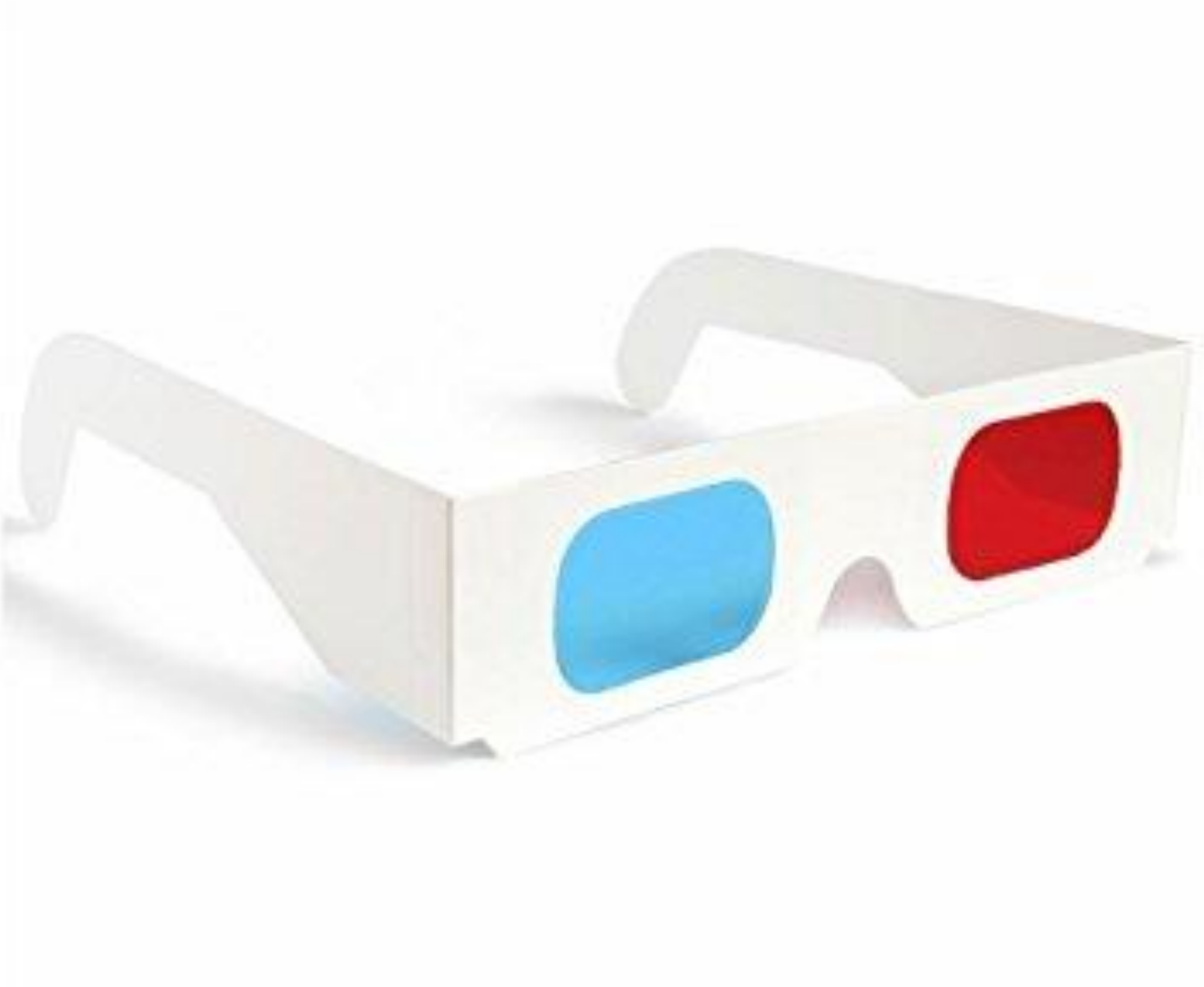
Stereo



16-385 Computer Vision
Spring 2020, Lecture 13

http://www.cs.cmu.edu/~16385/

Course announcements

- Homework 3 is due on March 4th.
  - How many of you have looked at/started/finished homework 3?

- Take-home quiz 5 is due on March 1st.

Overview of today’s lecture

- Leftover from two-view geometry.
- Revisiting triangulation.
- Disparity.
- Stereo rectification.
- Stereo matching.
- Improving stereo matching.
- Structured light.

Slide credits

Some of these slides were adapted directly from:

- Kris Kitani (16-385, Spring 2017).
- Srinivasa Narasimhan (16-823, Spring 2017).

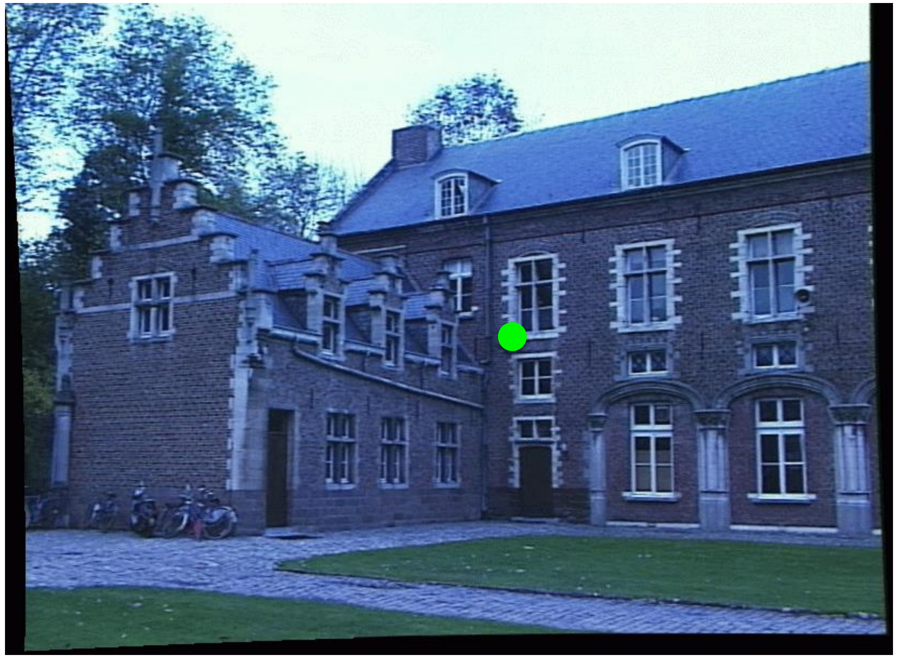
Revisiting triangulation

How would you reconstruct 3D points?

Left image

Right image

How would you reconstruct 3D points?

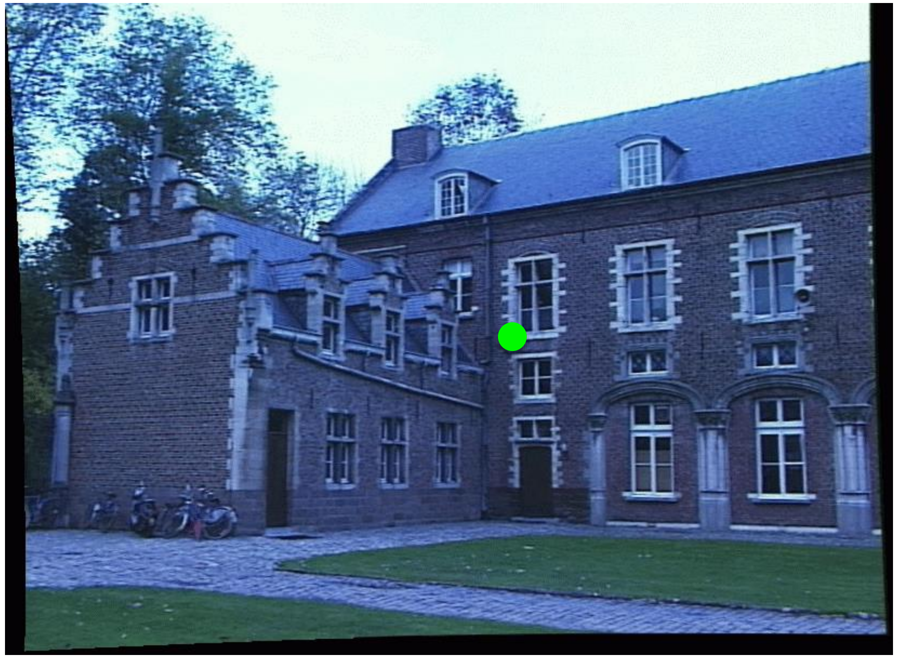


Left image

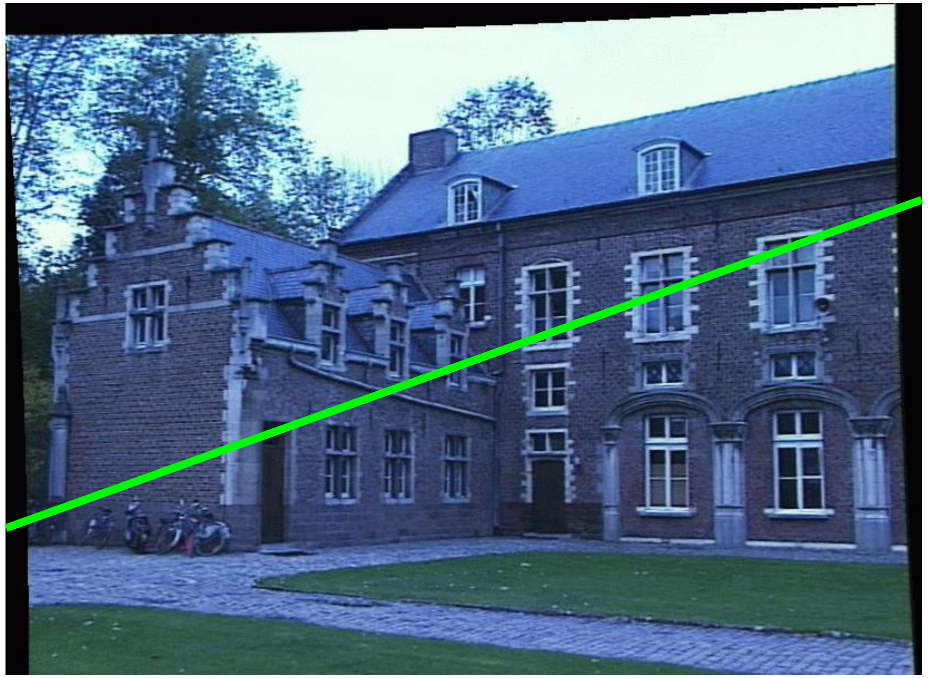
Right image

1. Select point in one image (how?)

How would you reconstruct 3D points?



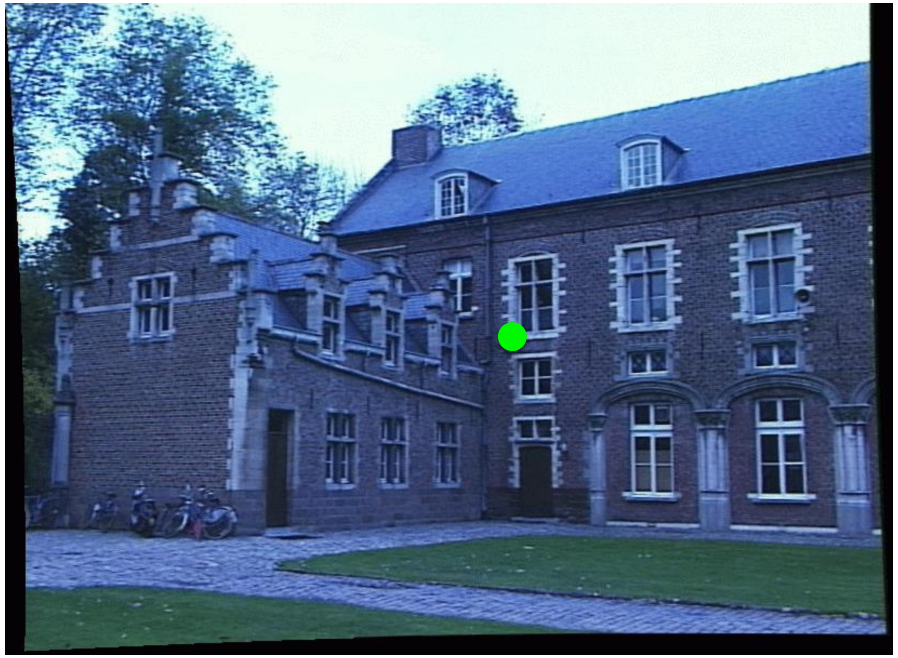
Left image



Right image

1. Select point in one image (how?)
2. Form epipolar line for that point in second image (how?)

How would you reconstruct 3D points?

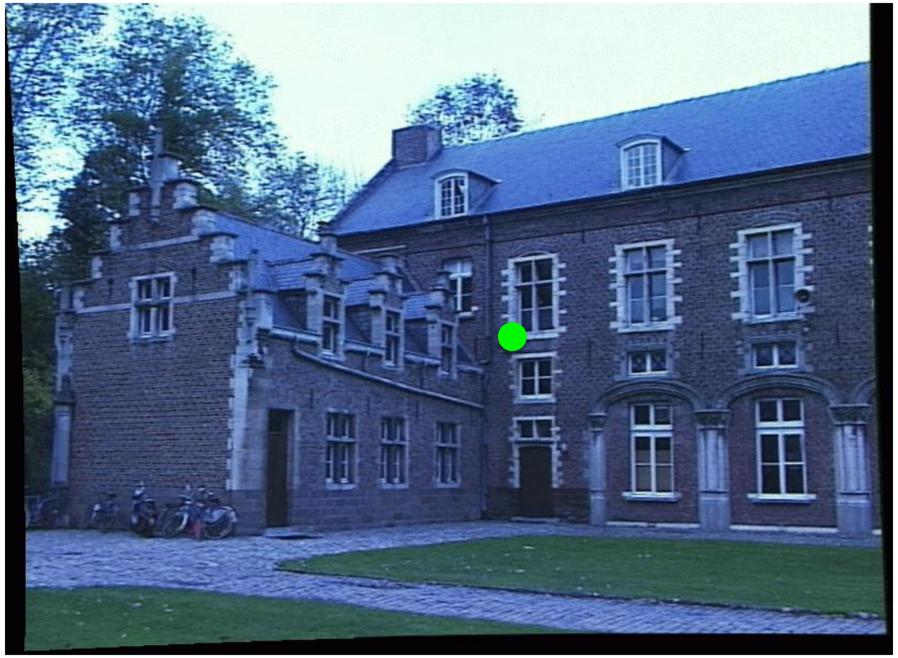


Left image

Right image

1. Select point in one image (how?)
2. Form epipolar line for that point in second image (how?)
3. Find matching point along line (how?)

How would you reconstruct 3D points?

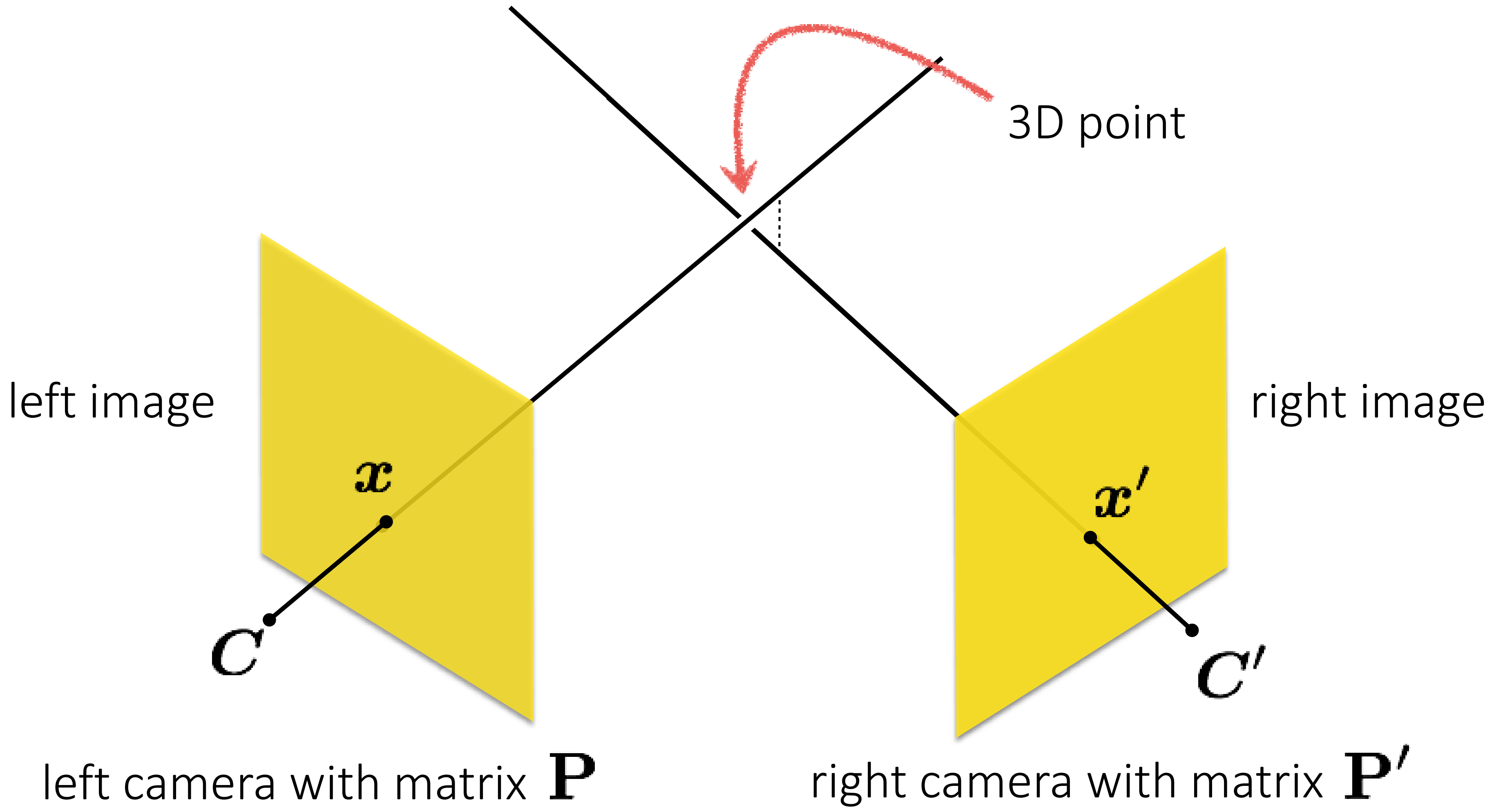


Left image

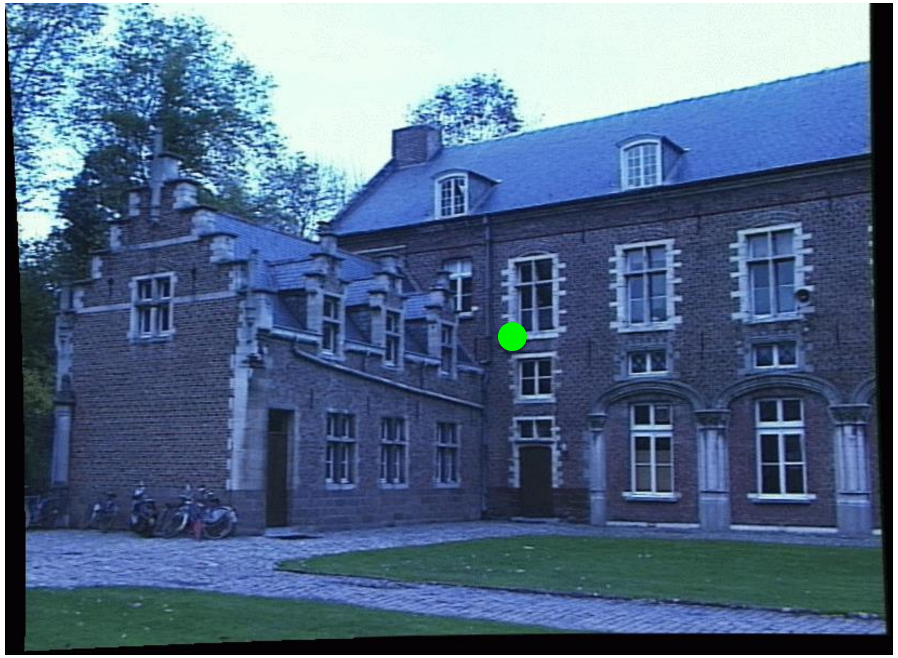
Right image

1. Select point in one image (how?)
2. Form epipolar line for that point in second image (how?)
3. Find matching point along line (how?)
4. Perform triangulation (how?)

Triangulation



How would you reconstruct 3D points?



Left image

Right image

1. Select point in one image (how?)
2. Form epipolar line for that point in second image (how?)
3. Find matching point along line (how?)
4. Perform triangulation (how?)

What are the disadvantages of this procedure?

Stereo rectification

What's different between these two images?

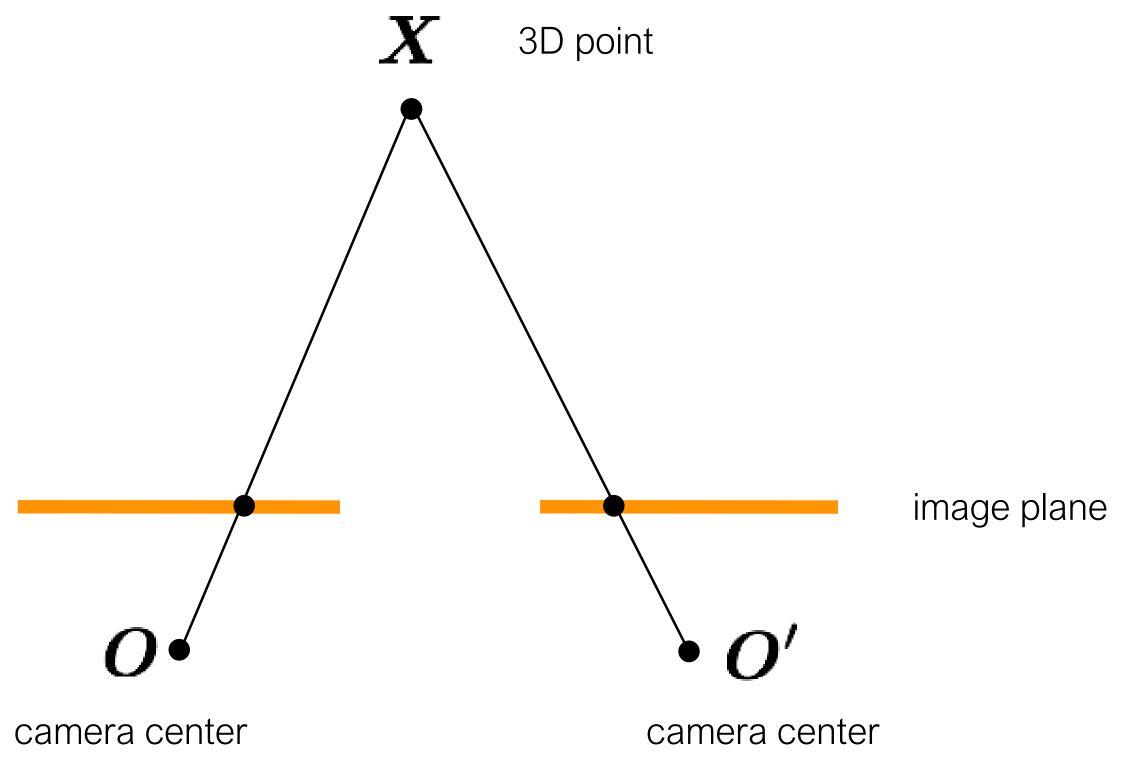
Objects that are close move more or less?

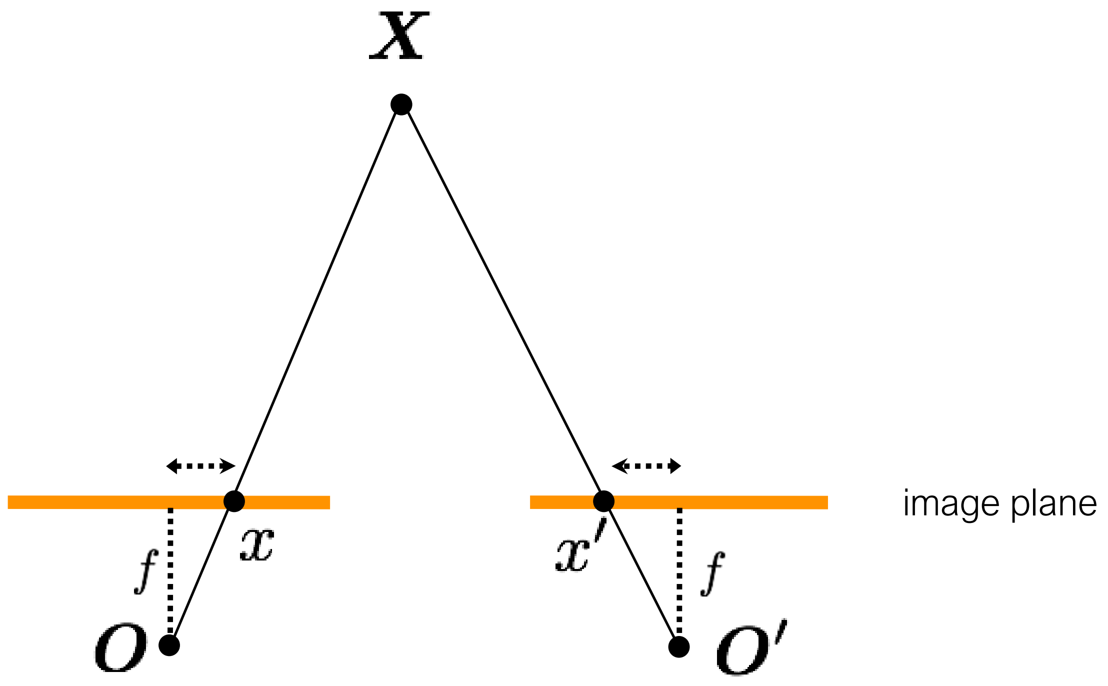
The amount of horizontal movement is inversely proportional to …

The amount of horizontal movement is inversely proportional to …

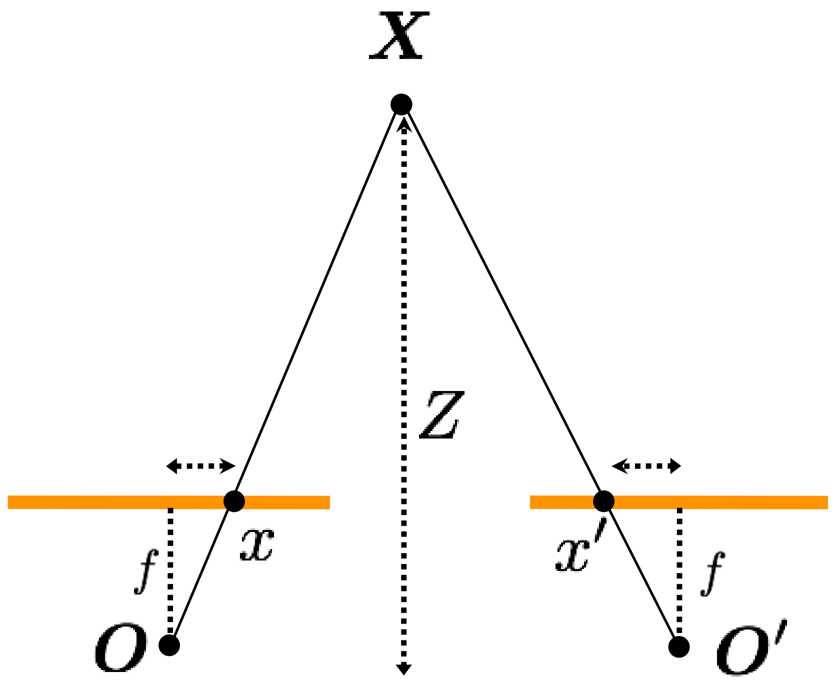
… the distance from the camera.

