CS4442/9542b: Artificial Intelligence II

Prof. Olga Veksler

Lecture 14: Computer Vision

3D shape from Images

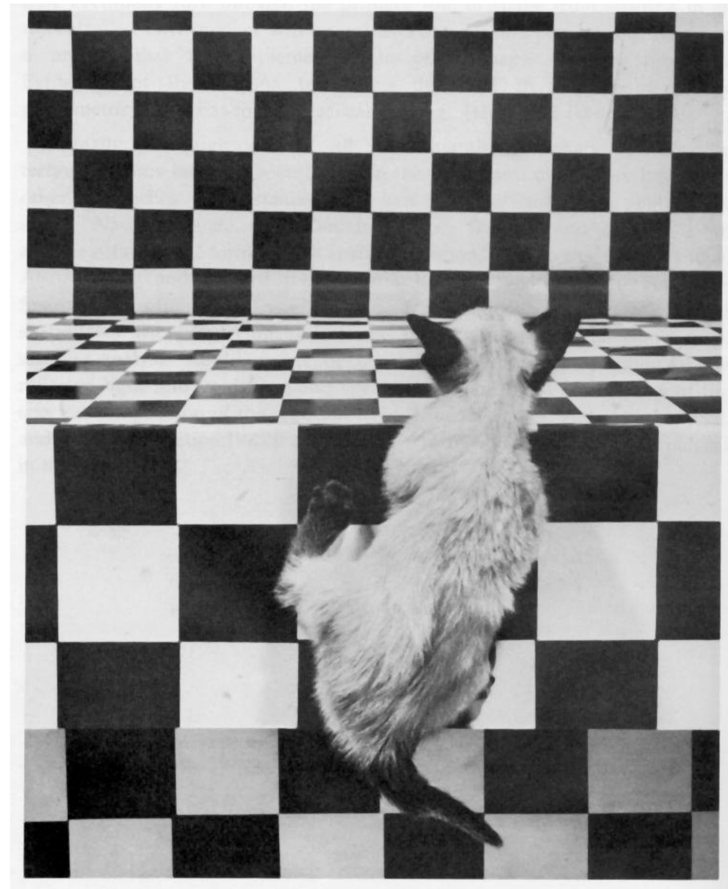
Stereo Reconstruction

Many Slides are from Steve Seitz (UW), S. Narasimhan

Outline

- Cues for 3D shape perception
- Stereo (3D shape from 2 stereo images)
  - Camera calibration and rectification (easier)
  - Stereo Correspondence (harder)

Babies and Animals Perceive Depth



***The Visual Cliff*, by William Vandivert, 1960**

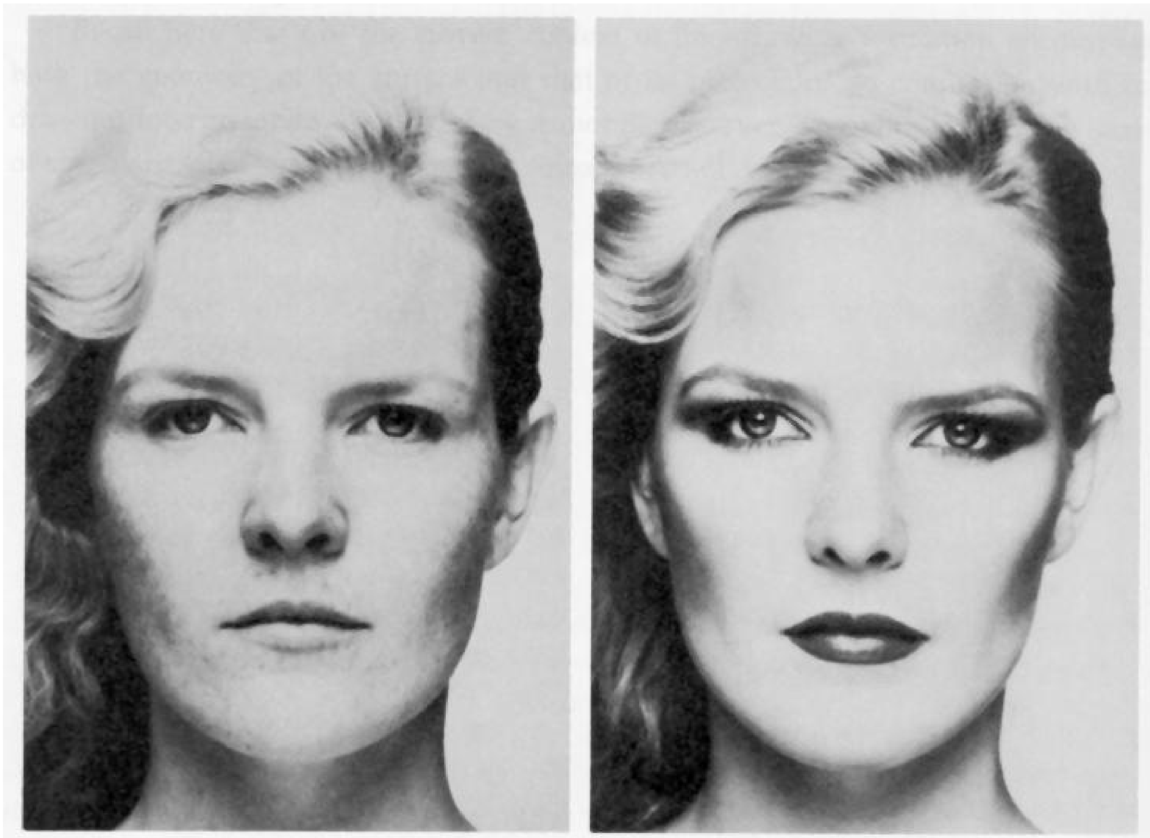
3D shape from images

How might we do this automatically?

- What cues in the image provide 3D information?

Single Image 3D Cues: Shading

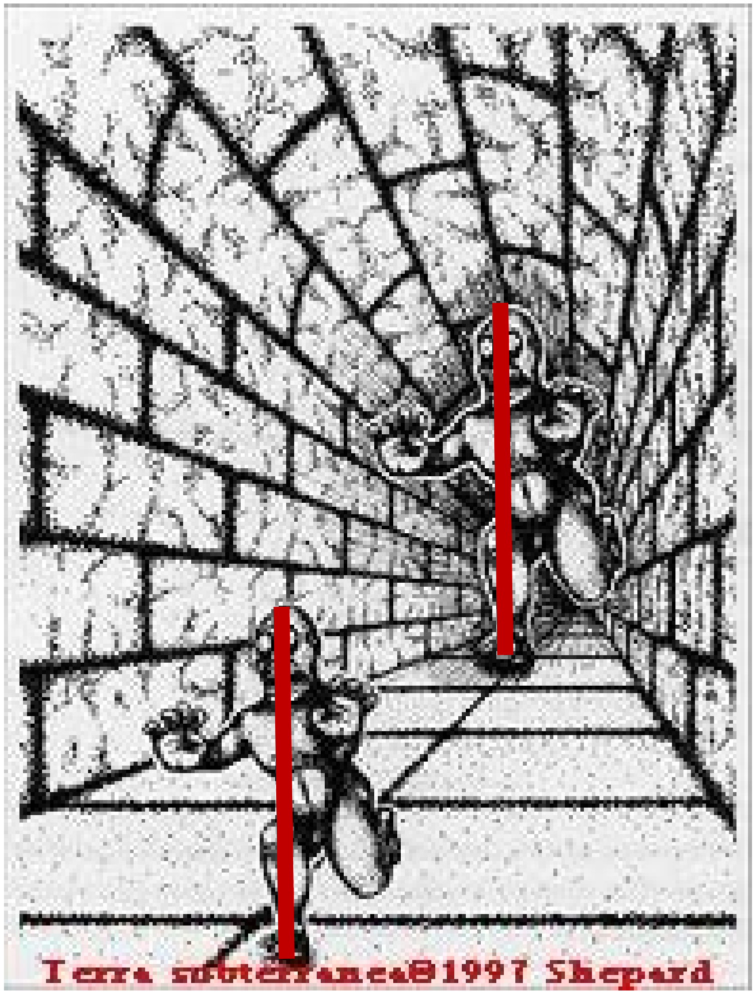
Pixels covered by shadow are perceived to be further away



Merle Norman Cosmetics, Los Angeles

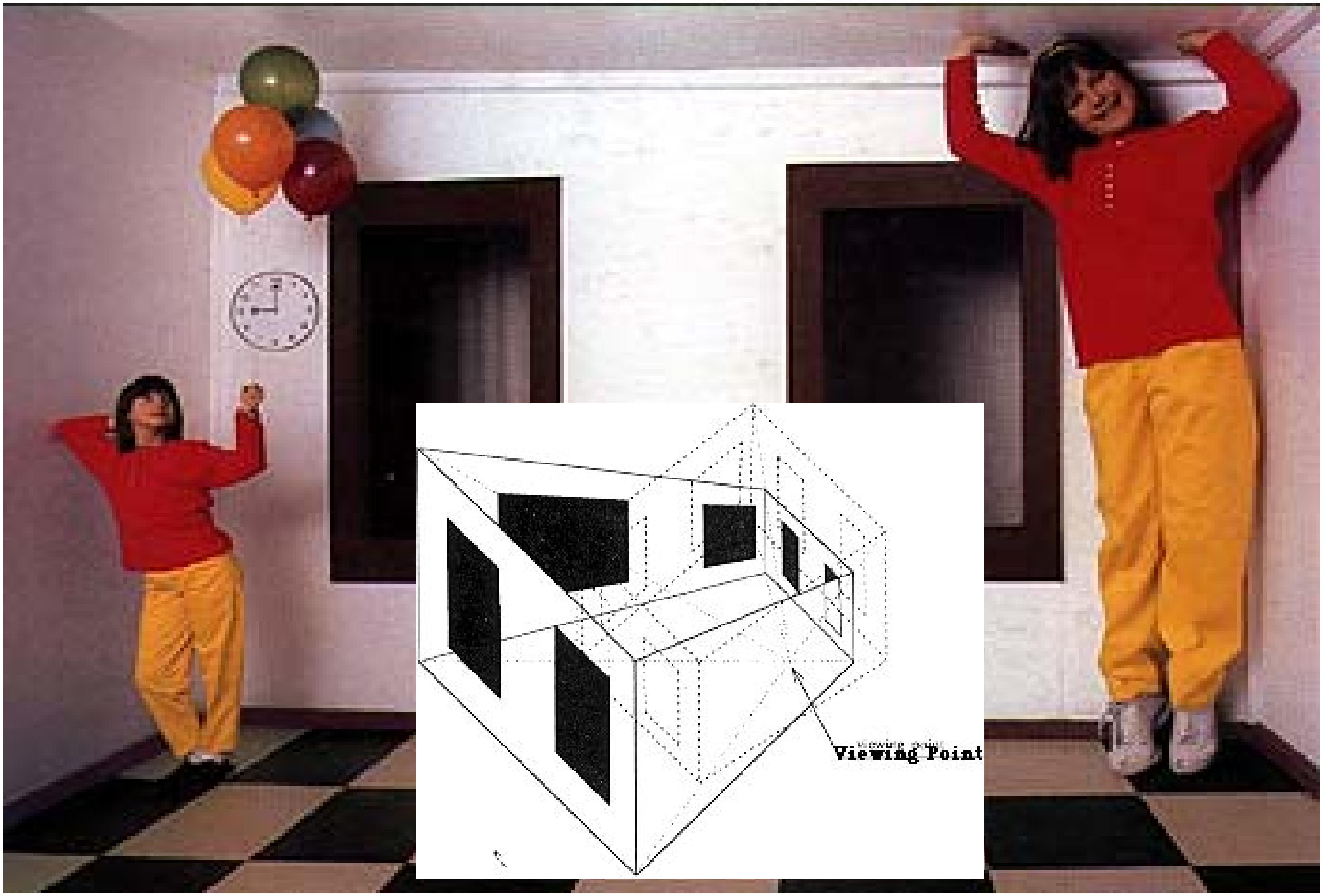
Single Image 3D Cues: Linear Perspective

