



COMPUTER VISION LECTURE 18 – STEREO

Prof. Dr. Francesco Maurelli
2018-10-30

Courtesy of Ioannis Gkioulekas, CMU

Revisiting triangulation

How would you reconstruct 3D points?

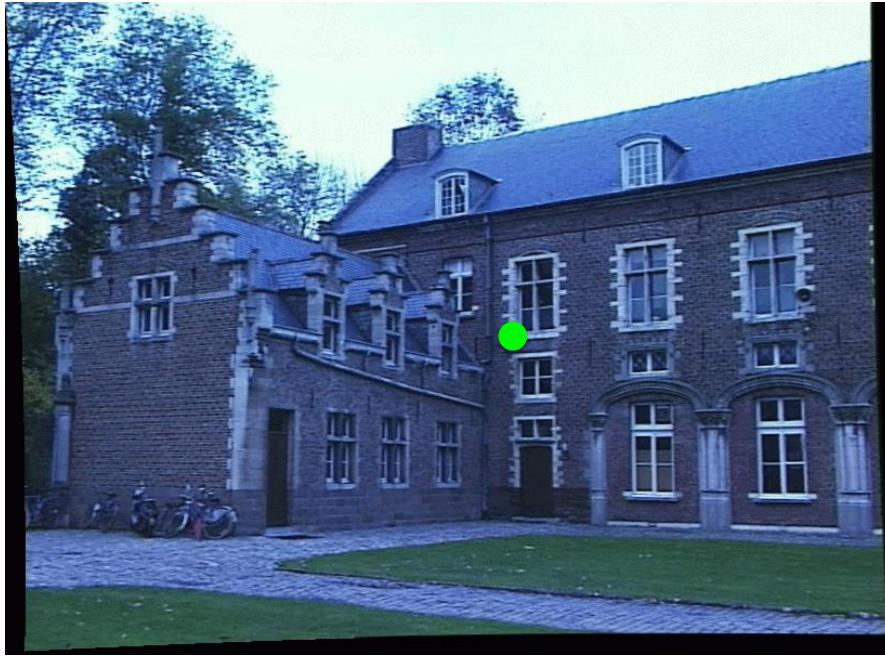


Left image



Right image

How would you reconstruct 3D points?



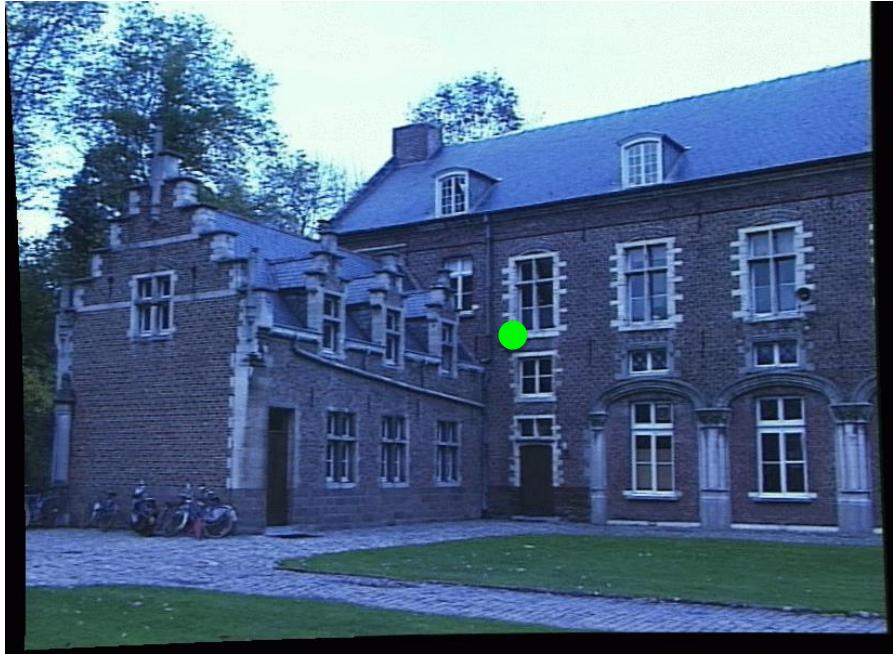
Left image



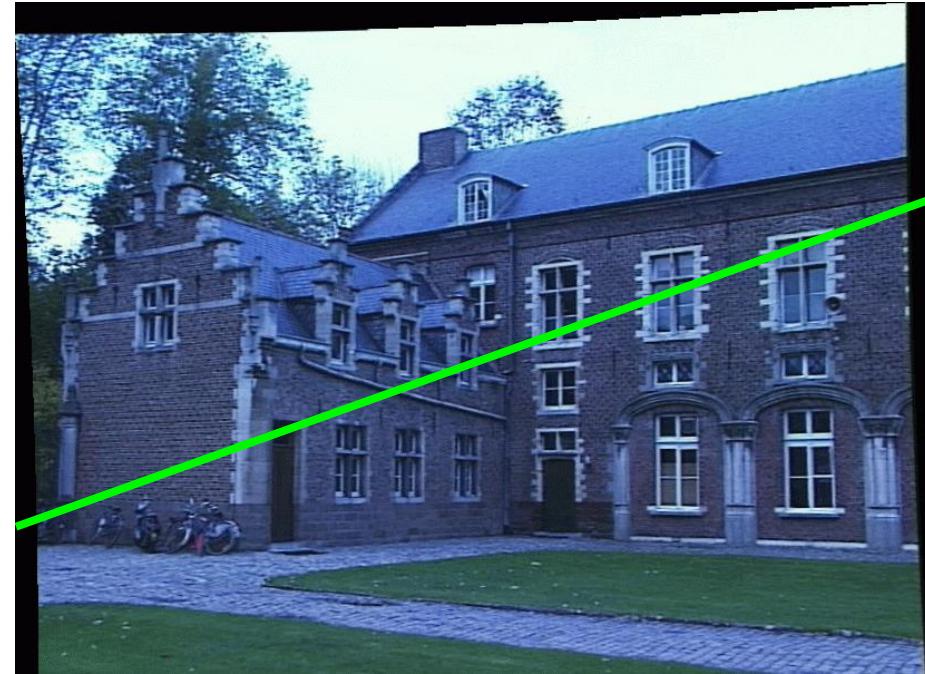
Right image

1. Select point in one image (how?)

How would you reconstruct 3D points?



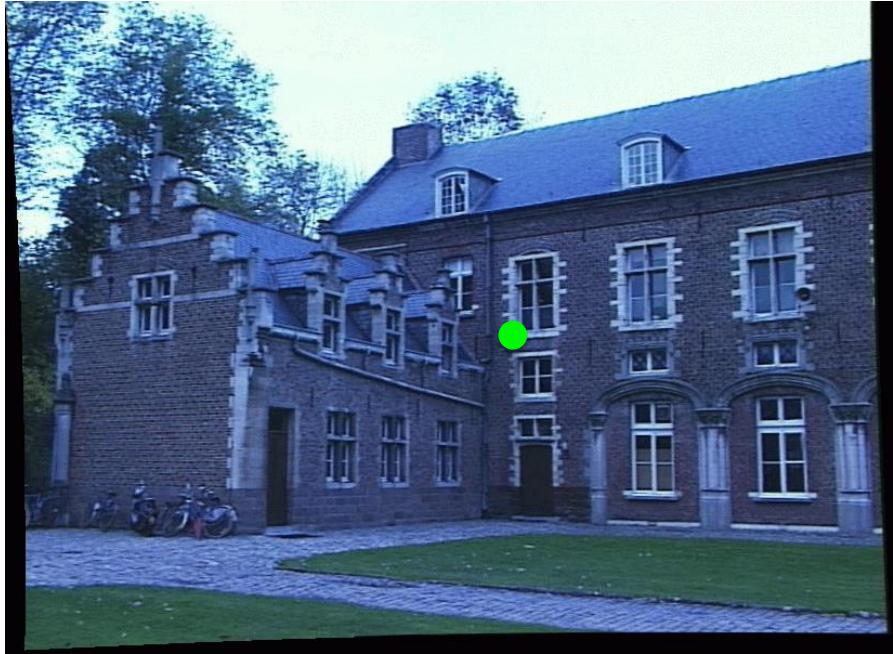
Left image



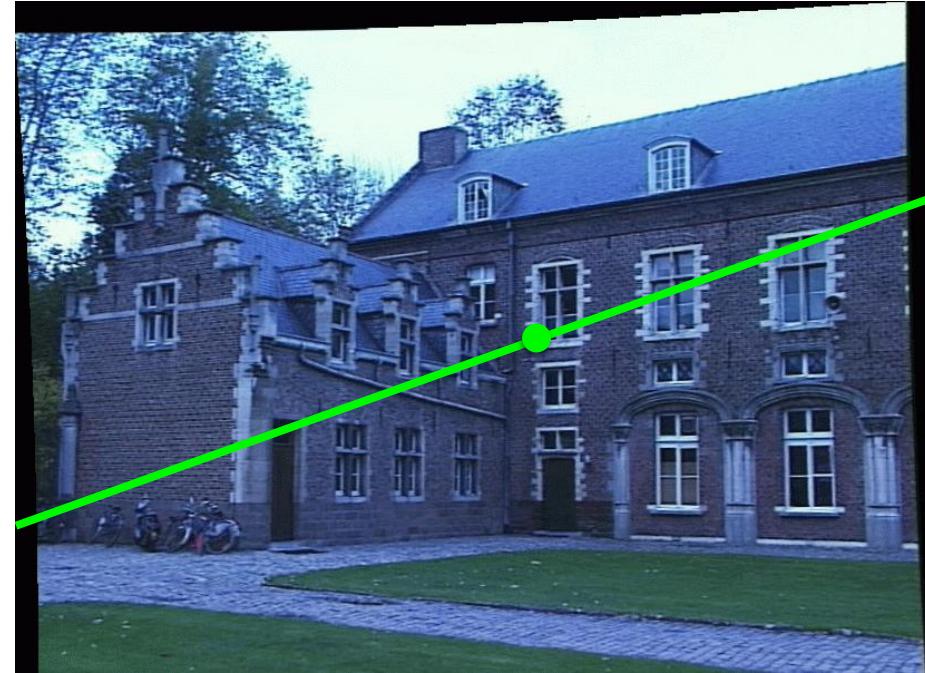
Right image

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

How would you reconstruct 3D points?



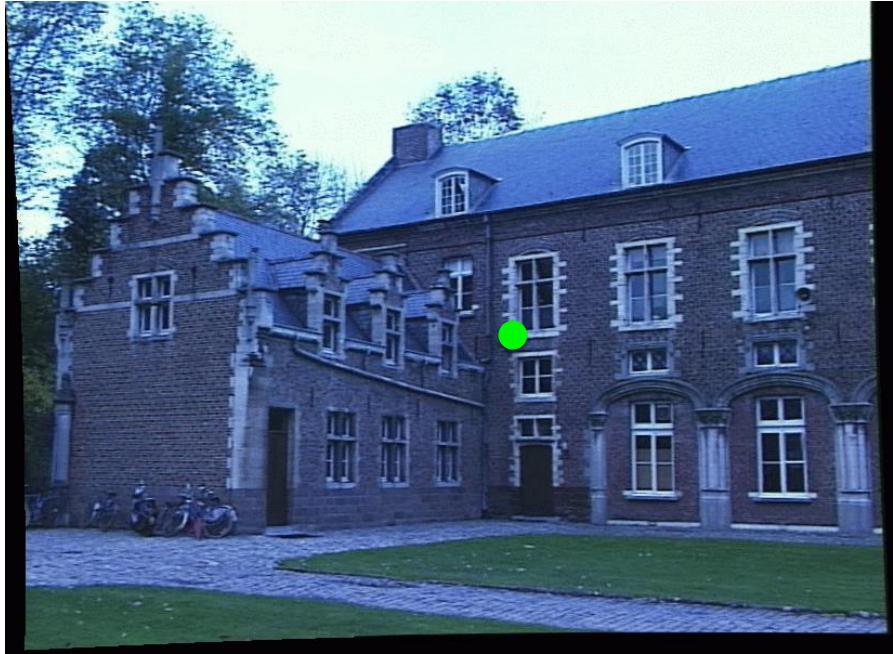
Left image



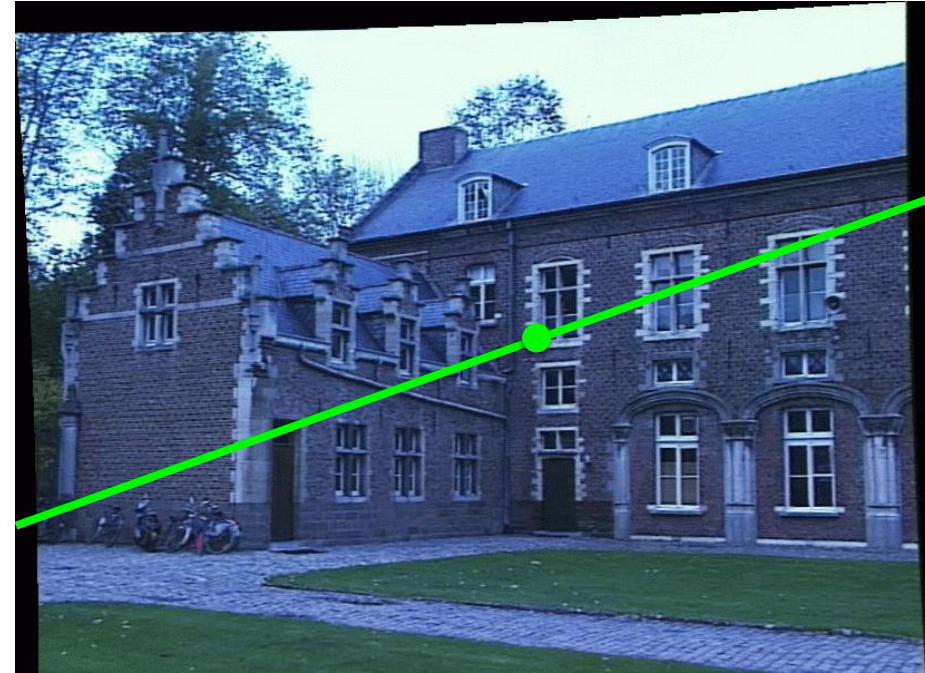
Right image

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?



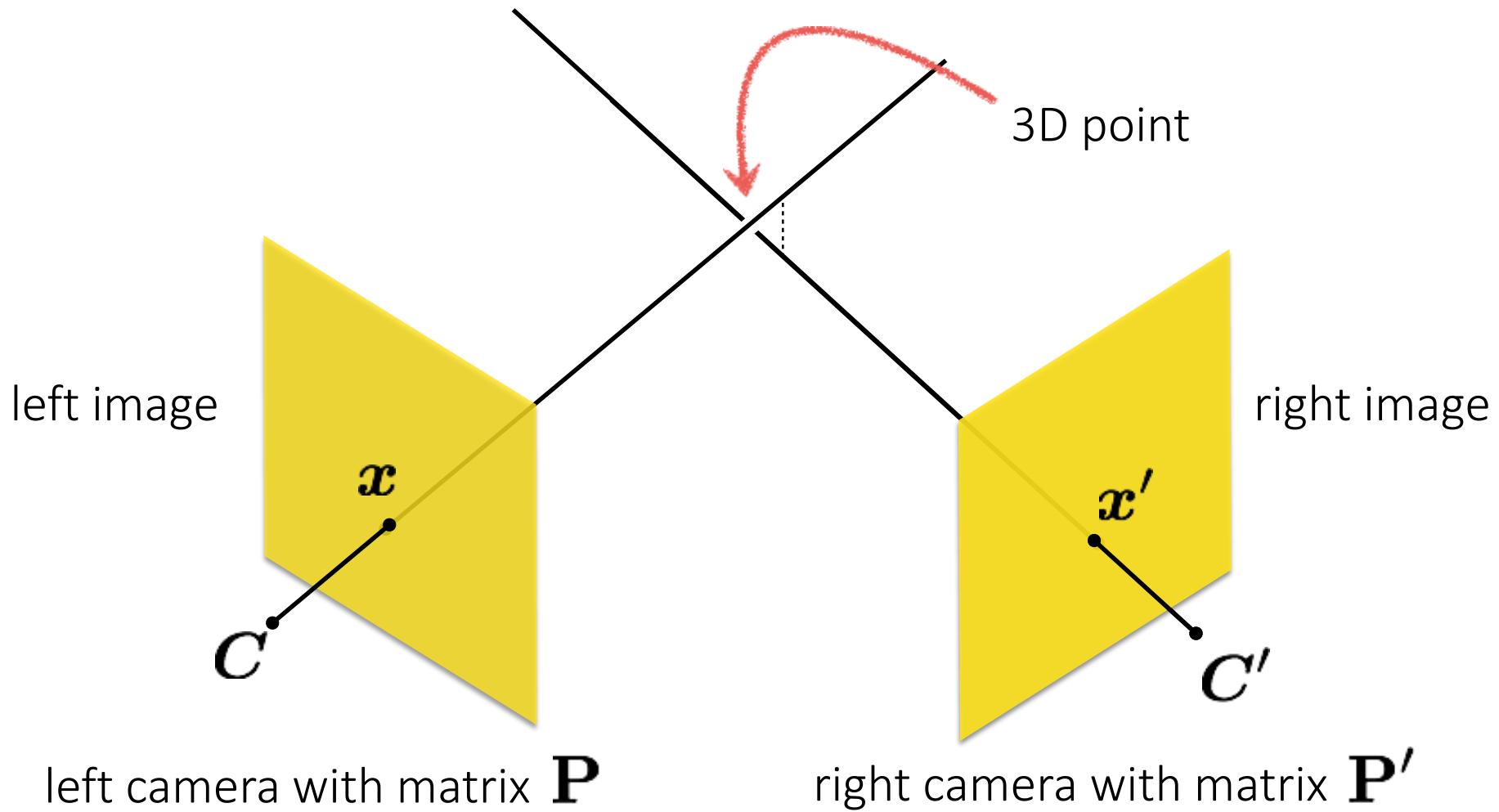
Left image



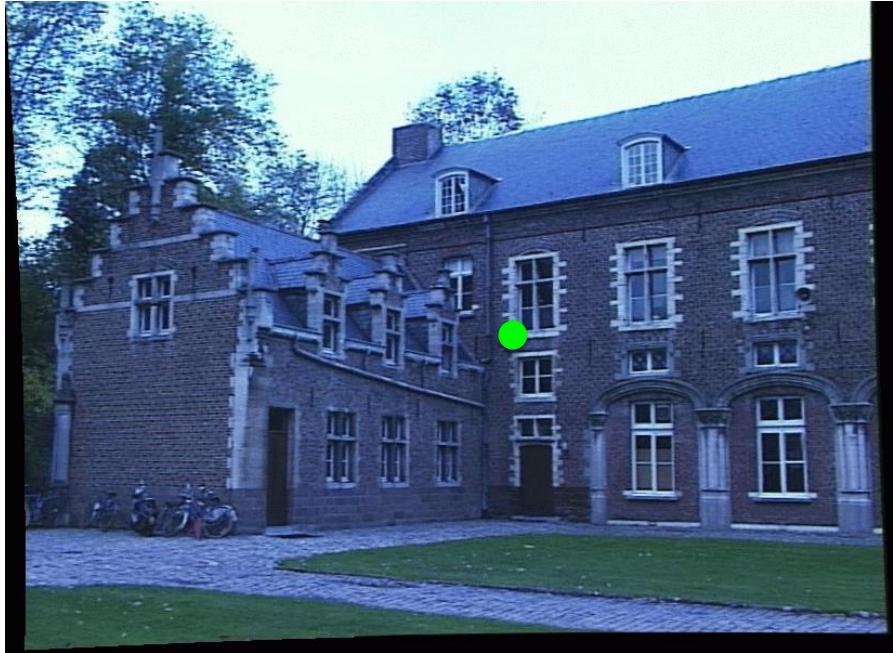
Right image

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?)

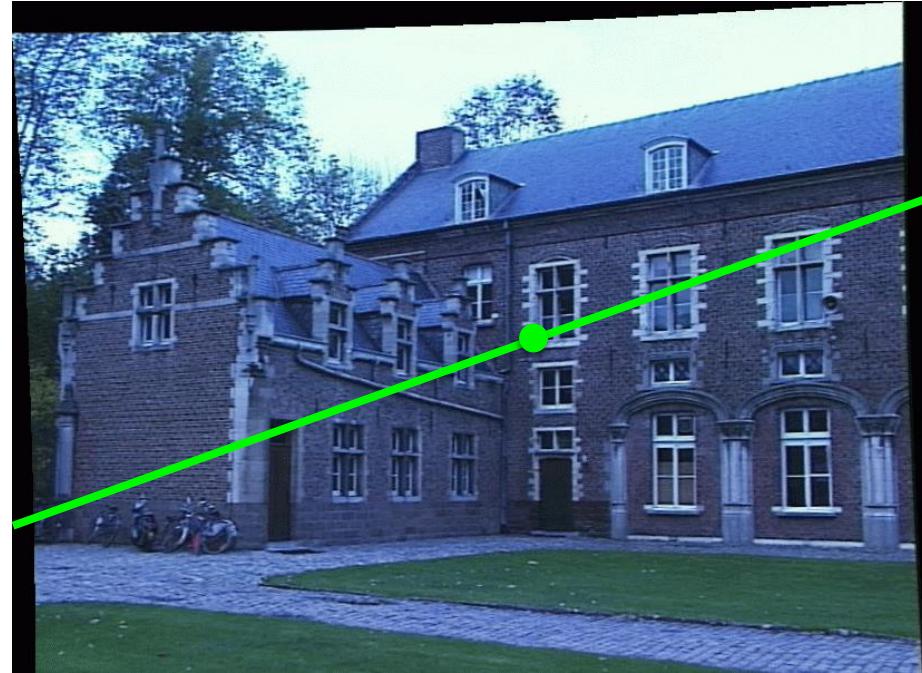
Triangulation



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?

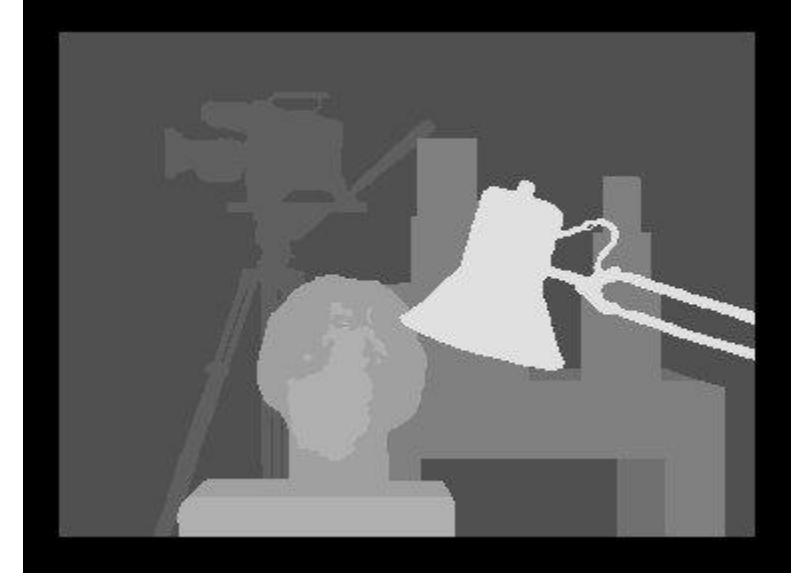
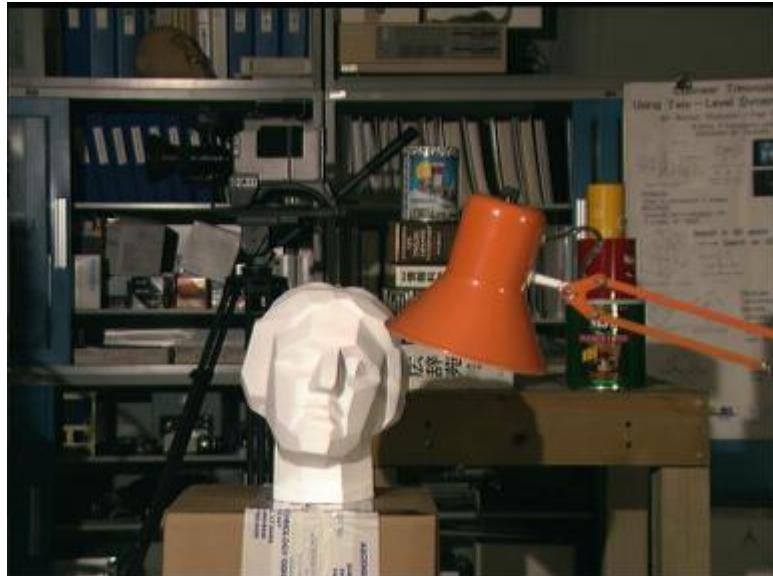






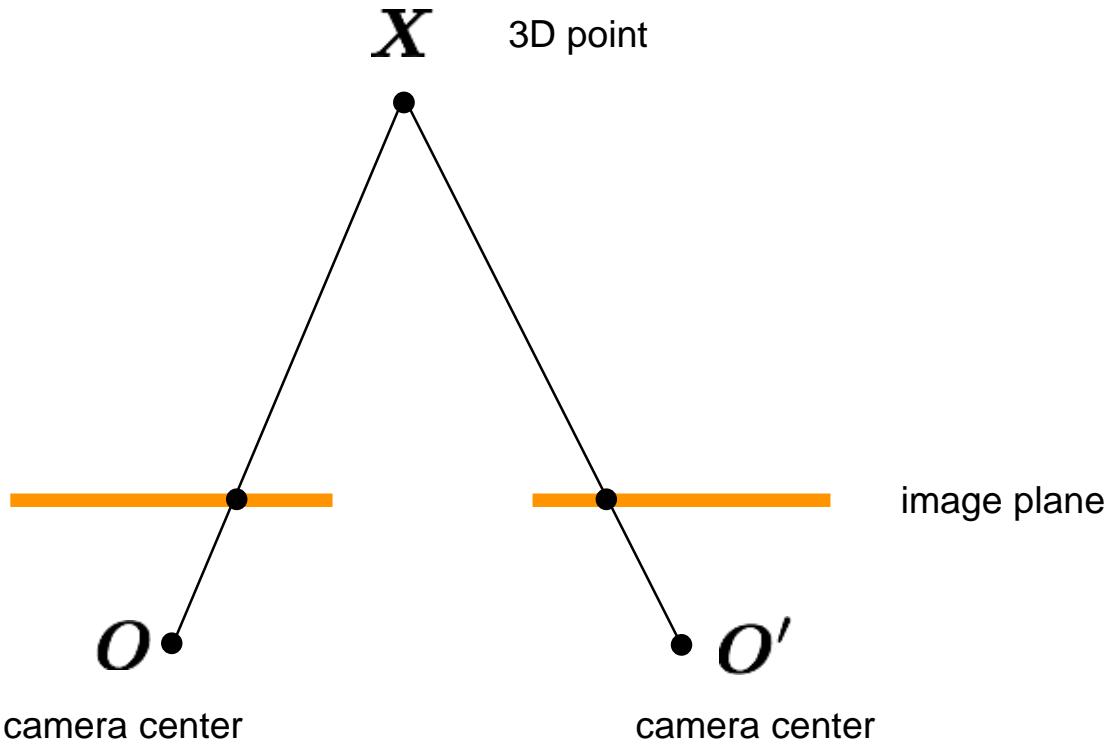
Objects that are close move more or less?