More formally…

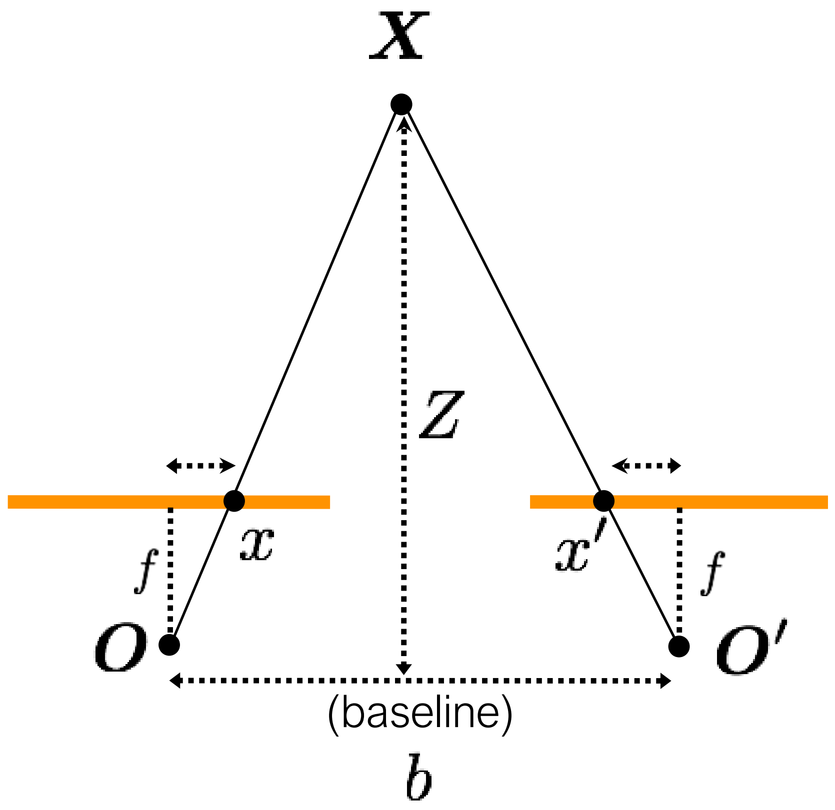

X
3D point
image plane
O
camera center
O'
camera center



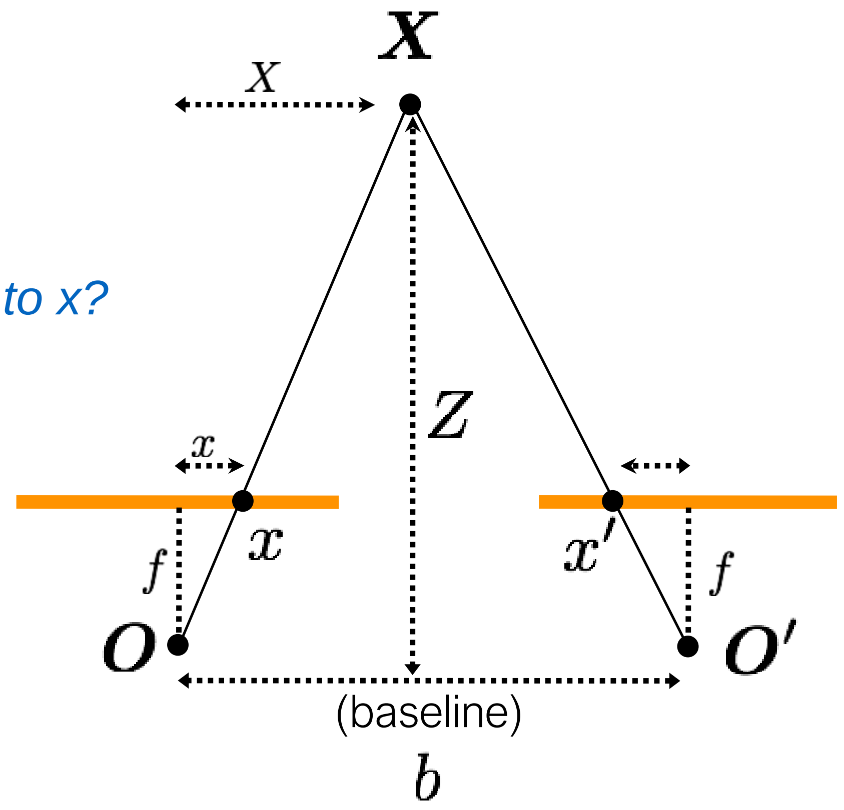
X
x
x'
f
f
O
O'
image plane



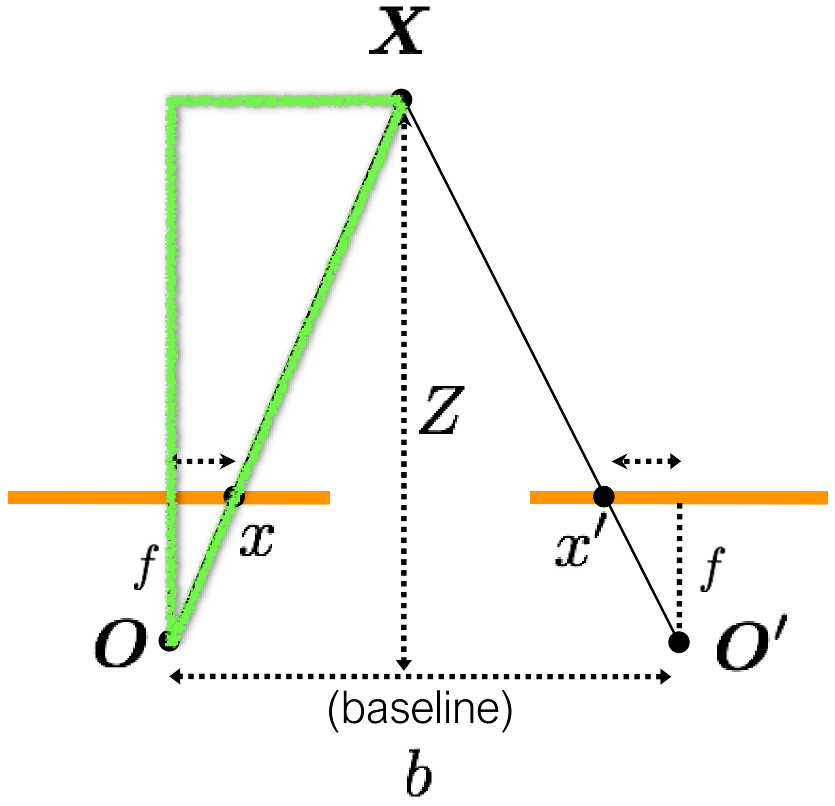
X
Z
x
x'
f
f
O
O'


$\boldsymbol{X}$
Z
x
x'
f
f
$\boldsymbol{O}$
$\boldsymbol{O}'$
(baseline)
b

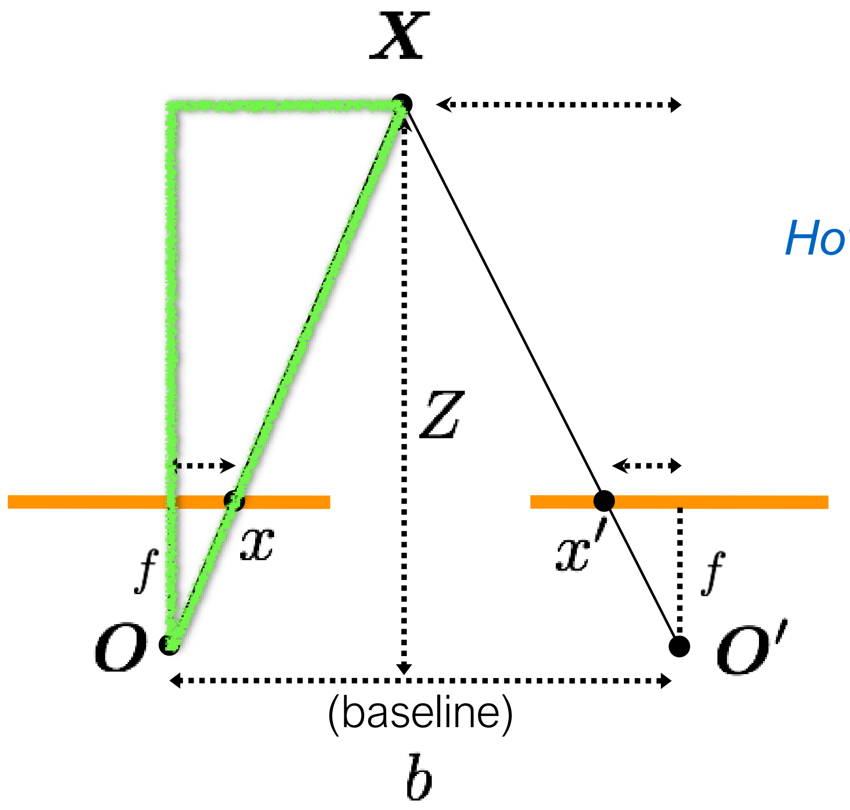
How is X related to x?



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

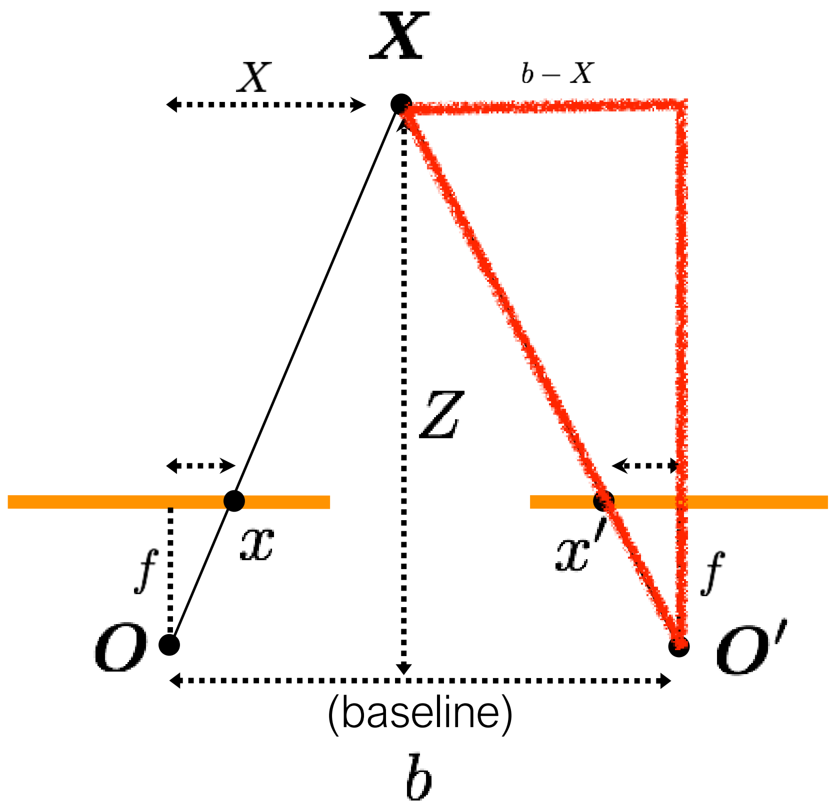


$$\frac{X}{Z} = \frac{x}{f}$$



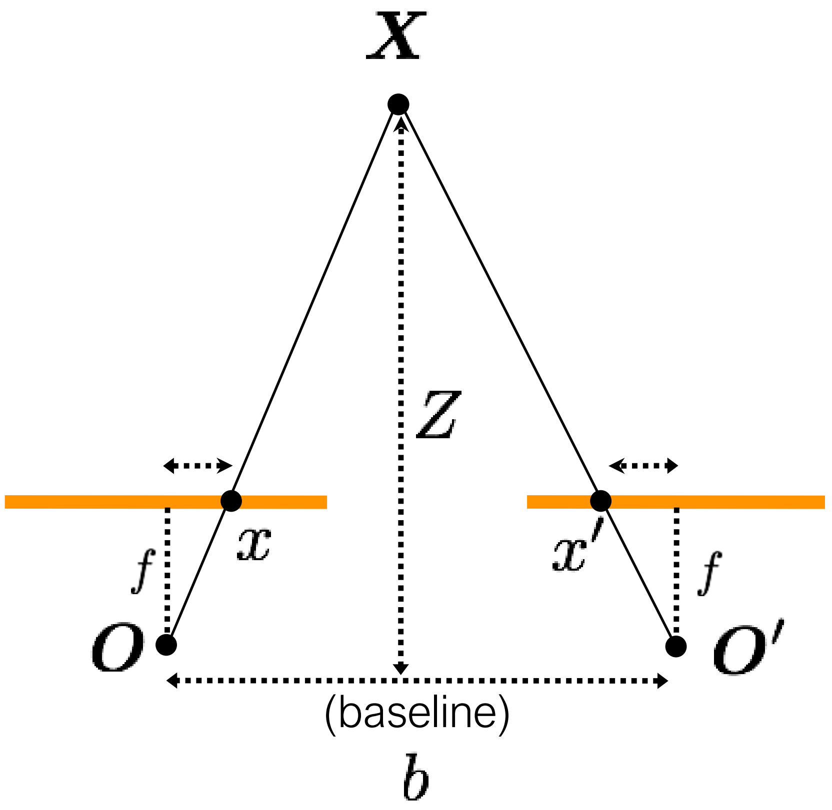
How is X related to x'?

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



$$\frac{b - X}{Z} = \frac{x'}{f}$$

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



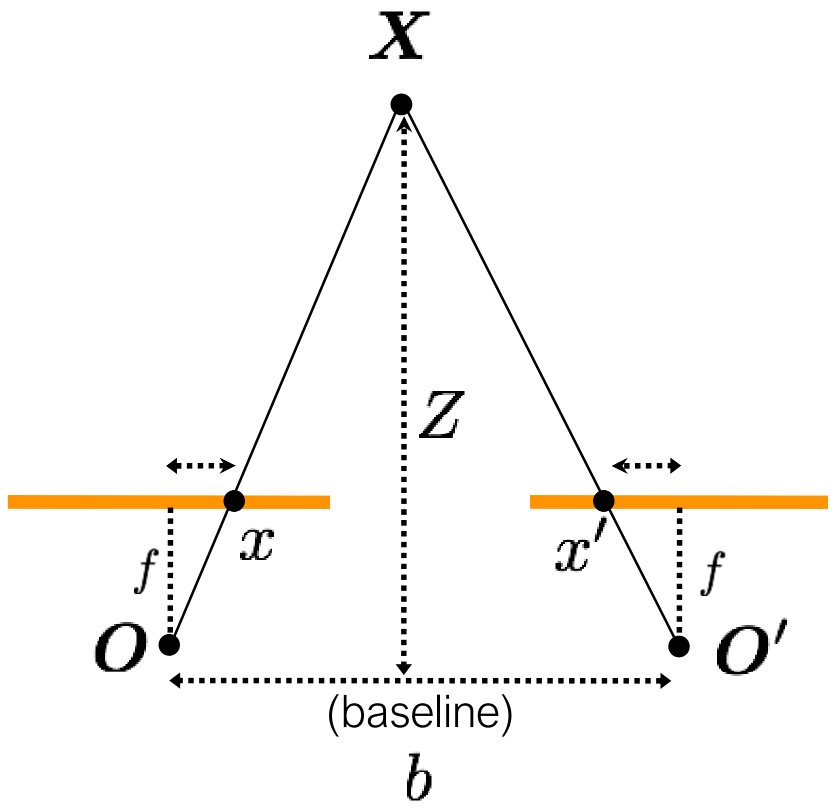
$$\frac{b - X}{Z} = \frac{x'}{f}$$

Disparity

$$d = x - x'$$ (wrt to camera origin of image plane)

$$= \frac{bf}{Z}$$

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



$$\frac{b - X}{Z} = \frac{x'}{f}$$

Disparity

$$d = x - x'$$
$$= \frac{bf}{Z}$$

inversely proportional to depth

Real-time stereo sensing

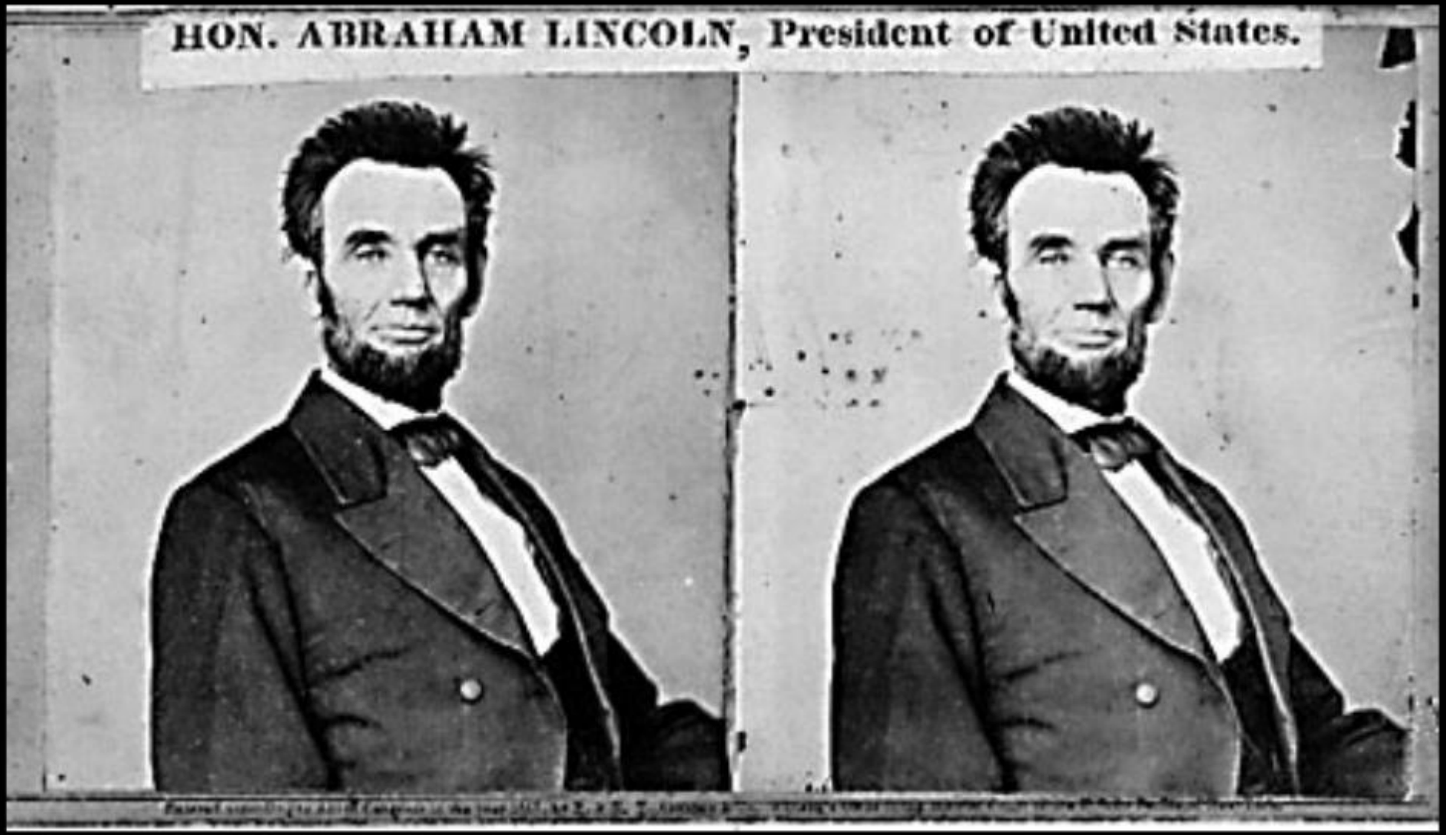
Nomad robot searches for meteorites in Antartica
http://www.frc.ri.cmu.edu/projects/meteorobot/index.html

Subaru
Eyesight system

Pre-collision
braking

What other vision system uses disparity for depth sensing?

Stereoscopes: A 19th Century Pastime

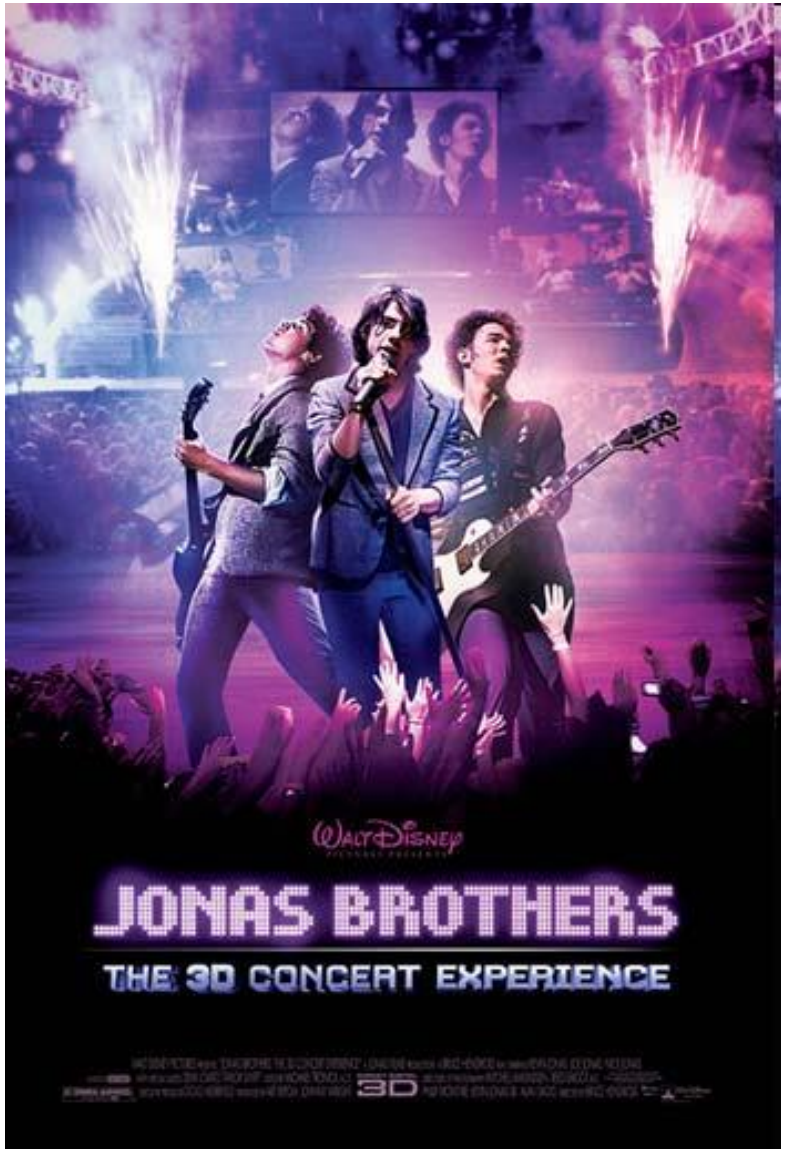

HON. ABRAHAM LINCOLN, President of United States.

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

Teesta suspension bridge-Darjeeling, India

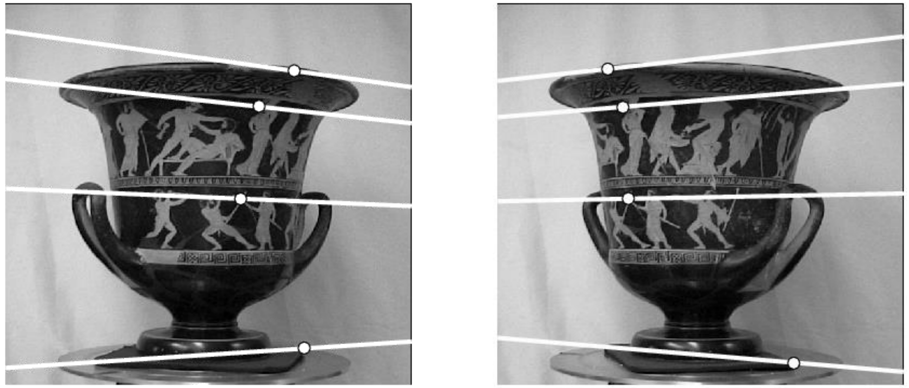
Mark Twain at Pool Table", no date, UCR Museum of Photography

This is how 3D movies work

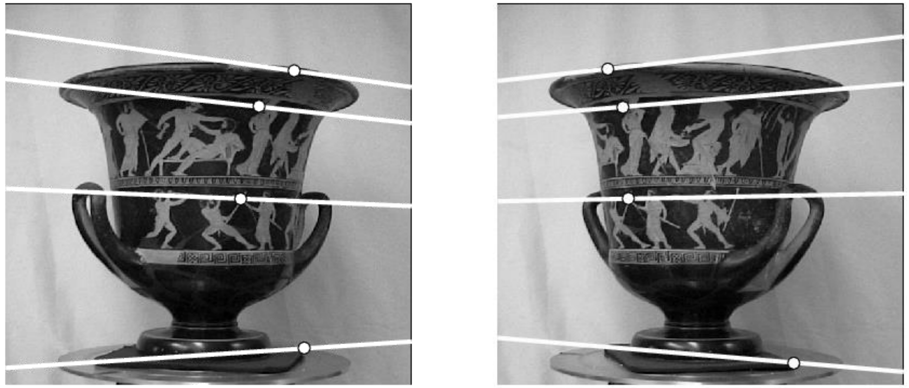


Is disparity the only depth cue the human visual system uses?

So can I compute depth from any two images of the same object?

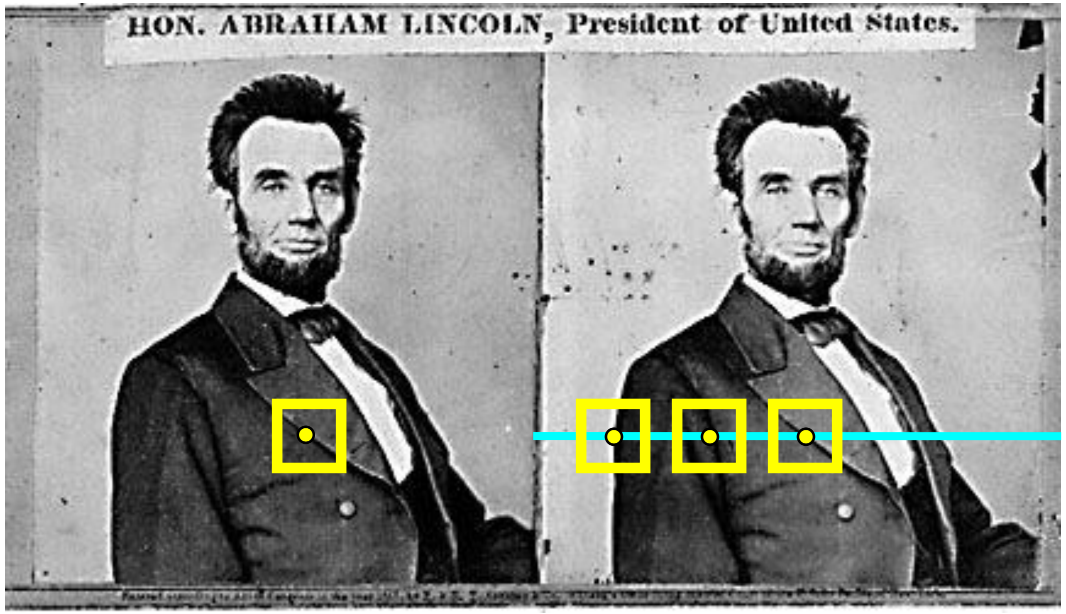


So can I compute depth from any two images of the same object?



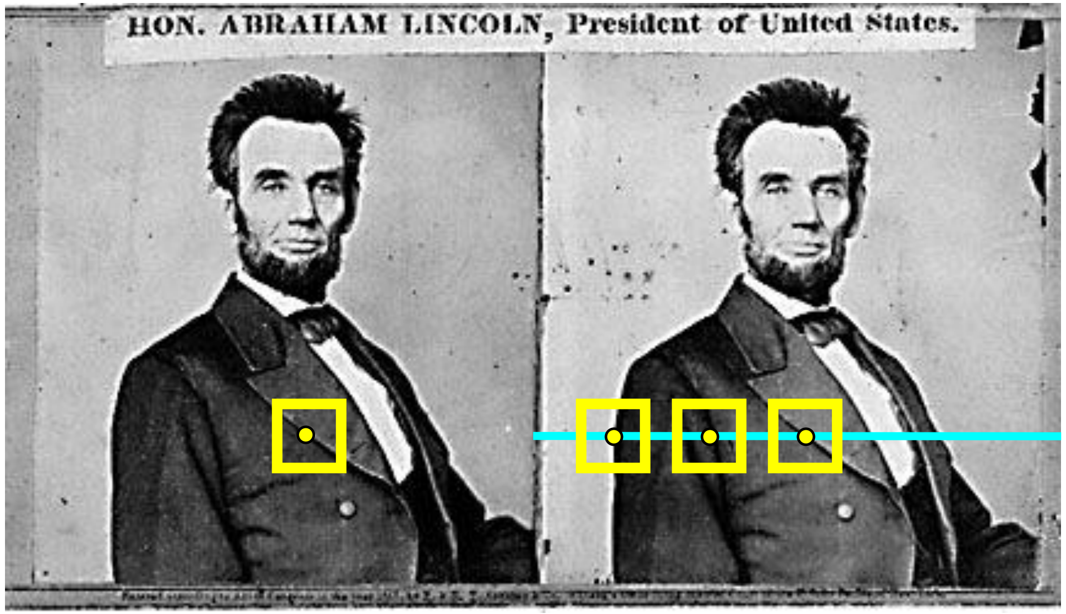
1. Need sufficient baseline

2. Images need to be ‘rectified’ first (make epipolar lines horizontal)

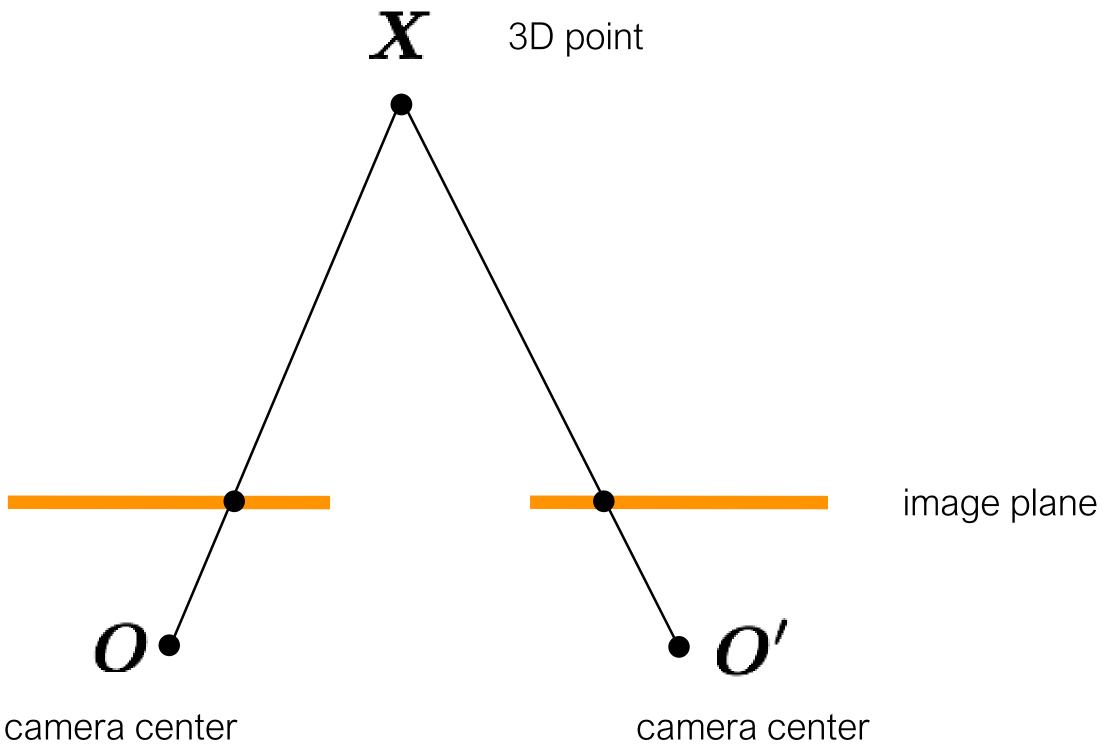


1. Rectify images
   (make epipolar lines horizontal)
2. For each pixel
   a. Find epipolar line
   b. Scan line for best match
   c. Compute depth from disparity

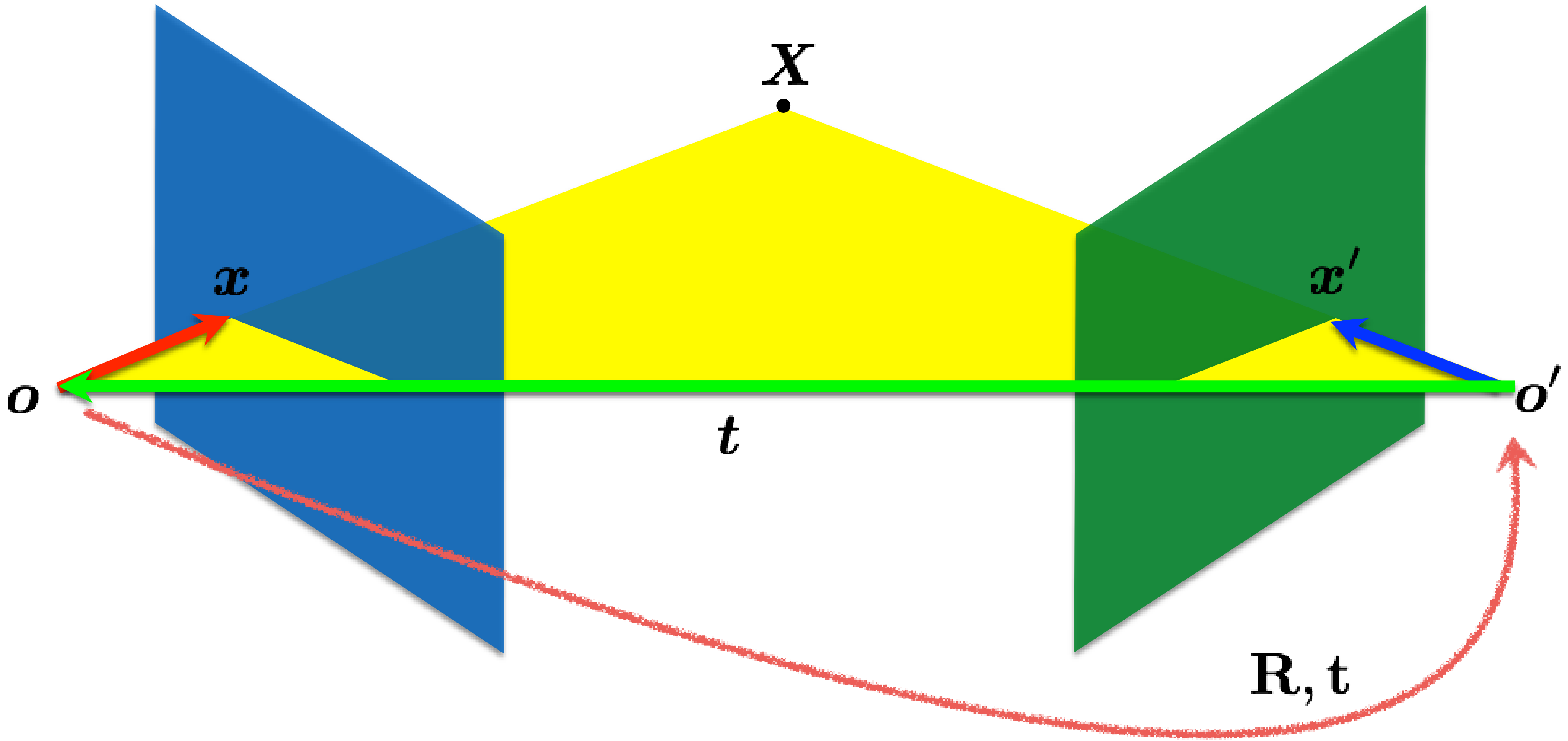
$$Z = \frac{bf}{d}$$



How can you make the epipolar lines horizontal?