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



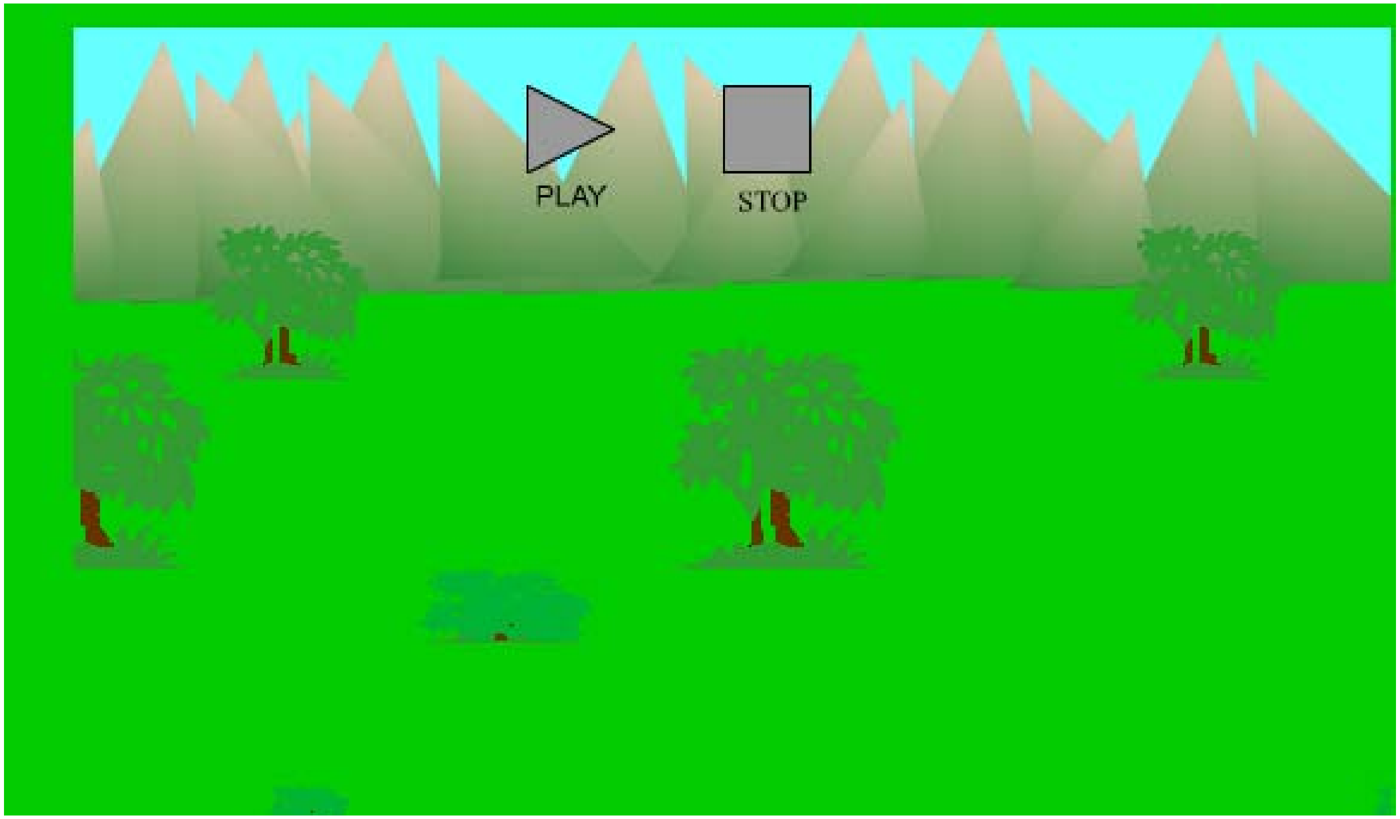
Ames Room: Size-Distance Cues

Ames Room: Size-Distance Cues



Visual cues: Motion Parallax

- Objects that are closer appear to more than the objects that are further away



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

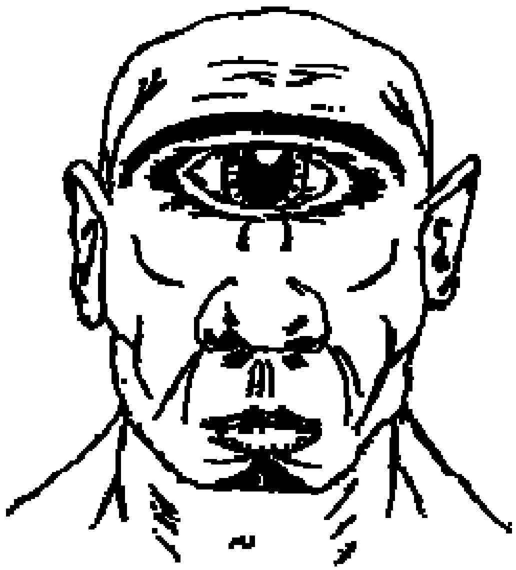
Single Image 3D Cues: Texture

- The further away the texture is, the finer it becomes

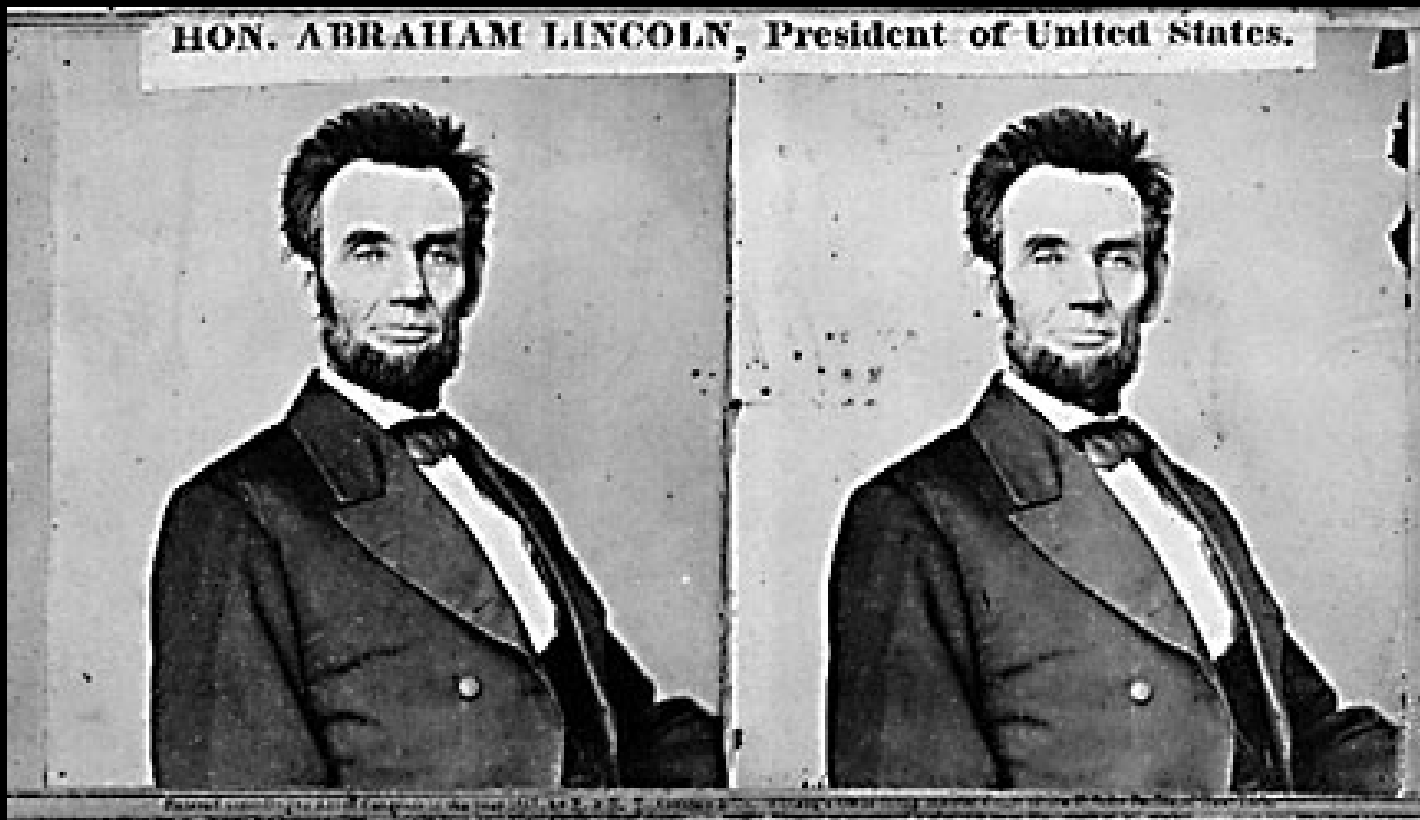
Visual cues

- Shape From X
  - X = shading, texture,motion, ...
  - In this class we’ll focus on stereo
    - Depth perception from two stereo images

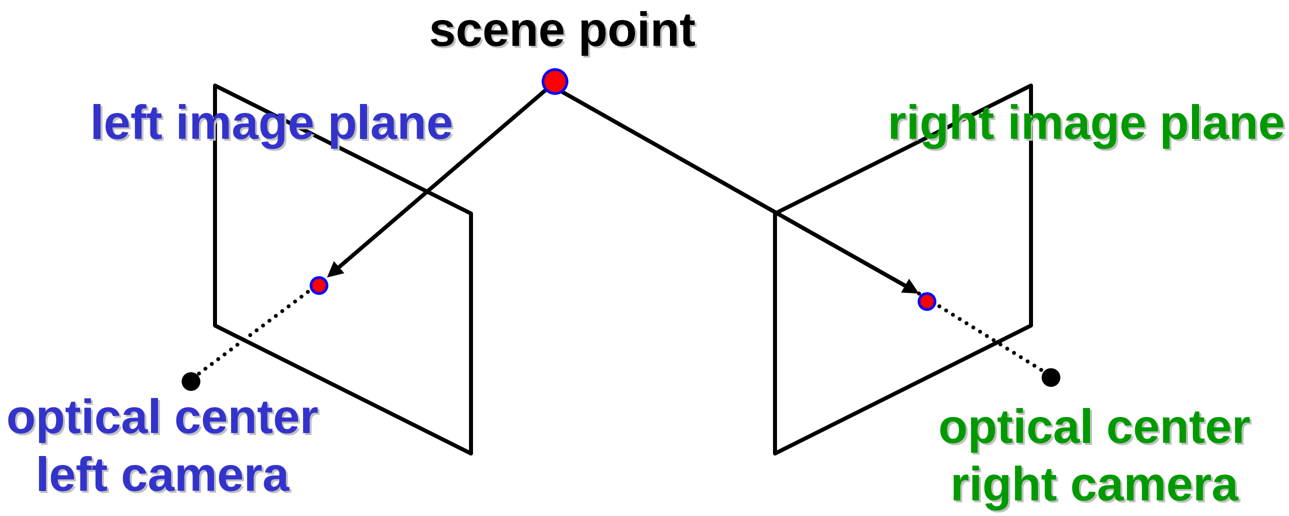
Why do we have two eyes?