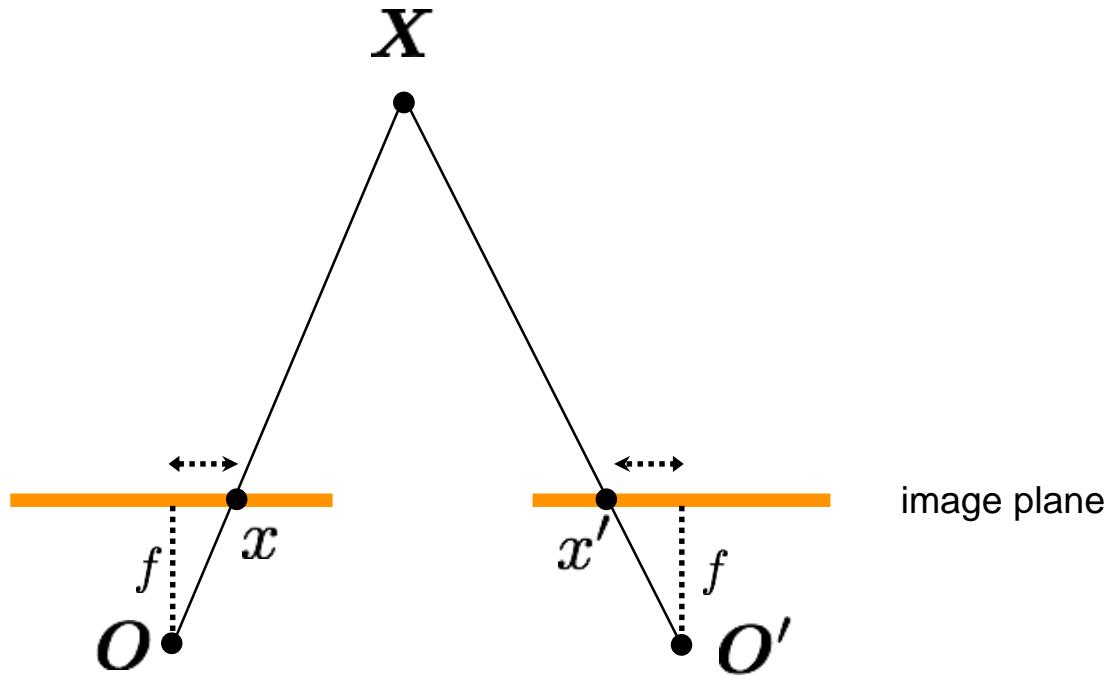
The amount of horizontal movement is
inversely proportional to ...

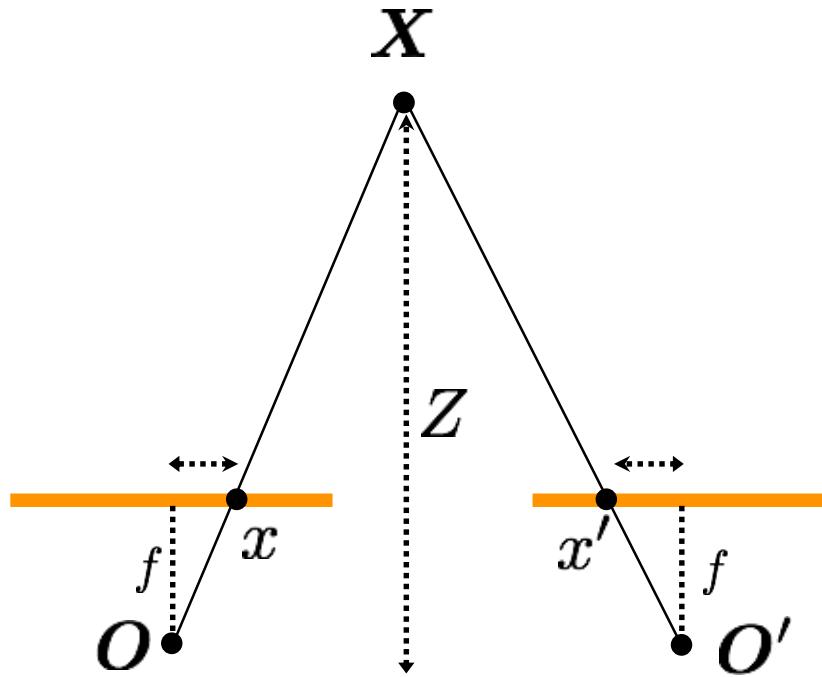


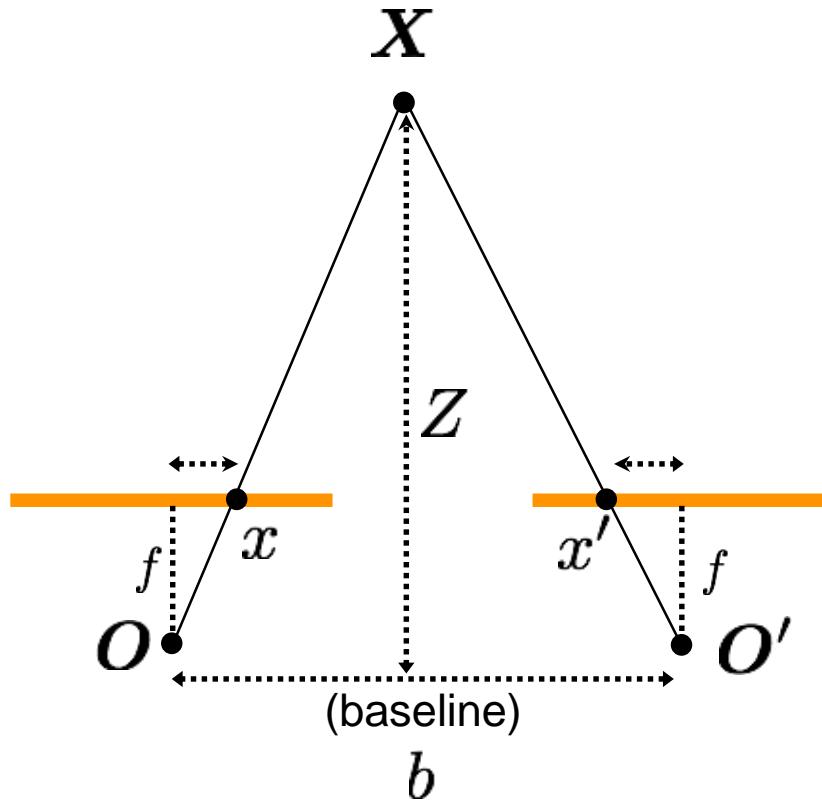
... the distance from the camera.

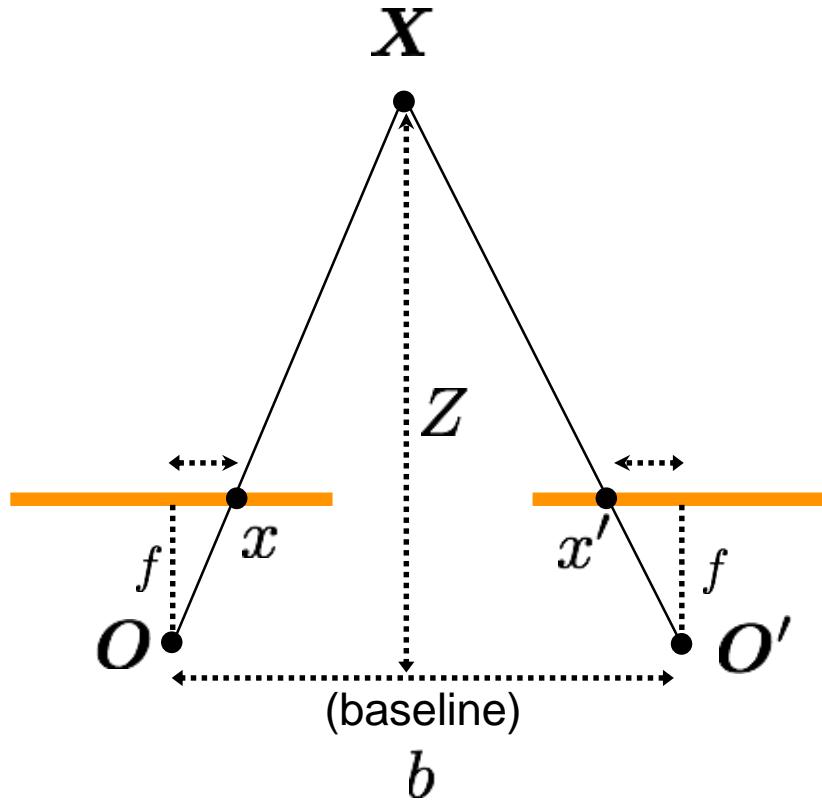
More formally...











Disparity

$$d = x - x'$$

inversely proportional
to depth

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

Real-time stereo sensing



Nomad robot searches for meteorites in Antarctica

<http://www.frc.ri.cmu.edu/projects/meteorobot/index.html>





What other vision system uses disparity for depth sensing?

Stereoscopes: A 19th Century Pastime



HON. ABRAHAM LINCOLN, President of United States.





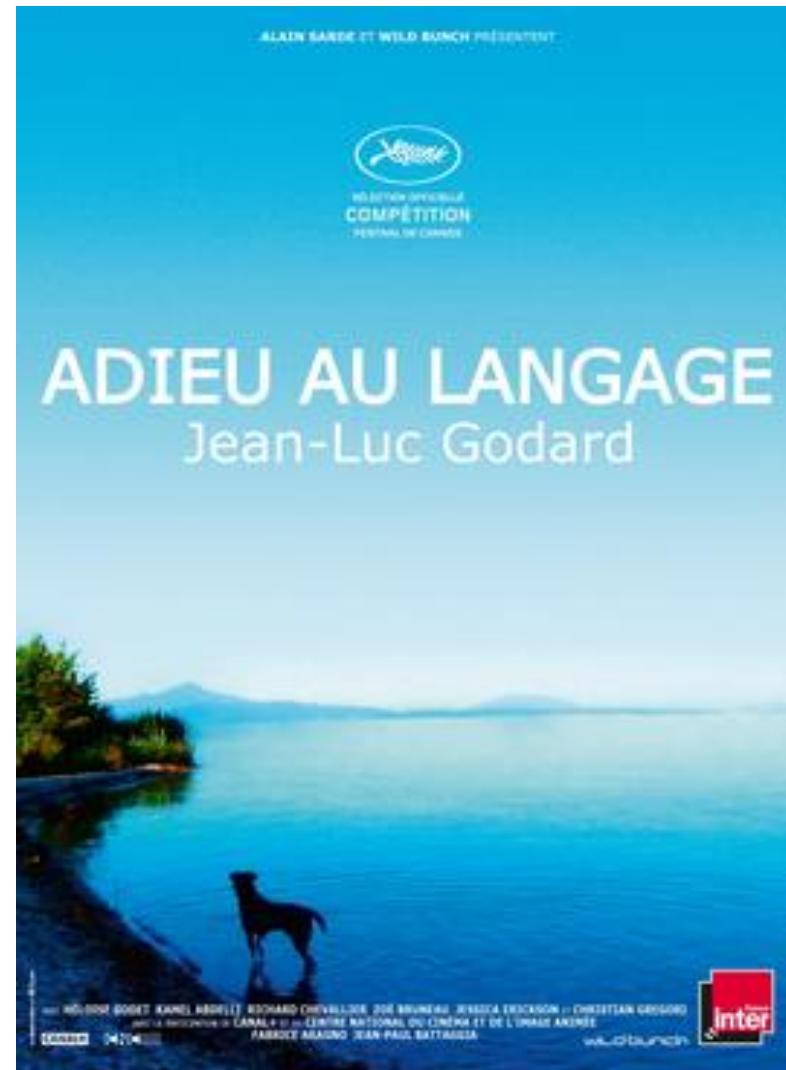
Public Library, Stereoscopic Looking Room, Chicago, by Phillips, 1923





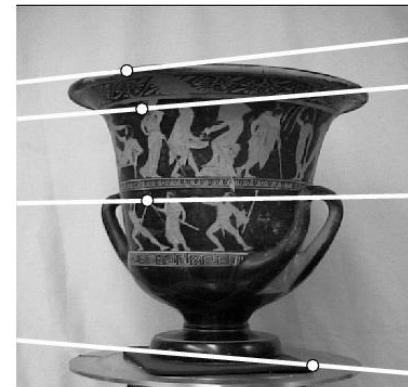
Mark Twain at Pool Table", no date, UCR Museum of Photography

This is how 3D movies work

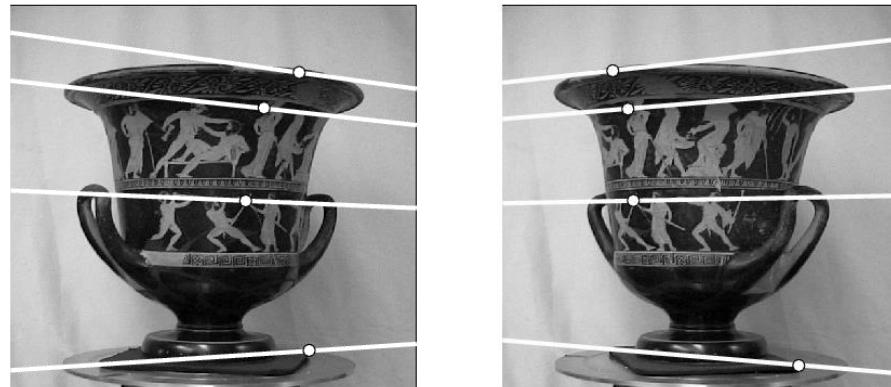


Is disparity the only depth cue
the human visual system uses?

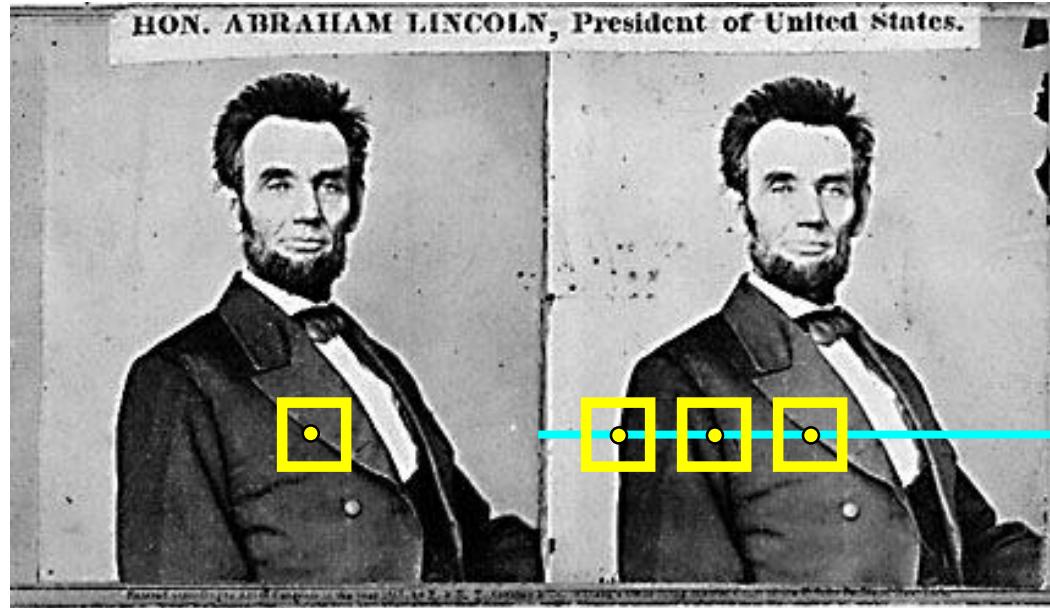
So can I compute depth from any two images of the same object?



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

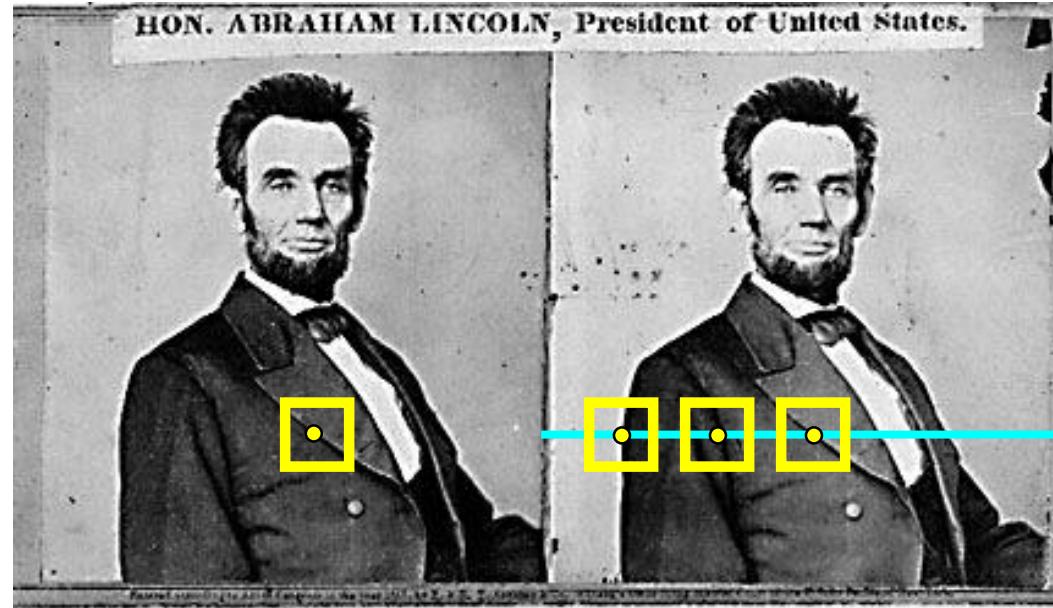


1. Need sufficient baseline
2. Images need to be ‘rectified’ first (make epipolar lines horizontal)

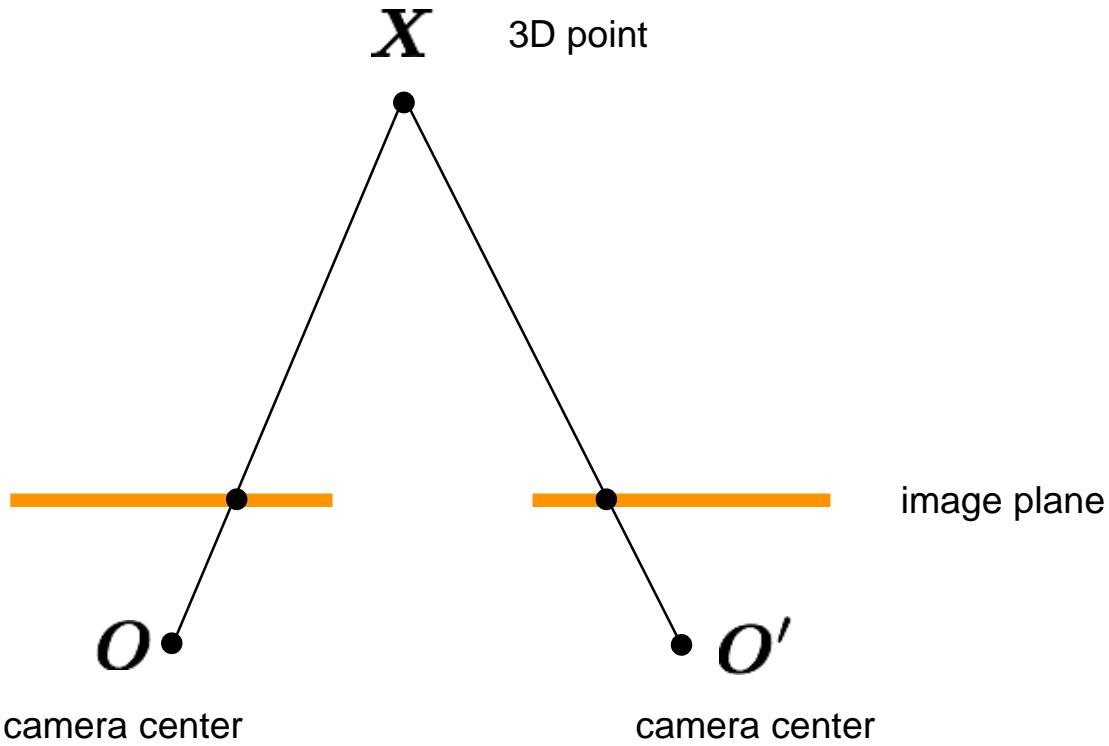


1. Rectify images
(make epipolar lines horizontal)
2. For each pixel
 - a. Find epipolar line
 - b. Scan line for best match
 - c. Compute depth from disparity

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



How can you make the epipolar lines horizontal?

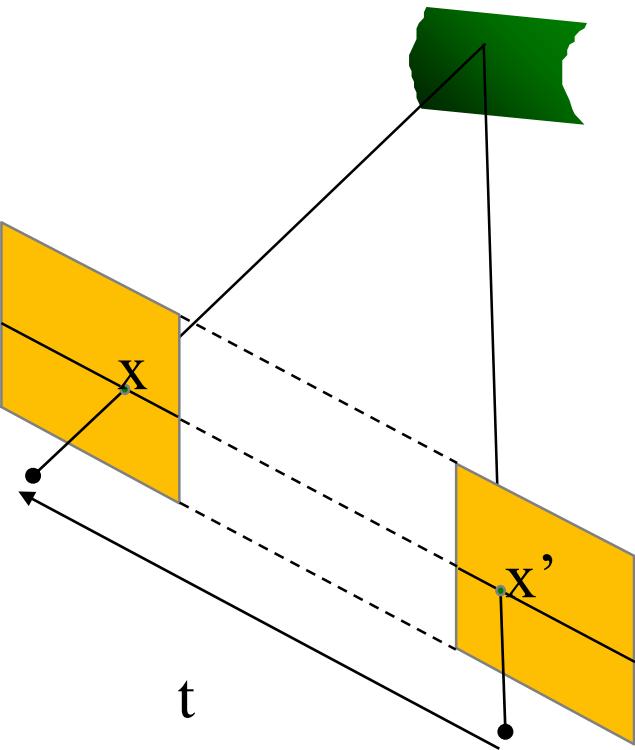


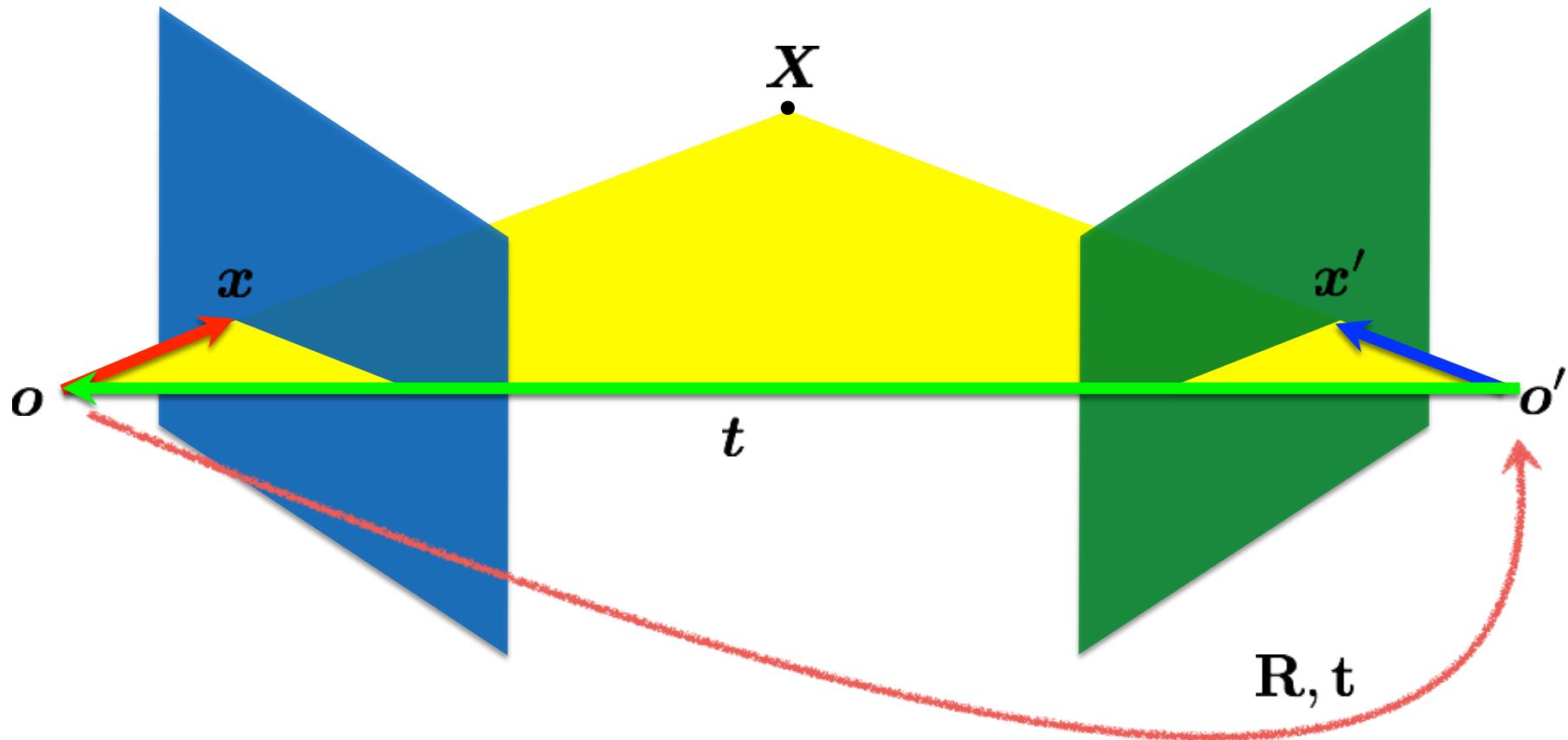
What's special about these two cameras?

When are epipolar lines horizontal?

When this relationship holds:

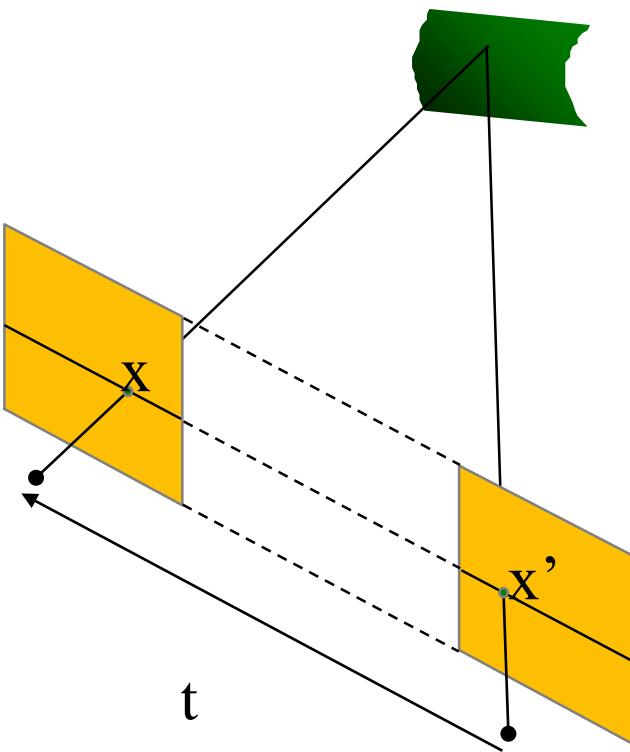
$$R = I \quad t = (T, 0, 0)$$





$$x' = R(x - t)$$

When are epipolar lines horizontal?



When this relationship holds:

$$R = I \quad t = (T, 0, 0)$$

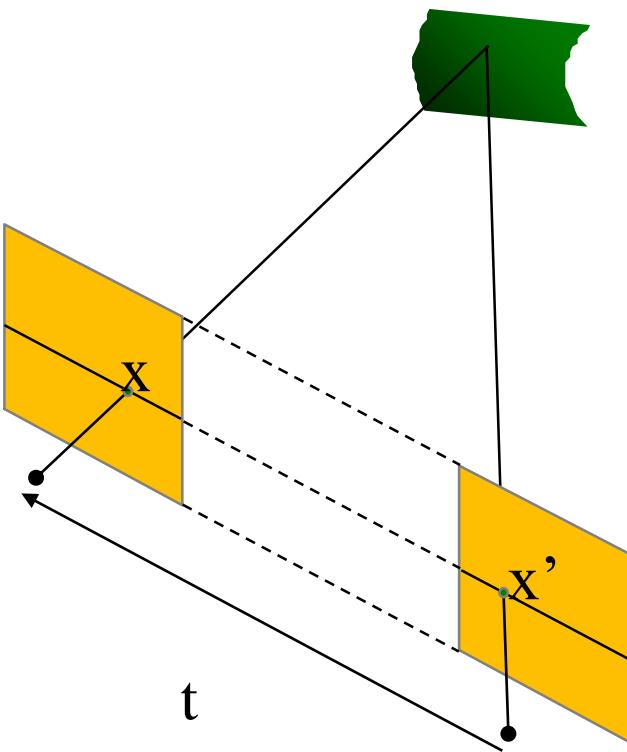
Let's try this out...

$$E = t \times R = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T \\ 0 & T & 0 \end{bmatrix}$$

This always has to hold for rectified images

$$x^T E x' = 0$$

When are epipolar lines horizontal?



