Stereo

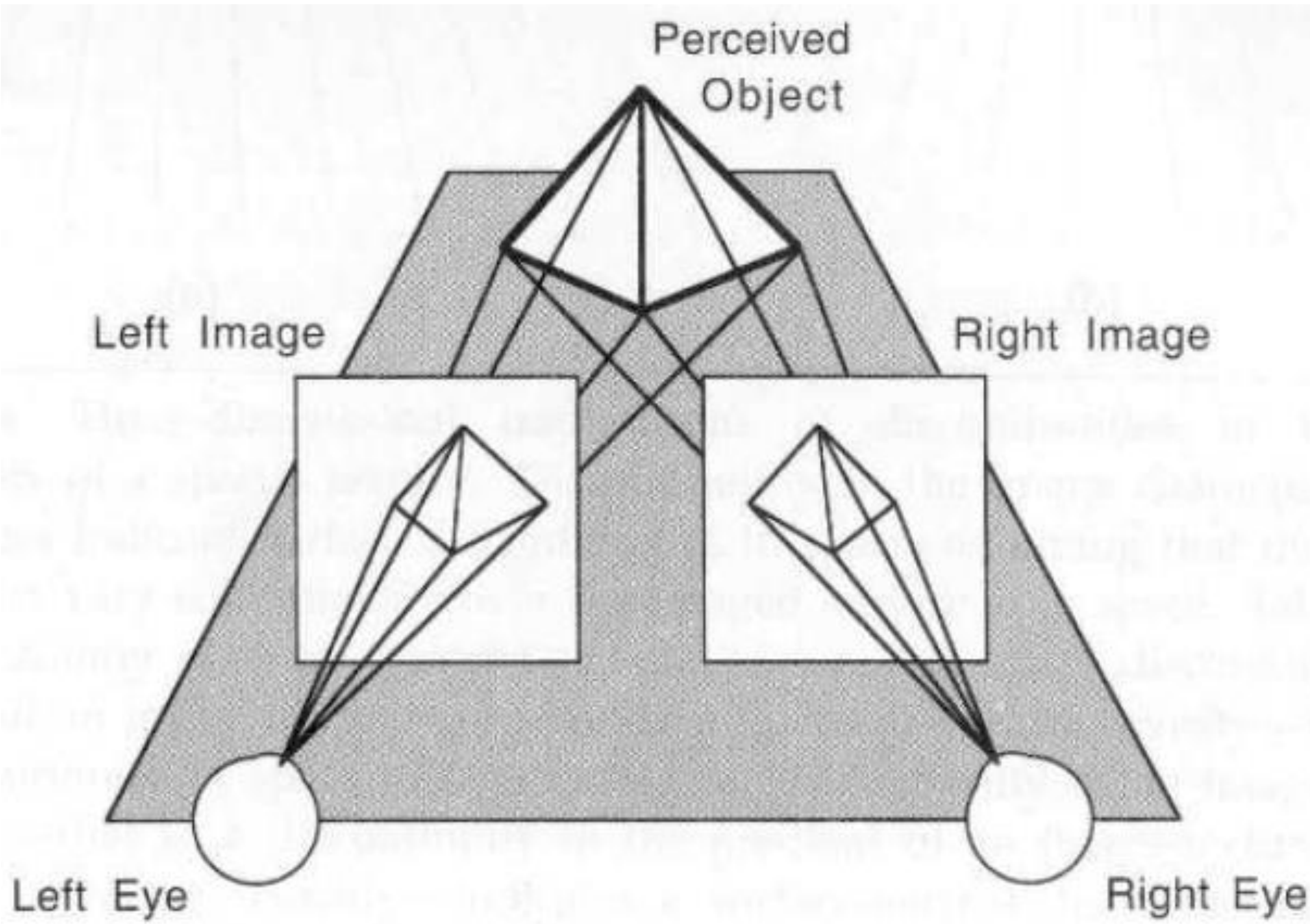
COS 429

Princeton University

Binocular Stereo Reconstruction

Binocular Stereo Reconstruction

Recover dense 3D structure of a scene using two images from different viewpoints



Bebis

Binocular Stereo Reconstruction

Recover dense 3D structure of a scene using two images from different viewpoints

image 1

image 2

Dense depth map

Applications?

Applications

Scene modeling

Segmentation

Human-computer interaction

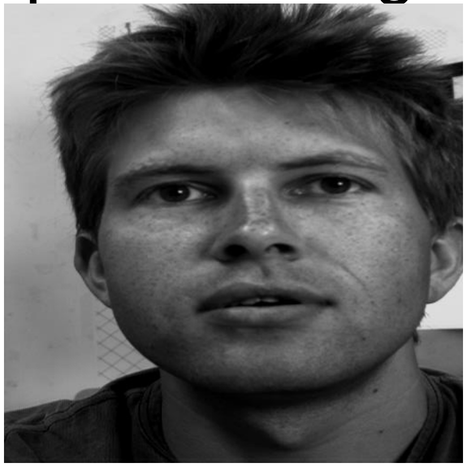
Autonomous driving

View interpolation

etc.

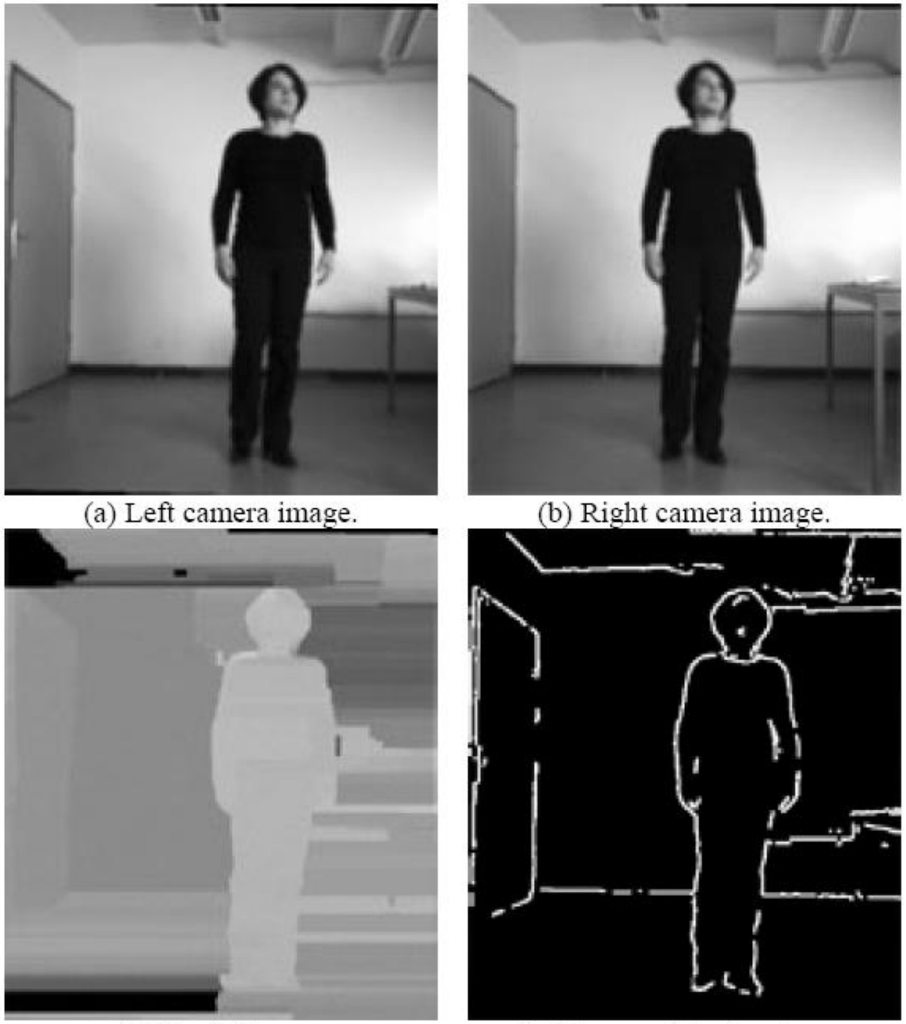
Scene Modeling

From a pair of images to a 3D head model



[Frederic Deverney, INRIA]

Segmentation



(a) Left camera image. (b) Right camera image.

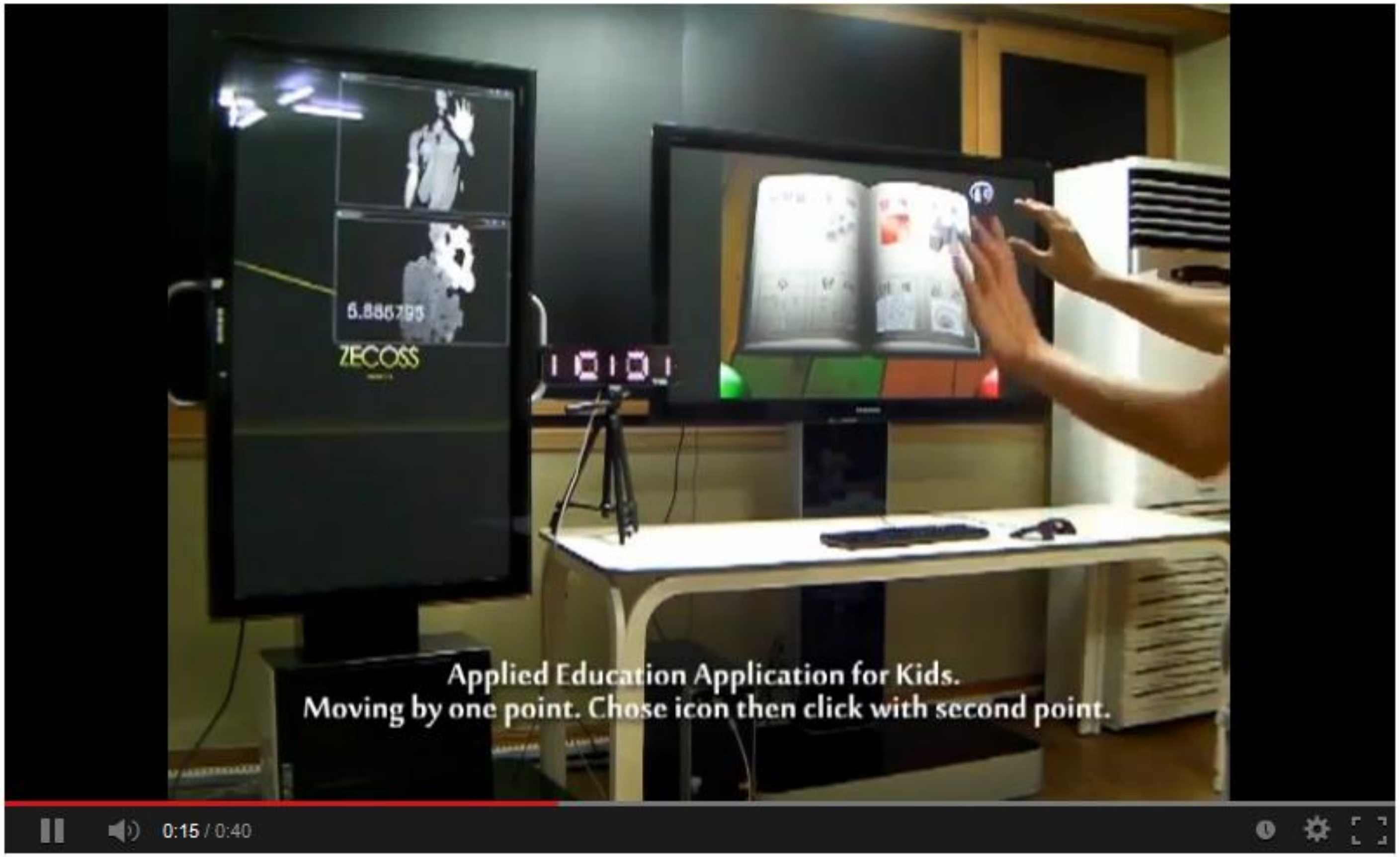
(c) Depth image. (d) Edge combination image.

Figure 3 Stereo video frames with computed depth map and edge combination result.

Edges in disparity in conjunction with image edges enhances contours found

Danijela Markovic and Margrit Gelautz, Interactive Media Systems Group, Vienna University of Technology

Human-Computer Interaction



http://www.youtube.com/watch?v=Q1NE_LIg9pY

Autonomous Driving

Autonomous Driving

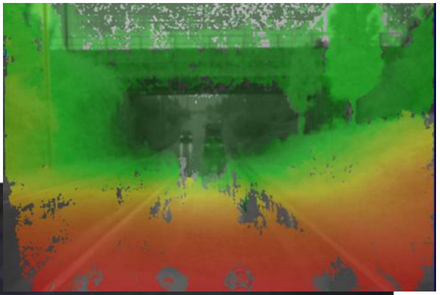
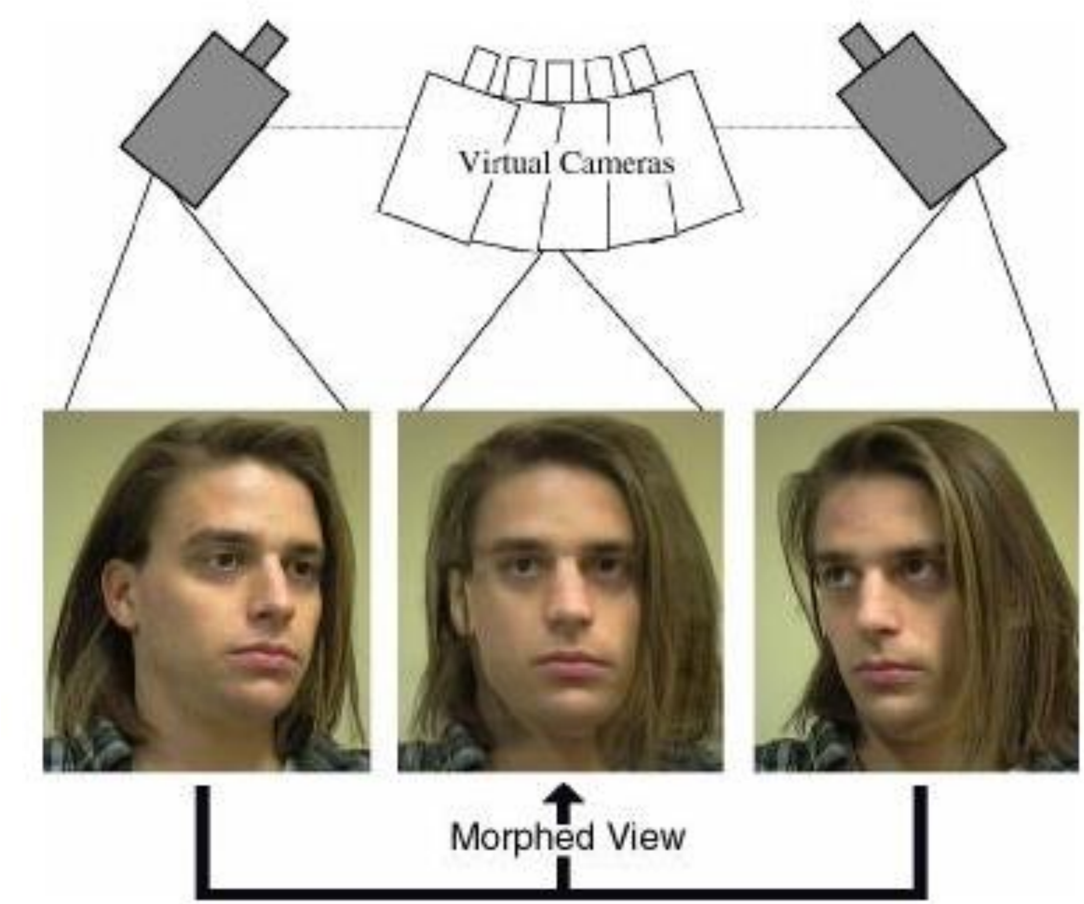


Figure 1: Mercedes-Benz S-class vehicle with stereo camera system behind the wind shield.