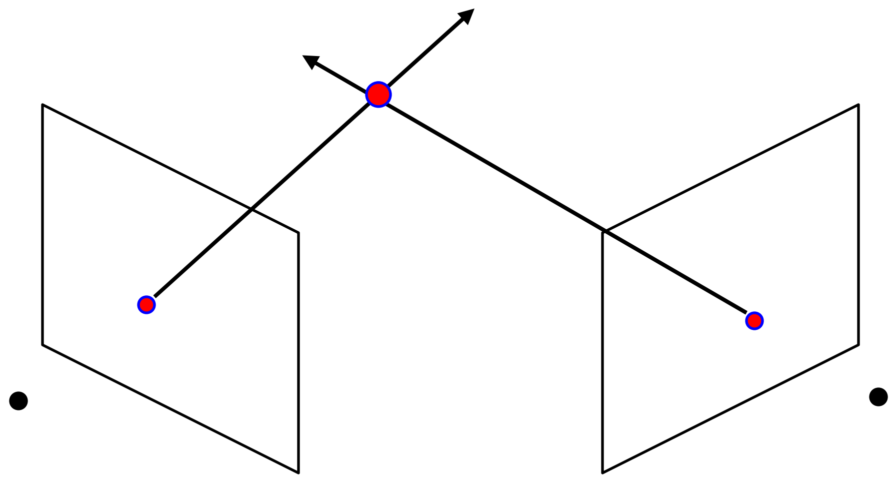
Cyclope **vs.** **Odysseus**


HON. ABRAHAM LINCOLN, President of United States.

Stereo Images



Stereo Images



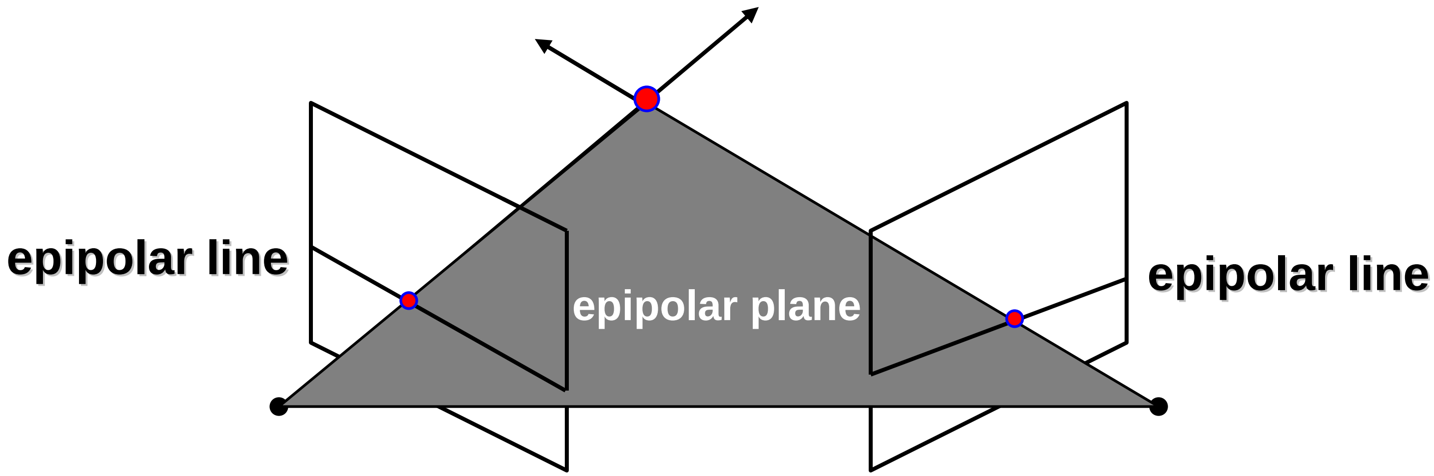
Basic Principle: Triangulation

- Gives reconstruction as intersection of two rays
- Requires
  1. position of cameras with respect to each other
     - performed with ***camera calibration*** relatively easy and well understood
  2. ***point correspondence***
     - hard problem, usually called ***stereo correspondence***

Stereo correspondence

Determine Pixel Correspondence

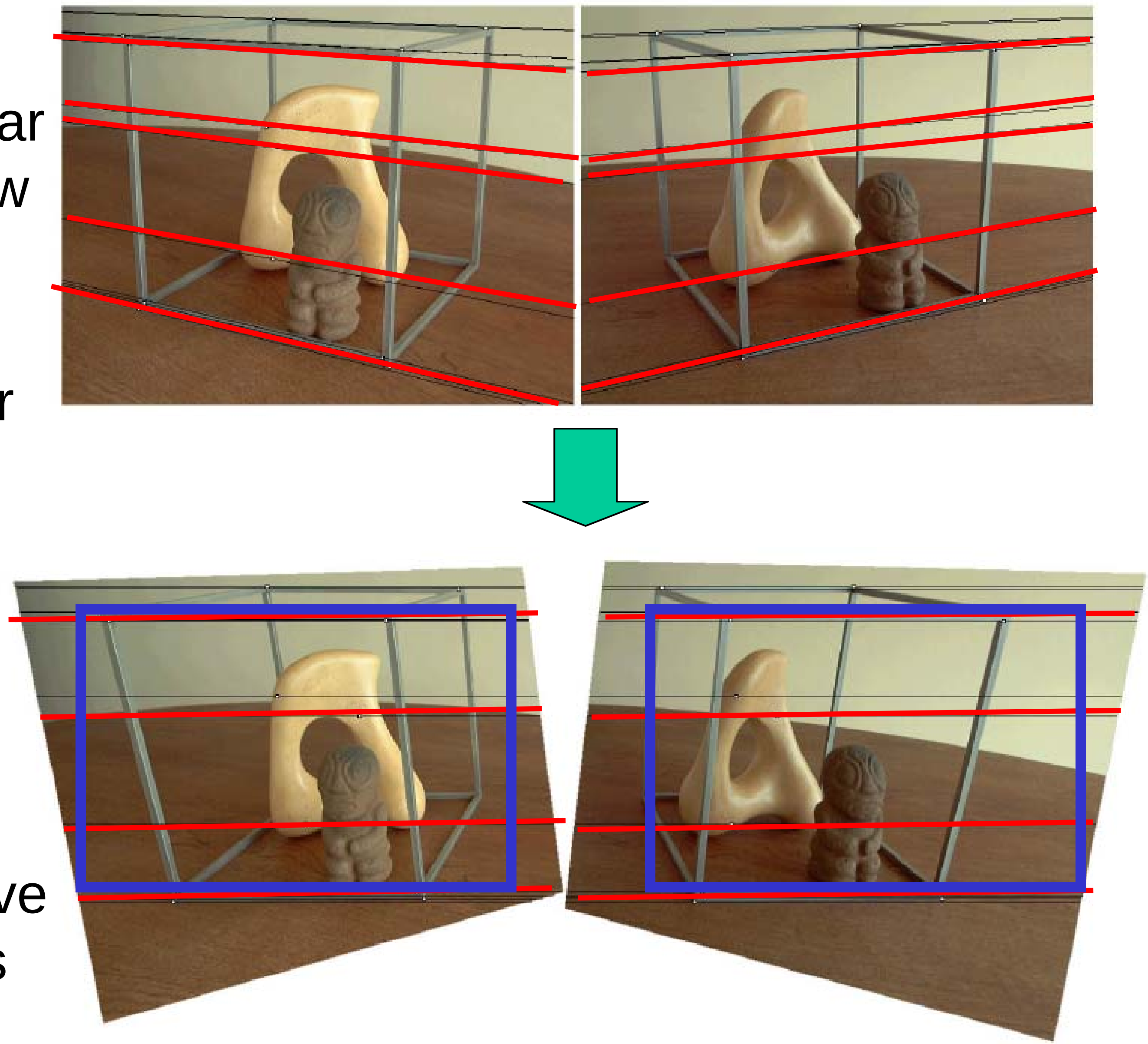
- Pairs of points that correspond to same scene point



- Epipolar Constraint
  - Reduces correspondence problem to 1D search along *conjugate epipolar lines*
  - Java demo: http://www.ai.sri.com/~luong/research/Meta3DViewer/EpipolarGeo.html

Stereo Rectification

- It's easy to compute epipolar lines given a few corresponding points
- Usually epipolar lines are not horizontal
- Can rectify images to have horizontal epipolar lines
- Human eyes give rectified images



Depth from disparity

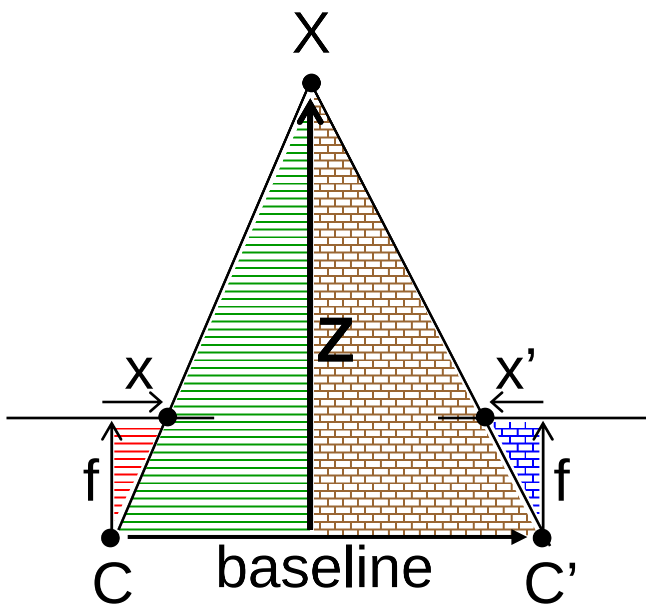
- From similarity of red and green striped triangles:

$$\frac{baseline/2}{Z} = \frac{x}{f}$$

- From similarity of brown and blue brick triangles:

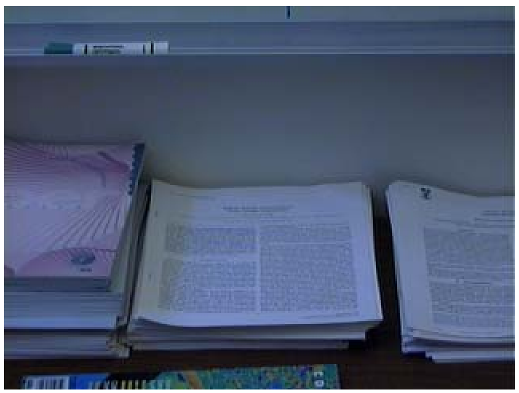
$$\frac{baseline/2}{Z} = \frac{-x'}{f}$$

- Adding two expressions above and simplifying:



$$disparity = x - x' = \frac{baseline \cdot f}{Z}$$

Depth from disparity


input image (1 of 2)


depth map
[Szeliski & Kang ‘95]


3D rendering

Stereo matching algorithms

- Rectifying images and figuring out *baseline* between camera and *f* (depth of focus) is relatively easy and well understood
- Matching pixels on the corresponding epipolar lines lines is a much harder problem
    - Still heavily researched
    - Numerous approaches
        - A good survey and evaluation: http://www.middlebury.edu/stereo/

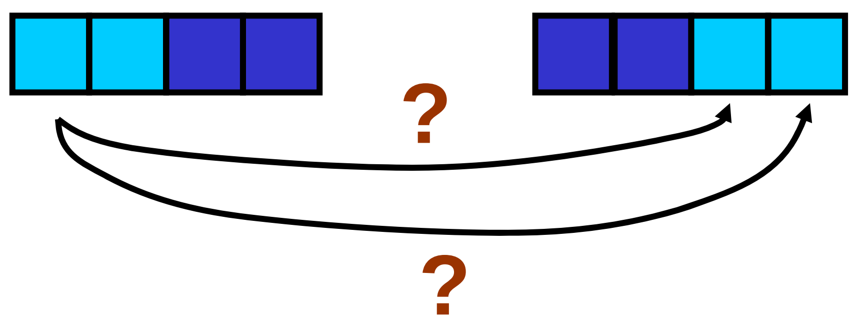
Difficulties in Stereo Correspondence

left image **right image**

Perfect case:
never happens!

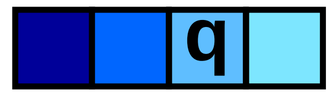
1) Image noise:

2) Low texture:



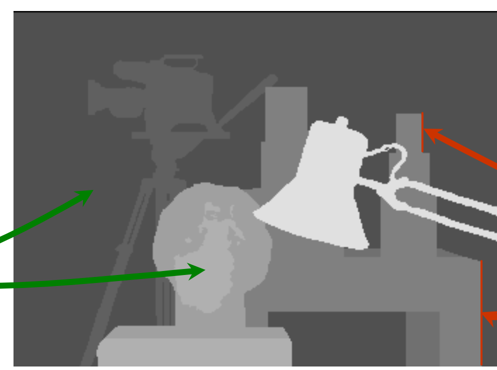
Constraints

1) corresponding pixels should be close in color



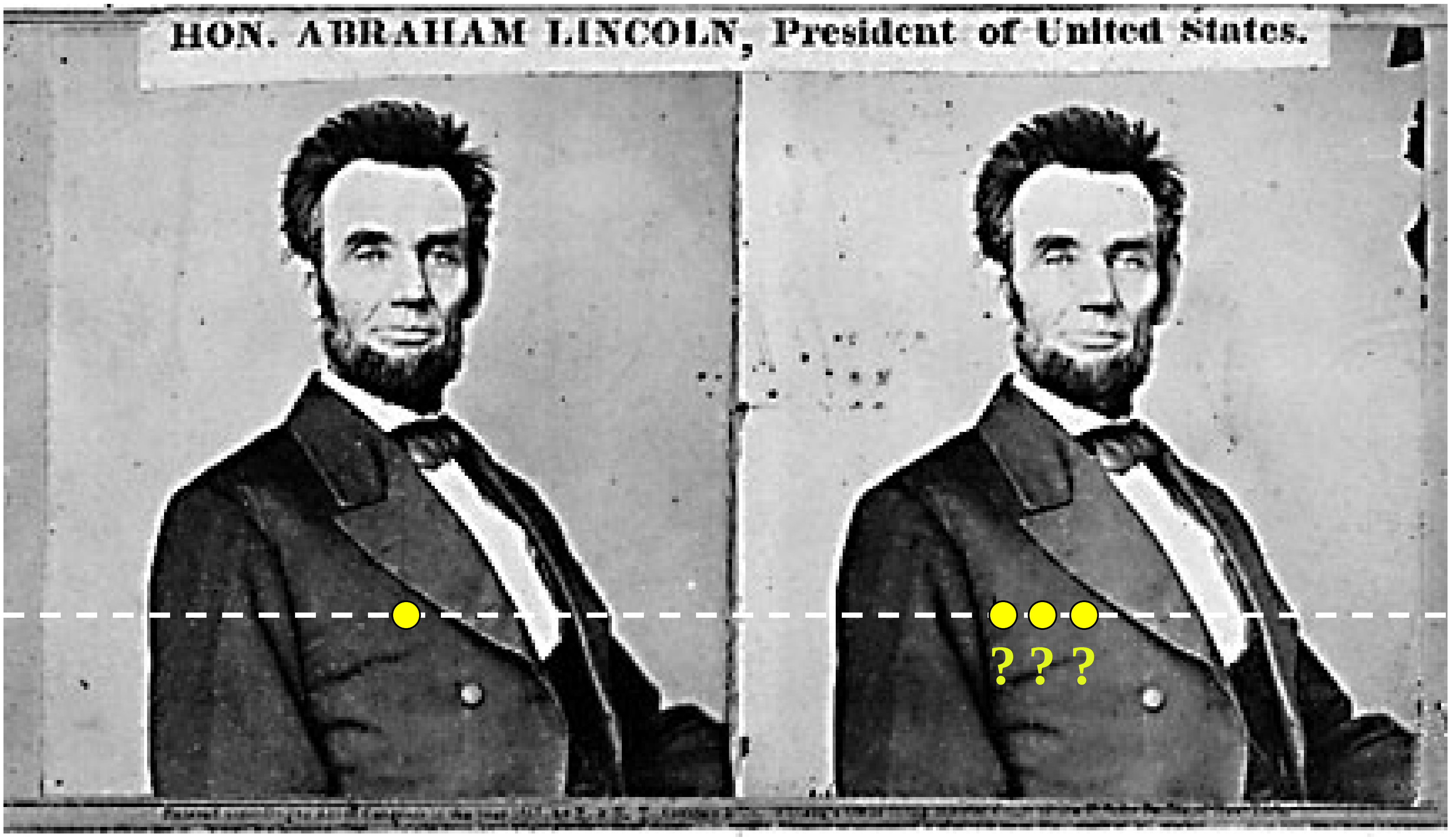
2) most nearby pixels should have close disparity

disparity continuous in most places



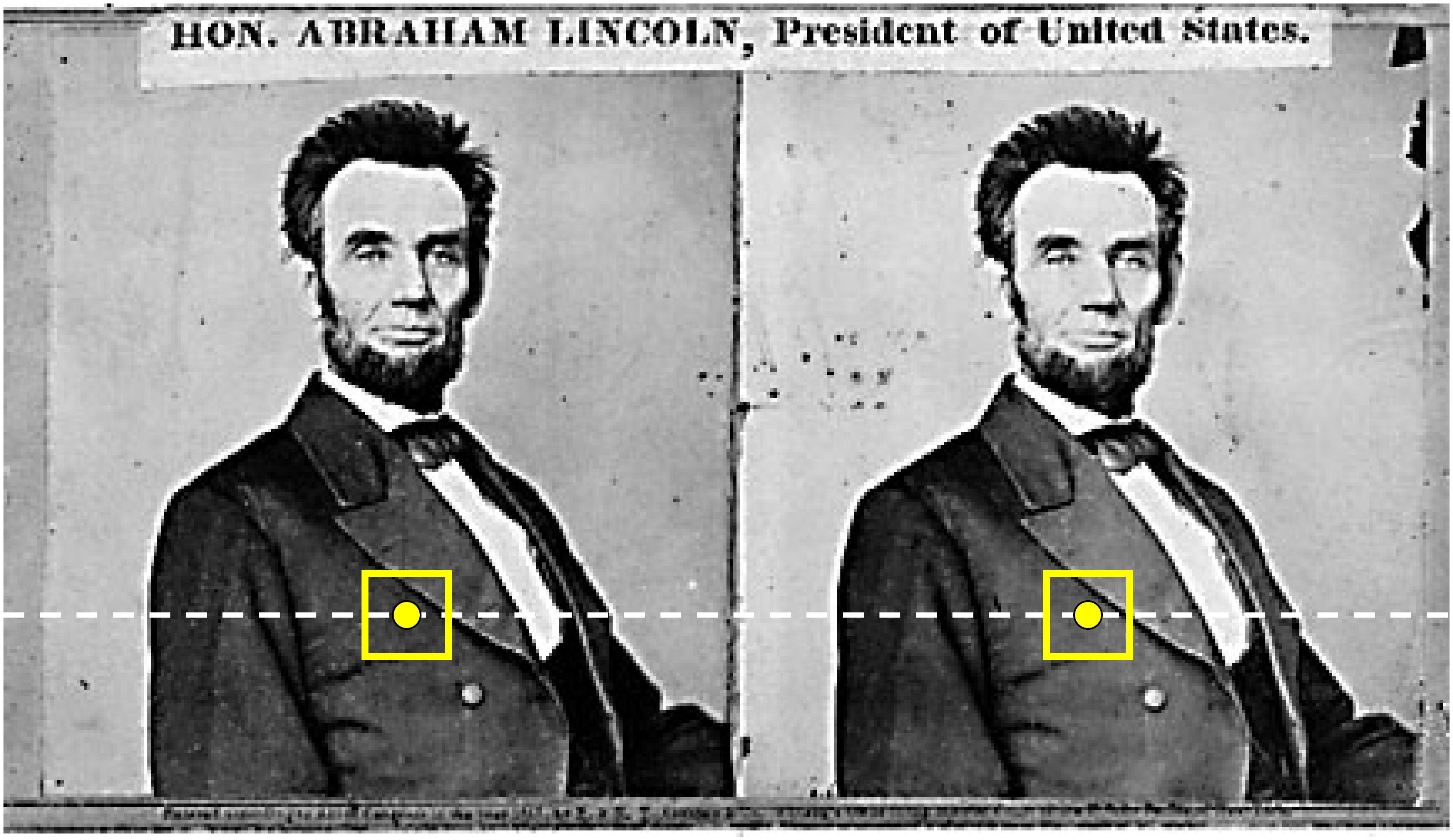
except a few places: disparity discontinuity

