Image pair transformed by the similarity $\mathbf{H}_r$ and $\mathbf{H}'_r$. Note that the image pair is now rectified (the epipolar lines are horizontally aligned).



(d) Final image rectification after shearing transform $\mathbf{H}_s$ and $\mathbf{H}'_s$. Note that the image pair remains rectified, but the horizontal distortion is reduced.

[Loop and Zhang, CVPR'99]

Correspondence: What to Match?

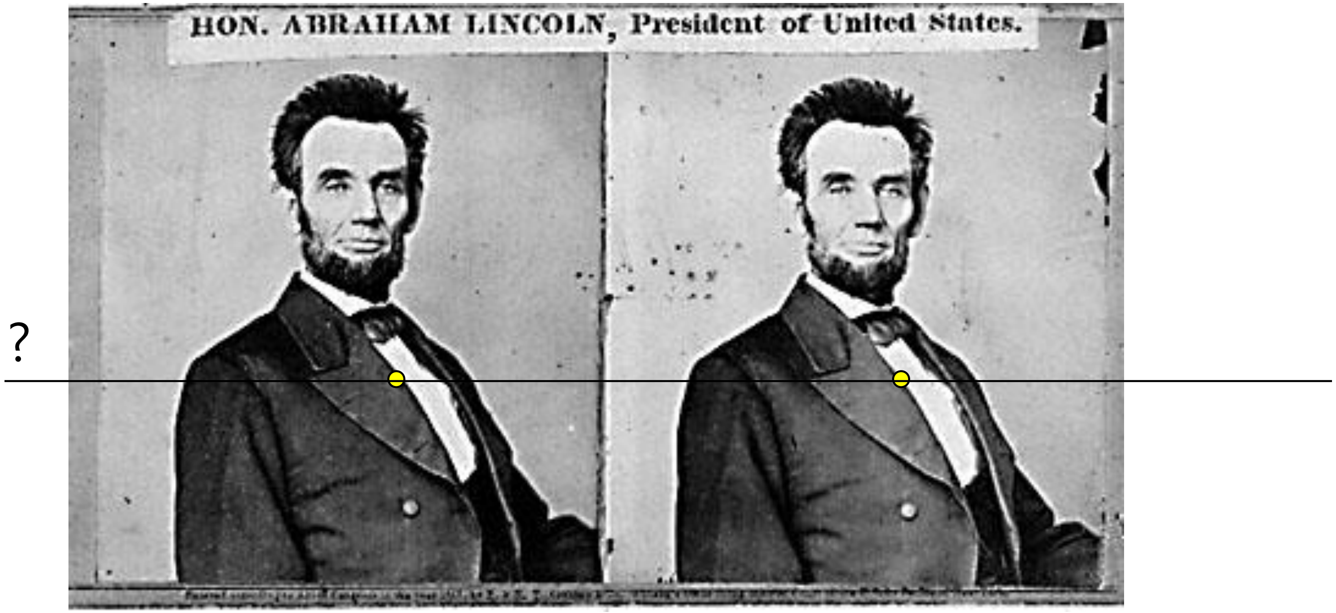
- Objects?
  - More identifiable, but difficult to compute
- Pixels?
  - Easier to handle, but maybe ambiguous
- Edges?
- Collections of pixels (regions)?

Correspondence: Photometric Constraint

- Assume that the same world point has the same intensity in both images.
  - However, it is not true in general
    - Noise
    - Illumination
    - Camera calibration

Pixel Matching

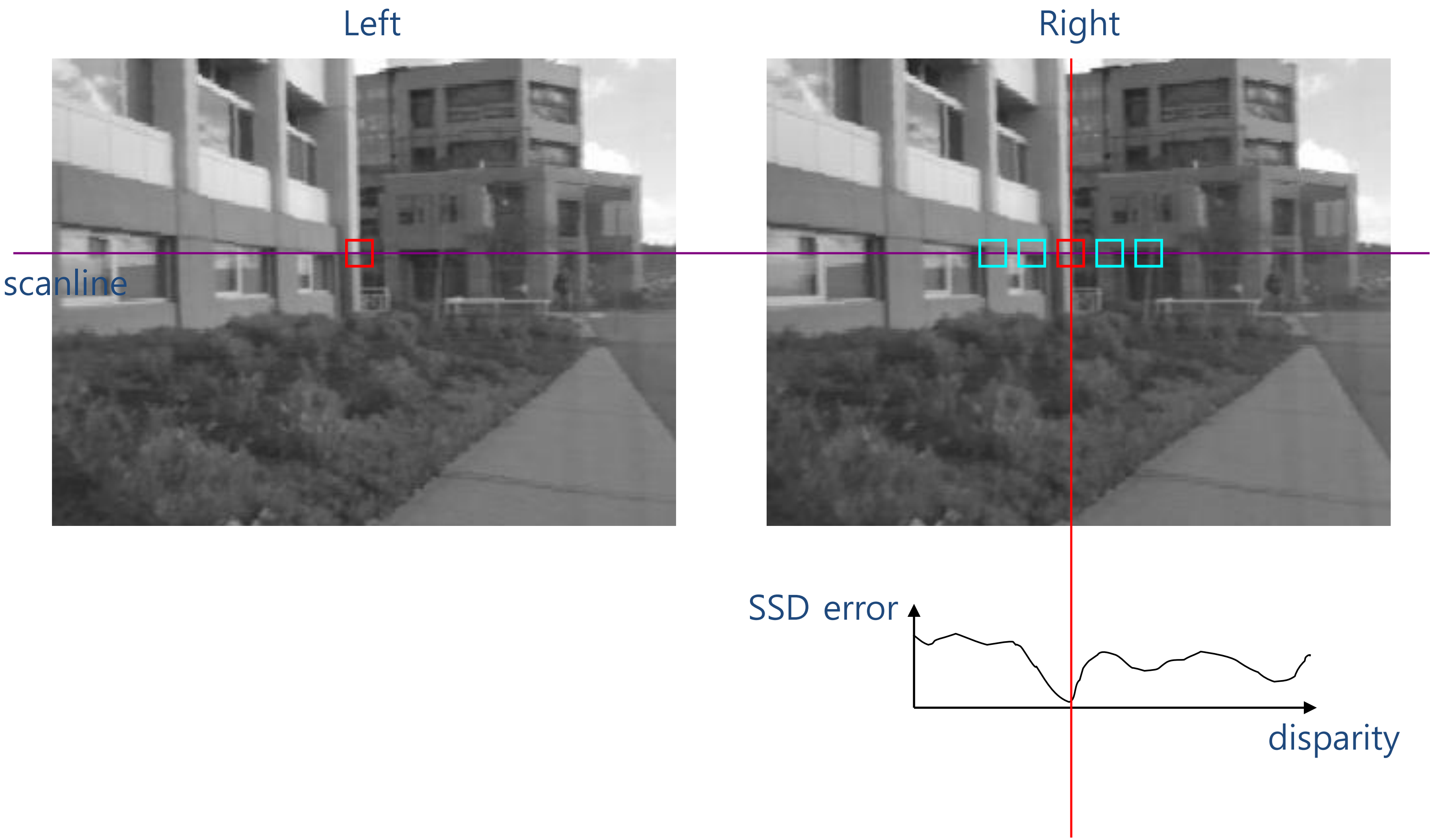
What if ?