Franke et al., “How Cars Learned to See”

View Interpolation

Given two images with correspondences, create novel image from in-between viewpoints



[Seitz & Dyer, SIGGRAPH'96]

View Interpolation

Given two images with correspondences, create novel image from in-between viewpoints

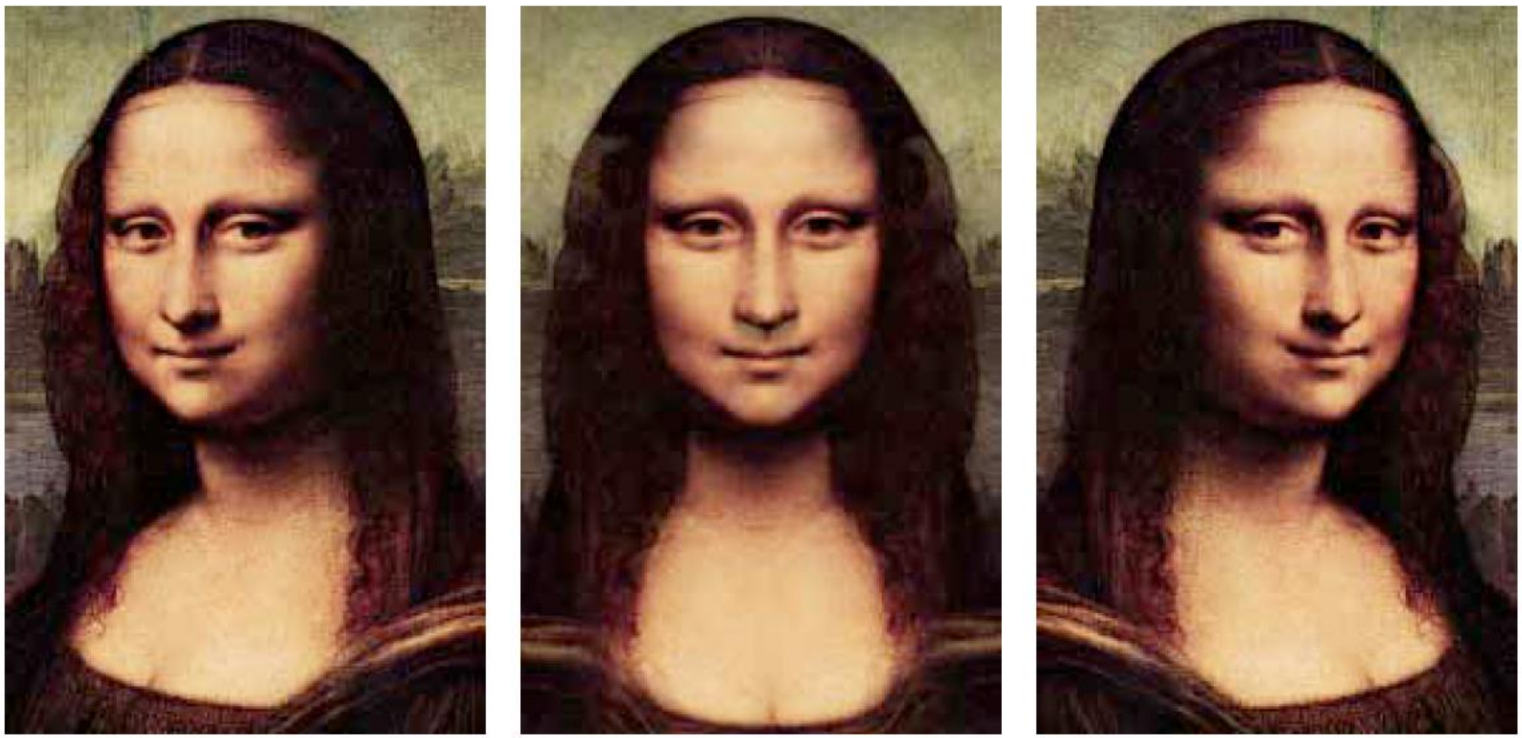


Figure 9: Mona Lisa View Morph. Morphed view (center) is halfway between original image (left) and it's reflection (right).

[Seitz & Dyer, SIGGRAPH'96]

Problem

What are the key steps of a stereo algorithm?

image 1

image 2

Dense depth map

Three Steps

(1) Camera calibration

(2) Dense pixel correspondence

(3) Depth estimation

Three Steps

(1) Camera calibration

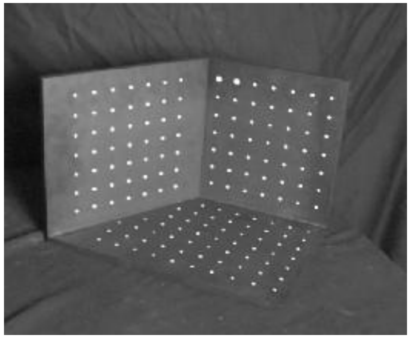
(2) Dense pixel correspondence

(3) Depth estimation

Review: Camera Calibration

Given a (pair of) image(s), compute intrinsic and extrinsic camera parameters

- We talked about calibration before break
- Use vanishing points, sparse correspondences, etc.



Scale to pixel size

Camera orientation

Translate to image center

Perspective projection

Camera location

3D point (homogeneous coords)

Then unwarp radial distortion

Then perform homogeneous divide, and get (u,v) coords

$$\mathbf{T}_{img}\mathbf{S}_{pix}\mathbf{P}_{cam}\mathbf{R}_{cam}\mathbf{T}_{cam}\vec{x}$$

Three Steps

(1) Camera calibration

(2) Dense pixel correspondence ←

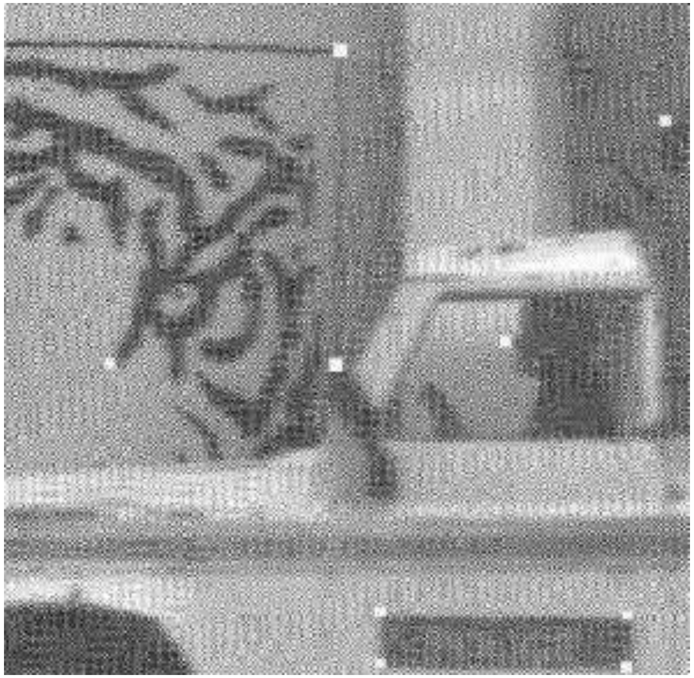
(3) Depth estimation

Dense Pixel Correspondence

Given two calibrated cameras, find a dense set of pixel pairs that correspond to the same 3D point

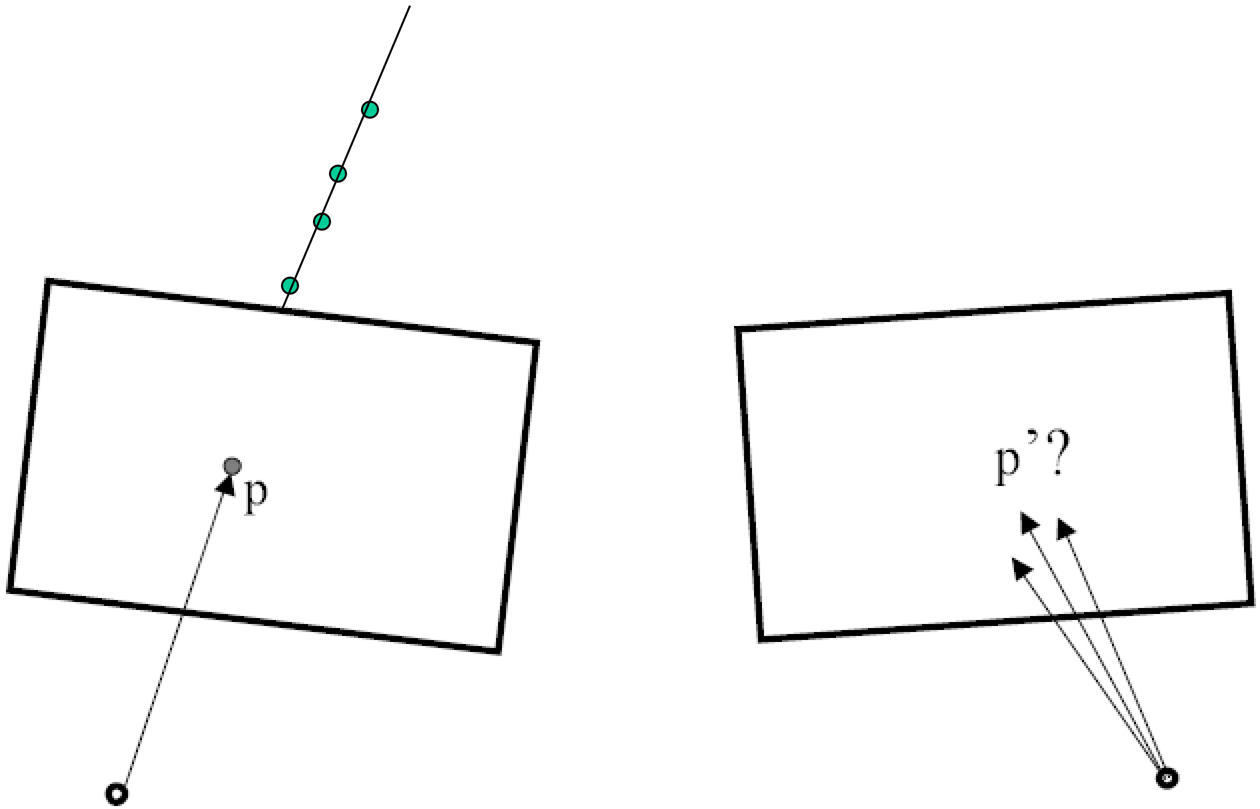
left camera

right camera



Bebis

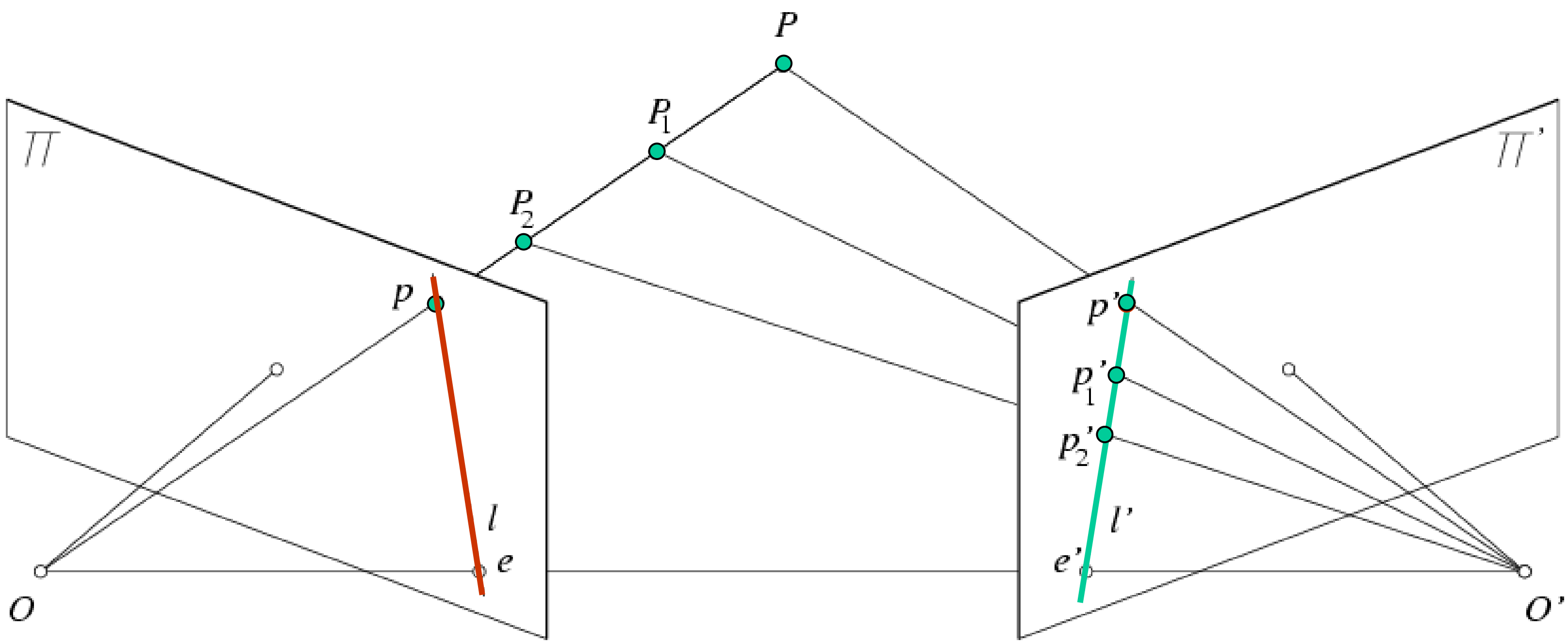
Dense Pixel Correspondence



Given p in left image, where can corresponding point p' be?

Kristen Grauman

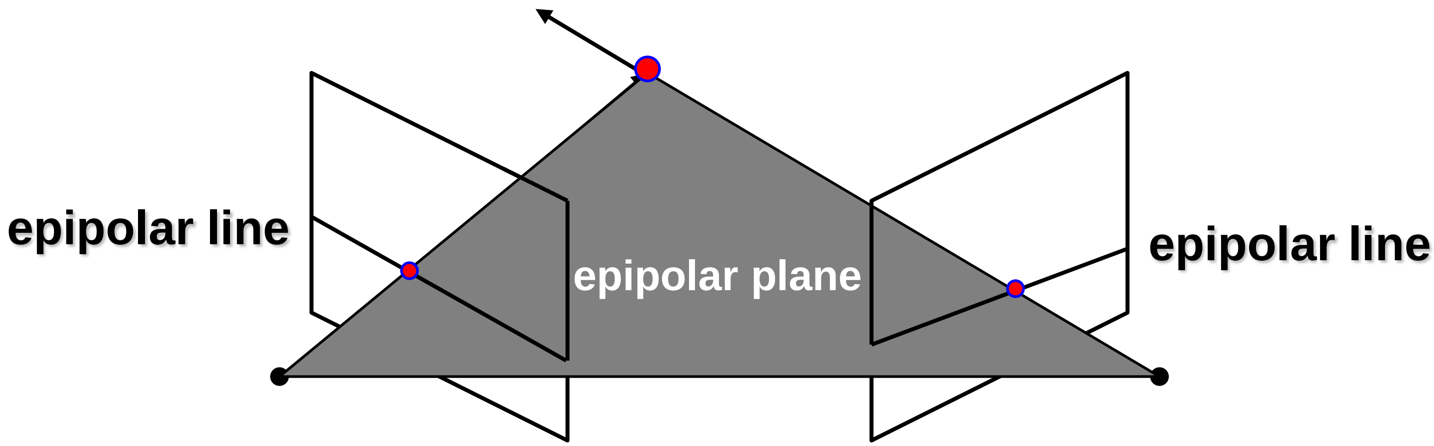
Review: Epipolar constraint



Geometry of two views constrains where the corresponding pixel for some image point in the first view must occur in the second view: it must be on the line carved out by a plane connecting the world point and optical centers.

Kristen Grauman

Review: 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

Review: Epipolar constraint

The epipolar constraint is particularly convenient if the images are “rectified”

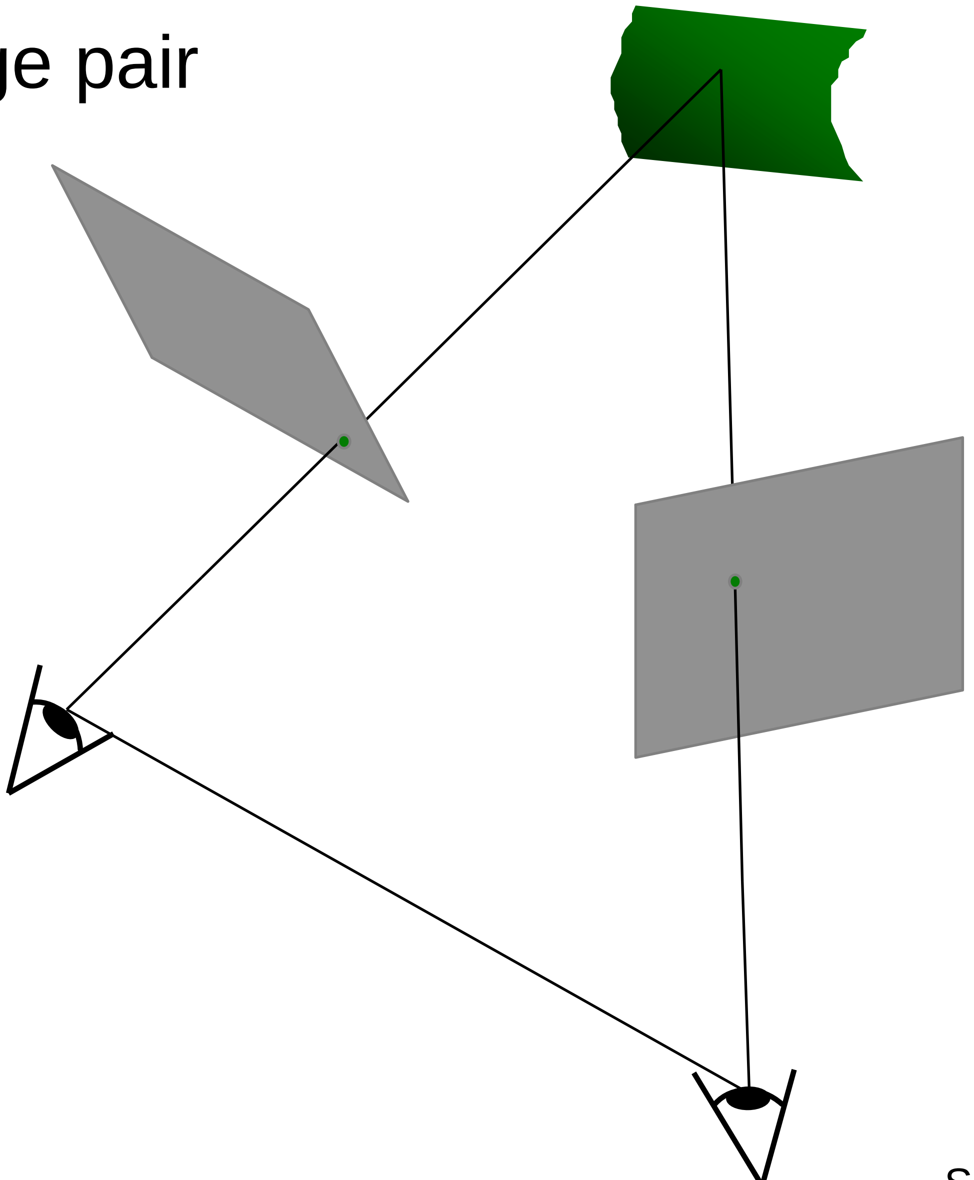
- Image planes of cameras are parallel.
- Focal points are at same height.
- Focal lengths same.

