

CS4442/9542b
Artificial Intelligence II
prof. Olga Veksler

Lecture 11
Computer Vision
Stereo

Some slides are from S. Seitz, S. Narasimhan, K. Grauman

Outline

- Cues for 3D reconstruction
- Stereo Cues
- Stereo Reconstruction
 - 1) camera calibration and rectification
 - an easier, mostly solved problem
 - 2) stereo correspondence
 - a harder problem

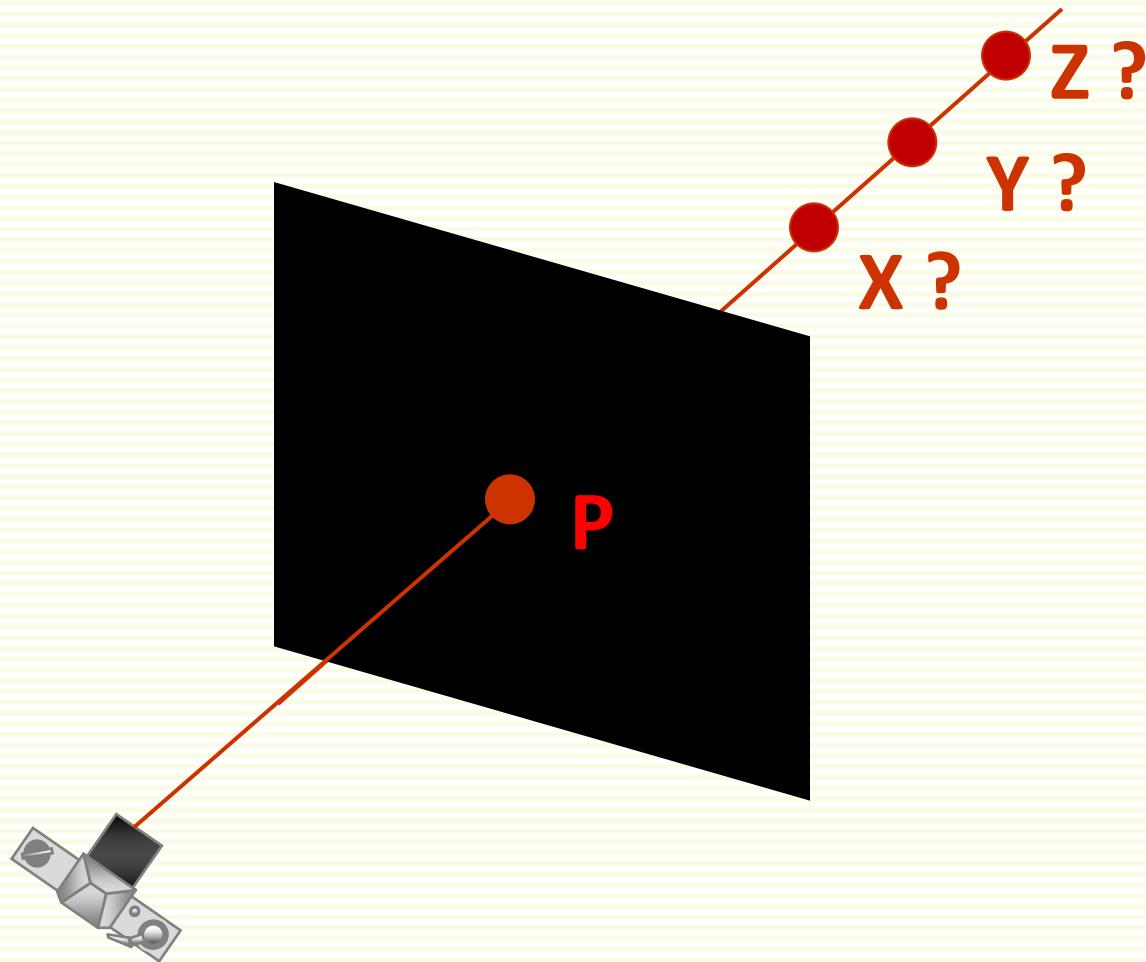
2D Images

- World is 3D
- In 2D images, depth (the third coordinate) is largely lost
 - includes human retina



2D Images

- Depth is inherently ambiguous from a single view



Street Pavement Art

- Viewed from the “right” side



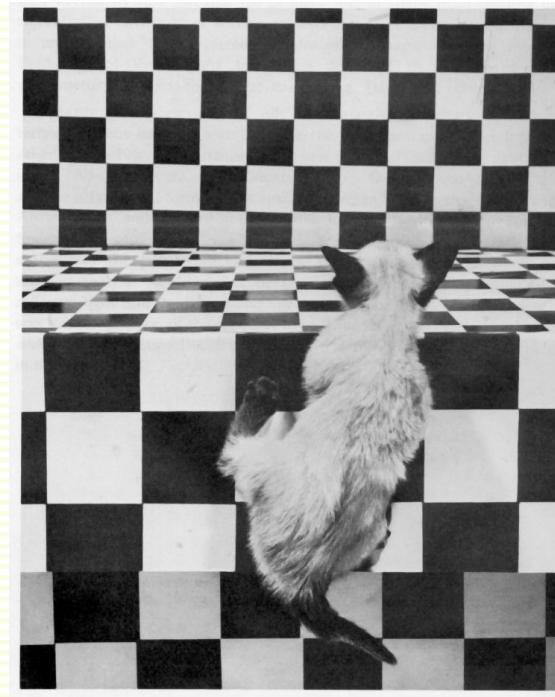
Street Pavement Art

- Viewed from the “wrong” side



Babies and Animals Perceive Depth

- Yet we perceive the world in 3D



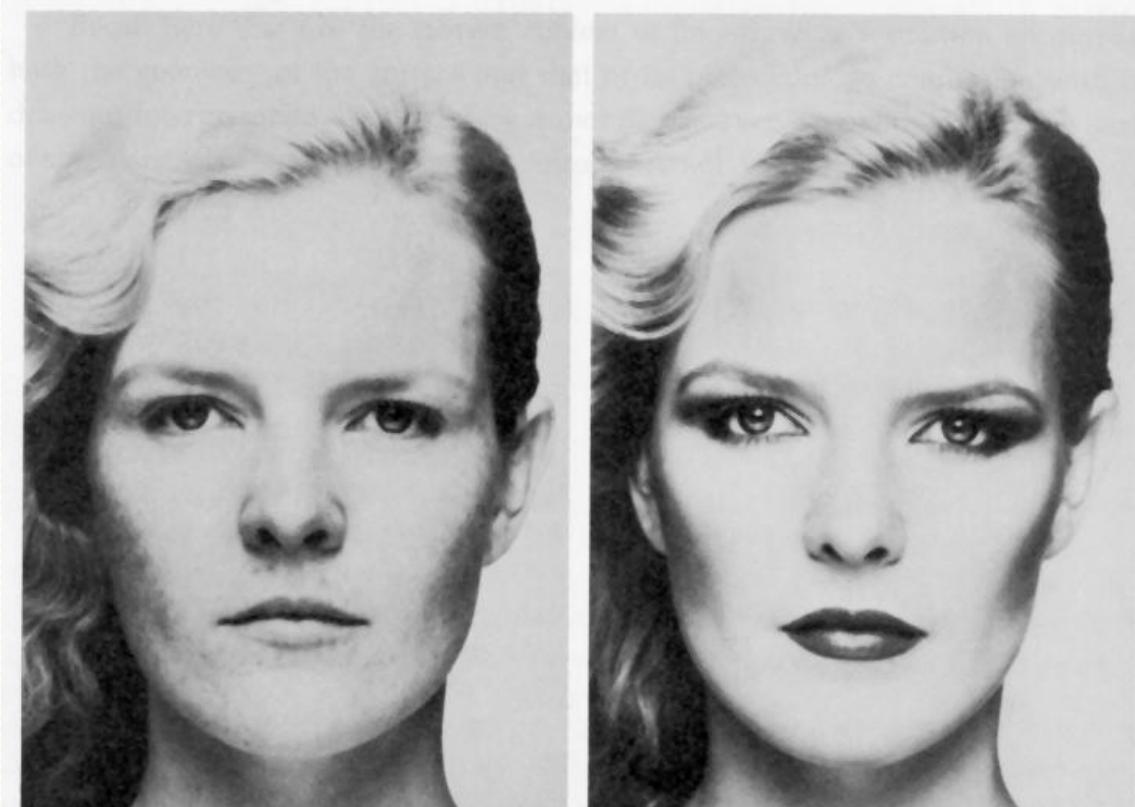
The Visual Cliff, by William Vandivert, 1960

3D Shape from Images

- What image cues provide 3D information?
- Cues from a single image
- Cues from multiple images
 - Motion cues
 - Stereo cues
- Can we use these cues in a computer vision system?

Single Image 3D Cues: Shading

- Pixels covered by shadow are perceived to be further away



Merle Norman Cosmetics, Los Angeles

Single Image 3D Cues: Linear Perspective

- The further away are parallel lines, the closer they come together



Single Image 3D Cues: Relative Size

- If objects have the same size, those further away appear smaller



Single Image 3D Cues: Texture

- Further away texture appears finer (smaller scale)



Single Image 3D Cues: Known Size

- Ducks are smaller than elephants, duck is closer

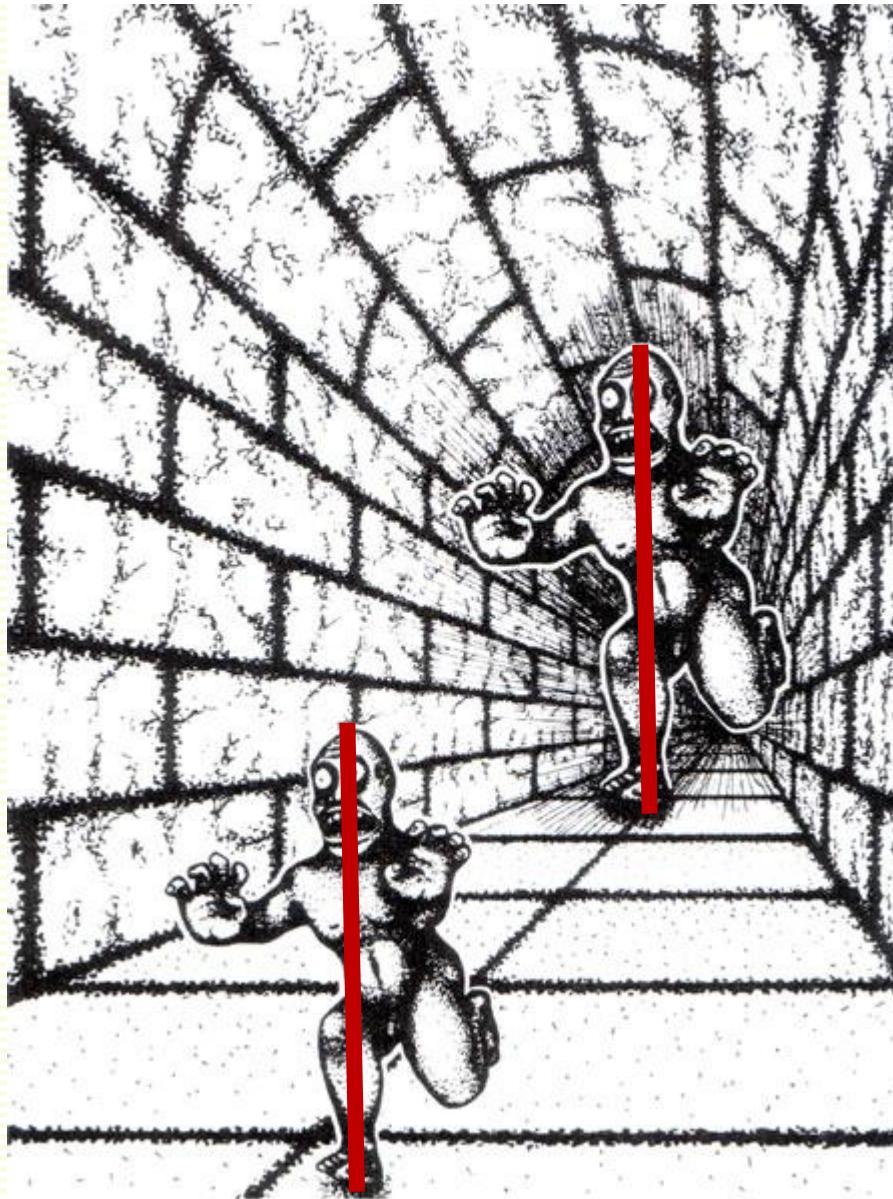


Illusions: Linear Perspective + Relative Size



ThePeople'sCube.com

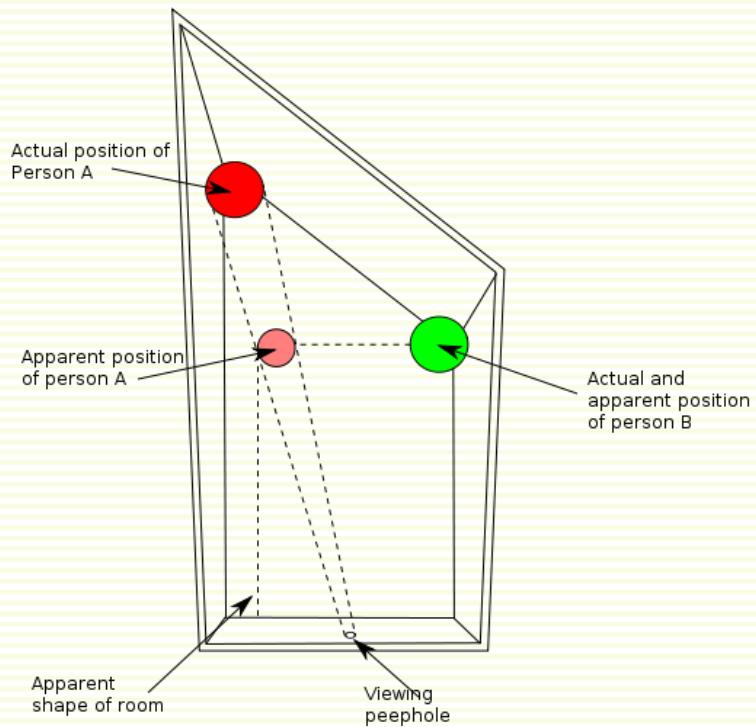
Illusions: Linear Perspective + Relative Size



Illusions: Linear Perspective + Relative Size

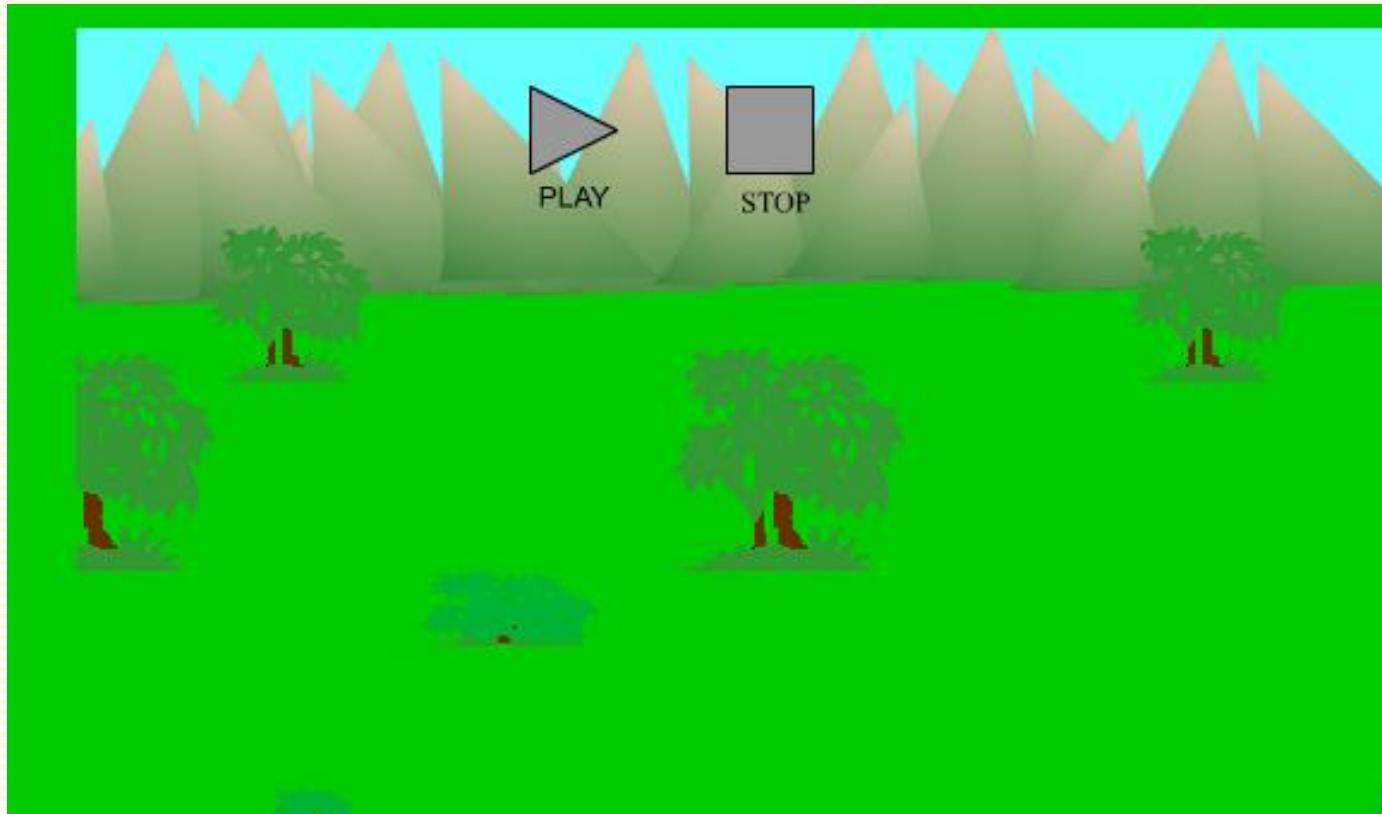


Illusions: Ames Room



Cues from Multiple Image: Motion Parallax

- Closer objects appear to move more than further away objects

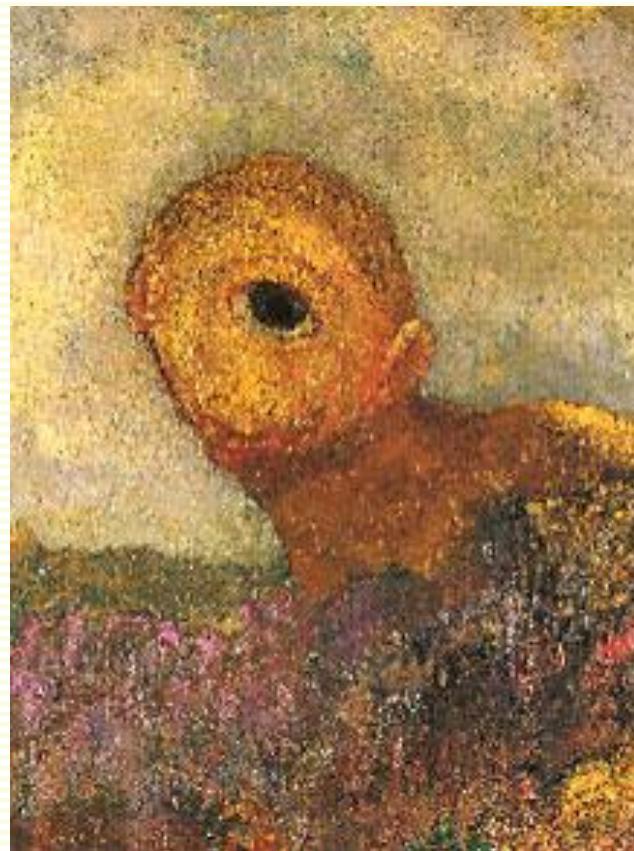


<http://psych.hanover.edu/KRANTZ/MotionParallax/MotionParallax.html>

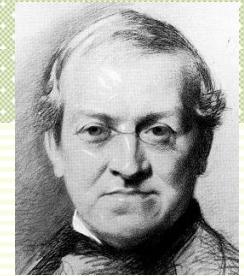
3D Shape from **X**

- **X** = shading, texture, motion, ...
- We will focus on **stereo**
 - depth perception from two **stereo images**