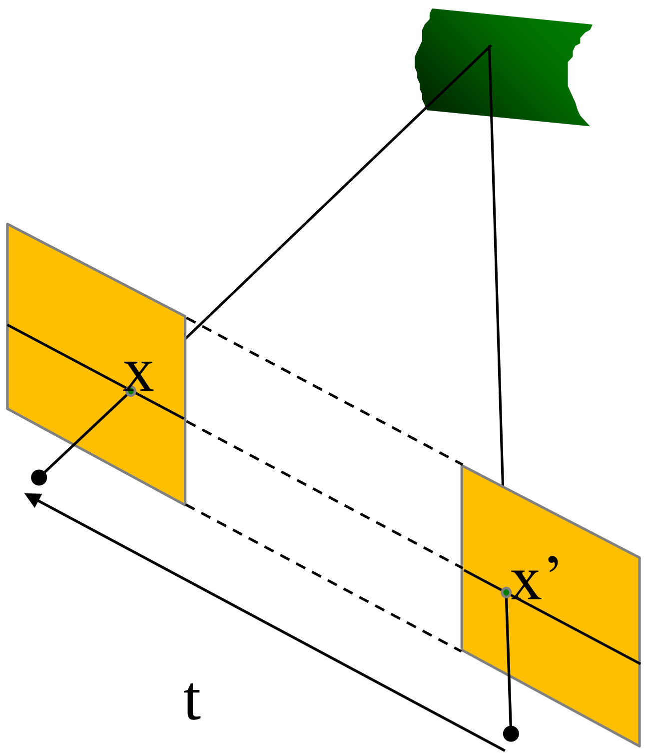


What’s special about these two cameras?



$$\boldsymbol{x}' = \mathbf{R}(\boldsymbol{x} - \boldsymbol{t})$$

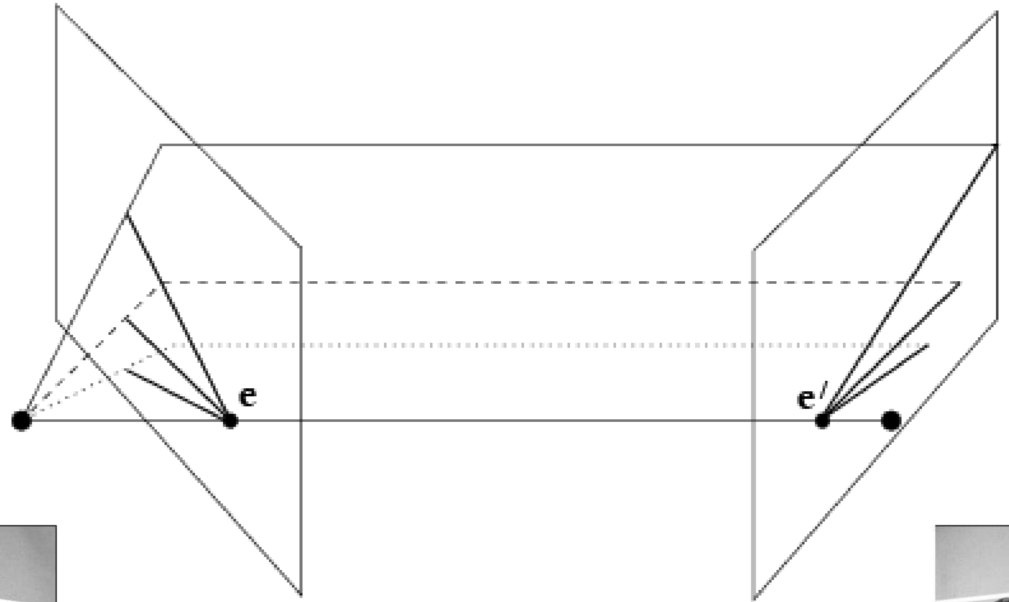
When are epipolar lines horizontal?

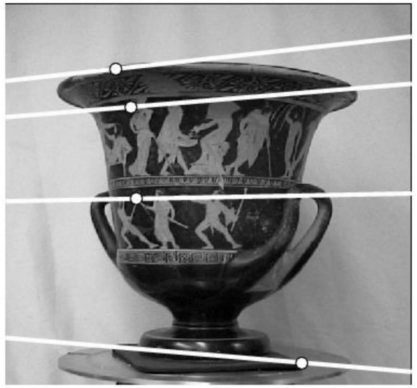


When this relationship holds:

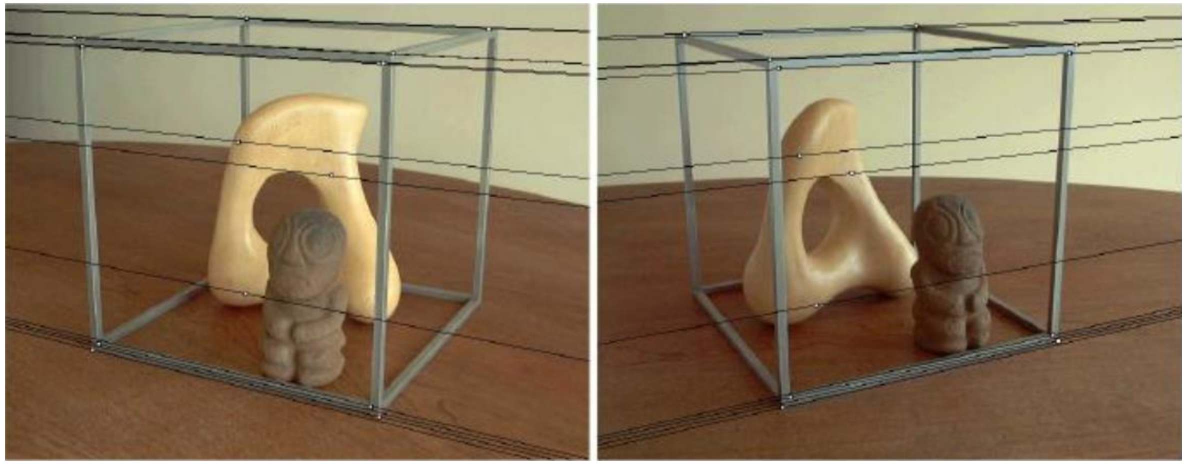
$$\boxed{R = I \qquad\qquad t = (T, 0, 0)}$$

Proof in take-home quiz 5

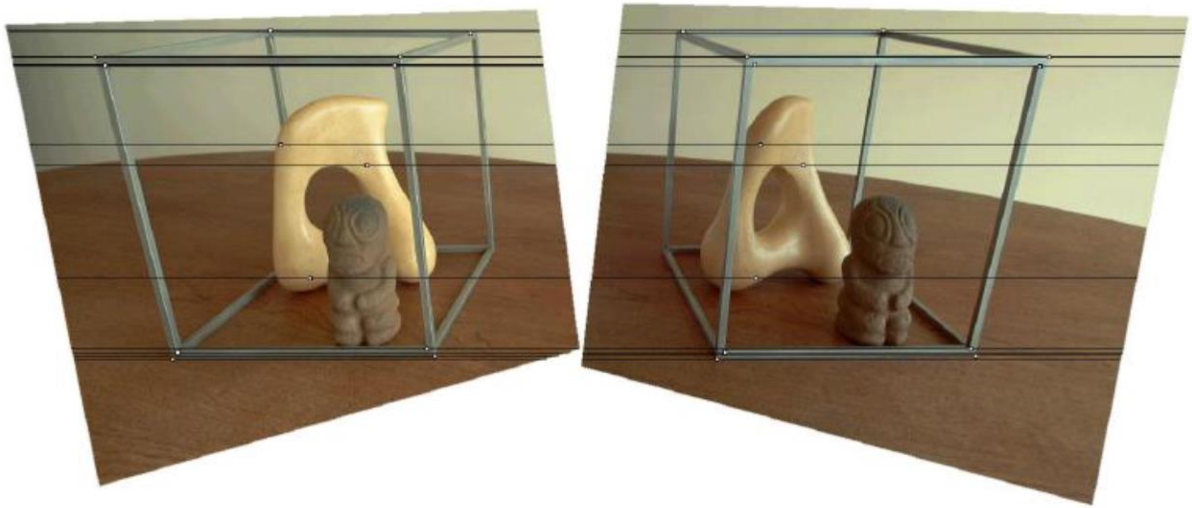


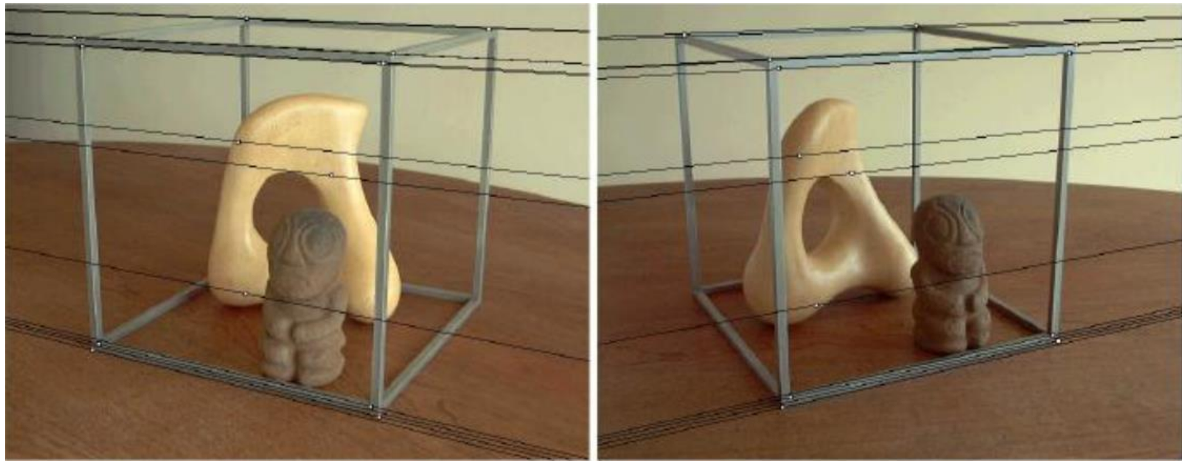


It’s hard to make the image planes exactly parallel

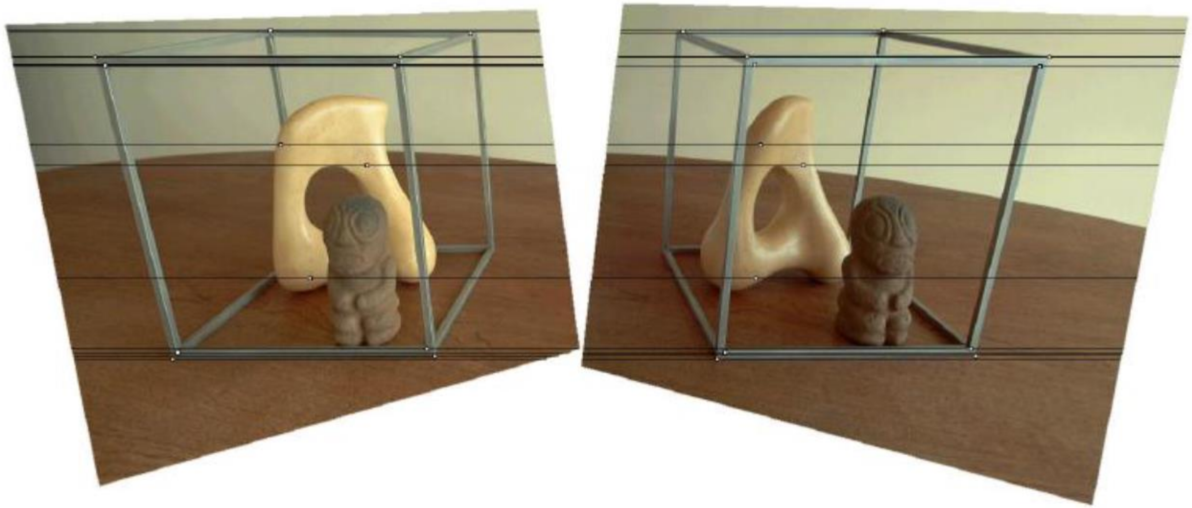


How can you make the epipolar lines horizontal?

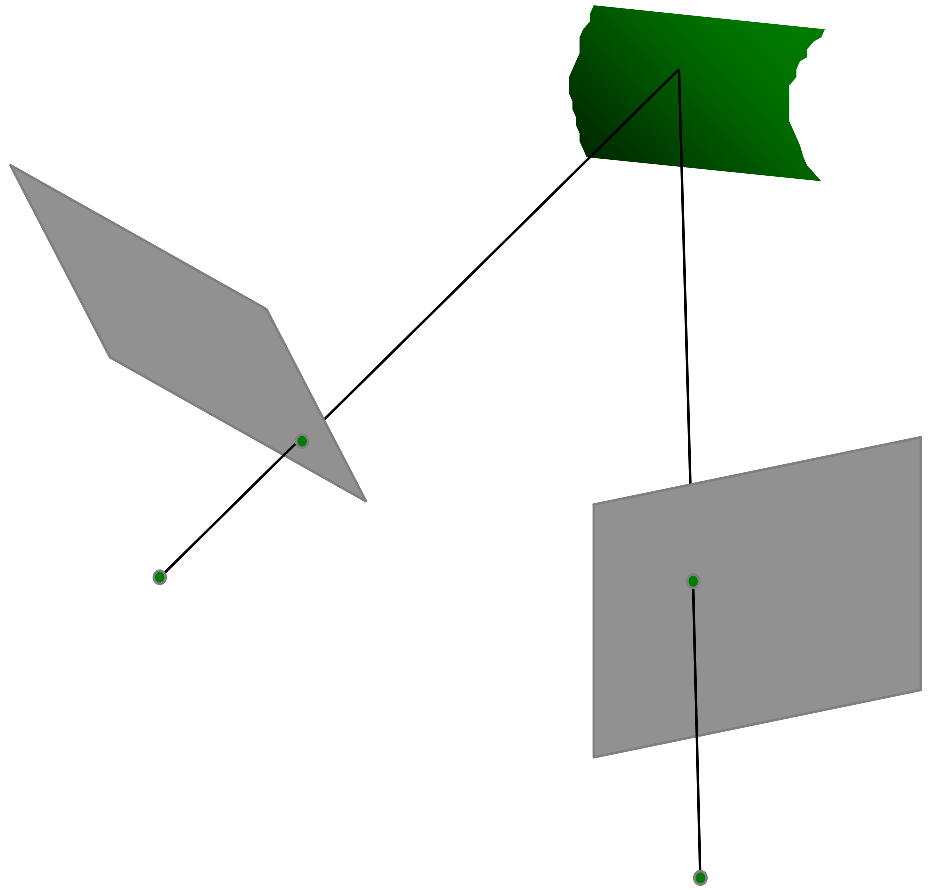




Use stereo rectification?

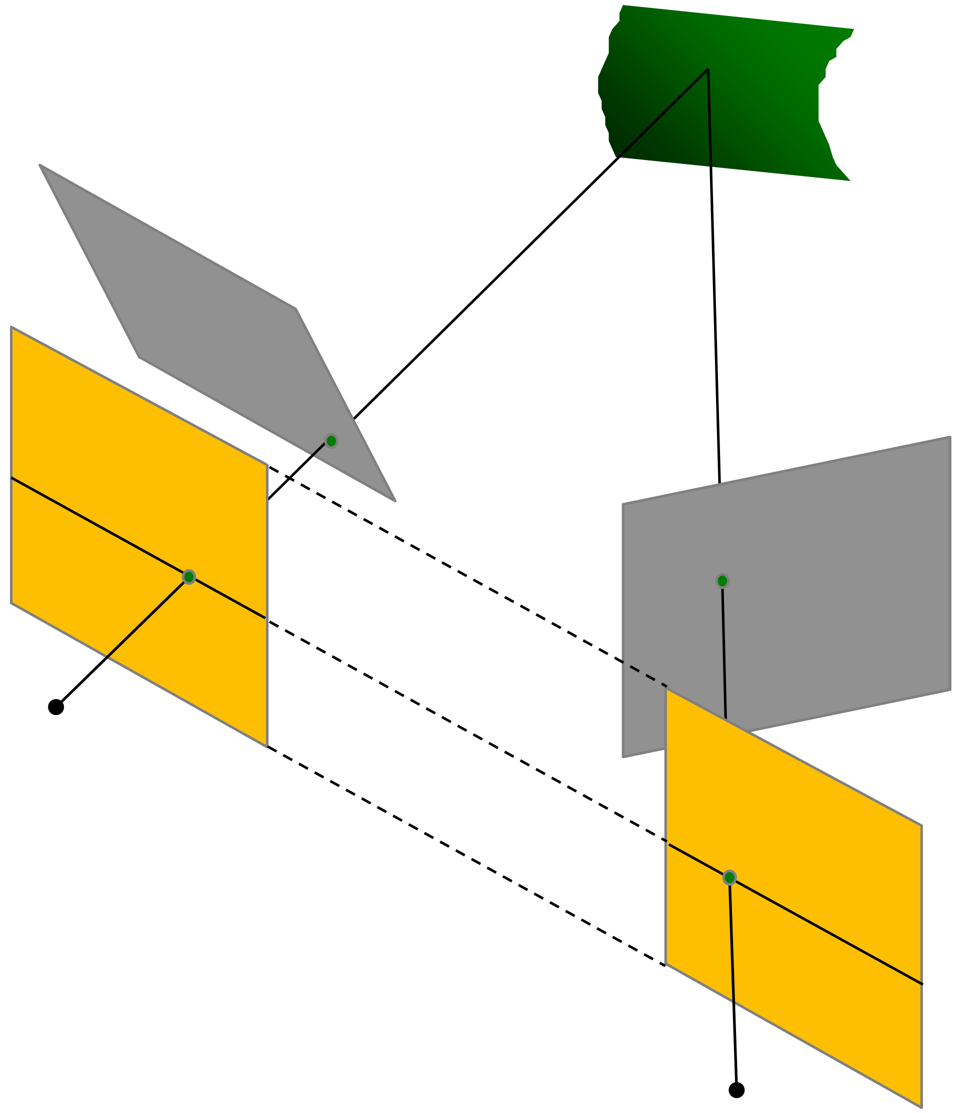


What is stereo rectification?



What is stereo rectification?

Reproject image planes onto a common plane parallel to the line between camera centers

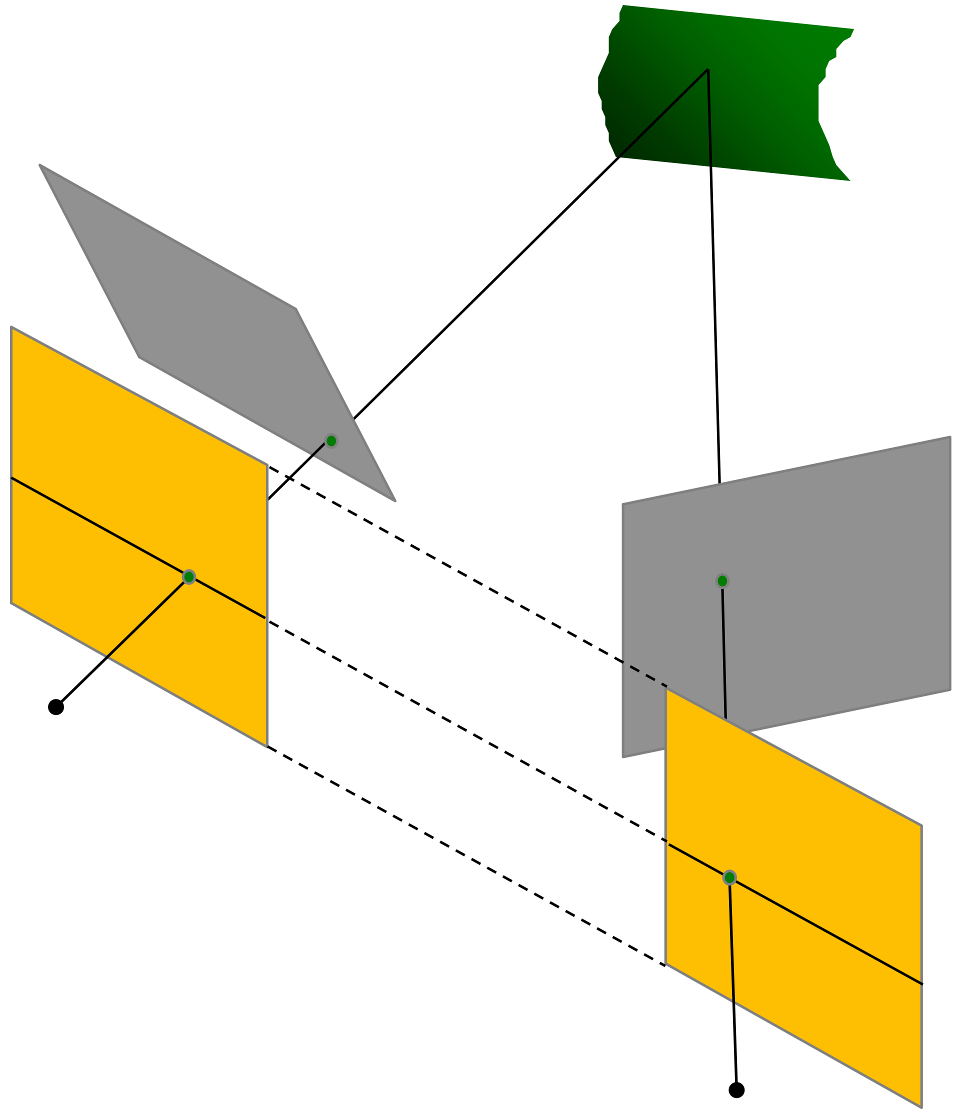


How can you do this?

What is stereo rectification?

Reproject image planes onto a common plane parallel to the line between camera centers

Need two homographies (3x3 transform), one for each input image reprojection



C. Loop and Z. Zhang. Computing Rectifying Homographies for Stereo Vision.Computer Vision and Pattern Recognition, 1999.

Stereo Rectification

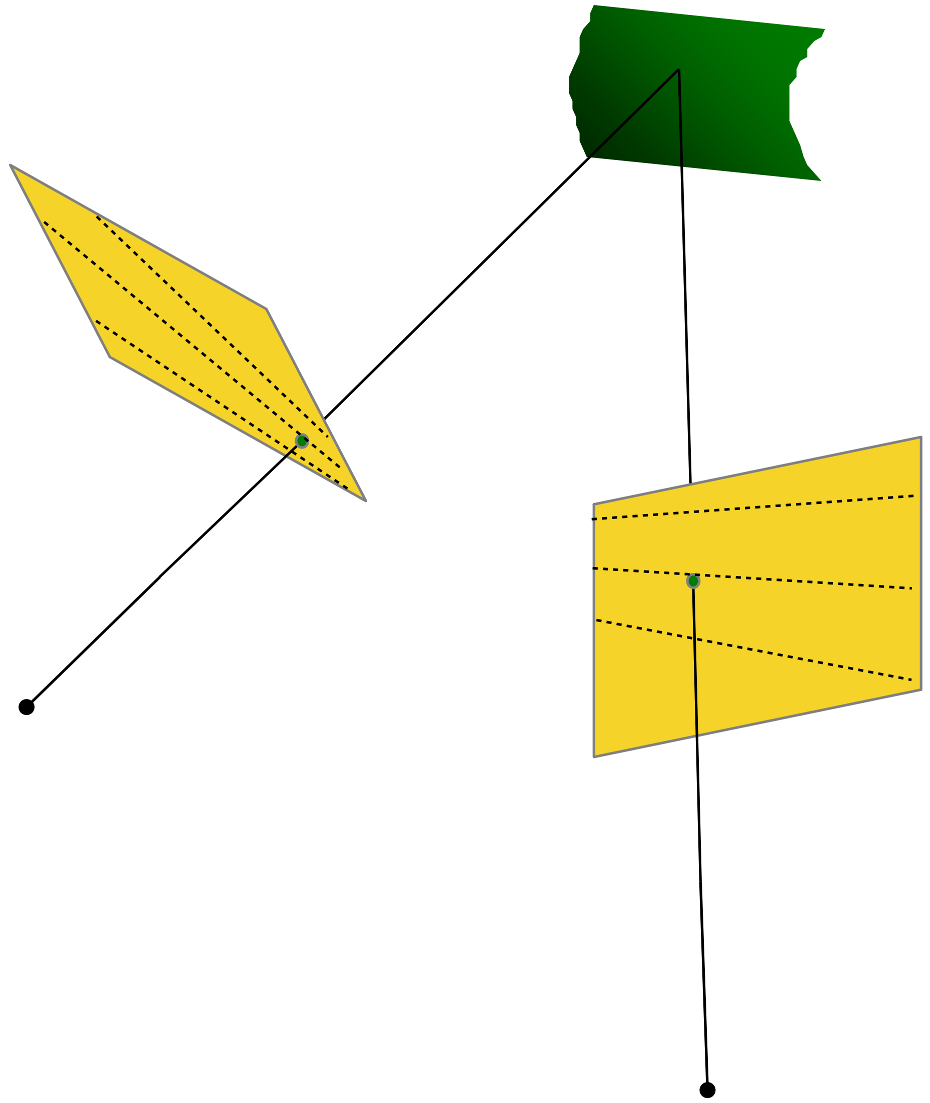
1. **Rotate** the right camera by **R**
   (aligns camera coordinate system orientation only)

2. Rotate (**rectify**) the left camera so that the epipole
   is at infinity

3. Rotate (**rectify**) the right camera so that the epipole
   is at infinity

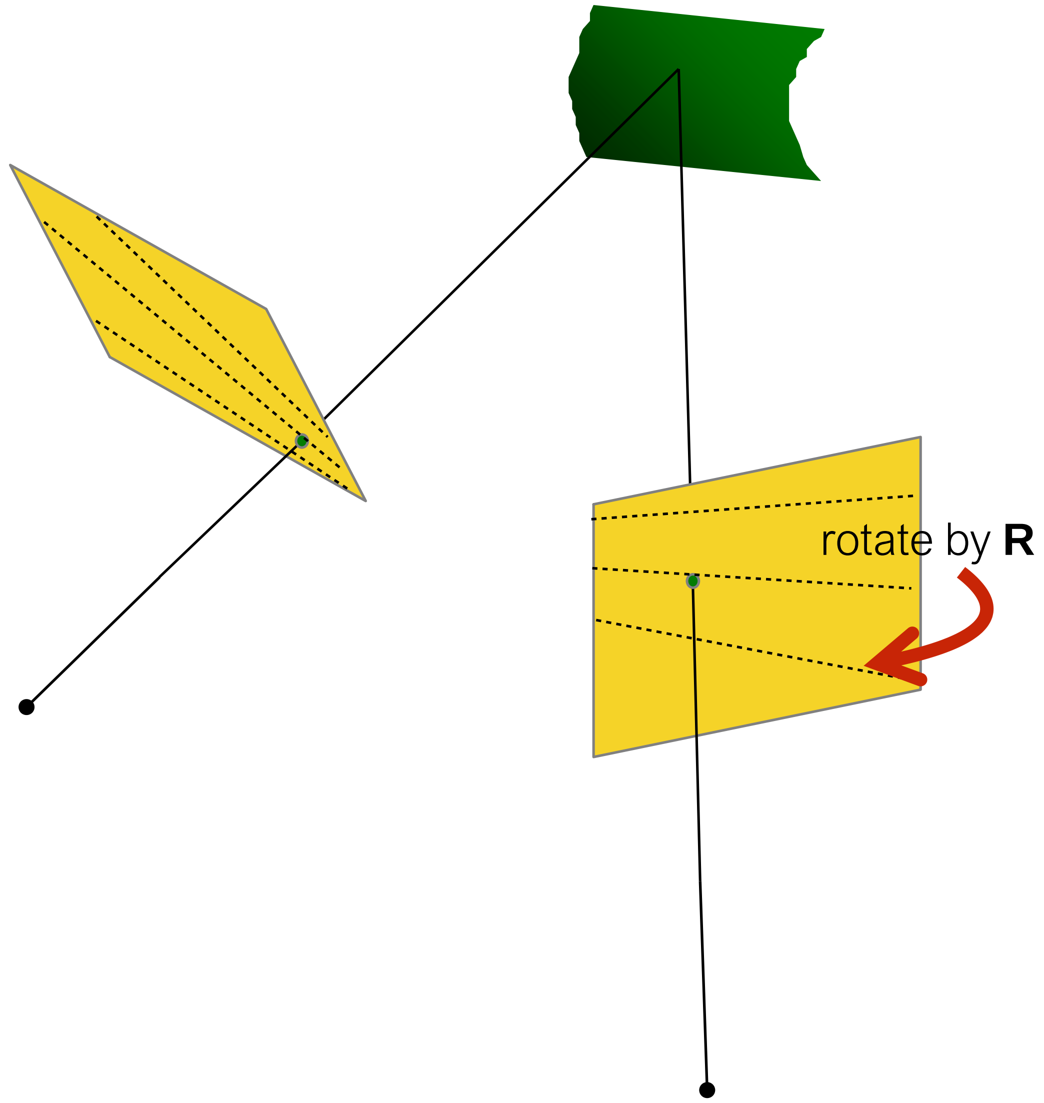
4. Adjust the **scale**

Stereo Rectification:



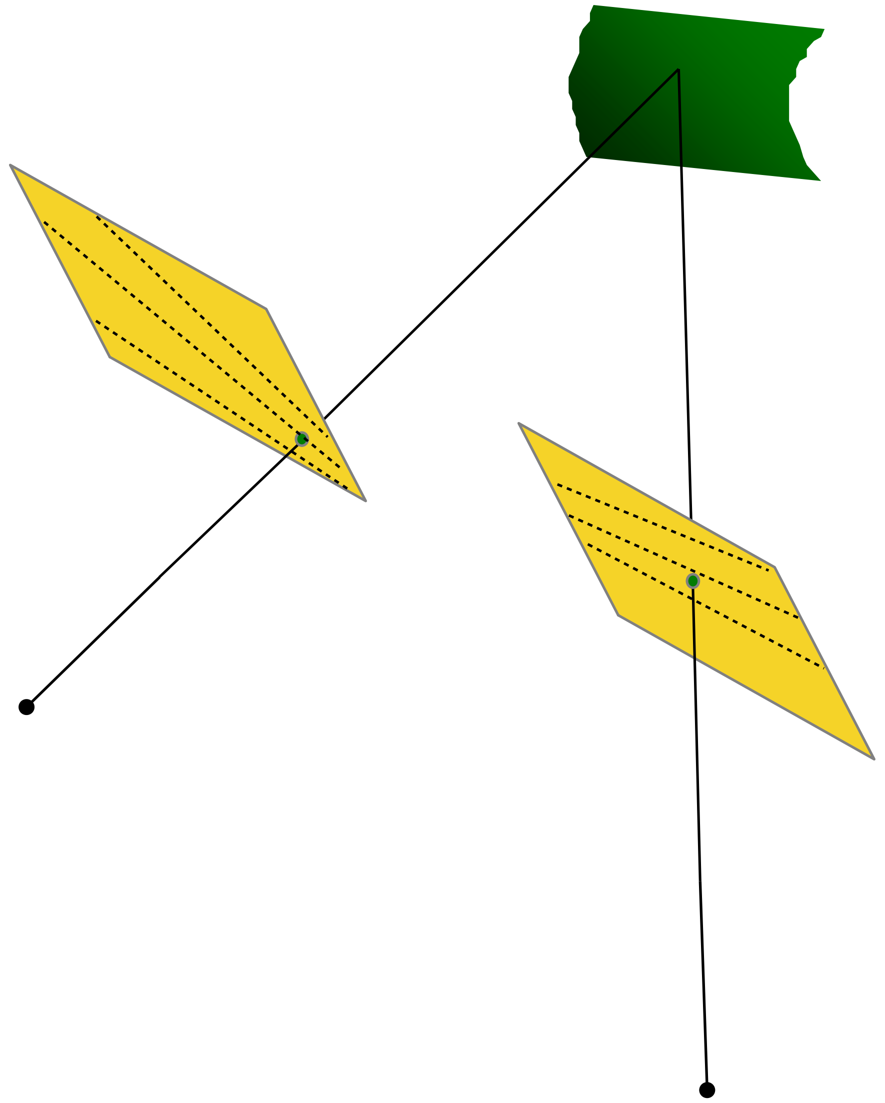
1. Compute **E** to get **R**
2. Rotate right image by **R**
3. Rotate both images by $\mathbf{R}_{\mathbf{rect}}$
4. Scale both images by **H**

Stereo Rectification:



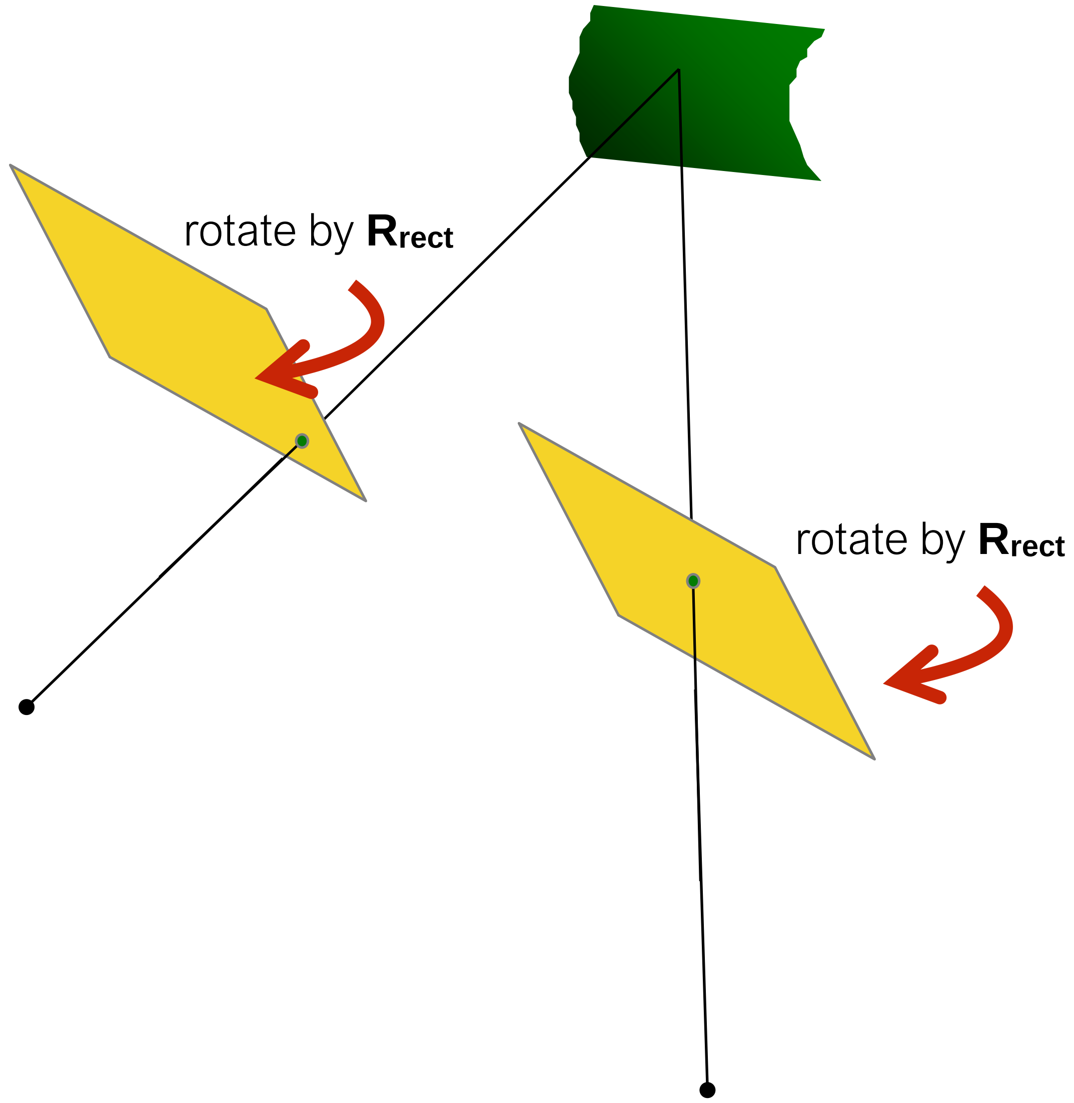
1. Compute **E** to get **R**
2. Rotate right image by **R**
3. Rotate both images by $\mathbf{R}_{\mathbf{rect}}$
4. Scale both images by **H**

Stereo Rectification:



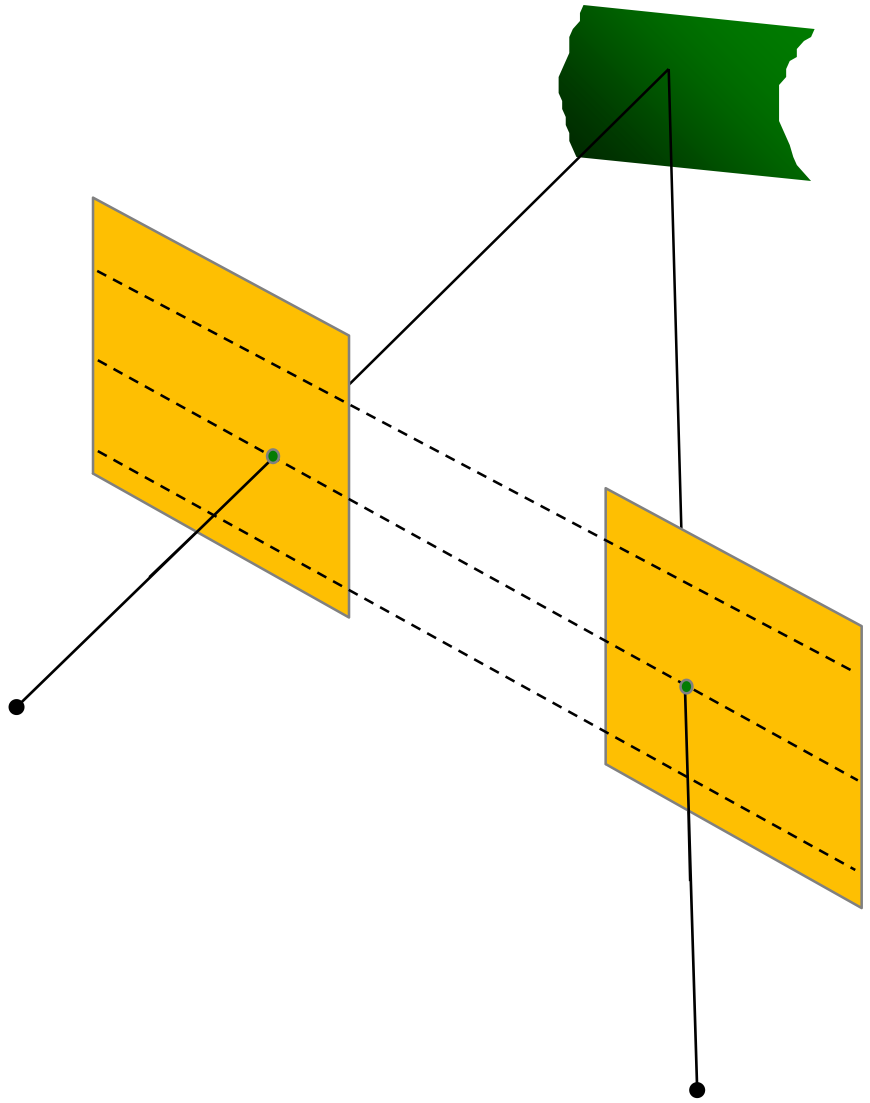
1. Compute **E** to get **R**
2. Rotate right image by **R**
3. Rotate both images by $\mathbf{R}_{\mathbf{rect}}$
4. Scale both images by **H**

Stereo Rectification:



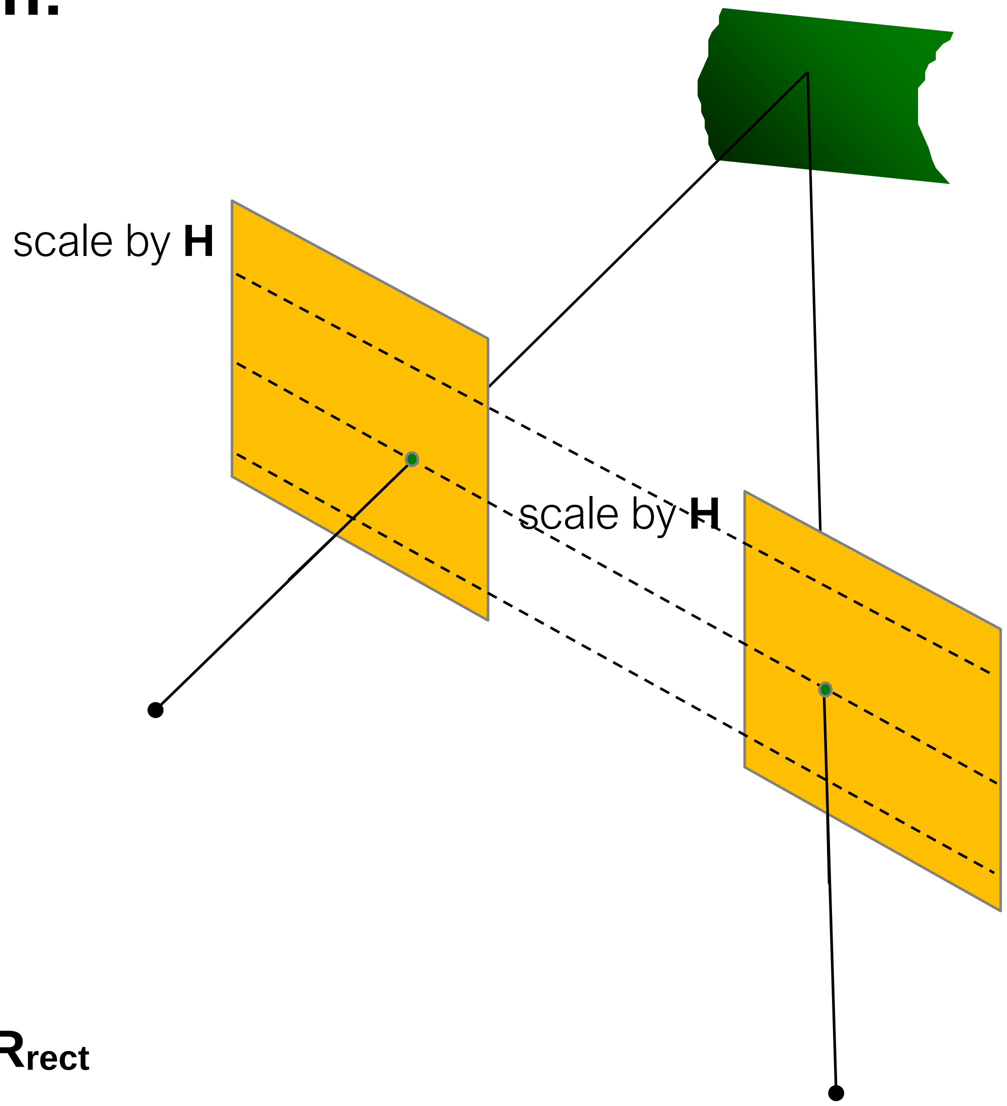
1. Compute **E** to get **R**
2. Rotate right image by **R**
3. Rotate both images by $\mathbf{R}_{\mathbf{rect}}$
4. Scale both images by **H**

Stereo Rectification:



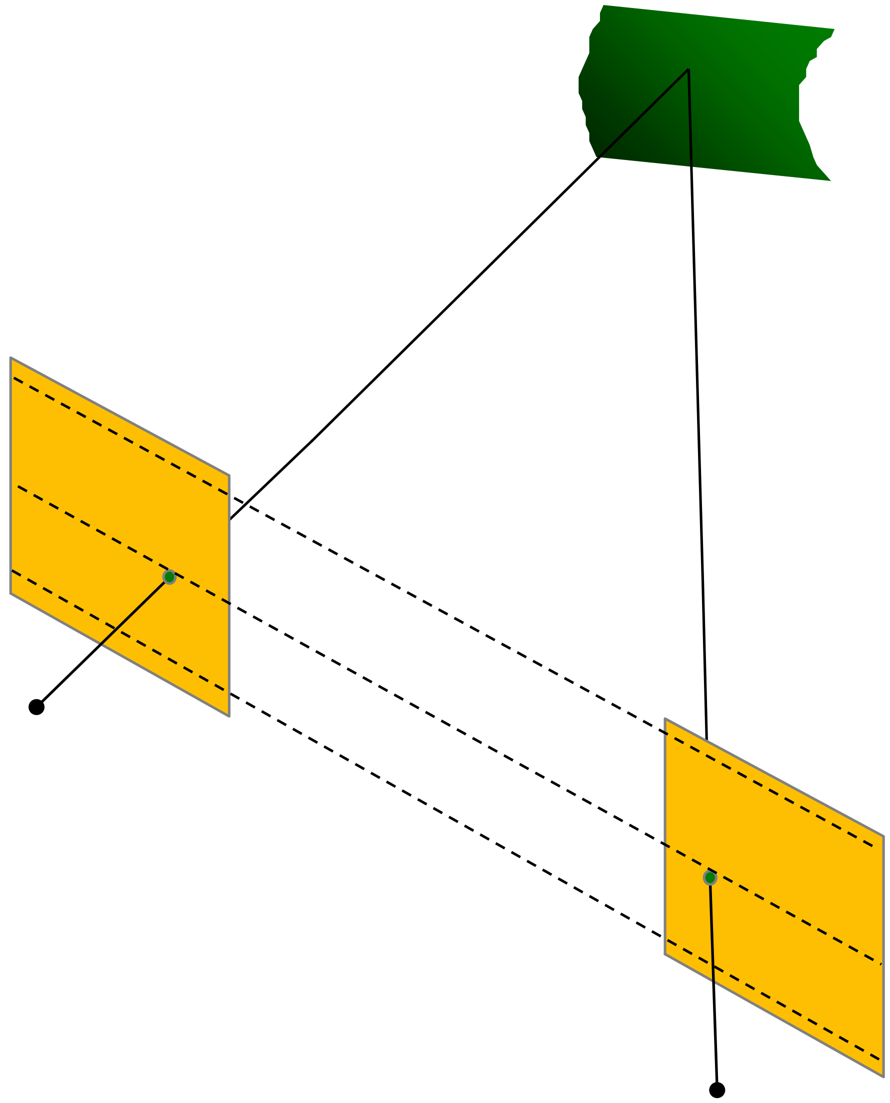
1. Compute **E** to get **R**
2. Rotate right image by **R**
3. Rotate both images by $\mathbf{R}_{\mathbf{rect}}$
4. Scale both images by **H**

Stereo Rectification:



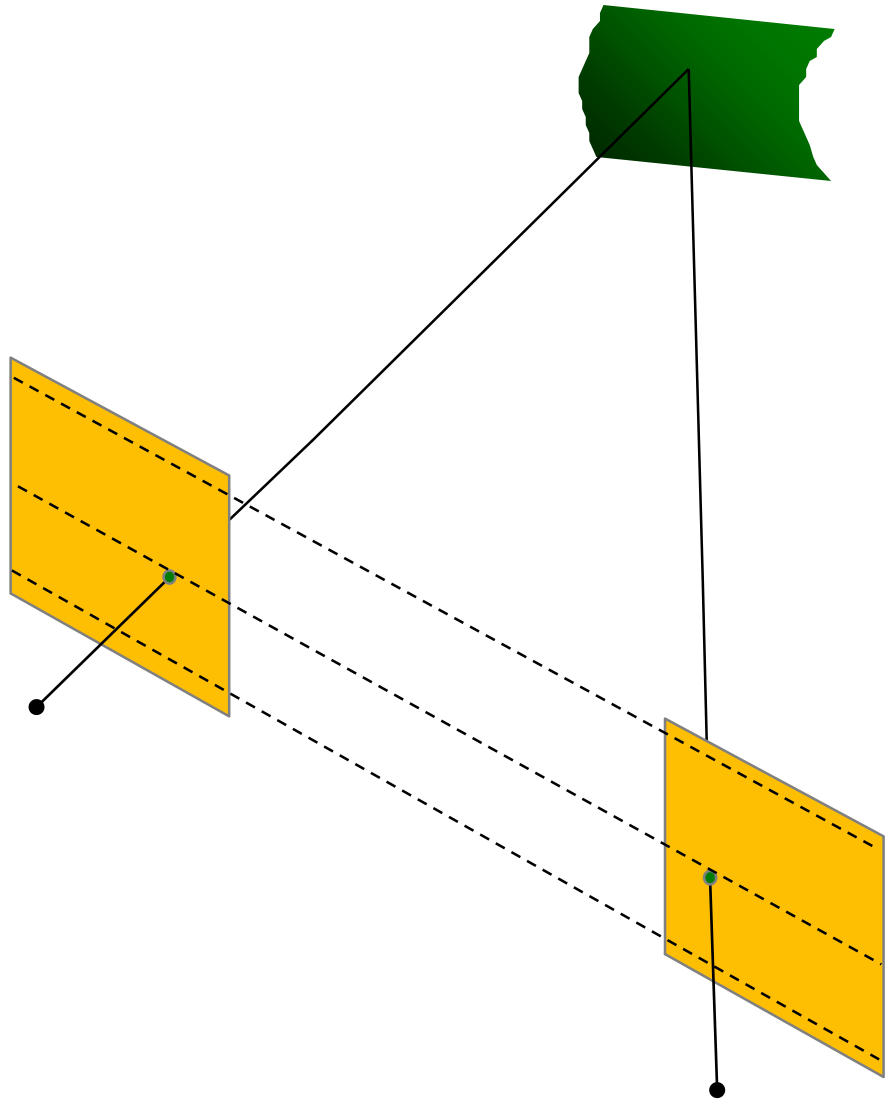
1. Compute **E** to get **R**
2. Rotate right image by **R**
3. Rotate both images by $\mathbf{R}_{\mathbf{rect}}$
4. Scale both images by **H**

Stereo Rectification:



1. Compute **E** to get **R**
2. Rotate right image by **R**
3. Rotate both images by $\mathbf{R_{rect}}$
4. Scale both images by **H**

Stereo Rectification:



1. Compute **E** to get **R**
2. Rotate right image by **R**
3. Rotate both images by $\mathbf{R}_{\mathbf{rect}}$
4. Scale both images by **H**

Step 1: Compute **E** to get **R**

SVD: $\mathbf{E} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$ Let $\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

We get FOUR solutions:

$$\mathbf{E} = [\mathbf{R}|\mathbf{T}]$$

$\mathbf{R}_1 = \mathbf{U}\mathbf{W}\mathbf{V}^\top$ $\mathbf{R}_2 = \mathbf{U}\mathbf{W}^\top\mathbf{V}^\top$ $\mathbf{T}_1 = U_3$ $\mathbf{T}_2 = -U_3$

two possible rotations

two possible translations

We get FOUR solutions:

$$\mathbf{R}_1 = \mathbf{U}\mathbf{W}\mathbf{V}^\top$$
$$\mathbf{T}_1 = U_3$$

$$\mathbf{R}_1 = \mathbf{U}\mathbf{W}\mathbf{V}^\top$$
$$\mathbf{T}_2 = -U_3$$

$$\mathbf{R}_2 = \mathbf{U}\mathbf{W}^\top\mathbf{V}^\top$$
$$\mathbf{T}_2 = -U_3$$

$$\mathbf{R}_2 = \mathbf{U}\mathbf{W}^\top\mathbf{V}^\top$$
$$\mathbf{T}_1 = U_3$$

Which one do we choose?

Compute determinant of R, valid solution must be equal to 1

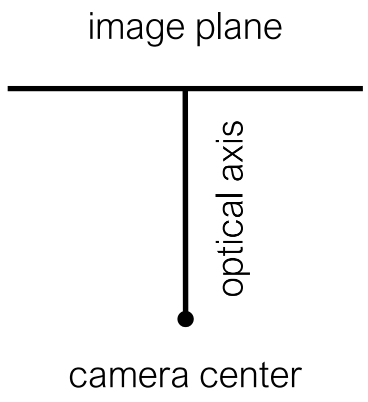
(note: det(R) = -1 means rotation and reflection)

Compute 3D point using triangulation, valid solution has positive Z value

(Note: negative Z means point is behind the camera)

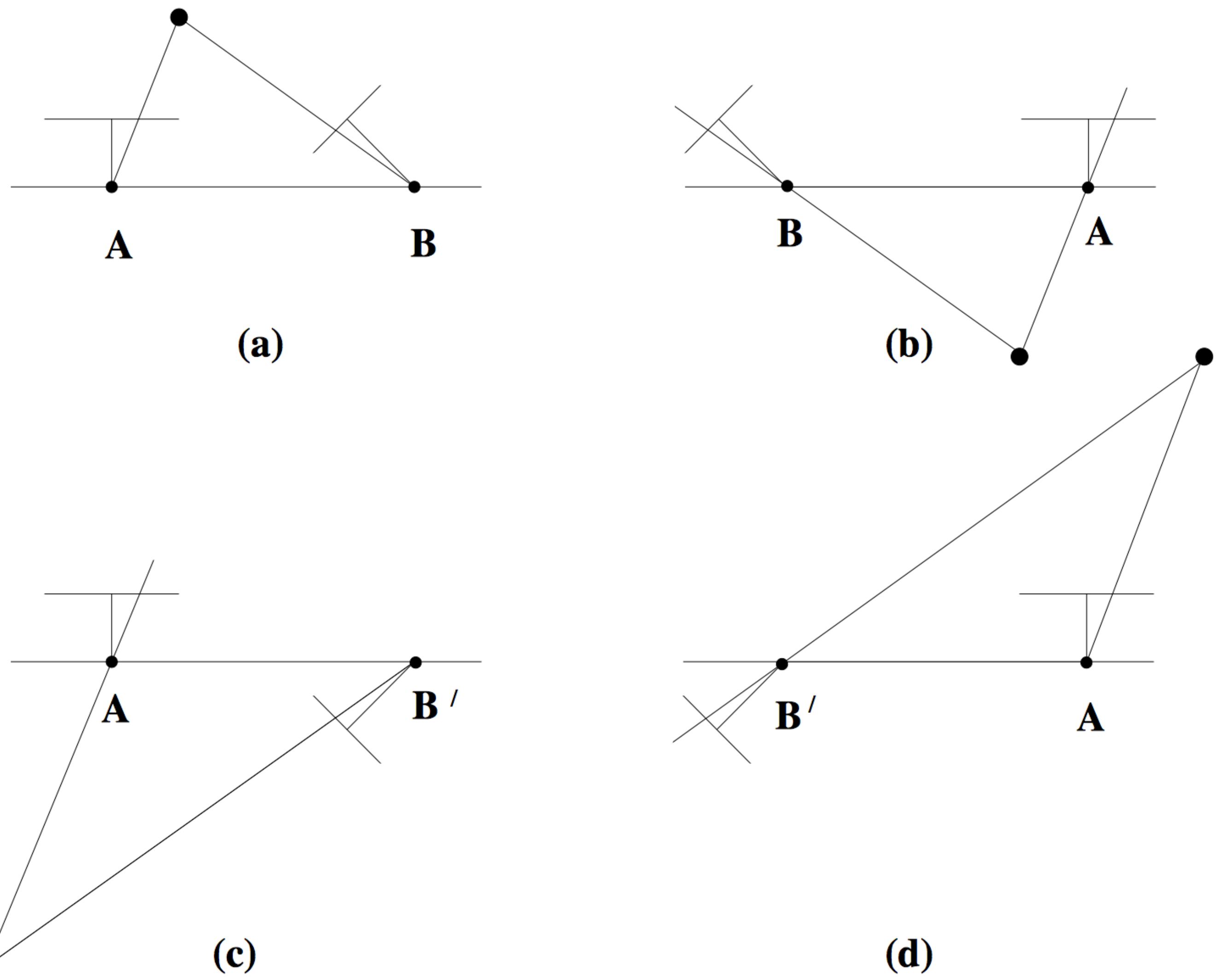
Let’s visualize the four configurations…

Camera Icon

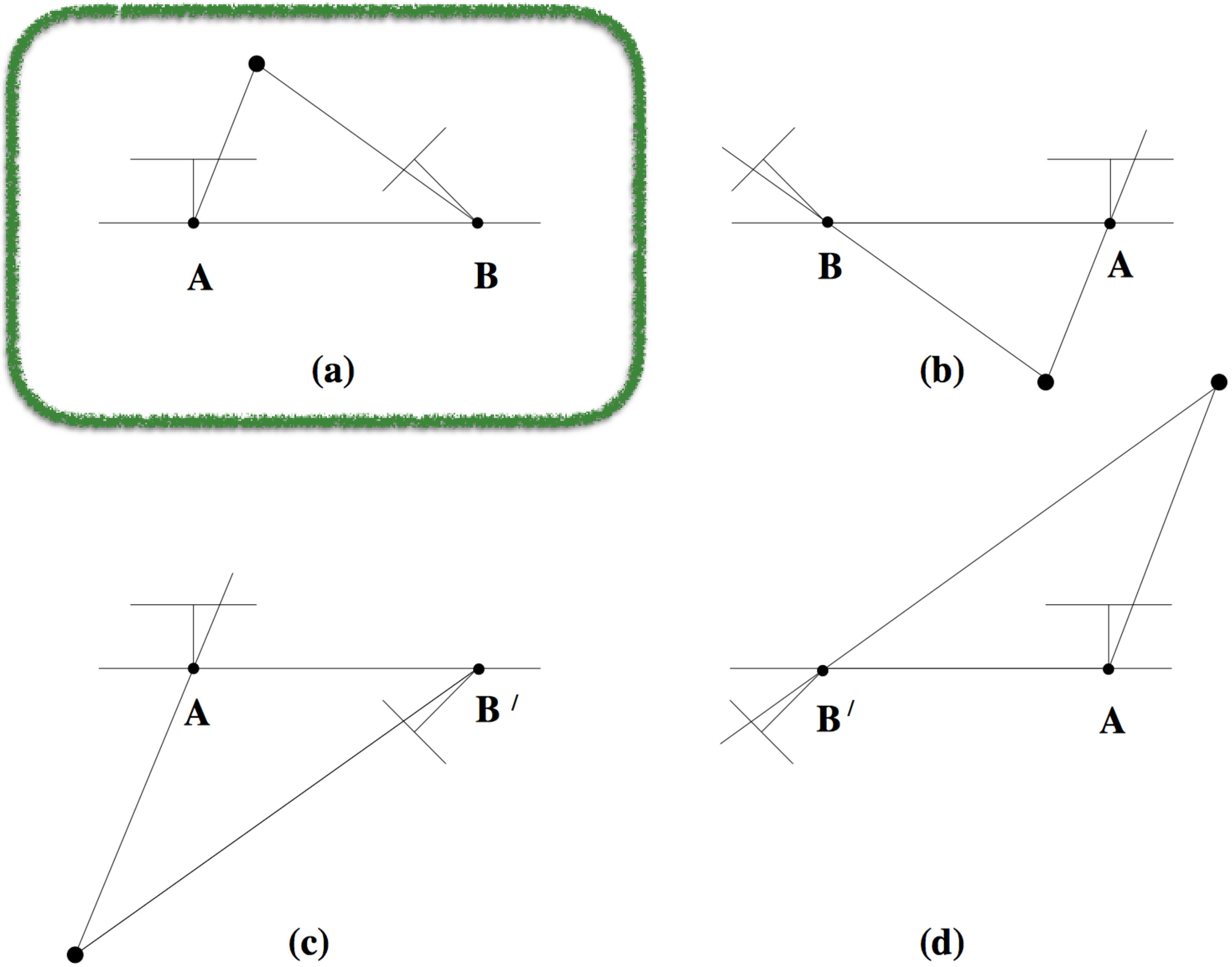


Find the configuration where the point is in front of both cameras

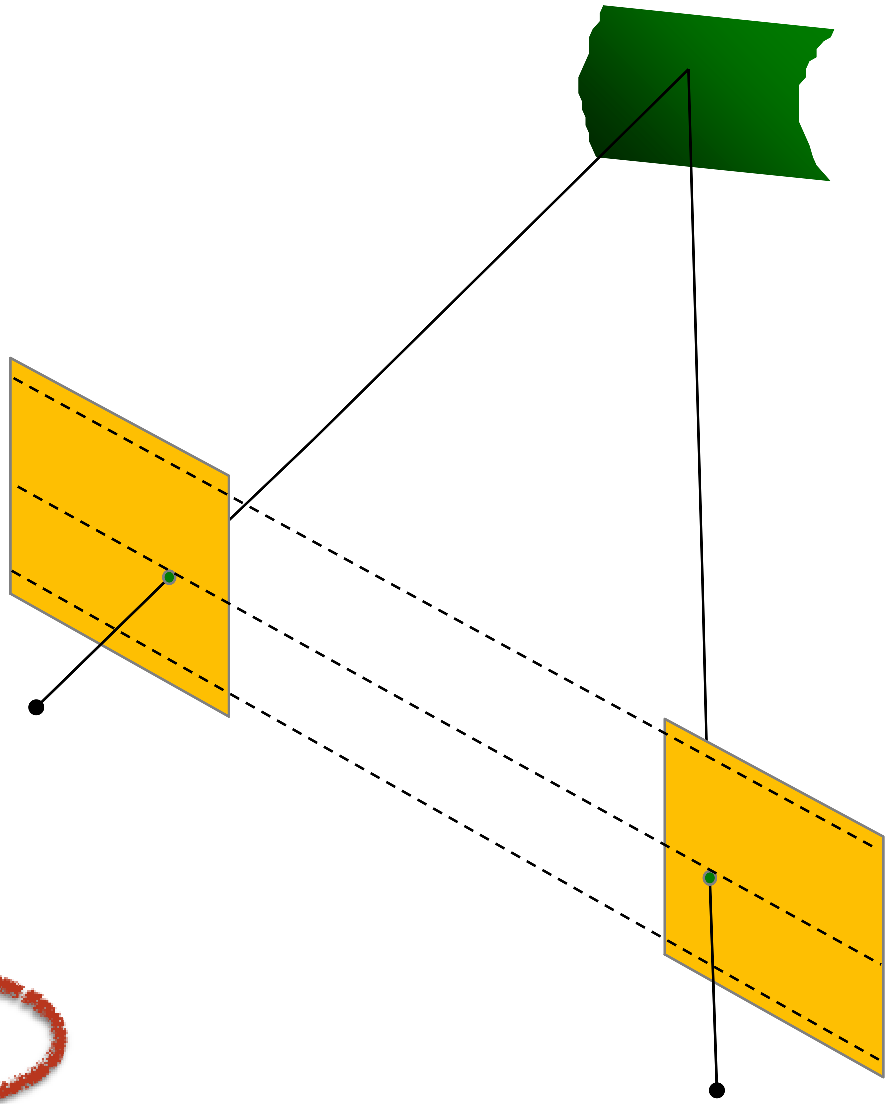
Find the configuration where the points is in front of both cameras