Write out the constraint

$$(u \ v \ 1) \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T \\ 0 & T & 0 \end{bmatrix} \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = 0 \quad (u \ v \ 1) \begin{pmatrix} 0 \\ -T \\ Tv' \end{pmatrix} = 0$$

When this relationship holds:

$$R = I \quad t = (T, 0, 0)$$

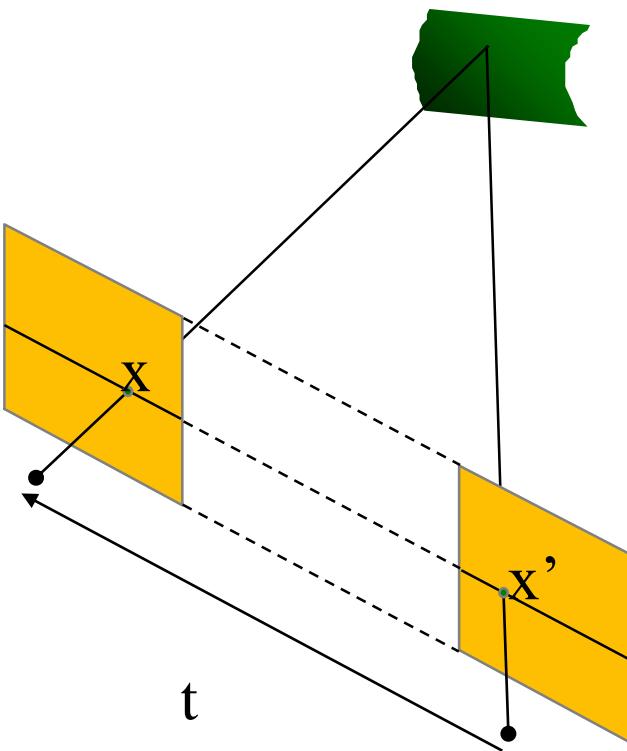
Let's try this out...

$$E = t \times R = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T \\ 0 & T & 0 \end{bmatrix}$$

This always has to hold for rectified images

$$x^T E x' = 0$$

When are epipolar lines horizontal?



Write out the constraint

$$(u \ v \ 1) \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T \\ 0 & T & 0 \end{bmatrix} \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = 0$$

$$(u \ v \ 1) \begin{pmatrix} 0 \\ -T \\ Tv' \end{pmatrix} = 0$$

When this relationship holds:

$$R = I \quad t = (T, 0, 0)$$

Let's try this out...

$$E = t \times R = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T \\ 0 & T & 0 \end{bmatrix}$$

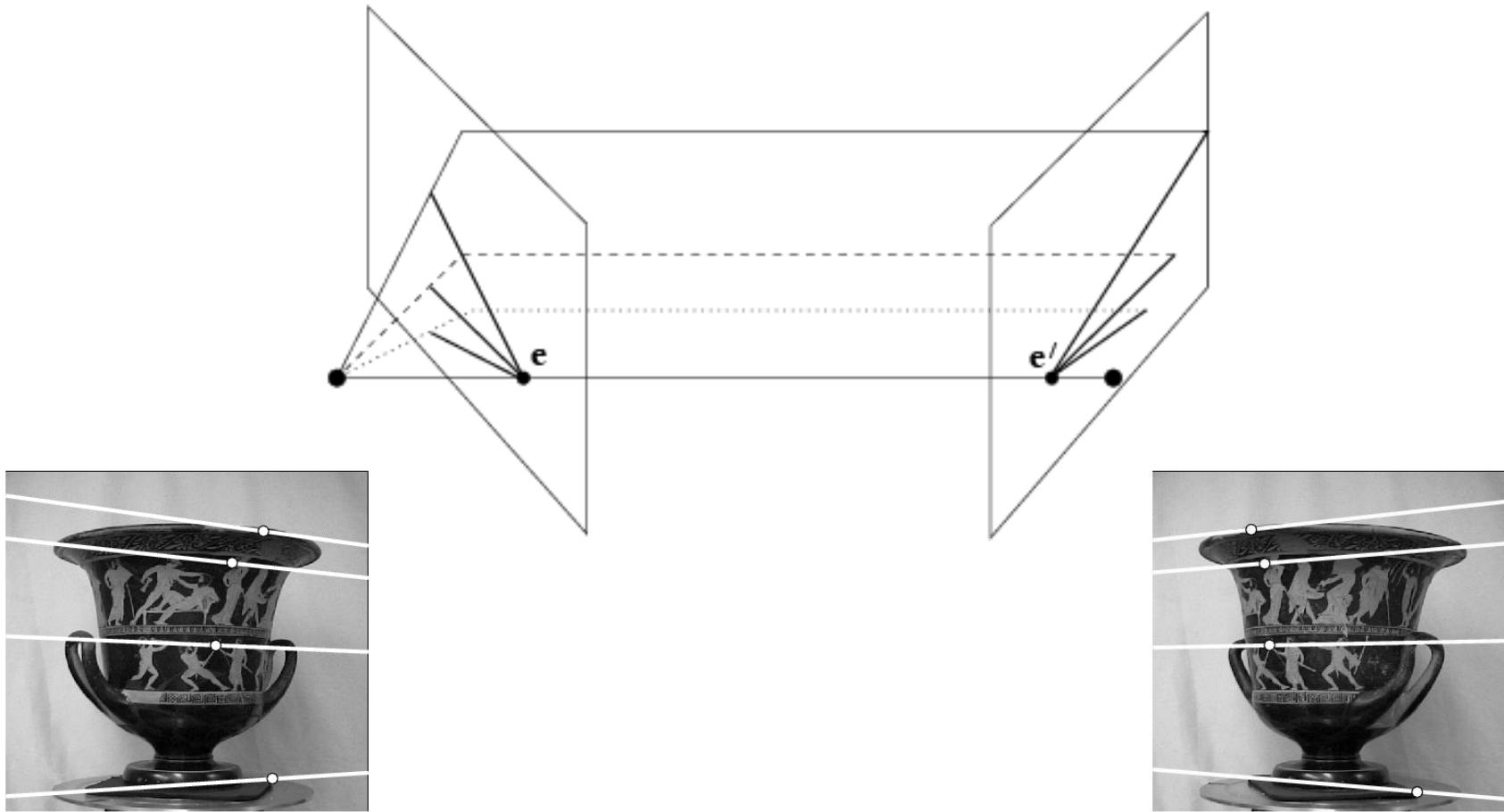
This always has to hold

$$x^T E x' = 0$$

The image of a 3D point will always be on the same horizontal line

$$Tv = Tv'$$

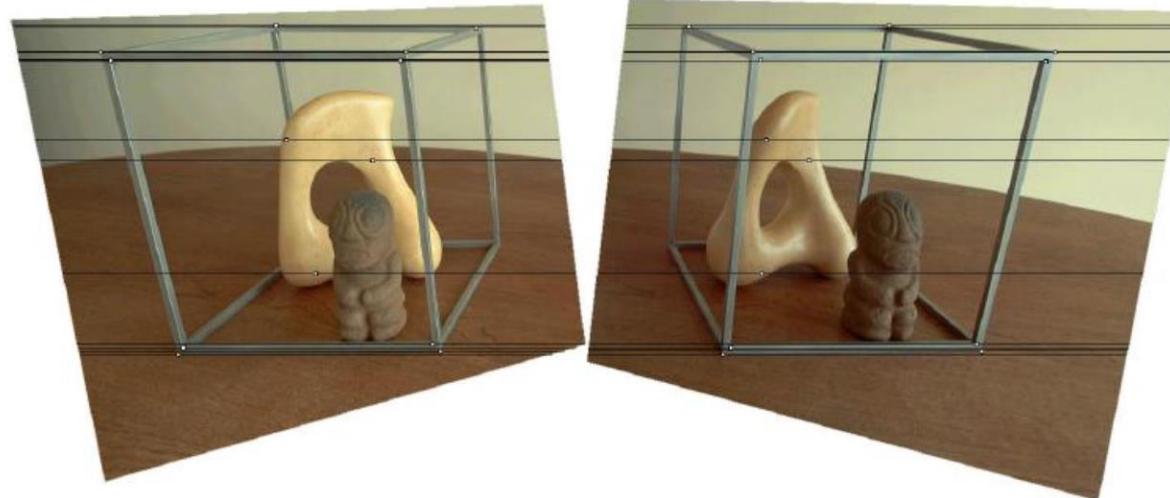
y coordinate is always the same!



It's hard to make the image planes exactly parallel

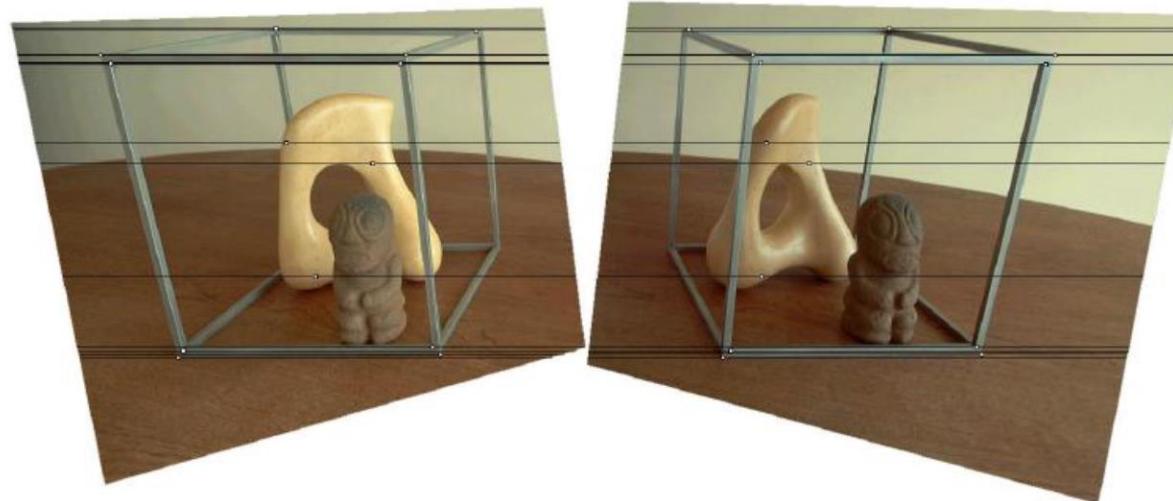


How can you make the epipolar lines horizontal?

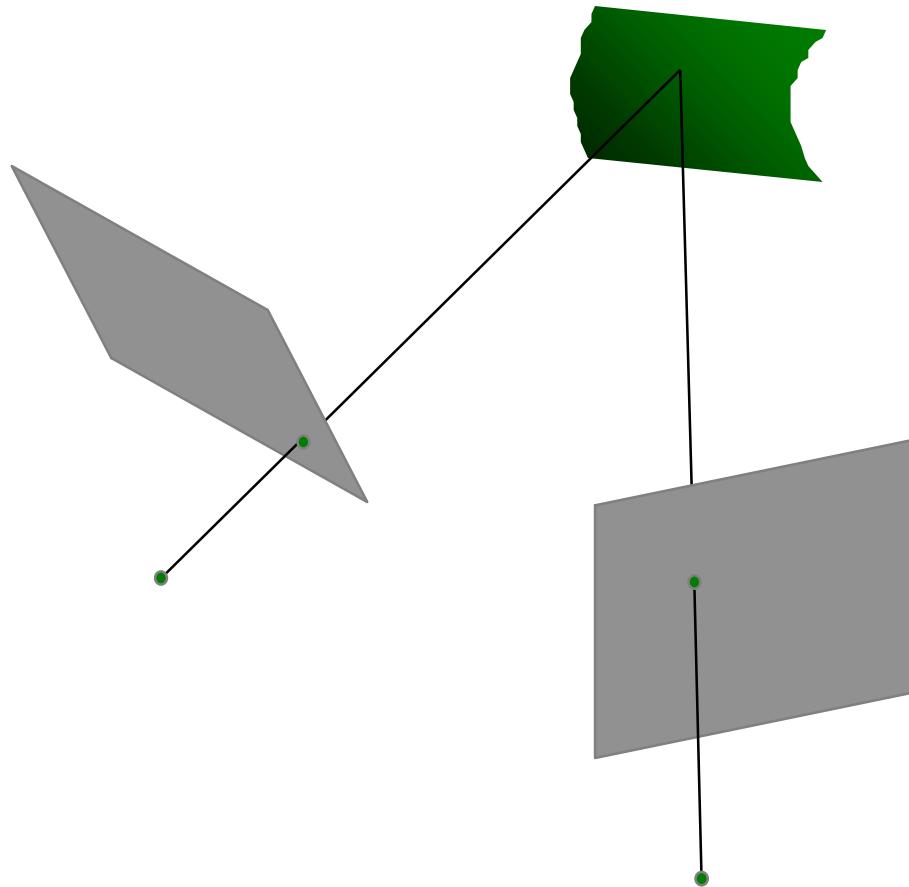




Use stereo rectification?



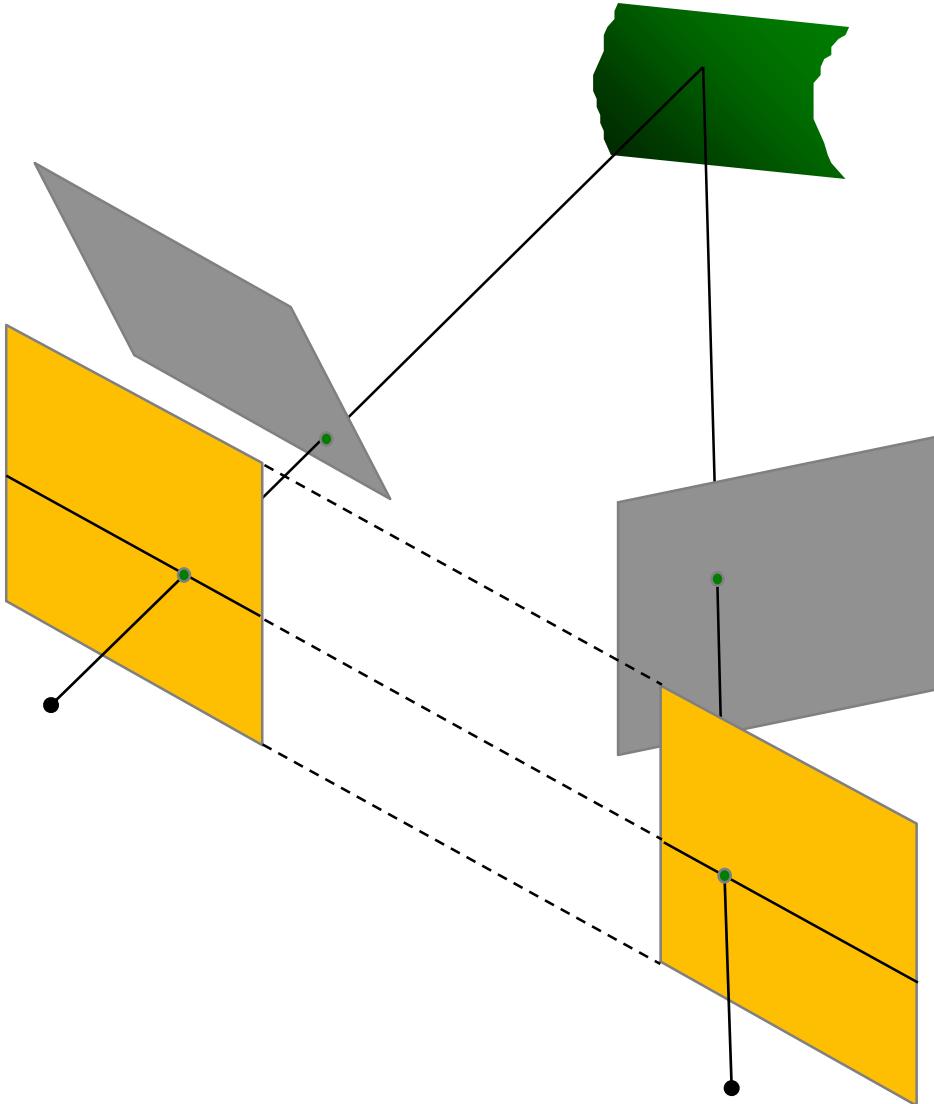
What is stereo rectification?



What is stereo rectification?

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

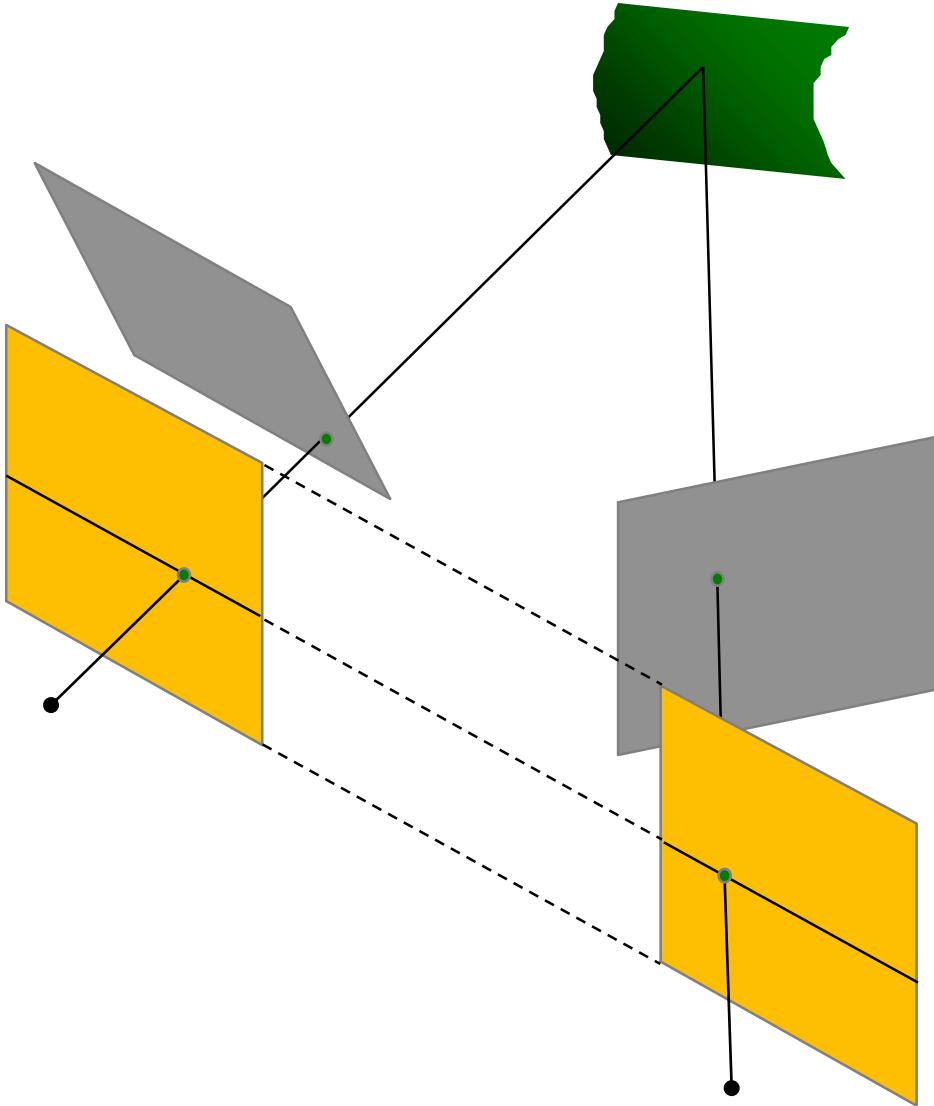
How can you do this?



What is stereo rectification?

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

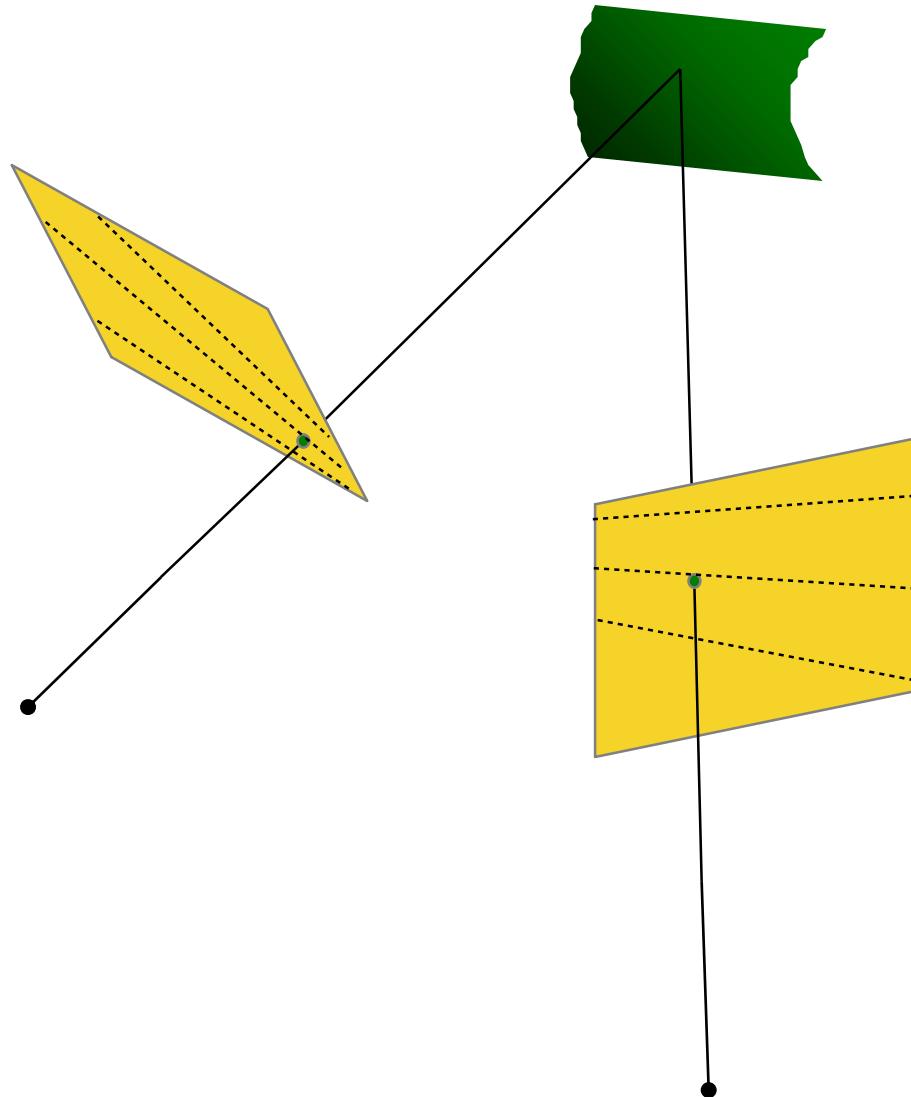
Need two homographies (3×3 transform), one for each input image reprojection



Stereo Rectification

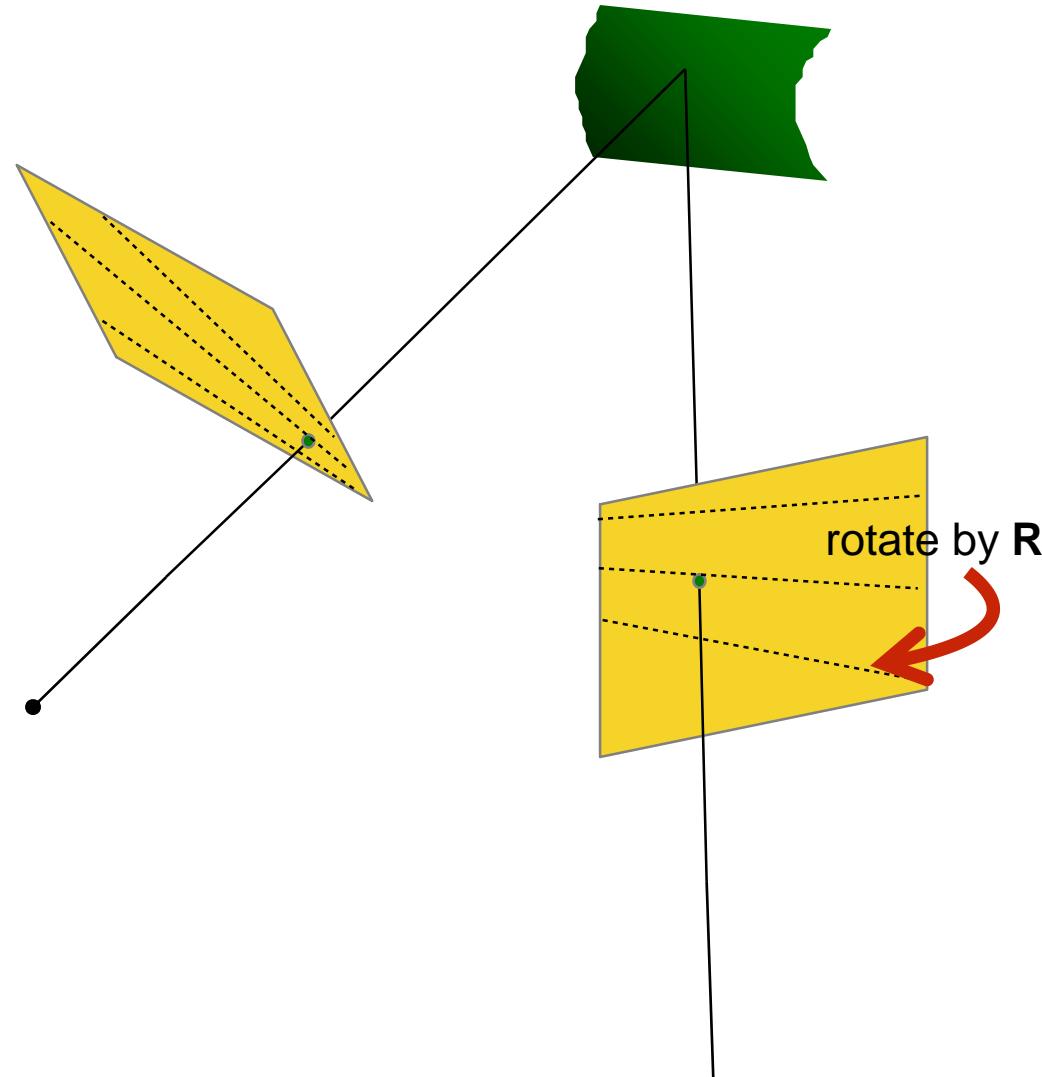
1. **Rotate** the right camera by **R**
(aligns camera coordinate system orientation only)
2. Rotate (**rectify**) the left camera so that the epipole
is at infinity
3. Rotate (**rectify**) the right camera so that the
epipole is at infinity
4. Adjust the **scale**

Stereo Rectification:



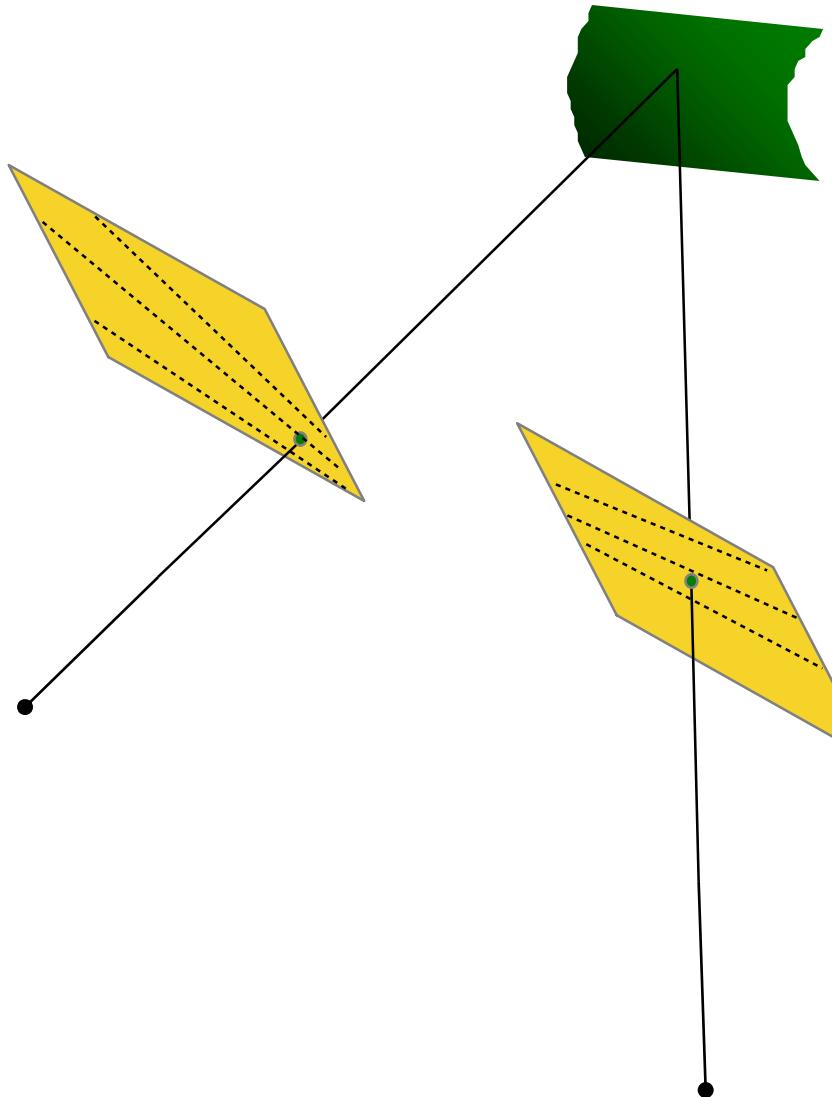
1. Compute \mathbf{E} to get \mathbf{R}
2. Rotate right image by \mathbf{R}
3. Rotate both images by \mathbf{R}_{rect}
4. Scale both images by \mathbf{H}

Stereo Rectification:



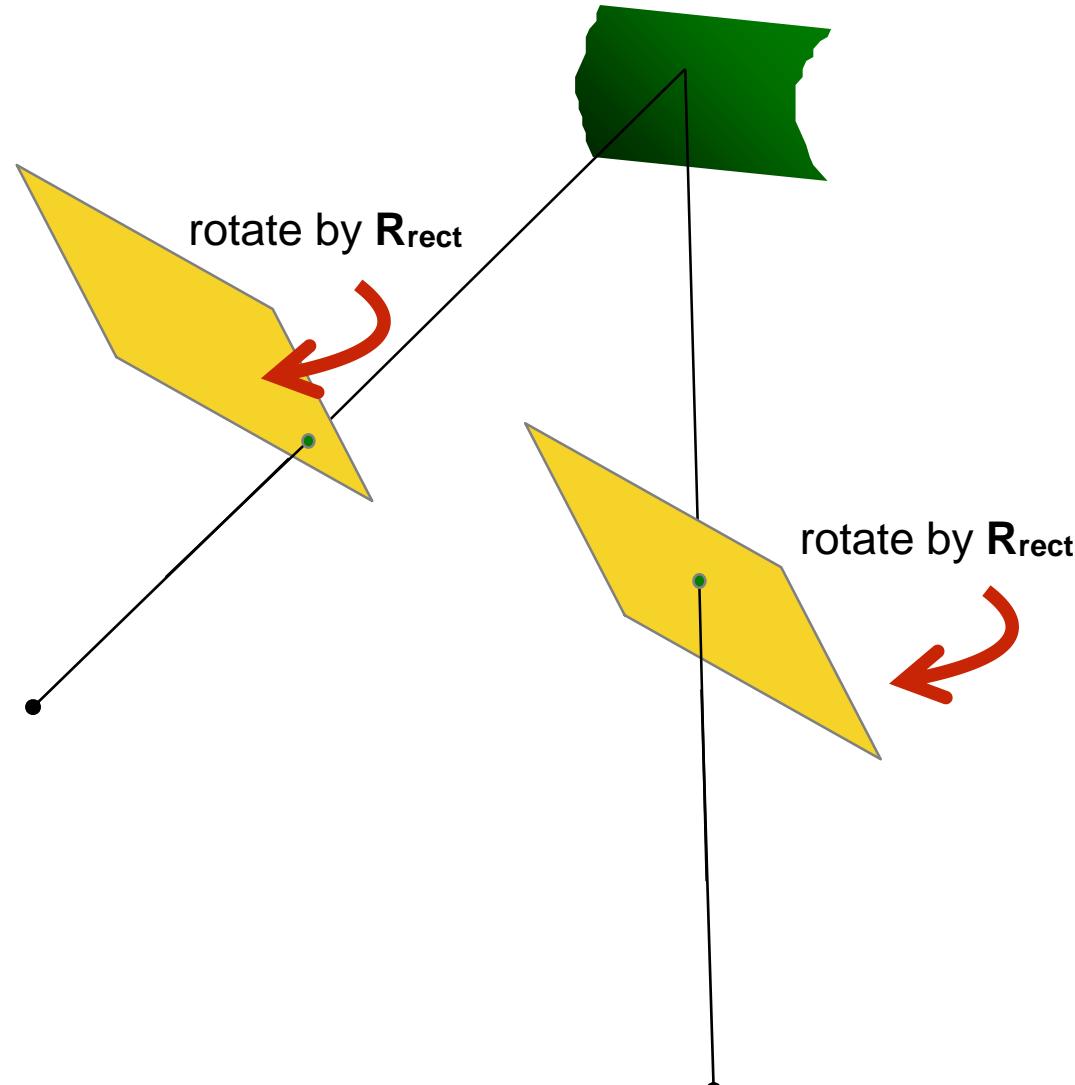
1. Compute \mathbf{E} to get \mathbf{R}
2. Rotate right image by \mathbf{R}
3. Rotate both images by \mathbf{R}_{rect}
4. Scale both images by \mathbf{H}

Stereo Rectification:



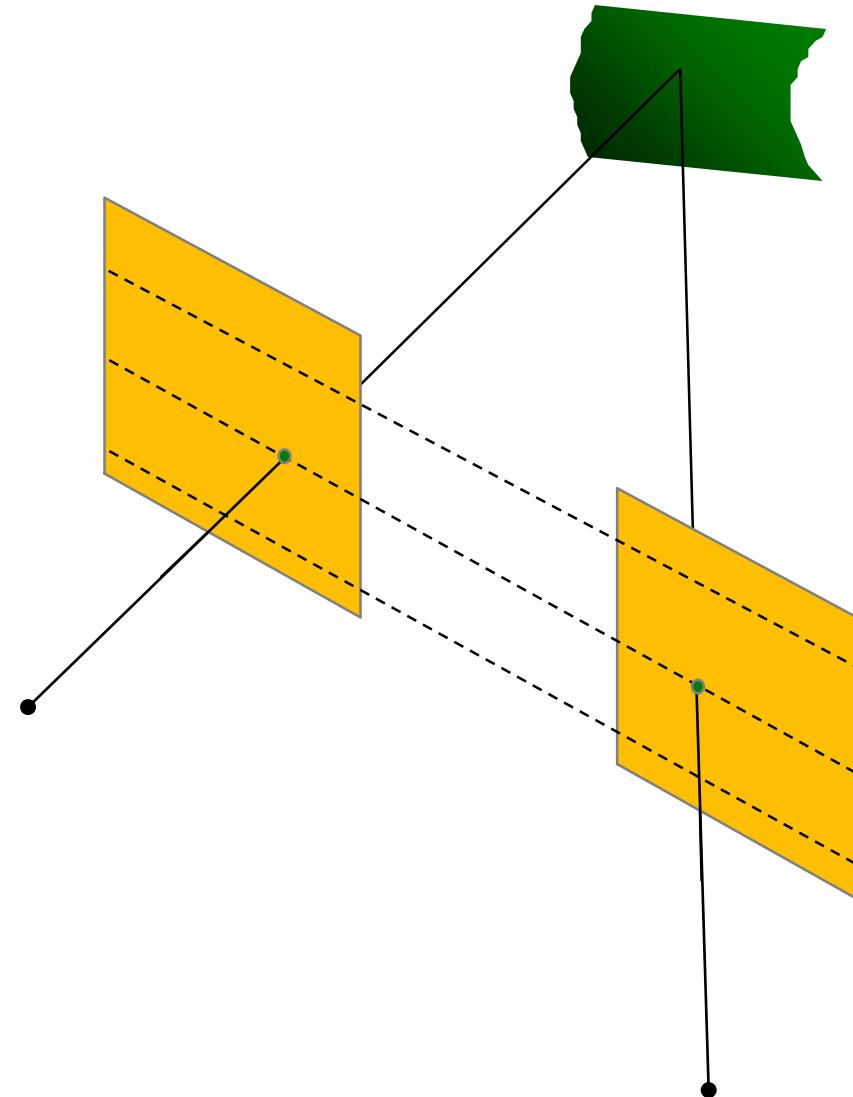
1. Compute \mathbf{E} to get \mathbf{R}
2. Rotate right image by \mathbf{R}
3. Rotate both images by \mathbf{R}_{rect}
4. Scale both images by \mathbf{H}

Stereo Rectification:



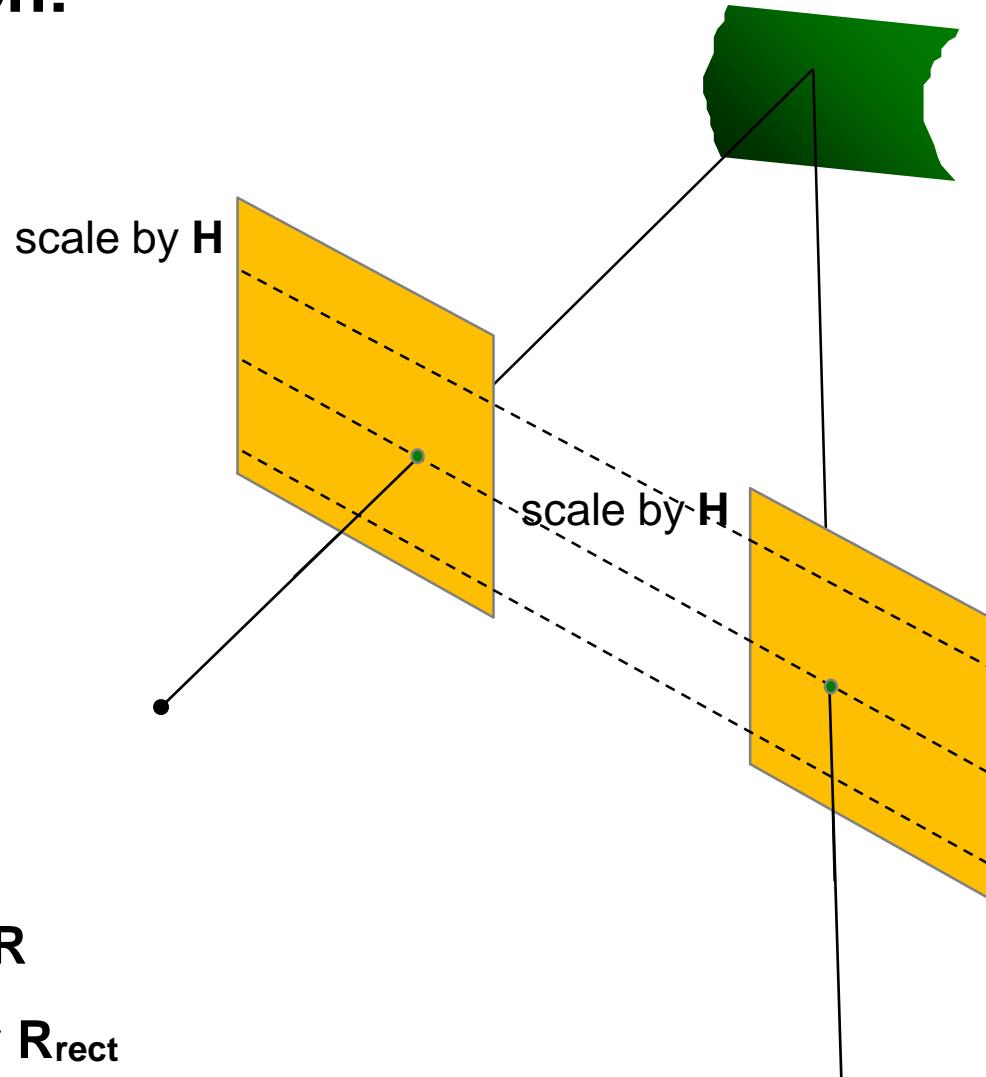
1. Compute \mathbf{E} to get \mathbf{R}
2. Rotate right image by \mathbf{R}
3. Rotate both images by \mathbf{R}_{rect}
4. Scale both images by \mathbf{H}

Stereo Rectification:



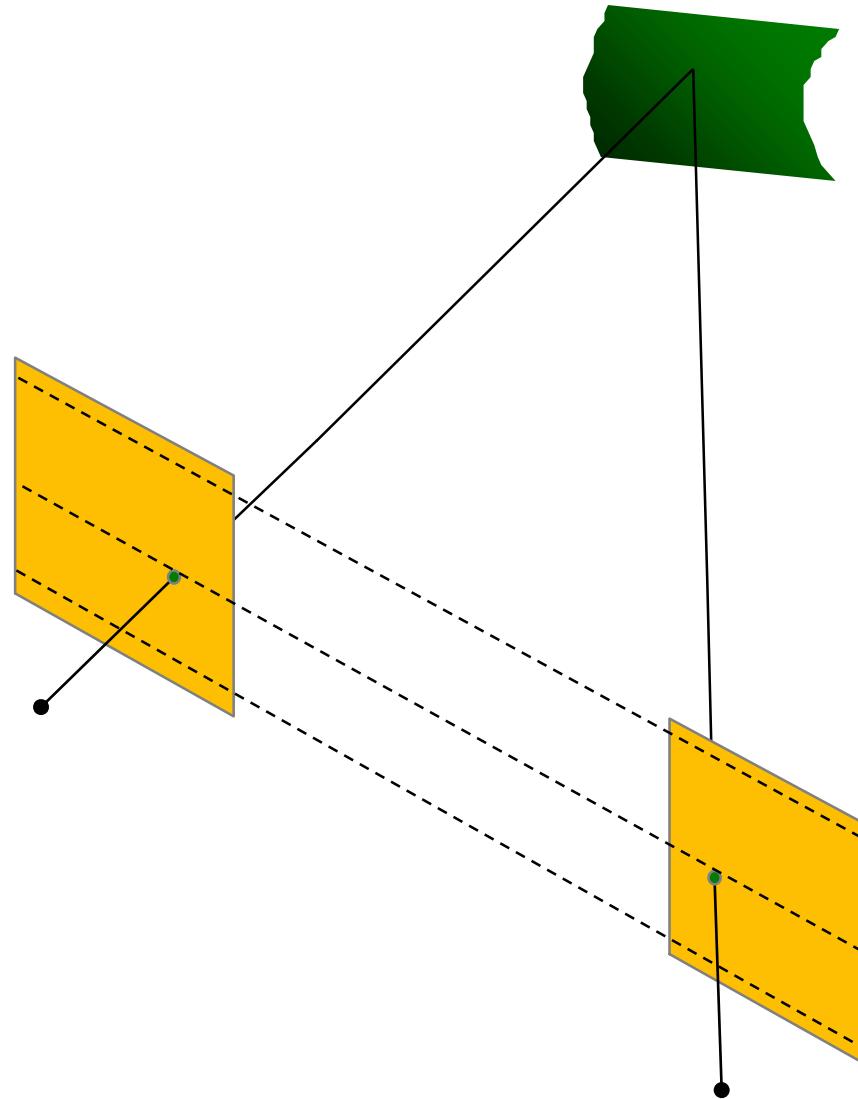
1. Compute \mathbf{E} to get \mathbf{R}
2. Rotate right image by \mathbf{R}
3. Rotate both images by \mathbf{R}_{rect}
4. Scale both images by \mathbf{H}

Stereo Rectification:



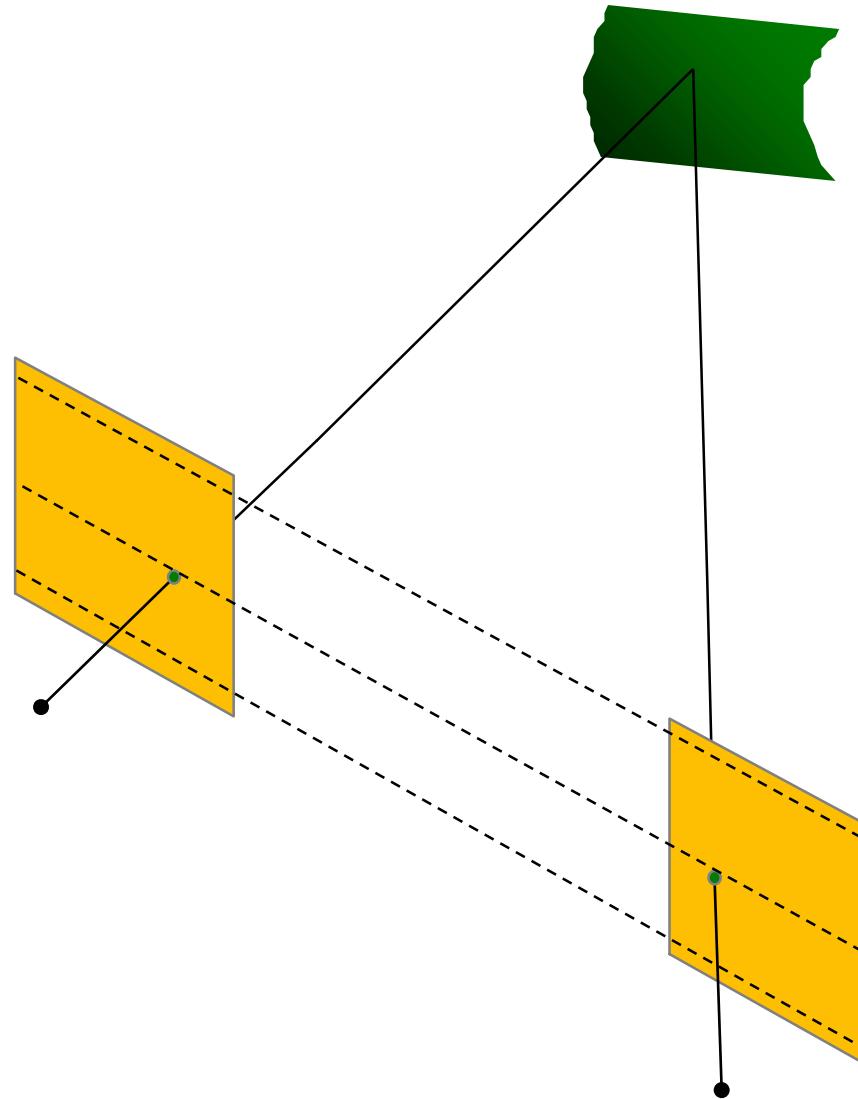
1. Compute \mathbf{E} to get \mathbf{R}
2. Rotate right image by \mathbf{R}
3. Rotate both images by \mathbf{R}_{rect}
4. Scale both images by \mathbf{H}

Stereo Rectification:



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

Stereo Rectification:



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

Step 1: Compute E to get R

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

We get FOUR solutions:

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

We get FOUR solutions:

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

$$\mathbf{T}_1 = U_3$$

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

$$\mathbf{T}_2 = -U_3$$

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

$$\mathbf{T}_2 = -U_3$$

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

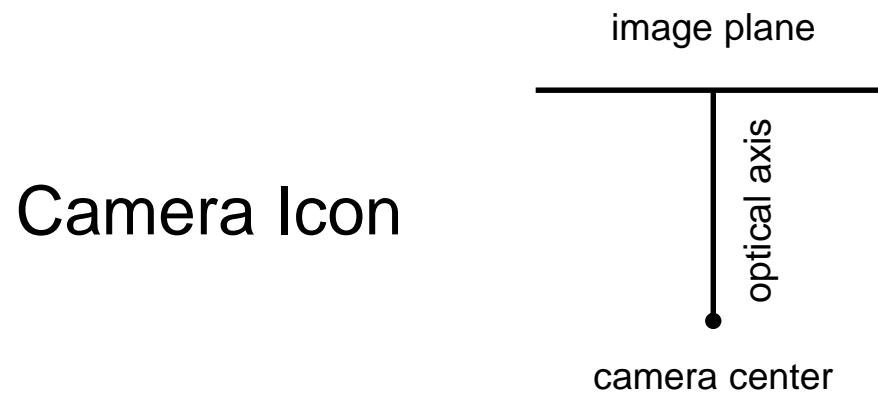
$$\mathbf{T}_1 = U_3$$

Which one do we choose?

Compute determinant of R, valid solution must be equal to 1
(note: $\det(\mathbf{R}) = -1$ means rotation and reflection)

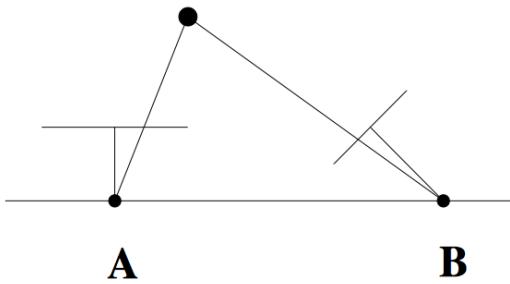
Compute 3D point using triangulation, valid solution has positive Z value
(Note: negative Z means point is behind the camera)

Let's visualize the four configurations...

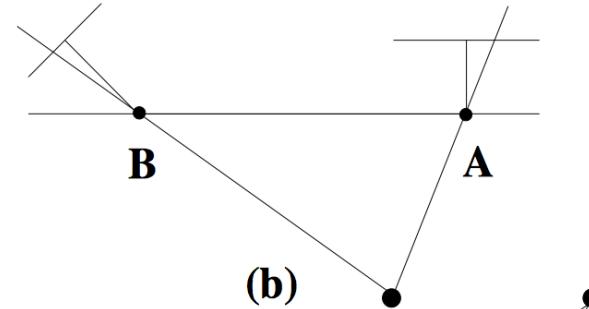


Find the configuration where the points are in front of both cameras

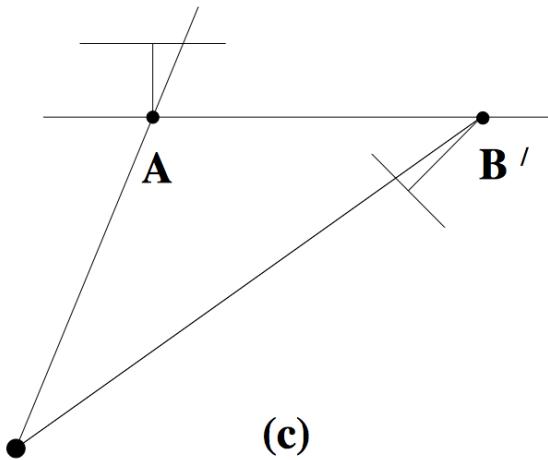
Find the configuration where the points is in front of both cameras



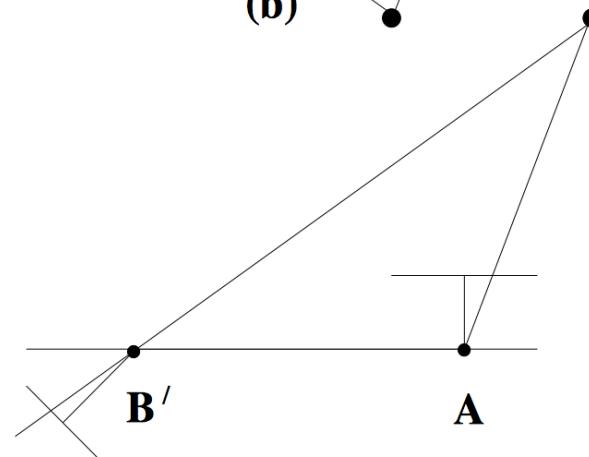
(a)



(b)

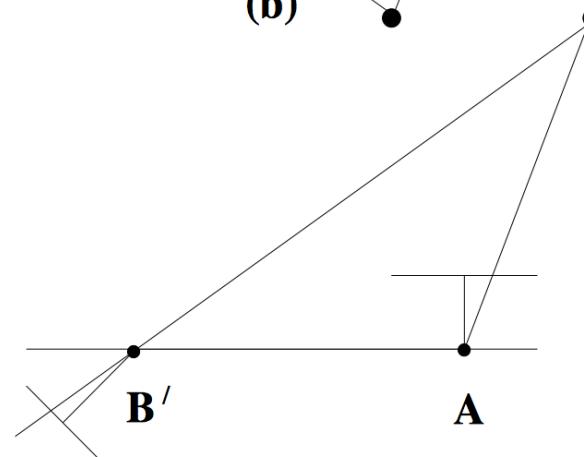
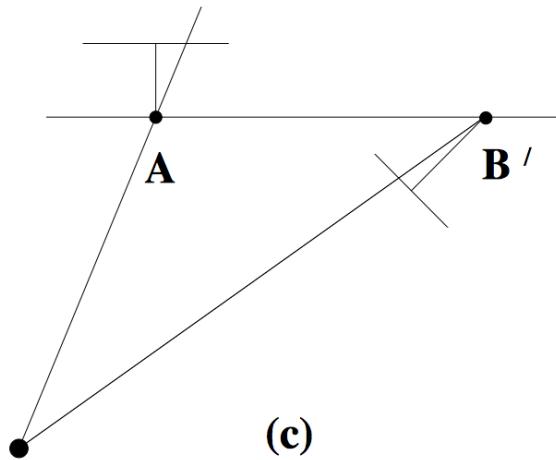
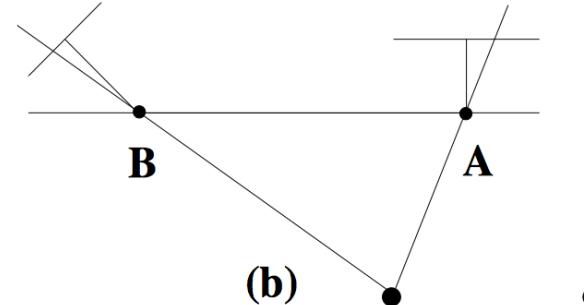
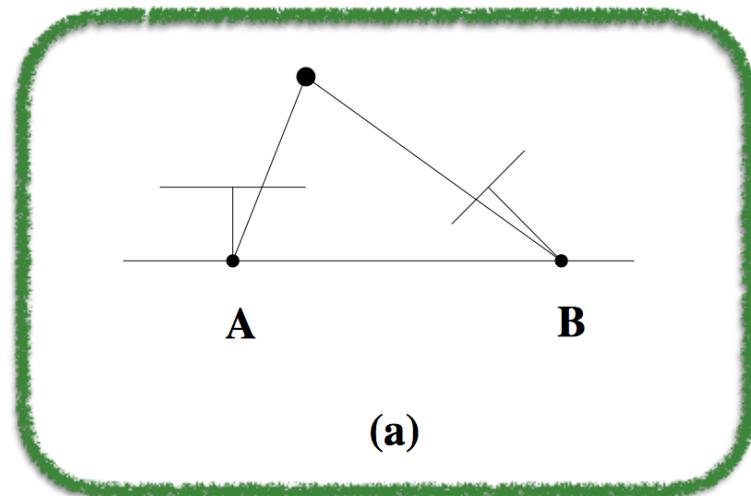