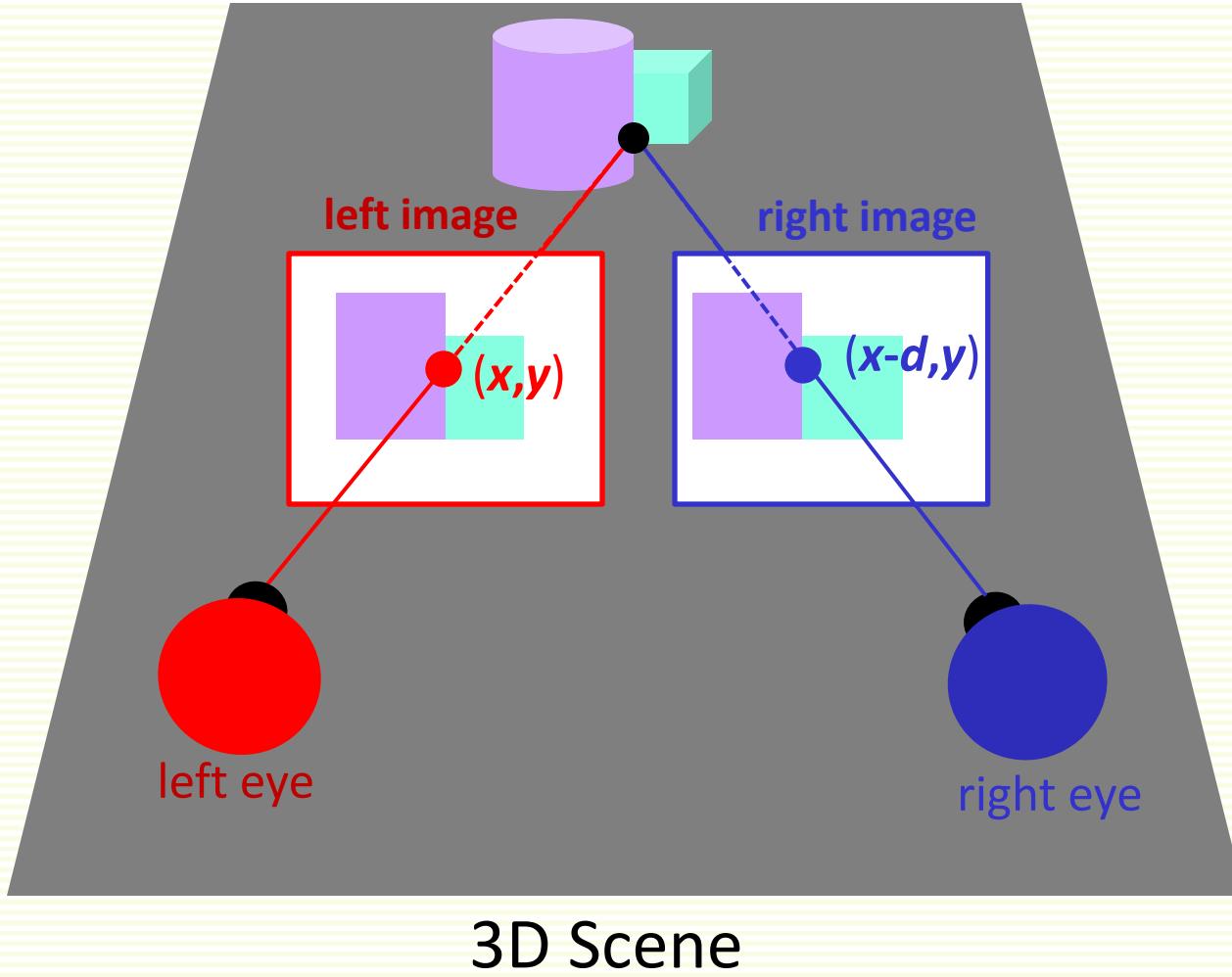
Why Two Eyes? Cylopes?



Why Two Eyes?

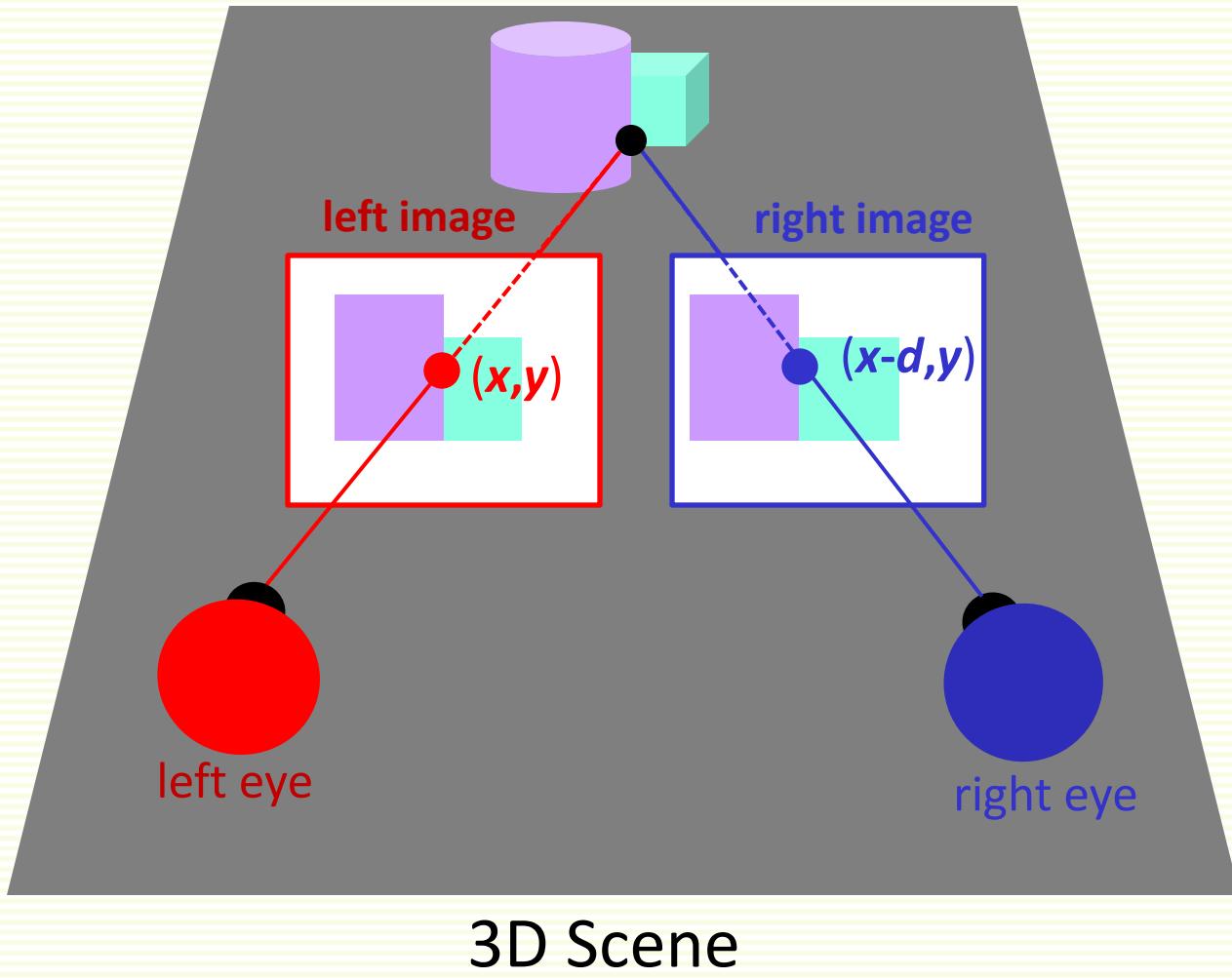


- Charles Wheatstone first explained stereopsis in 1838



Why Two Eyes?

- **Disparity d** is the difference in x coordinates of corresponding points



Stereoscopes

- Wheatstone invented the first stereoscope



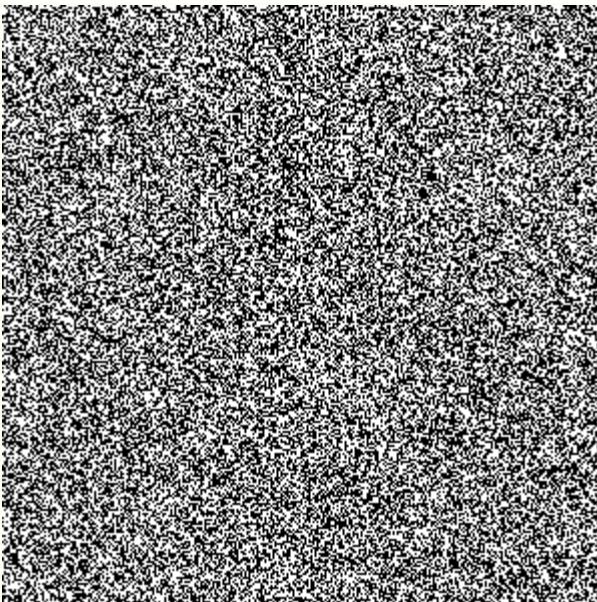
Anaglyph Images

- Encodes left and right image into a single picture
 - left eye image is transferred to the **red** channel
 - right eye image to the **green+blue** = **cyan** channel
- **Red** filter lets through only the left image
- **Cyan** filter lets through only the right eye image
- Brain fuses into 3D
- Similar technology for 3D movies
- Works for most of us



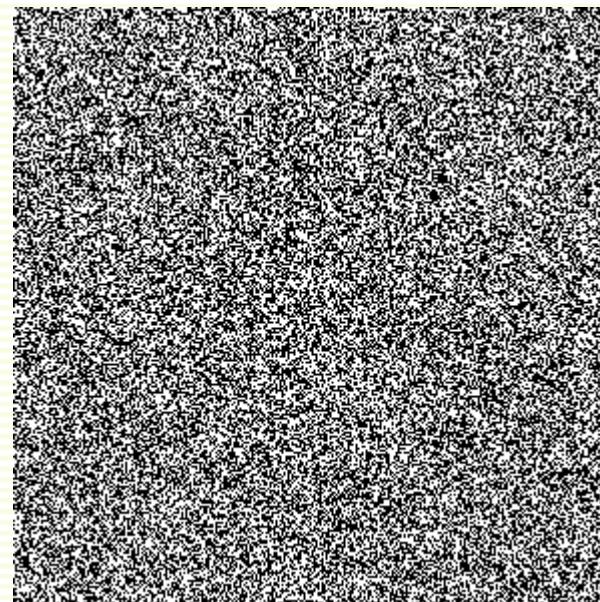
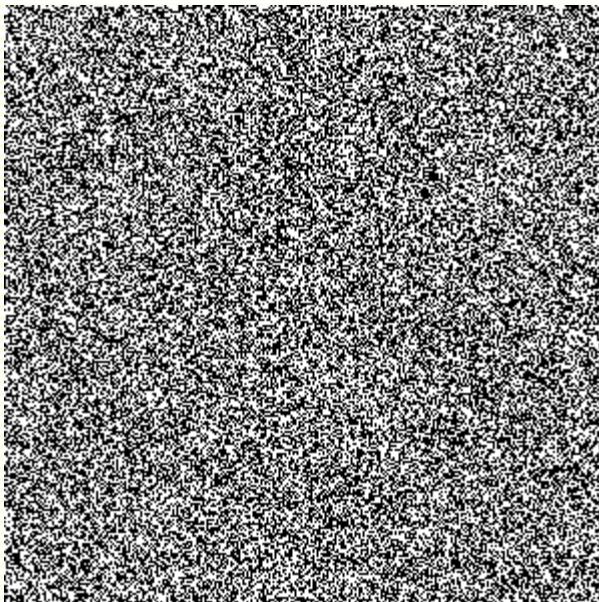
What is Needed for Stereopsis?

- Need monocular cues for stereopsis? Need object cues?
Answered by Julesz in 1960
- Image with no monocular cues and no recognizable objects: random dots



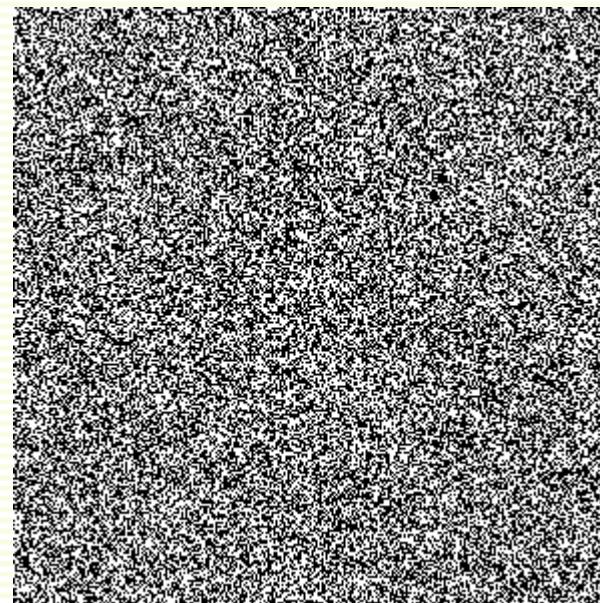
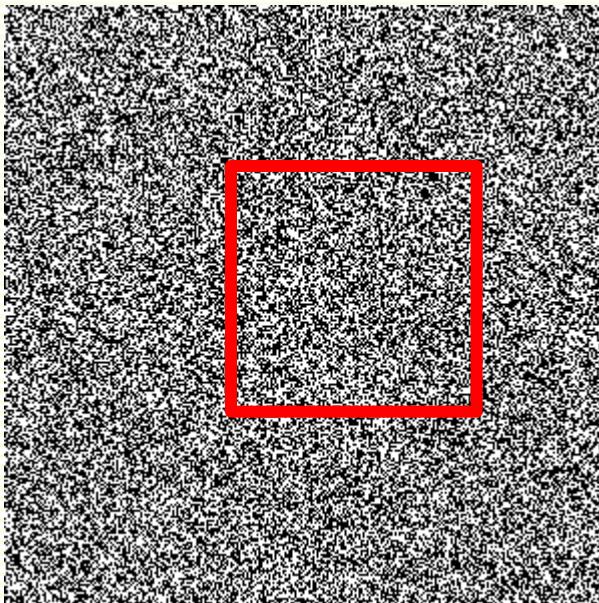
Need Object Recognition for Stereopsis?

- Answered by Julesz in 1960
- Make a copy of it



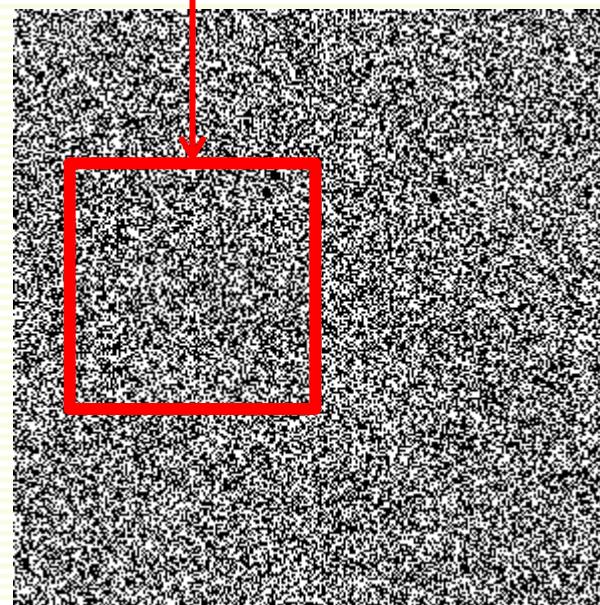
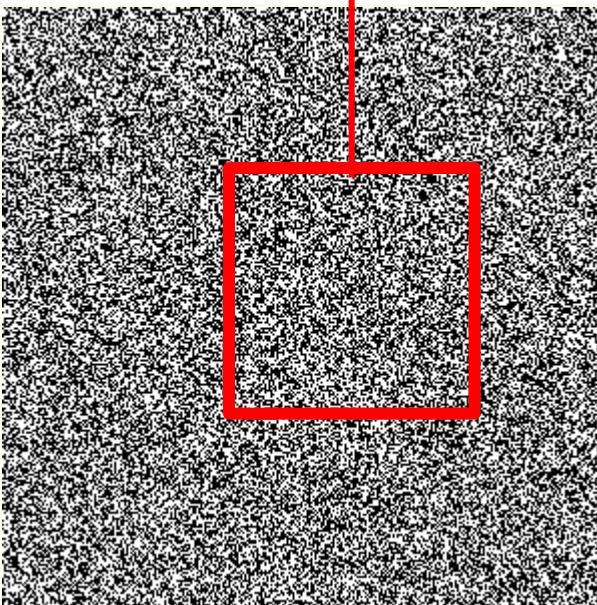
Need Object Recognition for Stereopsis?

- Answered by Julesz in 1960
- Select a square



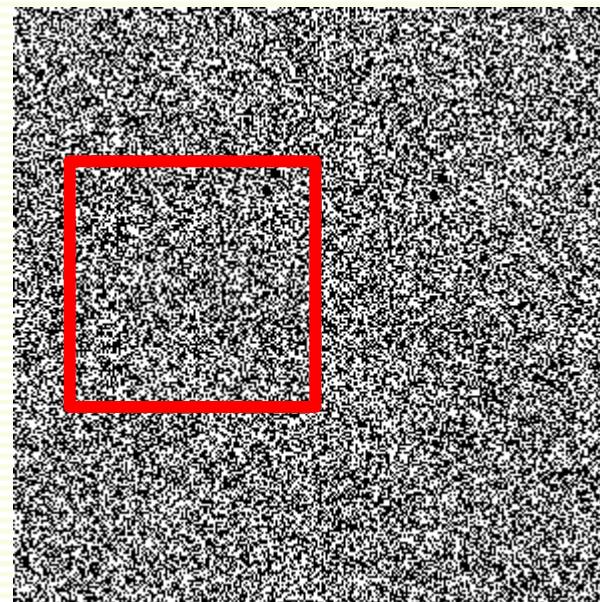
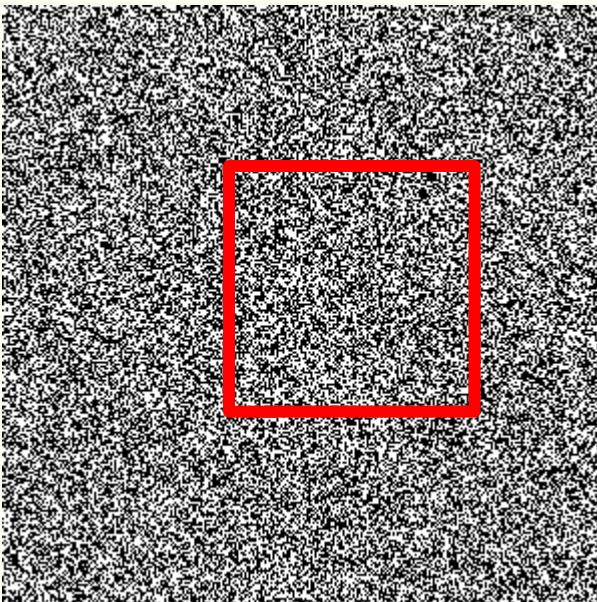
Need Object Recognition for Stereopsis?