Find the configuration where the points is in front of both cameras



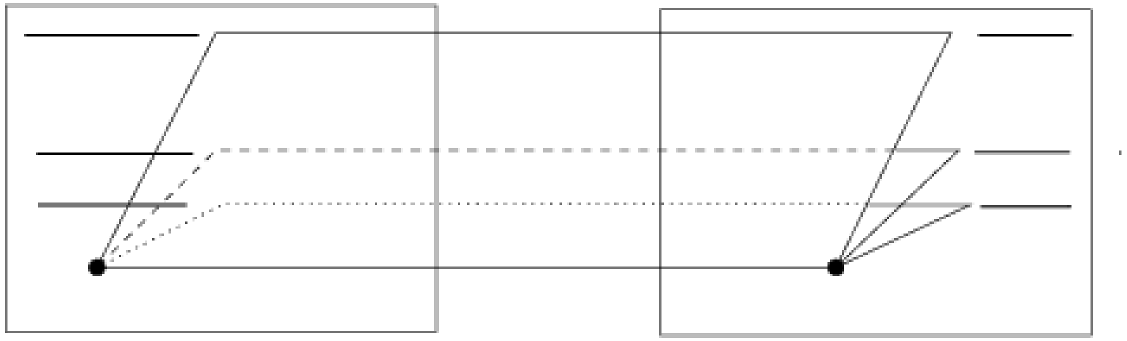
Stereo Rectification:

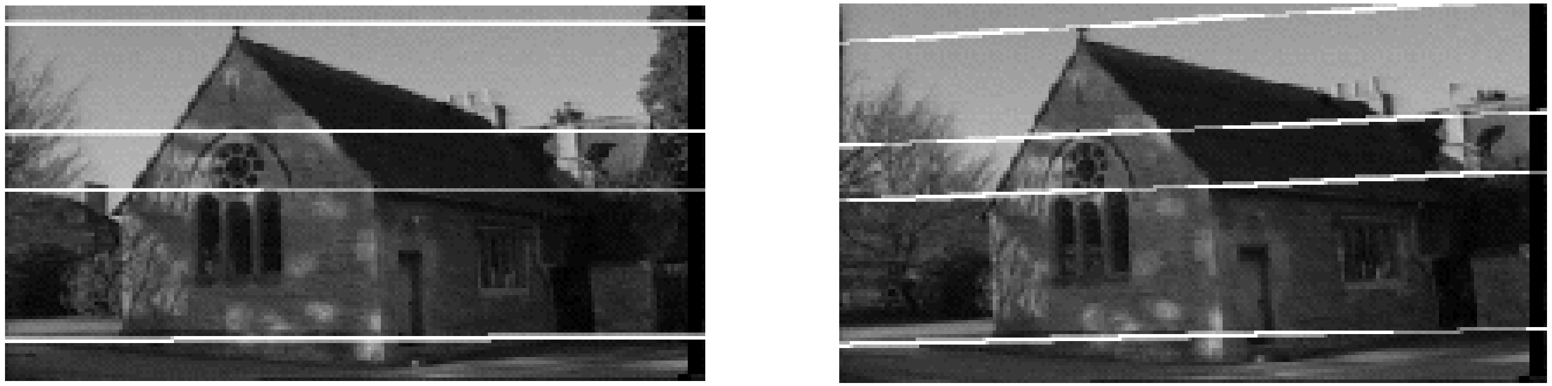


1. Compute **E** to get **R**
2. Rotate right image by **R**
3. Rotate both images by $\mathbf{R_{rect}}$
4. Scale both images by **H**

When do epipolar lines become horizontal?

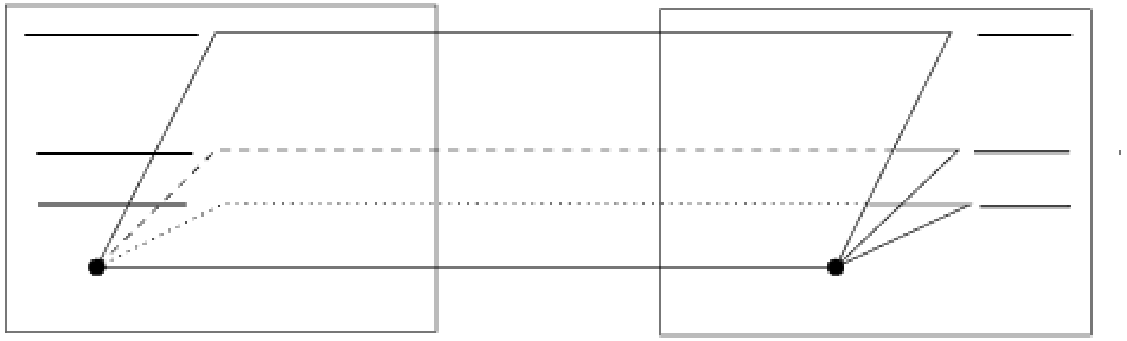
Parallel cameras

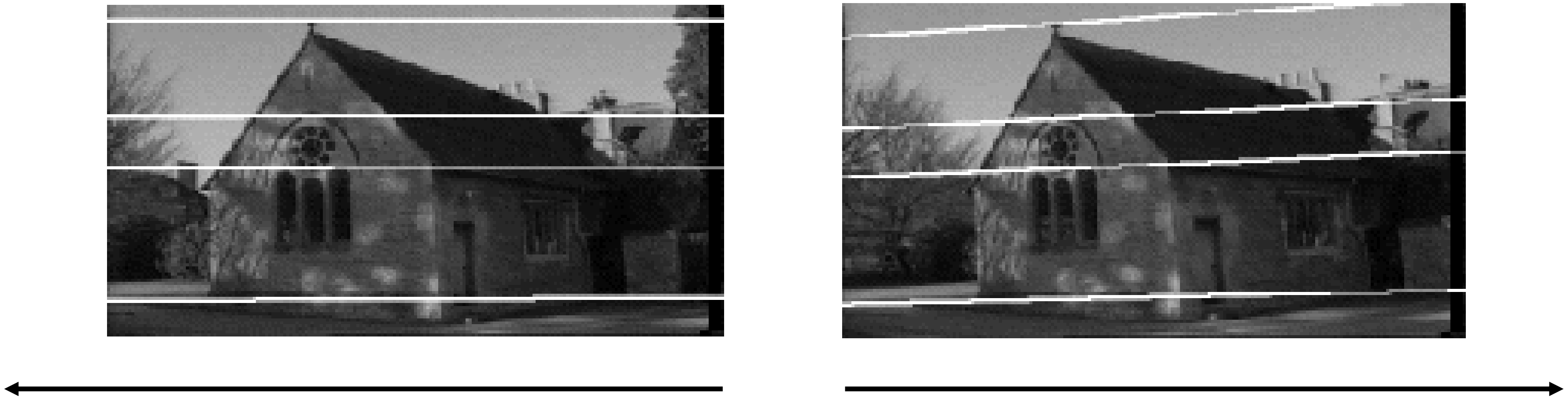




Where is the epipole?

Parallel cameras





epipole at infinity

Setting the epipole to infinity

(Building $\mathbf{R}_{\text{rect}}$ from **e**)

Let $$R_{\text{rect}} = \begin{bmatrix} \boldsymbol{r}_1^\top \\ \boldsymbol{r}_2^\top \\ \boldsymbol{r}_3^\top \end{bmatrix}$$

Given: epipole **e**
(using SVD on E)
(translation from **E)**

$$\boldsymbol{r}_1 = \boldsymbol{e}_1 = \frac{T}{||T||}$$

epipole coincides with translation vector

$$\boldsymbol{r}_2 = \frac{1}{\sqrt{T_x^2 + T_y^2}} \begin{bmatrix} -T_y & T_x & 0 \end{bmatrix}$$

cross product of e and the direction vector of the optical axis

$$\boldsymbol{r}_3 = \boldsymbol{r}_1 \times \boldsymbol{r}_2$$

orthogonal vector

If $\quad \boldsymbol{r}_1 = \boldsymbol{e}_1 = \frac{T}{||T||} \quad$ and $\quad \boldsymbol{r}_2 \quad \boldsymbol{r}_3 \quad$ orthogonal

then $$R_{\text{rect}}\boldsymbol{e}_1 = \begin{bmatrix} \boldsymbol{r}_1^\top \boldsymbol{e}_1 \\ \boldsymbol{r}_2^\top \boldsymbol{e}_1 \\ \boldsymbol{r}_3^\top \boldsymbol{e}_1 \end{bmatrix} = \begin{bmatrix} ? \\ ? \\ ? \end{bmatrix}$$

If $\quad \boldsymbol{r}_1 = \boldsymbol{e}_1 = \dfrac{T}{||T||} \quad$ and $\quad \boldsymbol{r}_2 \quad \boldsymbol{r}_3 \quad$ orthogonal

then $\quad R_{\text{rect}}\boldsymbol{e}_1 = \begin{bmatrix} \boldsymbol{r}_1^\top \boldsymbol{e}_1 \\ \boldsymbol{r}_2^\top \boldsymbol{e}_1 \\ \boldsymbol{r}_3^\top \boldsymbol{e}_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

Where is this point located on the image plane?

If $\quad \boldsymbol{r}_1 = \boldsymbol{e}_1 = \frac{T}{||T||} \quad$ and $\quad \boldsymbol{r}_2 \quad \boldsymbol{r}_3 \quad$ orthogonal

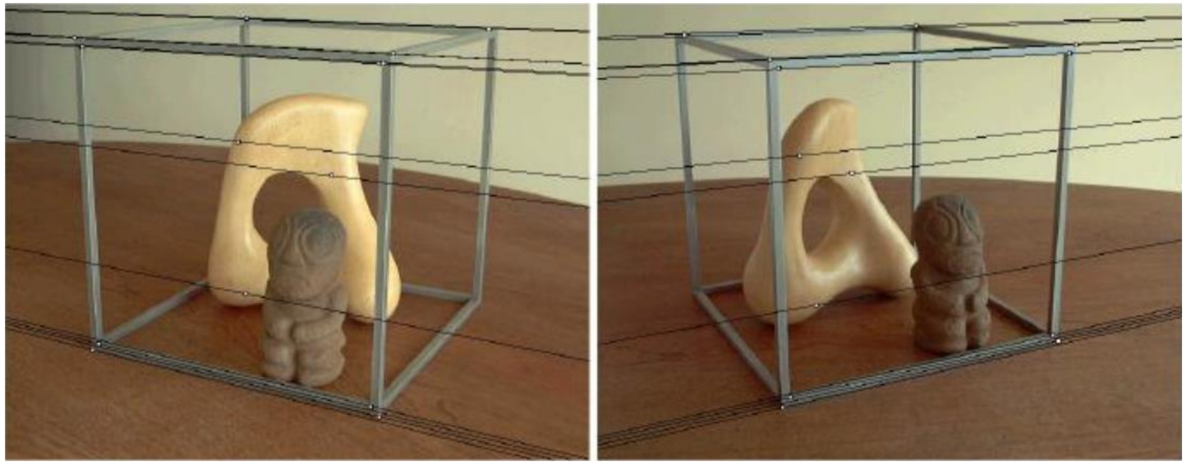
then $\quad R_{\text{rect}} \boldsymbol{e}_1 = \begin{bmatrix} \boldsymbol{r}_1^\top \boldsymbol{e}_1 \\ \boldsymbol{r}_2^\top \boldsymbol{e}_1 \\ \boldsymbol{r}_3^\top \boldsymbol{e}_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

Where is this point located on the image plane?

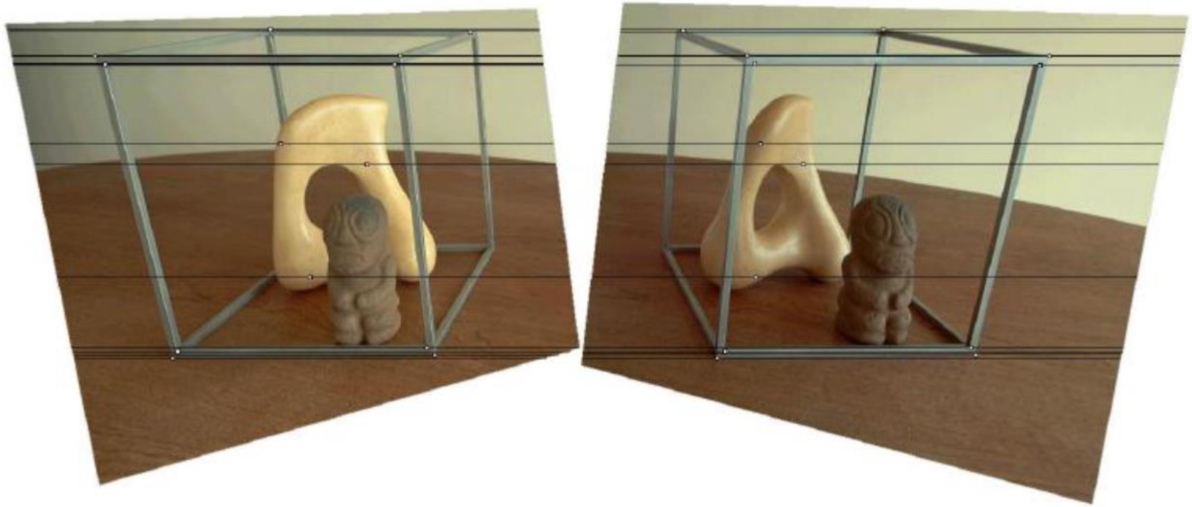
At x-infinity

Stereo Rectification Algorithm

1. Estimate **E** using the 8 point algorithm (SVD)
2. Estimate the epipole **e** (SVD of **E**)
3. Build $\mathbf{R}_{rect}$ from **e**
4. Decompose **E** into **R** and **T**
5. Set $\mathbf{R}_1 = \mathbf{R}_{rect}$ and $\mathbf{R}_2 = \mathbf{R}\mathbf{R}_{rect}$
6. Rotate each left camera point (warp image)
   $[x' \ y' \ z'] = \mathbf{R}_1 [x \ y \ z]$
7. Rectified points as $\mathbf{p} = f/z'[x' \ y' \ z']$
8. Repeat 6 and 7 for right camera points using $\mathbf{R}_2$

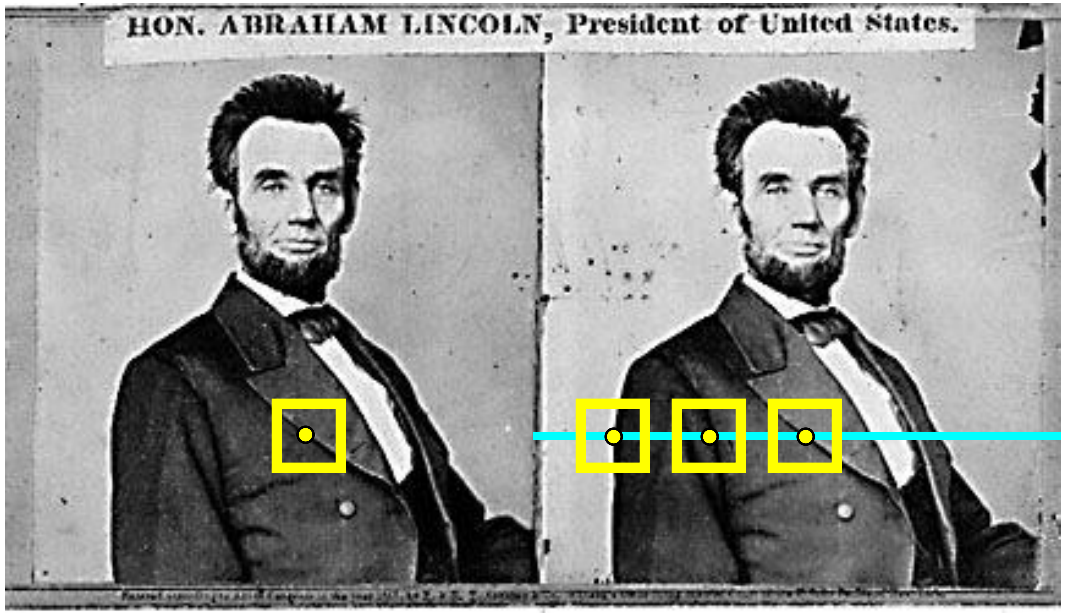


What can we do after rectification?



Stereo matching

Depth Estimation via Stereo Matching



1. Rectify images
   (make epipolar lines horizontal)
2. For each pixel
   a. Find epipolar line
   b. Scan line for best match
   c. Compute depth from disparity

$$Z = \frac{bf}{d}$$

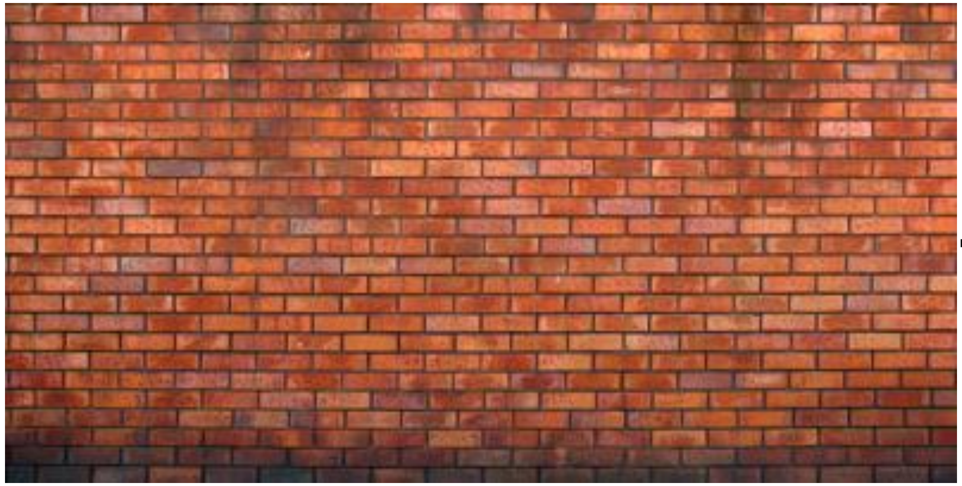
How would you do this?

Reminder from filtering

How do we detect an edge?

Reminder from filtering

How do we detect an edge?
- We filter with something that looks like an edge.

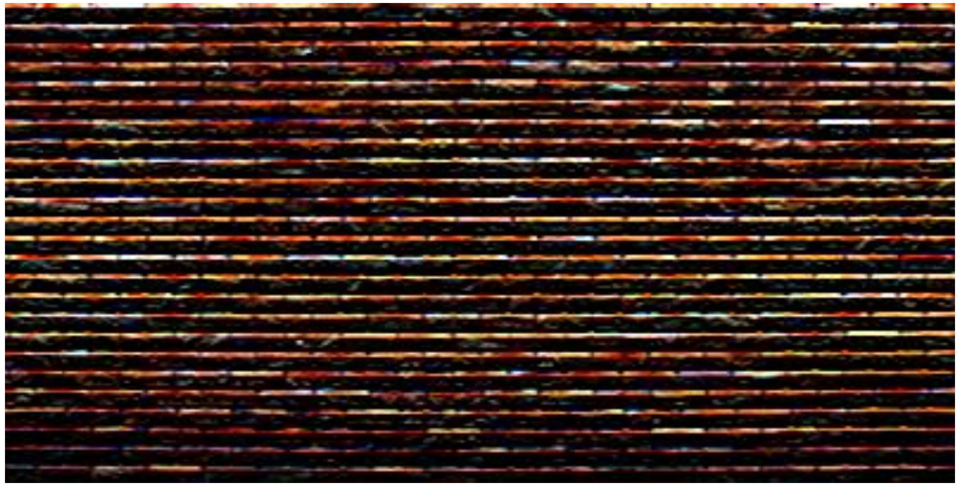


original

$$* \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

horizontal edge filter

$$* \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

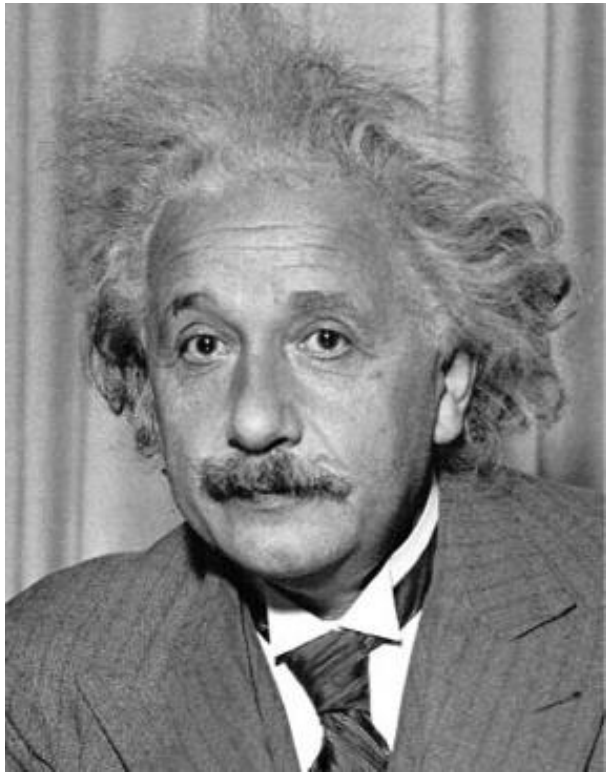


vertical edge filter

We can think of linear filtering as a way to evaluate how similar an image is *locally* to some template.

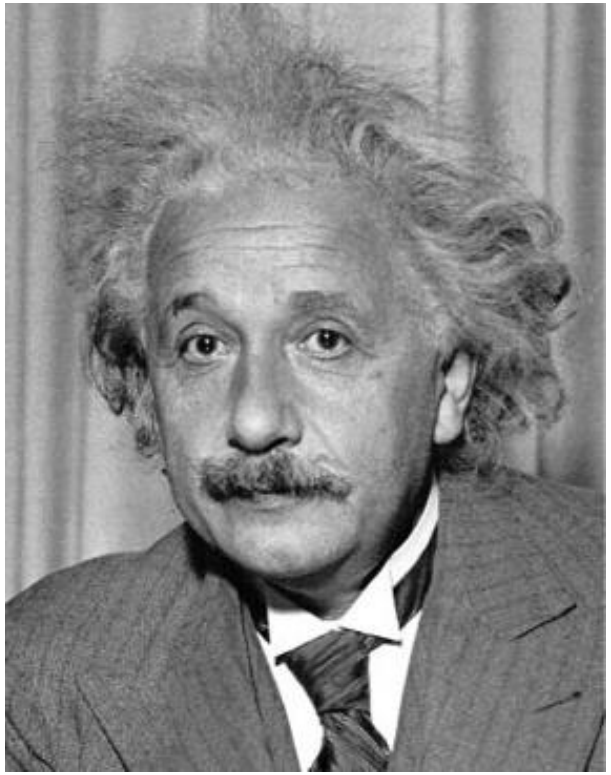
Find this template

How do we detect the template  in he following image?



Find this template

How do we detect the template  in he following image?



filter 

output

$$h[m,n] = \sum_{k,j} g[k,l] f[m+k, n+l]$$

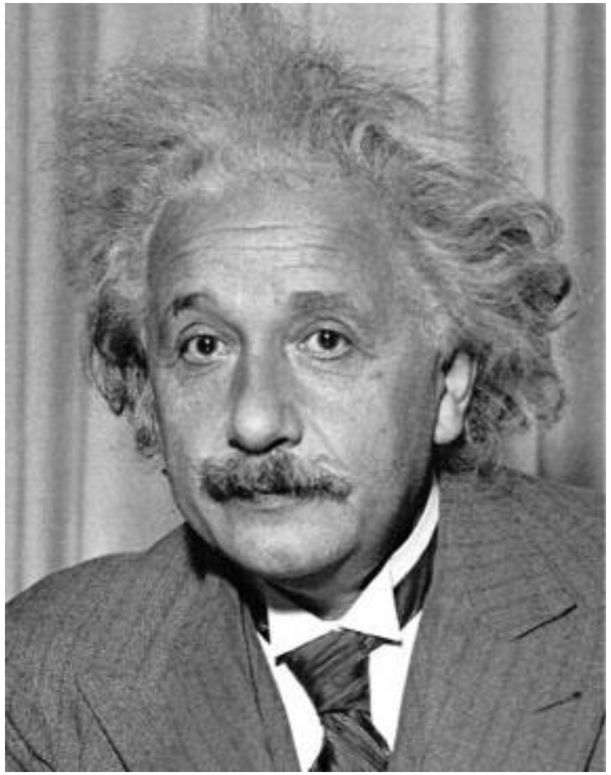
image

What will the output look like?

Solution 1: Filter the image using the template as filter kernel.

Find this template

How do we detect the template  in he following image?

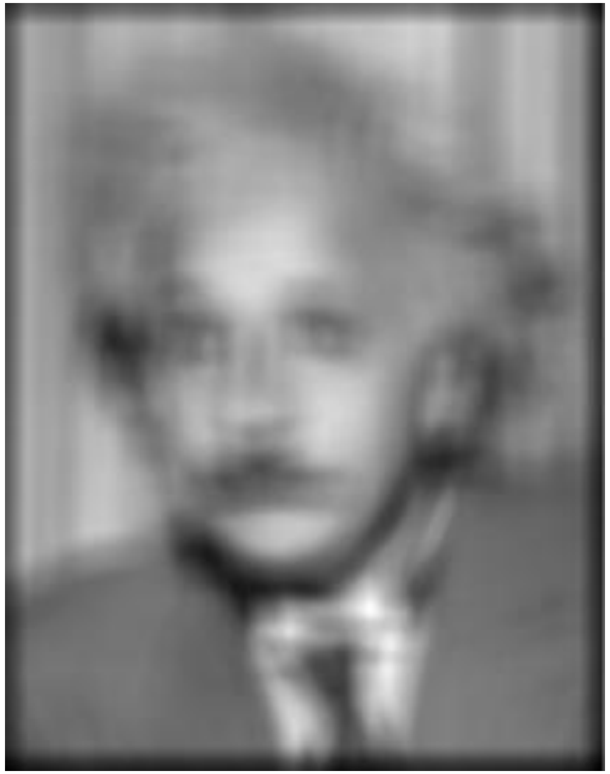


filter 

output

$$h[m,n] = \sum_{k,j} g[k,l] f[m+k, n+l]$$

image

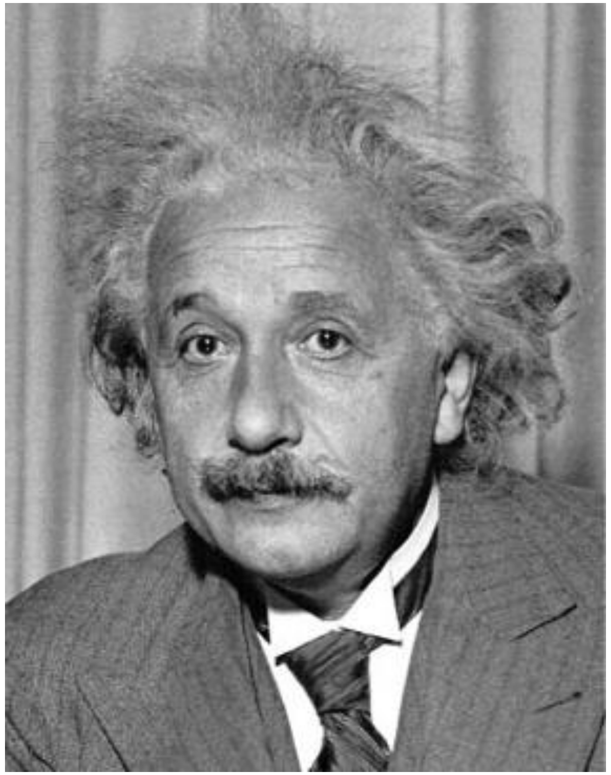


Solution 1: Filter the image using the template as filter kernel.

What went wrong?

Find this template

How do we detect the template  in he following image?