(c)

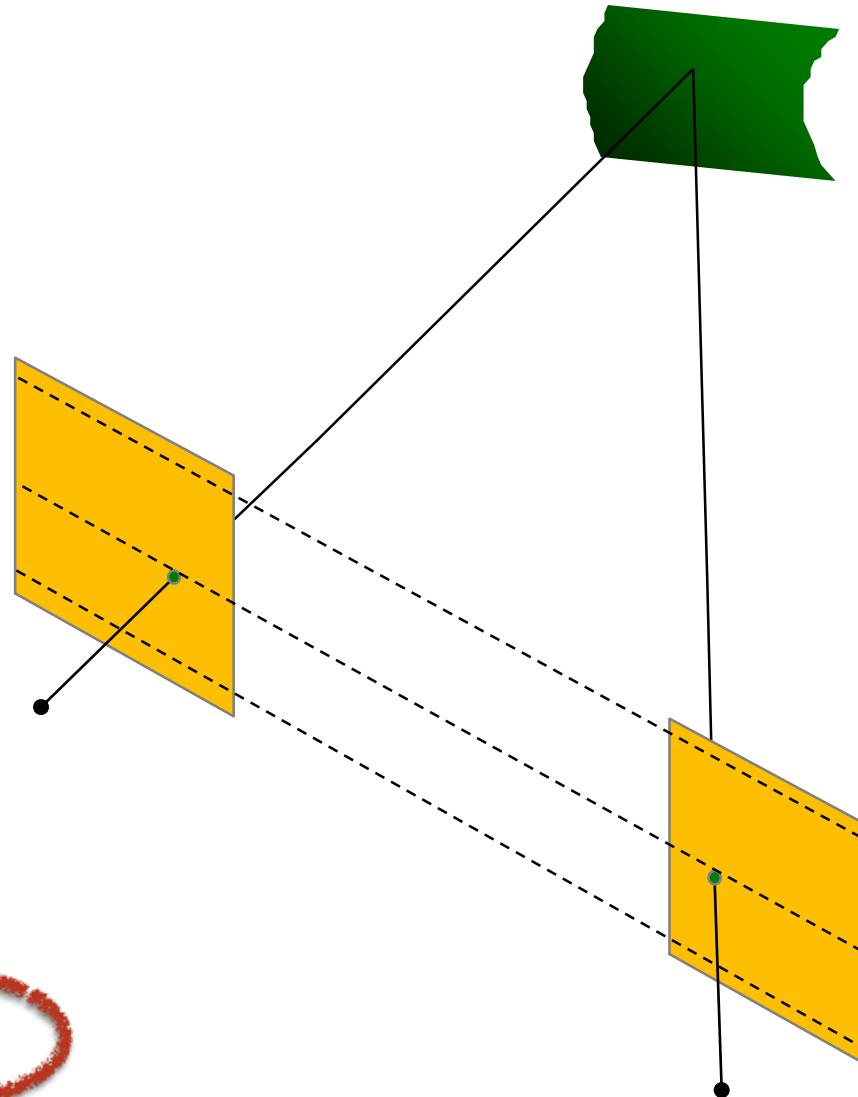


(d)

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



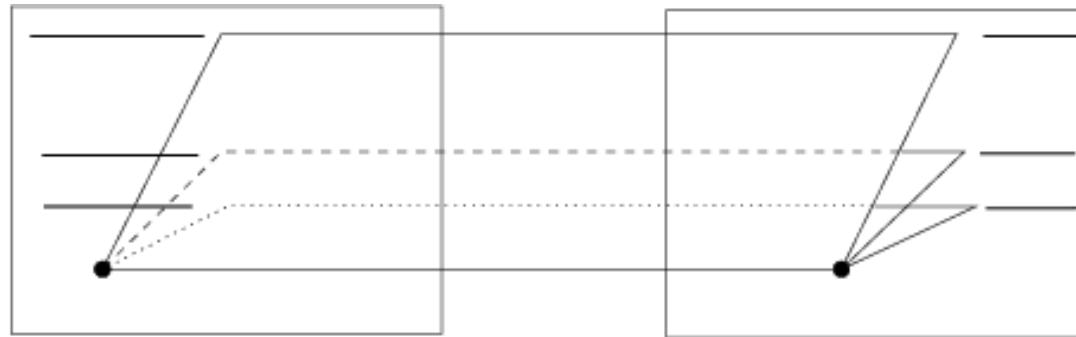
Stereo Rectification:



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

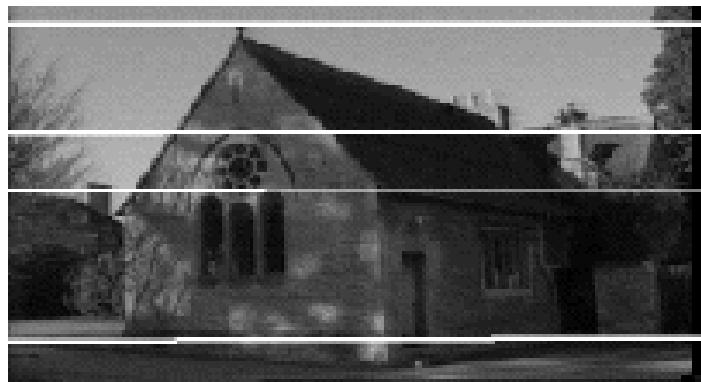
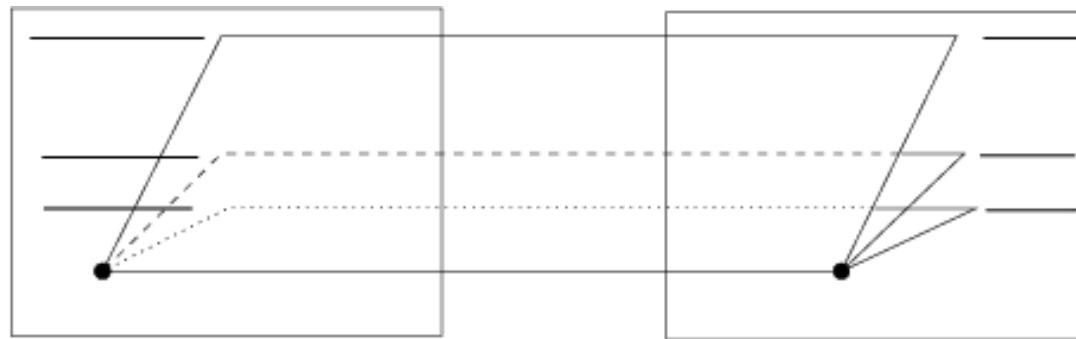
When do epipolar
lines become
horizontal?

Parallel cameras



Where is the epipole?

Parallel cameras



epipole at infinity

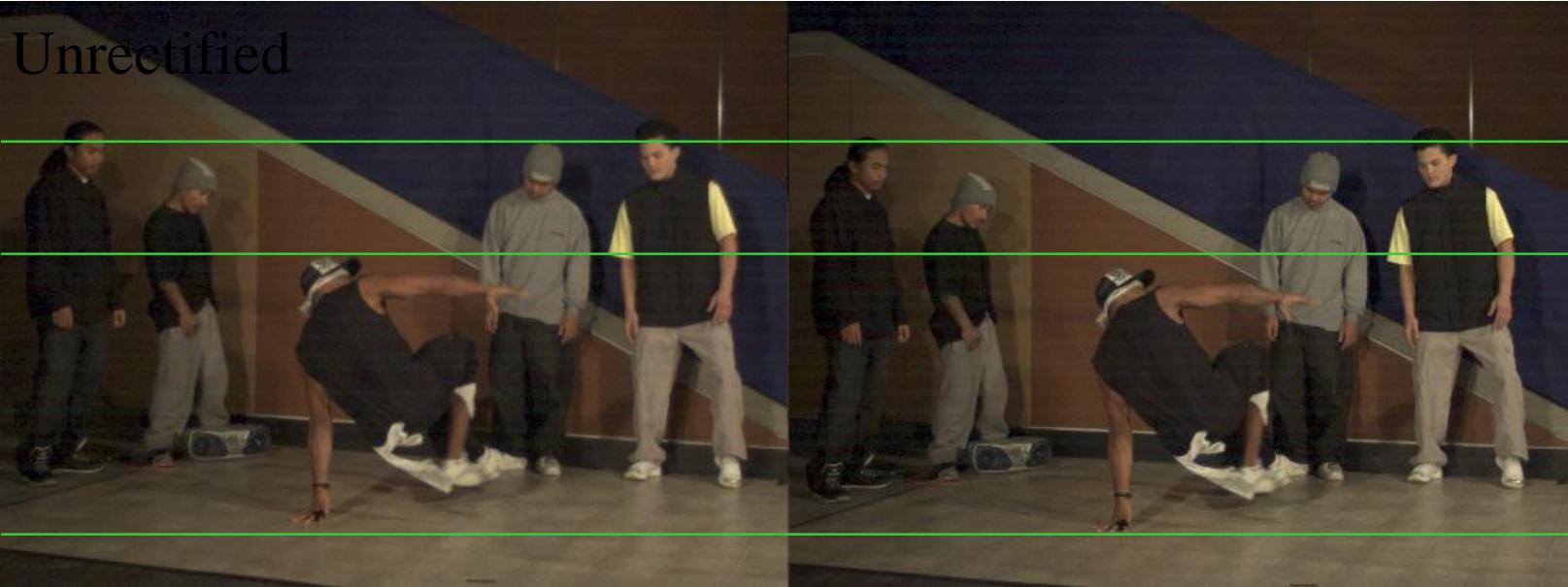
Stereo Rectification Algorithm

1. Estimate \mathbf{E} using the 8 point algorithm (SVD)
2. Estimate the epipole \mathbf{e} (SVD of \mathbf{E})
3. Build \mathbf{R}_{rect} from \mathbf{e}
4. Decompose \mathbf{E} into \mathbf{R} and \mathbf{T}
5. Set $\mathbf{R}_1 = \mathbf{R}_{\text{rect}}$ and $\mathbf{R}_2 = \mathbf{R}\mathbf{R}_{\text{rect}}$
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

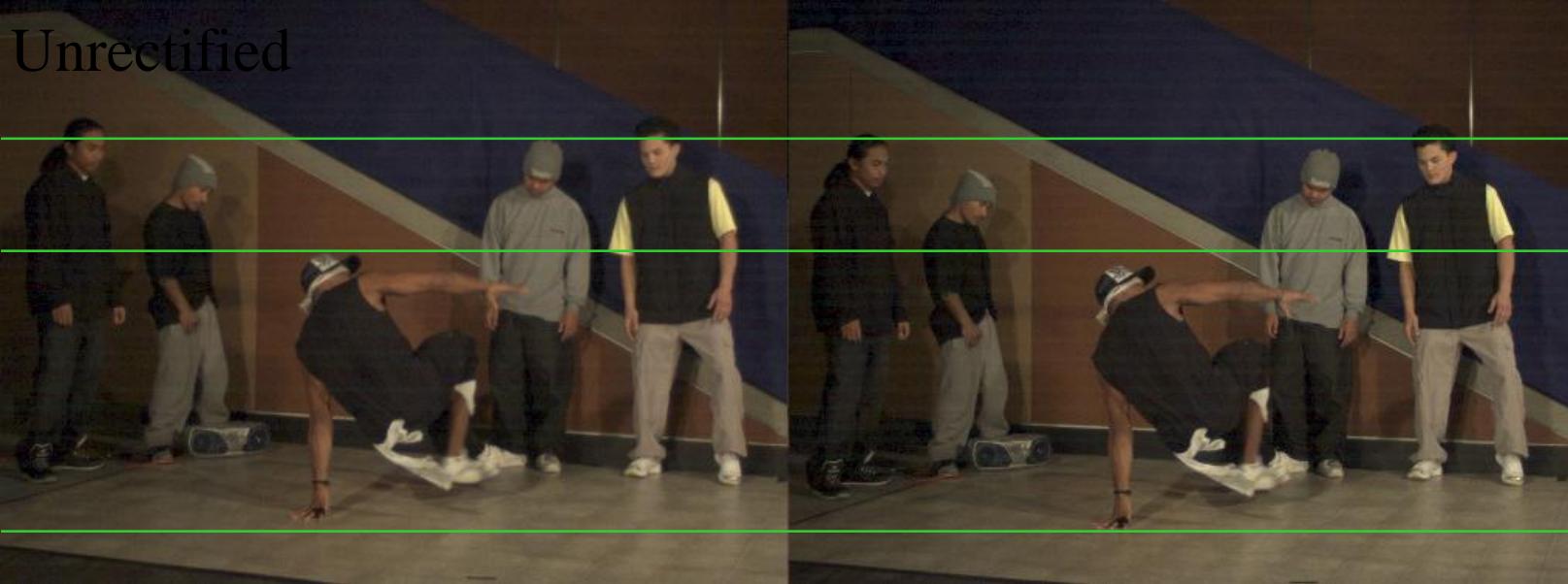
Stereo Rectification Algorithm

1. Estimate \mathbf{E} using the 8 point algorithm
2. Estimate the epipole \mathbf{e} (solve $\mathbf{E}\mathbf{e}=0$)
3. Build \mathbf{R}_{rect} from \mathbf{e}
4. Decompose \mathbf{E} into \mathbf{R} and \mathbf{T}
5. Set $\mathbf{R}_1 = \mathbf{R}_{\text{rect}}$ and $\mathbf{R}_2 = \mathbf{R}\mathbf{R}_{\text{rect}}$
6. Rotate each left camera point $\mathbf{x}' \sim \mathbf{Hx}$ where $\mathbf{H} = \mathbf{KR}_1$
*You may need to alter the focal length (inside \mathbf{K}) to keep points within the original image size
7. Rectified points as $\mathbf{p} = f/z' [x' \ y' \ z']$
8. Repeat 6 and 7 for right camera points using \mathbf{R}_2

Unrectified



Unrectified



Rectified



Unrectified

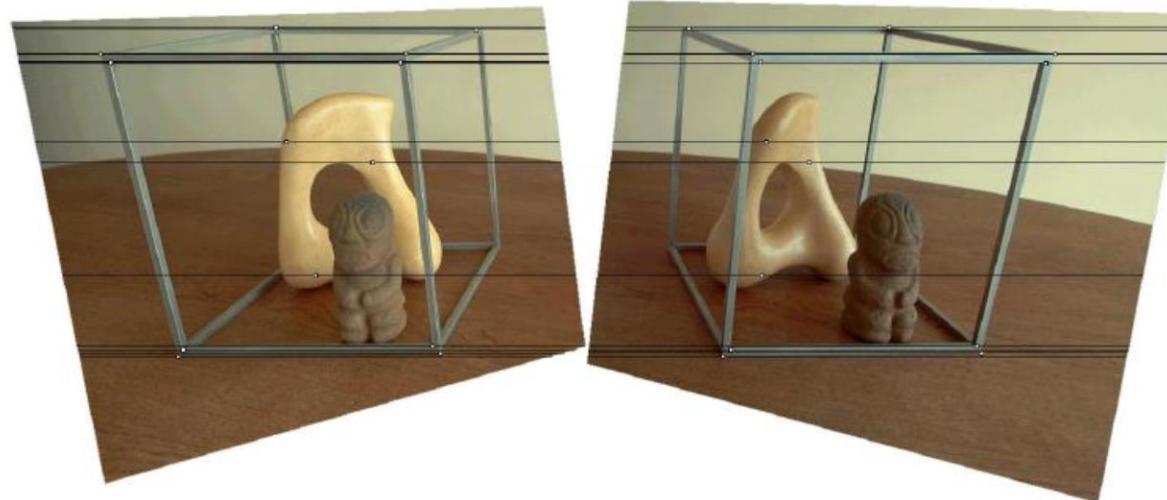


Rectified





What can we do after rectification?

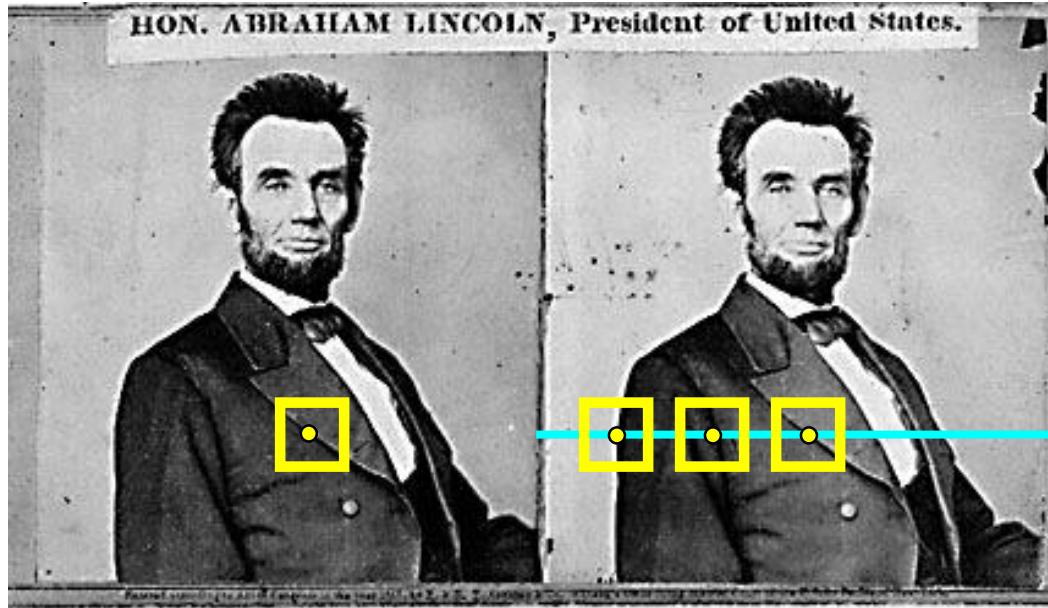


Stereo matching



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?

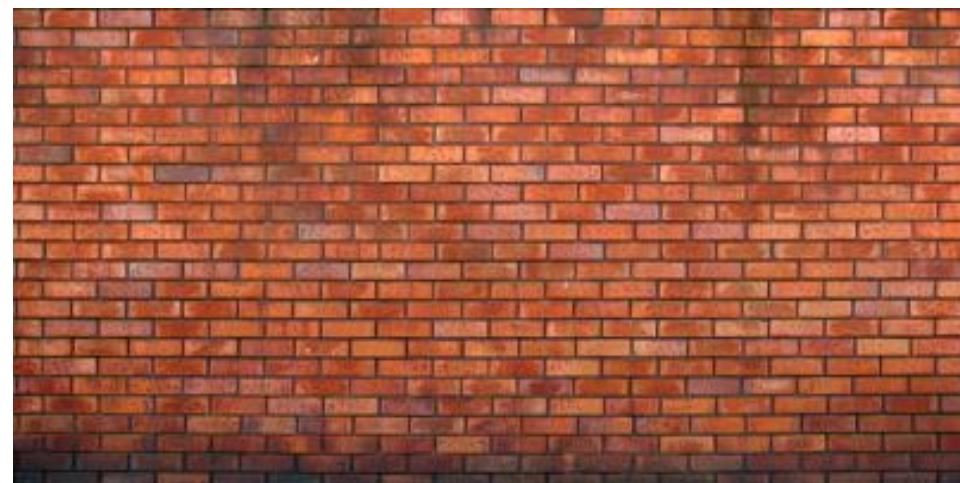
Reminder from filtering

How do we detect an edge?

Reminder from filtering

How do we detect an edge?

- We filter with something that looks like an edge.

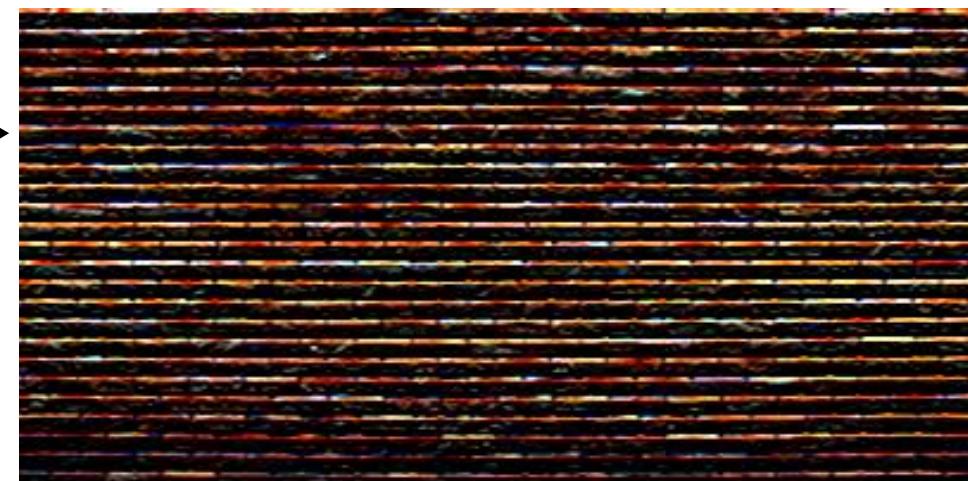


$$\xrightarrow{*} \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline\end{array} \xrightarrow{\hspace{1cm}}$$



horizontal edge filter

$$\xrightarrow{*} \begin{array}{|c|c|c|}\hline 1 \\ \hline 0 \\ \hline -1 \\ \hline\end{array} \xrightarrow{\hspace{1cm}}$$

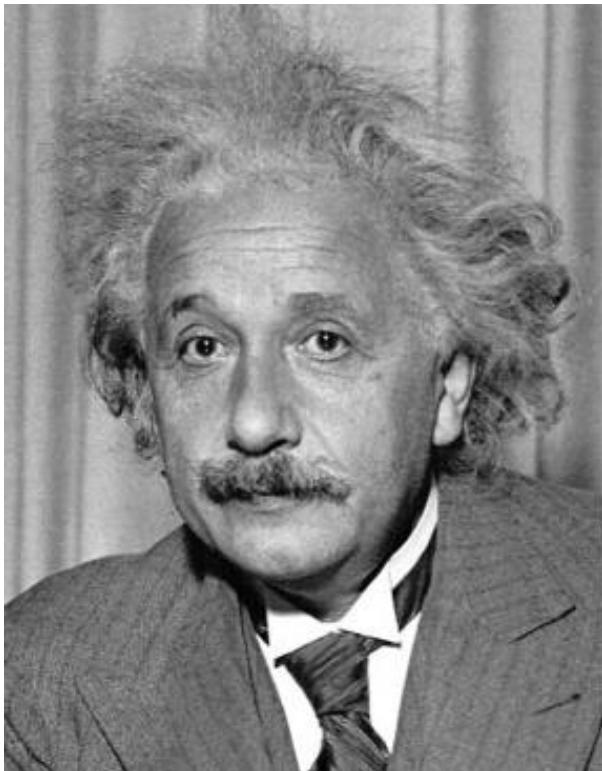


vertical edge filter

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

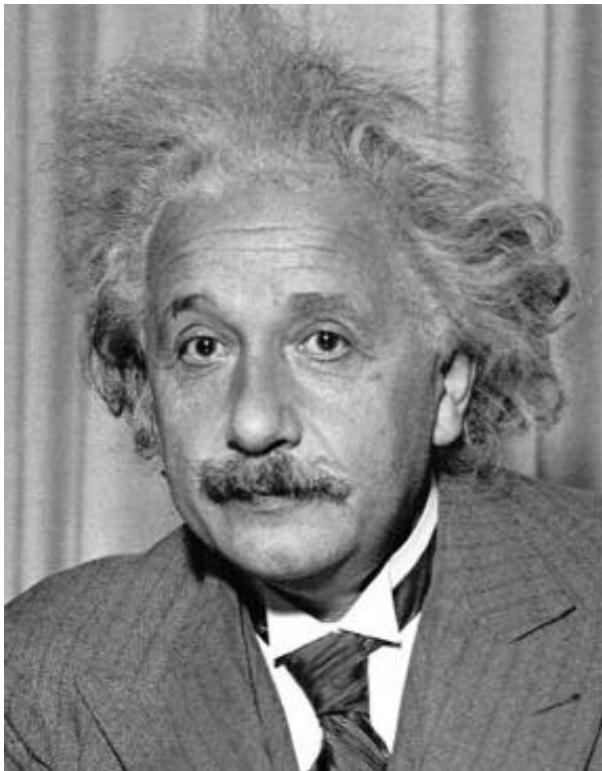
Find this template

How do we detect the template  in the following image?



Find this template

How do we detect the template  in the following image?



output

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

filter 

image

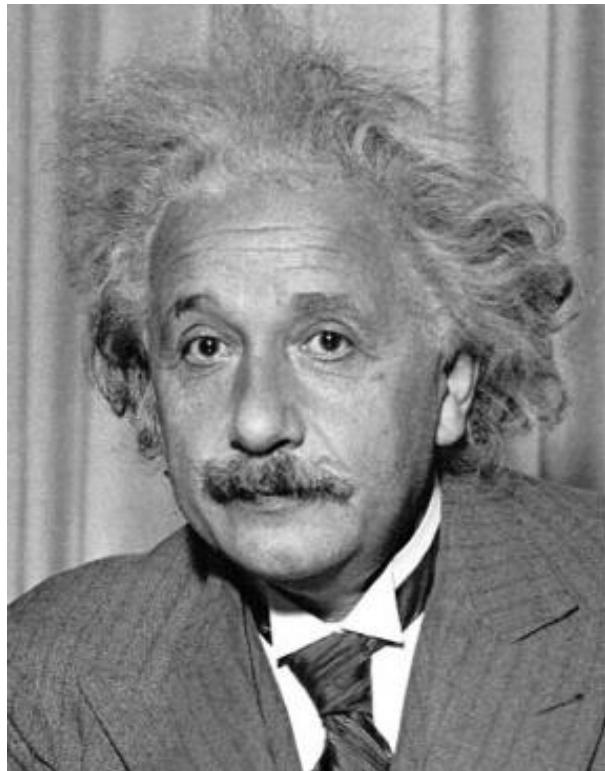
A diagram illustrating the convolution process. An arrow points from a small image of an eye (labeled "filter") down to the mathematical equation. Another arrow points from the word "image" at the bottom left to the "f" in the equation.

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 the following image?



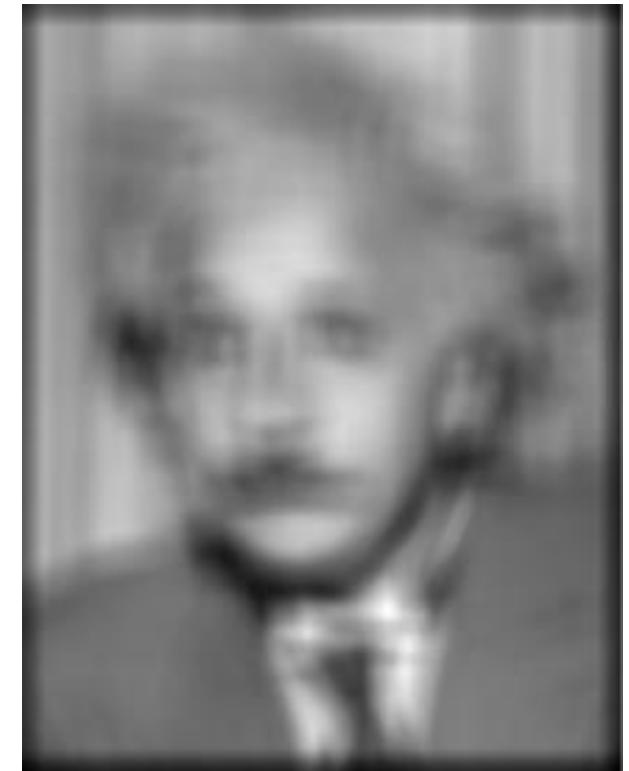
output

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

image

filter 

A diagram illustrating the convolution process. On the left is the input image of Einstein. An arrow labeled "image" points from the image to the formula. Above the formula is the output value $h[m, n]$. A second arrow labeled "filter" points from the eye icon to the term $f[m + k, n + l]$ in the summation formula.