Then, epipolar lines are horizontal scan lines of the images

Image from Andrew Zisserman

Image Rectification

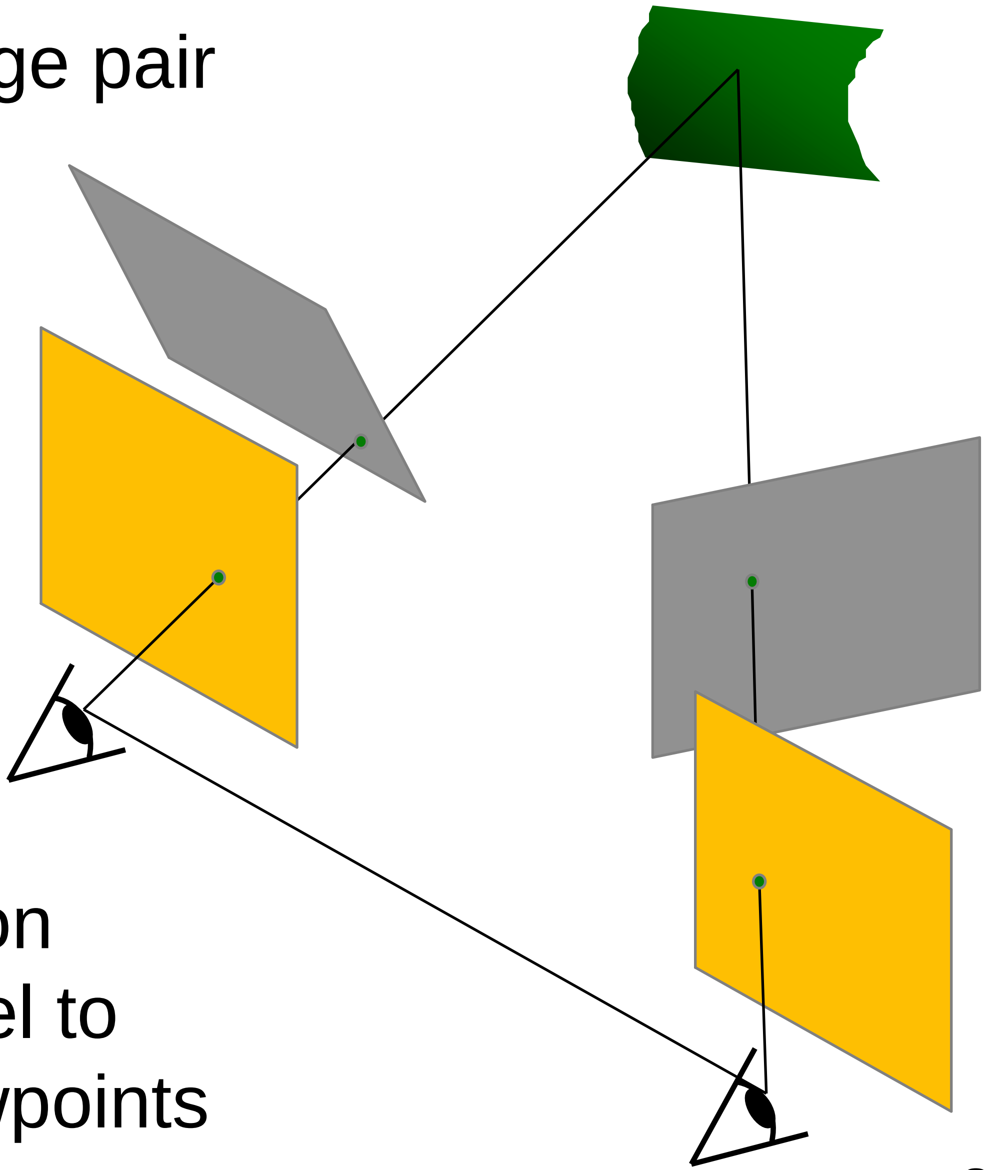
Can rectify any image pair



Seitz

Image Rectification

Can rectify any image pair

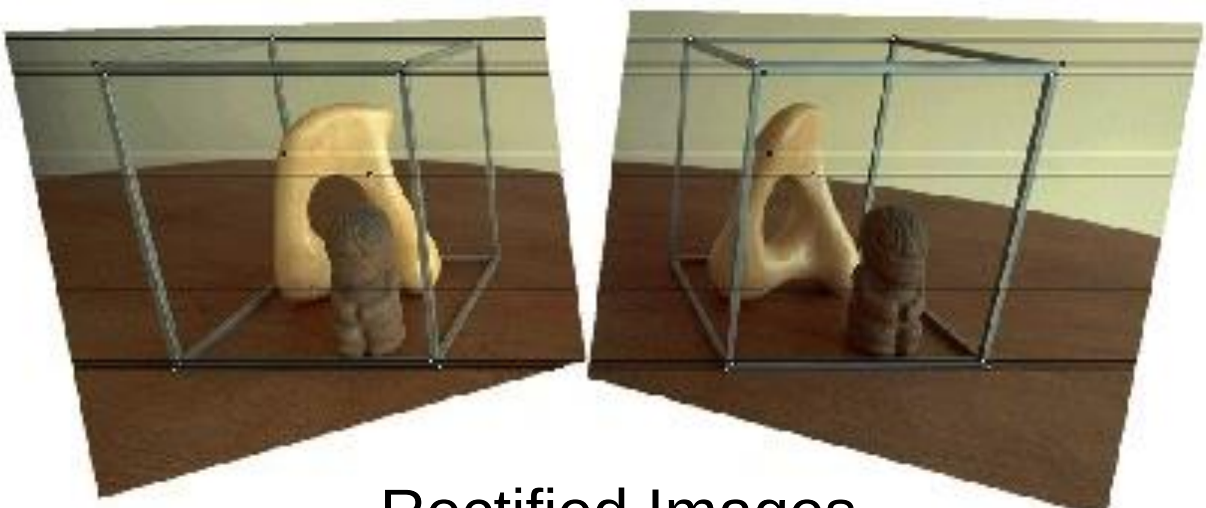


Project onto common view plane parallel to line between viewpoints

Seitz

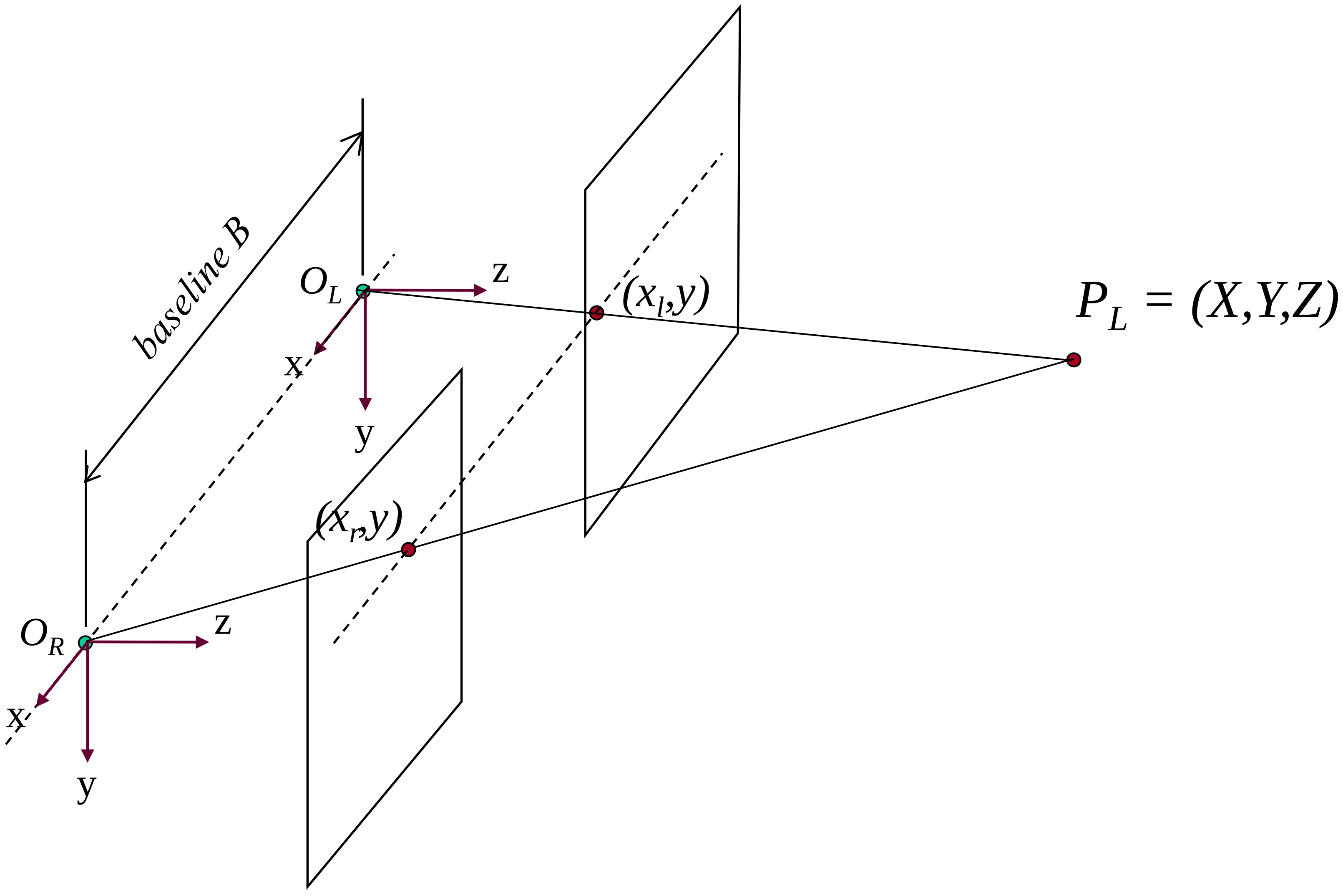
Image Rectification

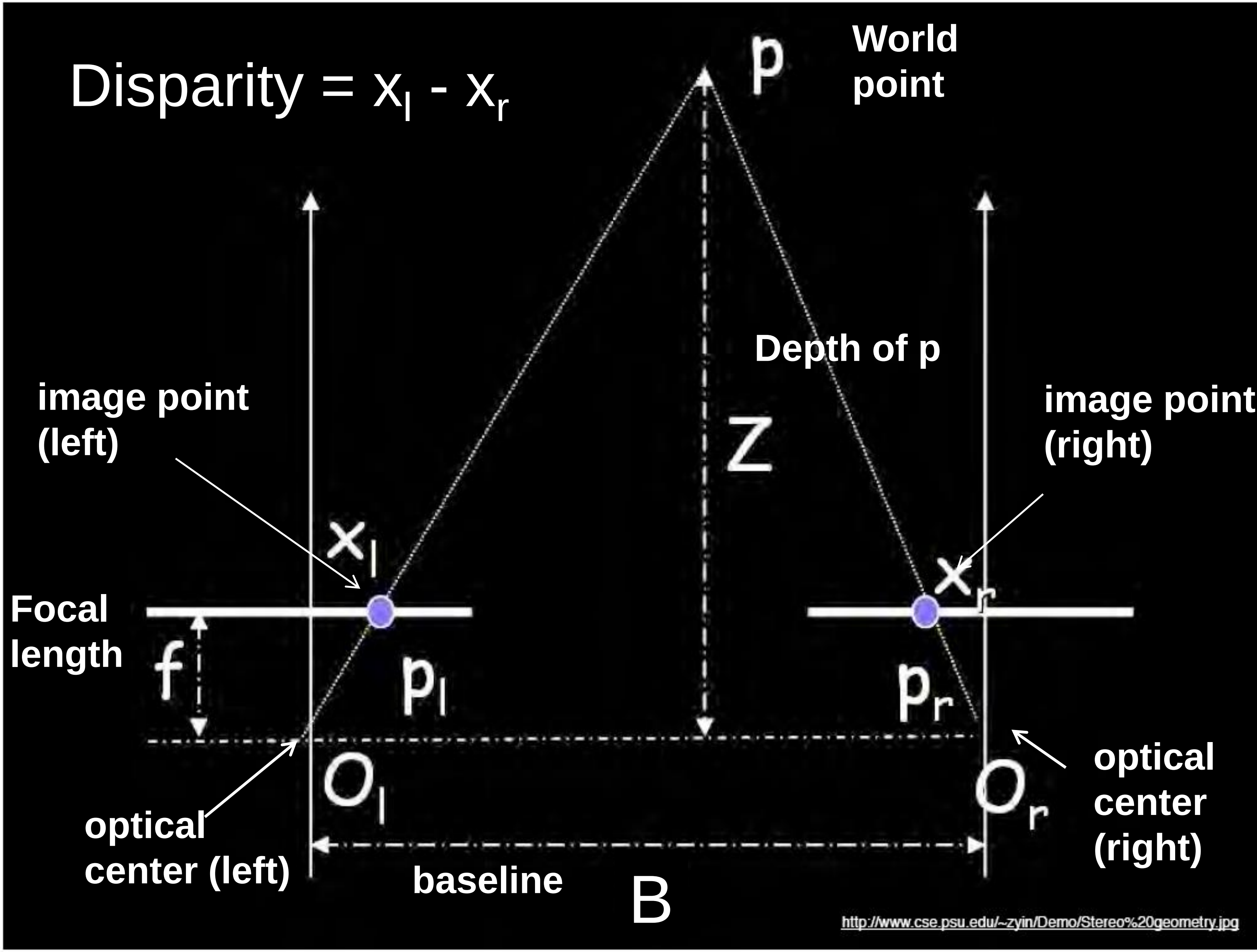
Original Images



Rectified Images

Stereo for Rectified Images





Slide credit: Kristen Grauman

Three Steps

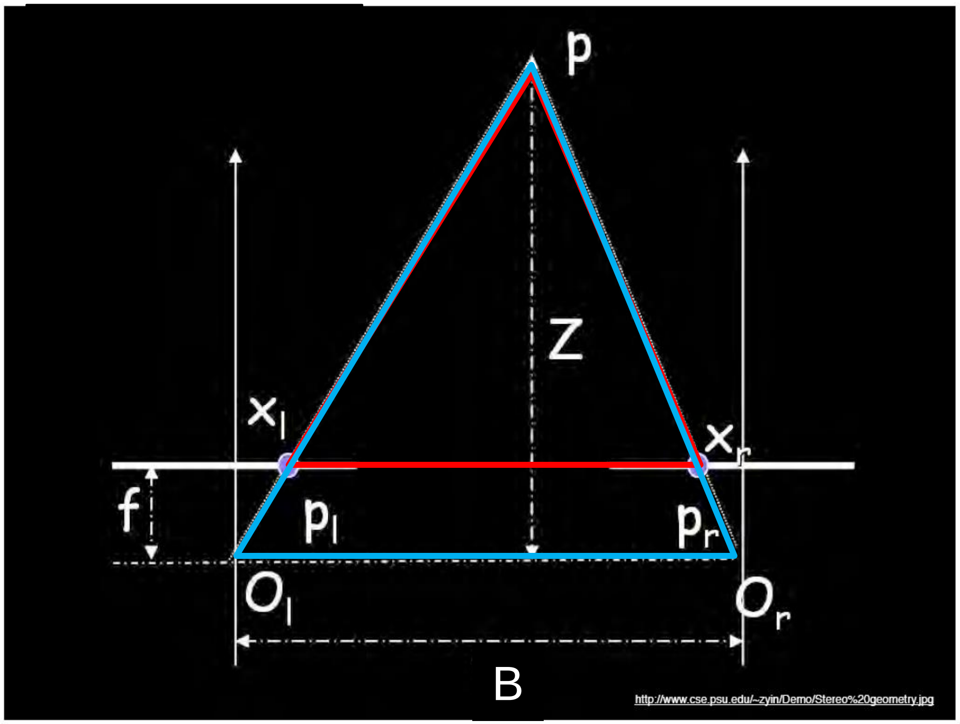
(1) Camera calibration

(2) Dense pixel correspondence

(3) Depth estimation (for one slide)

Depth Estimation

We can estimate depth from disparity using similar triangles (p_l, P, p_r) and (O_l, P, O_r):