Your basic stereo algorithm

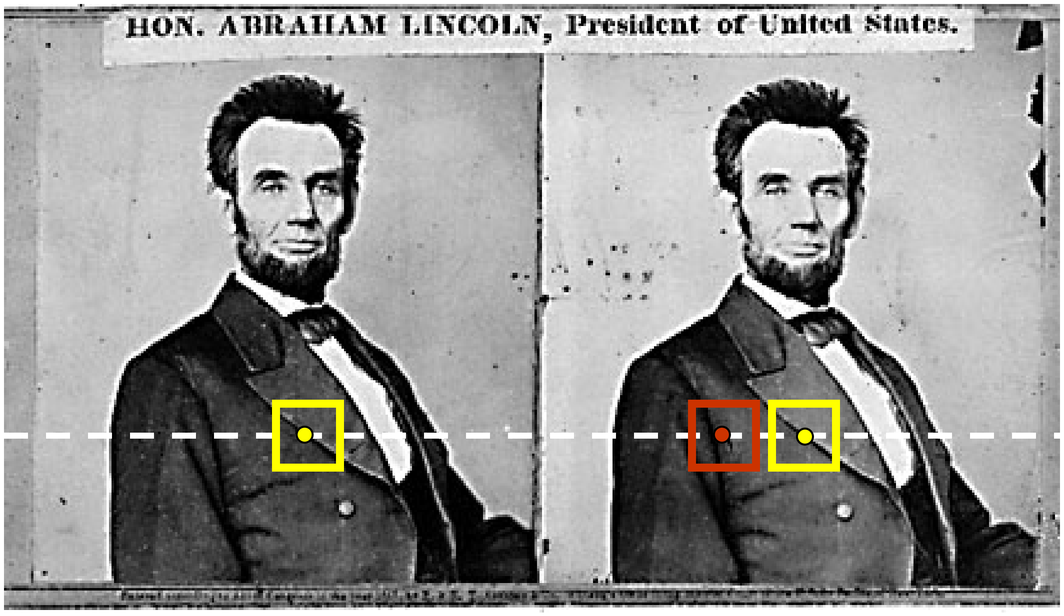


For each epipolar line
- For each pixel in the left image
  - compare with every pixel on same epipolar line in right image
  - pick pixel with minimum match cost
    - doesn't really work due to noise and presence of low texture areas

Your basic stereo algorithm



Improvement: Match Windows



For each epipolar line

For each pixel in the left image

- compare a window with several windows on same epipolar line in right image
- Pick window with minimum match cost
- Common window cost: sum of squared differences (SDD)

Sum of Squared (Pixel) Differences

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

- disparity can be only positive
- can limit disparity to be in a range 0,1,…, maxD
- to compute the disparity for the red pixel, take some window around it and compute SSD between that window and the same window shifted by disparity 0, 1,…, maxD in the right image
- Choose disparity corresponding to the smallest SSD

Sum of Squared (Pixel) Differences

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

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

- This shift corresponds to disparity 0
  - All pixels in blue window have the same x coordinate as the corresponding pixels in the green window

Sum of Squared (Pixel) Differences

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

$$(46-46)^2+(46-44)^2+(44-6)^2+$$
$$(47-47)^2+(47-7)^2+(47-7)^2+$$
$$(56-56)^2+(56-46)^2+(46-5)^2=6425$$

- This shift corresponds to disparity 1
  - All pixels in blue window have x coordinate 1 less than corresponding pixels in the green window

Sum of Squared (Pixel) Differences

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

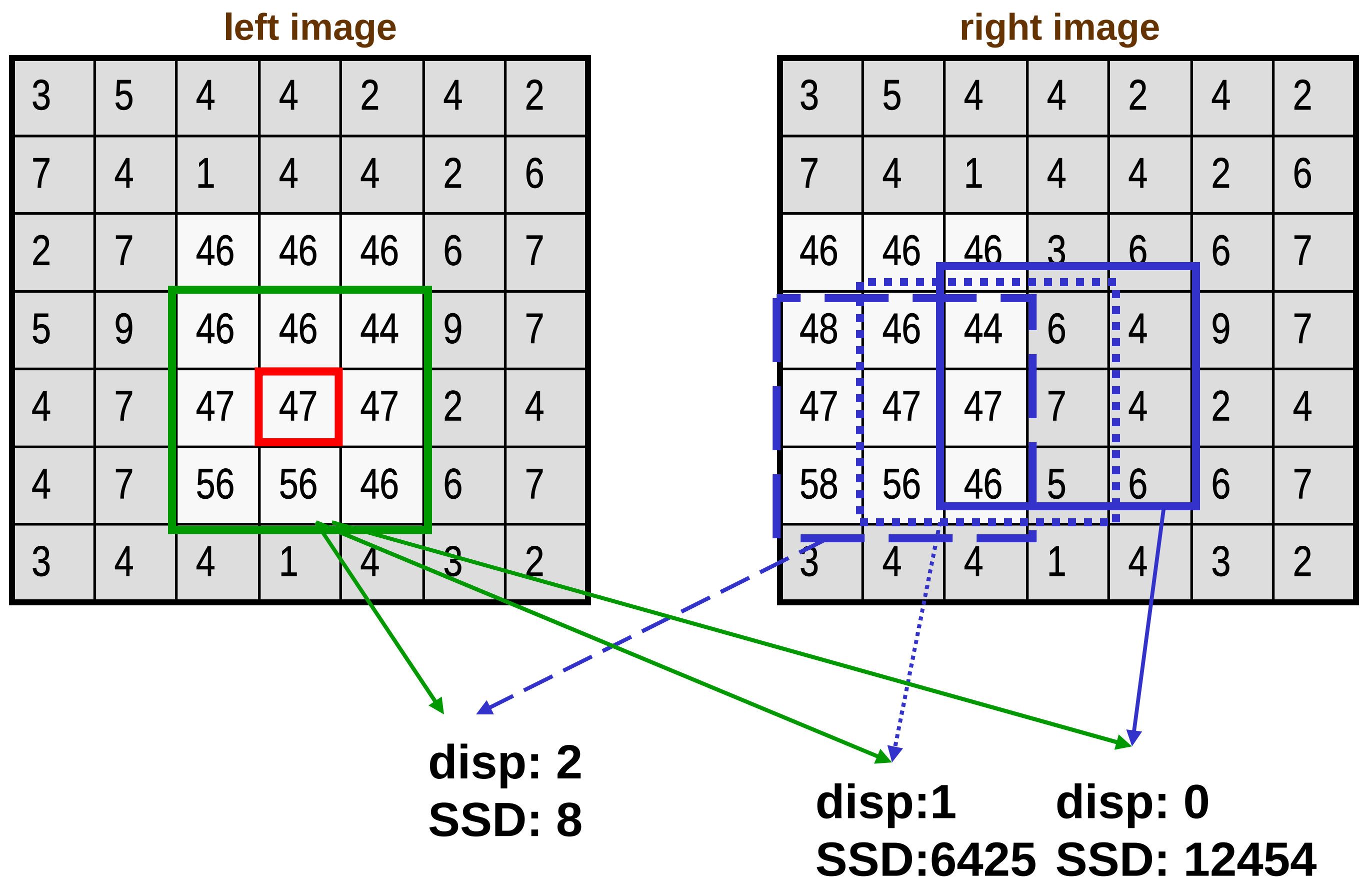
right image

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

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

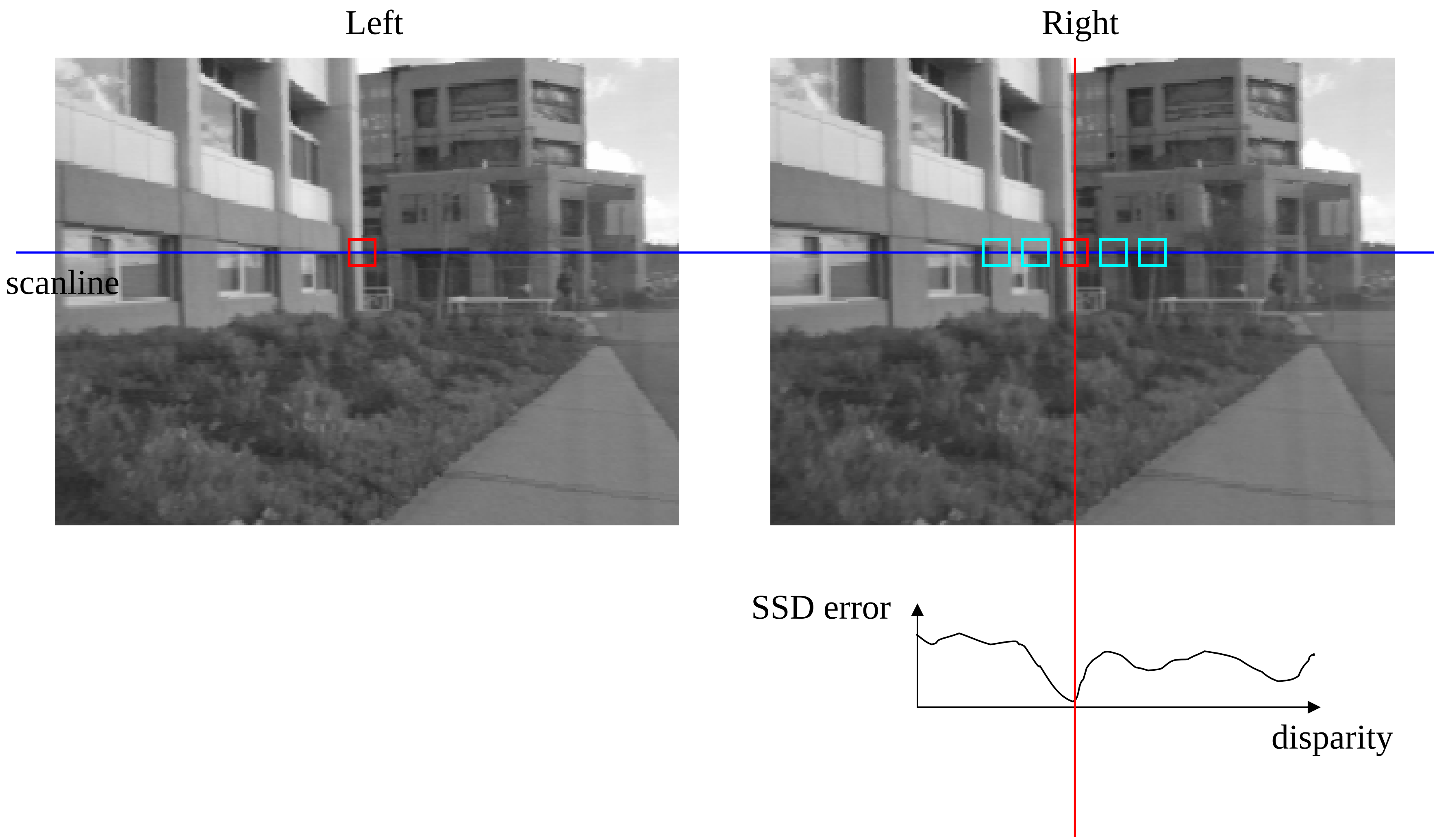
- This shift corresponds to disparity 2
  - All pixels in blue window have x coordinate 2 less than corresponding pixels in the green window

Sum of Squared (Pixel) Differences



- Best SSD window cost (=8) is at disparity 2
  - Red pixel is assigned disparity 2
- Repeat this procedure for all image pixels
- Instead of SSD, can use other window costs:
  - Sum of absolute differences (SAD), normalized correlation, etc.