- Answered by Julesz in 1960
- Copy square the right image, shifting by d to the left
 - random dot stereogram



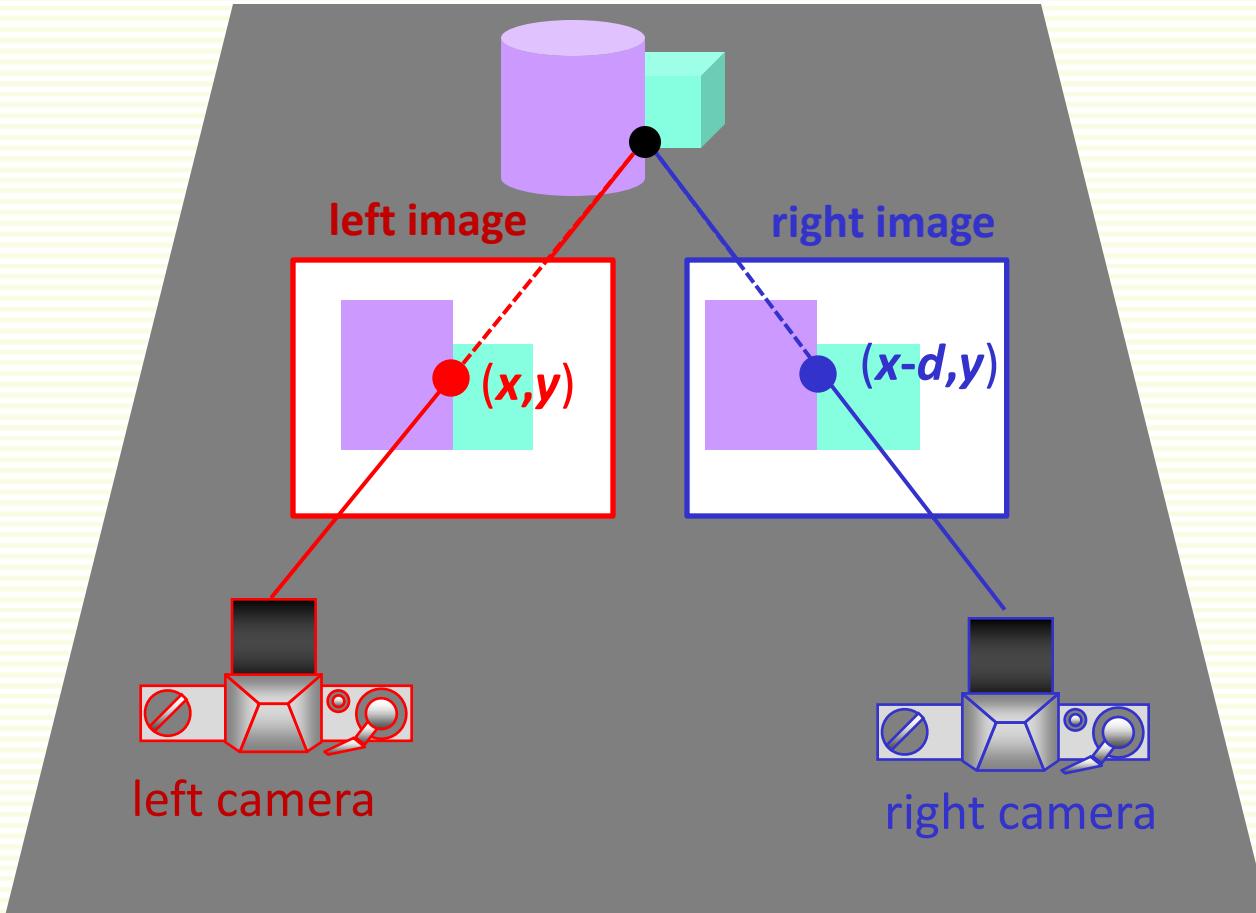
Need Object Recognition for Stereopsis?

- Answered by Julesz in 1960
- Random dot stereogram
- Humans perceive square floating in front of background



3D Shape from Stereo

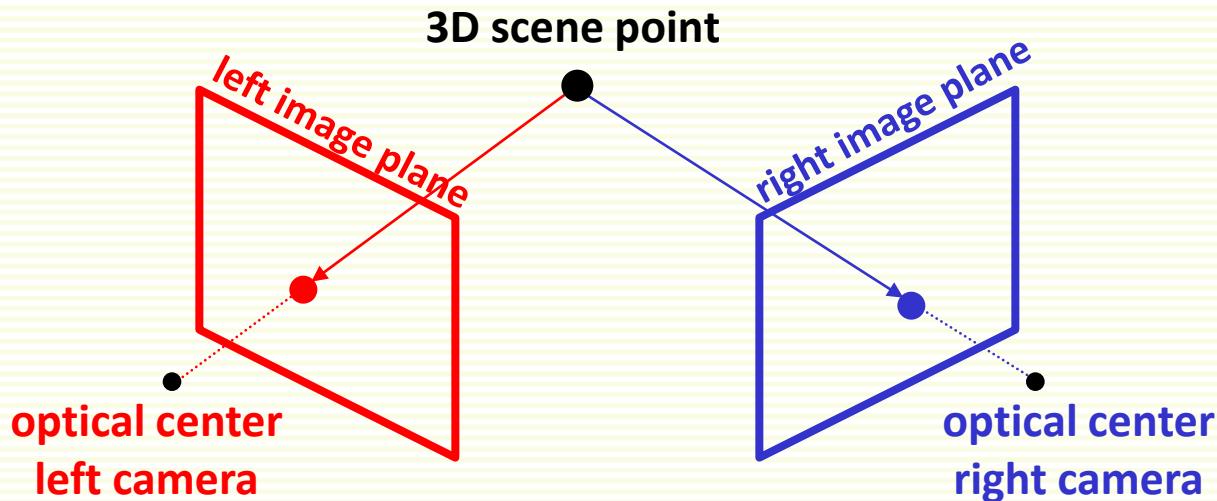
- Use two cameras instead of two eyes



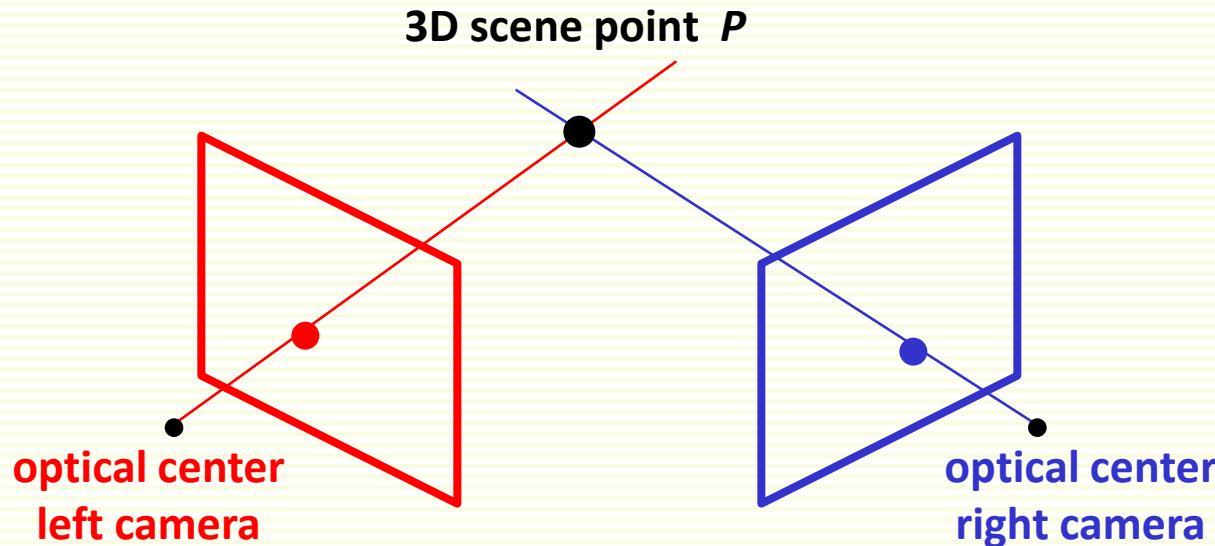
3D Scene

Stereo System

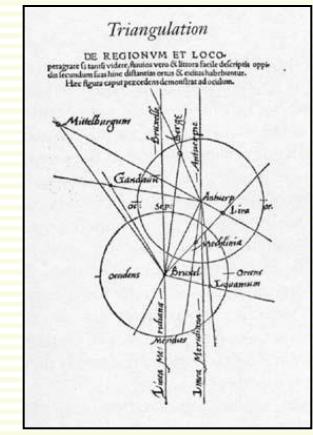
- Unlike eyes, usually stereo cameras are not on the same plane
 - better numerical stability



Stereo System: Triangulation



- Depth by triangulation
 - given two corresponding points in the left and right image
 - cast the rays through the optical camera centers
 - ray intersection is the corresponding 3D world point P
 - depth of P is based on camera positions and parameters
- Triangulation ideas can be traced to ancient Greece



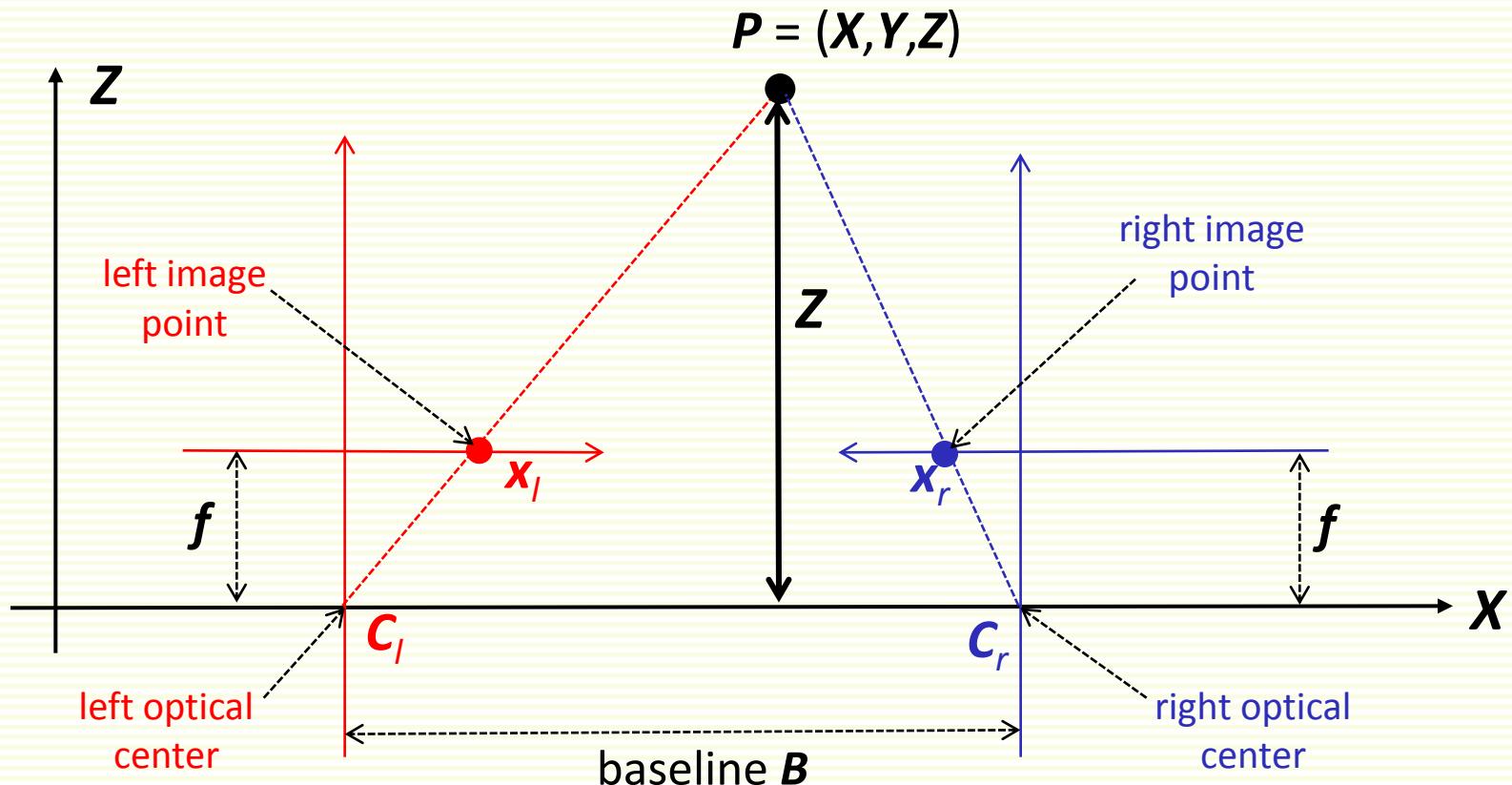
document from 1533

What is needed for Triangulation

1. Distance between cameras, camera focal length
 - Solved through **camera calibration**, essentially a solved problem
 - We will not talk about it
 - Code available on the web
 - OpenCV <http://www.intel.com/research/mrl/research/opencv/>
 - Matlab, J. Bouget http://www.vision.caltech.edu/bouguetj/calib_doc/index.html
 - Zhengyou Zhang <http://research.microsoft.com/~zhang/Calib/>
2. Pairs of corresponding pixels in left and right images
 - Called **stereo correspondence problem**, still much researched

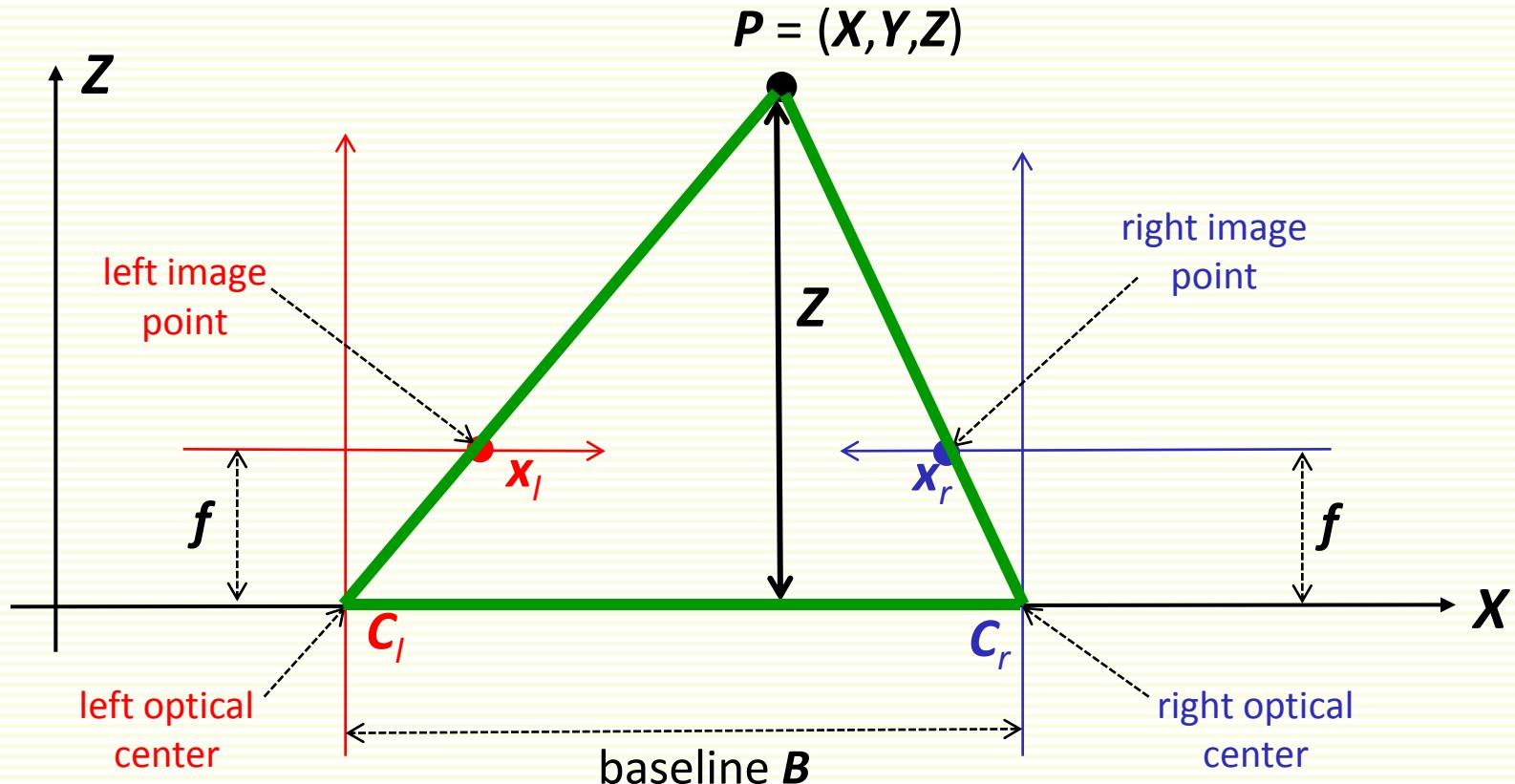
Formula: Depth from Disparity

- Top down view on geometry (slice through XZ plane)
 - from camera calibration, know the distance between camera optical centers called **baseline B** , and camera focal length f