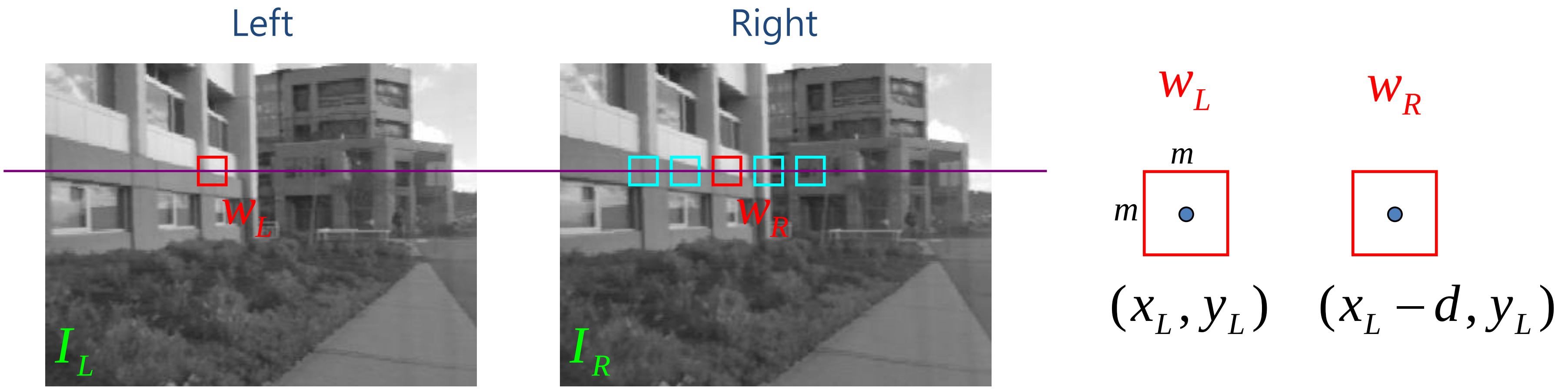
For each scanline , for each pixel in the left image

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

Correspondence Using Window Matching



SSD



- Two blocks $\boldsymbol{w}_L$ and $\boldsymbol{w}_R$
- $SSD = \|\boldsymbol{w}_L - \boldsymbol{w}_R\|^2$

Normalization

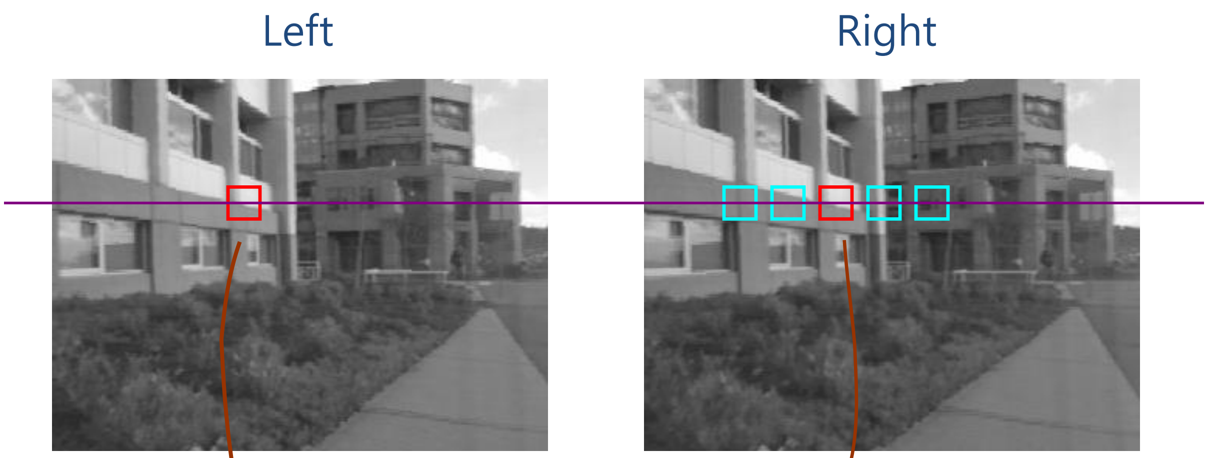
- There can be differences in gain and sensitivity
- Normalize the pixels in each window

$$\tilde{\boldsymbol{w}} = \frac{\boldsymbol{w} - \mu\mathbf{1}}{\|\boldsymbol{w} - \mu\mathbf{1}\|}$$

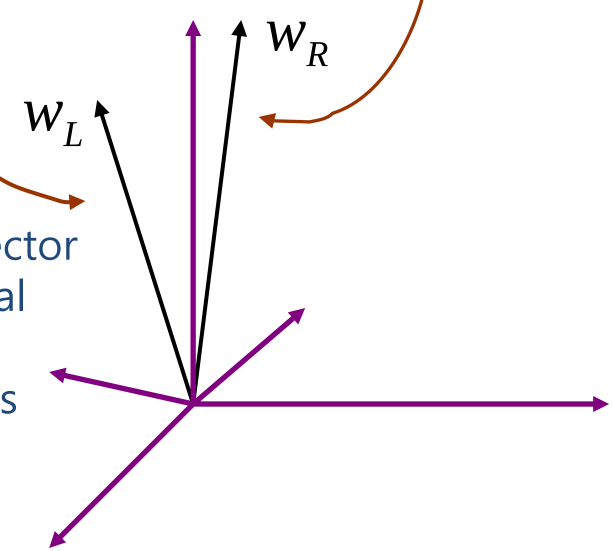
- Minimizing SSD becomes maximizing NCC (normalized cross correlation)

$$\|\tilde{\boldsymbol{w}}_L - \tilde{\boldsymbol{w}}_R\|^2 = 2 - 2\tilde{\boldsymbol{w}}_L \cdot \tilde{\boldsymbol{w}}_R$$

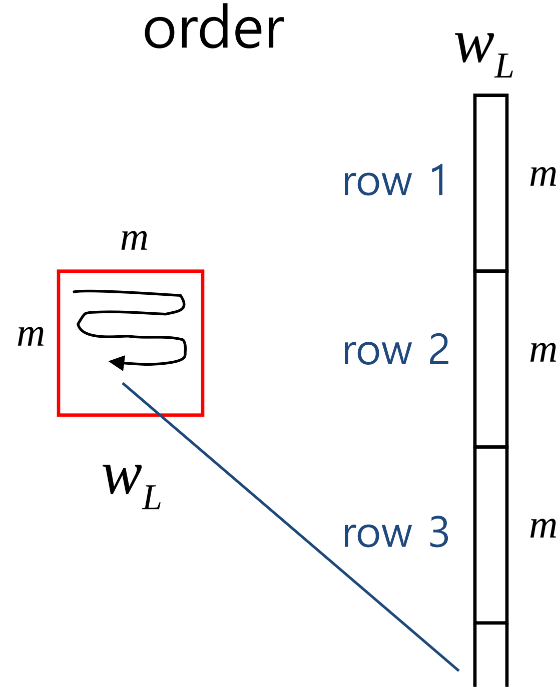
Normalization



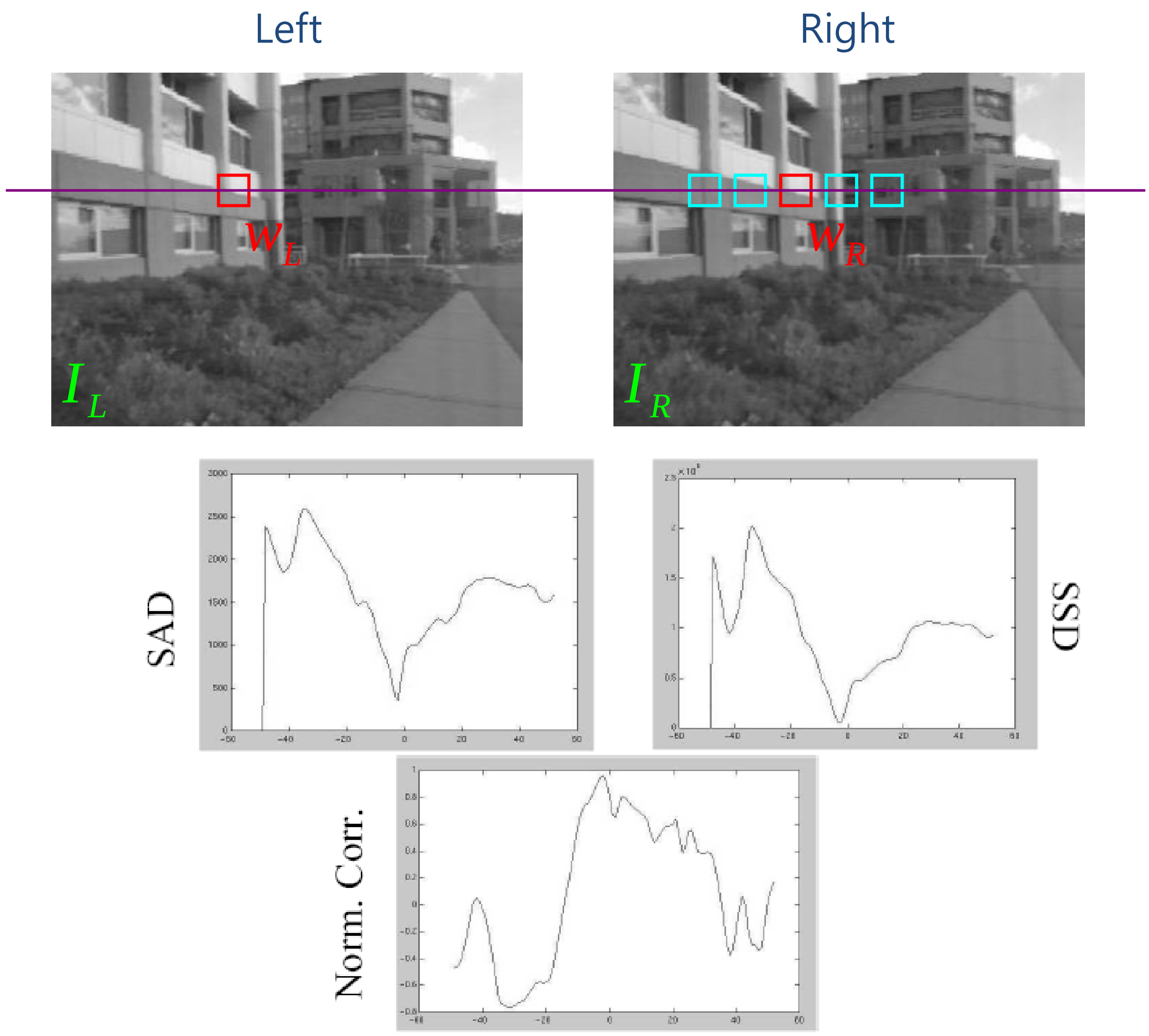
"Unwrap" image to form vector, using raster scan order