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

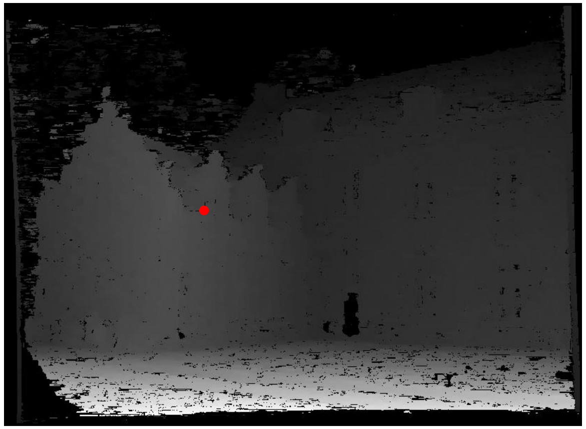
$$Z = f \frac{B}{x_r - x_l}$$

disparity

Slide credit: Kristen Grauman

Main Challenge: Compute Disparity

image Left(x,y) **Disparity map d(x,y)** **image Right(x´,y´)**



$$(x',y') = (x - d(x,y),\ y)$$

Slide credit: Kristen Grauman

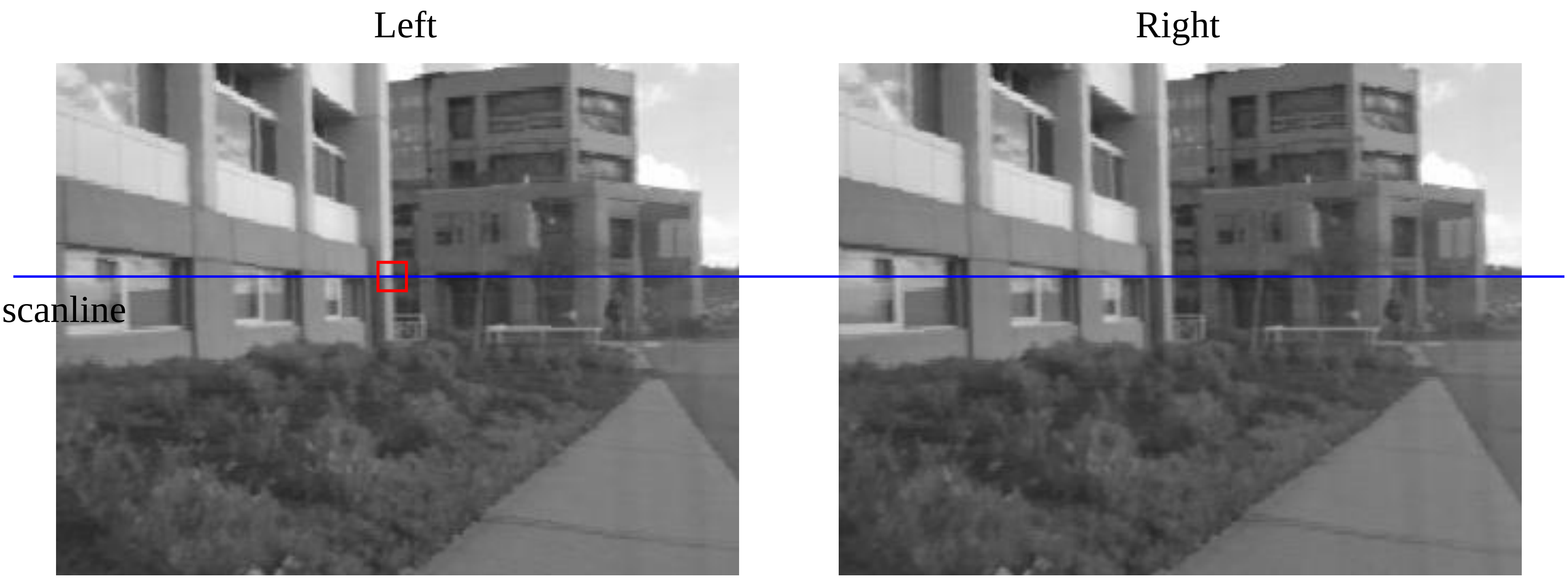
Three Steps

(1) Camera calibration

(2) Dense pixel correspondence (again)

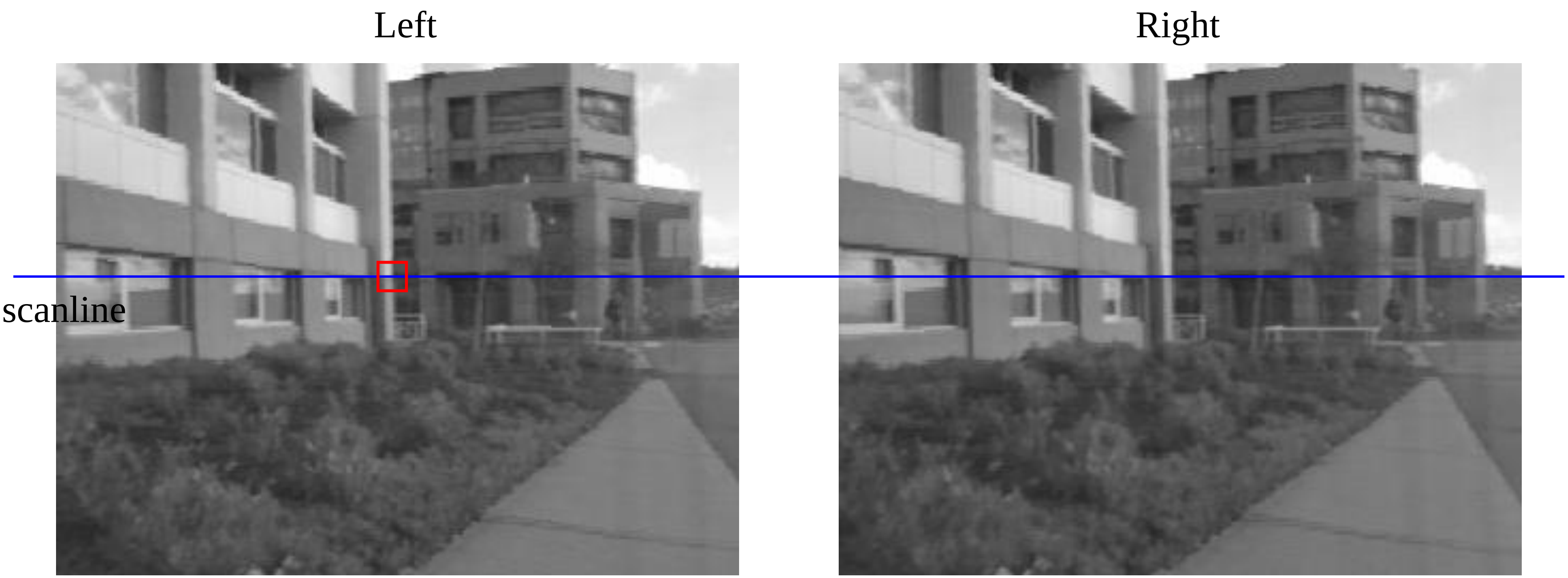
(3) Depth estimation

Stereo Correspondence for Rectified Images



Goal: find the optimal disparity
for every pixel of the left image

Stereo Correspondence for Rectified Images



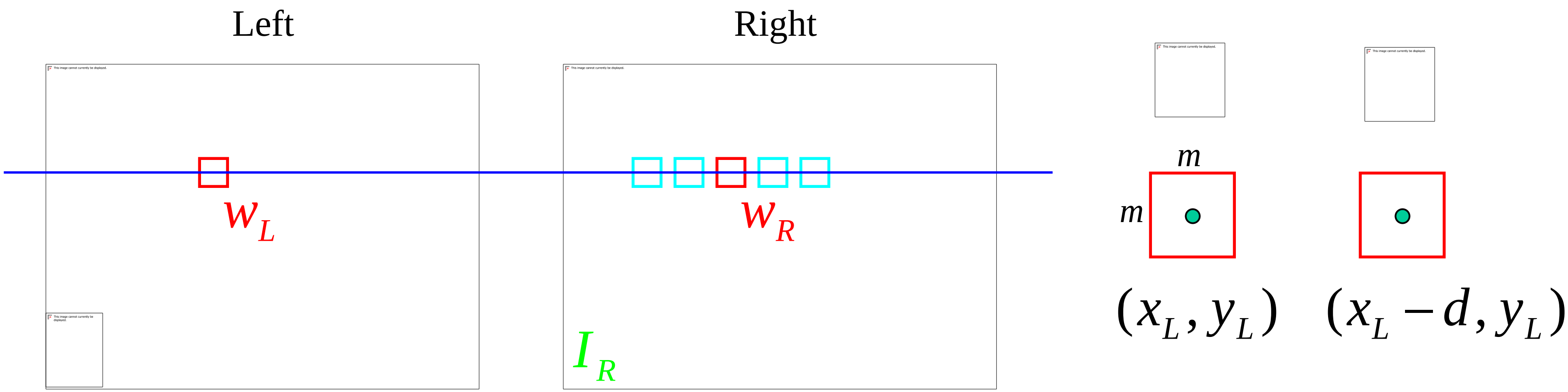
What do we mean by “optimal disparity?”

Optimal Disparity?

Solution should:

- Align similar looking pixels
- Adhere to (expected) constraints of stereo geometry

Measuring Pixel Dissimilarity



'Window' matching error:

Measuring Pixel Dissimilarity

$$L1(x,y,d) = \sum_{(u,v)\in W_m(x,y)} | I_L(u,v) - I_R(u-d,v) |$$

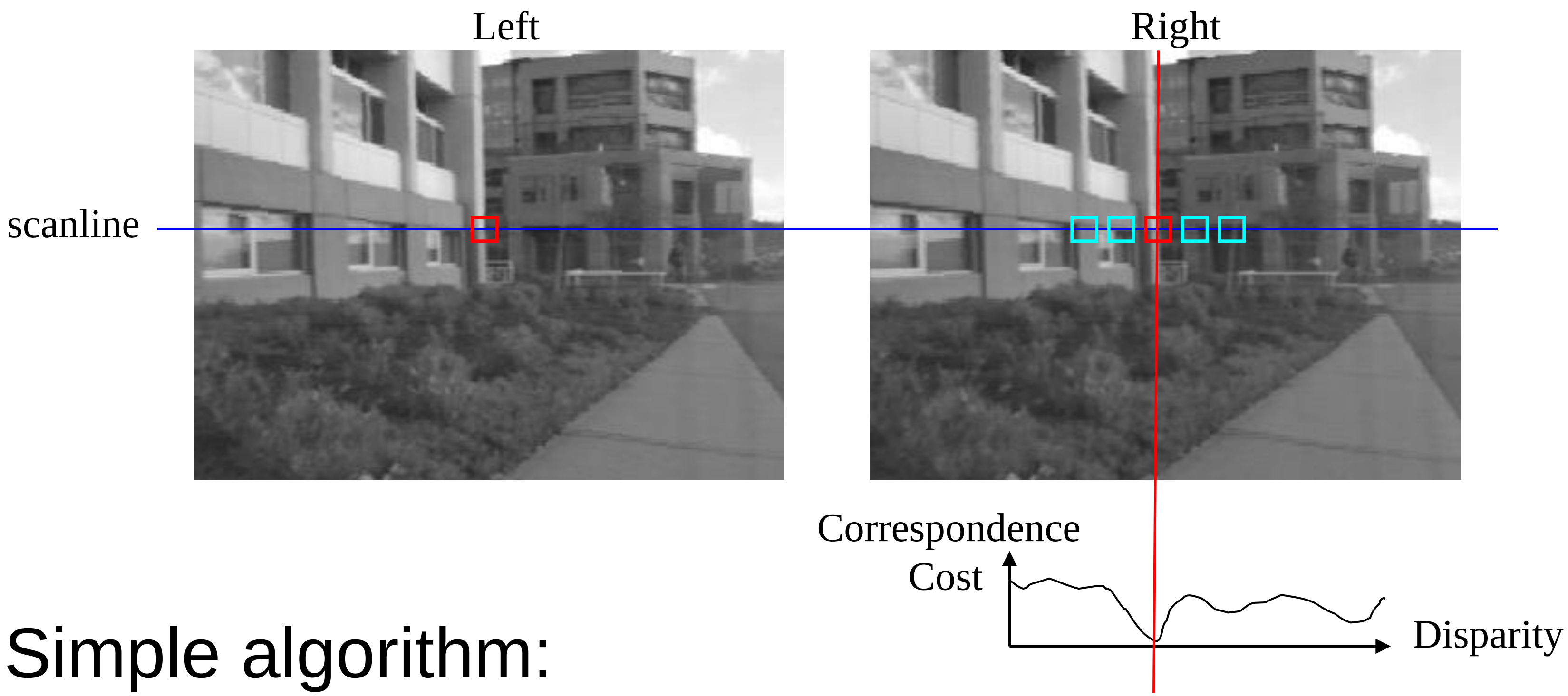
$$L2(x,y,d) = \sqrt{\sum_{(u,v)\in W_m(x,y)} (I_L(u,v) - I_R(u-d,v))^2}$$

$$NCC(x,y,d) = \sum_{(u,v)\in W_m(x,y)} \frac{(I_L(u,v) - \bar{I}_L)(I_R(u-d,v) - \bar{I}_R)}{\sigma_L \sigma_R}$$

$$\bar{I} = \sum_{(u,v)\in W_m(x,y)} \frac{I(u,v)}{m^2}$$ (Mean)

$$\sigma = \sqrt{\sum_{(u,v)\in W_m(x,y)} \frac{(I(u,v) - \bar{I})^2}{m^2}}$$ (Standard devaition)

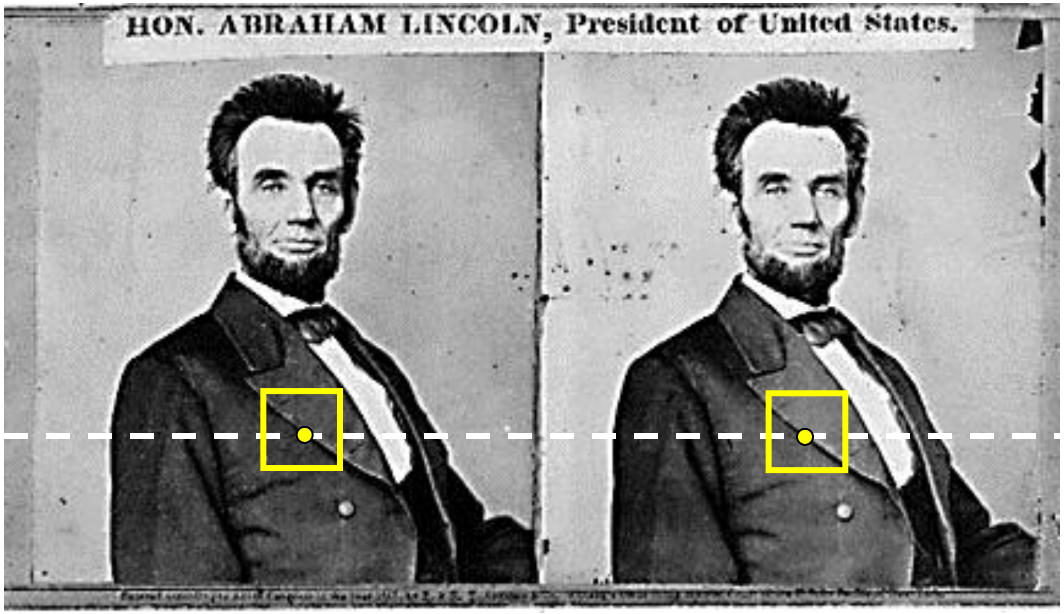
Simple Algorithm



Simple algorithm:

- Slide a window along the right scanline and compare contents of that window with the reference window in the left image
- Return disparity with minimal pixel dissimilarity

Simple Algorithm

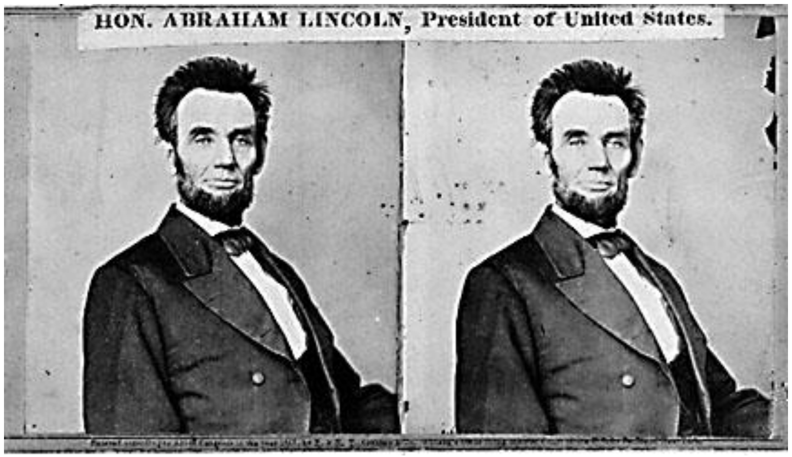


For each epipolar line

For each pixel / window in the left image

- compare with every pixel / window on same epipolar line in right image
- pick position with minimum dissimilarity (e.g., luminance difference)

Failures of Simple Algorithm



Textureless surfaces

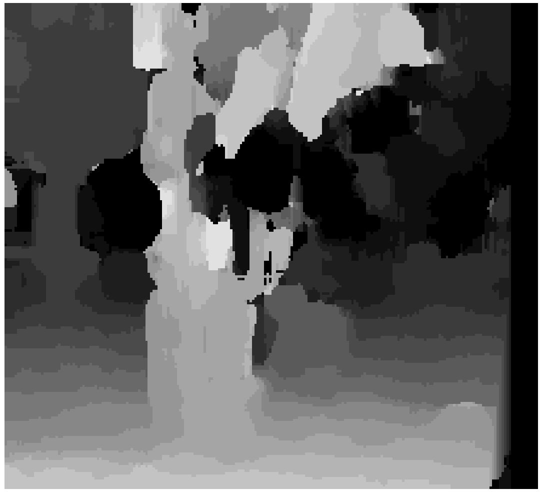
Occlusions, repetition

Non-Lambertian surfaces, specularities

Seitz

What about larger window sizes?