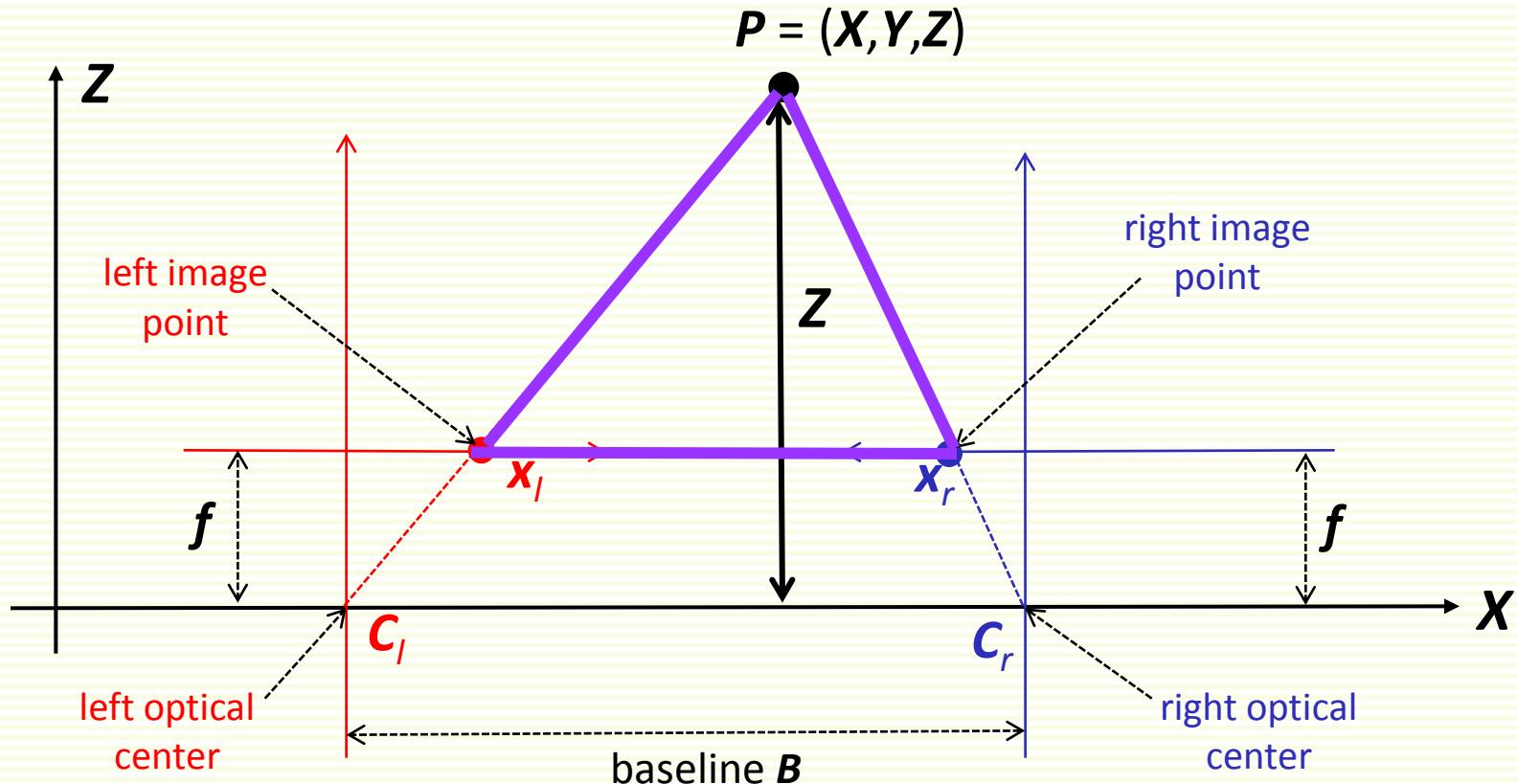
Formula: Depth from Disparity

- Height to base ratio of triangle $C_l P C_r$: $\frac{Z}{B}$



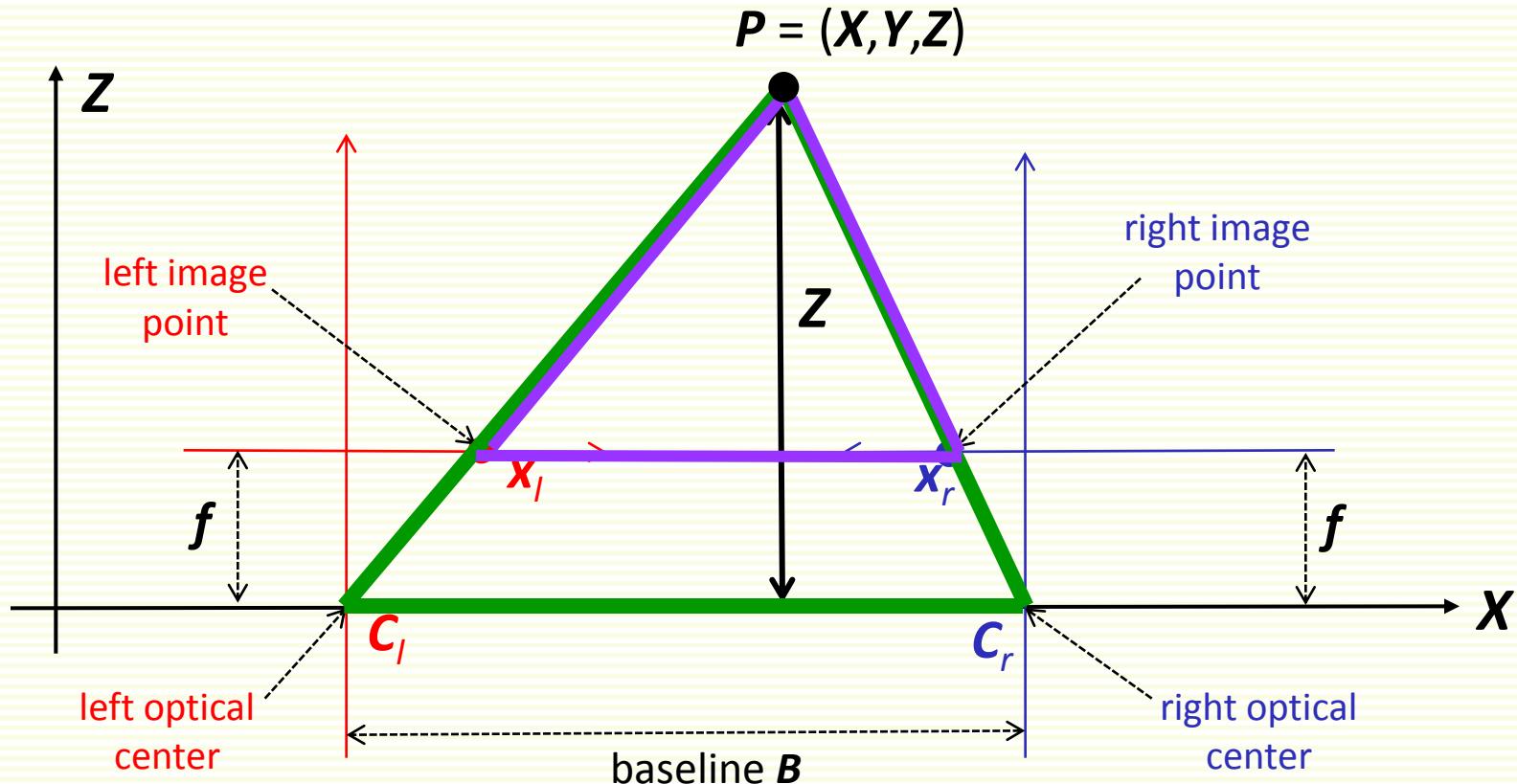
Formula: Depth from Disparity

- Height to base ratio of triangle $x_l P x_r$: $\frac{Z - f}{B - x_l + x_r}$
- x_l is positive, x_r is negative



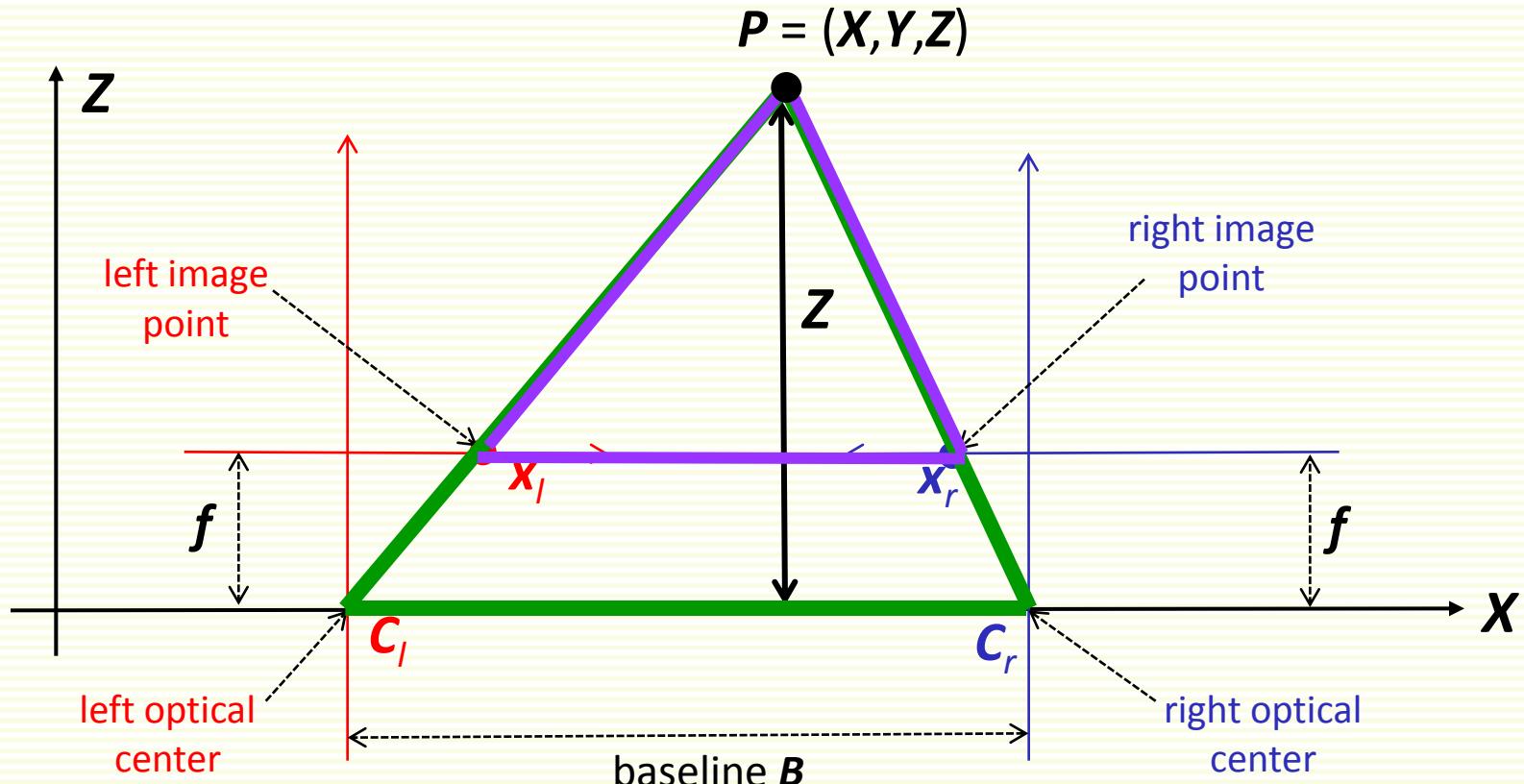
Formula: Depth from Disparity

- $C_l P C_r$ and $\Delta x_l P x_r$ are similar:
$$\frac{Z}{B} = \frac{Z - f}{B - x_l + x_r}$$



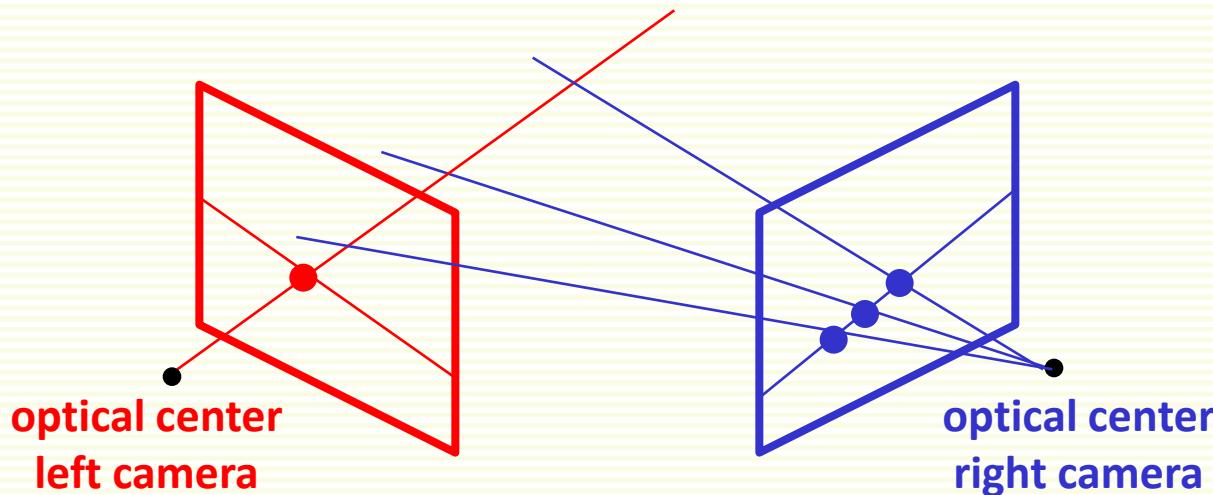
Formula: Depth from Disparity

- Rewriting: $z = \frac{B \cdot f}{x_l - x_r}$
- $x_l - x_r$ is the **disparity**



Stereo Correspondence: Epipolar Lines

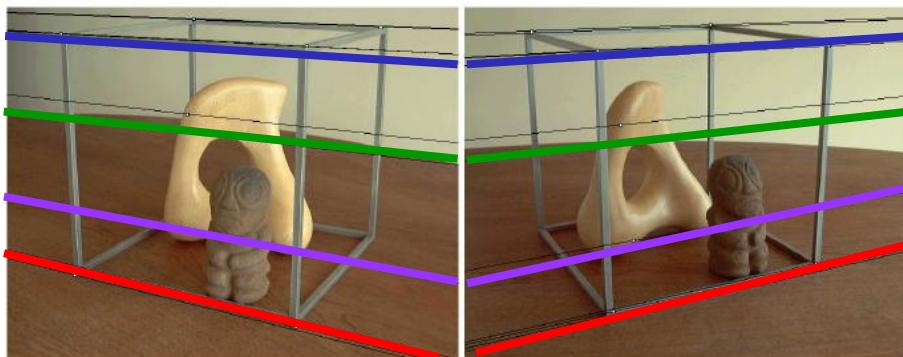
- Which pairs of pixels correspond to the same scene element ?



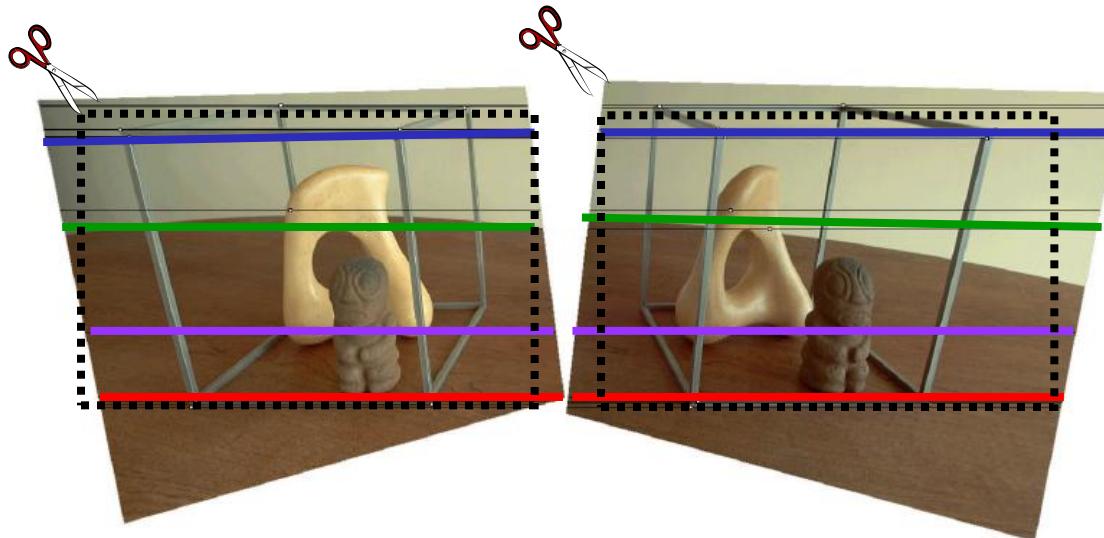
- Epipolar constraint
 - Given a left image pixel, the corresponding pixel in the right image must lie on a line called the **epipolar** line
 - reduces correspondence to 1D search along **conjugate** epipolar lines
 - demo: <http://www.ai.sri.com/~luong/research/Meta3DViewer/EpipolarGeo.html>

Stereo Rectification

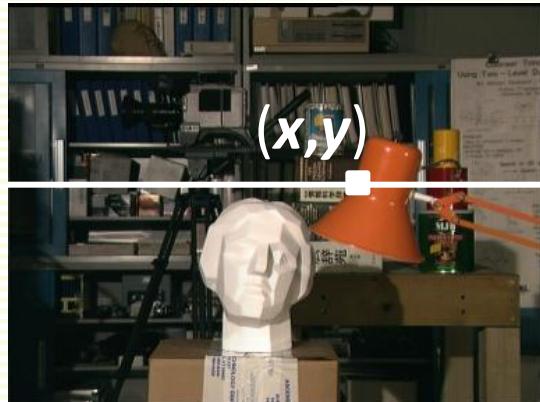
- Epipolar lines can be computed from camera calibration



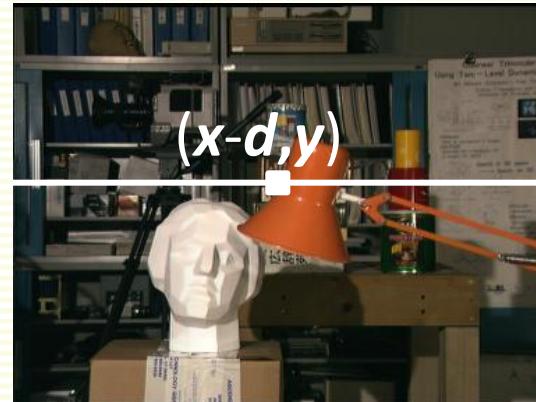
- Usually they are not horizontal
- Can **rectify** stereo pair to make epipolar lines horizontal



Stereo Correspondence



left image

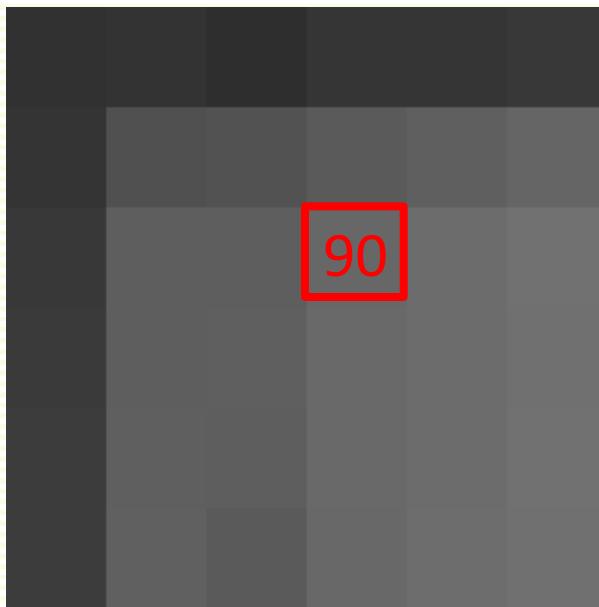


right image

- From now on assume stereo pair is rectified
- How to solve the correspondence problem?
- Corresponding pixels should be similar in intensity
 - or color, or something else