Correspondence Using SSD matching

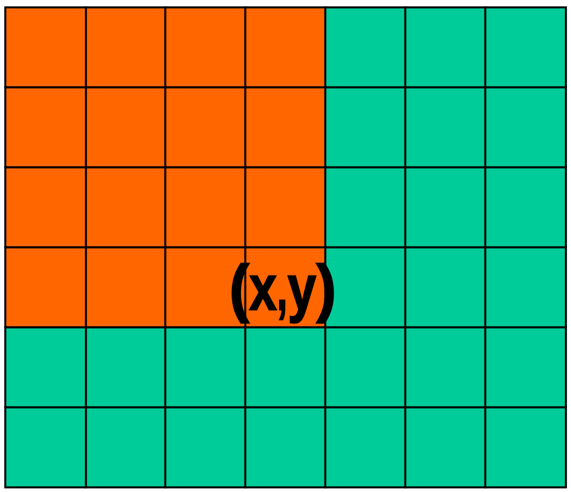


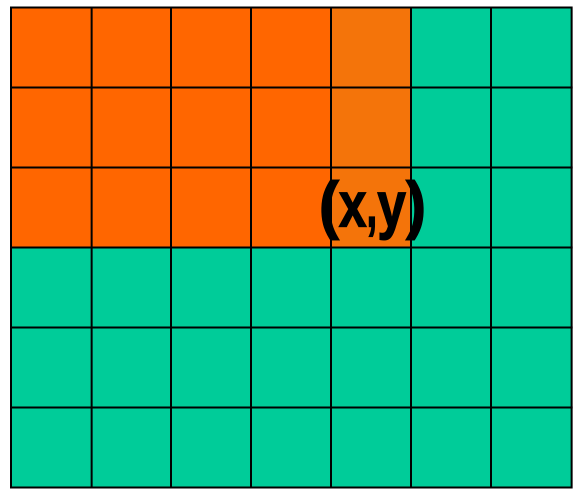
How do we perform window matching efficiently?

- Suppose image is n by n
- Suppose window is 11 by 11
  - Typically windows are taken to be from 11 by 11 to 21 by 21
- Need 11*11=121 additions and multiplications to match 1 window
  - Multiply it by n*n number of image pixels
  - Multiply by number of disparities (maxD+1)
  - TOOOOO SLOOOOOOOW
- For 21 by 21 window, need 21*21=441 multiplications and additions per pixel
  - Multiply it by n*n number of image pixels
  - Multiply by number of disparities (maxD+1)

Integral Image (Crow'84, Viola'2001)

- Let I(x,y) be the sum of image values to the left and above pixel (x,y) **including** pixel (x,y)
  - I(x,y) is the sum of pixel values in the orange area

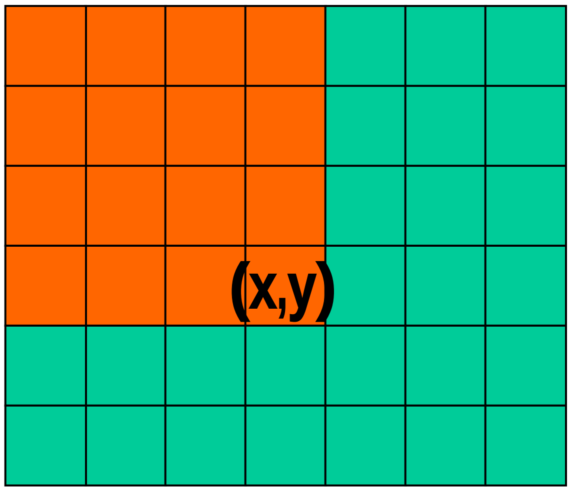


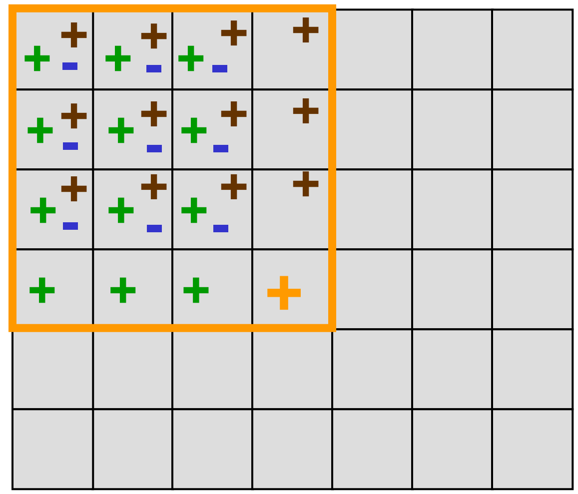


$$I(x,y) = \sum_{(x,y) \in \blacksquare} f(p)$$

Integral Image (Crow'84, Viola'2001)

- How do we compute I(x,y) efficently?





$$I(x,y)=\sum_{(x,y)\,\in} f(p)$$

$$I(x,y)= f(x,y) + I(x-1,y) + I(x,y-1) - I(x-1,y-1)$$

Computing Integral Image I(x,y)

f(0,0)	f(1,0)+I(0,0)	f(2,0)+I(1,0)	f(3,0)+I(2,0)	f(4,0)+I(3,0)
f(0,1)+I(0,0)	f(1,1)+I(0,1)+ I(1,0)-I(0,0)	f(2,1)+I(1,1)+ I(2,0)-I(1,0)	f(3,1)+I(2,1)+ I(3,0)-I(2,0)	f(4,1)+I(3,1)+ I(4,0)-I(3,0)
f(0,2)+I(0,1)	f(1,2)+I(0,2)+ I(1,1)-I(0,1)	f(2,2)+I(1,2)+ I(2,1)-I(1,1)	f(3,2)+I(2,2)+ I(3,1)-I(2,1)	f(4,2)+I(3,2)+ I(4,1)-I(3,1)

Integral Image Cont.

- Integral Image is computed in one pass over the image, with 3 additions/subtractions per pixel
- Start at the top left corner
- Proceed first to the left, and then downwards
  - That is first process the first row, from left to right, then the second row, from left to right,… so on until last row

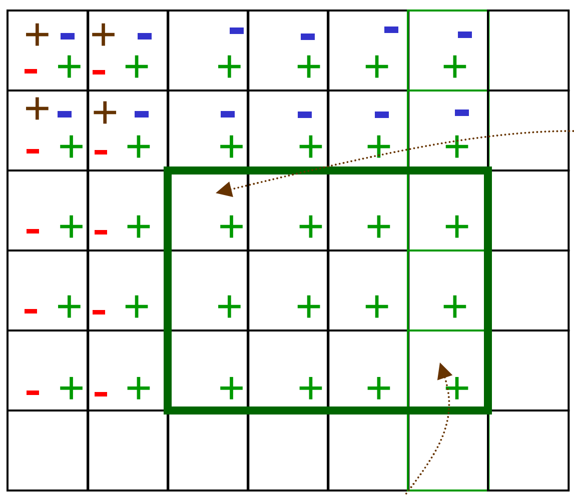
Algorithm Compute ***IntegralImage***

Assumes image has height ***h*** and width ***w*** that is indexes are in [0,***w***-1]x[0,***h***-1]

```
I(0,0) = f(0,0)  // set top left pixel, that is pixel (0,0)
for  x = 1,2,…w-1 do  // set the top row (y = 0) except pixel (0,0)
    I(x,0) = I(x-1,0) + f(x,0)
for  y = 1,2,…h-1 do  // set  leftmost column  (x = 0) except pixel (0,0)
    I(x,y) = I(0,y-1) + f(0,y)
for  y = 1,2,…h-1 do  // set everything else
    for  x = 1,2,…w-1 do
        I(x,y) = I(x,y-1)+I(x-1,y)-I(x-1,y-1)+f(x,y)
```

Integral Image Cont.

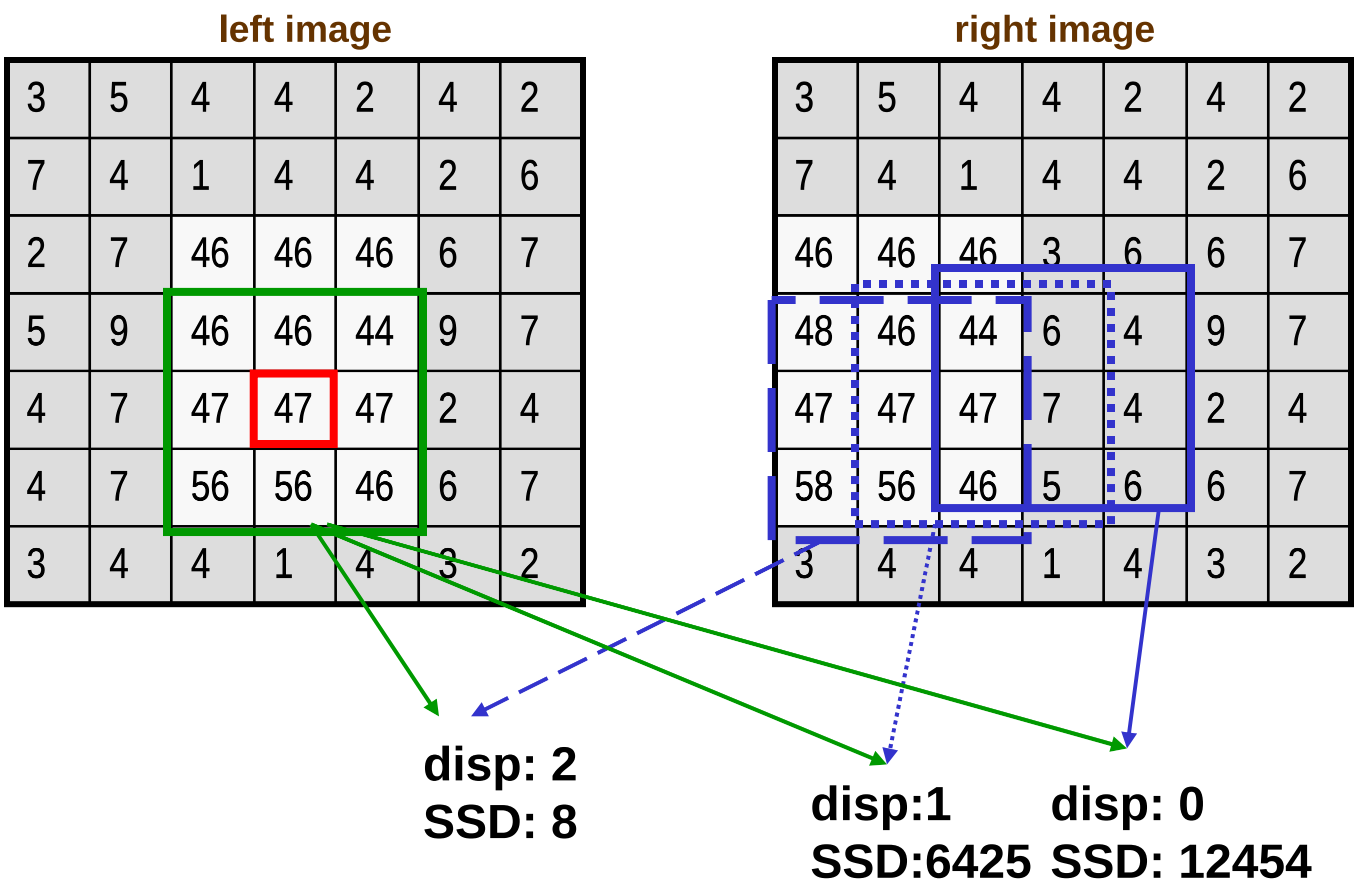
After we have computed the integral image, sum over any rectangular window is computed with only 4 operations!