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



Distance Metrics



Stereo Results

Images courtesy of Point Grey Research

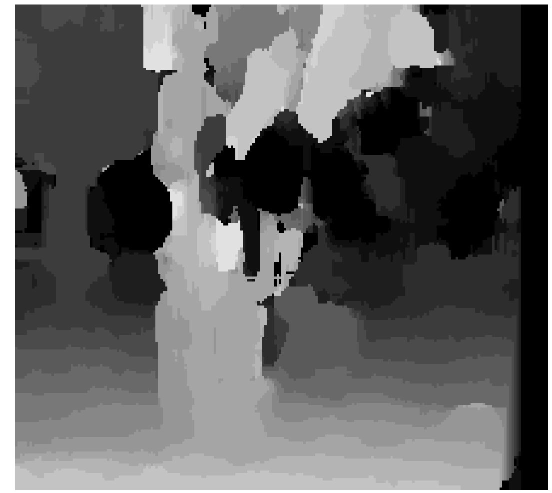
Disparity Map

Problems with Window-Based Matching

- Disparity within the window may not be constant
- Blur across depth discontinuities
- Poor performance in textureless regions
- Erroneous results in occluded regions

Window Size

W = 3



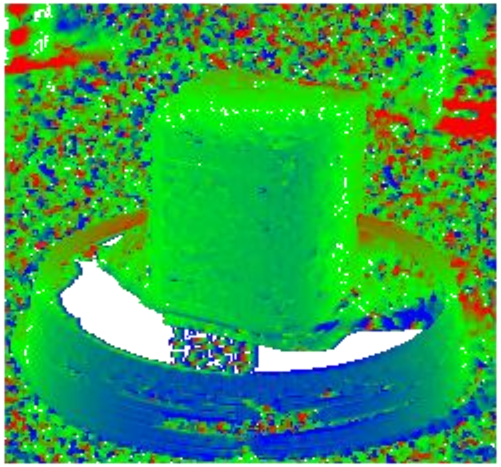
W = 20

- The results depend on the window size
- Some approaches have been developed to use an adaptive window size (try multiple sizes and select best match)

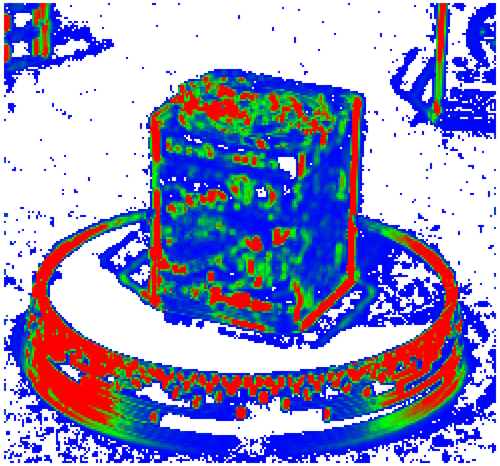
Certainty Modeling

- Compute certainty map from correlations

input



depth map



certainty map

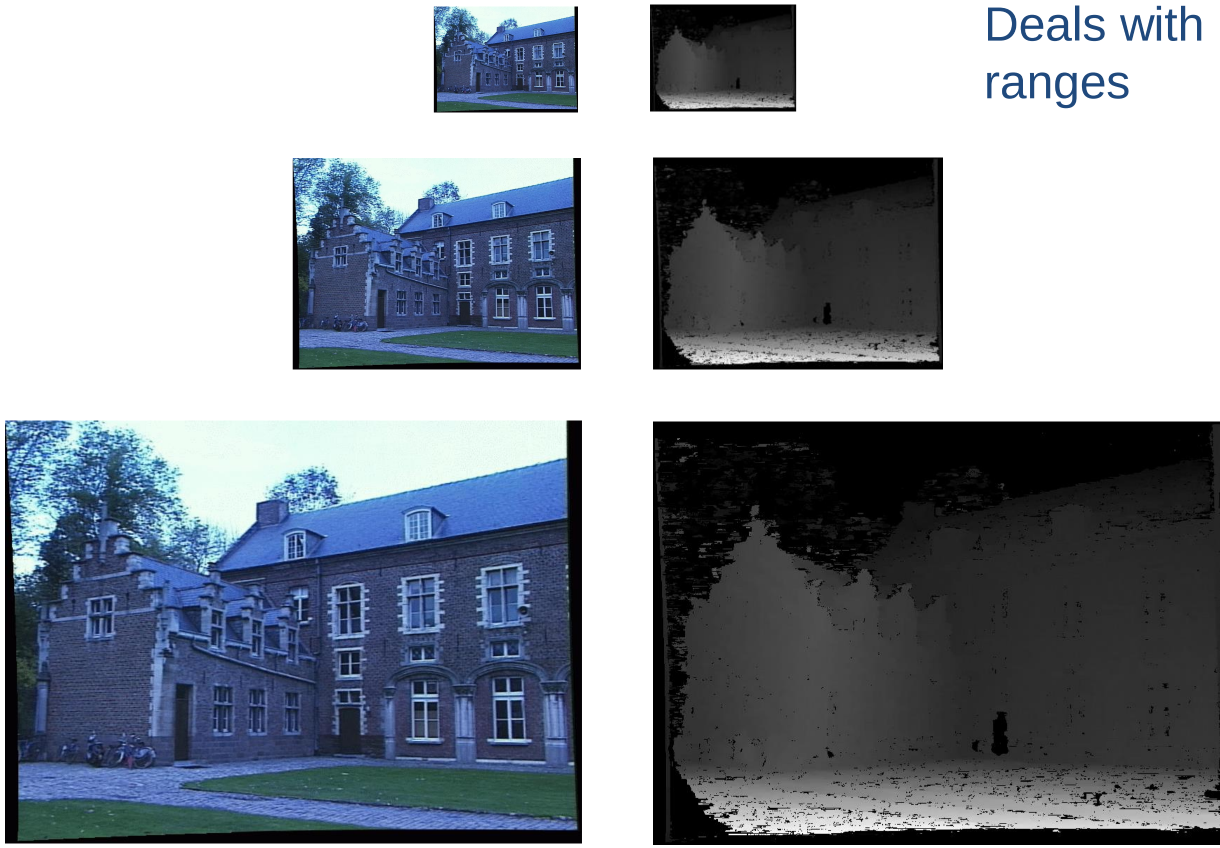
[Szeliski, 1991]

Hierarchical Stereo Matching

Allows faster computation

Deals with large disparity ranges

Downsampling (Gaussian pyramid)