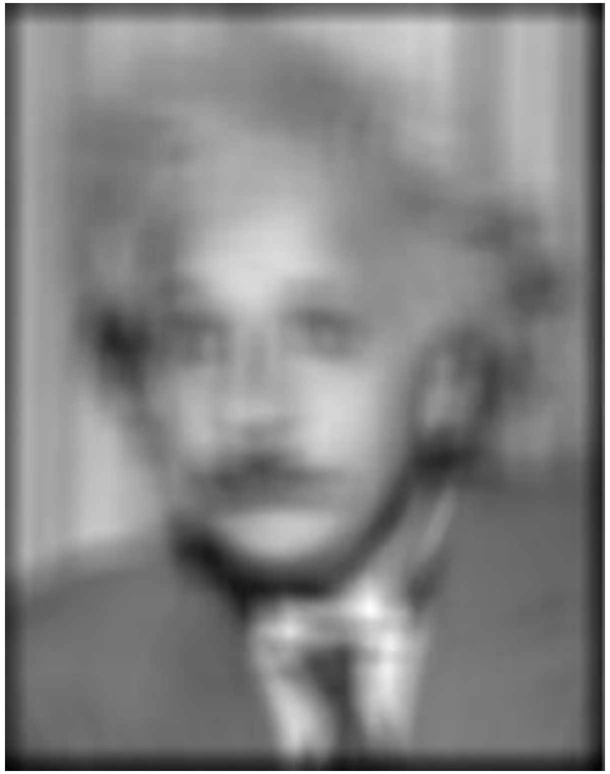


filter 

output

$$h[m,n] = \sum_{k,j} g[k,l] f[m+k, n+l]$$

image

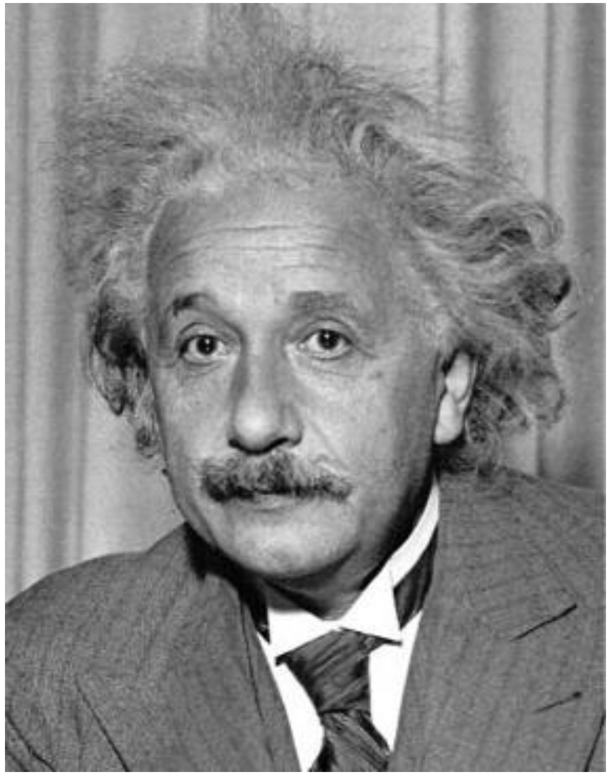


Increases for higher local intensities.

Solution 1: Filter the image using the template as filter kernel.

Find this template

How do we detect the template  in he following image?



filter 

template mean

output

$$h[m,n] = \sum_{k,l} (g[k,l] - \bar{g}) f[m+k, n+l]$$

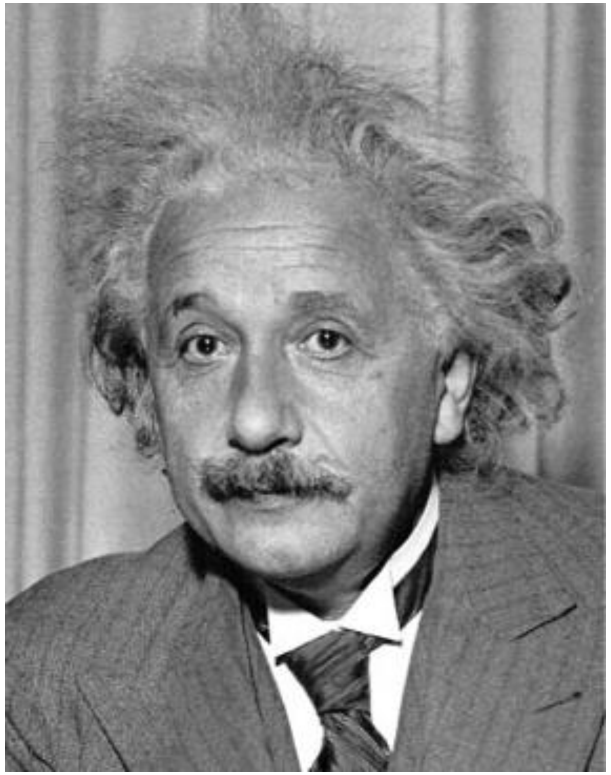
image

What will the output look like?

Solution 2: Filter the image using a *zero-mean* template.

Find this template

How do we detect the template  in he following image?



filter  template mean

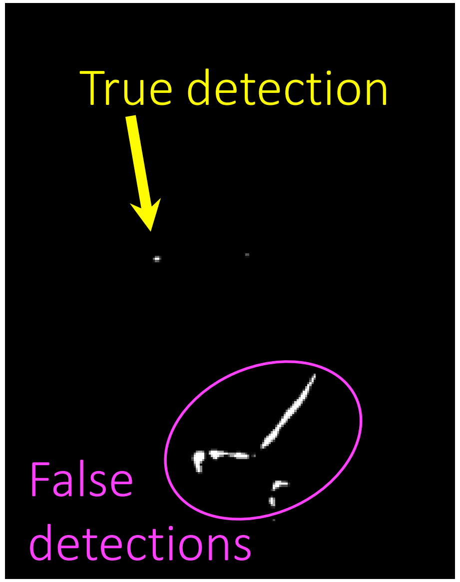
output

$$h[m,n] = \sum_{k,l} (g[k,l] - \bar{g}) f[m+k, n+l]$$

image

output

thresholding

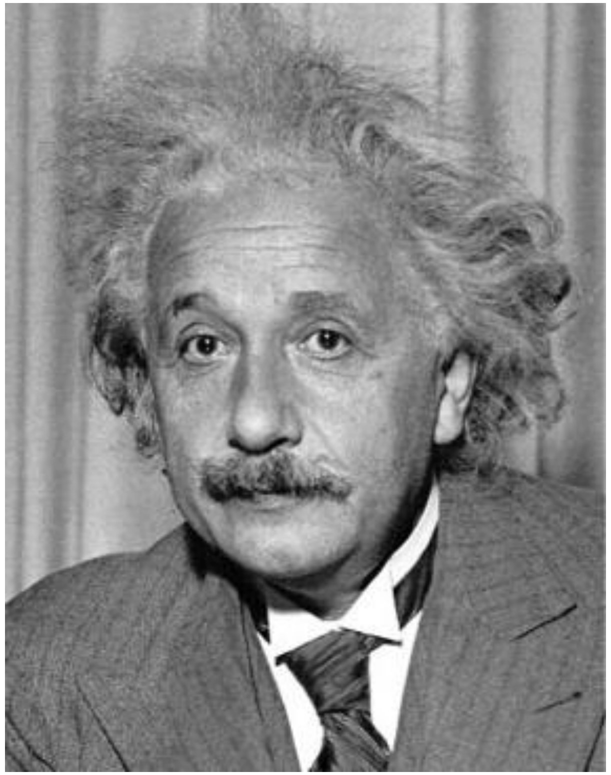


What went wrong?

Solution 2: Filter the image using a *zero-mean* template.

Find this template

How do we detect the template  in he following image?



filter 

template mean

output

$$h[m,n] = \sum_{k,l} (g[k,l] - \bar{g}) f[m+k, n+l]$$

image

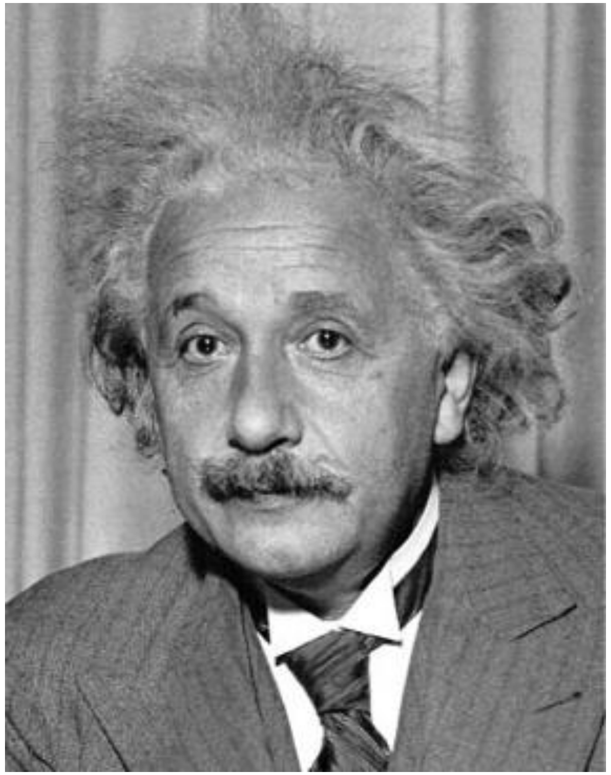
output

Not robust to high-contrast areas

Solution 2: Filter the image using a *zero-mean* template.

Find this template

How do we detect the template  in he following image?



filter 

output

$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k, n+l])^2$$

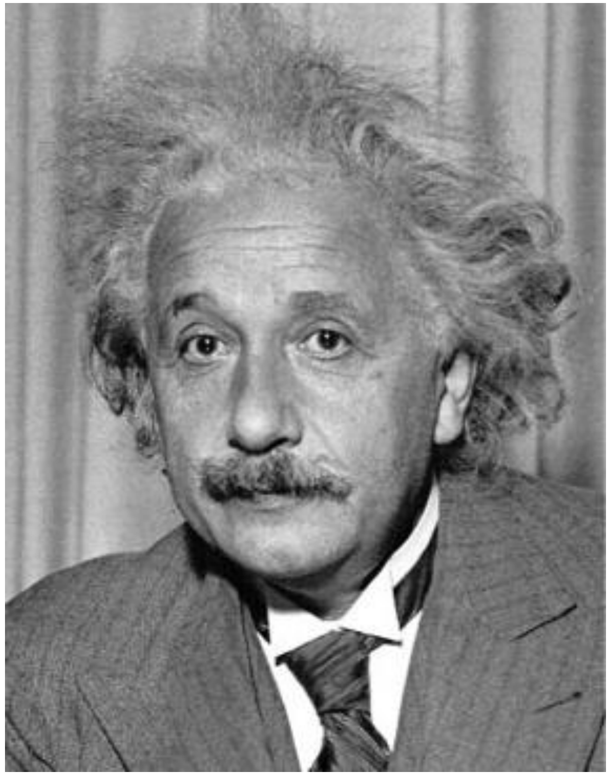
image

What will the output look like?

Solution 3: Use sum of squared differences (SSD).

Find this template

How do we detect the template  in he following image?



filter 

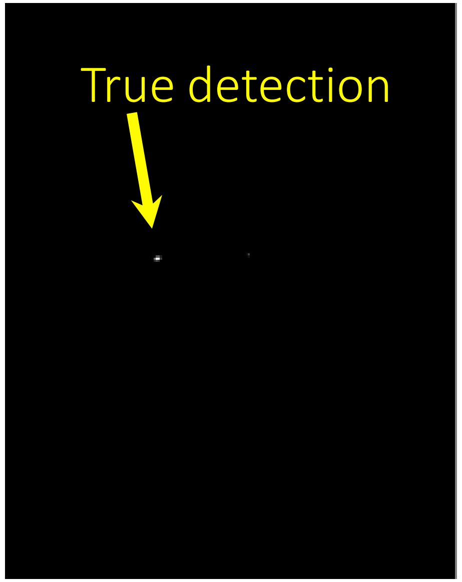
output

$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k, n+l])^2$$

image

1-output

thresholding

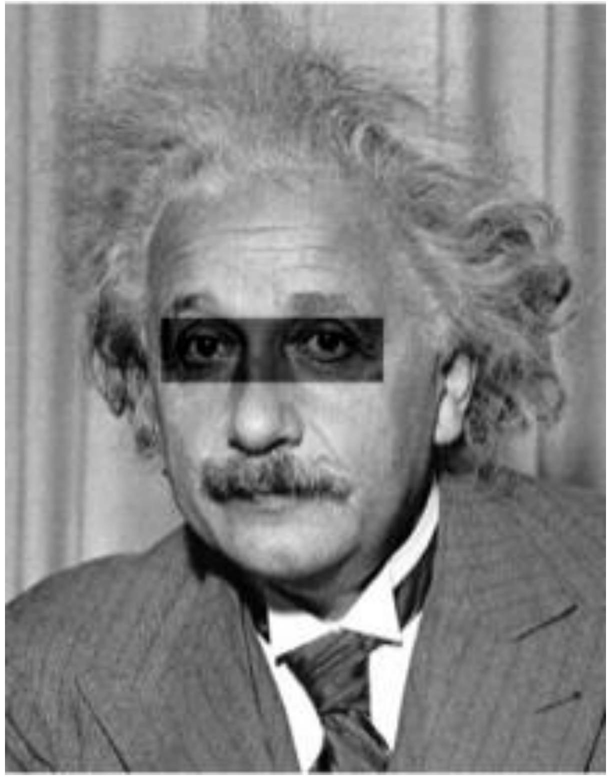


Solution 3: Use sum of squared differences (SSD).

What could go wrong?

Find this template

How do we detect the template  in he following image?



filter 

output

$$\underbrace{h[m,n]}_{\text{output}} = \sum_{k,l} (\underbrace{g[k,l]}_{\text{filter}} - \underbrace{f[m+k,n+l]}_{\text{image}})^2$$

image

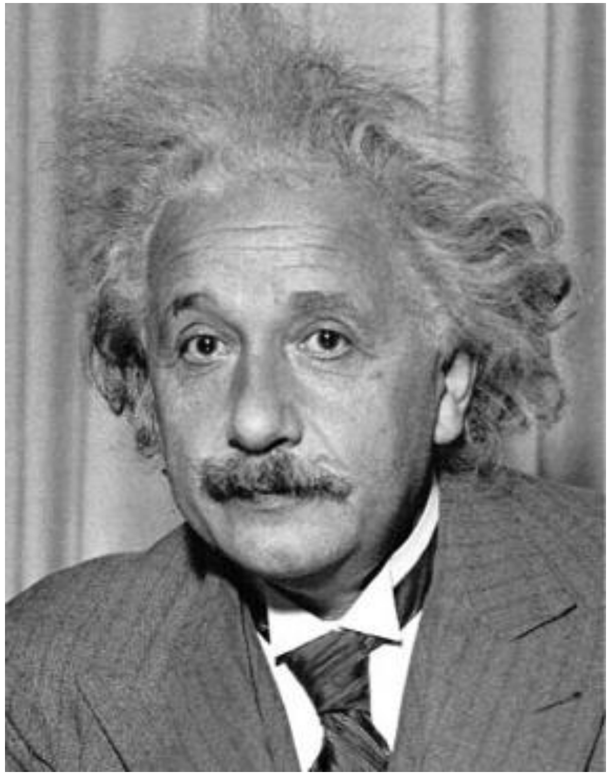
1-output 

Not robust to local intensity changes

Solution 3: Use sum of squared differences (SSD).

Find this template

How do we detect the template  in he following image?



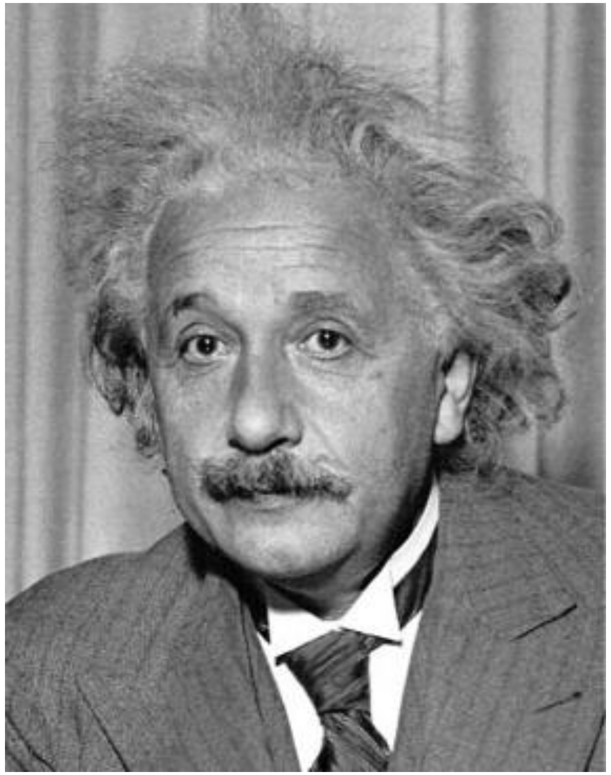
Observations so far:

- subtracting mean deals with brightness bias
- dividing by standard deviation removes contrast bias

Can we combine the two effects?

Find this template

How do we detect the template  in he following image?



What will the output look like?

filter 

template mean

output

$$h[m,n] = \frac{\sum_{k,l}(g[k,l]-\bar{g})(f[m+k,n+l]-\bar{f}_{m,n})}{\sqrt{\left(\sum_{k,l}(g[k,l]-\bar{g})^2 \sum_{k,l}(f[m+k,n+l]-\bar{f}_{m,n})^2\right)}}$$

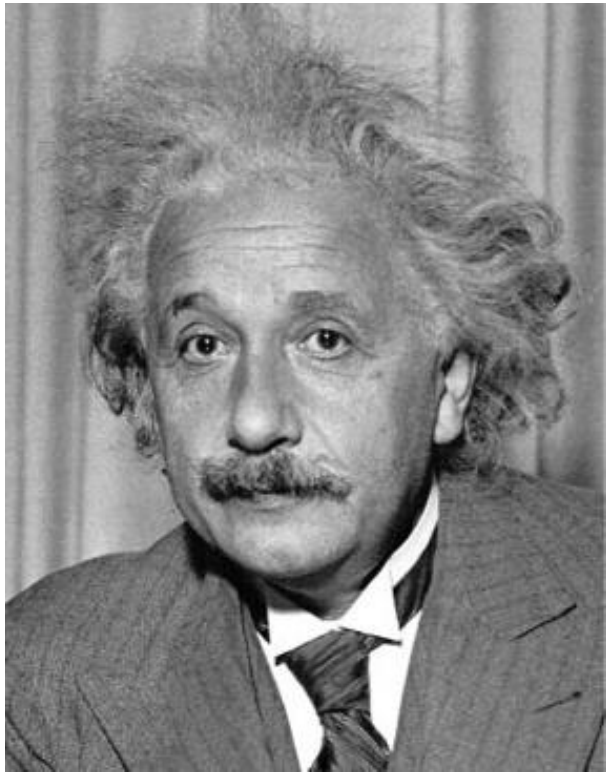
image

local patch mean

Solution 4: Normalized cross-correlation (NCC).

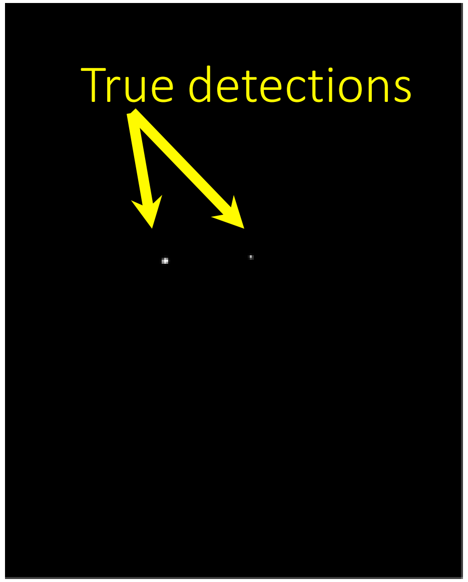
Find this template

How do we detect the template  in he following image?



1-output

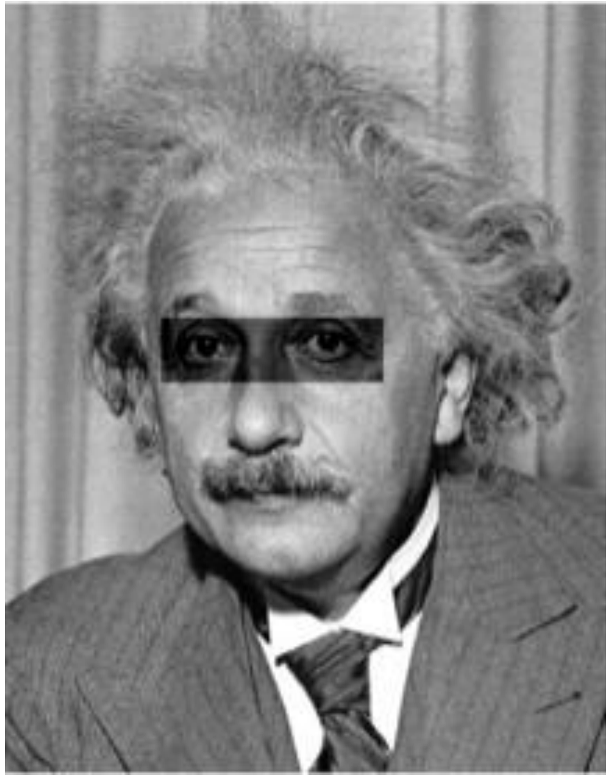
thresholding



Solution 4: Normalized cross-correlation (NCC).

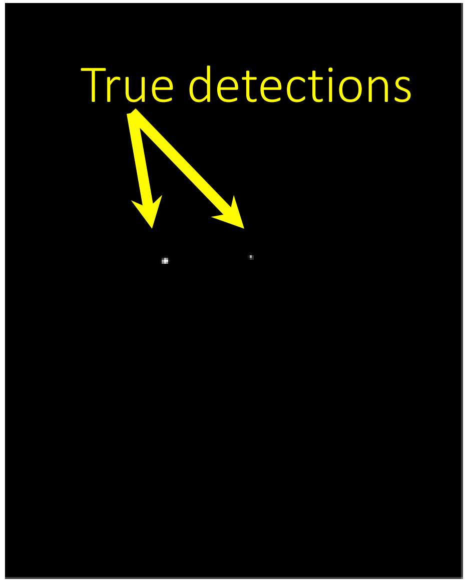
Find this template

How do we detect the template  in he following image?



1-output

thresholding



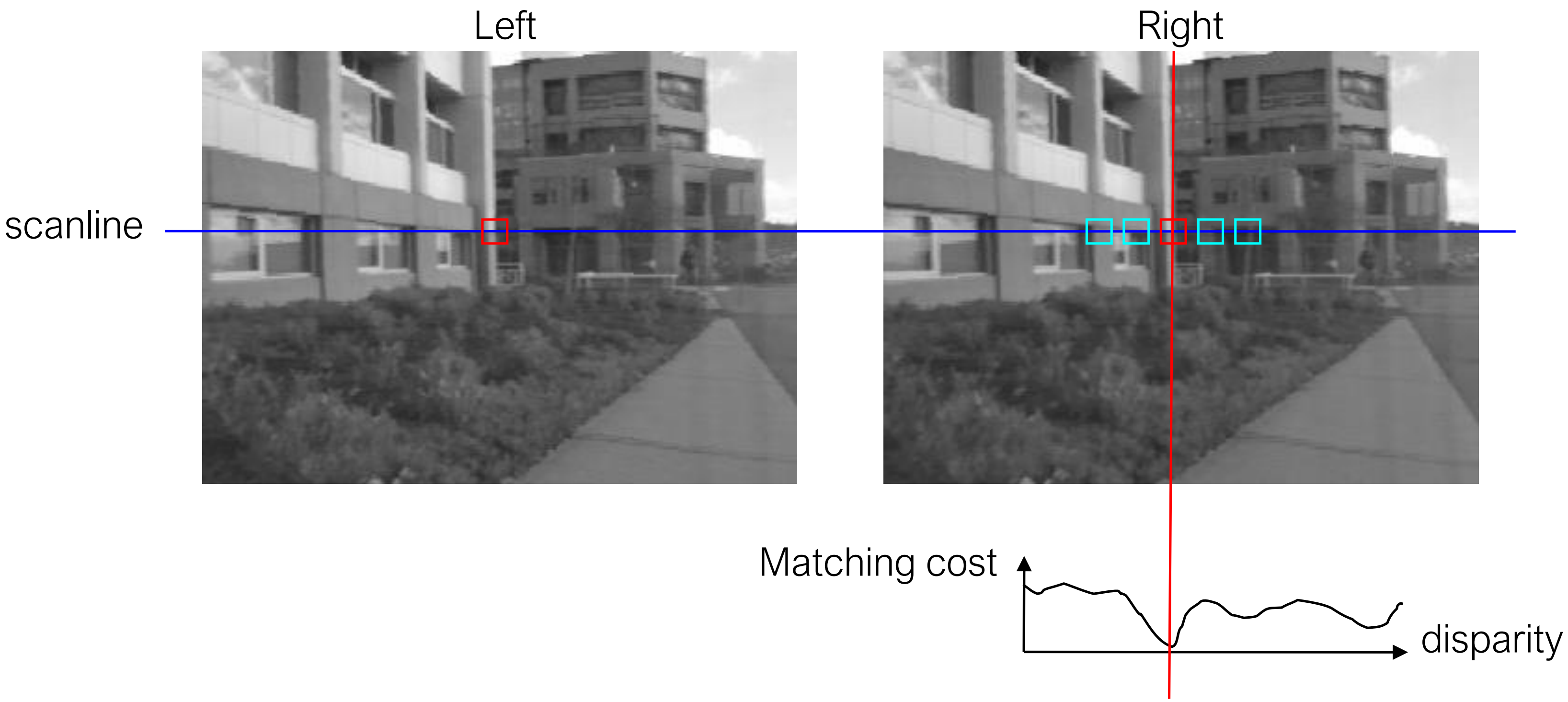
Solution 4: Normalized cross-correlation (NCC).

What is the best method?

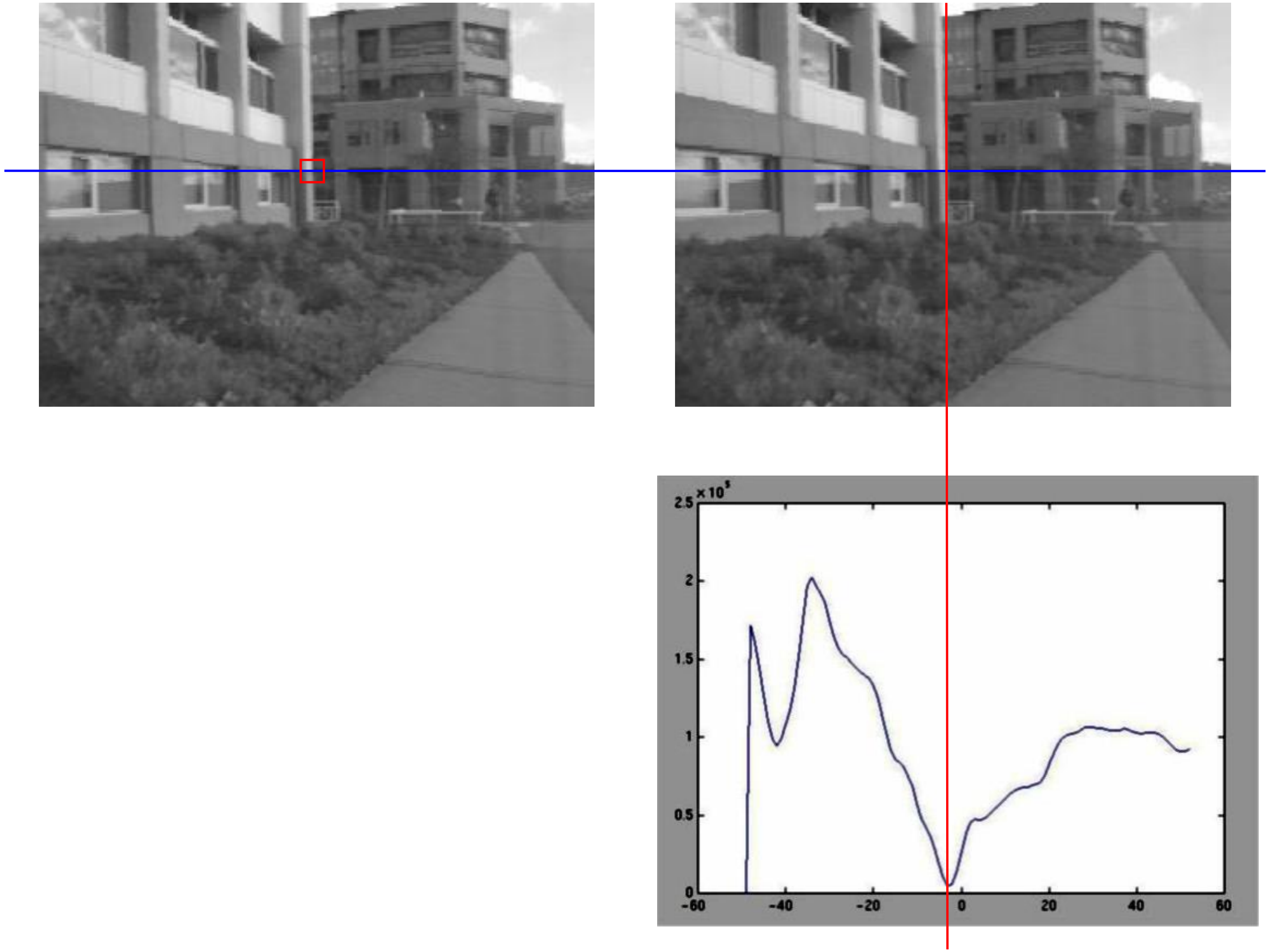
It depends on whether you care about speed or invariance.

- Zero-mean: Fastest, very sensitive to local intensity.
- Sum of squared differences: Medium speed, sensitive to intensity offsets.
- Normalized cross-correlation: Slowest, invariant to contrast and brightness.