Difficulties in Stereo Correspondence

- Image noise
 - corresponding pixels have similar, but not exactly the same intensities



left image patch

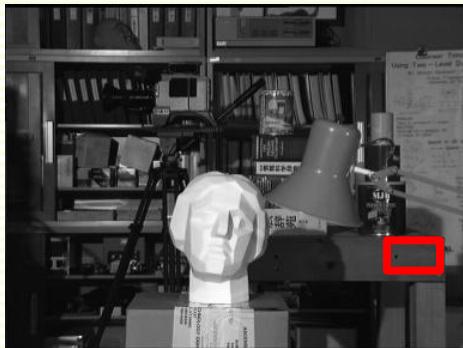


right image patch

- Matching each pixel individually is unreliable

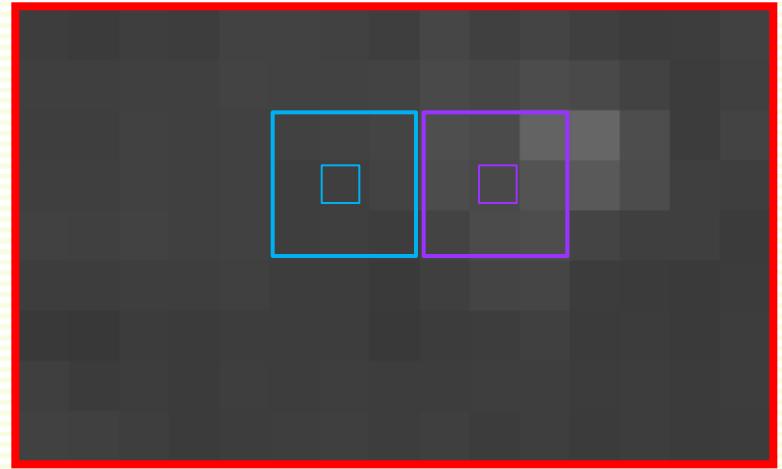
Difficulties in Stereo Correspondence

- regions with (almost) constant intensity



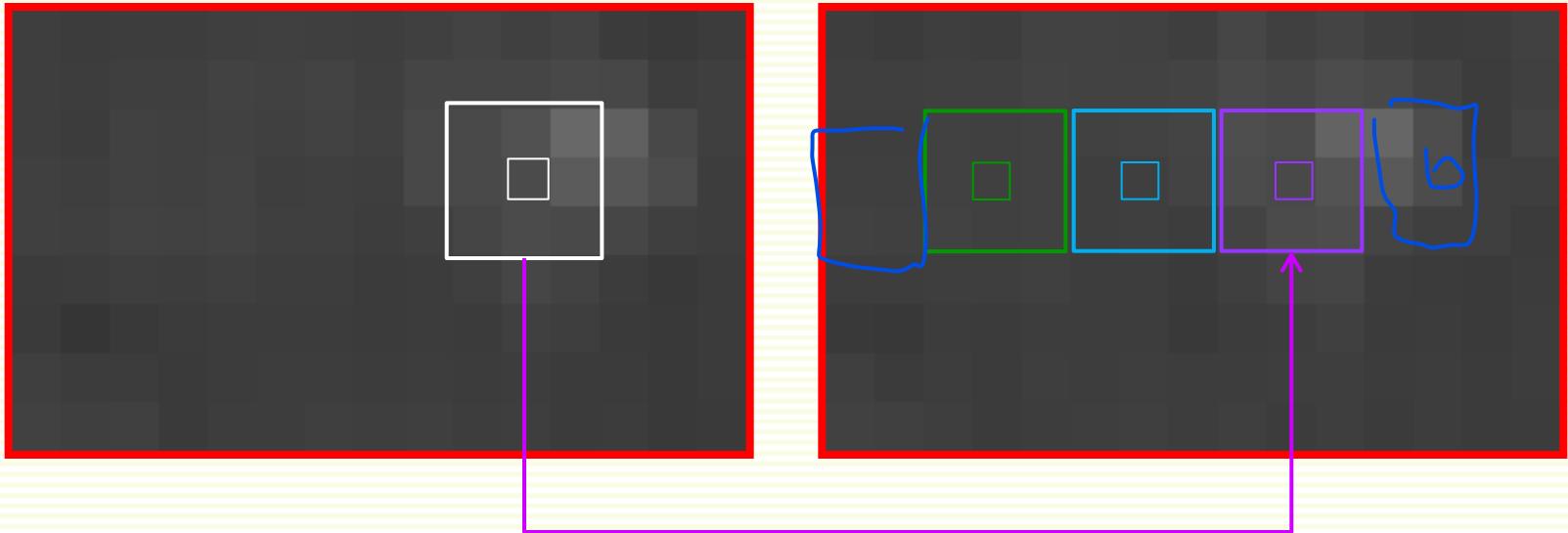
- Matching each pixel individually is unreliable

Window Matching Correspondence



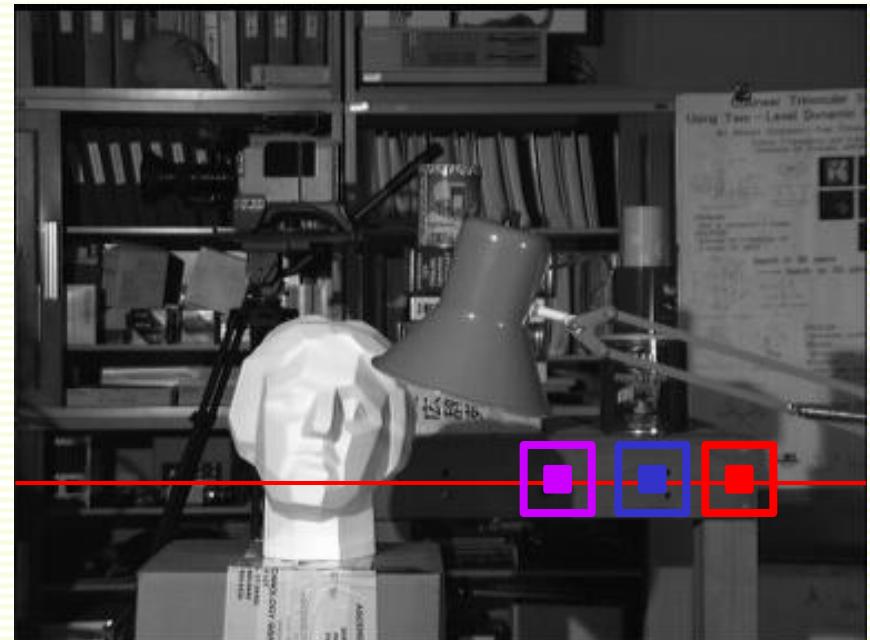
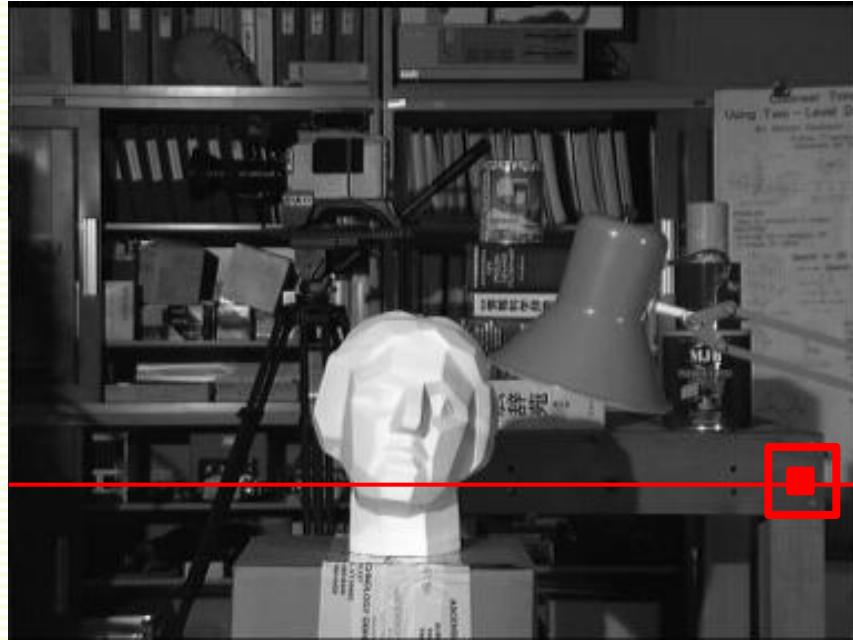
- Use a window (patch) of pixels
 - more likely to have enough intensity variation to form a distinguishable pattern
 - also more robust to noise

Window Matching Correspondence



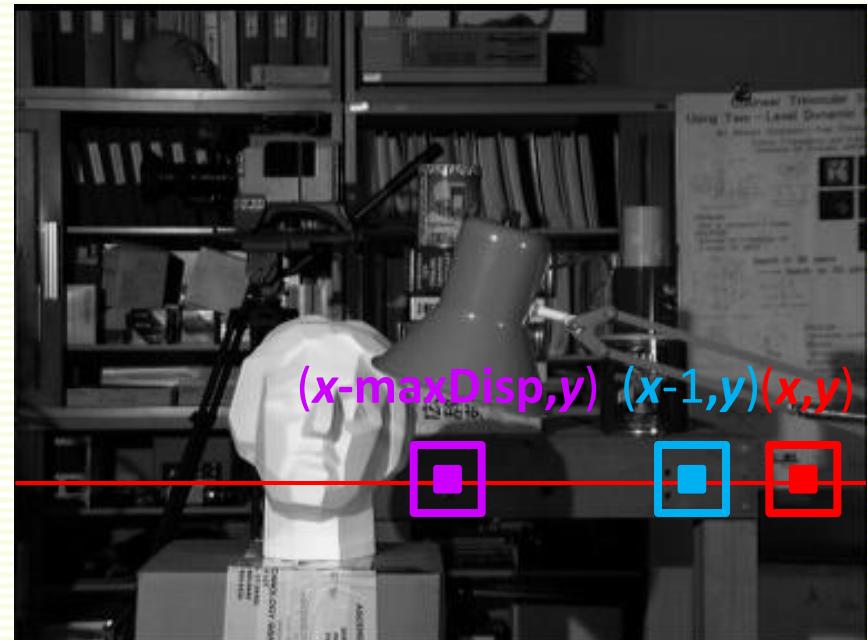
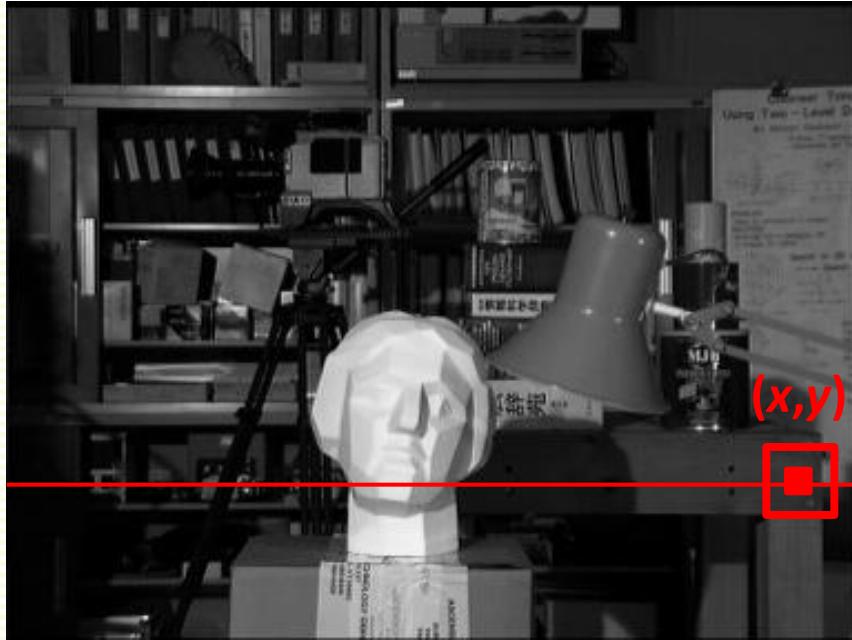
- Use a window (patch) of pixels
 - more likely to have enough intensity variation to form a distinguishable pattern
 - also more robust to noise

Window Matching: Basic Algorithm



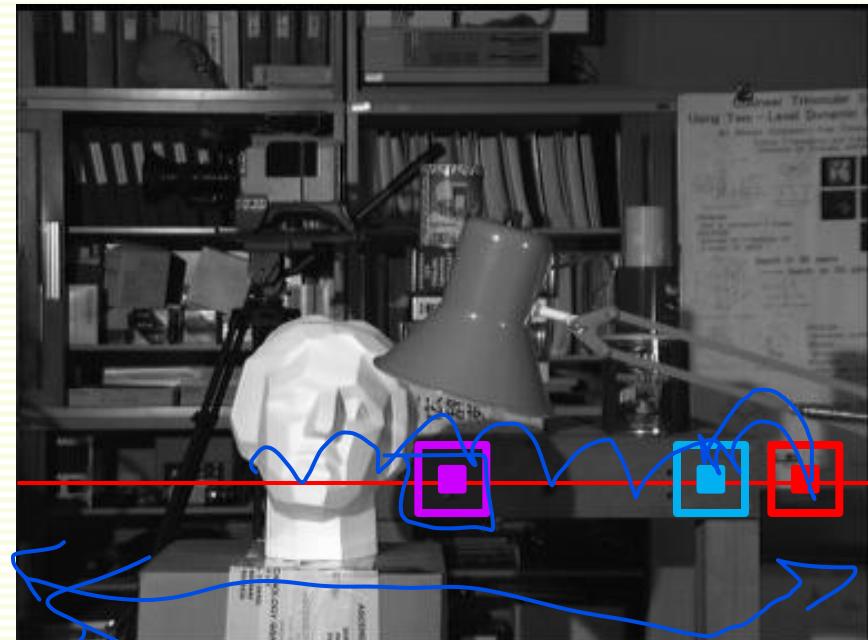
- for each epipolar line
 - for each pixel p on the left line
 - compare window around p with same window shifted to many right window locations on corresponding epipolar line
 - pick location corresponding to the best matching window

Which Locations to Try?



- Disparity cannot be negative
- Maximum possible disparity is limited by the camera setup
 - assume we know **maxDisp**
- Disparity can range from **0** to **maxDisp**
 - consider only (x,y) , $(x-1,y)$, ..., $(x-\text{maxDisp},y)$ in the right image

Window Matching Cost



- How to define the best matching window?
- Define window cost
 - sum of squared differences (SSD)
 - or sum of absolute differences (SAD)
 - many other possibilities
- Pick window of best (smallest) cost

1
33
... 33
for

SSD Window Cost

left image

3	5	4	4	2	4	2
7	4	1	4	4	2	6
2	7	46	46	46	6	7
5	9	46	46	44	9	7
4	7	47	47	47	2	4
4	7	56	56	46	6	7
3	4	4	1	4	3	2

right image

3	5	4	4	2	4	2
7	4	1	4	4	2	6
46	46	46	3	6	6	7
48	46	44	6	4	9	7
47	47	47	7	4	2	4
58	56	46	5	6	6	7
3	4	4	1	4	3	2

$$\begin{aligned} & (46-44)^2 + (46-6)^2 + (44-4)^2 + \\ & (47-47)^2 + (47-7)^2 + (47-4)^2 + \\ & (56-46)^2 + (56-5)^2 + (46-6)^2 = 12454 \end{aligned}$$

Algorithm with SSD Window Cost

left image							right image						
3	5	4	4	2	4	2	3	5	4	4	2	4	2
7	4	1	4	4	2	6	7	4	1	4	4	2	6
2	7	46	46	46	6	7	46	46	46	3	6	6	7
5	9	46	46	44	9	7	48	46	44	6	4	9	7
4	7	47	47	47	2	4	47	47	47	7	4	2	4
4	7	56	56	46	6	7	58	56	46	5	6	6	7
3	4	4	1	4	3	2	3	4	4	1	4	3	2

same position, then the disparity is 0

$$\begin{aligned} & (46-44)^2 + (46-6)^2 + (44-4)^2 + \\ & (47-47)^2 + (47-7)^2 + (47-4)^2 + \\ & (56-46)^2 + (56-5)^2 + (46-6)^2 = 12454 \end{aligned}$$

- This shift corresponds to disparity 0

Algorithm with SSD Window Cost

left image							right image						
3	5	4	4	2	4	2	3	5	4	4	2	4	2
7	4	1	4	4	2	6	7	4	1	4	4	2	6
2	7	46	46	46	6	7	46	46	46	3	6	6	7
5	9	46	46	44	9	7	48	46	44	6	4	9	7
4	7	47	47	47	2	4	47	47	47	7	4	2	4
4	7	56	56	46	6	7	58	56	46	5	6	6	7
3	4	4	1	4	3	2	3	4	4	1	4	3	2

$$\begin{aligned} & (46-46)^2 + (46-44)^2 + (44-6)^2 + \\ & (47-47)^2 + (47-7)^2 + (47-7)^2 + \\ & (56-56)^2 + (56-46)^2 + (46-5)^2 = 6425 \end{aligned}$$