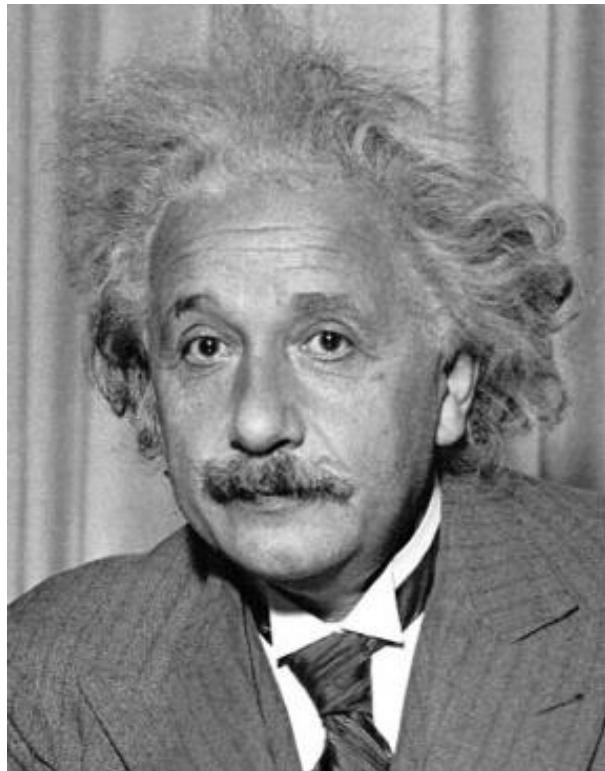


Solution 1: Filter the image using the template as filter kernel.

What went wrong?

Find this template

How do we detect the template  in the following image?



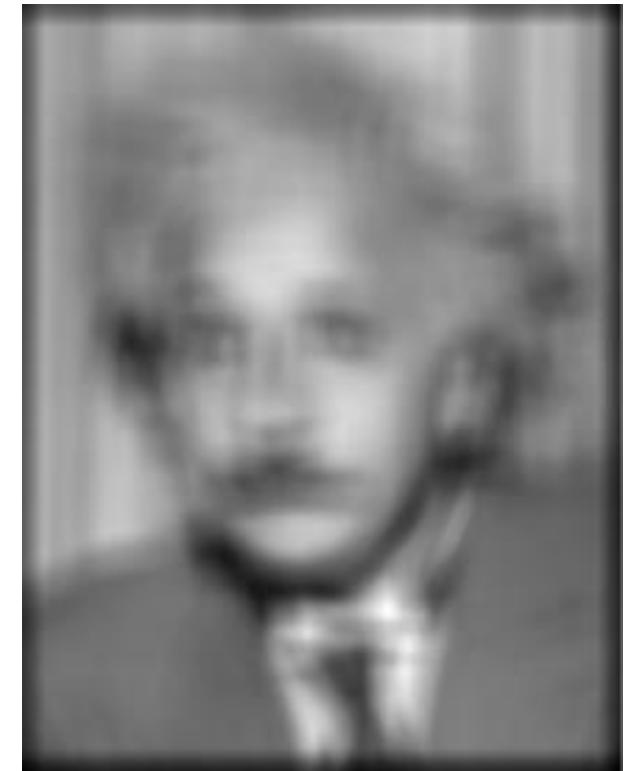
output

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

image

filter 

A diagram illustrating the convolution process. On the left is the input image of Einstein. An arrow labeled "image" points to it. In the center is the output equation: $h[m, n] = \sum_{k,j} g[k, l] f[m + k, n + l]$. Above the equation is the word "output". Above the summation term is the word "filter" with an arrow pointing to a small image of an eye.

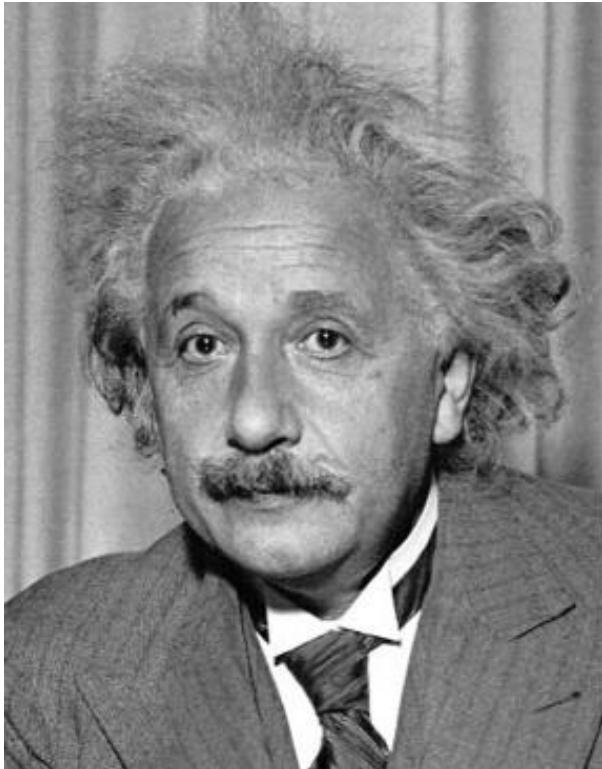


Solution 1: Filter the image using the template as filter kernel.

Increases for higher local intensities.

Find this template

How do we detect the template  in the following image?



output

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

filter 

template mean

image

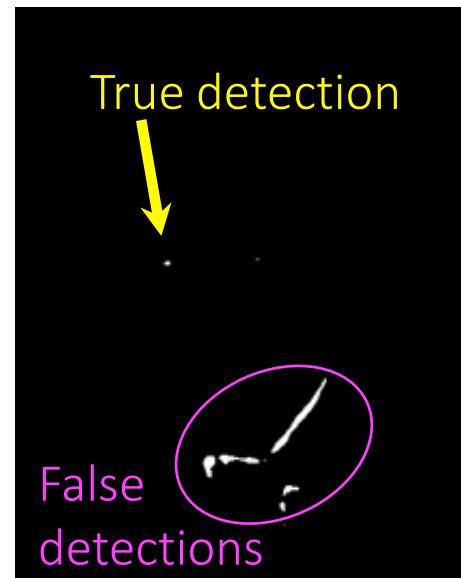
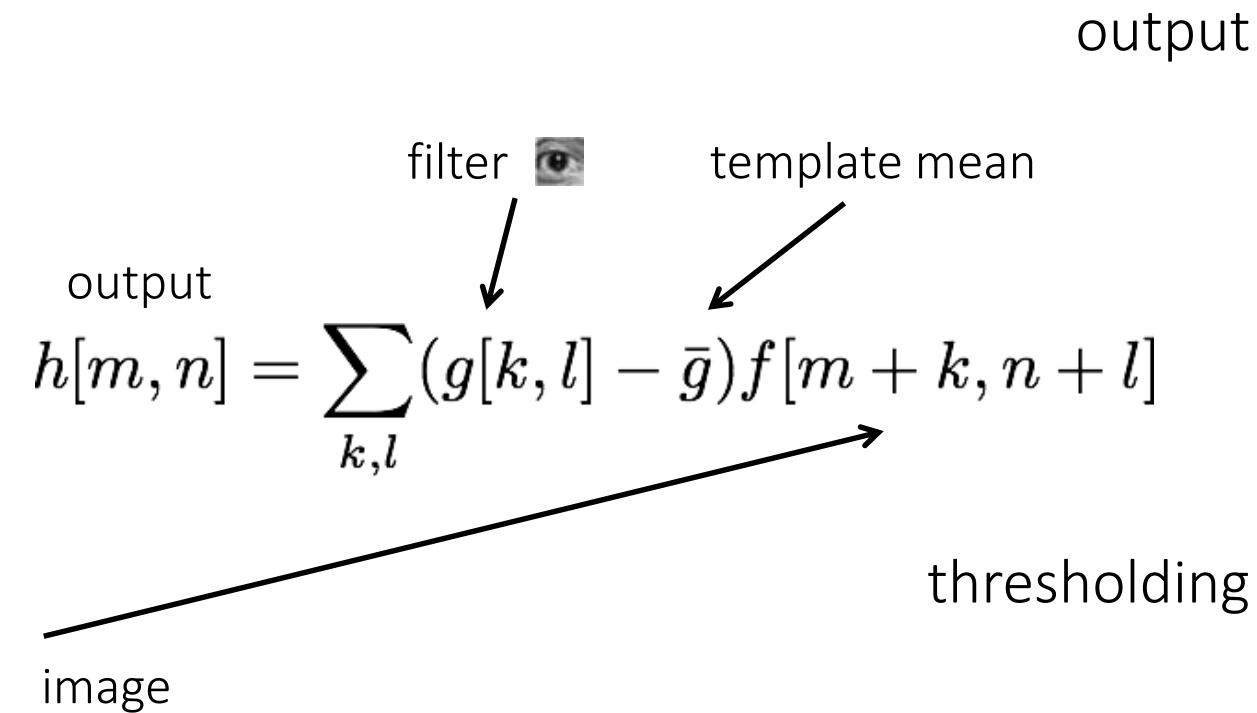
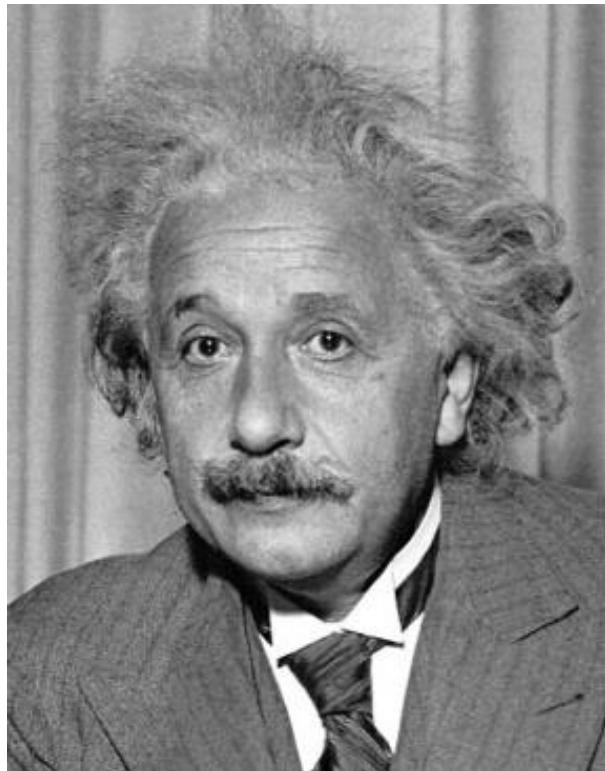
The diagram illustrates the convolution process. An input image is shown on the left. A small eye icon labeled "filter" is positioned above a portion of the image. A horizontal arrow points from the image to the right, labeled "output". Above the output equation, the label "template mean" is placed next to an arrow pointing towards the subtraction term \bar{g} . Another arrow points from the "image" label to the right side of the equation, indicating the result of the convolution step.

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 the following image?

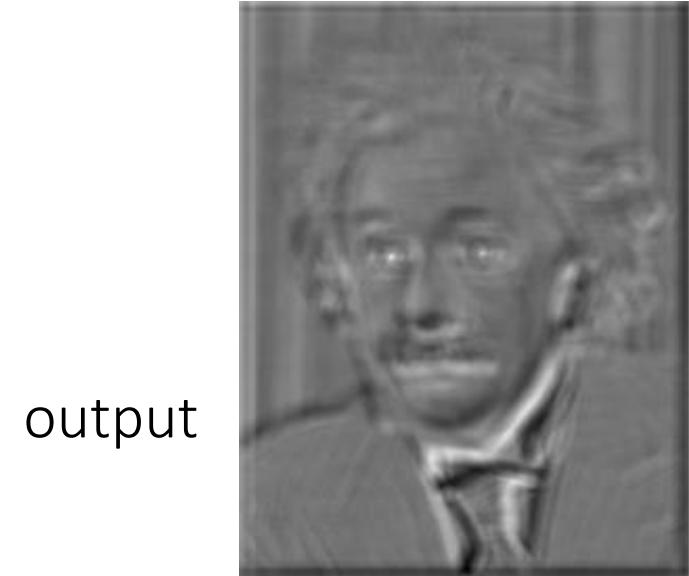
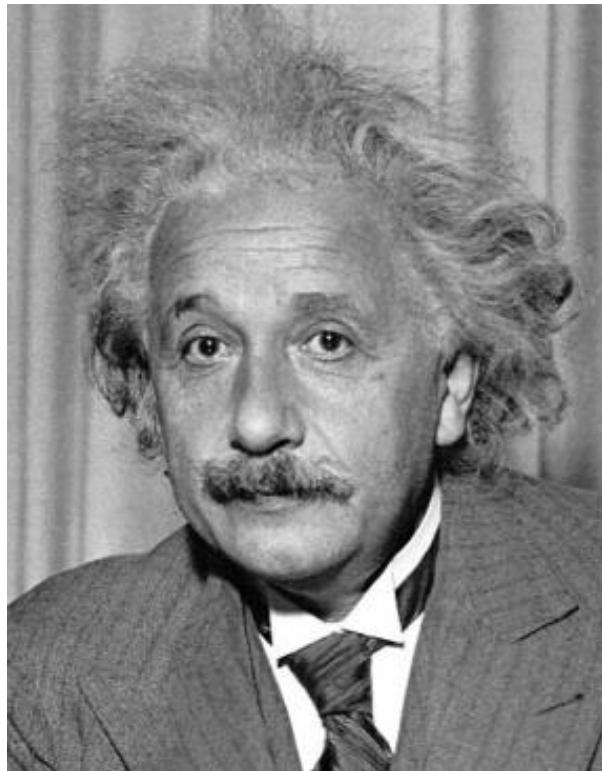


Solution 2: Filter the image using a *zero-mean* template.

What went wrong?

Find this template

How do we detect the template  in the following image?



output

filter 

template mean

image

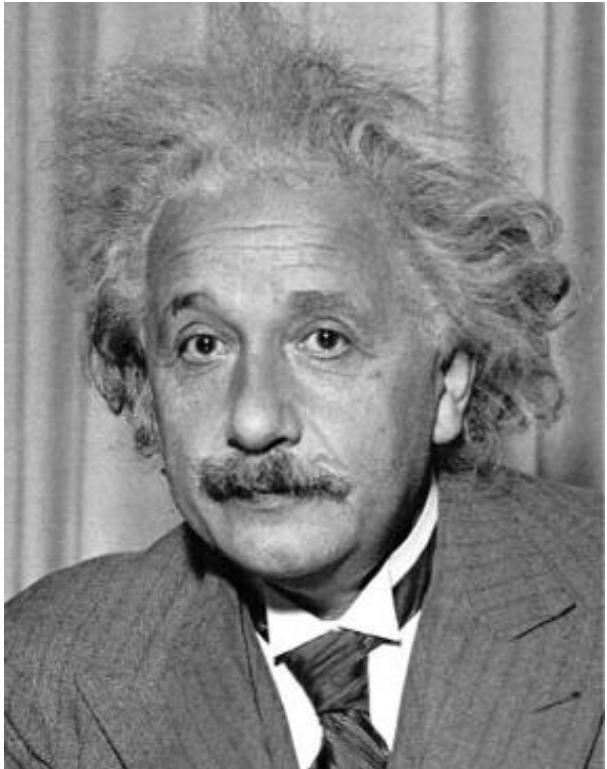
$$h[m, n] = \sum_{k, l} (g[k, l] - \bar{g})f[m + k, n + l]$$

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 the following image?



output

filter 

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

image

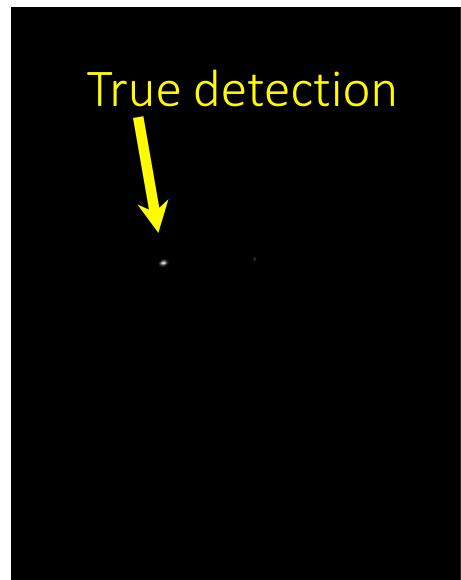
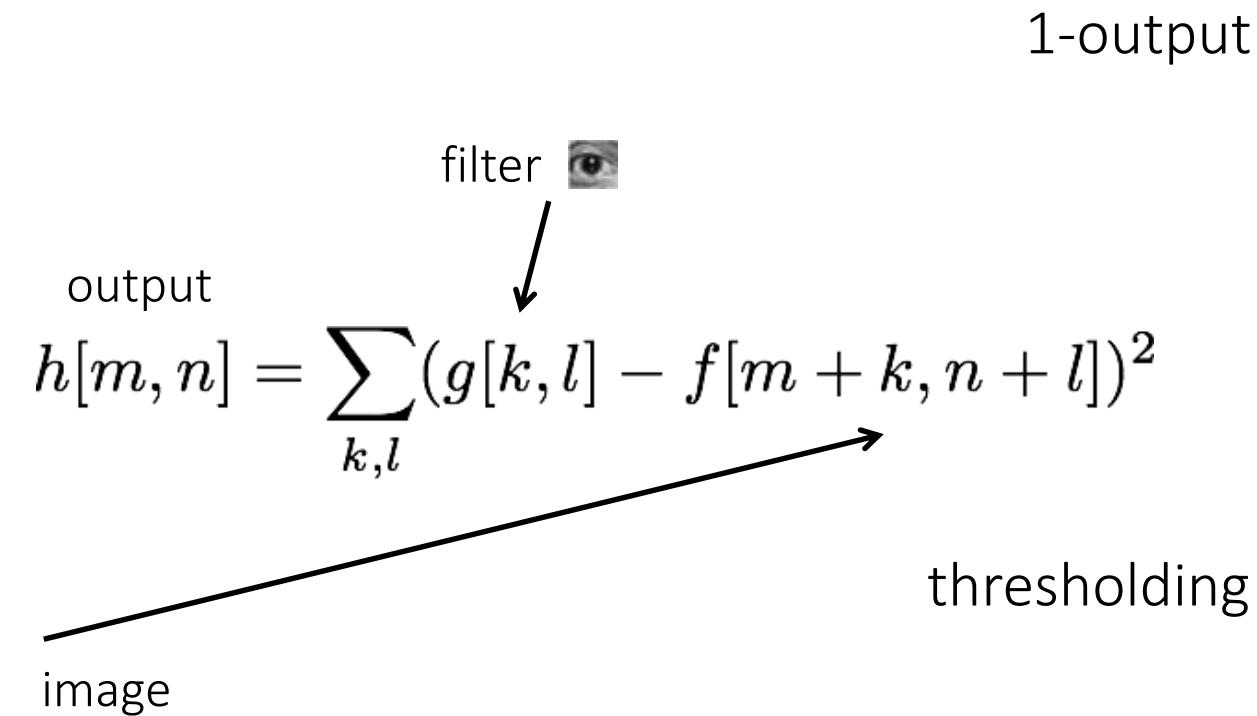
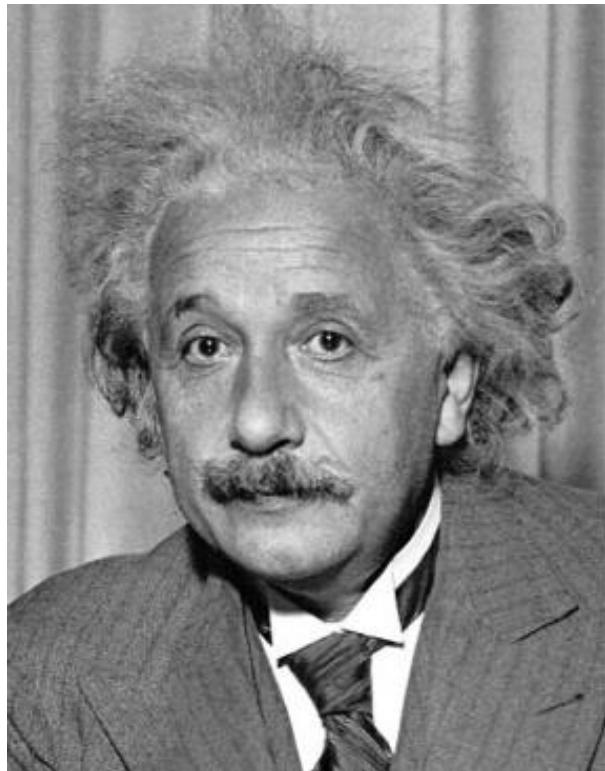
A diagram illustrating the template matching process. An arrow points from the word "filter" to a small icon of an eye. Another arrow points from the word "image" to the mathematical formula for the sum of squared differences (SSD) metric.

What will
the output
look like?

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

Find this template

How do we detect the template  in the following image?

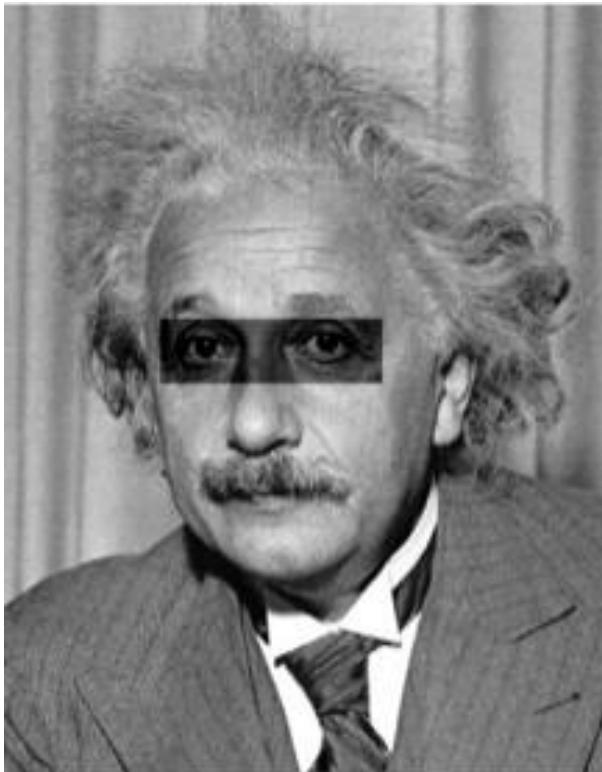


Solution 3: Use sum of squared differences (SSD).

What could go wrong?

Find this template

How do we detect the template  in the following image?



output

filter 

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

image

1-output

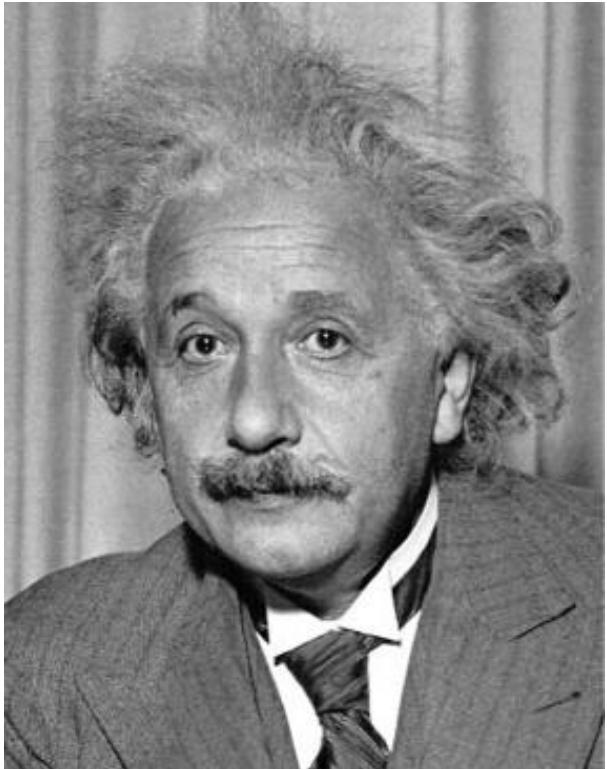


Not robust to local intensity changes

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

Find this template

How do we detect the template  in the following image?



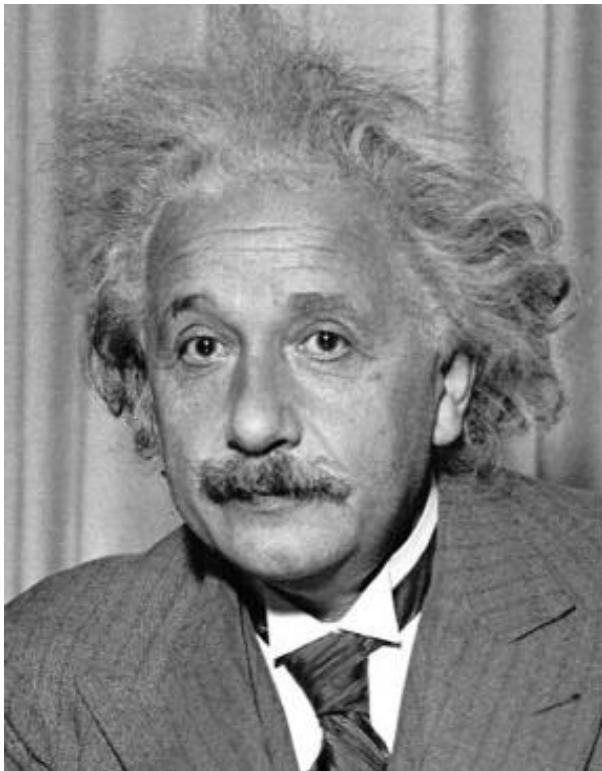
Observations so far:

- 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 the following image?



What will
the output
look like?

filter 

template mean

output

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

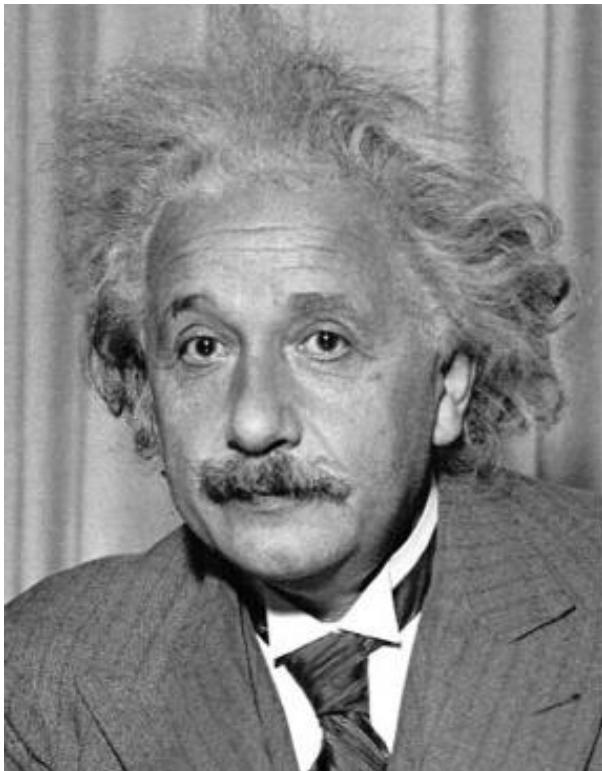
image

local patch mean

Solution 4: Normalized cross-correlation (NCC).