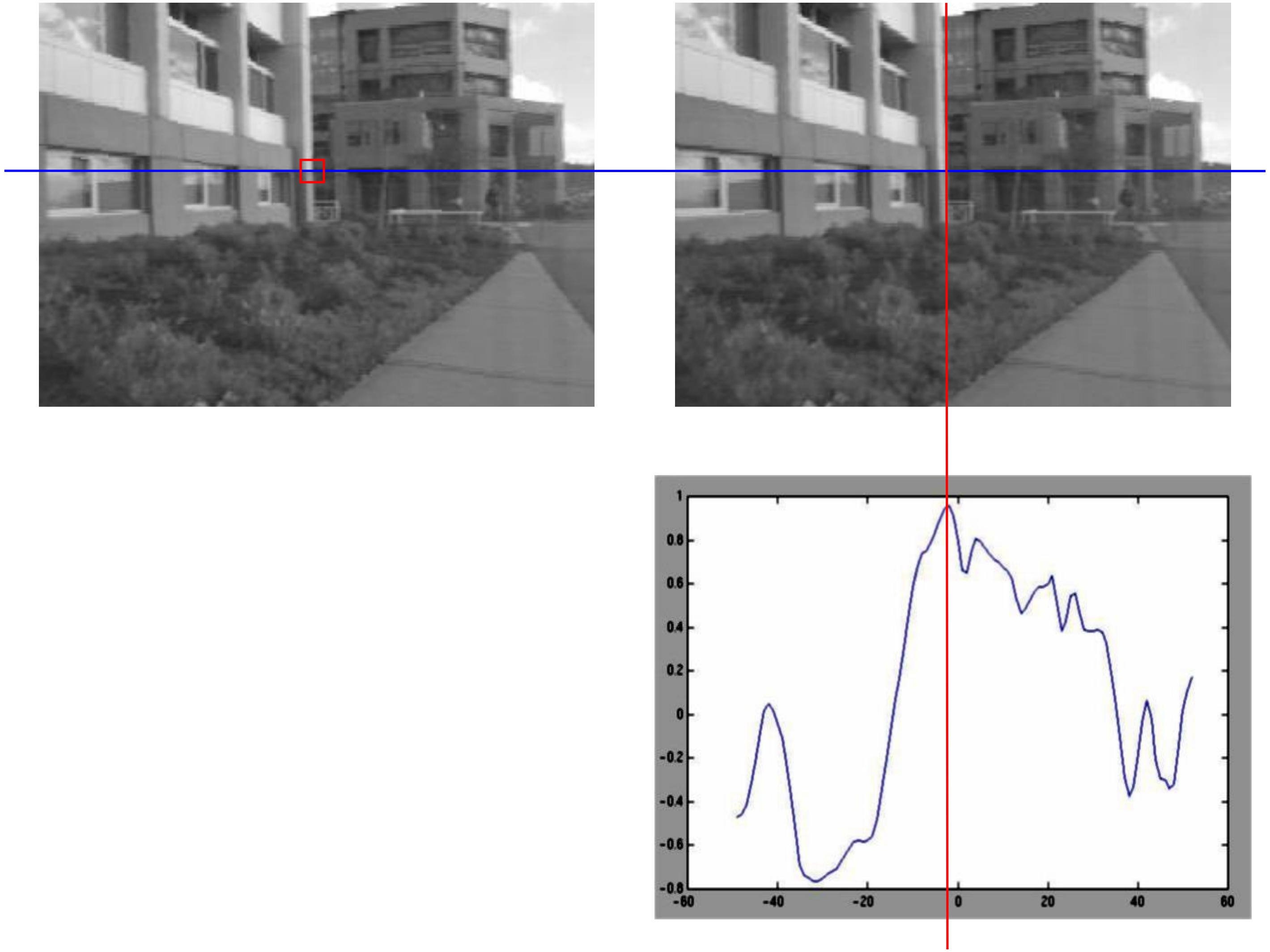
Stereo Block Matching



- Slide a window along the epipolar line and compare contents of that window with the reference window in the left image
- Matching cost: SSD or normalized correlation

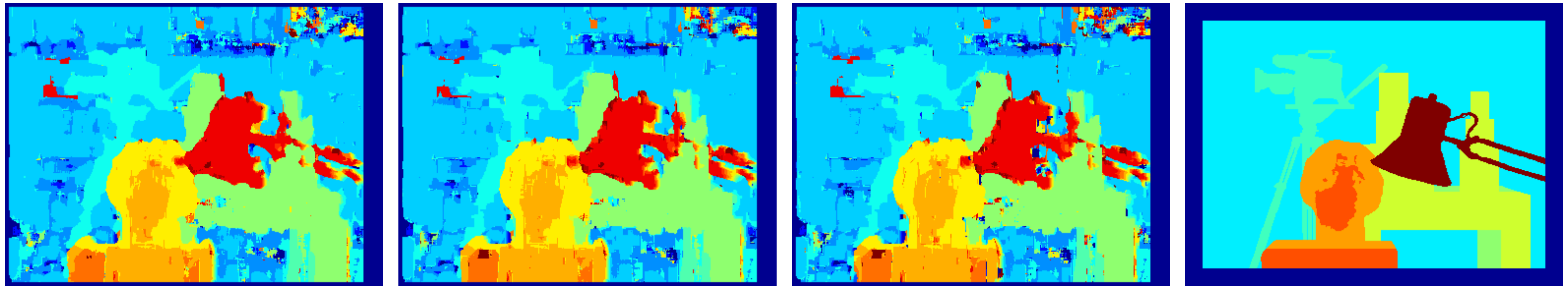


SSD



Normalized cross-correlation

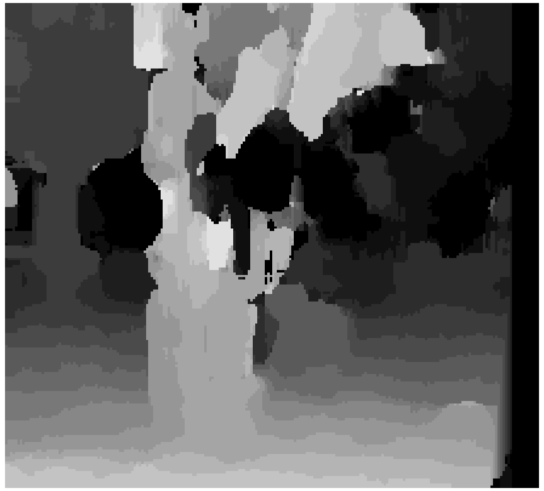
Similarity Measure	Formula
Sum of Absolute Differences (SAD)	$\sum_{(i,j)\in W} \lvert I_1(i,j) - I_2(x+i,y+j) \rvert$
Sum of Squared Differences (SSD)	$\sum_{(i,j)\in W} \left(I_1(i,j) - I_2(x+i,y+j)\right)^2$
Zero-mean SAD	$\sum_{(i,j)\in W} \lvert I_1(i,j) - \bar{I}_1(i,j) - I_2(x+i,y+j) + \bar{I}_2(x+i,y+j) \rvert$
Locally scaled SAD	$\sum_{(i,j)\in W} \lvert I_1(i,j) - \frac{\bar{I}_1(i,j)}{\bar{I}_2(x+i,y+j)} I_2(x+i,y+j) \rvert$
Normalized Cross Correlation (NCC)	$\frac{\sum_{(i,j)\in W} I_1(i,j).I_2(x+i,y+j)}{\sqrt[2]{\sum_{(i,j)\in W} I_1^2(i,j).\sum_{(i,j)\in W} I_2^2(x+i,y+j)}}$



SAD SSD NCC Ground truth

Effect of window size

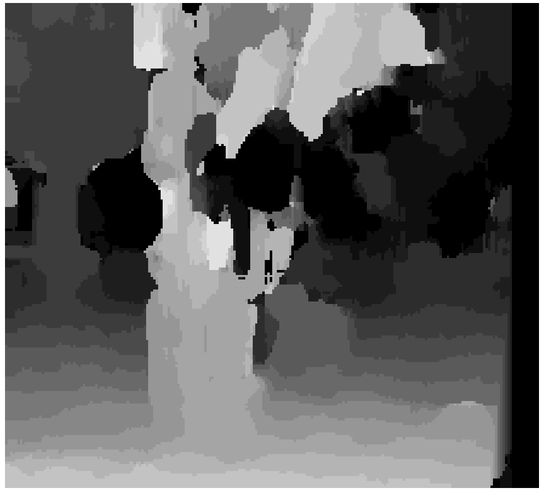
W = 3



W = 20

Effect of window size

W = 3



W = 20

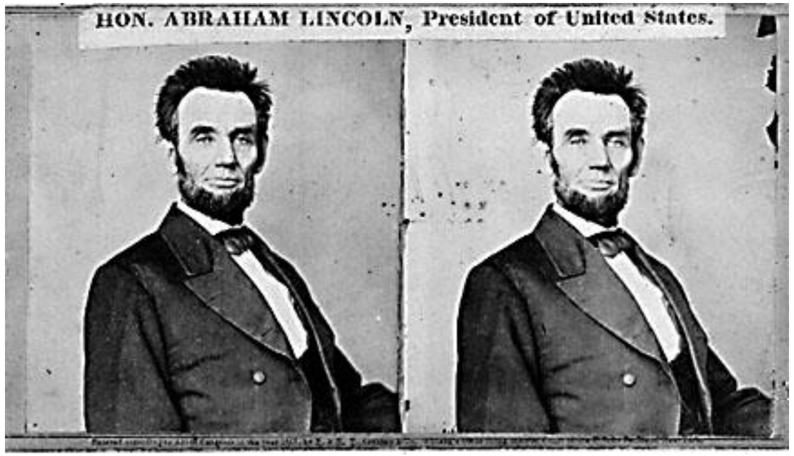
Smaller window

- \+ More detail
- \- More noise

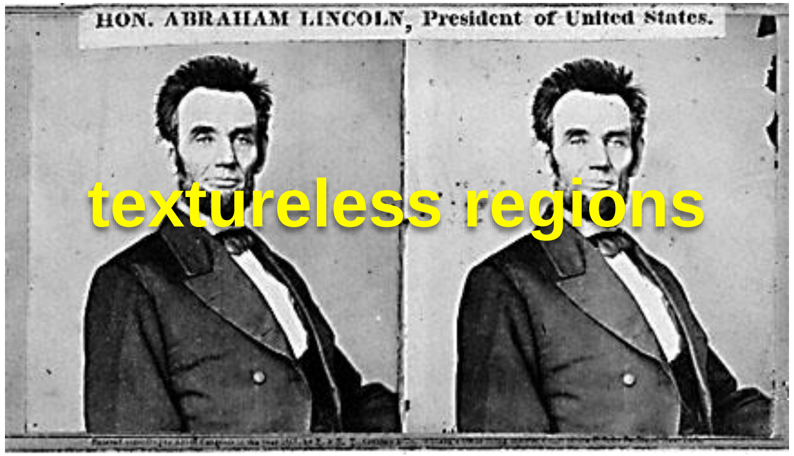
Larger window

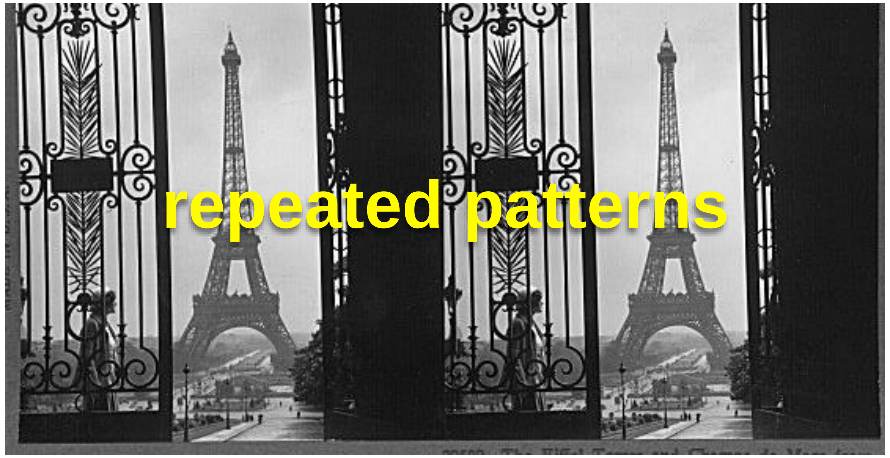
- \+ Smoother disparity maps
- \- Less detail
- \- Fails near boundaries

When will stereo block matching fail?



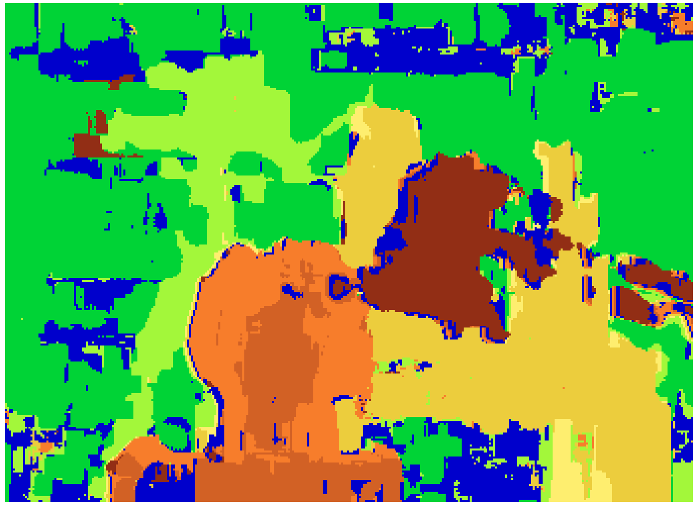
When will stereo block matching fail?





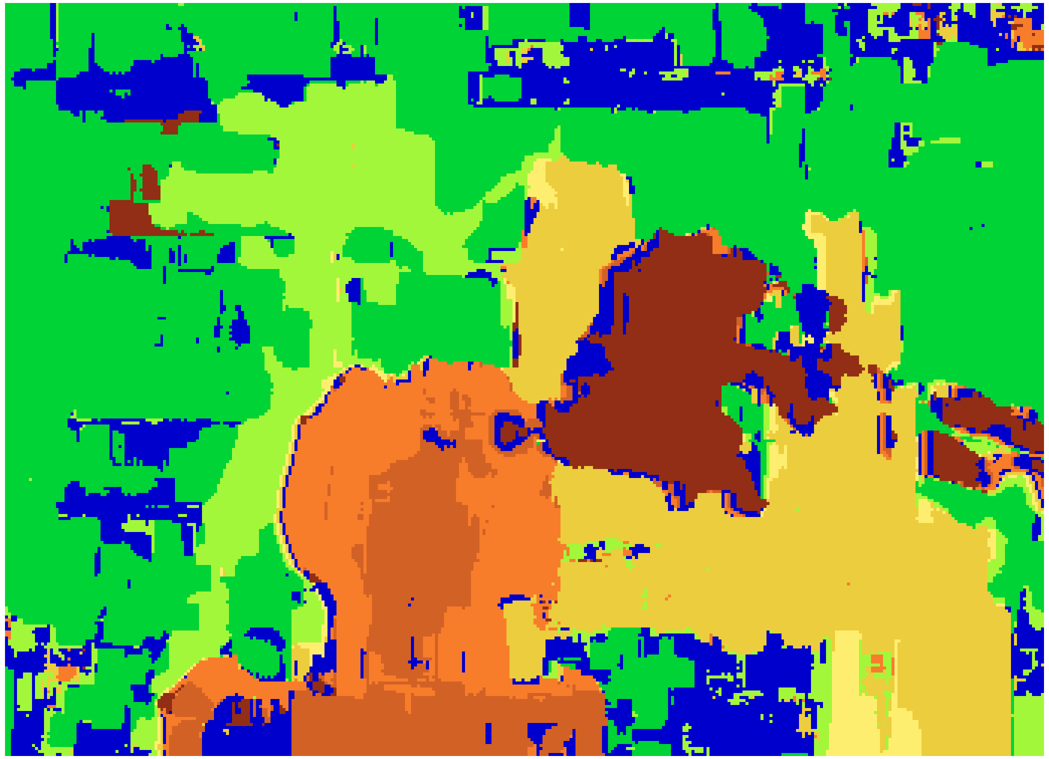
Improving stereo matching

Block matching

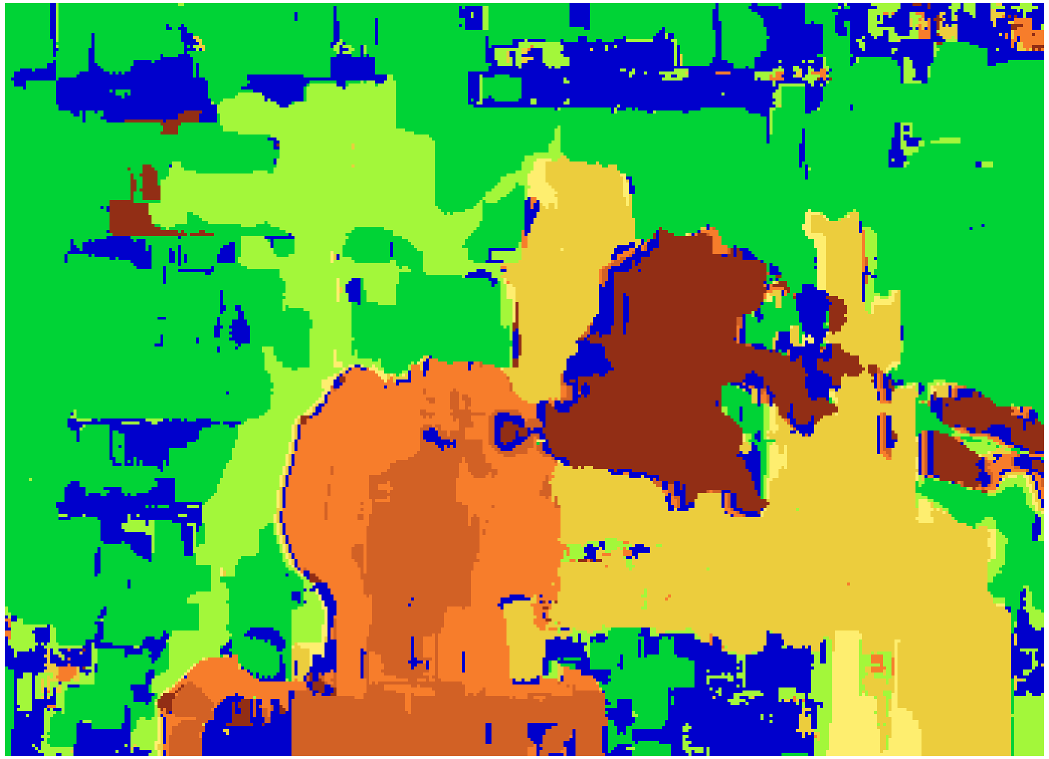


Ground truth

What are some problems with the result?



How can we improve depth estimation?



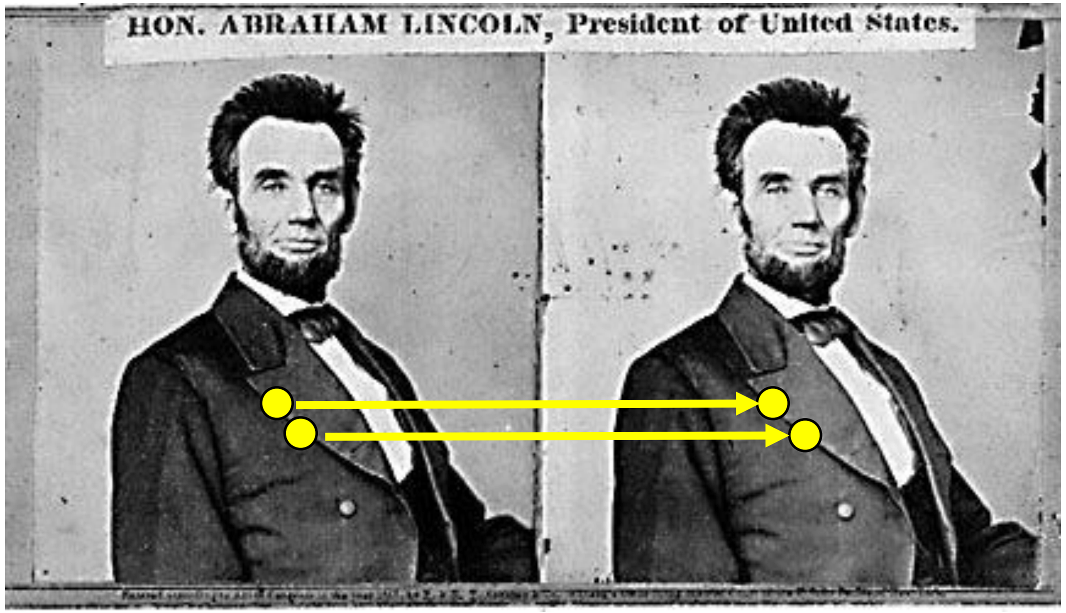
How can we improve depth estimation?

Too many discontinuities.
We expect disparity values to change slowly.

Let's make an assumption:
depth should change smoothly

Stereo matching as …

Energy Minimization



What defines a good stereo correspondence?

1. **Match quality**
   – Want each pixel to find a good match in the other image
2. **Smoothness**
   – If two pixels are adjacent, they should (usually) move about the same amount

energy function
(for one pixel)

$$E(d) = \underbrace{E_d(d)}_{\text{data term}} + \lambda \underbrace{E_s(d)}_{\text{smoothness term}}$$

Want each pixel to find a good match in the other image

(block matching result)

Adjacent pixels should (usually) move about the same amount

(smoothness function)

$$E(d) = \boxed{E_d(d)} + \lambda E_s(d)$$

$$E_d(d) = \sum_{(x,y)\in I} C(x, y, d(x, y))$$

data term

SSD distance between windows
centered at I(x, y) and J(x+ d(x,y), y)

$$E(d) = E_d(d) + \lambda \boxed{E_s(d)}$$

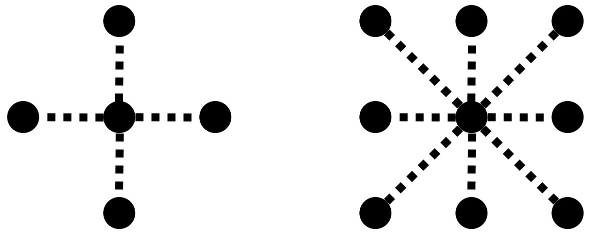
$$E_d(d) = \sum_{(x,y)\in I} C(x, y, d(x, y))$$

SSD distance between windows
centered at I(x, y) and J(x+ d(x,y), y)

$$E_s(d) = \sum_{(p,q)\in \mathcal{E}} V(d_p, d_q)$$

smoothness term

$\mathcal{E}$: set of neighboring pixels



4-connected neighborhood

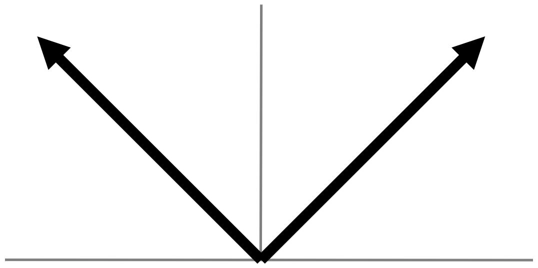
8-connected neighborhood

$$E_s(d) = \sum_{(p,q)\in\mathcal{E}} V(d_p, d_q)$$

smoothness term

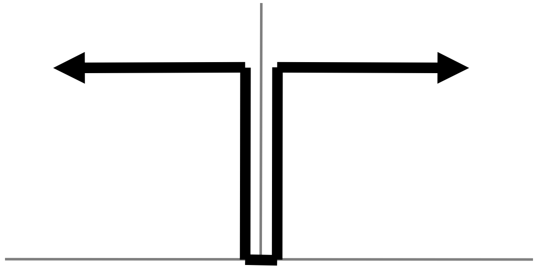
$$V(d_p, d_q) = |d_p - d_q|$$

L_1 distance



$$V(d_p, d_q) = \begin{cases} 0 & \text{if } d_p = d_q \\ 1 & \text{if } d_p \neq d_q \end{cases}$$

"Potts model"



One possible solution…

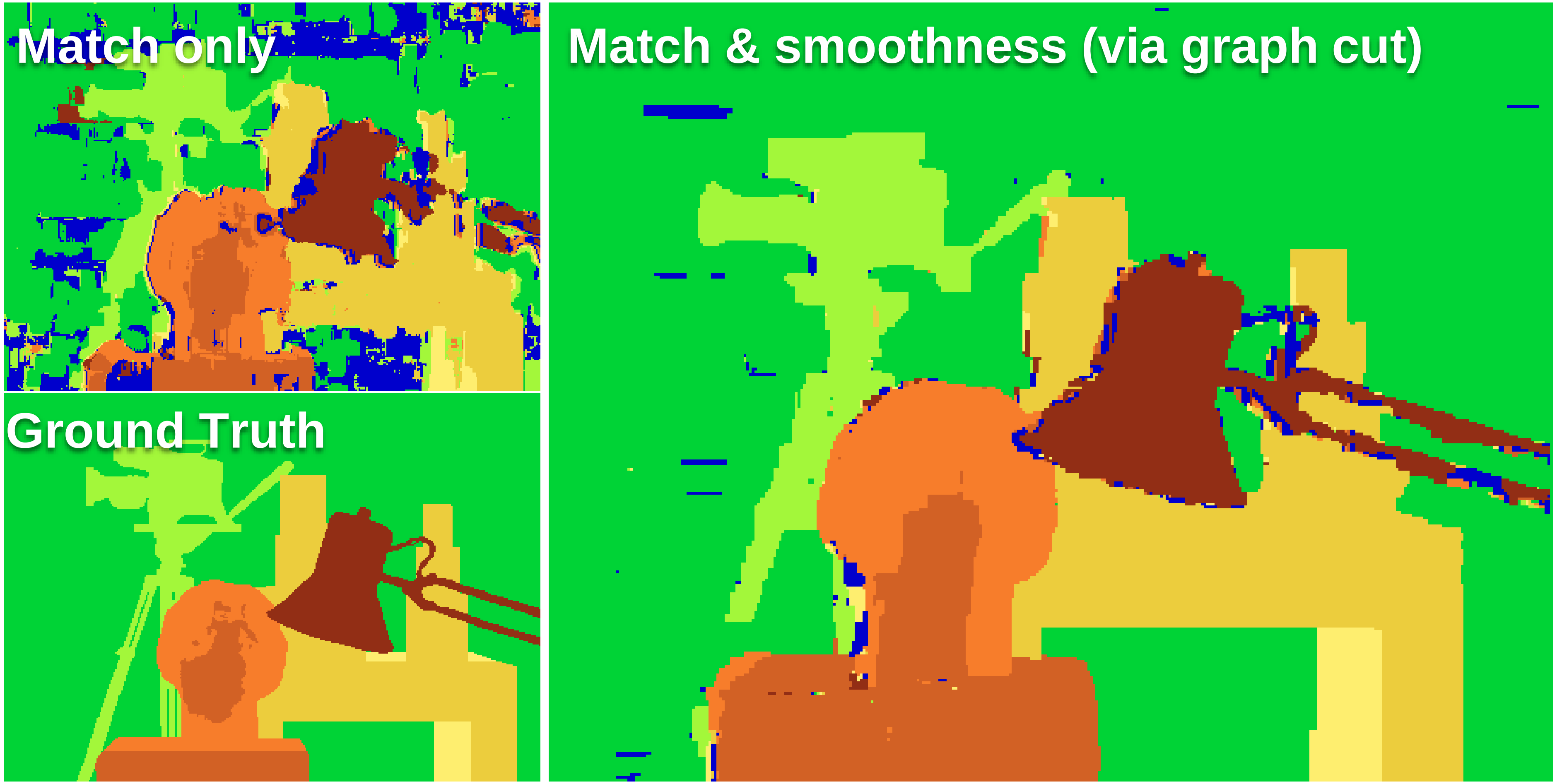
Dynamic Programming

$$E(d) = E_d(d) + \lambda E_s(d)$$

Can minimize this independently per scanline
using dynamic programming (DP)

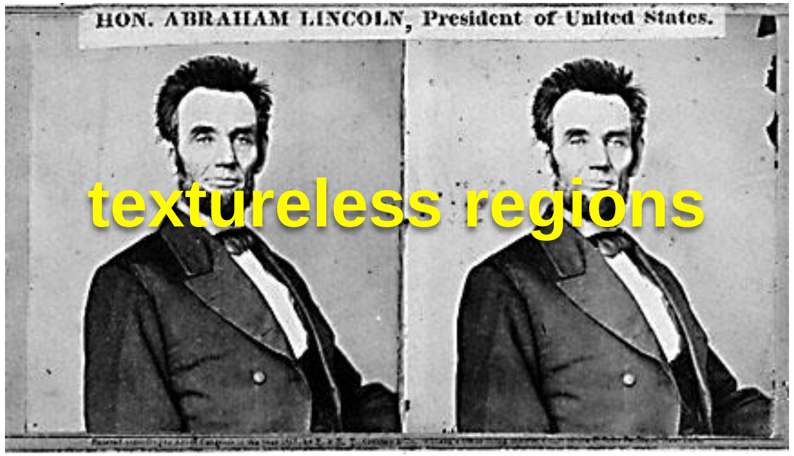
$D(x, y, d)$: minimum cost of solution such that d(x,y) = d

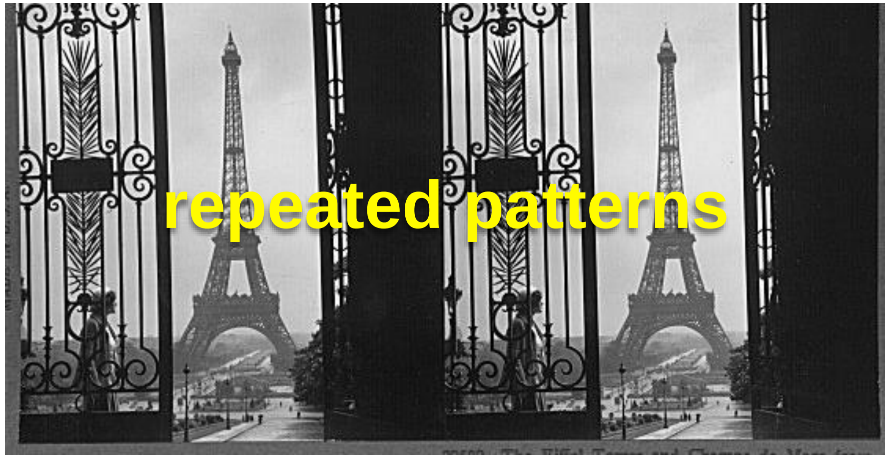
$$D(x, y, d) = C(x, y, d) + \min_{d'} \{D(x-1, y, d') + \lambda \, |d - d'|\}$$



Y. Boykov, O. Veksler, and R. Zabih, Fast Approximate Energy Minimization via Graph Cuts, PAMI 2001

All of these cases remain difficult, what can we do?