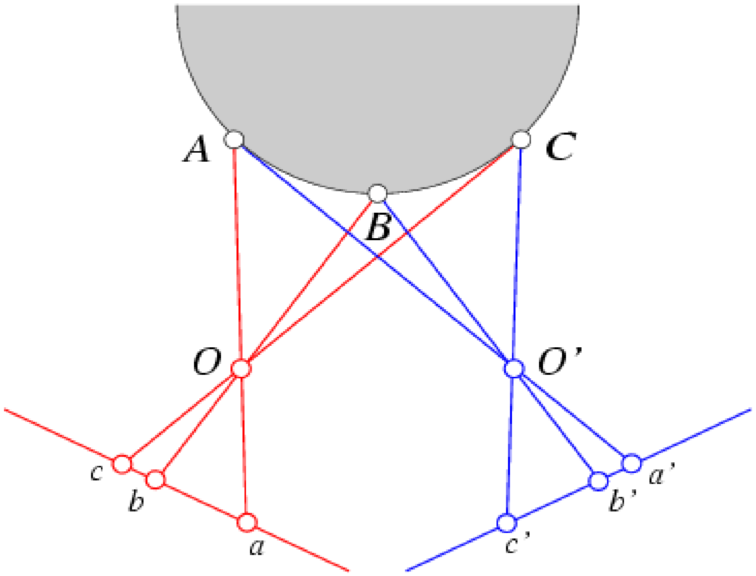
Disparity propagation

(Falkenhagen´97;Van Meerbergen,Vergauwen,Pollefeys,VanGool IJCV'02)

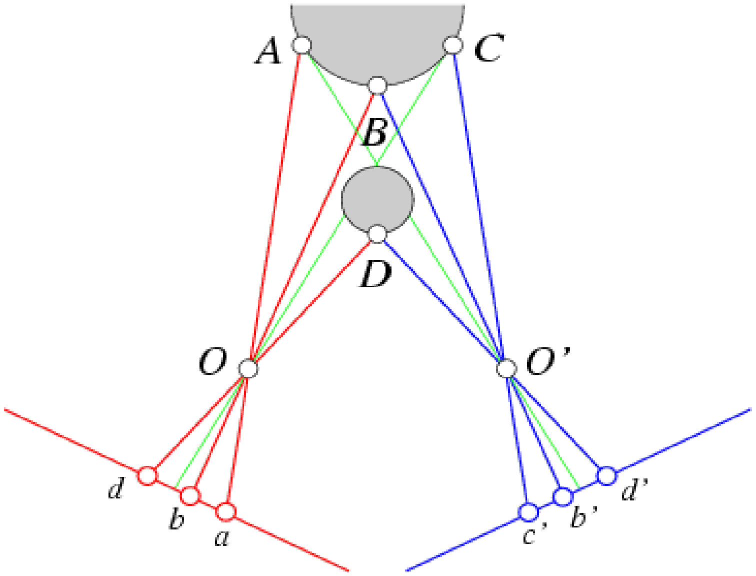
Stereo Matching Using Dynamic Programming

Ordering Constraint

- Points on the epipolar lines appear in the same order
- It may not be true in some cases, but can be assumed for most cases
- This is the basic assumption of the stereo matching using dynamic programming