- This shift corresponds to disparity 1

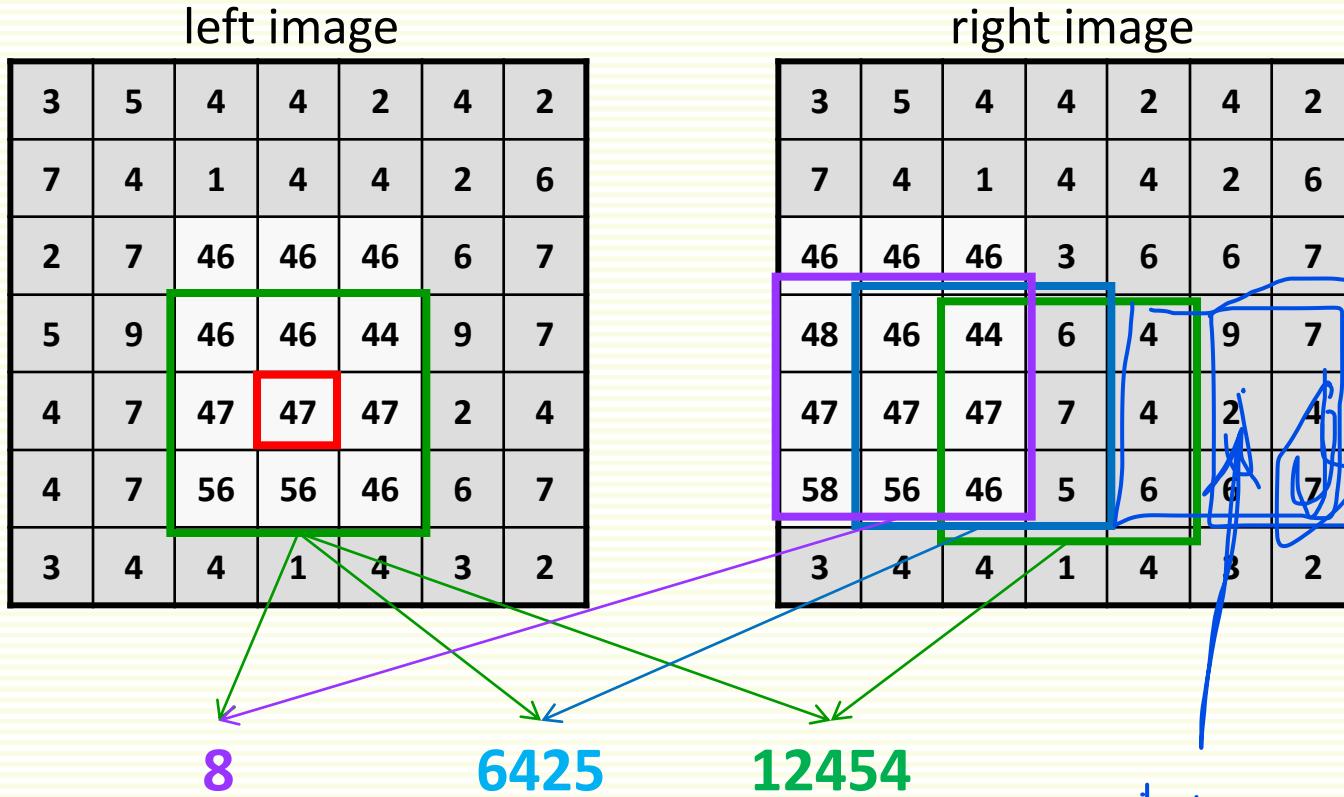
Algorithm with SSD Window Cost

left image							right image						
3	5	4	4	2	4	2	3	5	4	4	2	4	2
7	4	1	4	4	2	6	7	4	1	4	4	2	6
2	7	46	46	46	6	7	46	46	46	3	6	6	7
5	9	46	46	44	9	7	48	46	44	6	4	9	7
4	7	47	47	47	2	4	47	47	47	7	4	2	4
4	7	56	56	46	6	7	58	56	46	5	6	6	7
3	4	4	1	4	3	2	3	4	4	1	4	3	2

$$\begin{aligned} & (46-48)^2 + (46-46)^2 + (44-44)^2 + \\ & (47-47)^2 + (47-47)^2 + (47-47)^2 + \\ & (56-58)^2 + (56-56)^2 + (46-46)^2 = 8 \end{aligned}$$

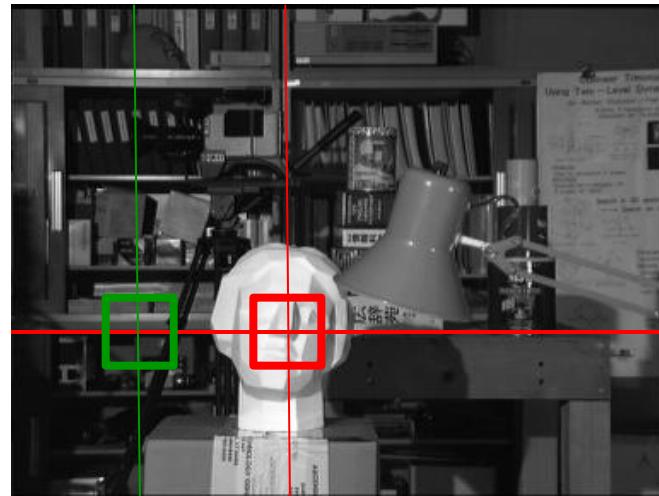
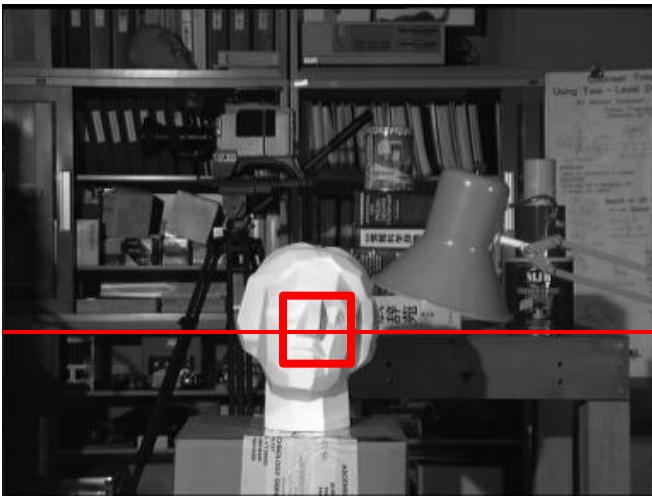
- This shift corresponds to disparity 2

Algorithm with SSD Window Cost

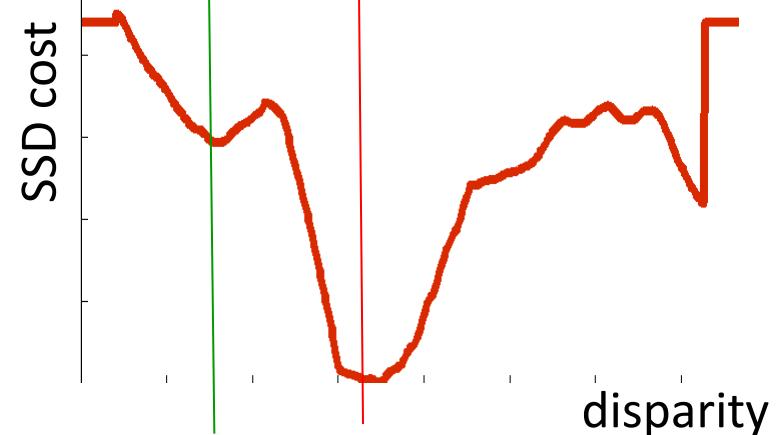


- Best SSD window cost is **8** at disparity **2**
- Red pixel is assigned disparity **2**
- Repeat this for all image pixels

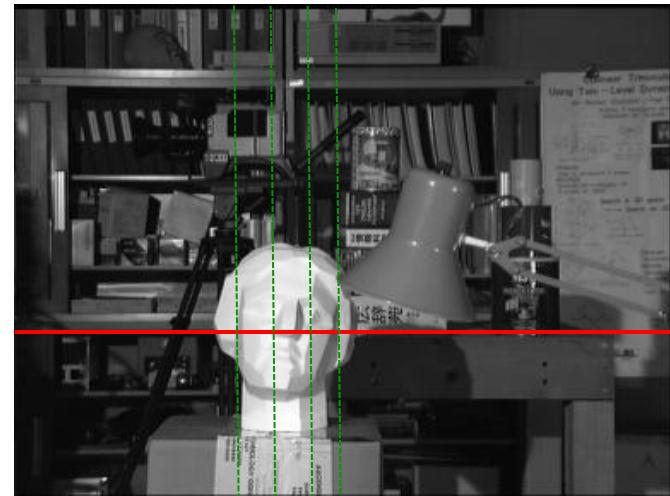
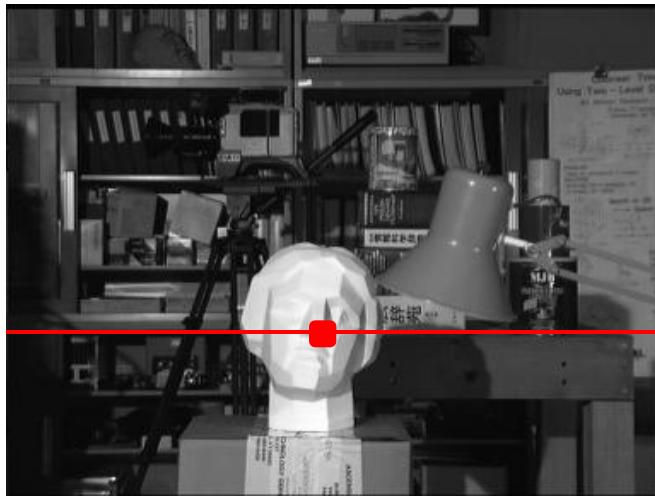
Correspondence with SSD Matching



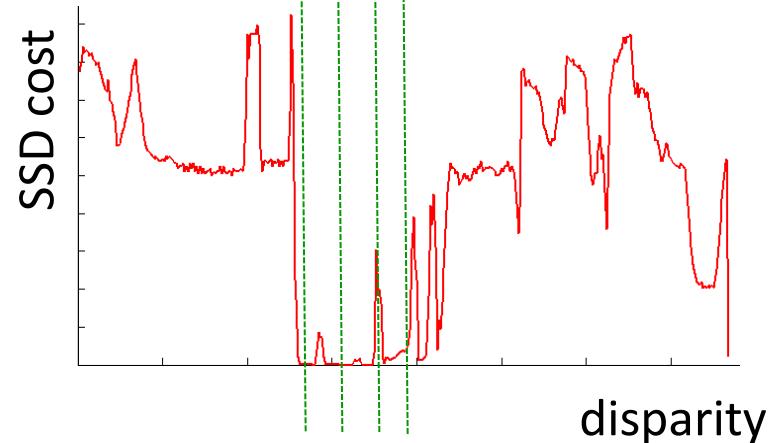
- Unique best cost location



Compare to One Pixel “Window”



- No unique best cost location



SAD Window Cost

- SSD is fragile to outliers

1	1	10
1	1	10
1	1	19

1	1	10
1	1	10
1	1	99

1	1	10
1	1	10
1	1	19

31	31	31
31	31	31
31	31	29

$$\text{SSD cost} = 80^2 = 6400$$

SSD cost = 6384  best

- SAD (Sum of Absolute Differences) is more robust

1	1	10
1	1	10
1	1	19

1	1	10
1	1	10
1	1	99

1	1	10
1	1	10
1	1	19

31	31	31
31	31	31
31	31	29

$$\text{SAD cost} = 80 \quad \checkmark \text{best}$$

$$\text{SAD cost} = 232$$

Window Matching Efficiency

- Suppose
 - image has n pixels
 - matching window is **11** by **11**
- Need **11·11 = 121** additions and multiplications to compute one window cost
- Multiply that by number of locations to check (**maxDisp+1**)
- Multiply that by n image pixels
- ~~$121 \cdot n \cdot (\text{maxDisp}+1)$~~
- Tooooooooooooow
 - gets worse for larger windows
- Can get cost down to $n \cdot (\text{maxDisp}+1)$ with integral images

Speedups: Integral Image

- Given image $f(x,y)$, the **integral** image $I(x,y)$ is the sum of values in $f(x,y)$ to the left and above (x,y) , including (x,y)

0	0	0	5	5
0	0	5	5	5
0	5	5	5	10
5	5	5	10	0
5	5	10	0	0

$f(x,y)$

0	0	0	5	10
0	0	5	15	25
0	5	15	30	50
5	15	30	55	75
10	25	50	75	95

$I(x,y)$

- Example: $I(2,2) = 0 + 0 + 0 + 0 + 0 + 5 + 0 + 5 + 5 = 15$

Speedups: Integral Image

- Given image $f(x,y)$, the **integral** image $I(x,y)$ is the sum of values in $f(x,y)$ to the left and above (x,y) , including (x,y)

0	0	0	5	5
0	0	5	5	5
0	5	5	5	10
5	5	5	10	0
5	5	10	0	0

$f(x,y)$

0	0	0	5	10
0	0	5	15	25
0	5	15	30	50
5	15	30	55	75
10	25	50	75	95

$I(x,y)$

- Example: $I(4,1) = 0 + 0 + 0 + 5 + 5 + 0 + 0 + 5 + 5 + 5 = 25$

Efficiently Computing Integral Image

- Suppose computed integral image up to location (x,y)

$$I(x,y) = \mathbf{f}(x,y)$$

0	0	0	5	5
0	0	5	5	5
0	5	5	5	10
5	5	5	10	0
5	5	10	0	0

$$\mathbf{f}(x,y)$$

$$I(x,y)$$

+

Efficiently Computing Integral Image

- Suppose computed integral image up to location (x,y)

$$I(x,y) = \mathbf{f}(x,y) + I(x-1,y)$$

0	0	0	5	5
0	0	5	5	5
0	5	5	5	10
5	5	5	10	0
5	5	10	0	0

$$\mathbf{f}(x,y)$$

+	+			
+	+			
+	+			
+	+	+		

$$I(x,y)$$

Efficiently Computing Integral Image

- Suppose computed integral image up to location (x,y)

$$I(x,y) = \mathbf{f}(x,y) + I(x-1,y) + I(x,y-1)$$

0	0	0	5	5
0	0	5	5	5
0	5	5	5	10
5	5	5	10	0
5	5	10	0	0

$f(x,y)$

+	+	+			
+	+	+			
+	+	+			
+	+	+			

$I(x,y)$

Efficiently Computing Integral Image

- Suppose computed integral image up to location (x,y)

$$I(x,y) = \mathbf{f}(x,y) + I(x-1,y) + I(x,y-1) - I(x-1,y-1)$$

0	0	0	5	5
0	0	5	5	5
0	5	5	5	10
5	5	5	10	0
5	5	10	0	0

$f(x,y)$

+	+	+			
+	+	+	+		
+	+	+	+		
+	+	+			

$I(x,y)$

Integral Image: Order of Computation

- Convenient order of computation
 1. first row
 2. first column
 3. the rest in row-wise fashion

1	2	3	4	5
6	10	11	12	13
7	14	15	16	17
8	18	19	20	21
9	22	23	24	25

$$I(x,y)$$

Using Integral Image

- After computed integral image, sum over any rectangular window is computed with four operations
- Top left corner (x_1, y_1) and bottom right corner (x_2, y_2)

$I(x_2, y_2)$

0	0	0	5	5
0	0	5	5	5
0	5	5	5	10
5	5	5	10	0
5	5	10	0	0

$f(x, y)$

+	+	+	+	+
+	+	+	+	+
+	+	+	+	+
+	+	+	+	+

$I(x, y)$

Using Integral Image

- After computed integral image, sum over any rectangular window is computed with four operations
- Top left corner (x_1, y_1) and bottom right corner (x_2, y_2)

$$I(x_2, y_2) - I(x_1-1, y_2)$$

0	0	0	5	5
0	0	5	5	5
0	5	5	5	10
5	5	5	10	0
5	5	10	0	0

$$f(x, y)$$

-	+	-	+	+
-	+	-	+	+
-	+	-	+	+
-	+	-	+	+

$$I(x, y)$$

Using Integral Image

- After computed integral image, sum over any rectangular window is computed with four operations
- Top left corner (x_1, y_1) and bottom right corner (x_2, y_2)

$$I(x_2, y_2) - I(x_1-1, y_2) - I(x_2, y_1-1)$$

0	0	0	5	5
0	0	5	5	5
0	5	5	5	10
5	5	5	10	0
5	5	10	0	0

$$f(x, y)$$

-	-	-	-	-
-	-	+	+	+
-	-	+	+	-
-	-	+	+	+
-	-	+	+	+

$$I(x, y)$$

Using Integral Image

- After computed integral image, sum over any rectangular window is computed with four operations
- Top left corner (x_1, y_1) and bottom right corner (x_2, y_2)

$$I(x_2, y_2) - I(x_1-1, y_2) - I(x_2, y_1-1) + \textcolor{red}{I(x_1-1, y_1-1)}$$

0	0	0	5	5
0	0	5	5	5
0	5	5	5	10
5	5	5	10	0
5	5	10	0	0

$f(x, y)$

$+$	$-$	$+$	$-$	$-$
$-$	$+$	$-$	$+$	$+$
$+$	$-$	$-$	$+$	$-$
$-$	$+$	$-$	$+$	$+$
$-$	$+$	$+$	$+$	$+$

$I(x, y)$

Using Integral Image

- After computed integral image, sum over any rectangular window is computed with four operations
- Top left corner (x_1, y_1) and bottom right corner (x_2, y_2)

$$I(x_2, y_2) - I(x_1-1, y_2) - I(x_2, y_1-1) + I(x_1-1, y_1-1)$$

0	0	0	5	5
0	0	5	5	5
0	5	5	5	10
5	5	5	10	0
5	5	10	0	0

$f(x, y)$

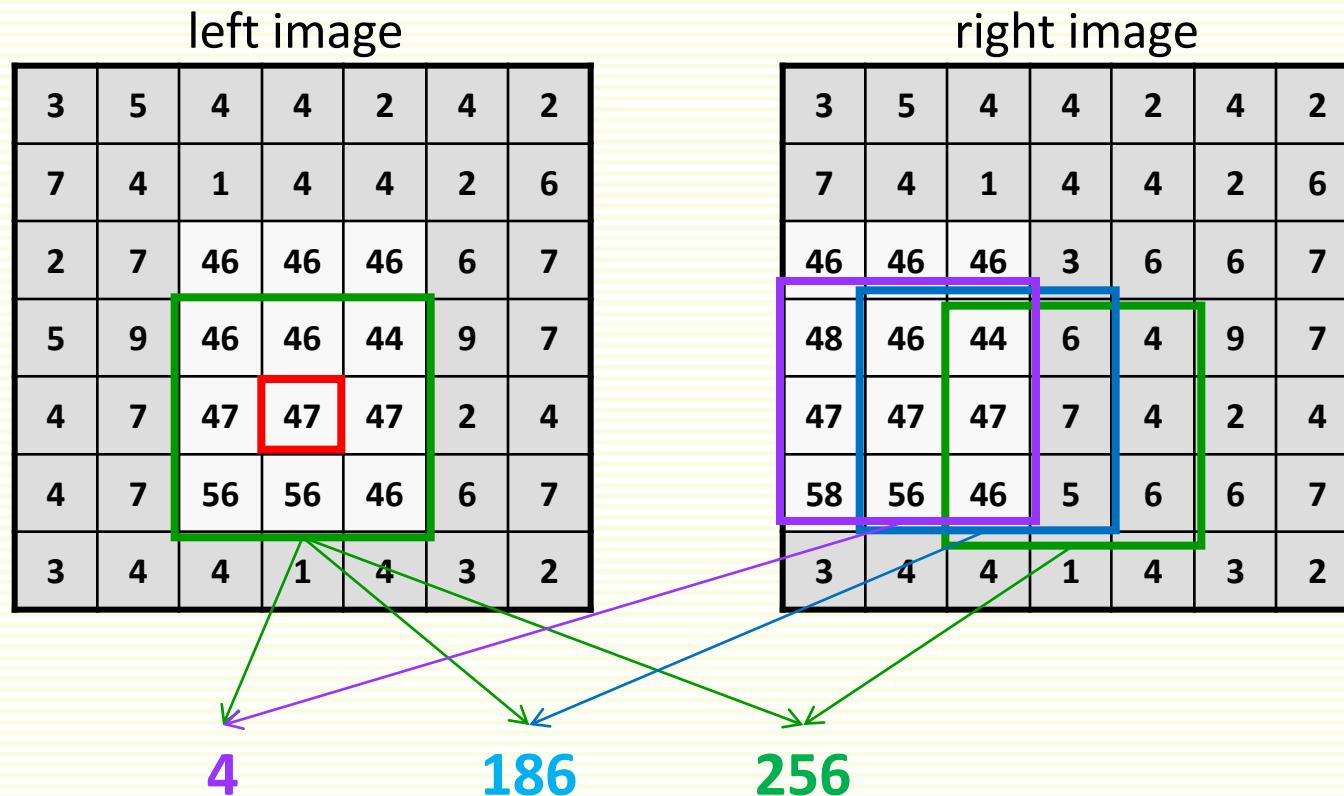
0	0	0	5	10
0	0	5	15	25
0	5	15	30	50
5	15	30	55	75
10	25	50	75	95