W = 3



W = 20

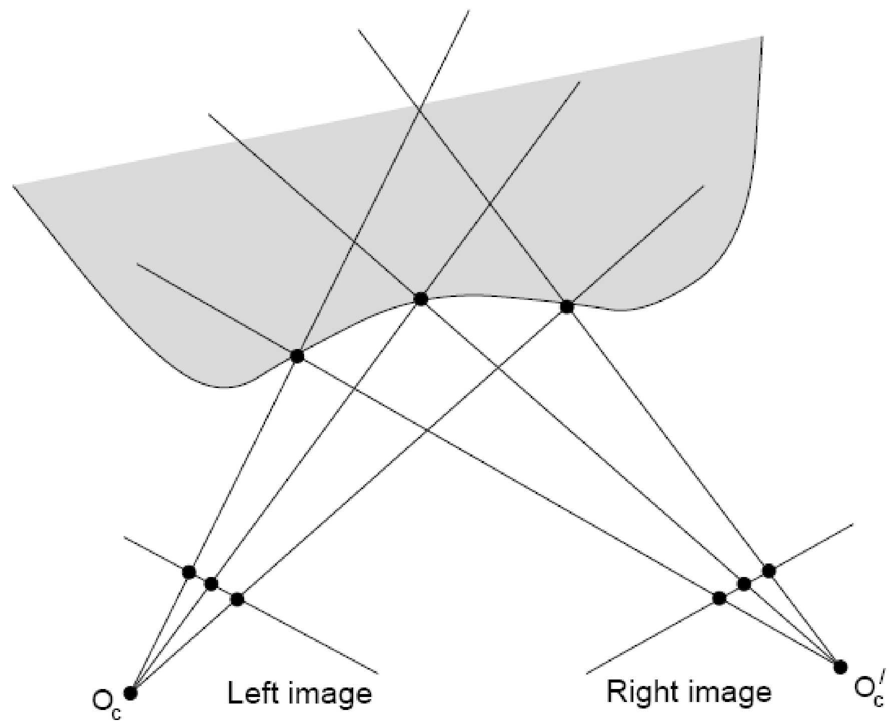
Want window large enough to have sufficient intensity variation, yet small enough to contain only pixels with about the same disparity.

Figures from Li Zhang

Global Correspondence problem

Beyond the hard constraint of epipolar geometry and the soft constraint of pixel similarities, other considerations can be used to help identify correspondences

- Uniqueness
- Ordering
- Smoothness



Kristen Grauman

Uniqueness

One match in right image for every point in left image

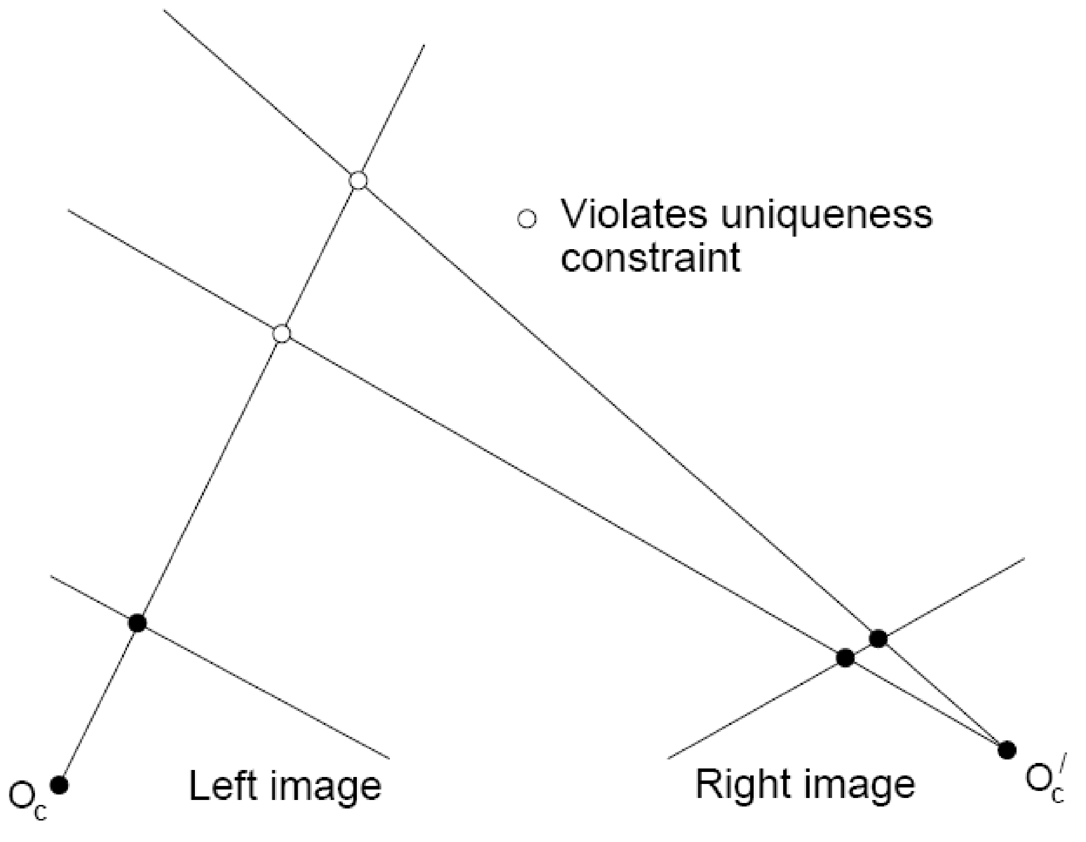


Figure from Gee & Cipolla 1999

Ordering

Points on **same surface** (opaque object) should be in same order in both views

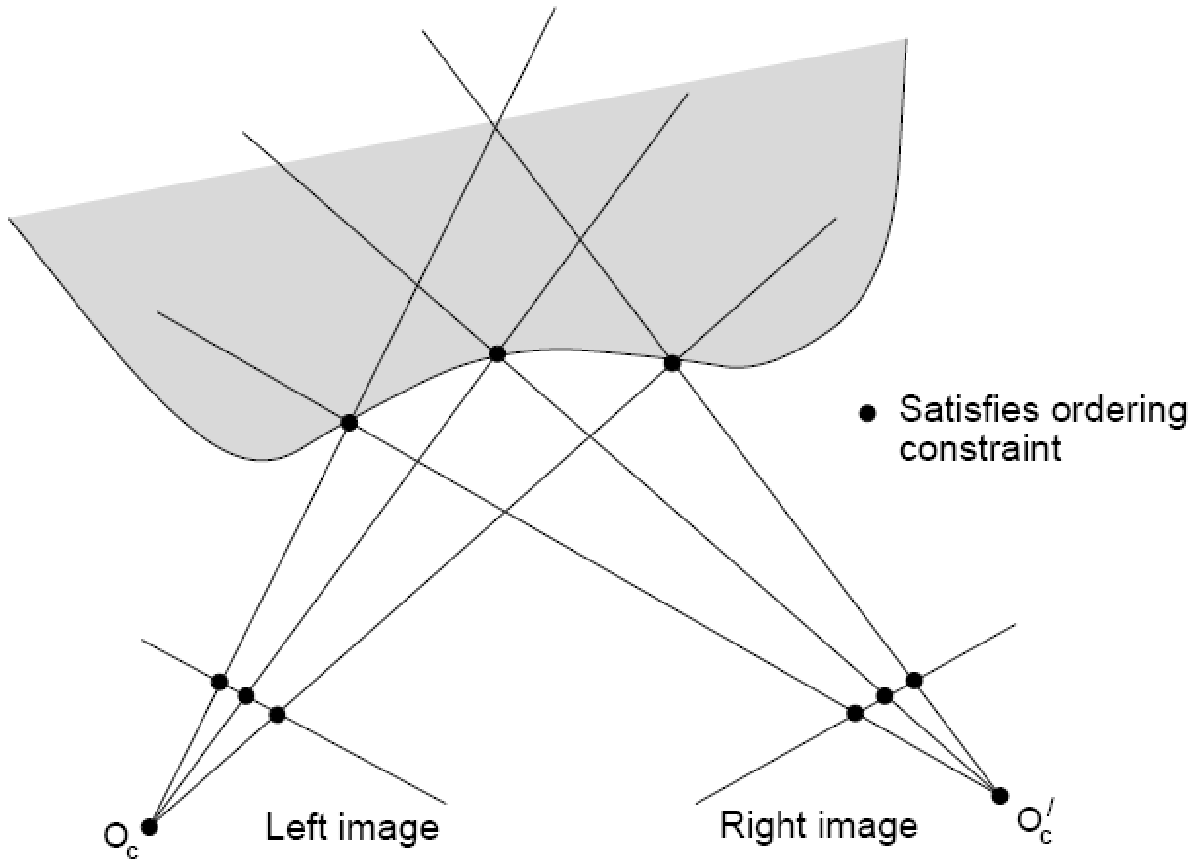
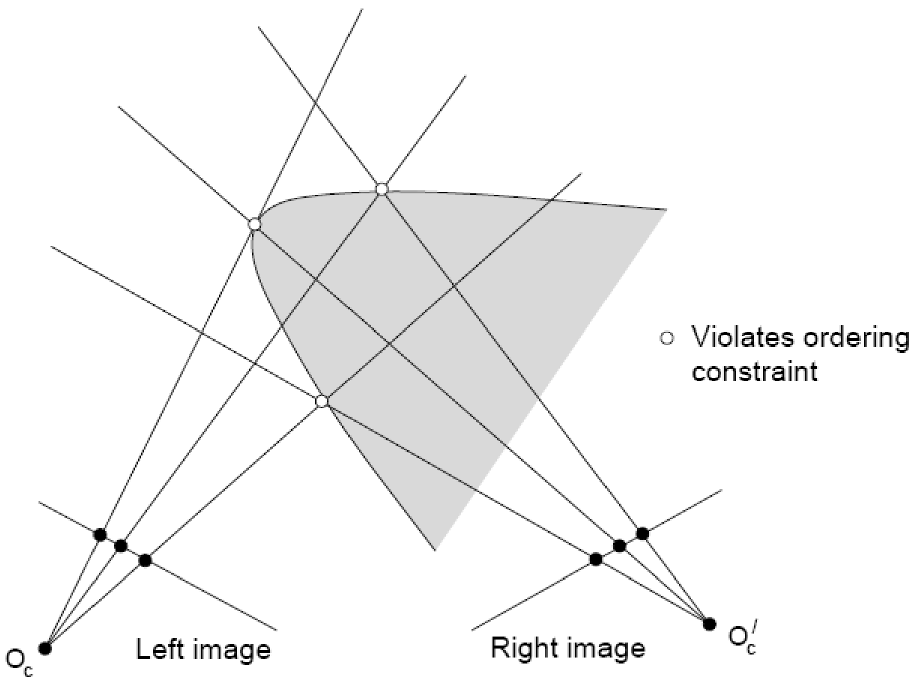