- To compute sum in a window with top left corner (x_1,y_1) and bottom right corner (x_2,y_2):
  - $I(x_2,y_2)-I(x_1-1,y_2)-I(x_2,y_1-1)+I(x_1-1,y_1-1)$

How to Use Integral Image for window matching?

- Assume we use SSD (sum of absolute differences) window cost
- Recall that we need to find SSD for every pixel and every disparity in a window



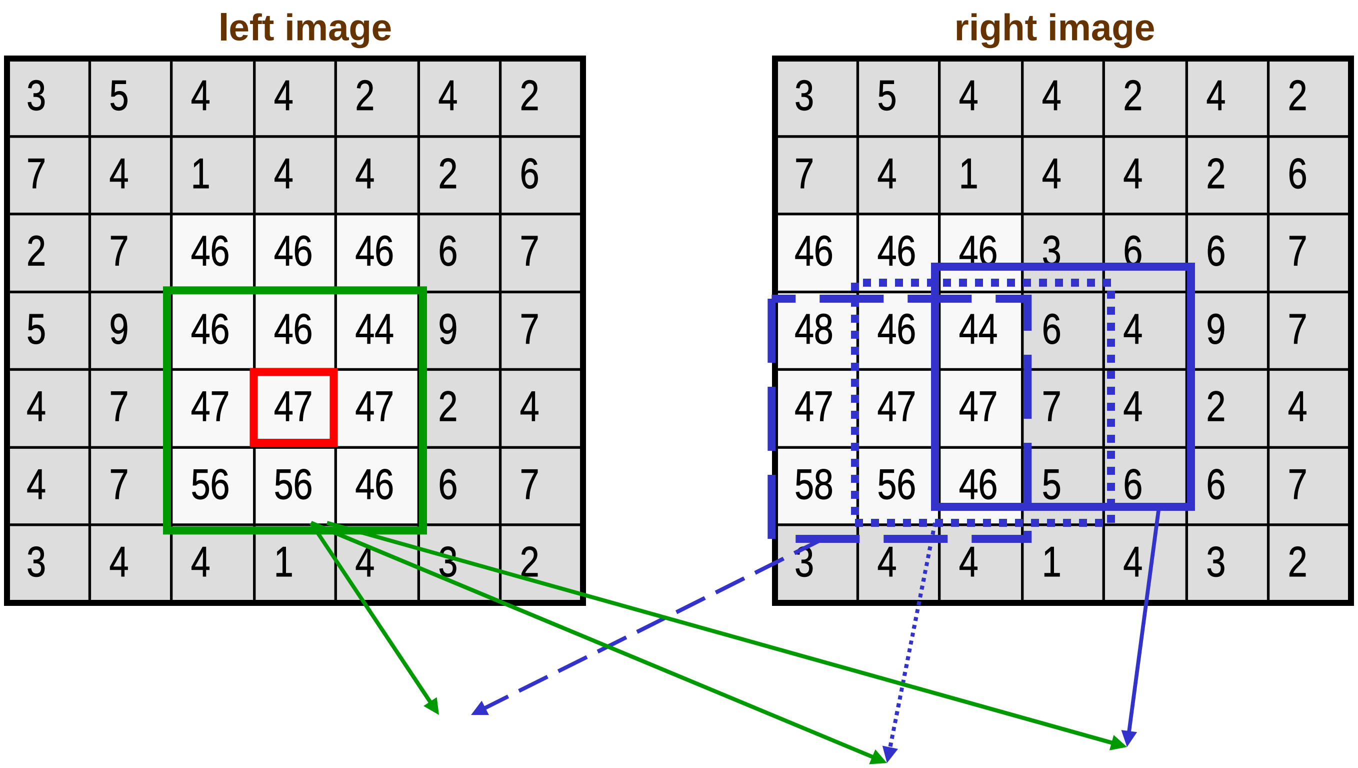
How to Use Integral Image for window matching?

- Old Inefficient Algorithm:

    for every pixel ***p***

        for every disparity ***d***

            compute cost between window around ***p*** in the left image and window around ***p*** shifted by ***d*** to the left in the right image



How to Use Integral Image for window matching?

- For any disparity, say disparity 1, we need to compute window sum **for all pixels**

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

How to Use Integral Image for window matching?

- Old Inefficient Algorithm:

  for every pixel ***p***

  for every disparity ***d*** **reverse**

  compute cost between window around ***p*** in the left image and window around ***p*** shifted by ***d*** to the left in the right image

- What if we reverse the order of computation?
- New Algorithm (can be made efficient):

  for every disparity ***d***

  for every pixel ***p***

  compute cost between window around ***p*** in the left image and window around ***p*** shifted by ***d*** to the left in the right image

  can be done very efficiently with integeral image computation

How to Use Integral Image for window matching?

- Suppose current disaprity is 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

- This is equivalent to
  - overlaying left and right image at disparity 1
  - Computing SAD between every pair of pixels for the overlaid part
  - Computing SAD in a window for every pixel

How to Use Integral Image for window matching?

- current disaprity is 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

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

SAD 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

How to Use Integral Image for window matching?

Current disparity is 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

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

⇨

SAD 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

How to Use Integral Image for window matching?

Current disparity is 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

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

SAD 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

How to Use Integral Image for window matching?

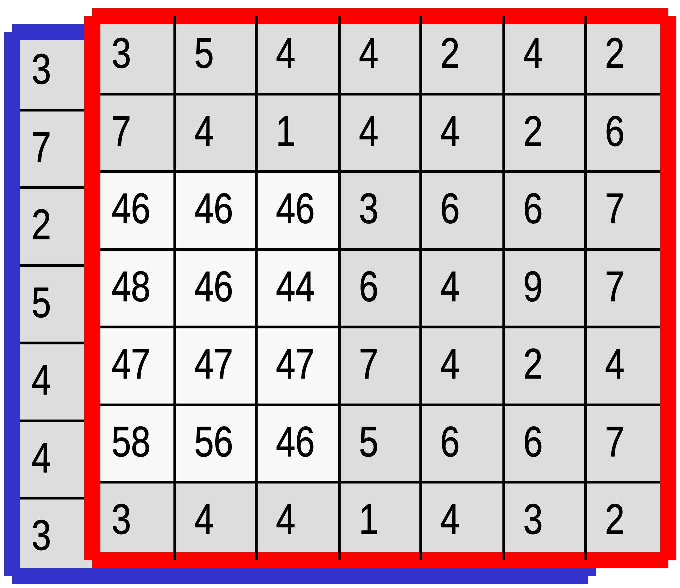
Current disparity is 1

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



SAD 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

How to Use Integral Image for window matching?

- Current disparity is 1
- Notice how we have to compute window sums in SAD image for disparity 1
  - 1 window sum for each image pixel
- Use the integral image technique on the SAD image!

SAD 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

Integral Image for stereo

New Efficient Algorithm :

for every pixel ***p*** **do**

bestDisparity[p] = 0

bestWindowCost[p] = HUGE

for disparity ***d*** **=** 0, 1,…,maxD **do**

Overlay images at disparity ***d***

Compute SAD image for disparity ***d***

Compute Integral image from SAD image

for every pixel ***p*** **do**

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

if currentCost < bestCost[***p***]

bestCost[***p***] = curentCost

bestDisparity[***p***] = ***d***

return bestDisparity

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	1	0
1	0	3	3	1	0

SAD image

How to Use Integral Image for window matching?

- For simpler implementation, make SAD image the same size as the left image and add ***d*** columns of zeros on the left
  - for disparity 1, add 1 “fake” column of zeros
  - For disparity 2, add 2 “fake” columns of zeros
  - …..
- Now (x,y) coordinates between left image and SAD image coincide
- If you want to simplify things even further, pad the SAD image with a border of zeros on all sides
  - size of the border = window radius

SAD image for disparity 1

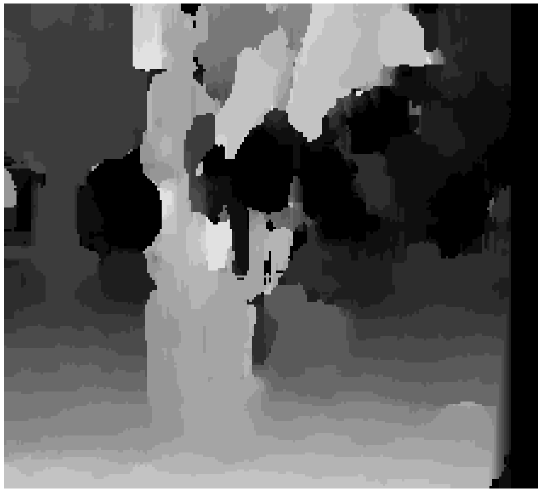
0	2	1	0	2	2	2
0	3	3	3	0	4	0
0	39	0	0	43	1	0
0	39	0	2	38	2	0
0	40	0	0	40	2	0
0	51	0	10	41	0	0
0	1	0	3	3	1	0