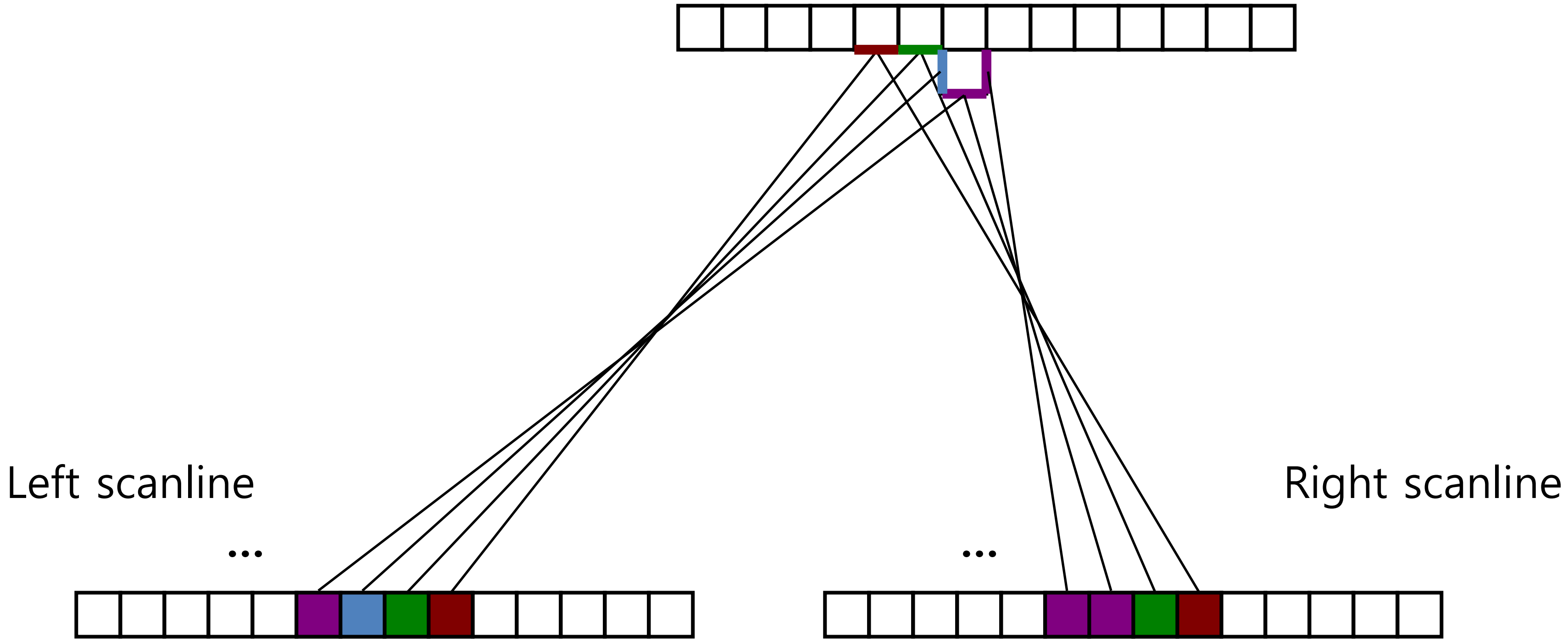


Ordering constraint…

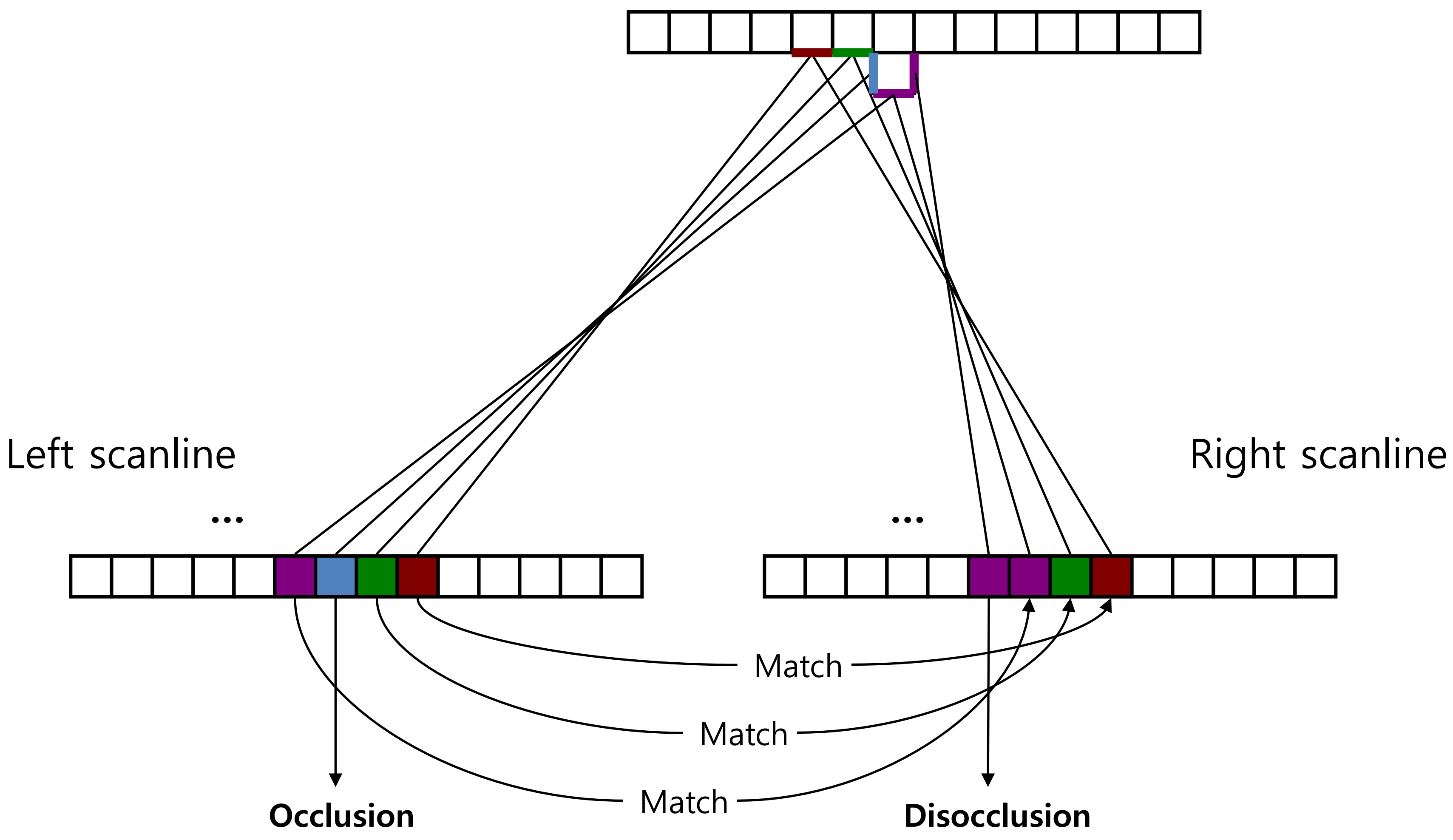


…and its failure

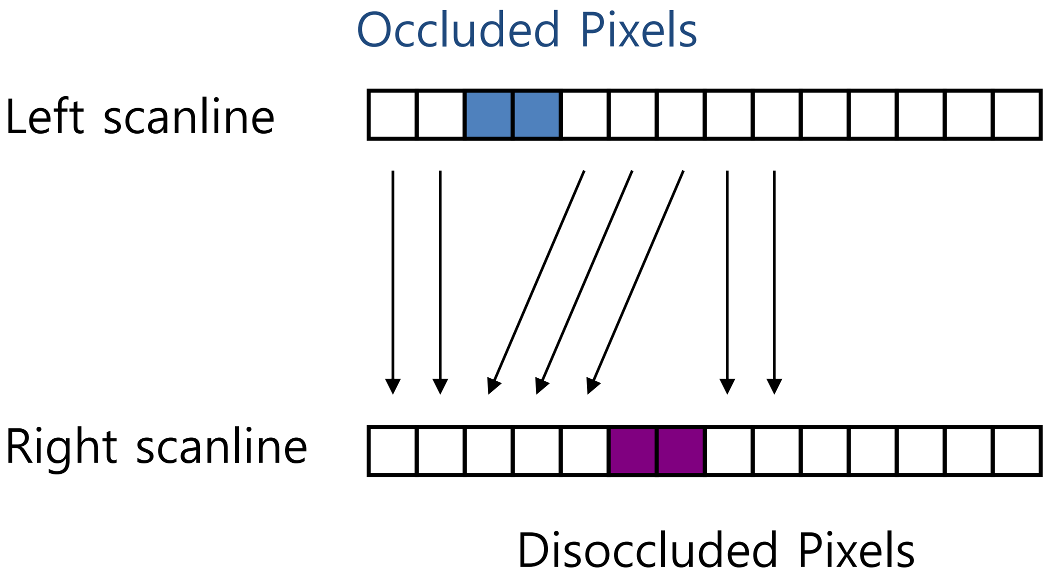
Occlusion and Disocclusion



Occlusion and Disocclusion



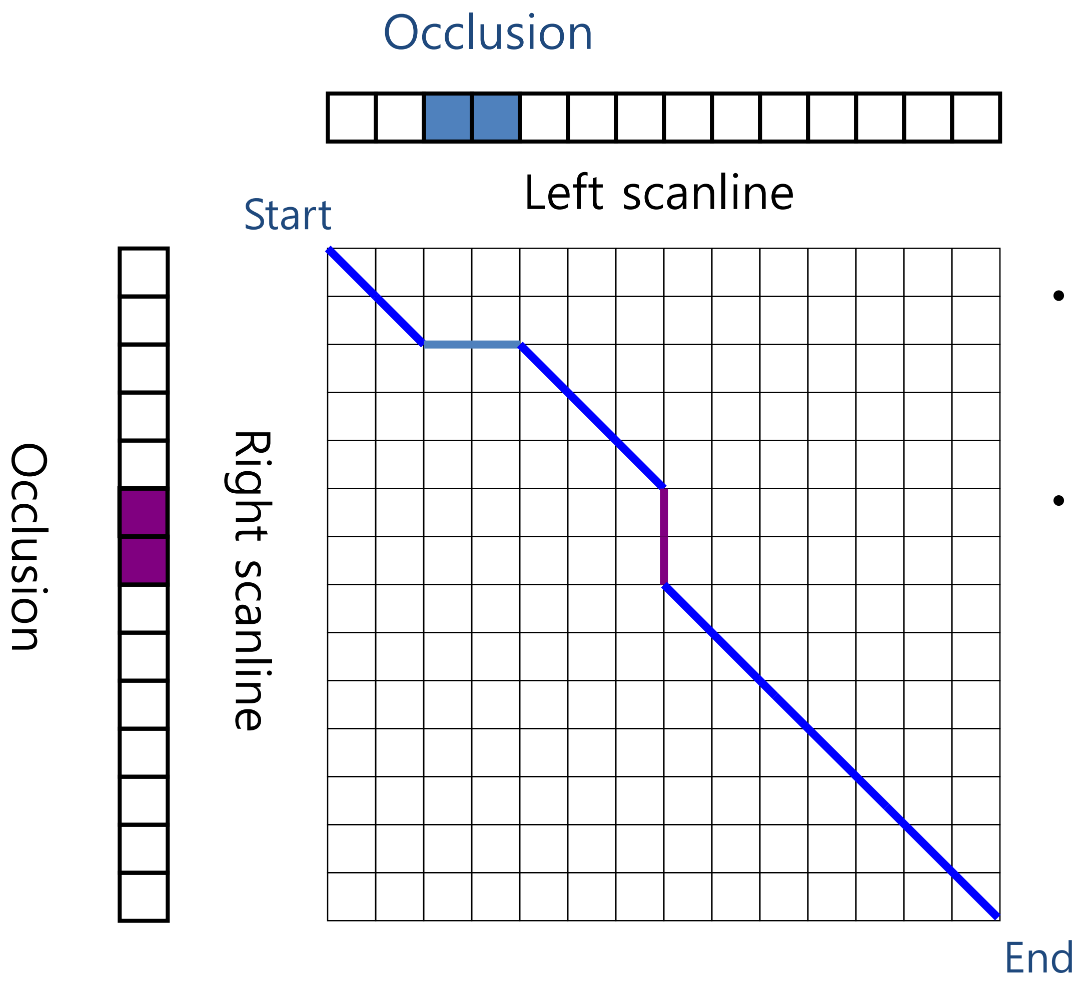
Search over Correspondences



Three cases:

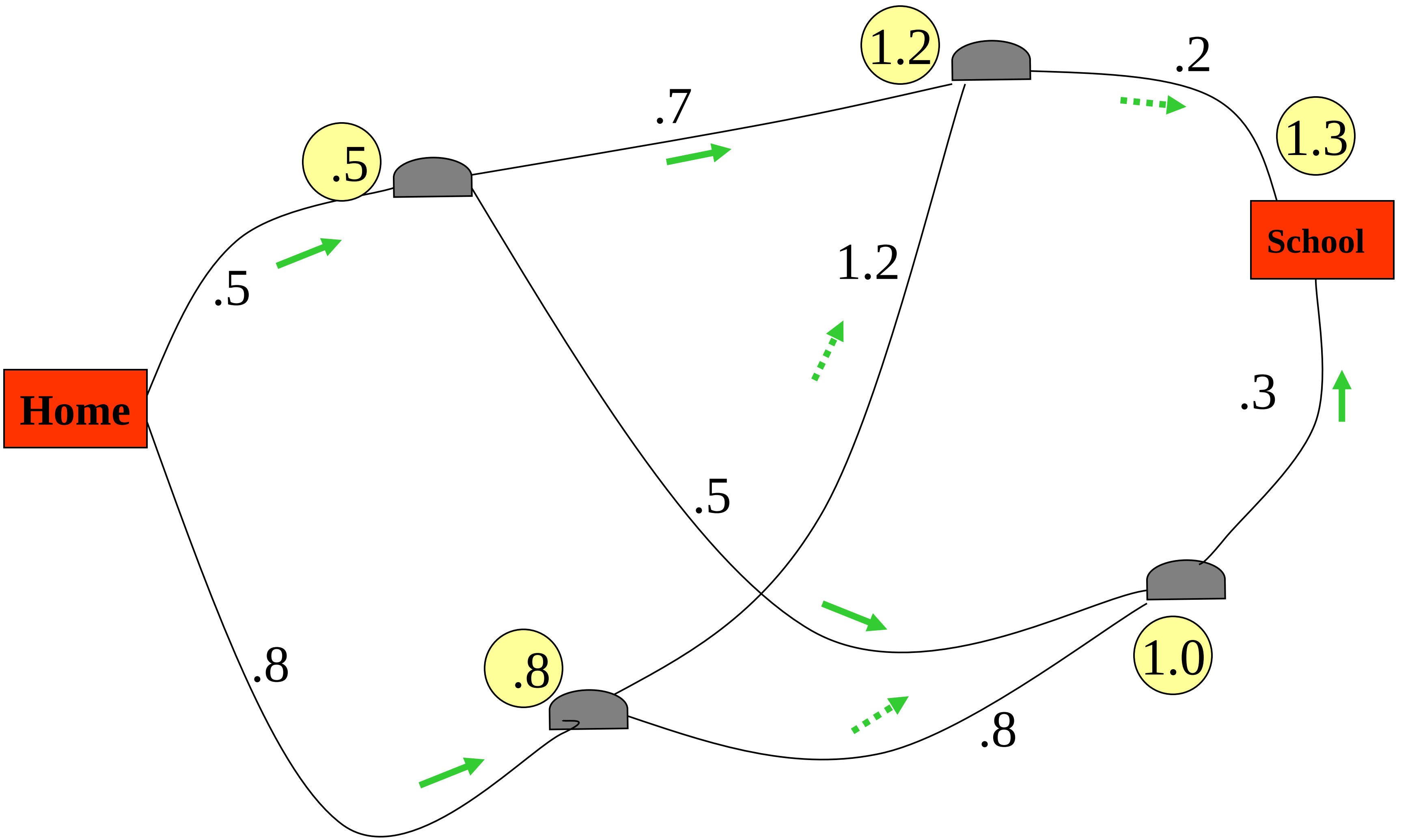
- Sequential – add cost of match (small if intensities agree)
- Occluded – add cost of no match (large cost)
- Disoccluded – add cost of no match (large cost)

Dynamic Programming Approach

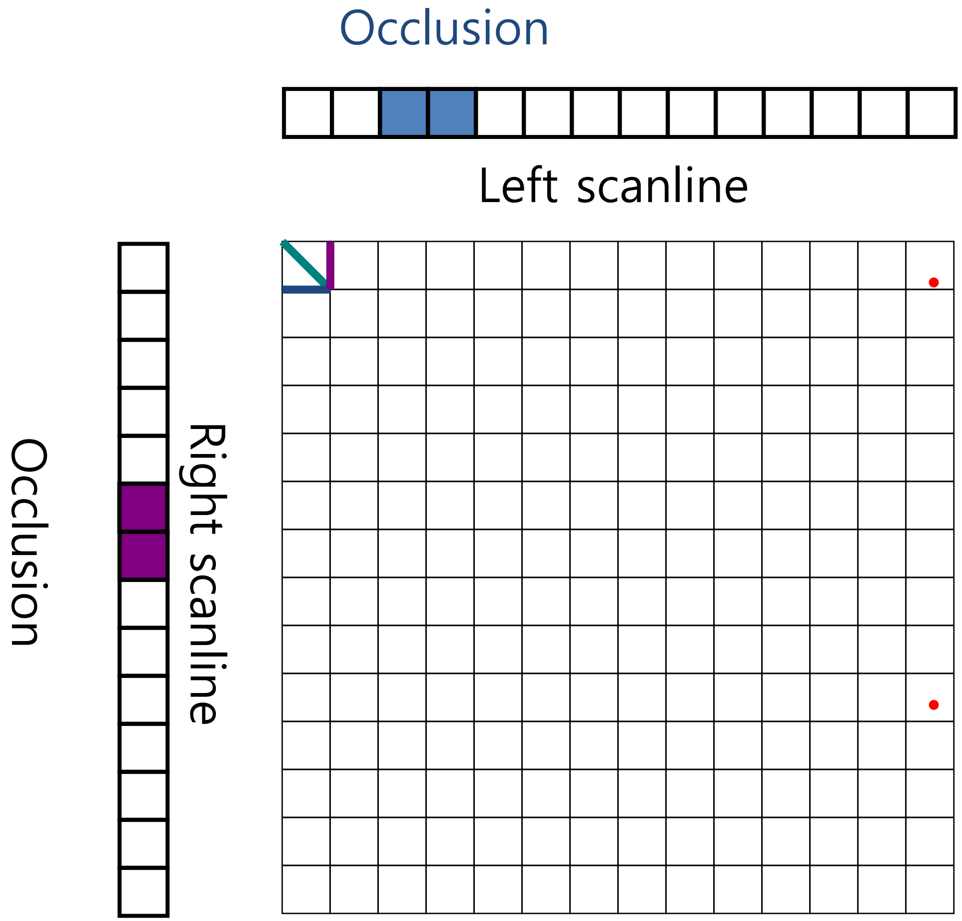


- Dynamic programming yields the optimal path, satisfying the ordering constraint
- Every segment on each scan line will be labeled as either matching or occlusion
  - Diagonal arc: matching
  - Horizontal arc: left occlusion
  - Vertical arc: right occlusion