Find this template

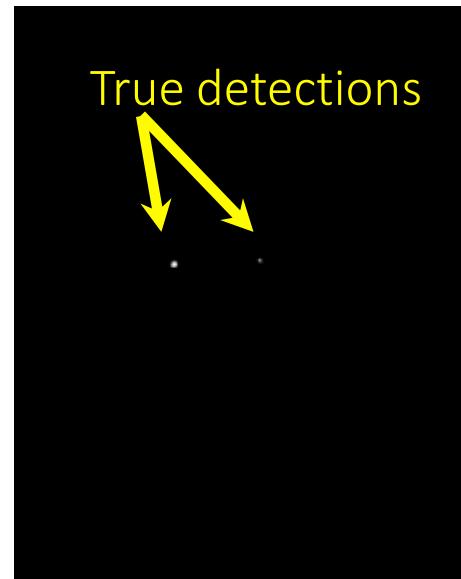
How do we detect the template  in the following image?



1-output



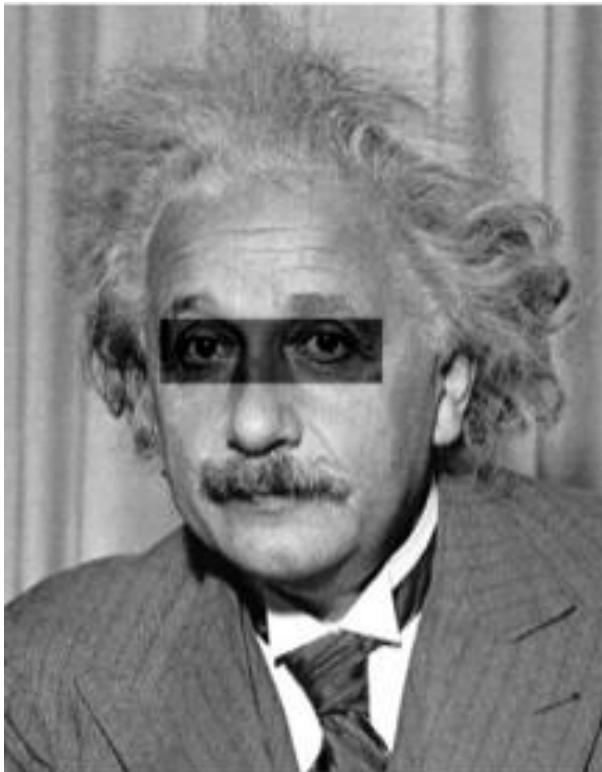
thresholding



Solution 4: Normalized cross-correlation (NCC).

Find this template

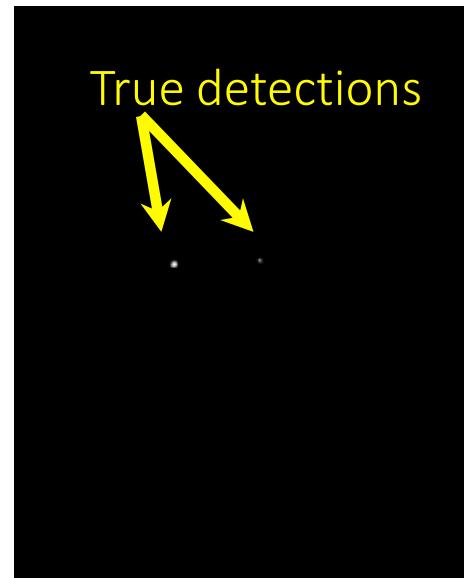
How do we detect the template  in the following image?



1-output



thresholding



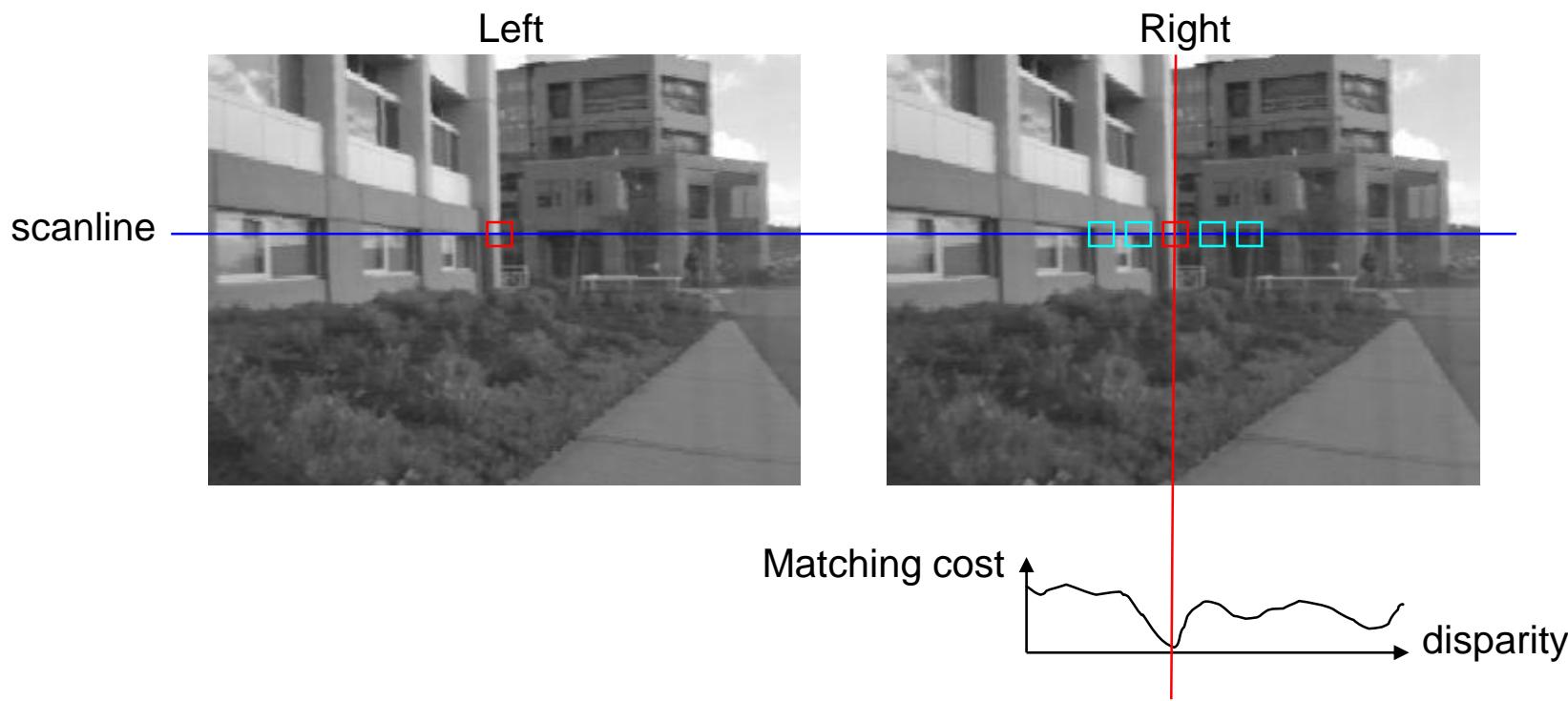
Solution 4: Normalized cross-correlation (NCC).

What is the best method?

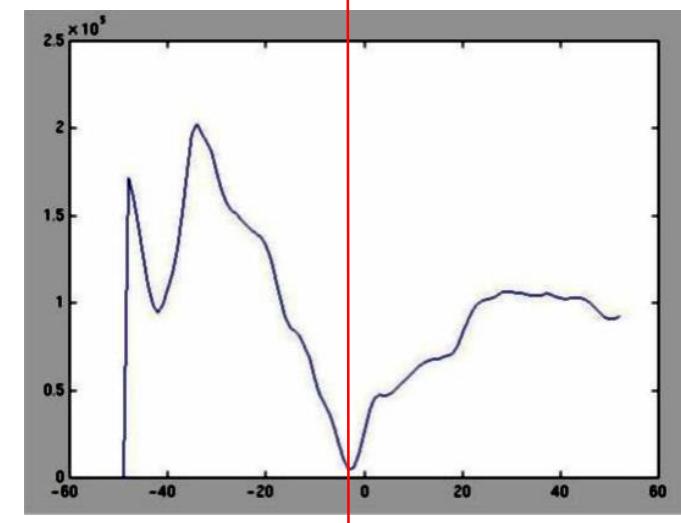
It depends on whether you care about speed or invariance.

- 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.

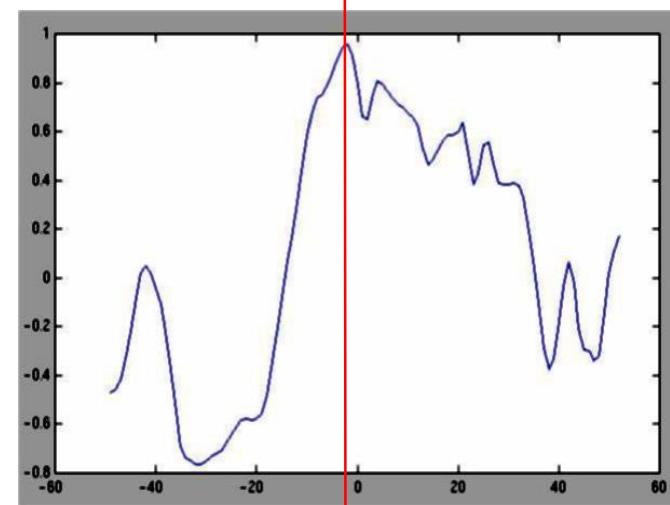
Stereo Block Matching



- 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

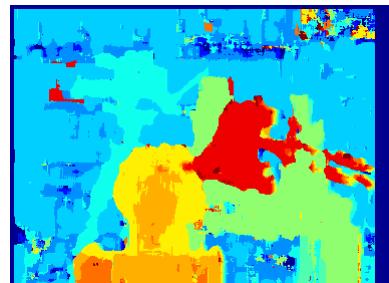


SSD

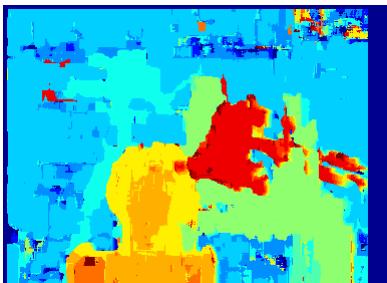


Normalized cross-correlation

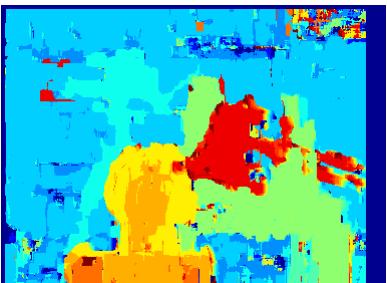
| Similarity Measure | Formula |
|------------------------------------|---|
| Sum of Absolute Differences (SAD) | $\sum_{(i,j) \in W} I_1(i,j) - I_2(x+i, y+j) $ |
| Sum of Squared Differences (SSD) | $\sum_{(i,j) \in W} (I_1(i,j) - I_2(x+i, y+j))^2$ |
| Zero-mean SAD | $\sum_{(i,j) \in W} I_1(i,j) -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



SAD



SSD



NCC

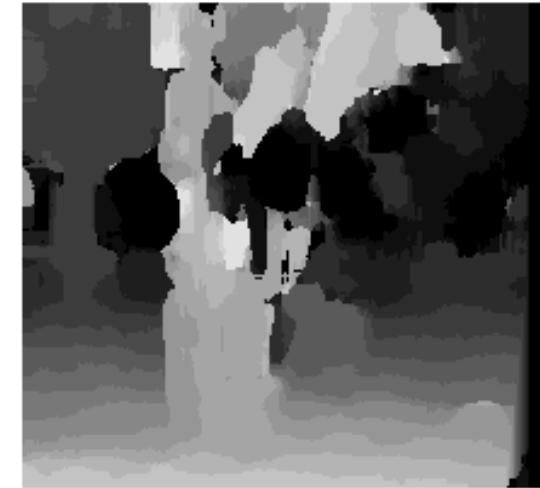


Ground truth

Effect of window size



$W = 3$

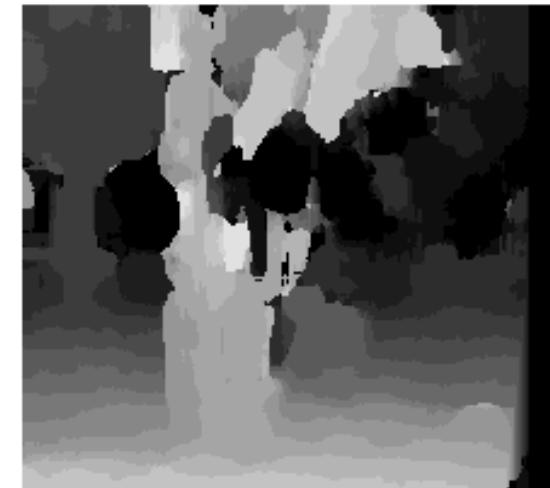


$W = 20$

Effect of window size



$W = 3$



$W = 20$

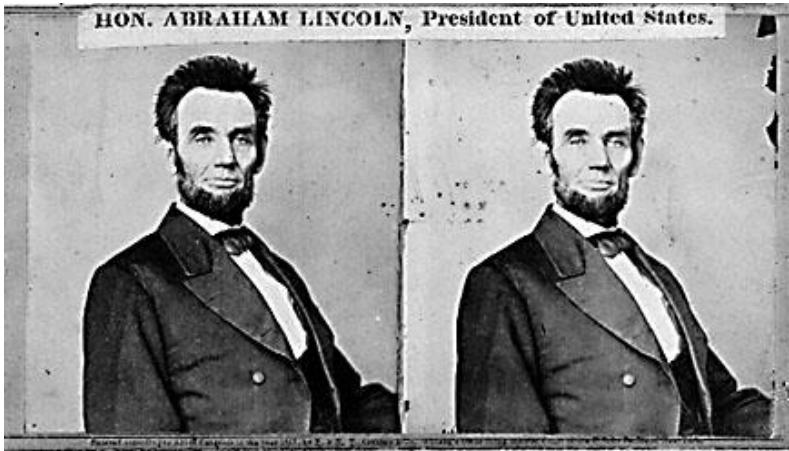
Smaller window

- + More detail
- More noise

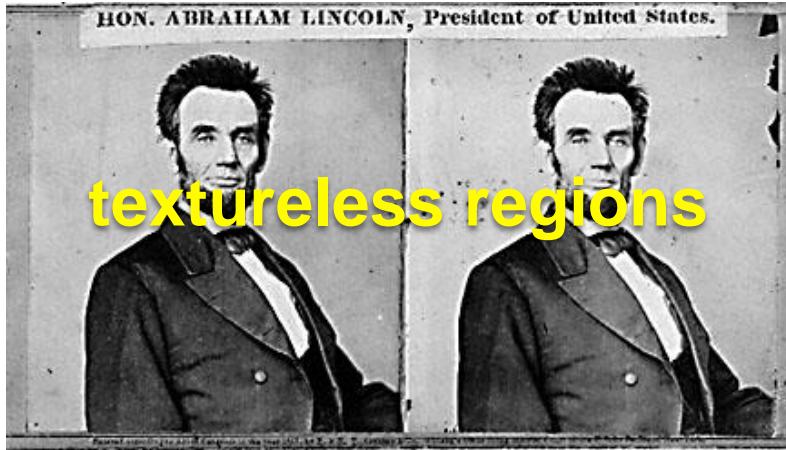
Larger window

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

When will stereo block matching fail?



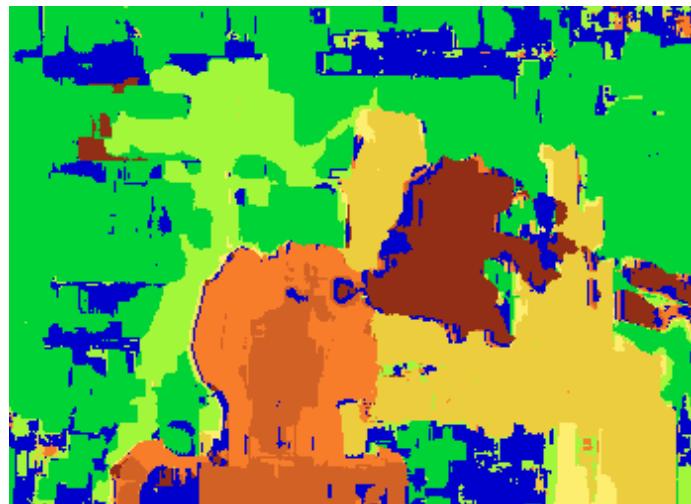
When will stereo block matching fail?



Improving stereo matching



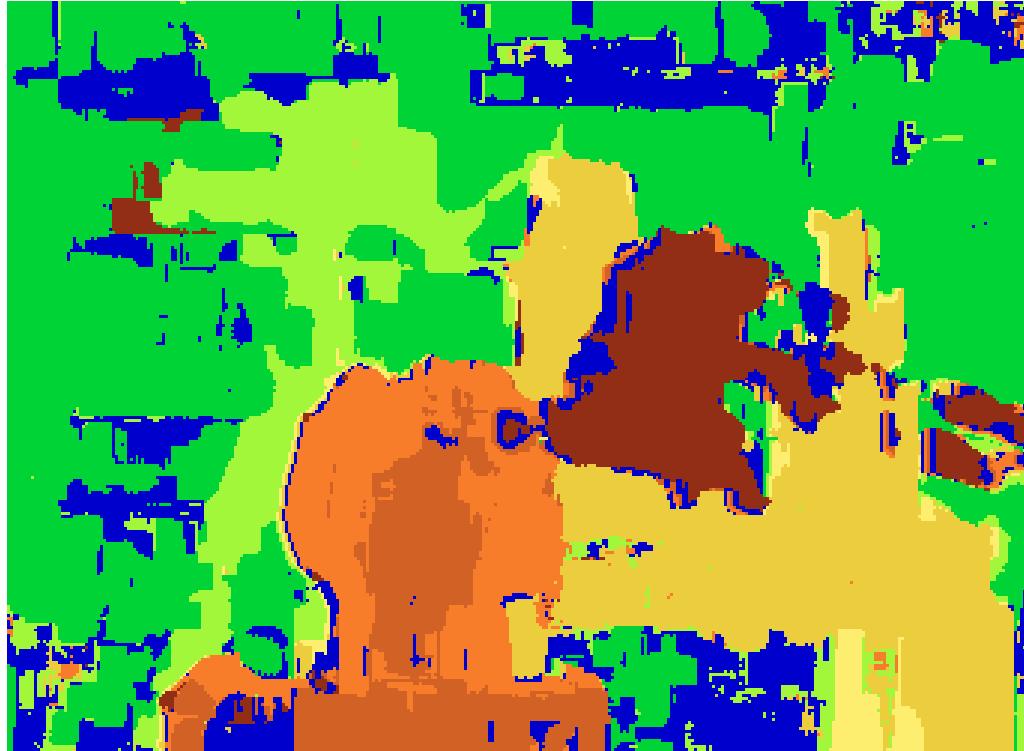
Block matching



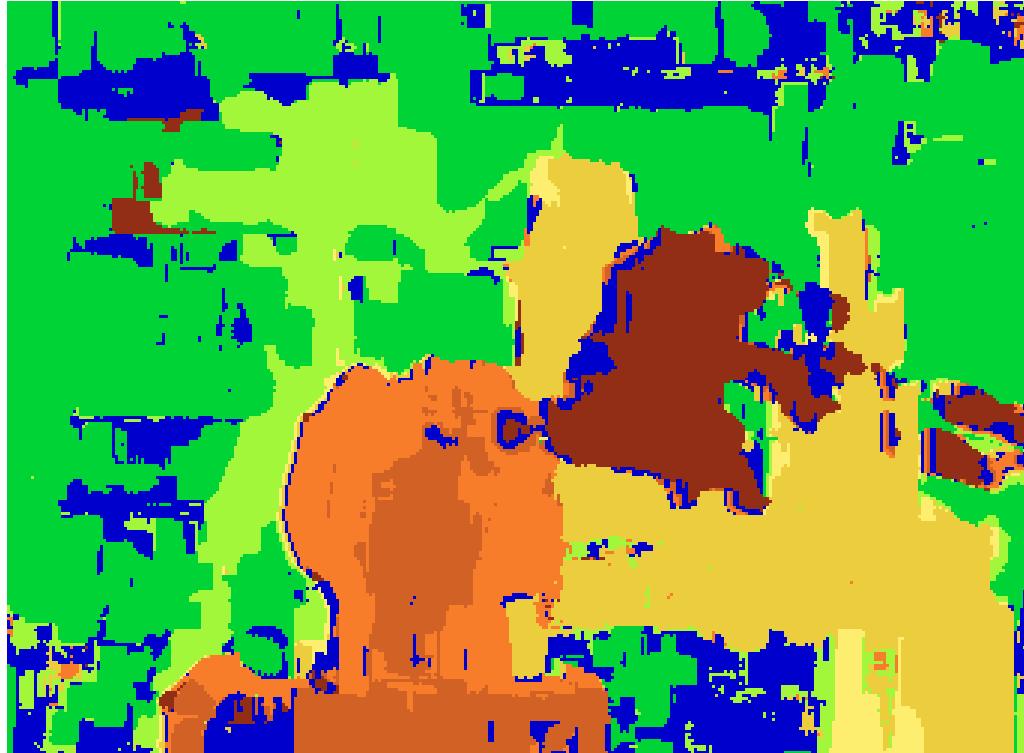
Ground truth



What are some problems with the result?



How can we improve depth estimation?



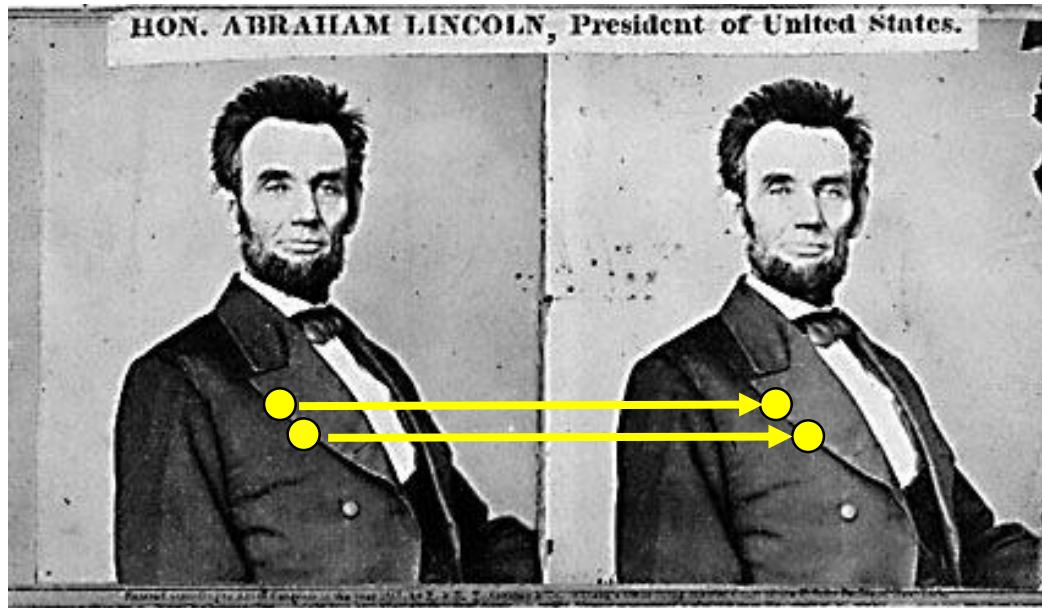
How can we improve depth estimation?

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)

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

data term

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)

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

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

data term

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

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

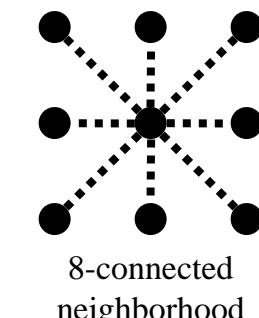
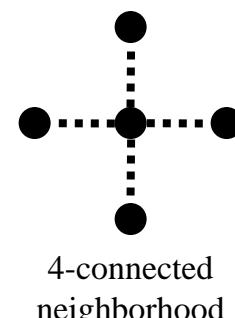
$$E_d(d) = \sum_{(x,y) \in I} C(x, y, d(x, y))$$

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

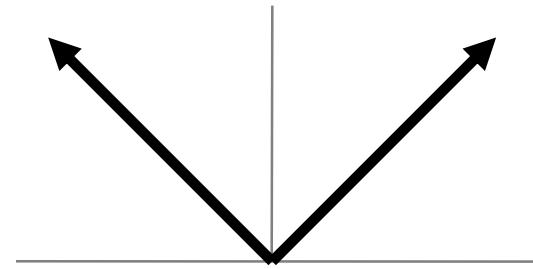
\mathcal{E} : set of neighboring pixels



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

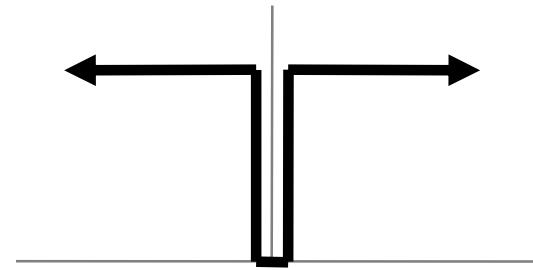
$$V(d_p, d_q) = |d_p - d_q|$$

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

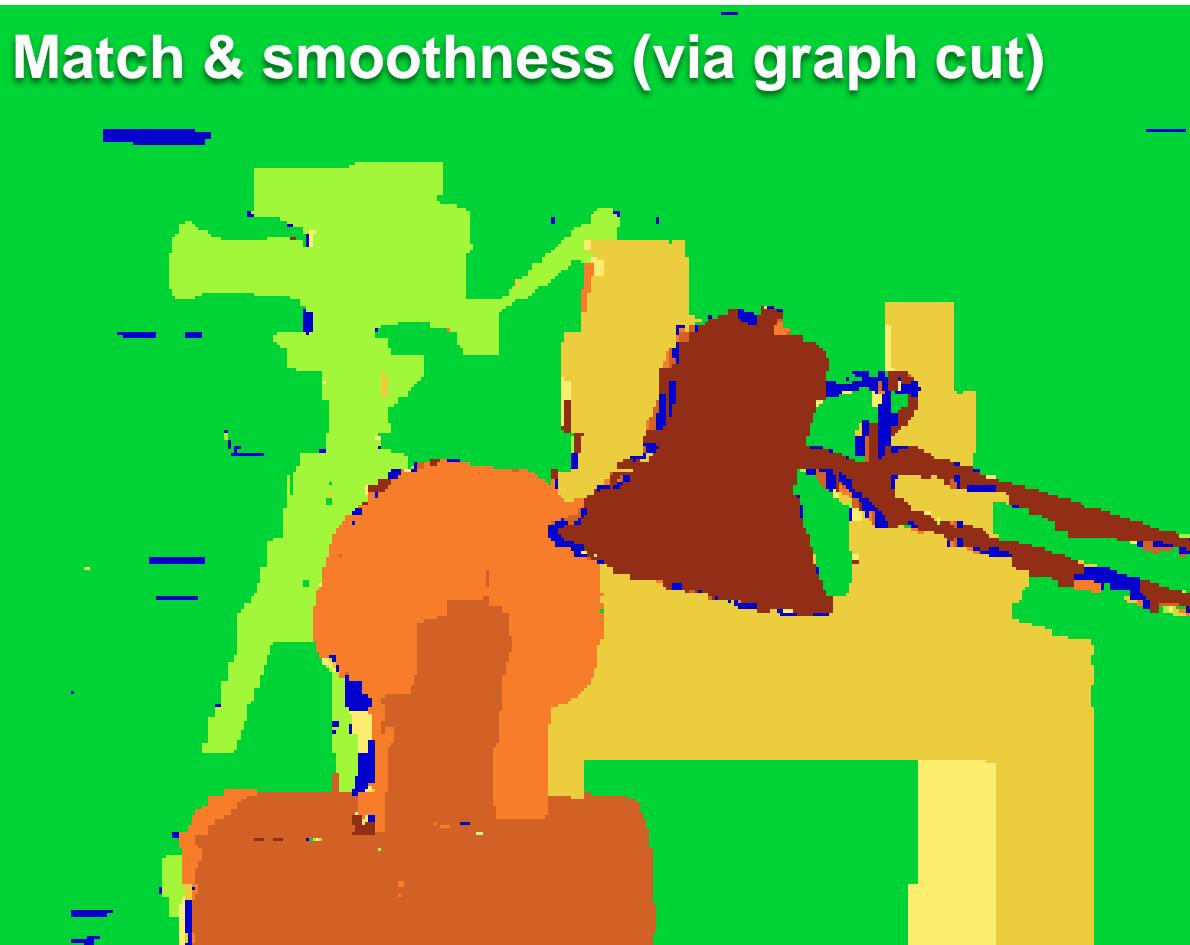
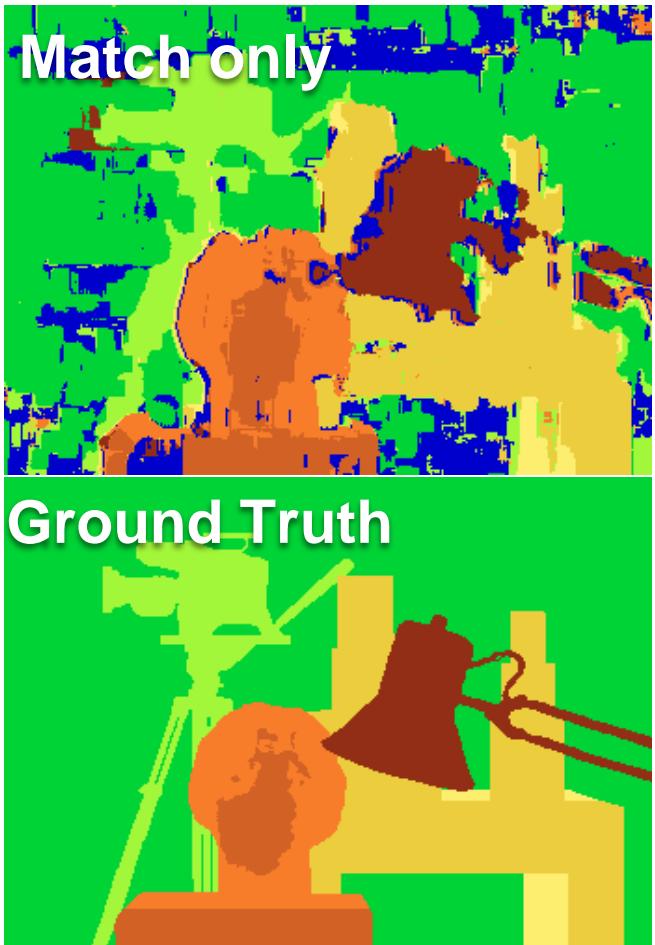
$$E(d) = E_d(d) + \lambda E_s(d)$$