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

Window size

W = 3



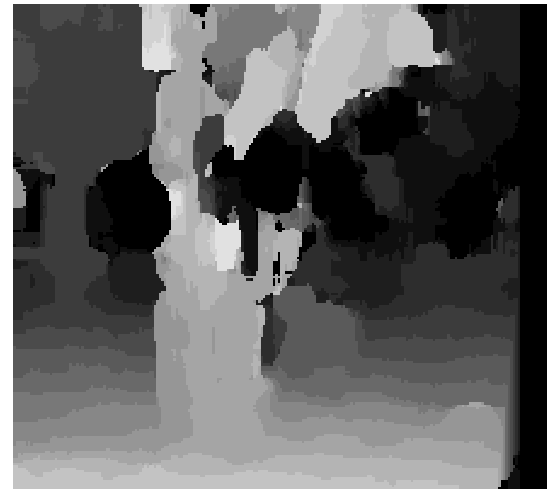
W = 20

Effect of window size

- Smaller window
  - + discontinuity boundaries are preserved
  - – low texture regions are noisy
- Larger window
  - + less noise in low texture regions are
  - – discontinuity boundaries are not preserved

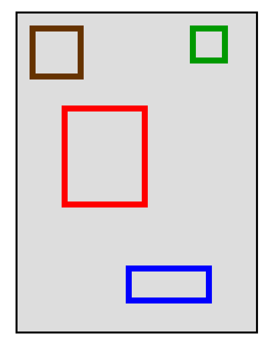
Window size

W = 3



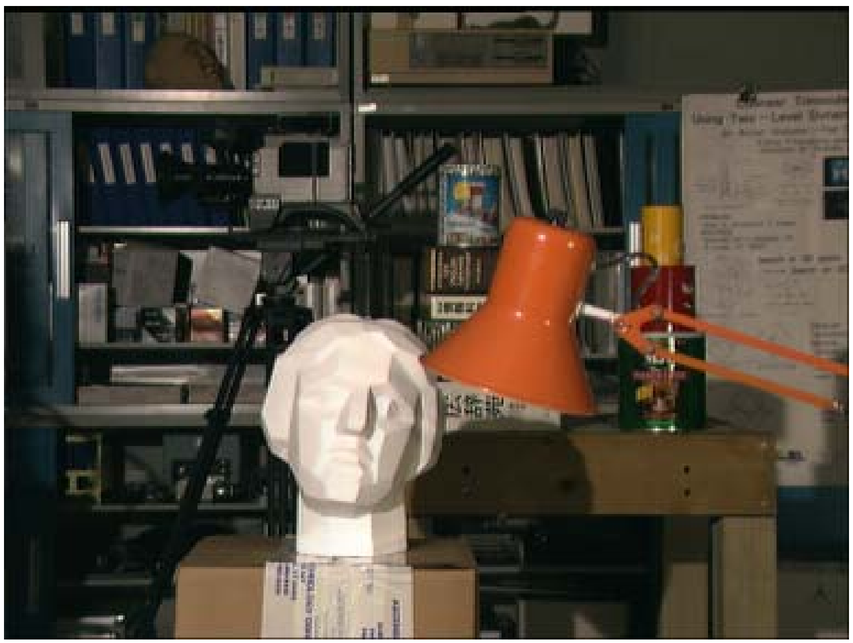
W = 20

- With integral image technique, can compute sum in a window of any rectangular size very efficiently
- Question: where to use a small window, where to use a large window?

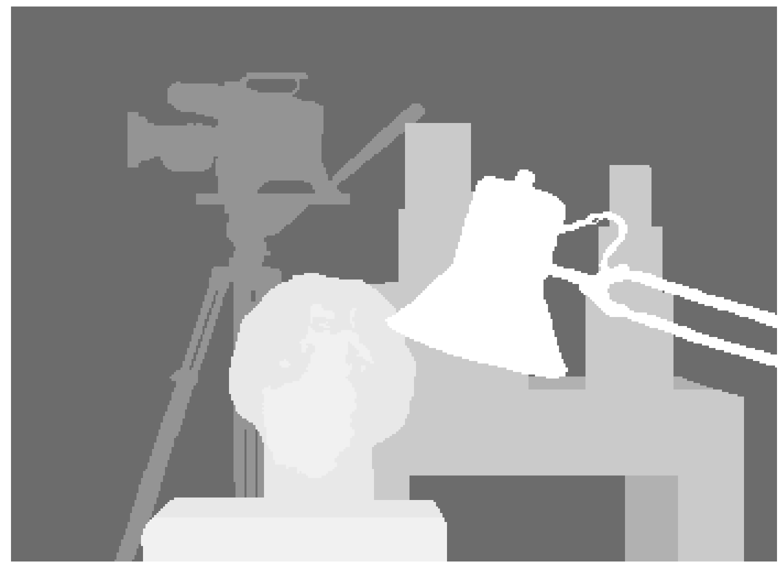


Stereo results

- Data from University of Tsukuba
- Similar results on other images without ground truth

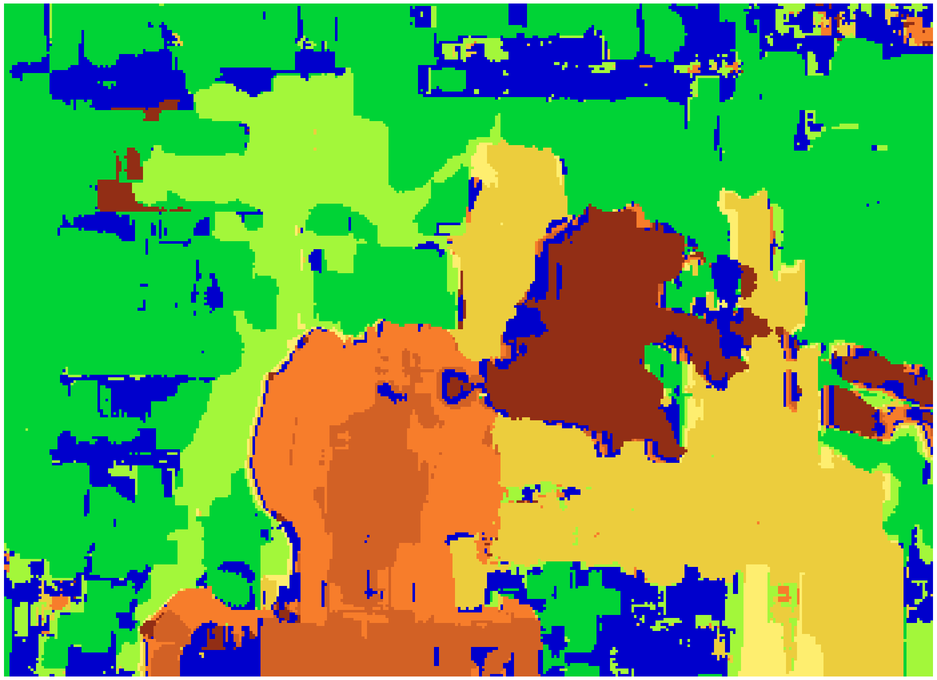


Scene



Ground truth

Results with window search



Window-based matching
(best window size)

Ground truth

Better methods exist...

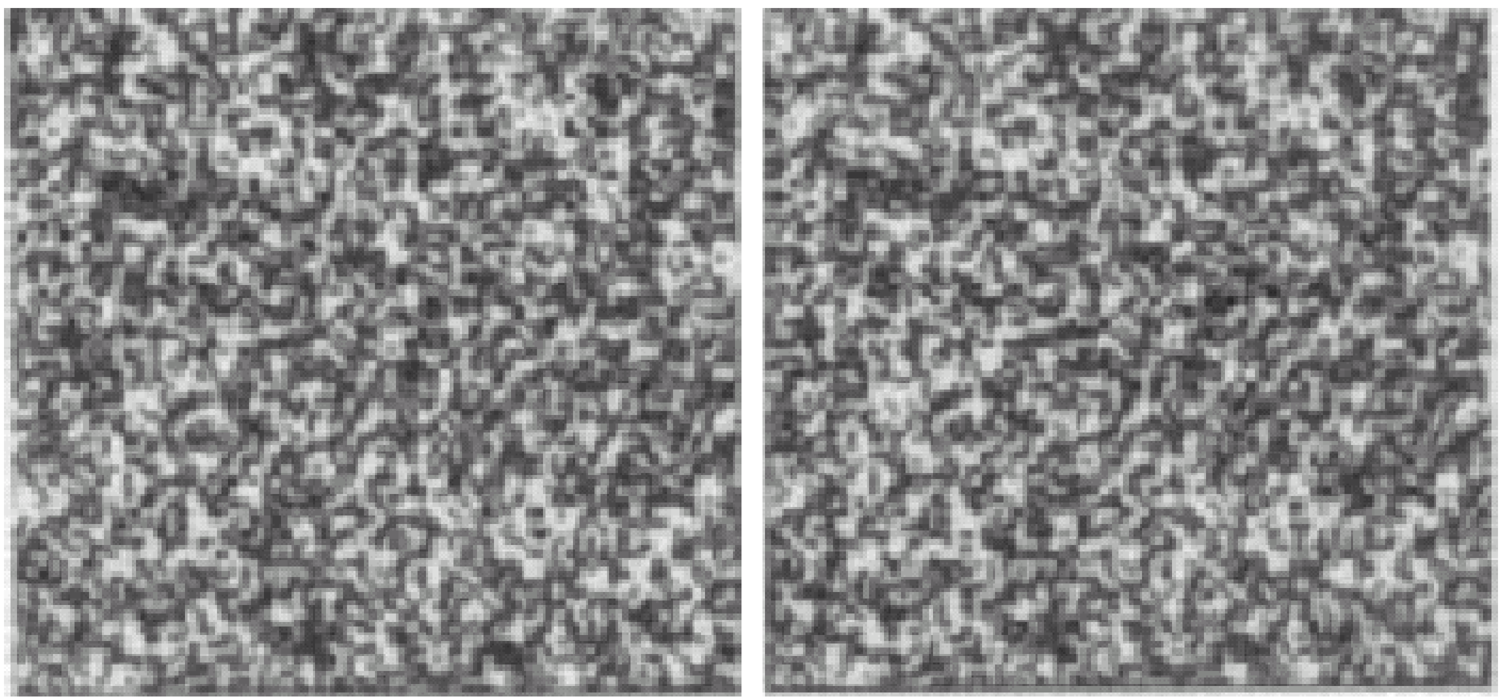
State of the art method

Ground truth

Boykov, Veksler, Zabih, [Fast Approximate Energy Minimization via Graph Cuts](), International Conference on Computer Vision, September 1999.

For the latest and greatest: http://www.middlebury.edu/stereo/

Random dot stereograms



Julesz: showed that recognition is not needed for stereo.

Video View Interpolation

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

Real-time stereo

Nomad robot searches for meteorites in Antartica
http://www.frc.ri.cmu.edu/projects/meteorobot/index.html

Used for robot navigation (and other tasks)

- Several software-based real-time stereo techniques have been developed (most based on simple window matching)

Stereo reconstruction pipeline

- Steps
  - Calibrate cameras
  - Rectify images
  - Compute disparity
  - Estimate depth

What will cause errors?

  - Camera calibration errors
  - Poor image resolution
  - Occlusions
  - Violations of brightness constancy (specular reflections)
  - Low-contrast image regions