Figure from Gee & Cipolla 1999

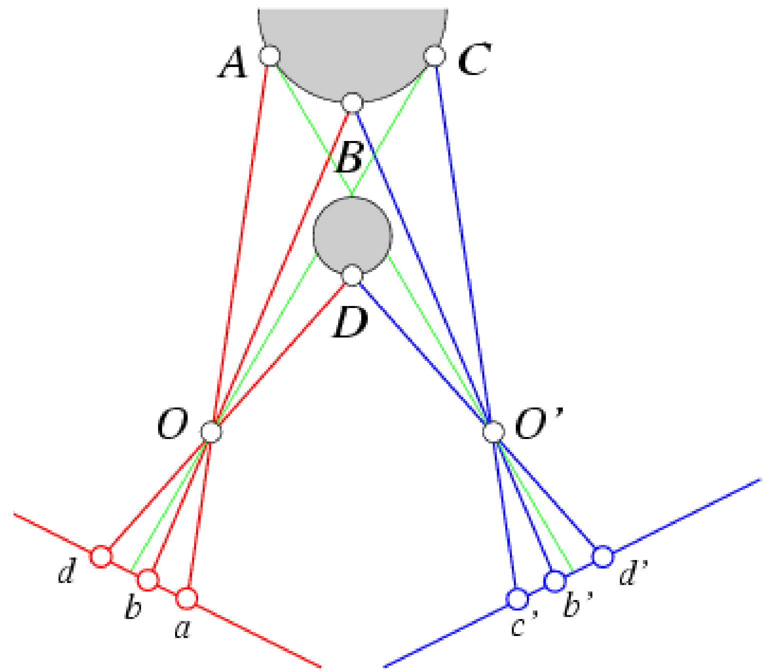
Ordering

Points on **same surface** (opaque object) should be in same order in both views

- Not always true, but still useful



Transparent surface

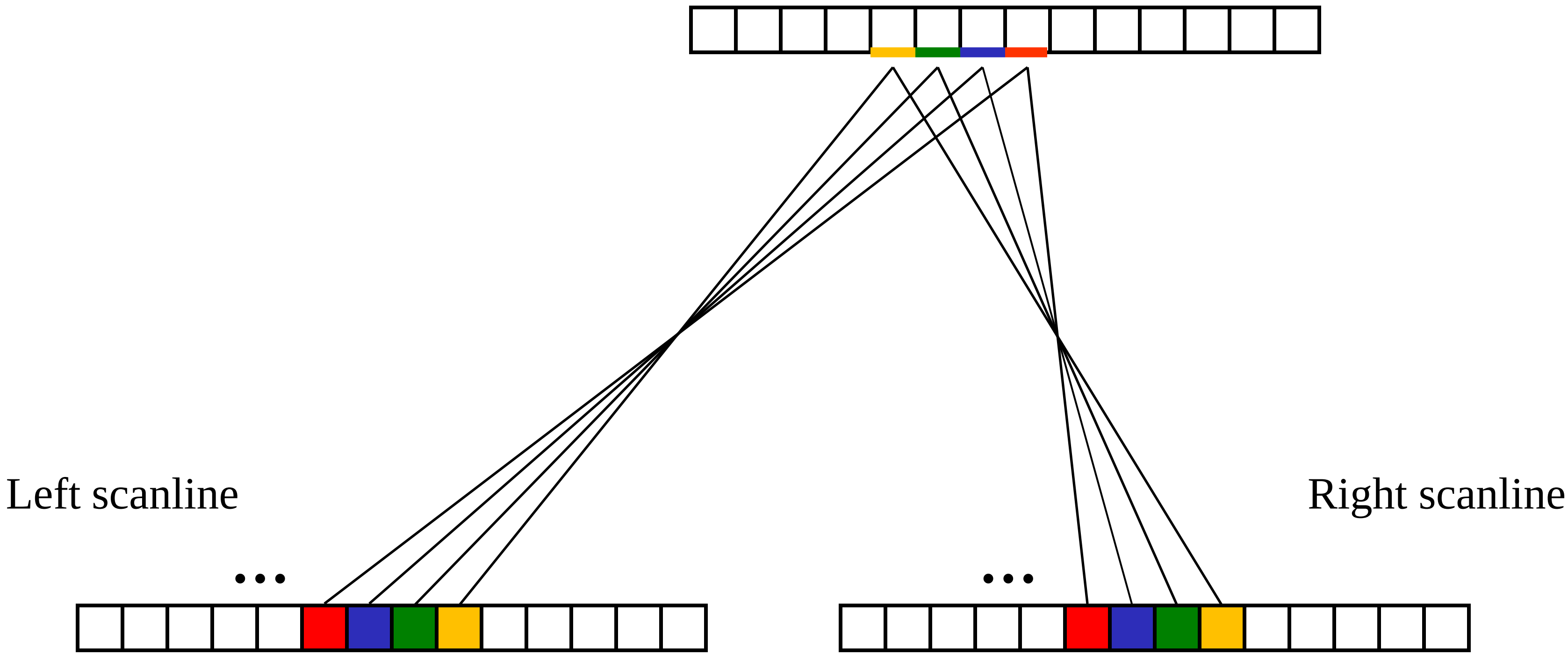


Thin occluder

Figures from Forsyth & Ponce

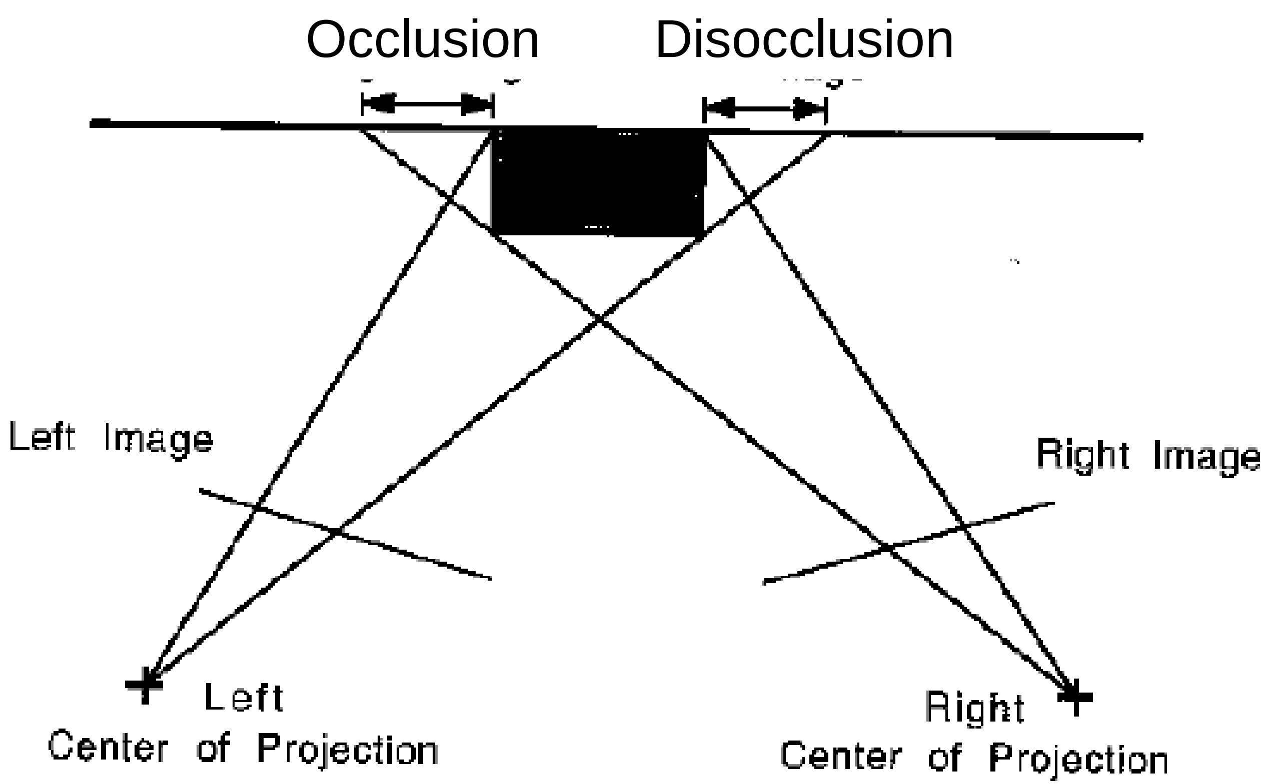
Smoothness

If surfaces are smooth and there are no occlusions, then disparities are smooth



Smoothness

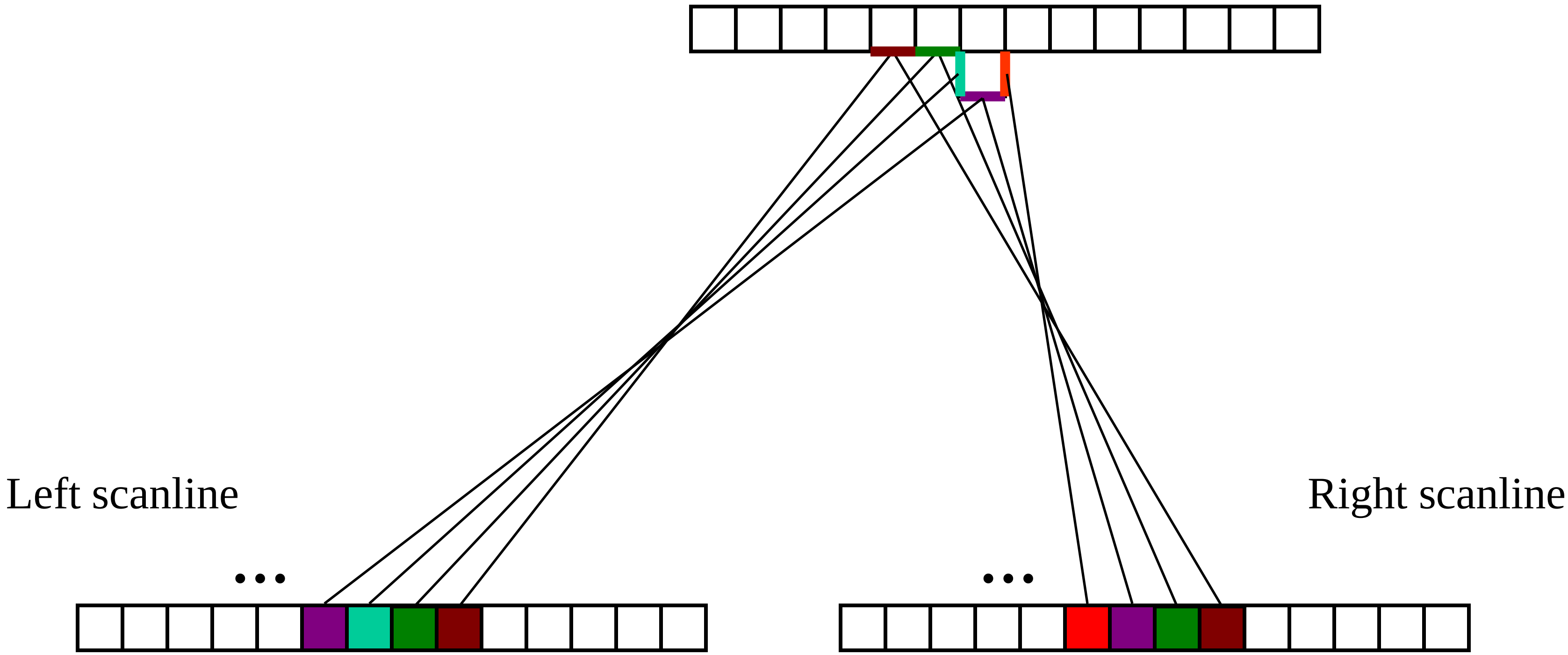
What is an occlusion?



Slide credit: David Kriegman

Smoothness

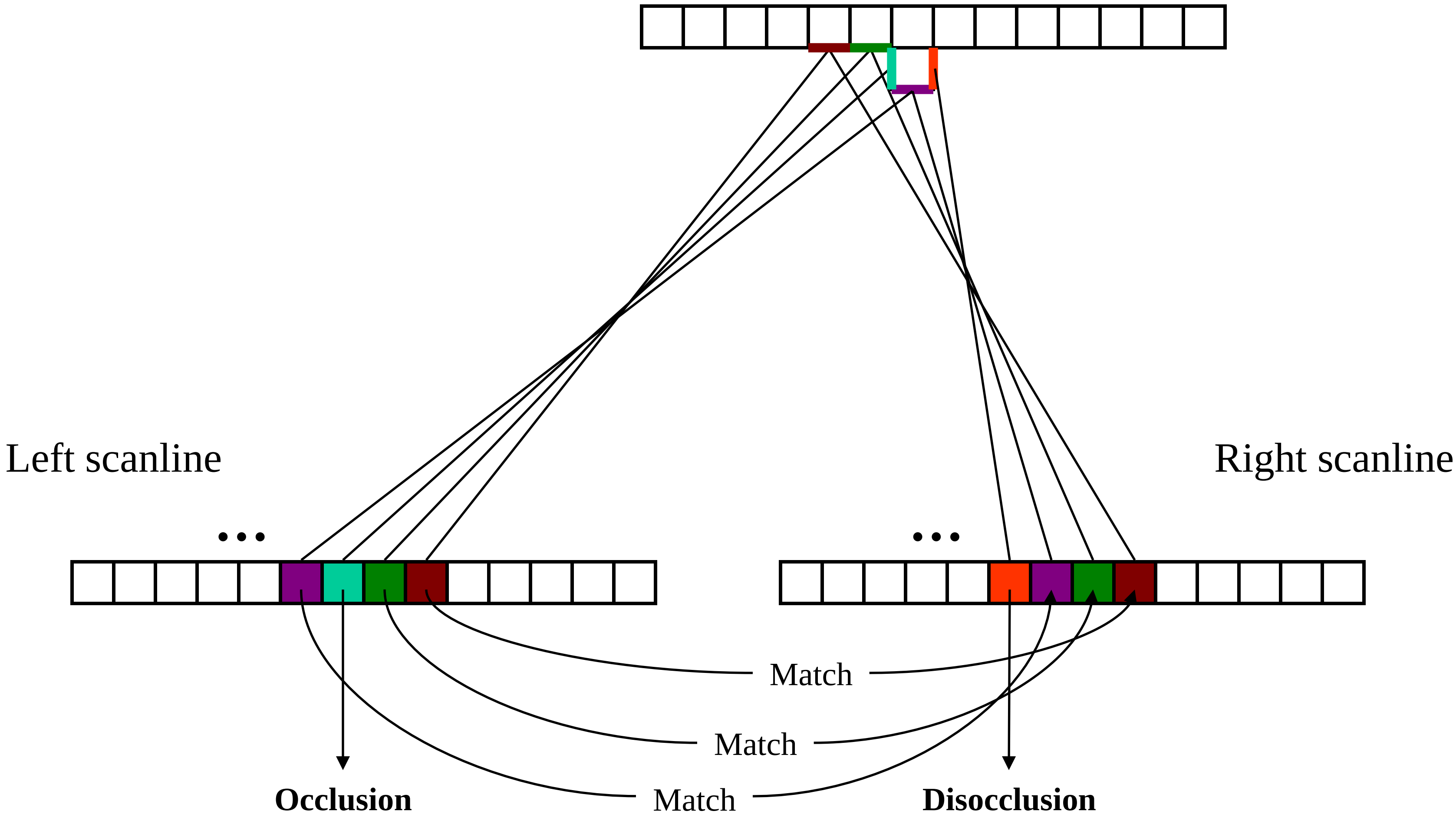
What happens to disparity for occlusions?



Thrun, Szeliski, Dahlkamp, and Morris

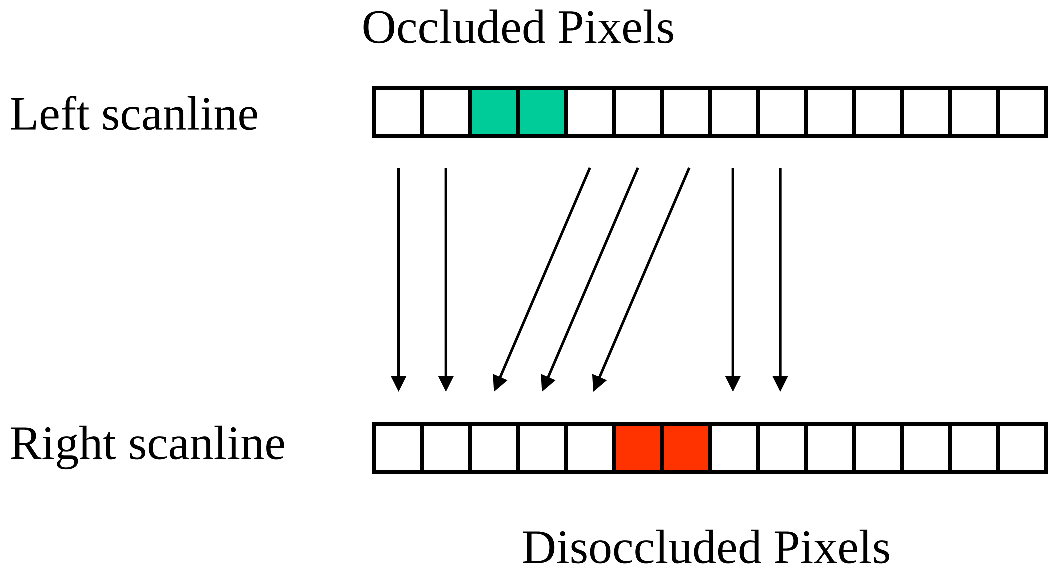
Smoothness

What happens to disparity for occlusions?



Thrun, Szeliski, Dahlkamp, and Morris

Smoothness



Three cases:

- Sequential – smooth disparity
- Occluded – causes negative jump in disparity
- Disoccluded – causes positive jump in disparity

Challenge

Solve for disparities that not only align similar pixels but also have soft constraints between them

- Uniqueness
- Ordering
- Smoothness

How?

Stereo as an Optimization Problem

Minimize error function:

$$E(x,y,d) = \sum_{x,y}^{Pixels} data(x,y,d(x,y)) + \sum_{x,y,nx,ny}^{\substack{Pixel \\ Neighbors}} smoothness\big(d(x,y), d(nx,ny)\big)$$

where:

data(x, y, k) = cost of assigning disparity k at pixel (x,y)

smoothness(d1, d2) = cost of assigning disparities d1 and d2 at neighboring pixels.

Stereo as an Optimization Problem

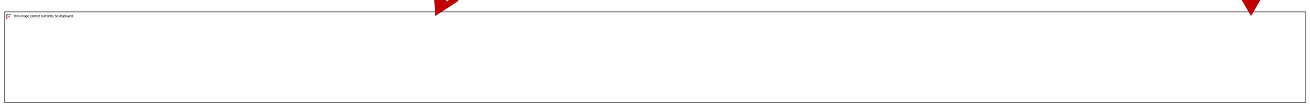
Minimize error function:

$$E(x,y,d) = \sum_{x,y}^{Pixels} data(x,y,d(x,y)) + \sum_{x,y,nx,ny}^{\substack{Pixel\\Neighbors}} smoothness\big(d(x,y),d(nx,ny)\big)$$

For example:

data_term_weight

max_data_term_value



$$smoothness(d1, d2) = min(|\, d1 - d2\, |, S)$$

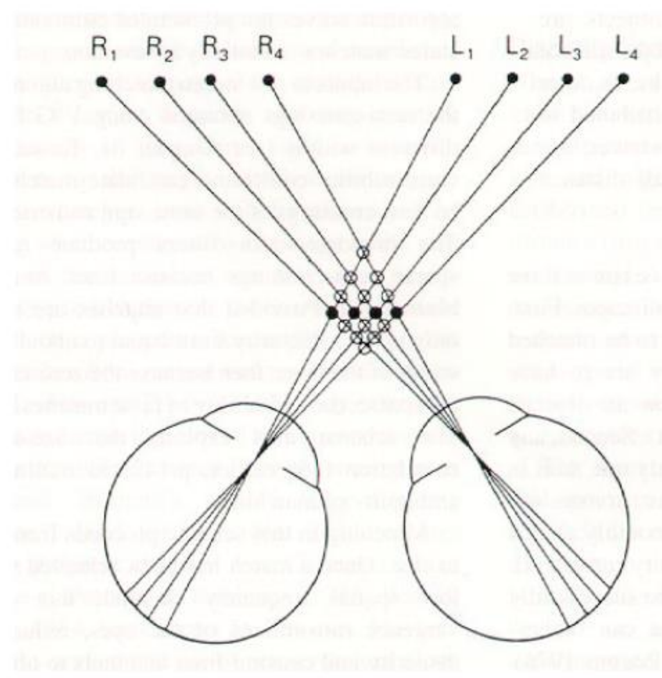
max_smoothness_term_value

Stereo as an Optimization Problem

Minimize error function:

$$E(x,y,d) = \sum_{x,y}^{Pixels} data(x,y,d(x,y)) + \sum_{x,y,nx,ny}^{\substack{Pixel\\Neighbors}} smoothness\big(d(x,y), d(nx,ny)\big)$$

Unfortunately, optimizing this error function is NP-Hard



N points have N! possible correspondences

Potetz

Two Possible Algorithms

Dynamic programming

Graph cuts

Two Possible Algorithms

Dynamic programming ←

Graph cuts

Dynamic Programming Algorithm

Simplify problem by ignoring smoothness costs between vertical neighbors

$$E(x,y,d) = \sum_{x,y}^{Pixels} data(x,y,d(x,y)) + \sum_{x,y,nx,ny}^{\substack{Horizontal\\Neighbors}} smoothness\big(d(x,y),d(nx,ny)\big)$$

Then can find optimal solution for each scanline independently with dynamic programming

- plus, maintains order of pixel correspondences

Dynamic Programming Algorithm

Like string alignment, but our formulation will include a smoothness term rather than a gap penalty

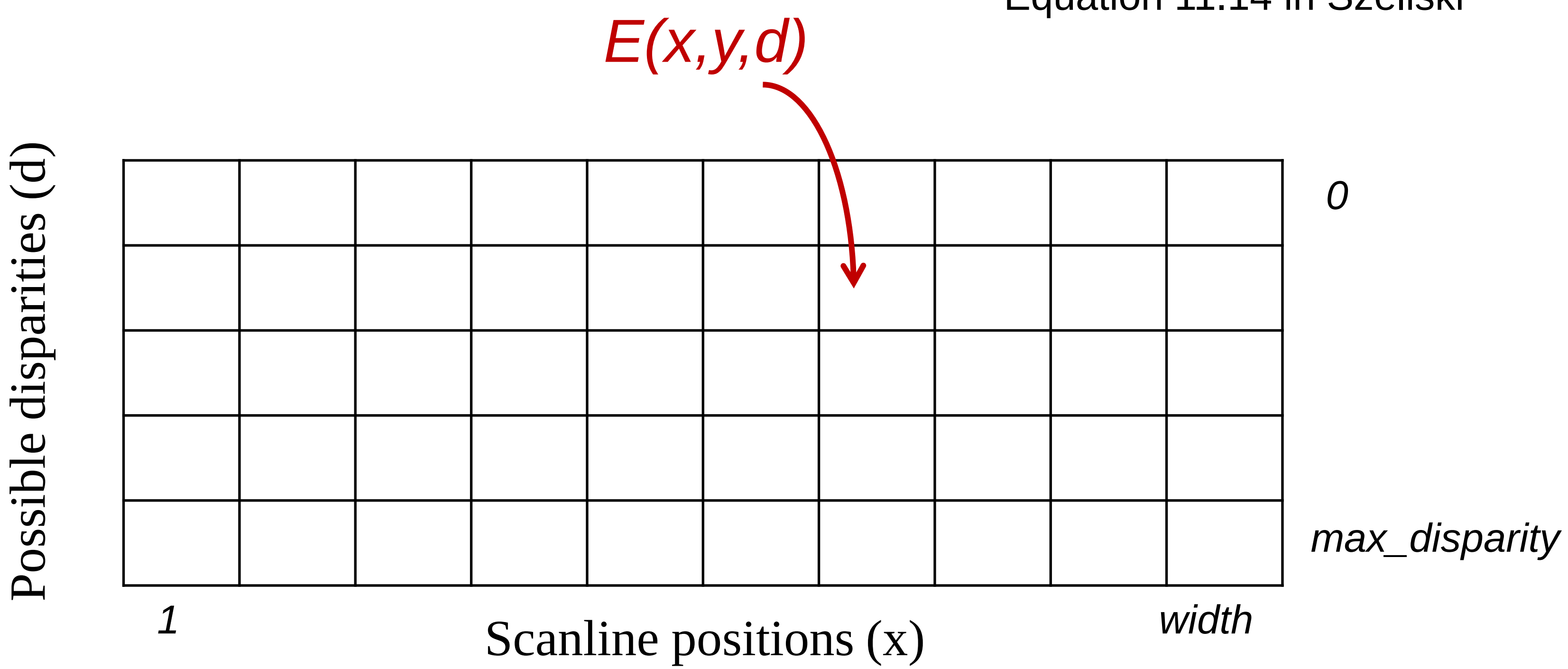
1) Compute error of prefixes
2) Find best overall error
3) Backtrack to find disparities