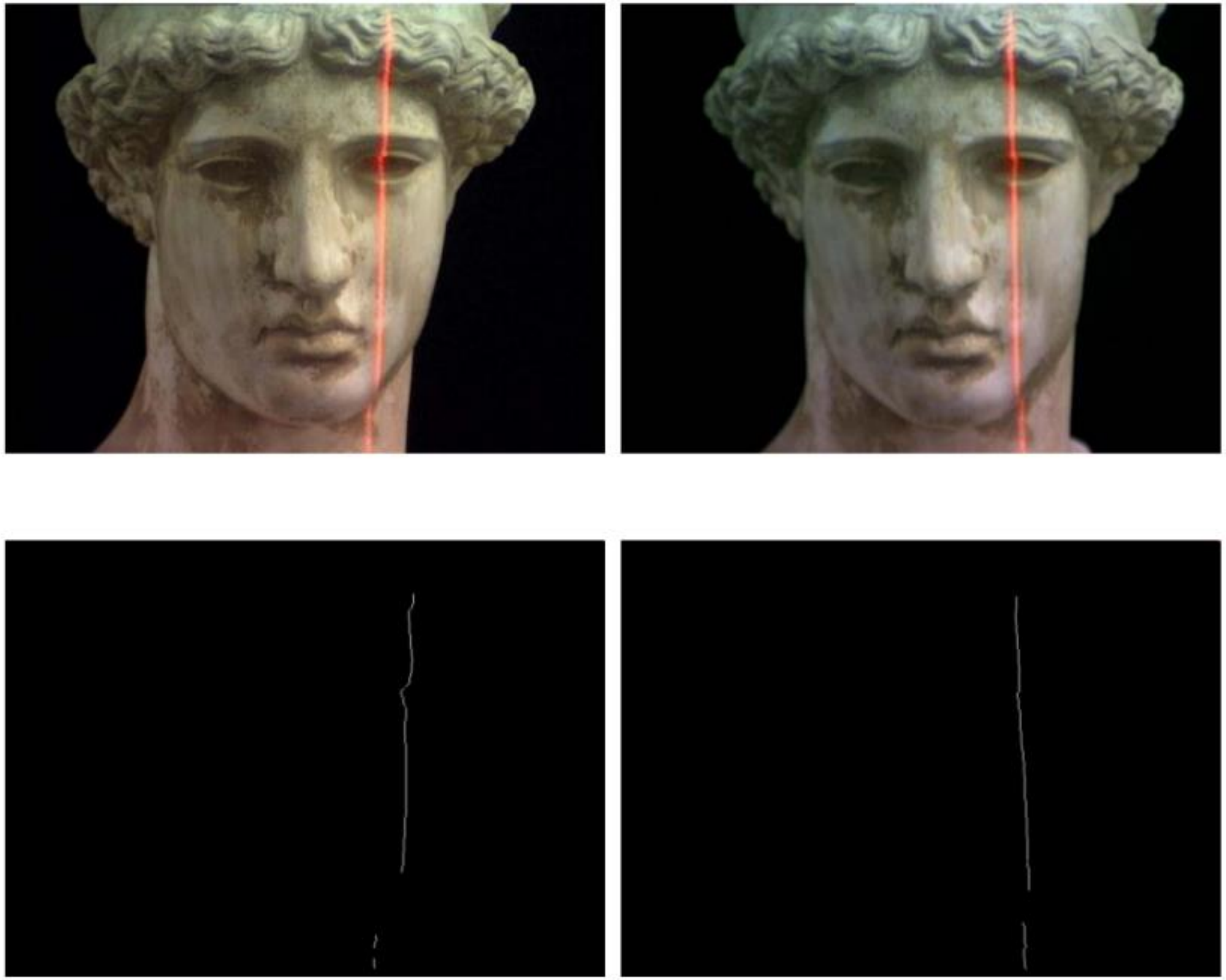




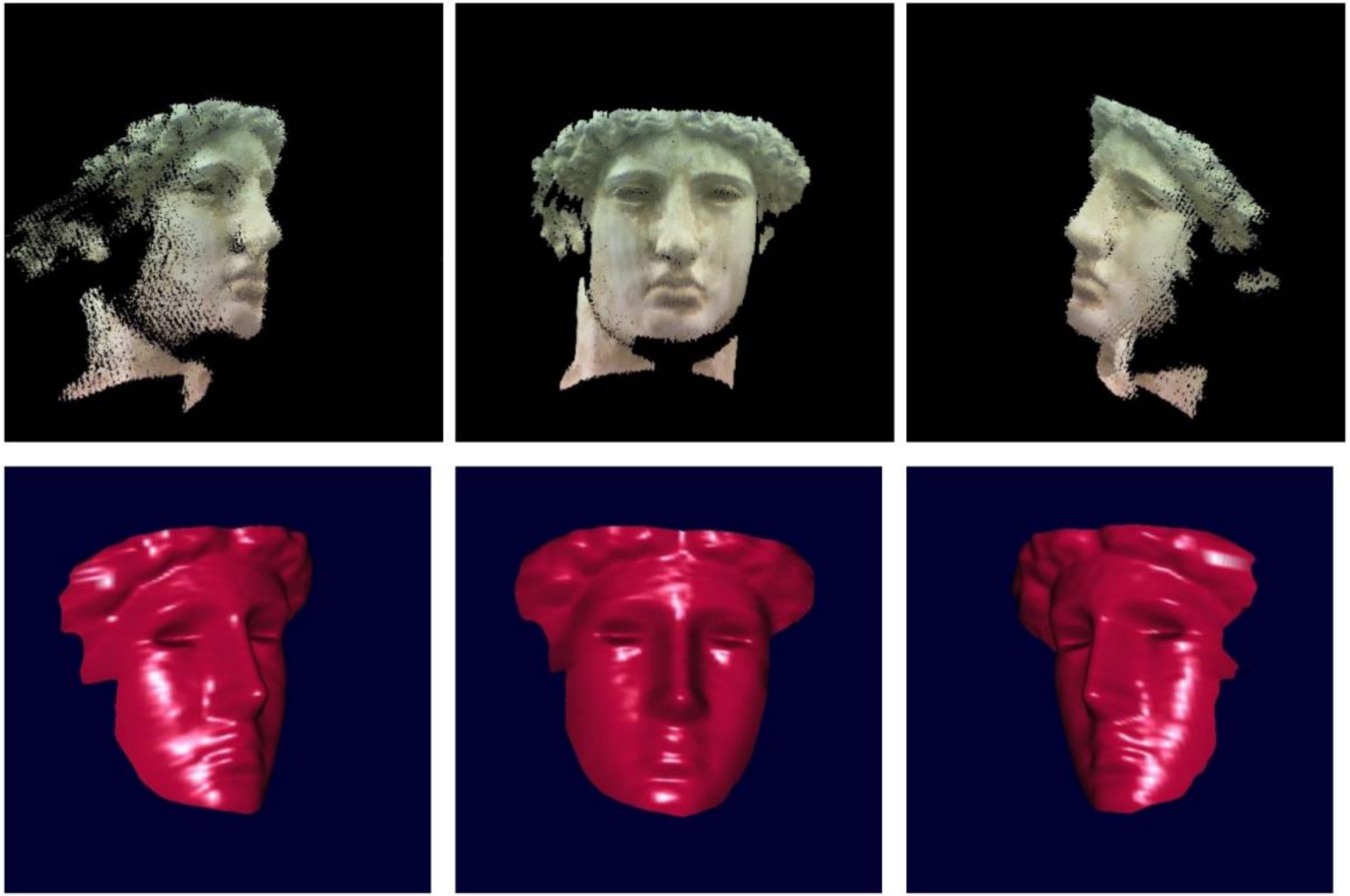
Structured light

Use controlled ("structured") light to make correspondences easier

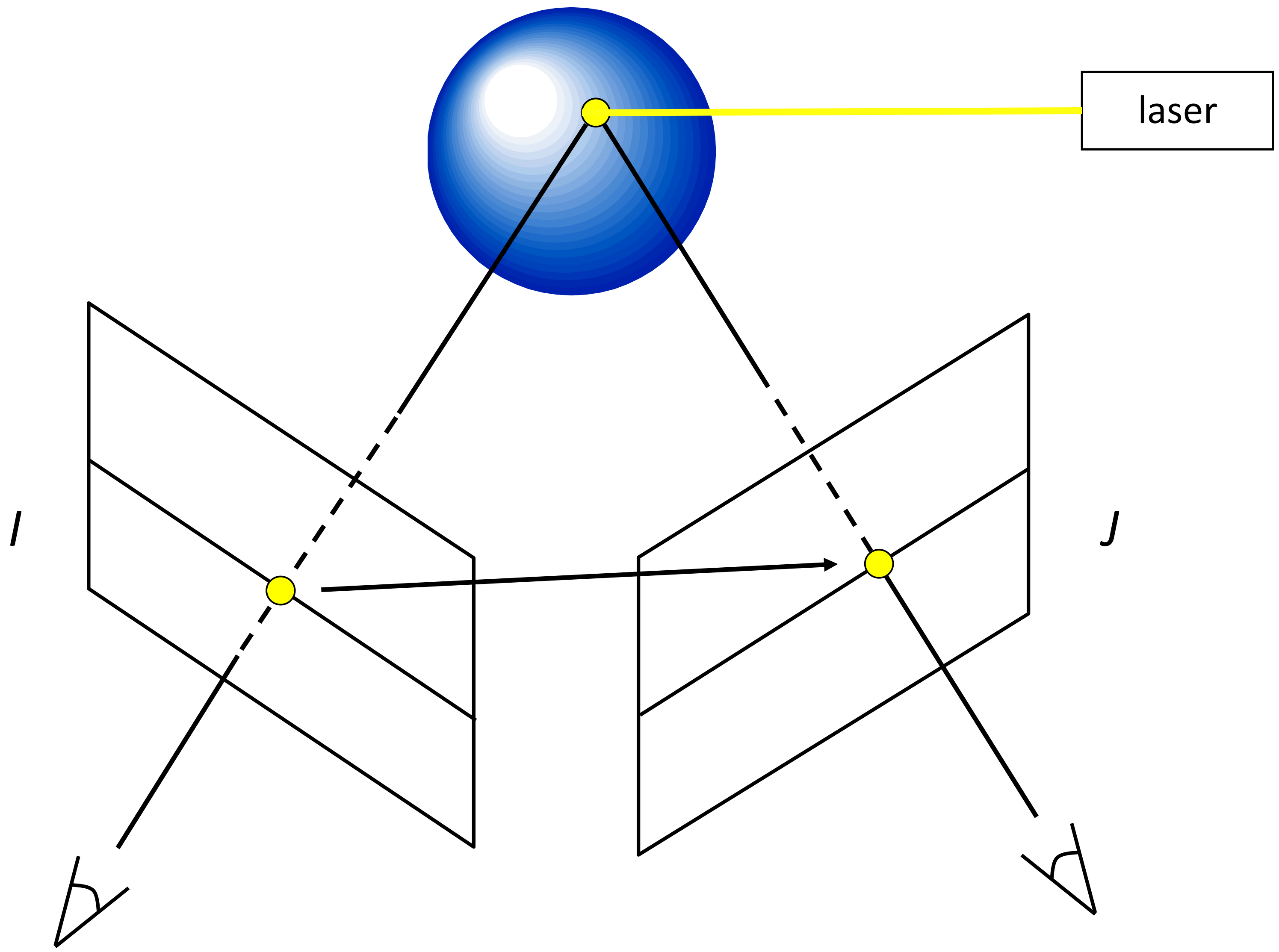
Disparity between laser points on the same scanline in the images determines the 3-D coordinates of the laser point on object



Use controlled (“structured”) light to make correspondences easier

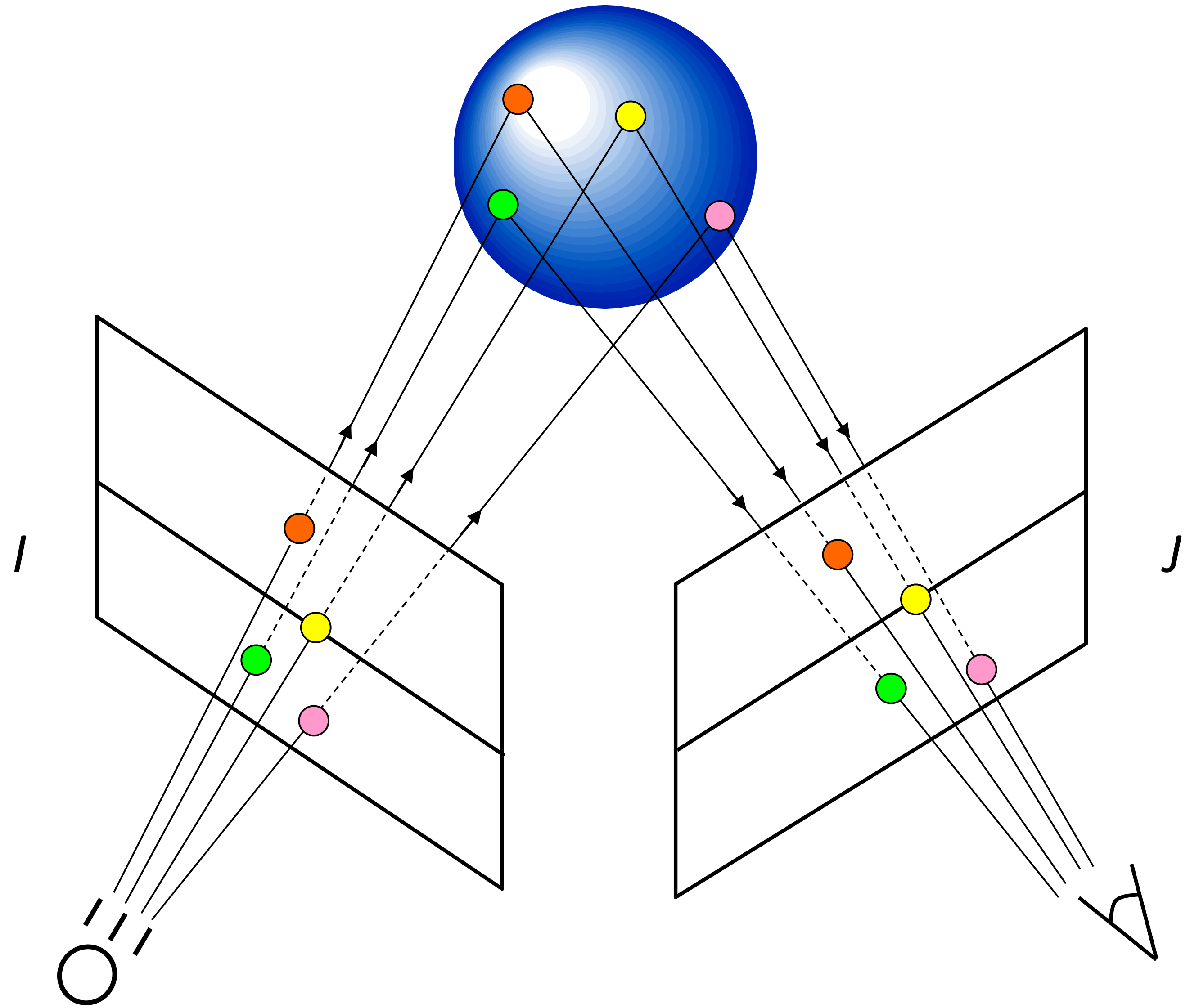


Structured light and two cameras

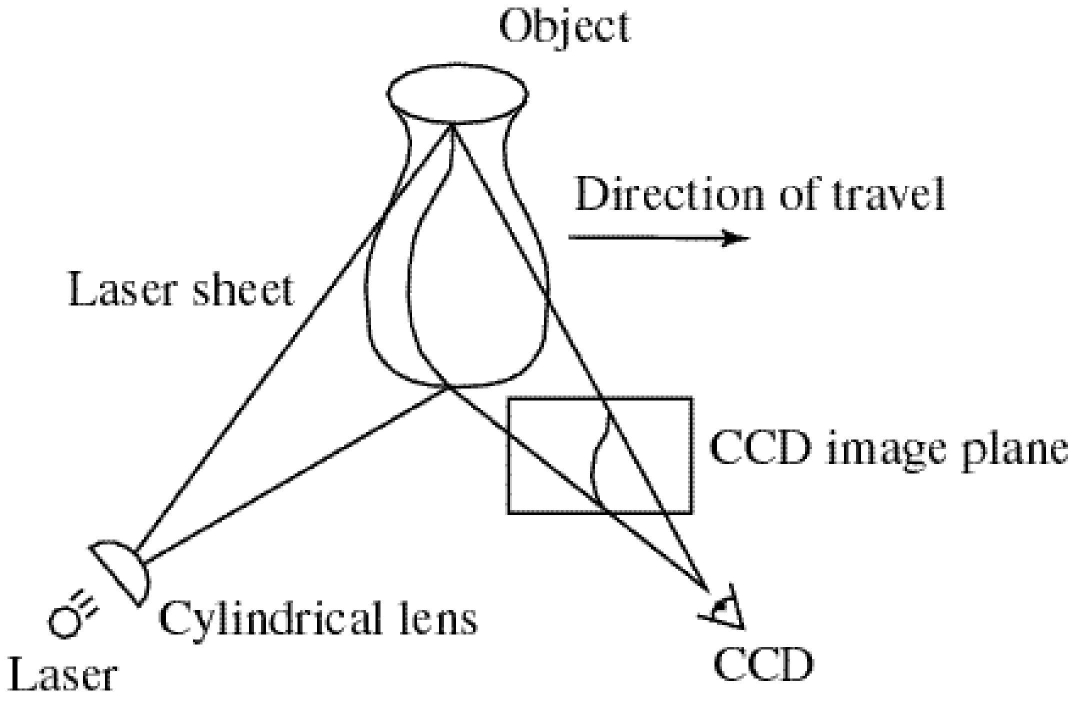


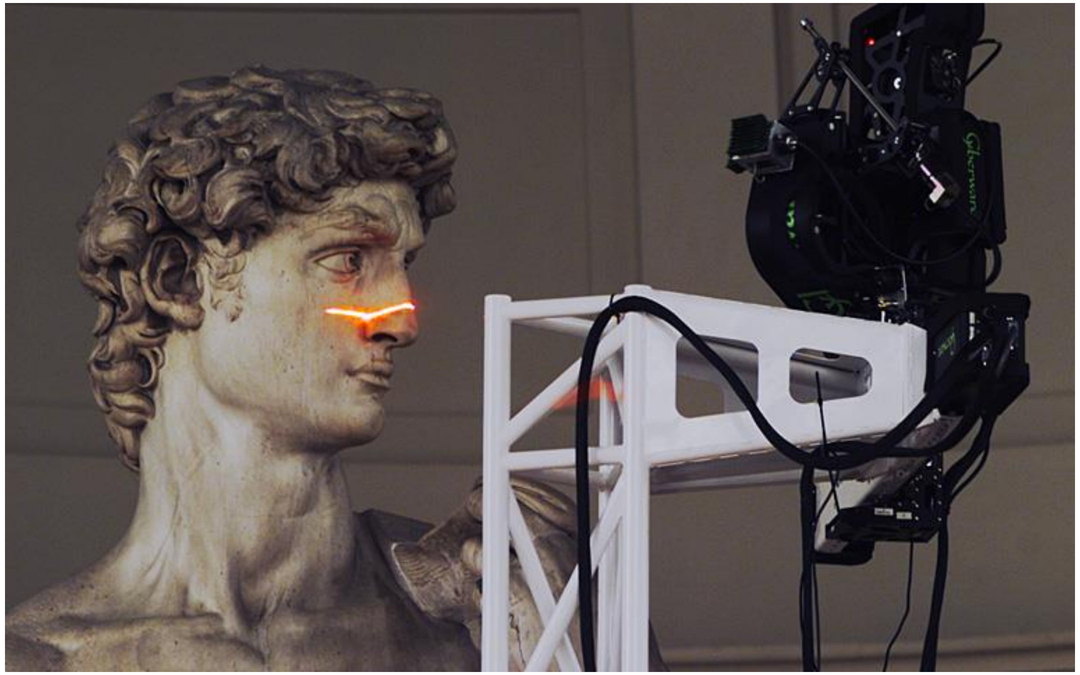
Structured light and one camera

Projector acts like
“reverse” camera



Example: Laser scanner

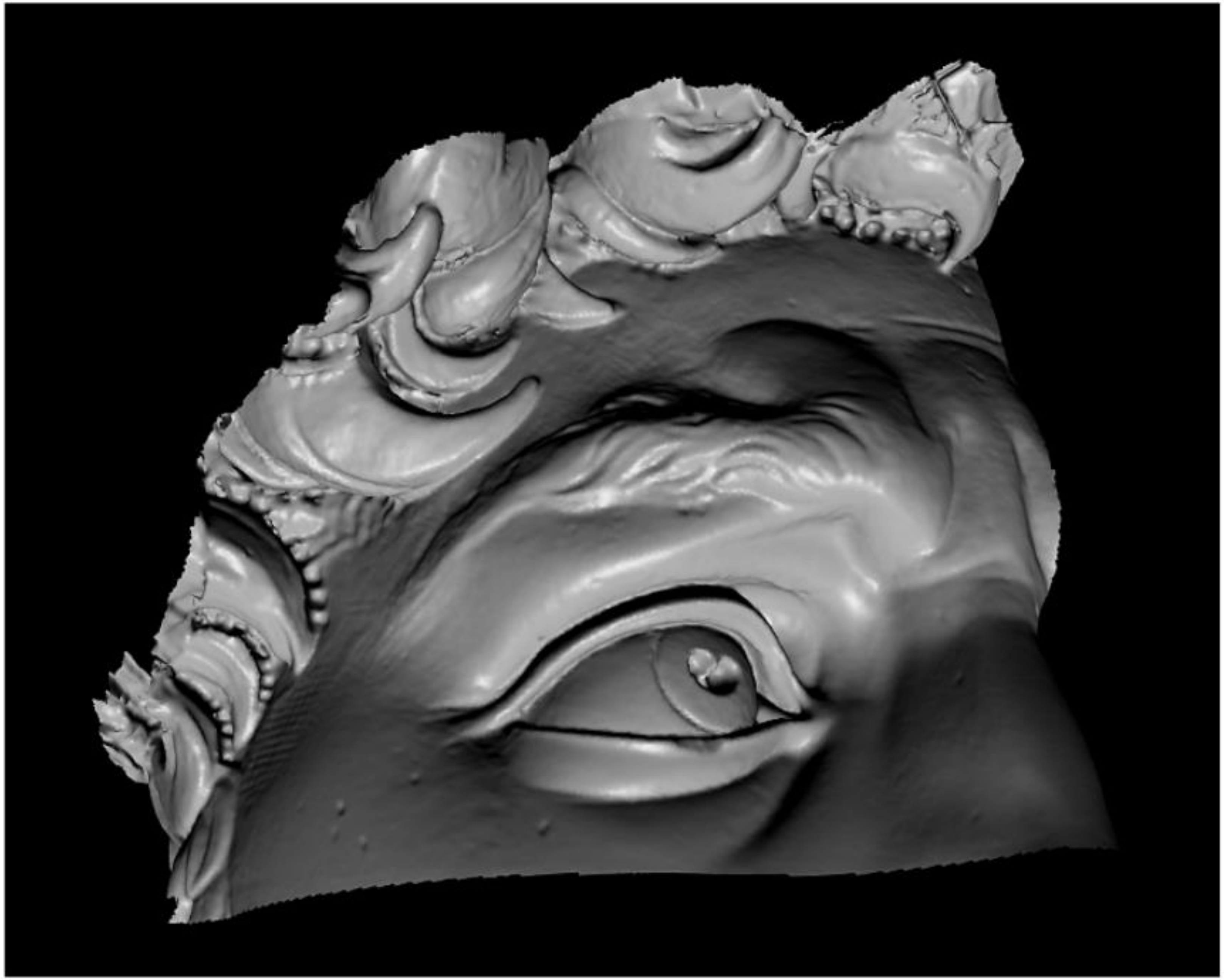




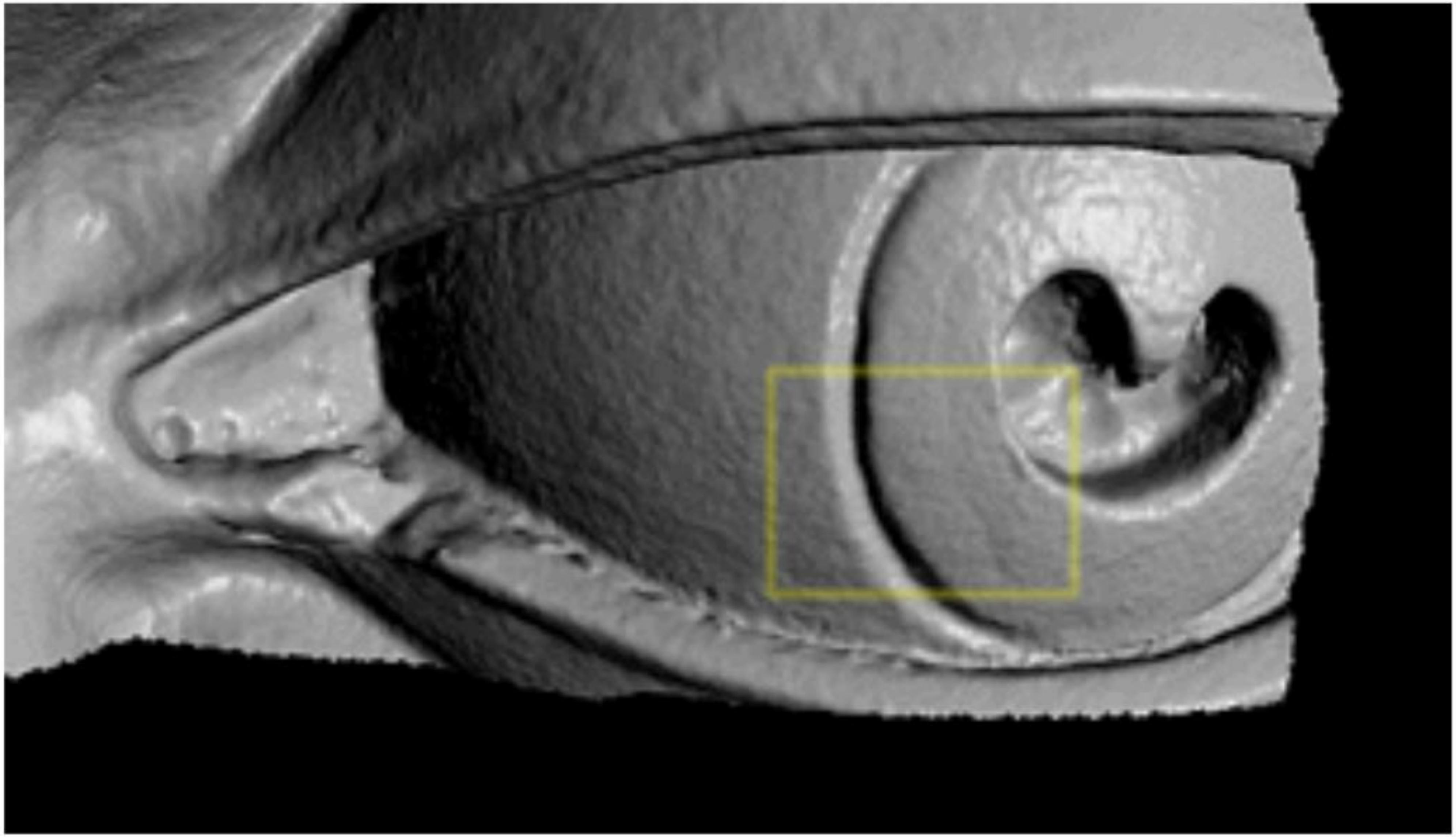
Digital Michelangelo Project
http://graphics.stanford.edu/projects/mich/

The Digital Michelangelo Project, Levoy et al.

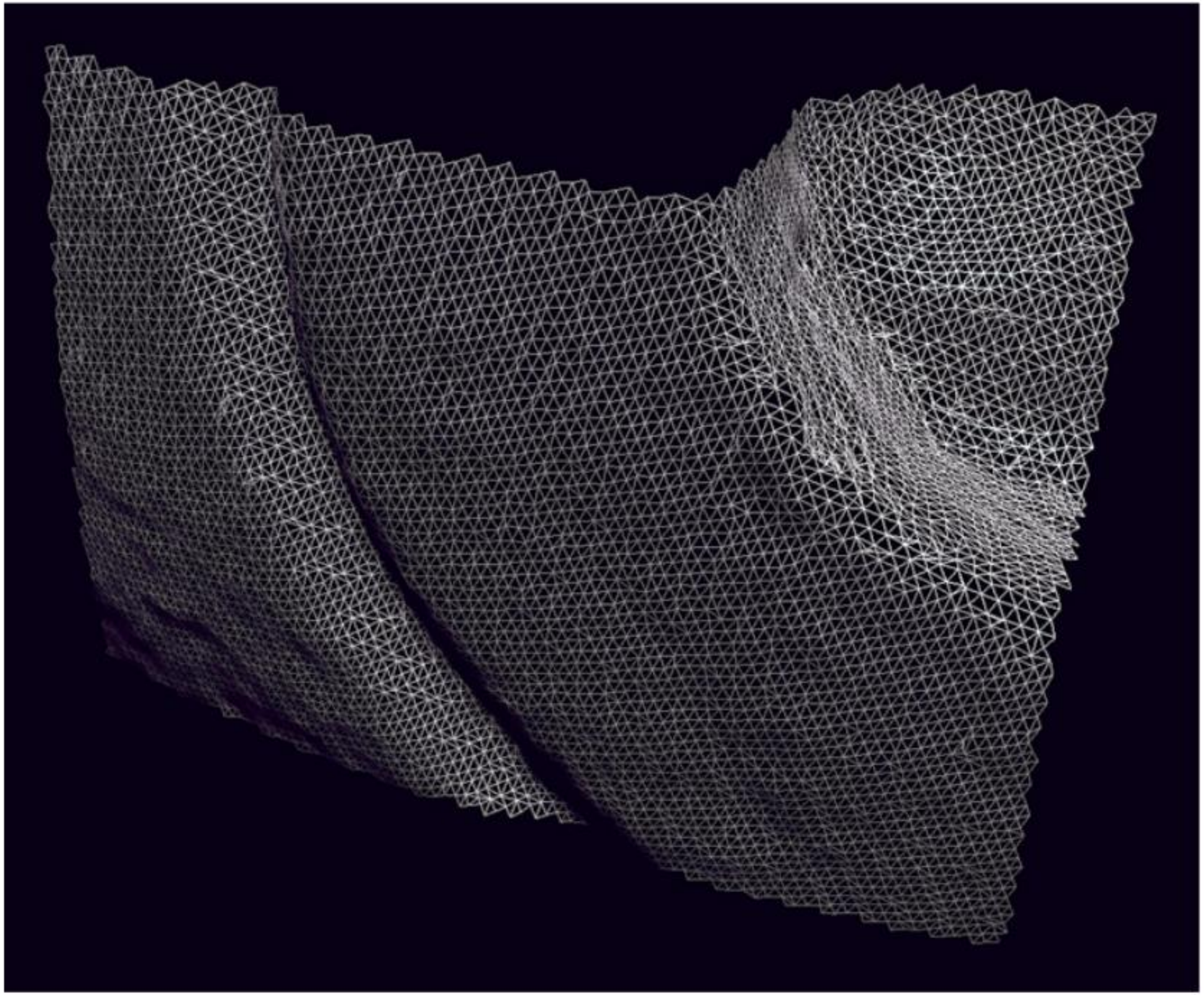
The Digital Michelangelo Project, Levoy et al.



The Digital Michelangelo Project, Levoy et al.



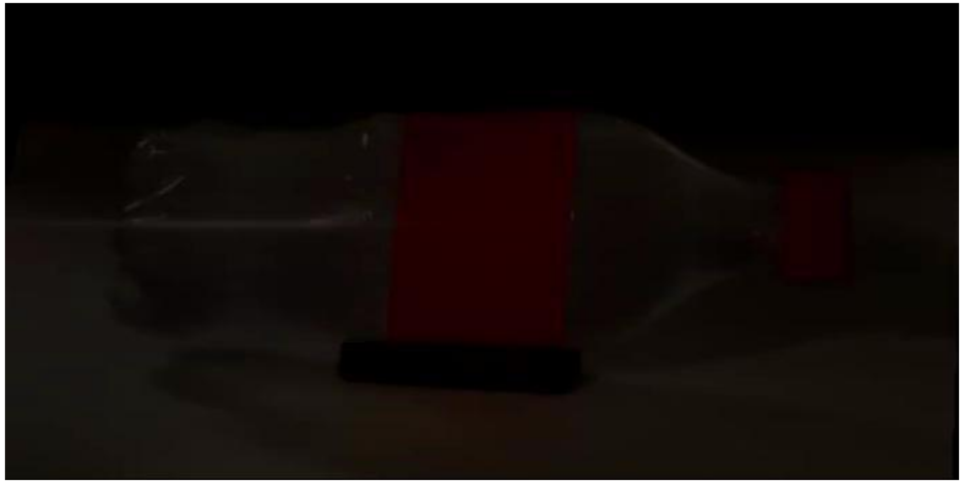
The Digital Michelangelo Project, Levoy et al.



The Digital Michelangelo Project, Levoy et al.

15-463/15-663/15-862 Computational Photography

Learn about structured light and other cameras – and build some on your own!



cameras that take video at the speed of light

cameras that measure depth in real time

cameras that see around corners

cameras that capture entire focal stacks

http://graphics.cs.cmu.edu/courses/15-463/

References

Basic reading:
- Szeliski textbook, Section 8.1 (not 8.1.1-8.1.3), Chapter 11, Section 12.2.
- Hartley and Zisserman, Section 11.12.