Bellman’s Optimality Principle



Dynamic Programming Approach



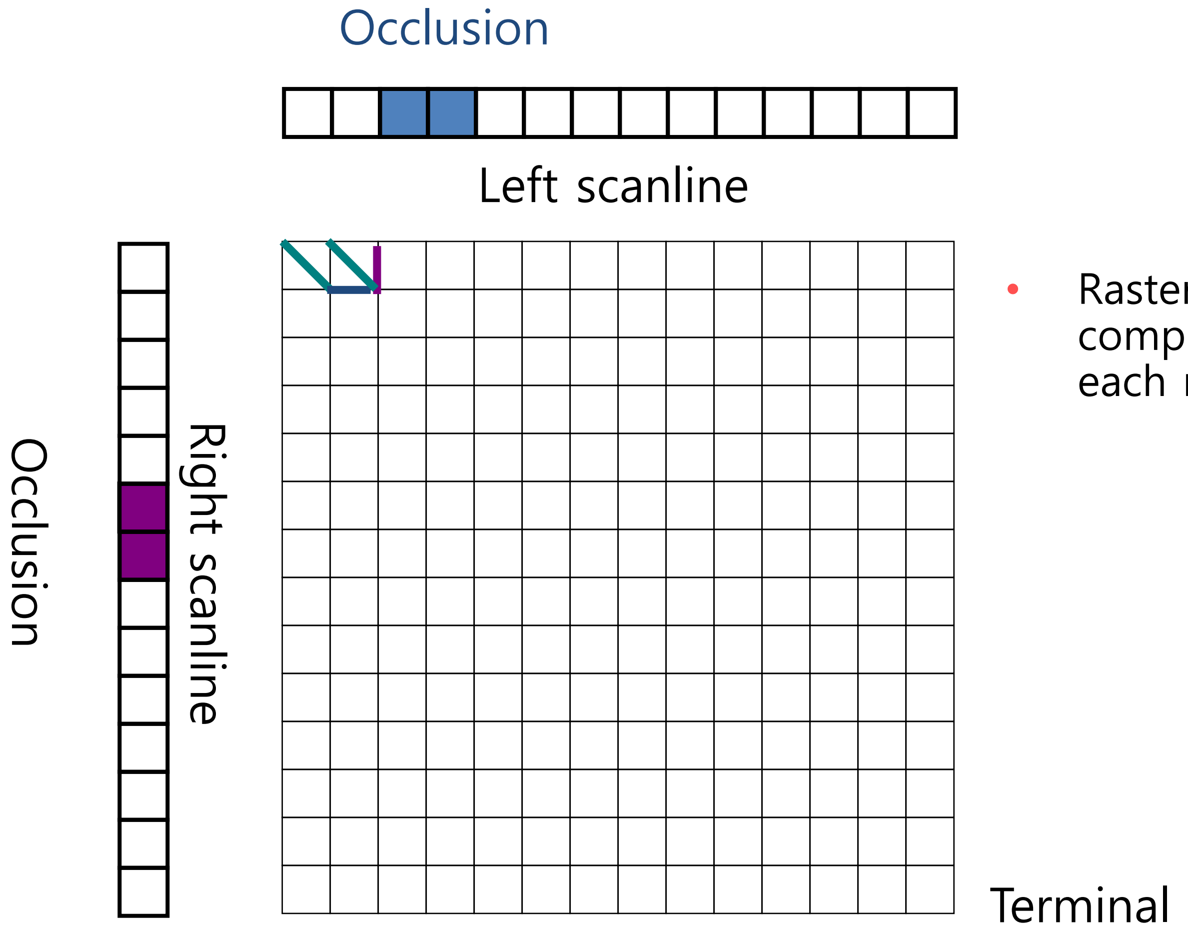
Cost function $C(i,j)$: the optimal cost up to node (i,j).

$C(i,j)$ = min{
    $C(i-1,j-1)$ +matching cost,
    $C(i-1,j)$ +left occlusion penalty,
    $C(i,j-1)$ +right occlusion penalty
}

While computing the cost, we record how node (i,j) is connected to one of the three candidates

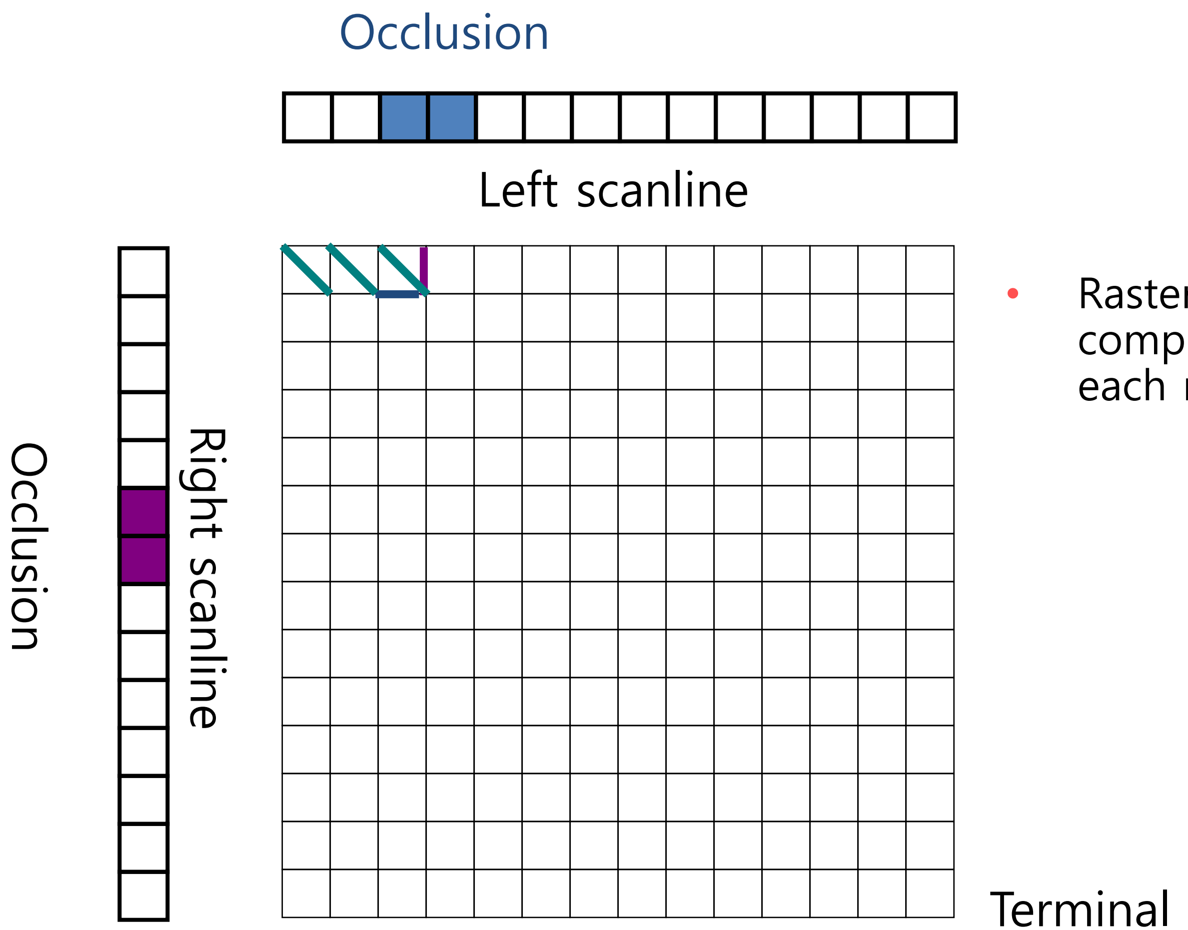
Terminal

Dynamic Programming Approach



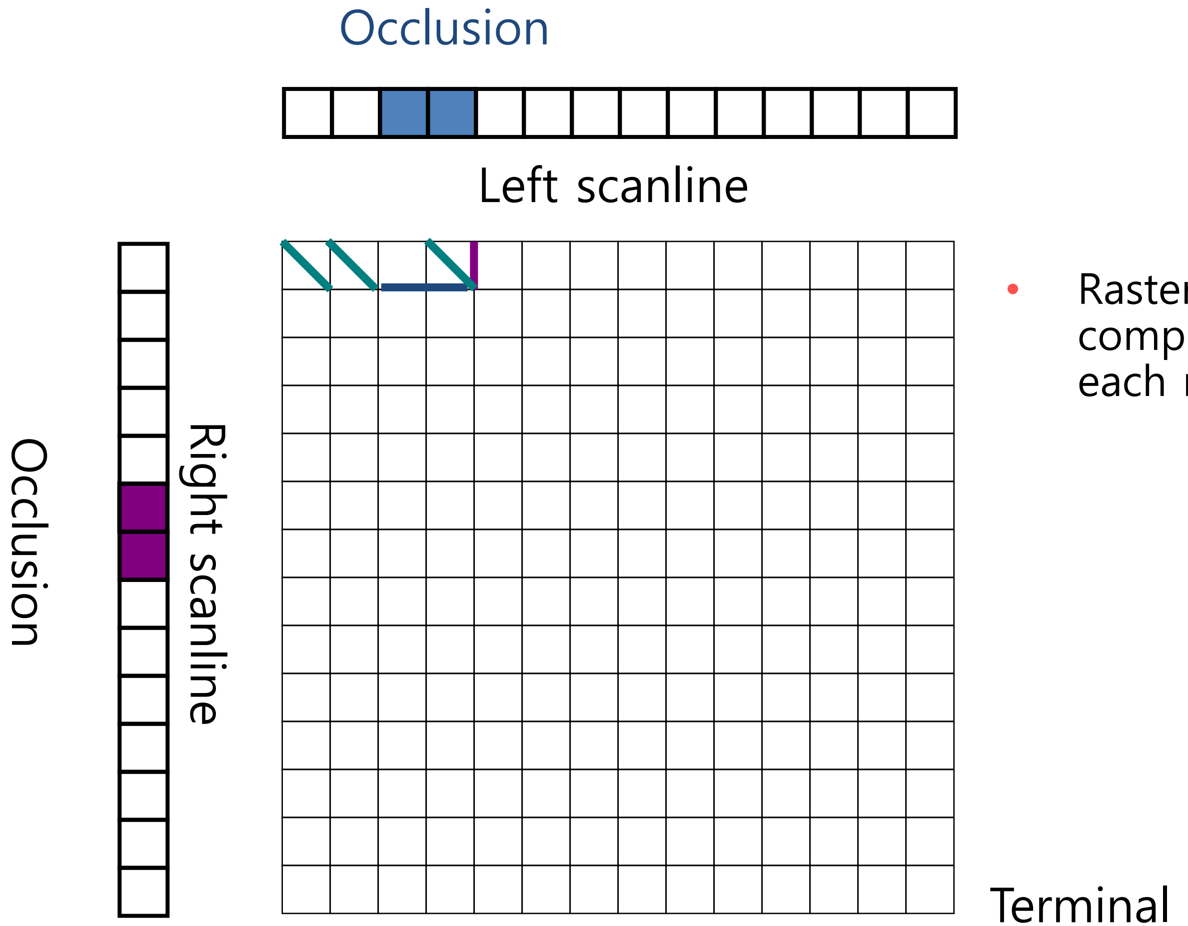
- Raster-scan the nodes, computing optimal cost for each node.