Dynamic Programming Algorithm

1) Incrementally update optimal energy *E(x,d) for prefix if assign* disparity *d* at pixel x

$$E(x,y,d) = \min_{d'}(data(x,y,d) + smoothness(d,d') + E\,(x-1,y,d'))$$

Equation 11.14 in Szeliski

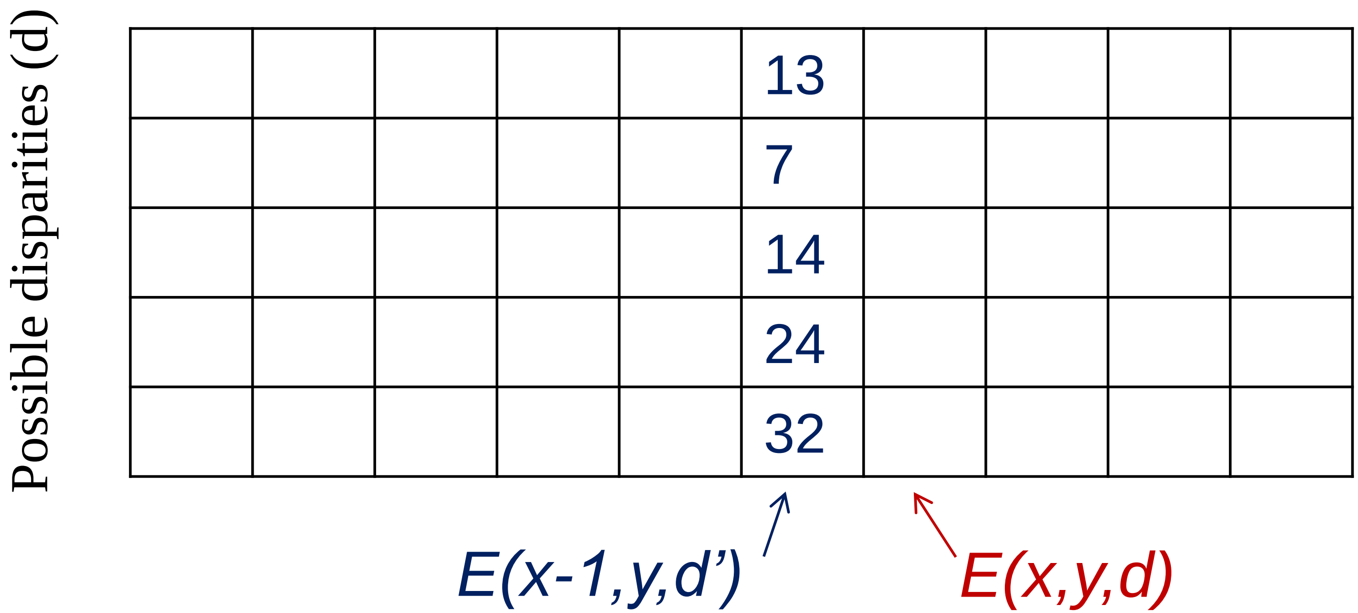


Dynamic Programming Algorithm

1) Incrementally update optimal energy *E(x,d) for prefix if assign* disparity *d* at pixel x

$$E(x, y, d) = \min_{d'}(data(x, y, d) + smoothness(d, d') + E\,(x-1, y, d'))$$

For example, if data(7,y,2) = 1 and data(7,y,not2)=10



Dynamic Programming Algorithm

1) Incrementally update optimal energy *E(x,d) for prefix if assign* disparity *d* at pixel x

$$E(x,y,d) = \min_{d'}\left(data(x,y,d) + smoothness(d,d') + E\,(x-1,y,d')\right)$$

For example, if data(7,y,2) = 1 and data(7,y,not2)=10

Possible disparities (d)

					13				
					7				
					14	9			
					24				
					32				

E(x-1,y,d')   *E(x,y,d)*

Dynamic Programming Algorithm

1) Incrementally update optimal energy *E(x,d) for prefix if assign* disparity *d* at pixel x

$$E(x, y, d) = \min_{d'}(data(x, y, d) + smoothness(d, d') + E\ (x - 1, y, d'))$$

For example, if data(7,y,2) = 1 and data(7,y,not2)=10

Possible disparities (d)

					13	18			
					7				
					14	9			
					24				
					32				

E(x-1,y,d') *E(x,y,d)*

Dynamic Programming Algorithm

1) Incrementally update optimal energy *E(x,d) for prefix if assign* disparity *d* at pixel x

$$E(x,y,d) = \min_{d'}\left(data(x,y,d) + smoothness(d,d') + E\,(x-1,y,d')\right)$$

For example, if data(7,y,2) = 1 and data(7,y,not2)=10

Possible disparities (d)

					13	18			
					7	17			
					14	9			
					24				
					32				

E(x-1,y,d') *E(x,y,d)*

Dynamic Programming Algorithm

1) Incrementally update optimal energy *E(x,d) for prefix if assign* disparity *d* at pixel x

$$E(x,y,d) = \min_{d'}\left(data(x,y,d) + smoothness(d,d') + E\,(x-1,y,d')\right)$$

For example, if data(7,y,2) = 1 and data(7,y,not2)=10

Possible disparities (d)

					13	18			
					7	17			
					14	9			
					24	19			
					32	20			

E(x-1,y,d') *E(x,y,d)*

Dynamic Programming Algorithm

2) Find the best alignment for entire string

Best error for entire string

Possible disparities (d)

2	4	7	8	11	13	18	19	12	12
∞	3	6	8	8	8	17	10	14	19
∞	∞	5	6	6	14	9	11	14	15
∞	∞	∞	12	15	24	19	14	19	20
∞	∞	∞	∞	16	32	20	17	22	23

Scanline positions (x)

Dynamic Programming Algorithm

3) Find path back through prefixes that "supported" the best alignment

$$d(x) = \underset{d'}{\operatorname{argmin}}(|data(x+1, y, d(x+1)) + smoothness(d(x+1), d') + E\ (x, y, d') - E(x+1, y, d(x+1))|$$

Find d(x-1) whose error increment "matches" forward step

Possible disparities (d)

2	4	7	8	11	13	18	19	12	12
∞	3	6	8	8	8	17	10	14	19
∞	∞	5	6	6	14	9	11	14	15
∞	∞	∞	12	15	24	19	14	19	20
∞	∞	∞	∞	16	32	20	17	22	23

Scanline positions (x)

Dynamic Programming Algorithm

Or, equivalently:

$$d(x-1) = \underset{d'}{\operatorname{argmin}} \left| data(x,y,d(x)) + smoothness(d(x),d') + E(x-1,y,d') - E(x,y,d(x)) \right|$$

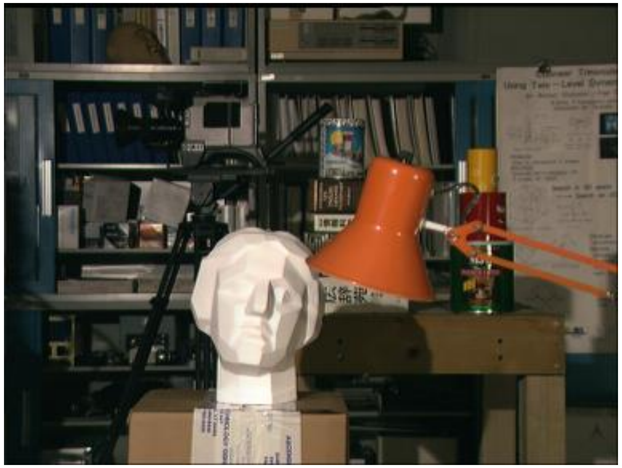
Find d(x-1) whose error increment “matches” forward step

Possible disparities (d)

2	4	7	8	11	13	18	19	12	12
∞	3	6	8	8	8	17	10	14	19
∞	∞	5	6	6	14	9	11	14	15
∞	∞	∞	12	15	24	19	14	19	20
∞	∞	∞	∞	16	32	20	17	22	23

Scanline positions (x)

Dynamic Programming Results




Two Possible Algorithms

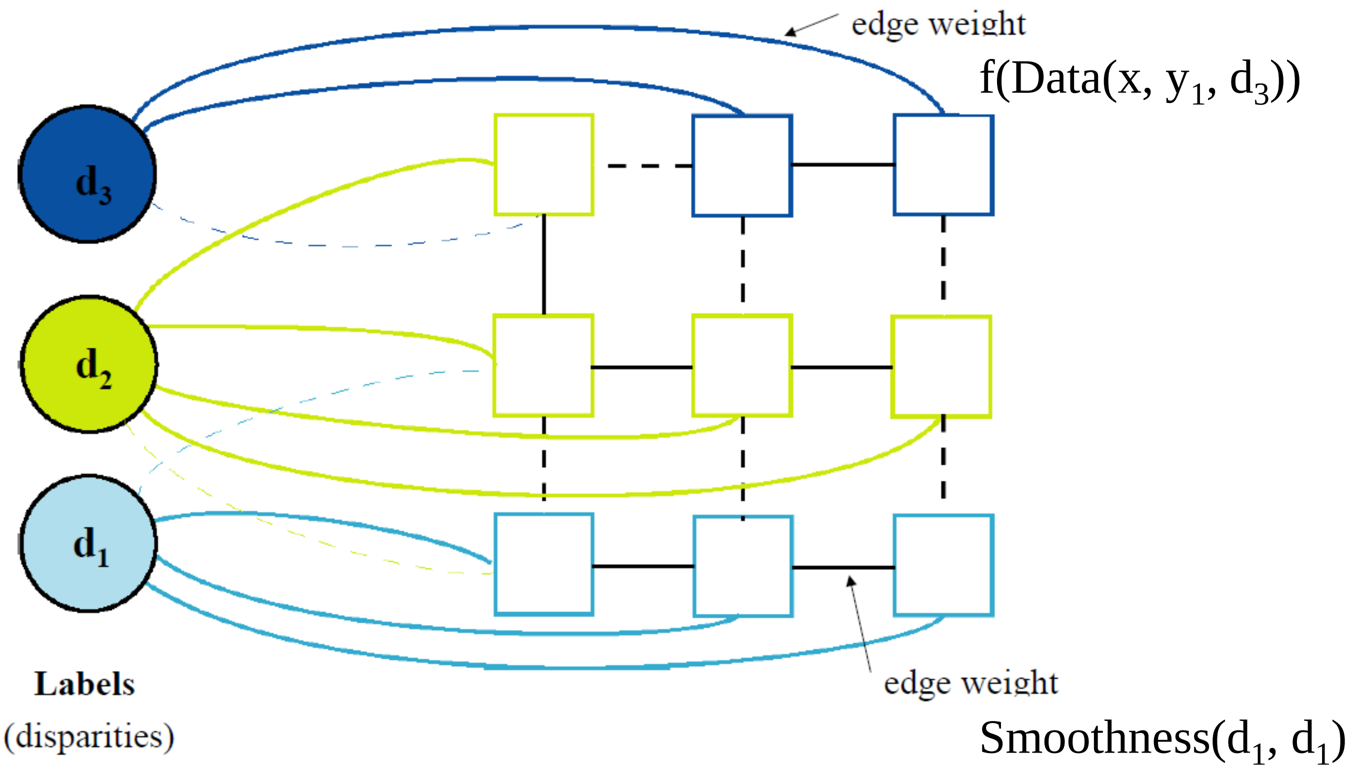
Dynamic programming

Graph cuts ←

Graph Cut Algorithm

Build graph where:

- Nodes represent pixels and disparities
- Edges from pixels to disparities (based on data term)
- Edges between neighbor pixels (smoothness term)



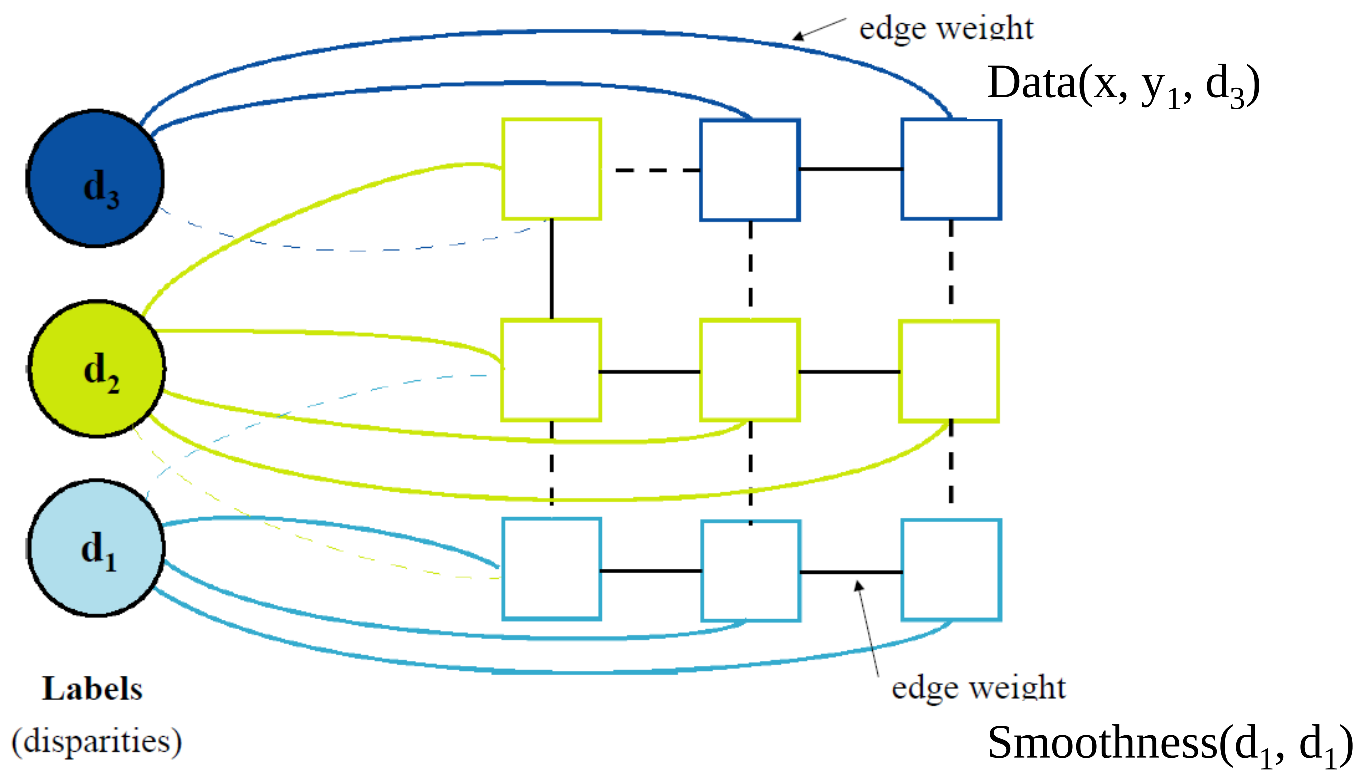
Boykov

Graph Cut Algorithm

Find graph cut where:

- Every pixel is connected to one disparity
- Sum of cut edge weights is minimized

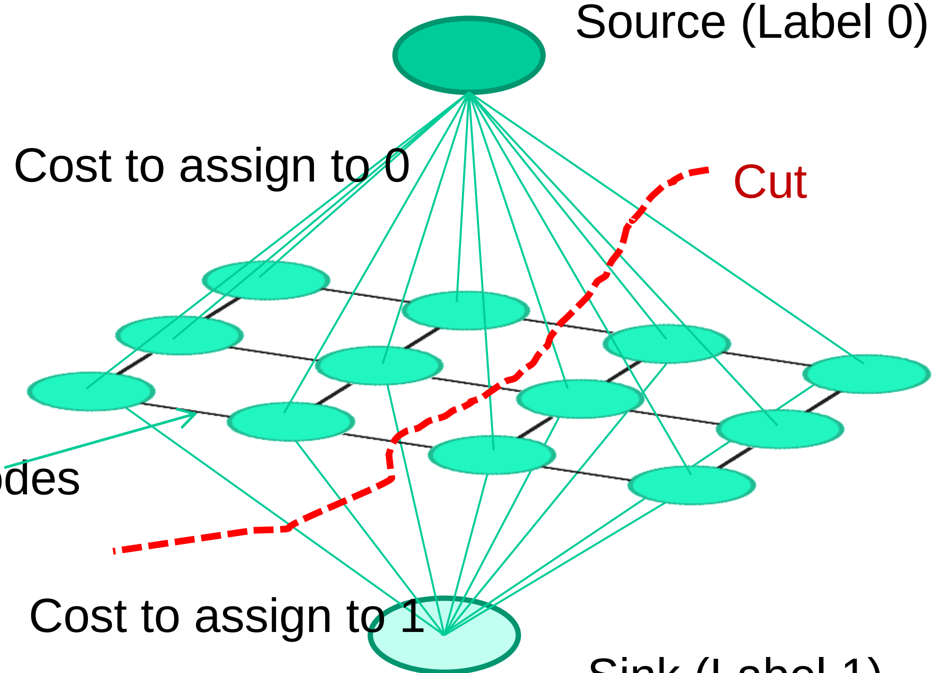
This equivalent to finding global minimum of energy function



Boykov

Graph Cut Algorithm

Optimal solutions available for 2-label problems (from segmentation slides)



$$Energy(x;\theta,data) = \underbrace{\sum_i \psi_1(x_i;\theta,data)}_{\text{Unary Potential}} \quad \underbrace{\sum_{i,j\in edges} \psi_2(x_i,x_j;\theta,data)}_{\text{Edge Potential}}$$