$I(x, y)$

- Example: $5 + 5 + 10 + 5 + 10 + 0 = 75 - 15 - 25 + 0 = 35$

Integral Image for Window Matching

- Assume SAD (sum of absolute differences) cost
- Need to find SAD for every pixel and every disparity in a window



Integral Image for Window Matching

- for each pixel p
 - for every disparity d
 - compute cost between window around p in the left image and the same window shifted by d in the right image
 - pick d corresponding to the best matching window

left image

3	5	4	4	2	4	2
7	4	1	4	4	2	6
2	7	46	46	46	6	7
5	9	46	46	44	9	7
4	7	47	47	47	2	4
4	7	56	56	46	6	7
3	4	4	1	4	3	2

right image

3	5	4	4	2	4	2
7	4	1	4	4	2	6
46	46	46	3	6	6	7
48	46	44	6	4	9	7
47	47	47	7	4	2	4
58	56	46	5	6	6	7
3	4	4	1	4	3	2

Integral Image for Window Matching

- For each disparity d need to compute window cost for all pixels, eventually
- For example, pick disparity $d = 1$

left image

3	5	4	4	2	4	2
7	4	1	4	4	2	6
2	7	46	46	46	6	7
5	9	46	46	44	9	7
4	7	47	47	47	2	4
4	7	56	56	46	6	7
3	4	4	1	4	3	2

right image

3	5	4	4	2	4	2
7	4	1	4	4	2	6
46	46	46	3	6	6	7
48	46	44	6	4	9	7
47	47	47	7	4	2	4
58	56	46	5	6	6	7
3	4	4	1	4	3	2

Integral Image for Window Matching

- Old inefficient algorithm:
 - for each pixel p ← **swap**
 - for every disparity d ←
 - compute cost between window around p in the left image and the same window shifted by d in the right image
 - pick d corresponding to the best matching window
- New efficient algorithm:
 - for each disparity d ← **use integral image**
 - for every pixel p ←
 - compute cost between window around p in the left image and the same window shifted by d in the right image
 - pick d corresponding to the best matching window

Integral Image for Window Matching

- Suppose current disparity is $d = 1$

left image								right image							
3	5	4	4	2	4	2		3	5	4	4	2	4	2	
7	4	1	4	4	2	6		7	4	1	4	4	2	6	
2	7	46	46	46	6	7		46	46	46	3	6	6	7	
5	9	46	46	44	9	7		48	46	44	6	4	9	7	
4	7	47	47	47	2	4		47	47	47	7	4	2	4	
4	7	56	56	46	6	7		58	56	46	5	6	6	7	
3	4	4	1	4	3	2		3	4	4	1	4	3	2	

- Overlay left and right image at disparity 1
- Compute AD (absolute difference) between every overlaid pair of pixels
- Compute SAD in a window for every pixel

Integral Image for Window Matching

- current disparity is $d = 1$

left image

3	5	4	4	2	4	2
7	4	1	4	4	2	6
2	7	46	46	46	6	7
5	9	46	46	44	9	7
4	7	47	47	47	2	4
4	7	56	56	46	6	7
3	4	4	1	4	3	2

right image

3	5	4	4	2	4	2
7	4	1	4	4	2	6
46	46	46	3	6	6	7
48	46	44	6	4	9	7
47	47	47	7	4	2	4
58	56	46	5	6	6	7
3	4	4	1	4	3	2

left image

3	3	5	4	4	2	4	2
7	7	4	1	4	4	2	6
2	46	46	46	3	6	6	7
5	48	46	44	6	4	9	7
4	47	47	47	7	4	2	4
4	58	56	46	5	6	6	7
3	3	4	4	1	4	3	2

AD image for disparity 1

2	1	0	2	2	2
3	3	3	0	2	0
39	0	0	43	0	0
39	0	2	38	5	0
40	0	0	40	2	0
51	0	10	41	0	0
1	0	3	3	1	0



Integral Image for Window Matching

- current disparity is $d = 1$
- Pad AD image with zeros

left image							
3	5	4	4	2	4	2	
7	4	1	4	4	2	6	
2	7	46	46	46	6	7	
5	9	46	46	44	9	7	
4	7	47	47	47	2	4	
4	7	56	56	46	6	7	
3	4	4	1	4	3	2	

right image							
3	5	4	4	2	4	2	
7	4	1	4	4	2	6	
46	46	46	3	6	6	7	
48	46	44	6	4	9	7	
47	47	47	7	4	2	4	
58	56	46	5	6	6	7	
3	4	4	1	4	3	2	

3	3	5	4	4	2	4	2
7	7	4	1	4	4	2	6
2	46	46	46	3	6	6	7
5	48	46	44	6	4	9	7
4	47	47	47	7	4	2	4
4	58	56	46	5	6	6	7
3	3	4	4	1	4	3	2



AD image for disparity 1

0	2	1	0	2	2	2
0	3	3	3	0	2	0
0	39	0	0	43	0	0
0	39	0	2	38	5	0
0	40	0	0	40	2	0
0	51	0	10	41	0	0
0	1	0	3	3	1	0

Integral Image for Window Matching

- current disparity is $d = 1$

left image							
3	5	4	4	2	4	2	
7	4	1	4	4	2	6	
2	7	46	46	46	6	7	
5	9	46	46	44	9	7	
4	7	47	47	47	2	4	
4	7	56	56	46	6	7	
3	4	4	1	4	3	2	

right image							
3	5	4	4	2	4	2	
7	4	1	4	4	4	2	6
46	46	46	3	6	6	7	
48	46	44	6	4	9	7	
47	47	47	7	4	2	4	
58	56	46	5	6	6	7	
3	4	4	1	4	3	2	

AD image for disparity 1

0	2	1	0	2	2	2	
0	3	3	3	0	2	0	
0	39	0	0	43	0	0	
0	39	0	2	38	5	0	
0	40	0	0	40	2	0	
0	51	0	10	41	0	0	
0	1	0	3	3	1	0	

Integral Image for Window Matching

- current disparity is $d = 1$

left image							
3	5	4	4	2	4	2	
7	4	1	4	4	2	6	
2	7	46	46	46	6	7	
5	9	46	46	44	9	7	
4	7	47	47	47	2	4	
4	7	56	56	46	6	7	
3	4	4	1	4	3	2	

right image							
3	5	4	4	2	4	2	
7	4	1	4	4	2	6	
46	46	46	3	6	6	7	
48	46	44	6	4	9	7	
47	47	47	7	4	2	4	
58	56	46	5	6	6	7	
3	4	4	1	4	3	2	

AD image for disparity 1

0	2	1	0	2	2	2	
0	3	3	3	0	2	0	
0	39	0	0	43	0	0	
0	39	0	2	38	5	0	
0	40	0	0	40	2	0	
0	51	0	10	41	0	0	
0	1	0	3	3	1	0	

Integral Image for Window Matching

- current disparity is $d = 1$

left image							
3	5	4	4	2	4	2	
7	4	1	4	4	2	6	
2	7	46	46	46	6	7	
5	9	46	46	44	9	7	
4	7	47	47	47	2	4	
4	7	56	56	46	6	7	
3	4	4	1	4	3	2	

right image							
3	5	4	4	2	4	2	
7	4	1	4	4	2	6	
46	46	46	3	6	6	7	
48	46	44	6	4	9	7	
47	47	47	7	4	2	4	
58	56	46	5	6	6	7	
3	4	4	1	4	3	2	

AD image for disparity 1

0	2	1	0	2	2	2	
0	3	3	3	0	2	0	
0	39	0	0	43	0	0	
0	39	0	2	38	5	0	
0	40	0	0	40	2	0	
0	51	0	10	41	0	0	
0	1	0	3	3	1	0	

Integral Image for Window Matching

- Current disparity is 1
- For each window pixel, have to compute window sums in AD image
- Apply integral image to AD image

AD image for disparity 1

0	2	1	0	2	2	2
0	3	3	3	0	2	0
0	39	0	0	43	0	0
0	39	0	2	38	5	0
0	40	0	0	40	2	0
0	51	0	10	41	0	0
0	1	0	3	3	1	0

Efficient Algorithm for Window Matching

for every pixel p do

 bestDisparity[p] = 0

 bestWindCost[p] = HUGE

for disparity $d = 0, 1, \dots, \text{maxD}$ do

 overlay images at disparity d

 compute AD image for disparity d

 compute Integral image from AD image

for every pixel p do

 currentCost = window cost at pixel p , computed from integral image

 if currentCost < bestWindCost[p]

 bestWindCost[p] = currentCost

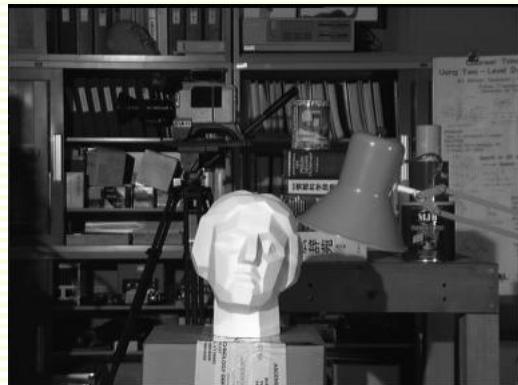
 bestDisparity[p] = d

return bestDisparity

AD image for disparity 1

2	1	0	2	2	2
3	3	3	0	4	0
39	0	0	43	1	0
39	0	2	38	2	0
40	0	0	40	2	0
51	0	10	41	0	0
1	0	3	3	1	0

Effect of Window size



left image



right image



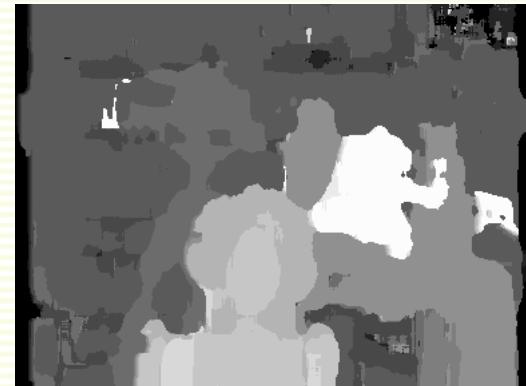
true disparities
bright means larger disparity



3x3 window



7x7 window

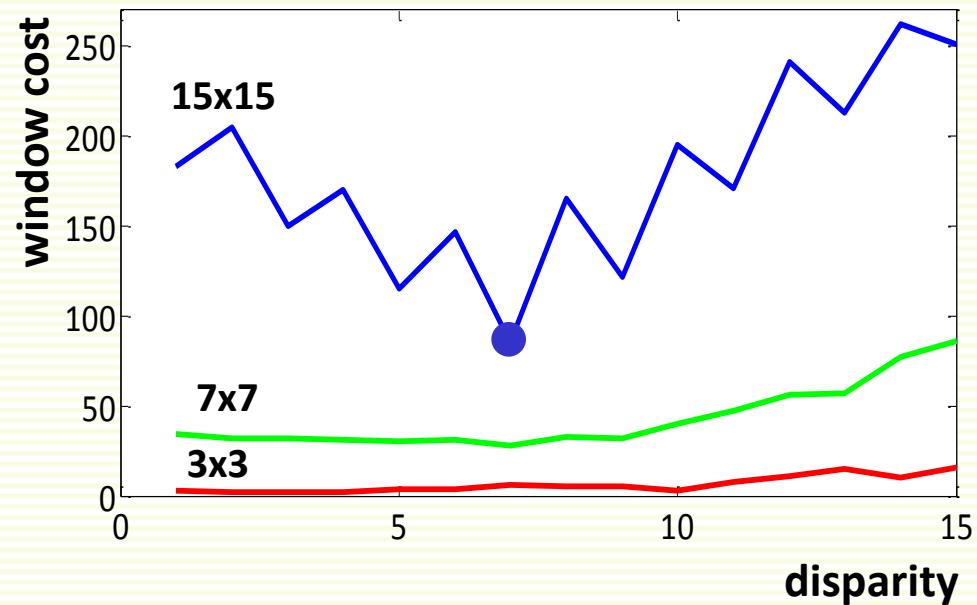


15x15 window

Effect of Window size: Low Texture Area



left image

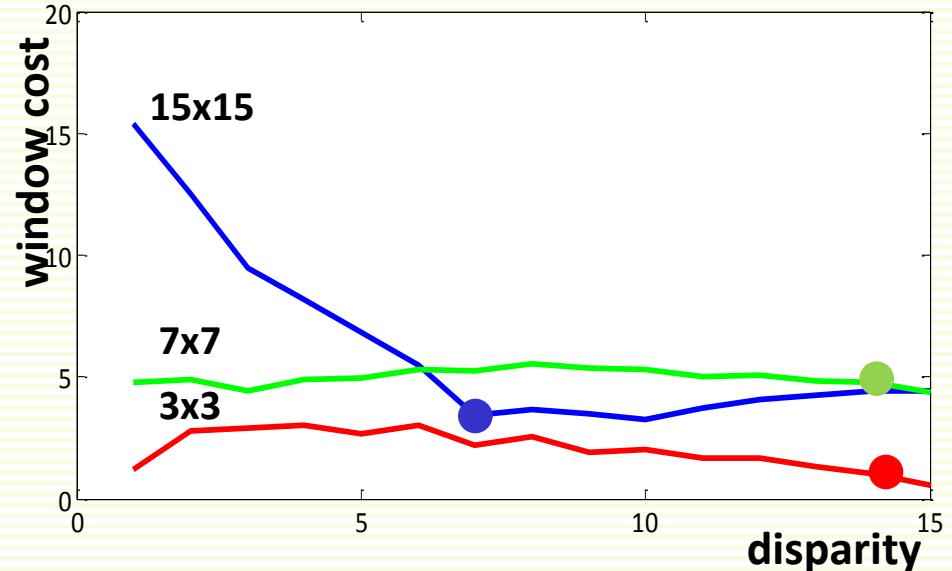


- windows of size 3×3 and 7×7 are too small to have a distinct pattern
 - no clearly best disparity
- window of size 15×15 is large enough to have a distinct pattern
 - 7 is clearly the best disparity
- **window has to be large enough**

Effect of Window size: Near Discontinuities



left image

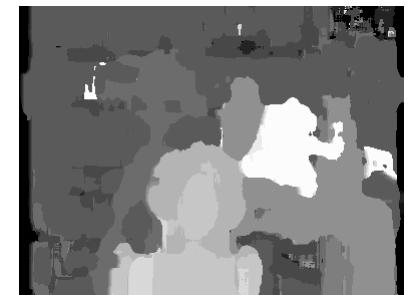


- central pixel (the one we are matching) is the lamp
- windows of size 3x3 and 7x7 contain mostly the lamp
- window of size 15x15 contains mostly the wall
 - we match the wall instead of the lamp!
- window must be **small enough** to contain mostly the same object as the central pixel



Effect of Window size

- No single window size is ‘perfect’ for the image
- Smaller window
 - works better around object boundaries
 - noisy results in low texture areas
- Larger window
 - better results in low texture areas
 - does not preserve object boundaries well
- **Adaptive** window algorithms exist [Veksler’2001]



Better Stereo Algorithms

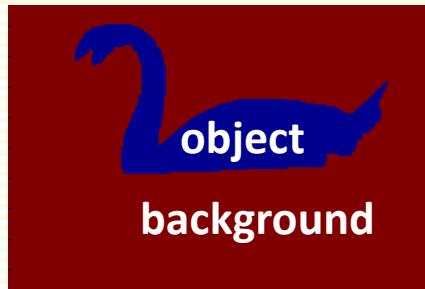


State of the art method
[Boykov, Veksler, Zabih, 2001]

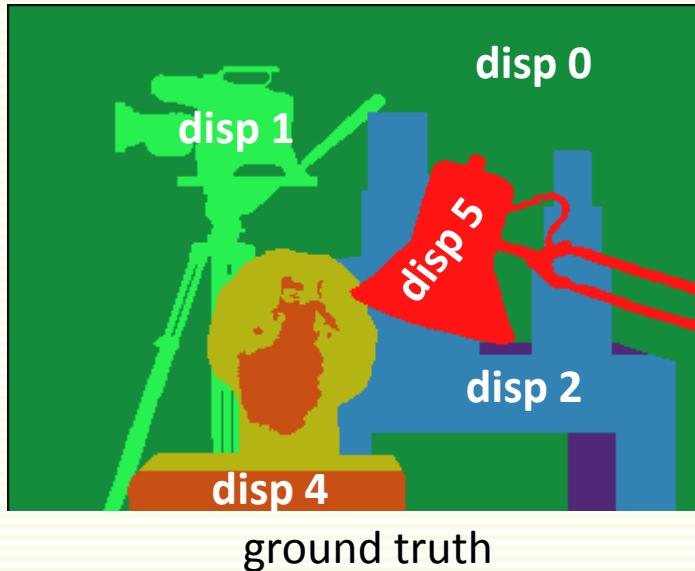


ground truth

- Formulate stereo as energy minimization
- Recall binary object/background segmentation problem

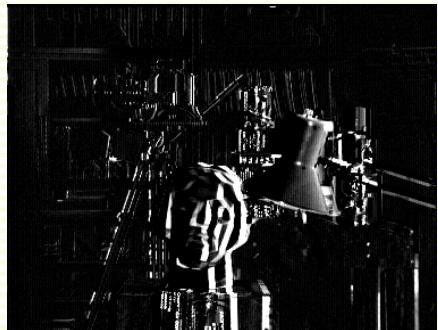


Better Stereo Algorithms



- Stereo is multi-label segmentation problem
 - region 0 = label 0 “likes” disparity 0
 - region 1 = label 1 “likes” disparity 1
 - ...
 - region maxDisp = label maxDisp “likes” disparity maxDisp

Stereo with Graph Cuts



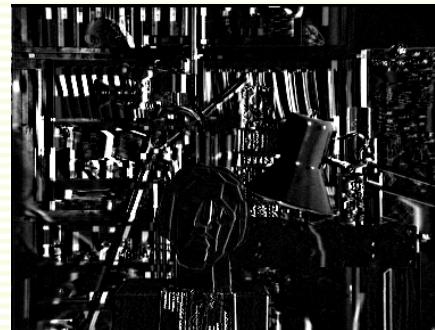
AD 5

data term for label 5



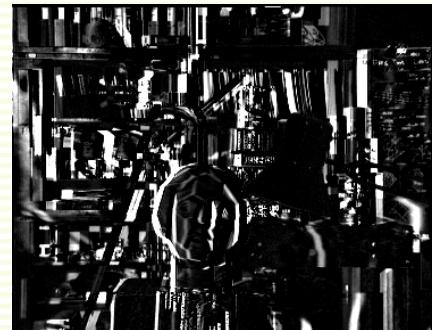
AD 8

data term for label 8



AD 10

data term for label 10



AD 14

data term for label 14

- Energy Function
 - Data Term: assign each pixel disparity label it likes
 - Smoothness Term: count number of label (disparity) discontinuities
- Solved with Graph Cuts: iteratively cuts out regions corresponding to disparities
- NP-hard with more than 2 labels, but computes a good approximation