Dynamic Programming Approach



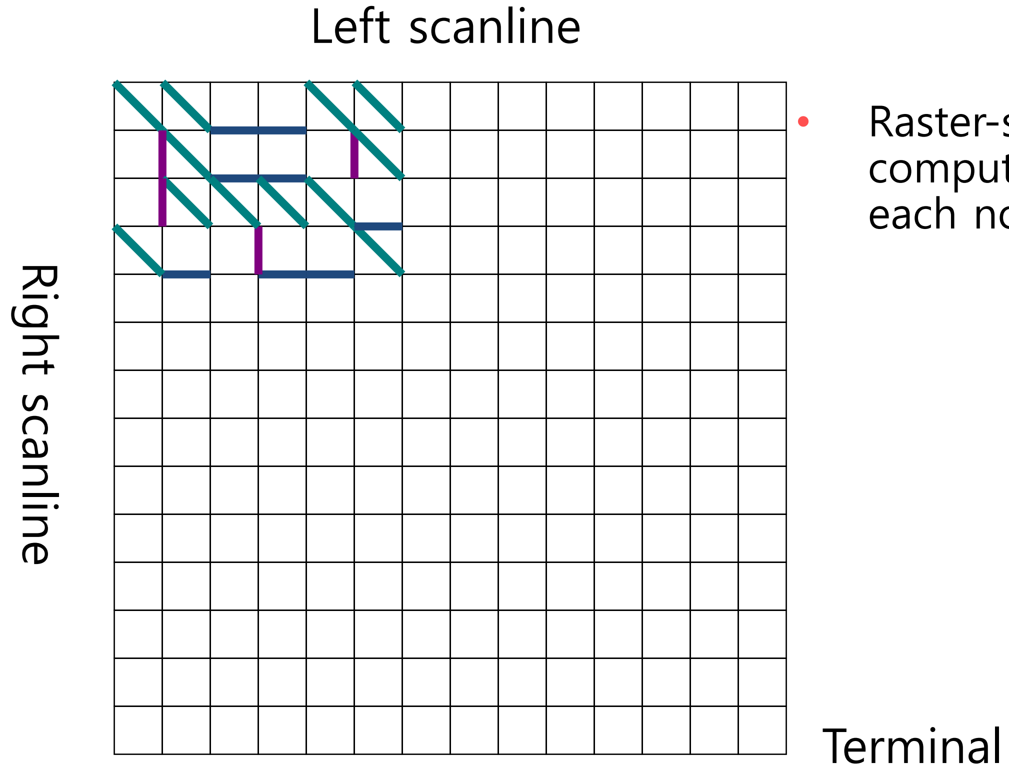
- Raster-scan the nodes, computing optimal cost for each node.

Dynamic Programming Approach



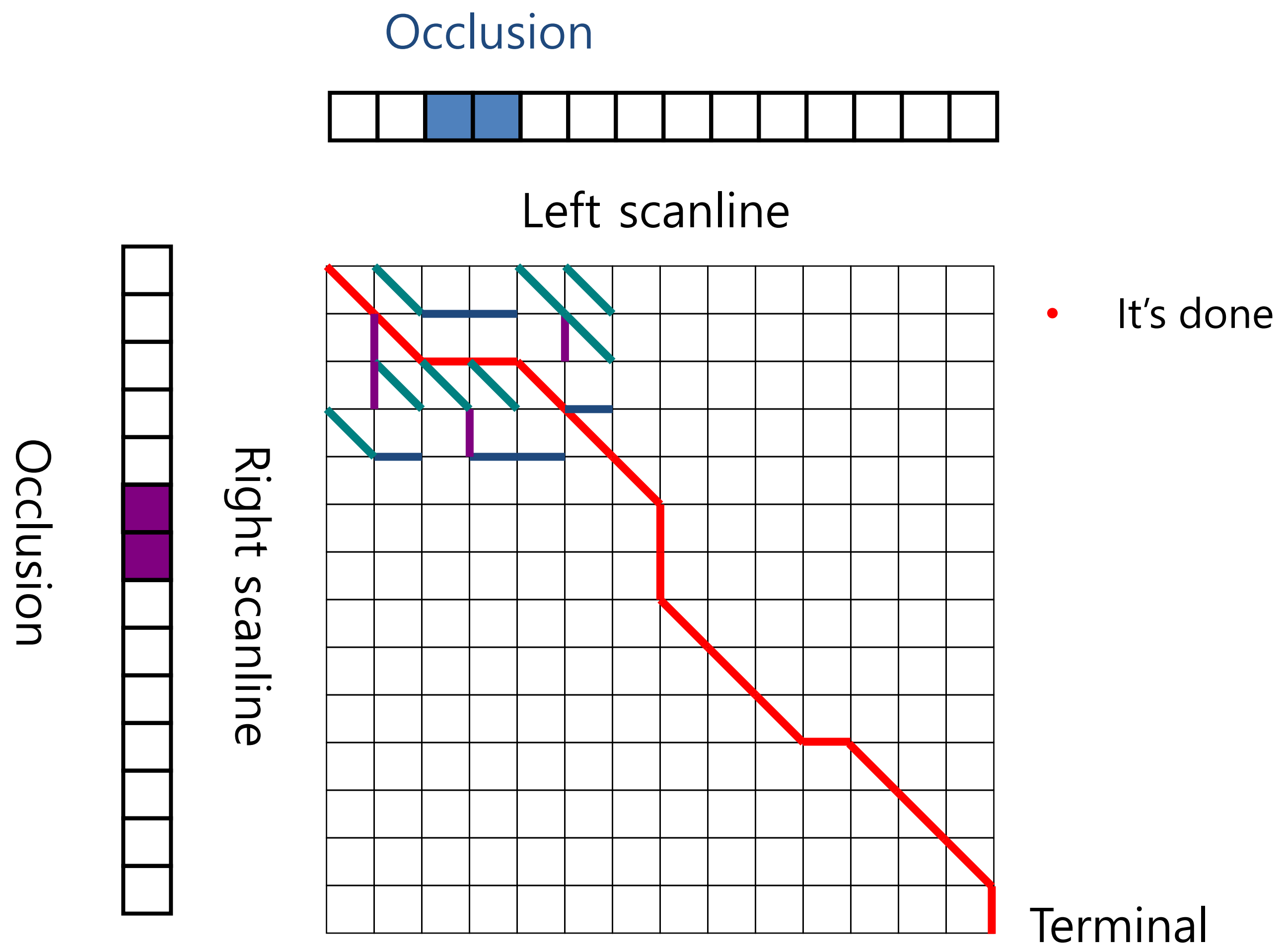
- Raster-scan the nodes, computing optimal cost for each node.

Dynamic Programming Approach



- Raster-scan the nodes, computing optimal cost for each node.

Dynamic Programming Approach



Dynamic Programming Approach

- It treats each scan line independently and thus may generate streaking artifacts
- An error can propagate