Graph Cut Algorithm

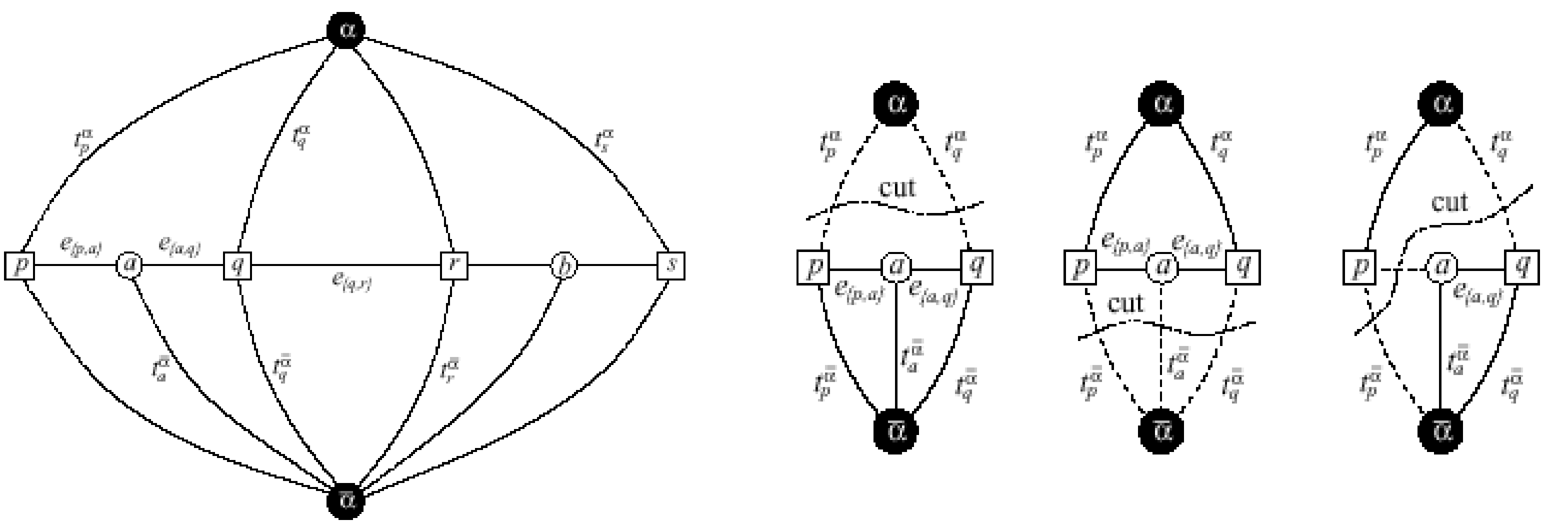
Approximation algorithms for multi-label algorithms:

- α expansion
- α-β swap

Basic idea: break multi-label cut computation into sequence of two-label cuts

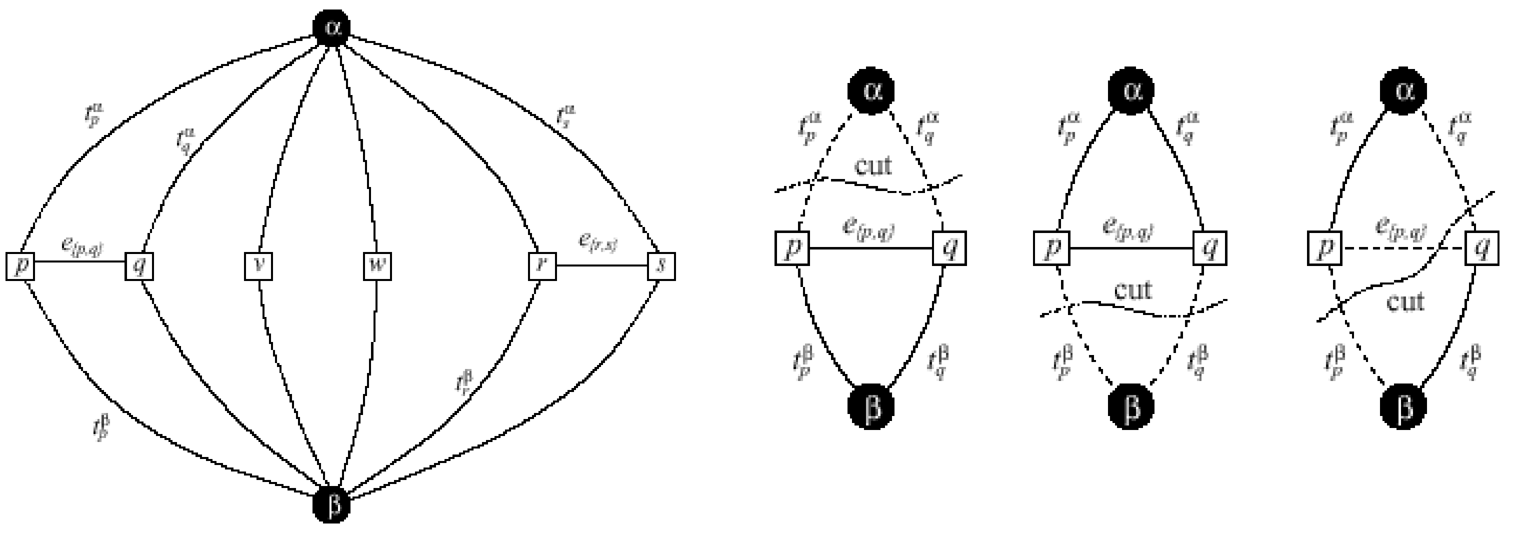
Graph Cut Algorithm

α expansion: add pixels to α class

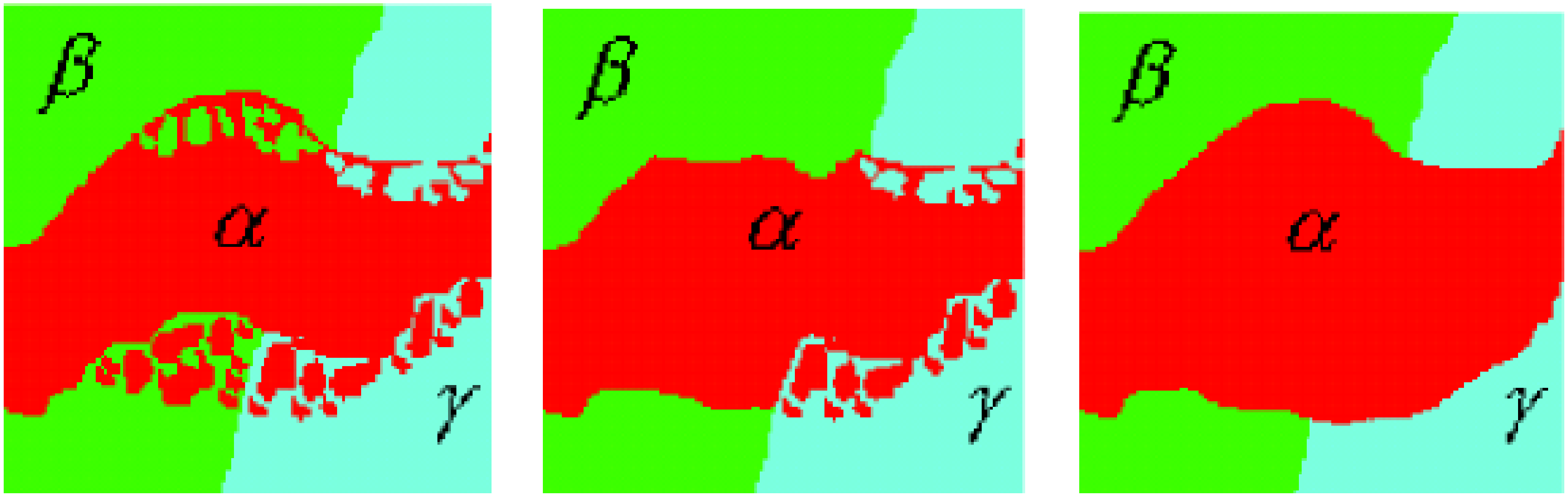


Graph Cut Algorithm

α-β swap: interchange α and β labels



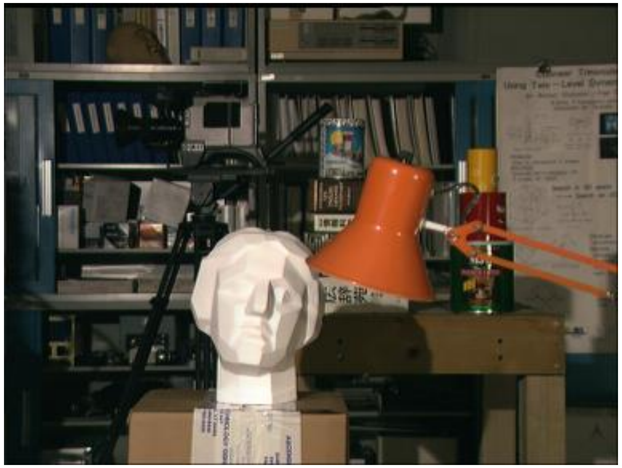
Graph Cut Algorithms

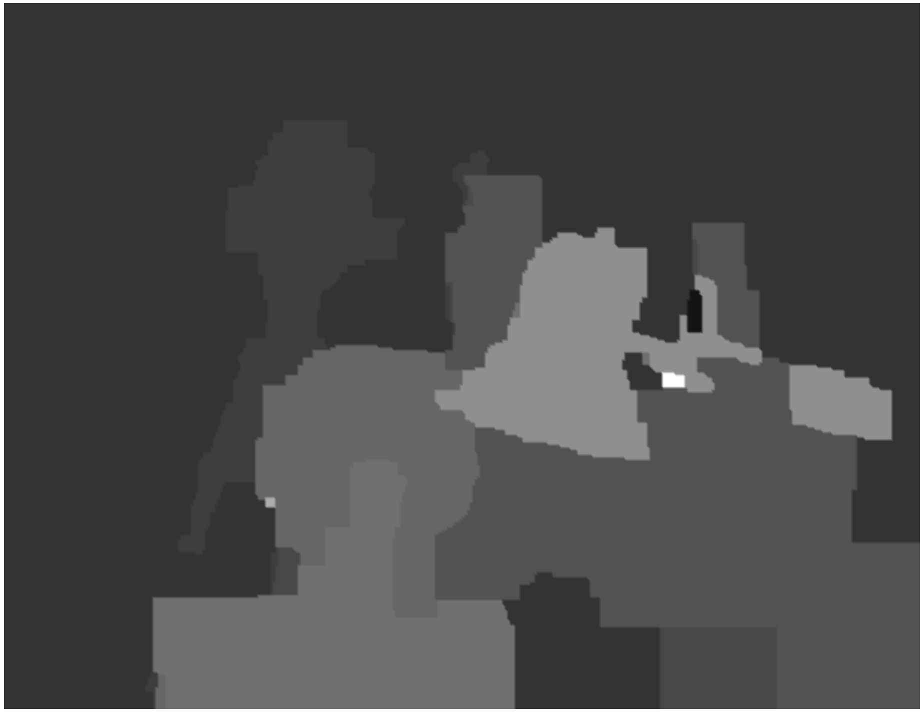


initial labeling α-β-swap α-expansion

Boykov

Graph Cut Results



Stereo evaluation

vision.middlebury.edu

stereo • mview • MRF • flow

Stereo Evaluation • Datasets • Code • Submit

Daniel Scharstein • Richard Szeliski

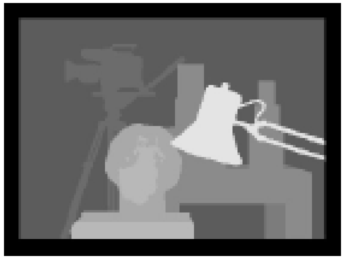
Welcome to the Middlebury Stereo Vision Page, formerly located at www.middlebury.edu/stereo. This website accompanies our taxonomy and comparison of two-frame stereo correspondence algorithms [1]. It contains:

- An on-line evaluation of current algorithms
- Many stereo datasets with ground-truth disparities
- Our stereo correspondence software
- An on-line submission script that allows you to evaluate your stereo algorithm in our framework

How to cite the materials on this website:
We grant permission to use and publish all images and numerical results on this website. If you report performance results, we request that you cite our paper [1]. Instructions on how to cite our datasets are listed on the datasets page. If you want to cite this website, please use the URL "**vision.middlebury.edu/stereo/**".

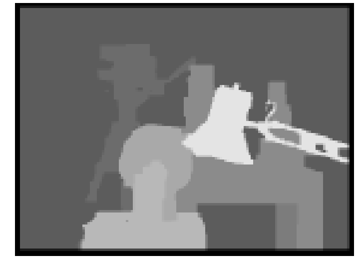
References:

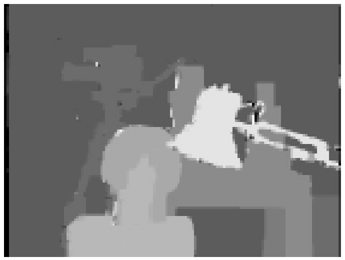
[1] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1/2/3):7-42, April-June 2002. Microsoft Research Technical Report MSR-TR-2001-81, November 2001.

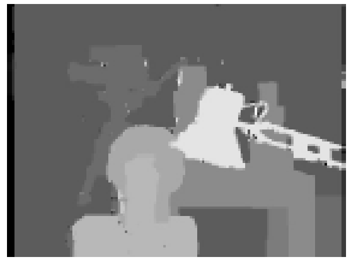

True disparities


19 – Belief propagation

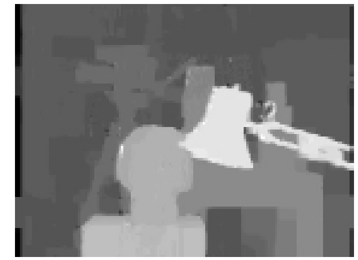

11 – GC + occlusions


20 – Layered stereo

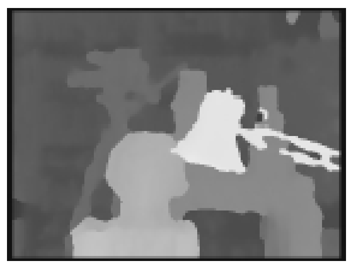

10 – Graph cuts


*4 – Graph cuts


13 – Genetic algorithm


6 – Max flow


12 – Compact windows


9 – Cooperative alg.


15 – Stochastic diffusion


*2 – Dynamic progr.


14 – Realtime SAD


*3 – Scanline opt.


7 – Pixel-to-pixel stereo


*1 – SSD+MF

Scharstein and Szeliski

Stereo—best algorithms

Error Threshold = 1

Error Threshold...

Sort by nonocc | Sort by all | Sort by disc

Algorithm	Avg. Rank	Tsukuba ground truth			Venus ground truth			Teddy ground truth			Cones ground truth		
		nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc
AdaptingBP [17]	2.8	1.11 6	1.37 3	5.79 7	**0.10** 1	0.21 2	**1.44** 1	4.22 4	7.06 2	11.8 4	**2.48** 1	7.92 2	**7.32** 1
DoubleBP2 [35]	2.9	**0.88** 1	**1.29** 1	**4.76** 1	0.13 3	0.45 5	1.87 5	3.53 2	8.30 3	**9.63** 1	2.90 3	8.78 8	7.79 2
DoubleBP [15]	4.9	0.88 2	1.29 2	4.76 2	0.14 5	0.60 13	2.00 7	3.55 3	8.71 5	9.70 2	2.90 4	9.24 11	7.80 3
SubPixDoubleBP [30]	5.6	1.24 10	1.76 13	5.98 8	0.12 2	0.46 6	1.74 4	**3.45** 1	8.38 4	10.0 3	2.93 5	8.73 7	7.91 4
AdaptOvrSegBP [33]	9.9	1.69 22	2.04 21	5.64 6	0.14 4	**0.20** 1	1.47 2	7.04 14	11.1 7	16.4 11	3.60 11	8.96 10	8.84 10
SymBP+occ [7]	10.8	0.97 4	1.75 12	5.09 4	0.16 6	0.33 3	2.19 8	6.47 8	10.7 6	17.0 14	4.79 24	10.7 21	10.9 20
PlaneFitBP [32]	10.8	0.97 5	1.83 14	5.26 5	0.17 7	0.51 8	1.71 3	6.65 9	12.1 13	14.7 7	4.17 20	10.7 20	10.6 19
AdaptDispCalib [36]	11.8	1.19 8	1.42 4	6.15 9	0.23 9	0.34 4	2.50 11	7.80 19	13.6 21	17.3 17	3.62 12	9.33 12	9.72 15
Segm+visib [4]	12.2	1.30 15	1.57 5	6.92 18	0.79 21	1.06 18	6.76 22	5.00 5	**6.54** 1	12.3 5	3.72 13	8.62 6	10.2 17
C-SemiGlob [19]	12.3	2.61 29	3.29 24	9.89 27	0.25 12	0.57 10	3.24 15	5.14 6	11.8 8	13.0 6	2.77 2	8.35 4	8.20 5
SO+borders [29]	12.8	1.29 14	1.71 9	6.83 15	0.25 13	0.53 9	2.26 9	7.02 13	12.2 14	16.3 9	3.90 15	9.85 16	10.2 18
DistinctSM [27]	14.1	1.21 9	1.75 11	6.39 11	0.35 14	0.69 16	2.63 13	7.45 18	13.0 17	18.1 19	3.91 16	9.91 18	8.32 7
CostAggr+occ [39]	14.3	1.38 17	1.96 17	7.14 19	0.44 16	1.13 19	4.87 19	6.80 11	11.9 10	17.3 16	3.60 10	8.57 5	9.36 13

Stereo Summary

Advantages:

- cheap hardware, passive
- works very well in non-occluded regions

Disadvantages:

- gets confused in texture-less regions
- gets confused in occluded regions
- gets confused by specular surfaces