Stereo with Graph Cuts

- Start with everything as label (disparity) 0



Stereo with Graph Cuts

- “Cut out” label (disparity) 1



Stereo with Graph Cuts

- “Cut out” label (disparity) 2



Stereo with Graph Cuts

- “Cut out” label (disparity) 3



Stereo with Graph Cuts

- “Cut out” label (disparity) 4



Stereo with Graph Cuts

- “Cut out” label (disparity) 5



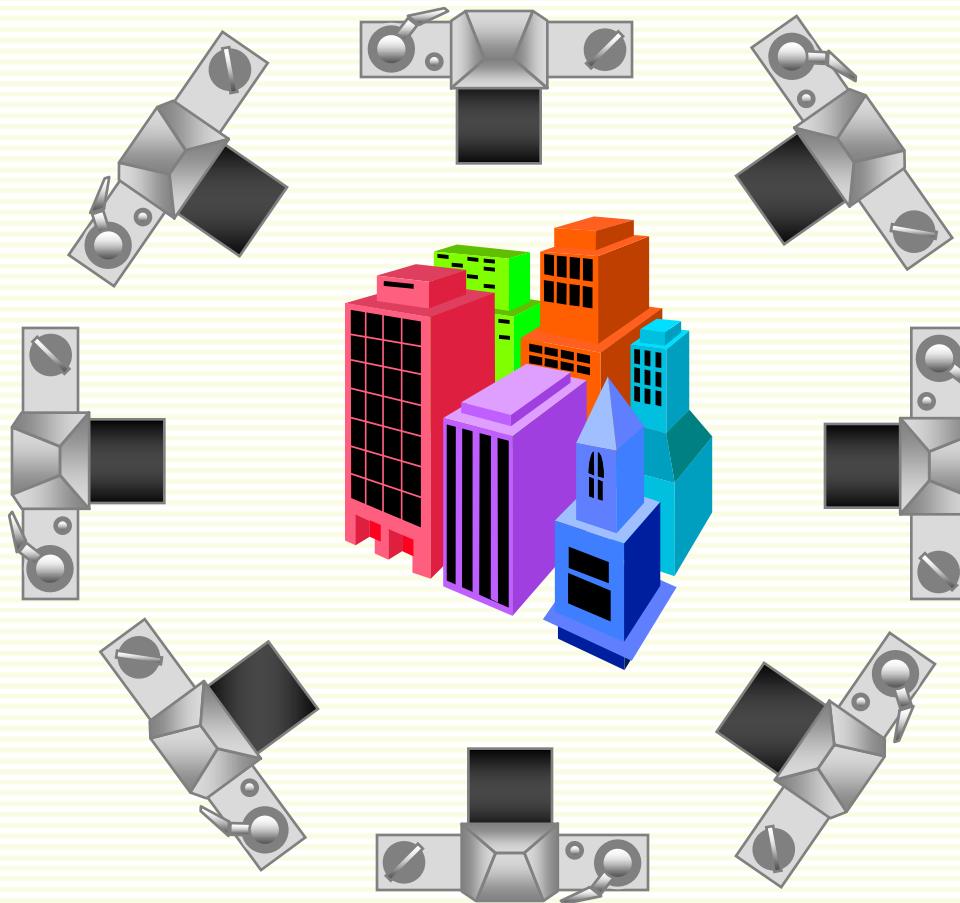
Stereo with Graph Cuts

- “Cut out” label (disparity) 6

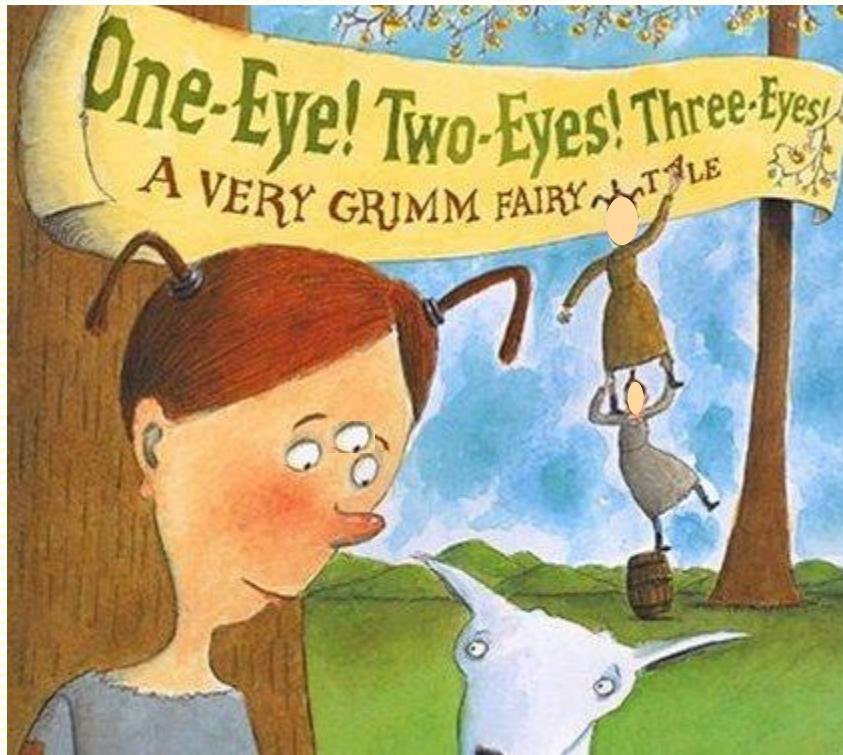


Multiple Artificial Eyes

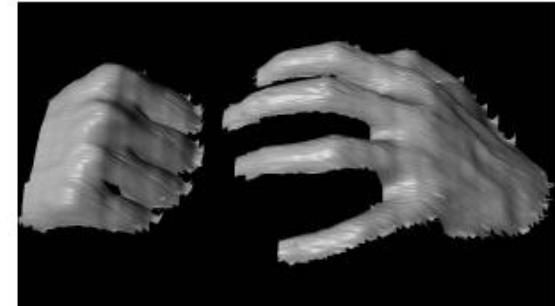
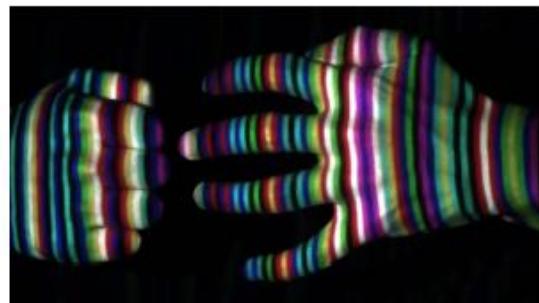
- Two eyes better than one → three eyes better than two → four eyes better than three → ... → the more, the better



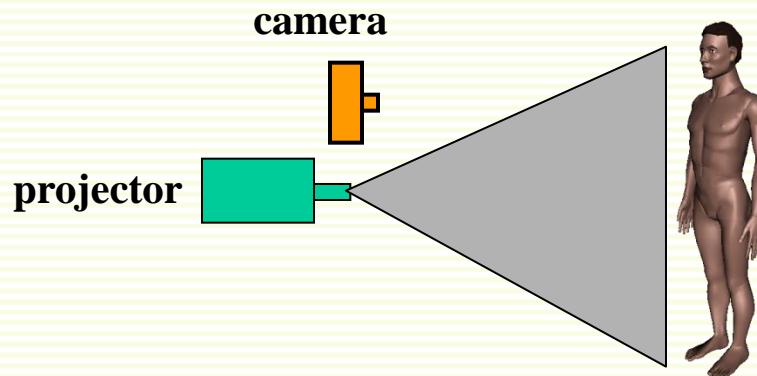
Common Folk New that Already



Stereo with Structured Light

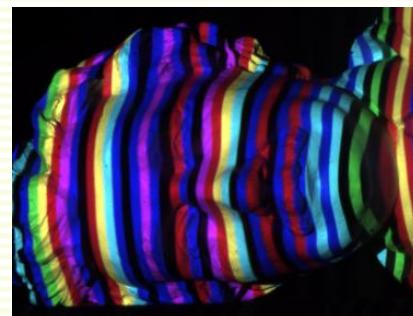
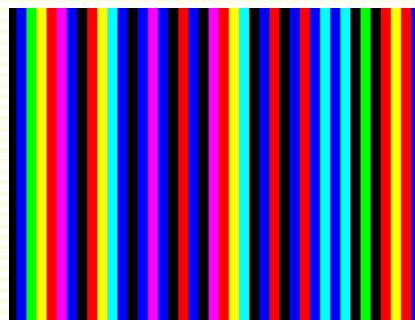
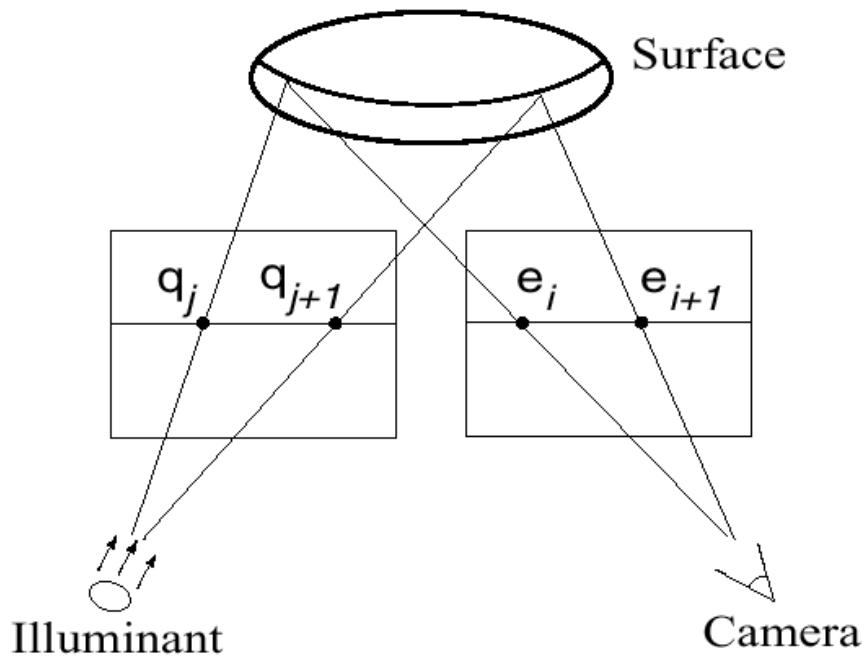


- Project “structured” light patterns onto the object
 - Simplifies correspondence problem
 - Need one camera and one projector



Stereo with Structured Light

- Triangulate between camera and projector

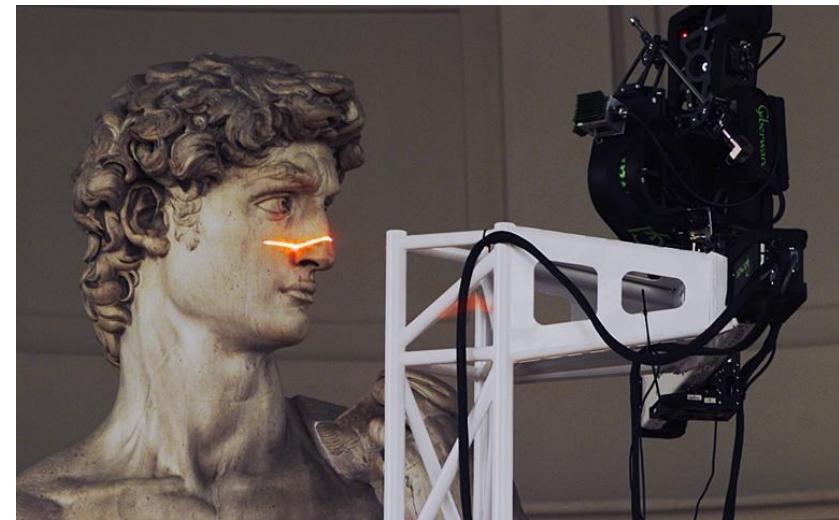
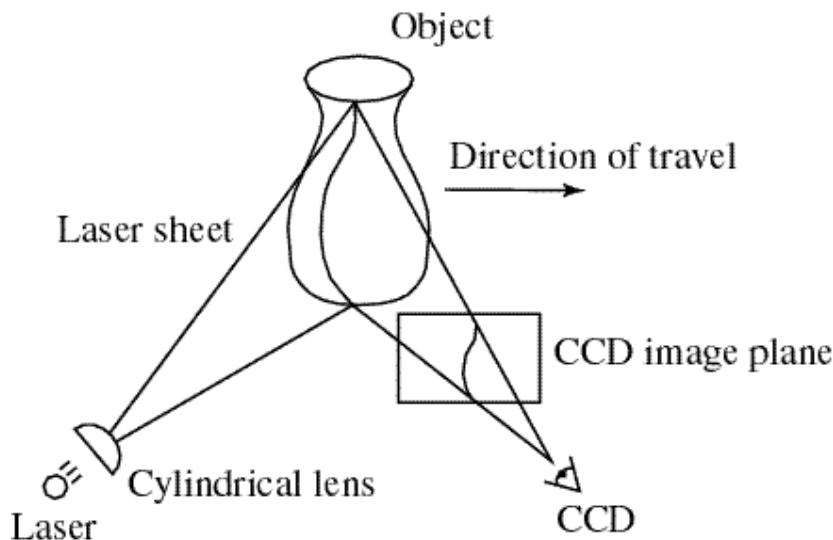


Kinect: Structured Infrared Light



<http://bbzippo.wordpress.com/2010/11/28/kinect-in-infrared/>

Laser Scanning



Digital Michelangelo Project
Levoy et al.

<http://graphics.stanford.edu/projects/mich/>

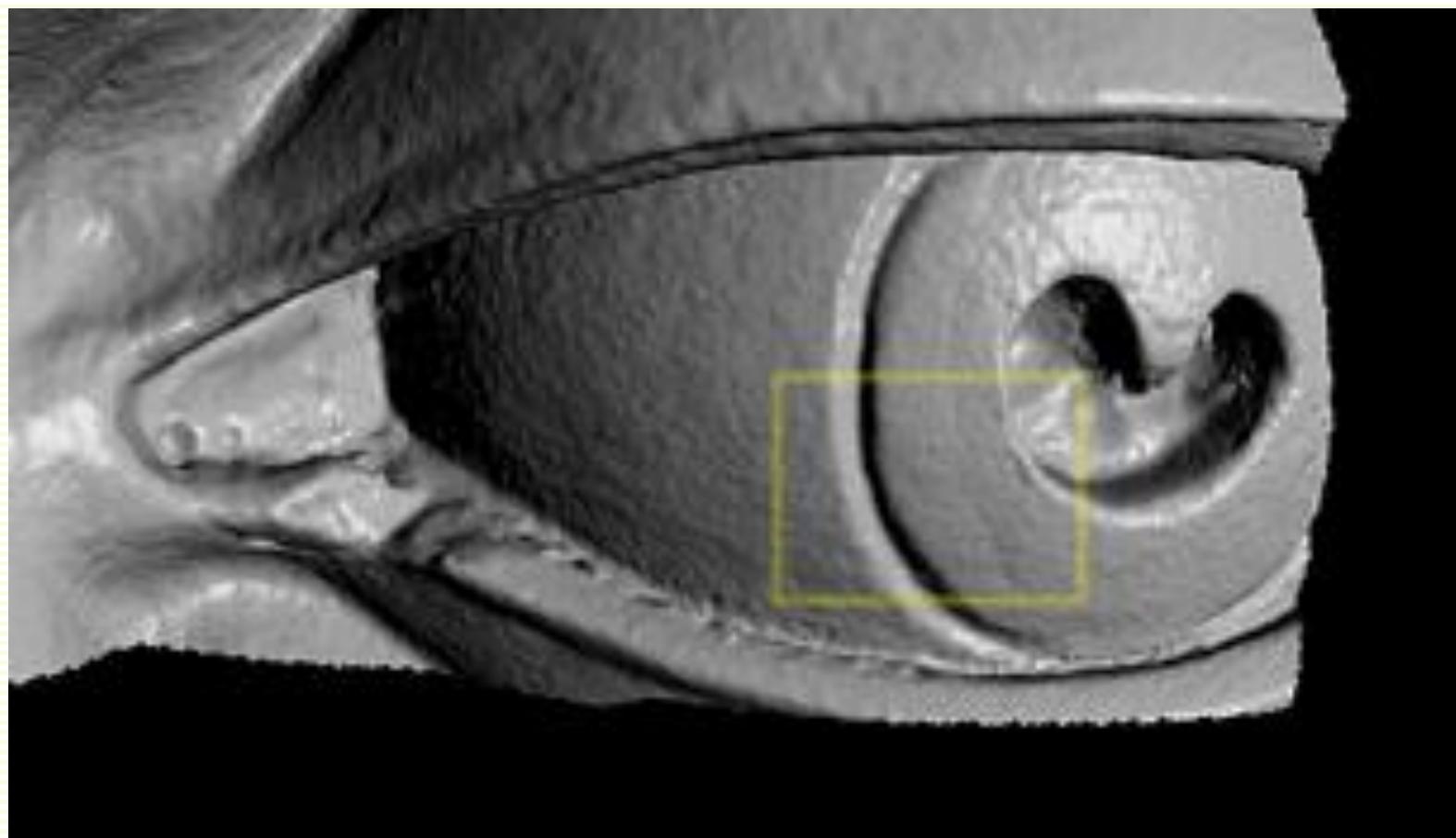
- Optical triangulation
 - Project a single stripe of laser light
 - Scan it across the surface of the object
 - This is a very precise version of structured light scanning

Laser Scanned Models



The Digital Michelangelo Project, Levoy et al.

Laser Scanned Models



The Digital Michelangelo Project, Levoy et al.

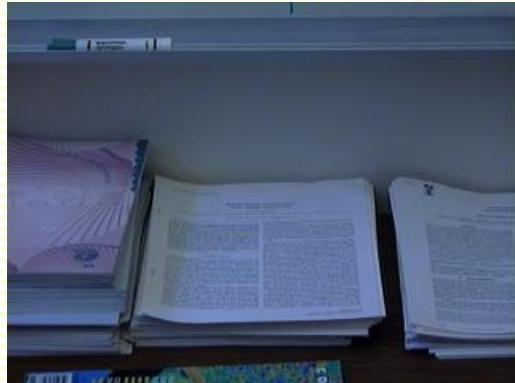
Numerous Applications

- Autonomous navigation



[Nomad robot](#) searches for meteorites in Antarctica
<http://www.frc.ri.cmu.edu/projects/meteorobot/index.html>

Novel View Synthesis



input image (1 of 2)



depth map
[Szeliski & Kang '95]



3D rendering

Applications: Video View Interpolation

<http://research.microsoft.com/users/larryz/videoviewinterpolation.htm>



Stereo Correspondence

- Steps:
 - Calibrate cameras
 - Rectify images
 - Stereo correspondence
 - Apply depth/disparity formula
- Stereo correspondence is still heavily researched
- The simple window matching algorithm we studied is heavily used in practice due to speed and simplicity
- Popular Benchmark
<http://www.middlebury.edu/stereo>