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



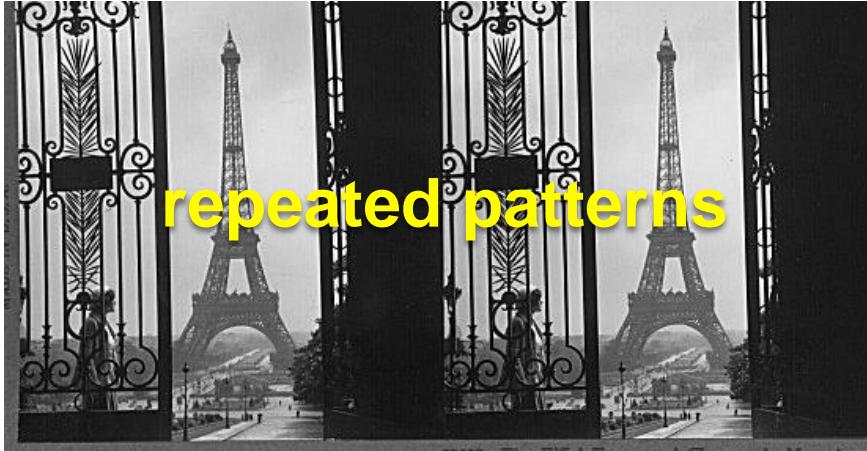
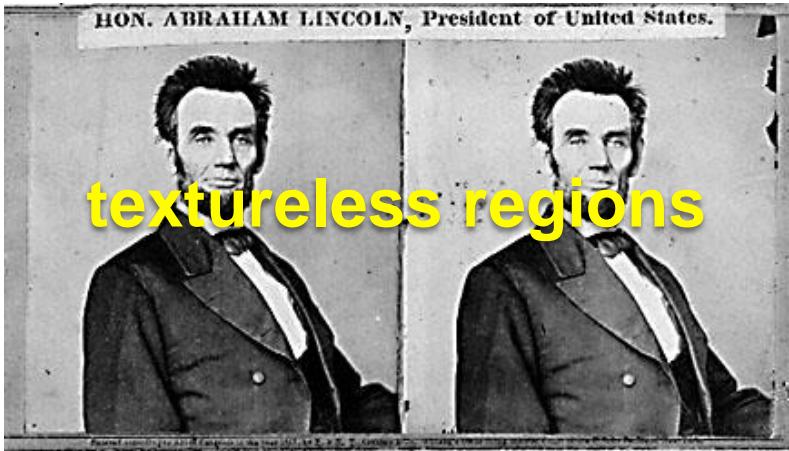
$D(x, y, d)$: minimum cost of solution such that $d(x,y) = d$

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



Y. Boykov, O. Veksler, and R. Zabih, [Fast Approximate Energy Minimization via Graph Cuts](#), PAMI 2001

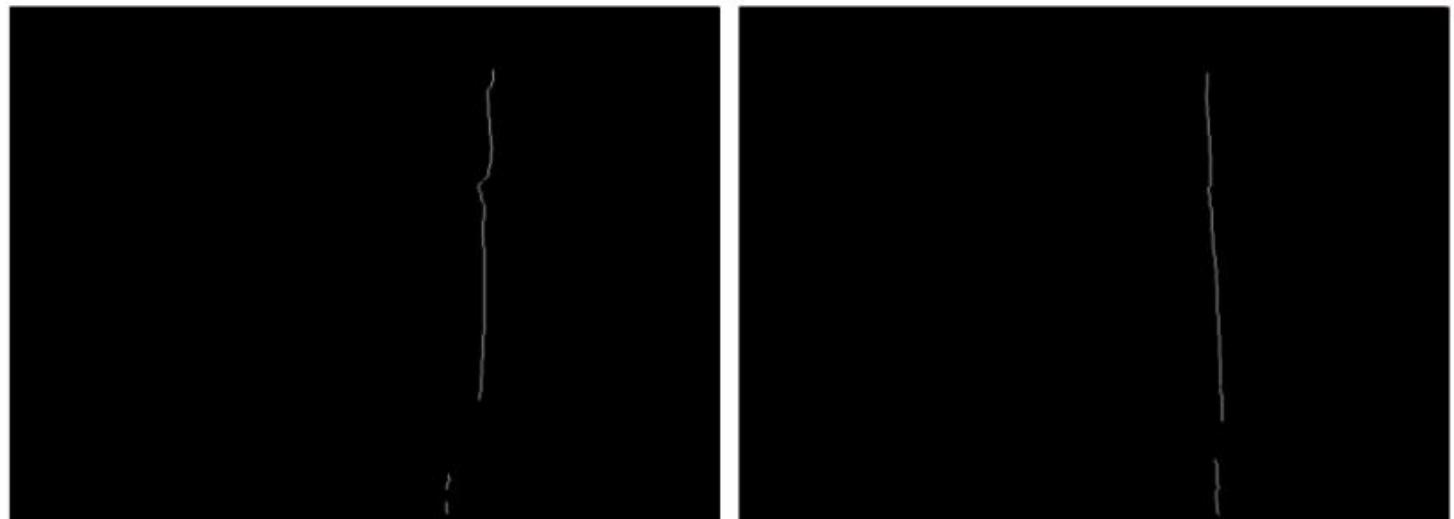
All of these cases remain difficult, what can we do?



Structured light

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

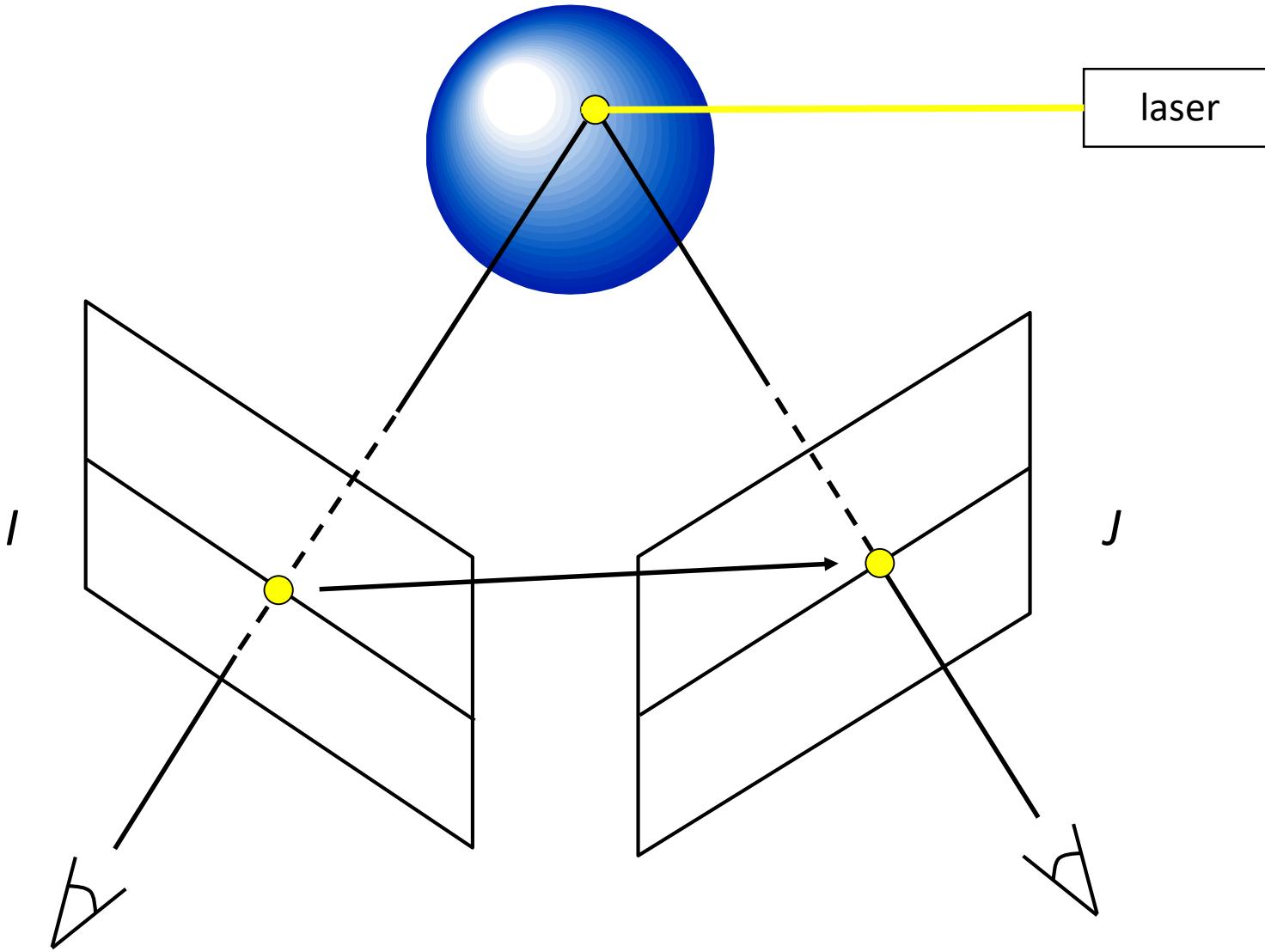
Disparity between laser points on the same scanline in the images determines the 3-D coordinates of the laser point on object



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

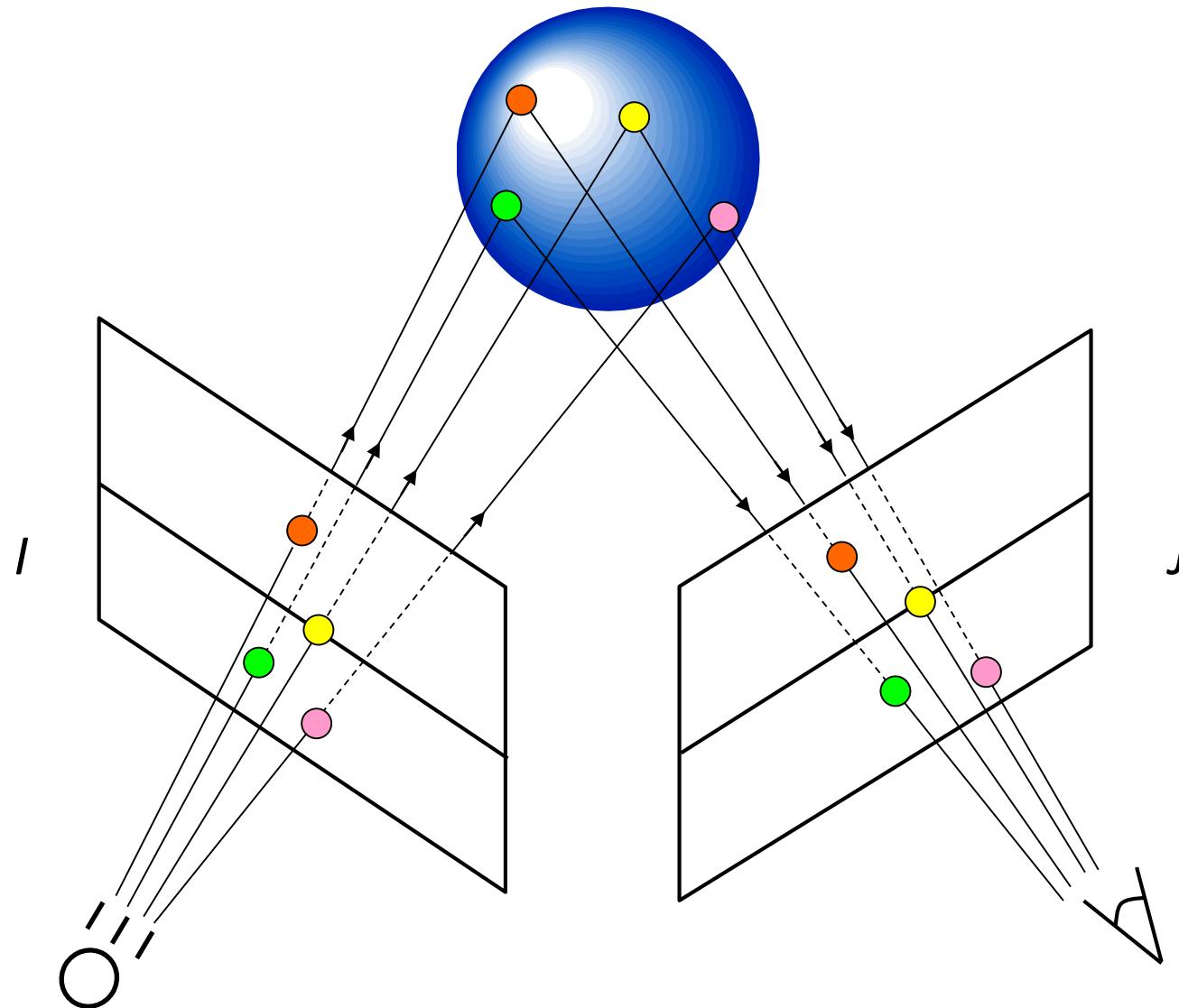


Structured light and two cameras

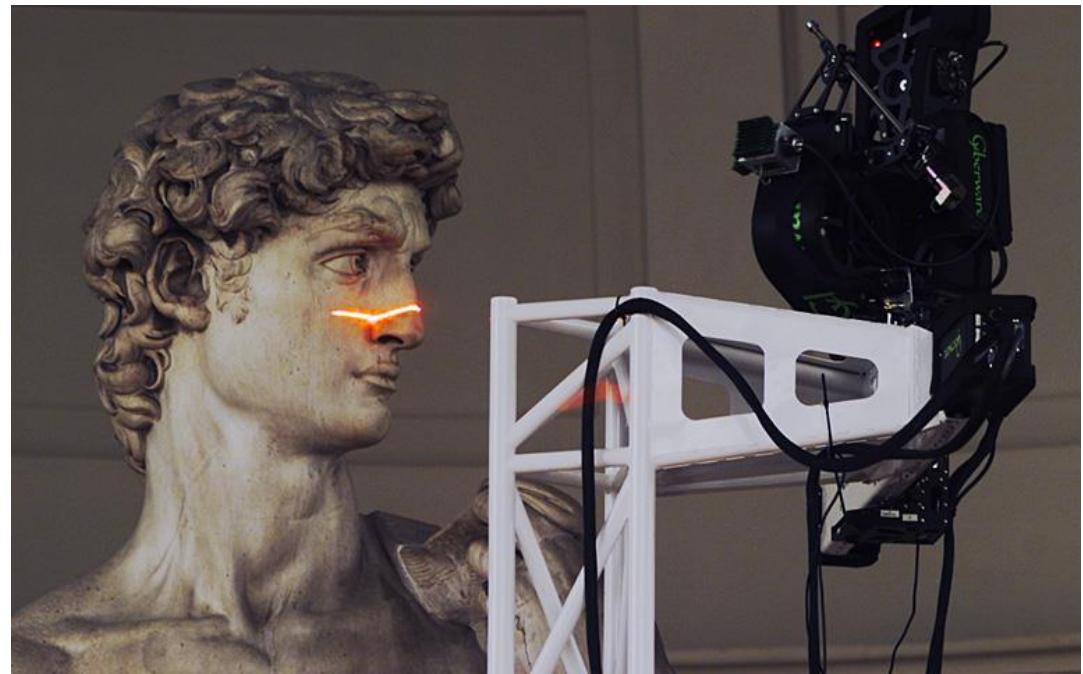
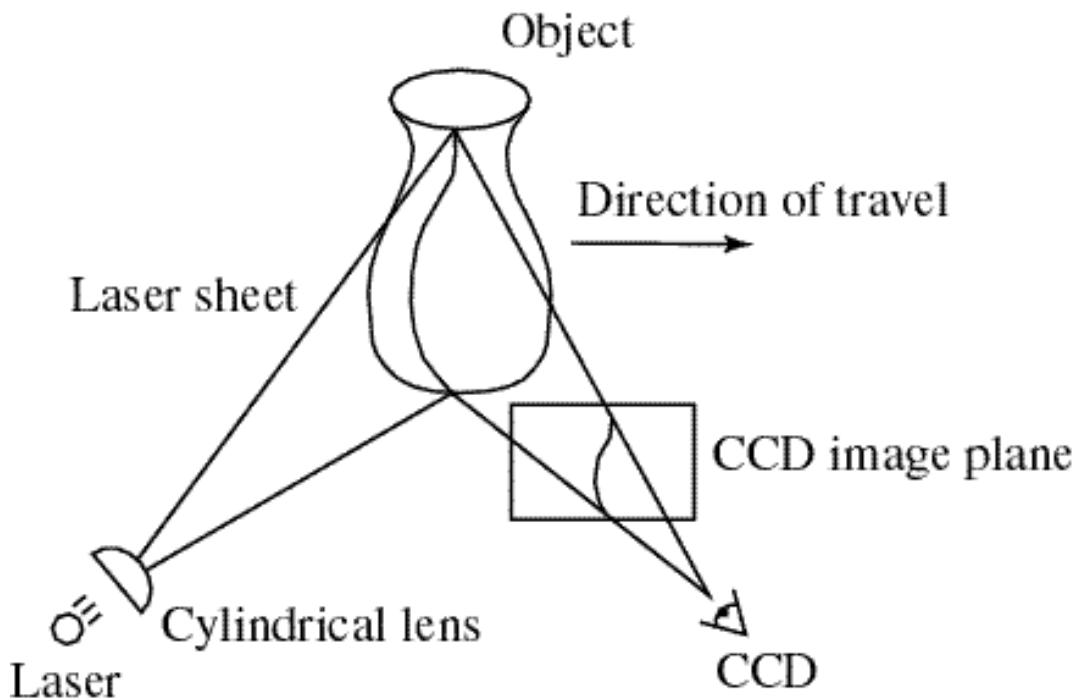


Structured light and one camera

Projector acts like
“reverse” camera



Example: Laser scanner



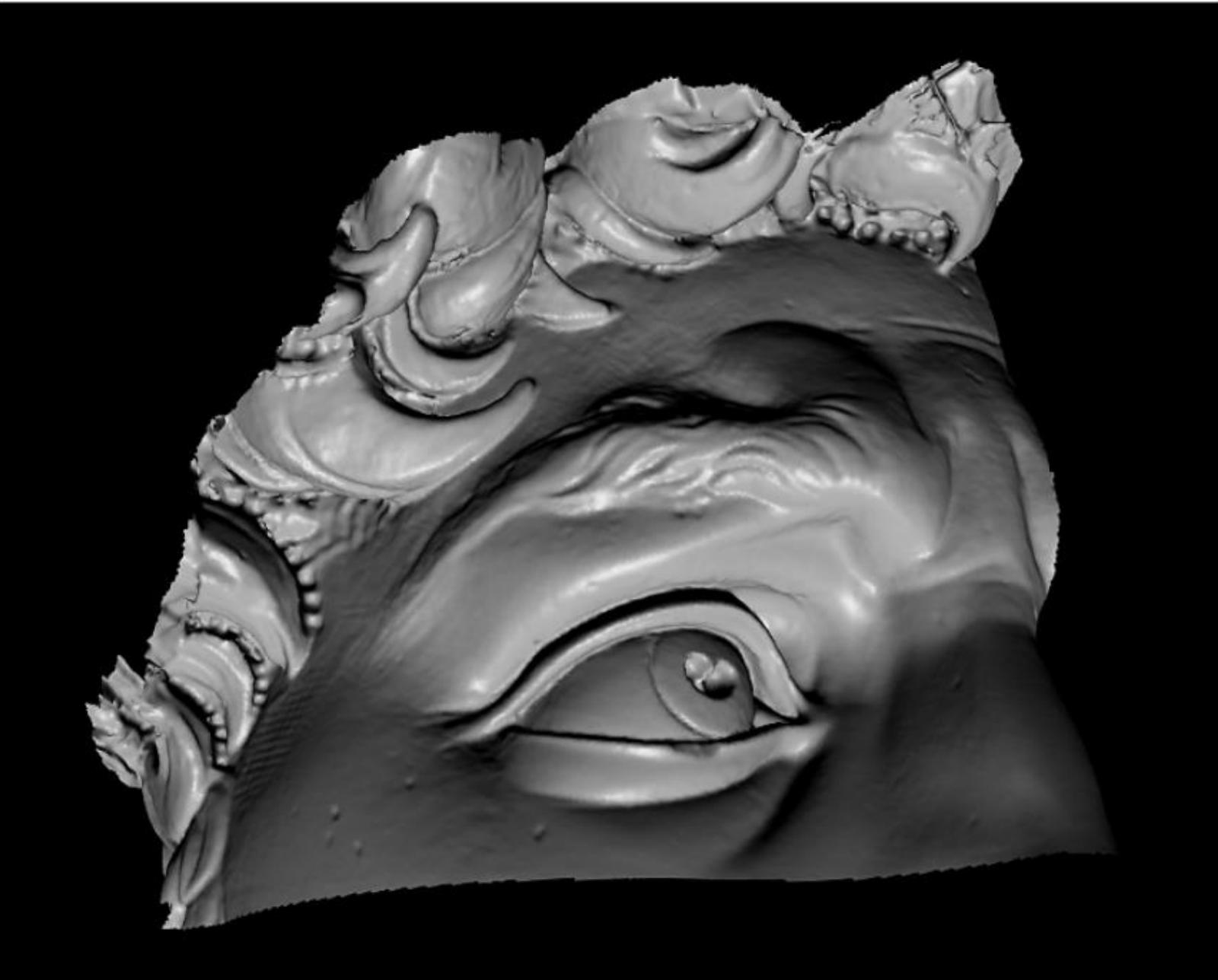
Digital Michelangelo Project
<http://graphics.stanford.edu/projects/mich/>



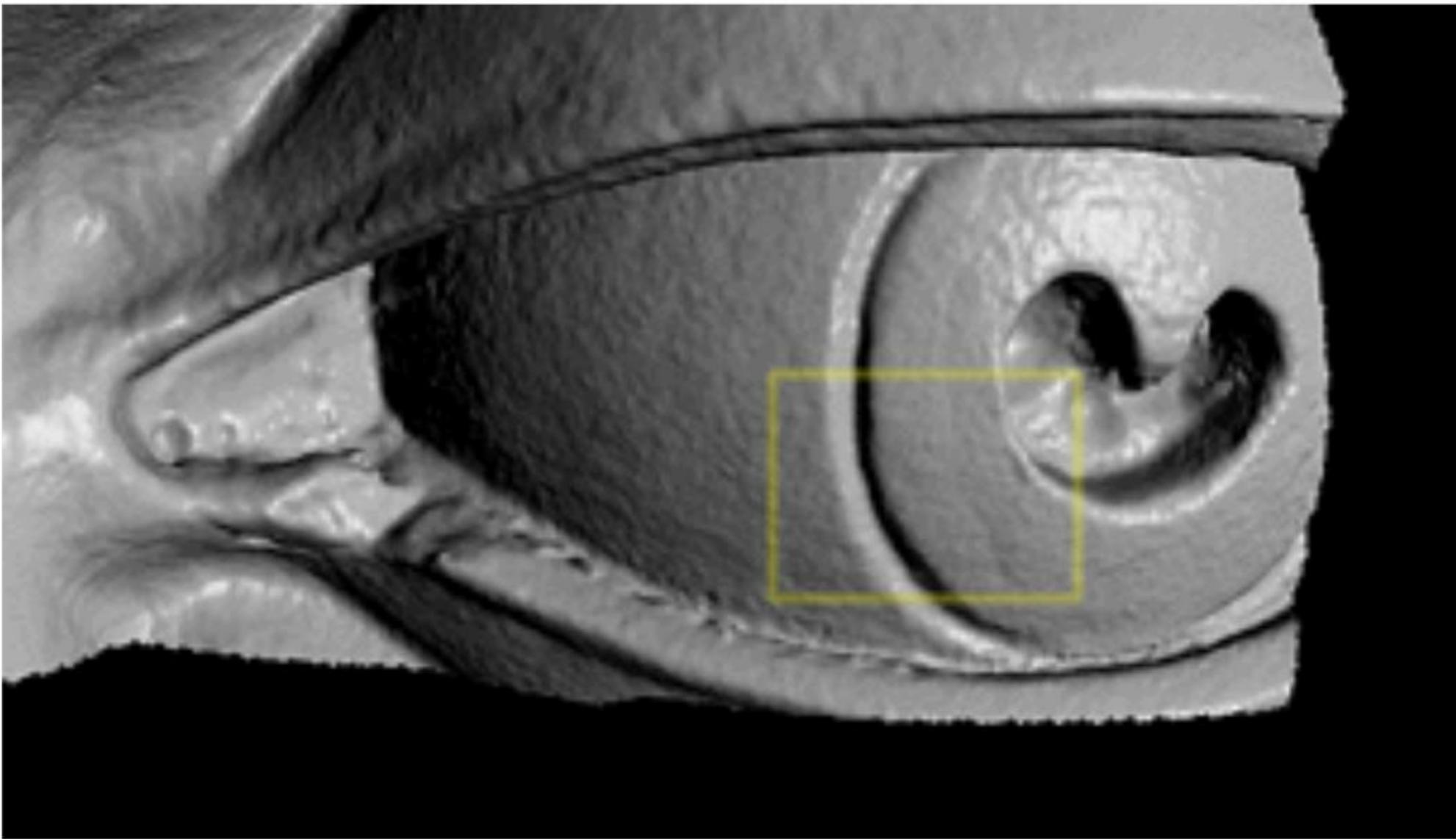
The Digital Michelangelo Project, Levoy et al.



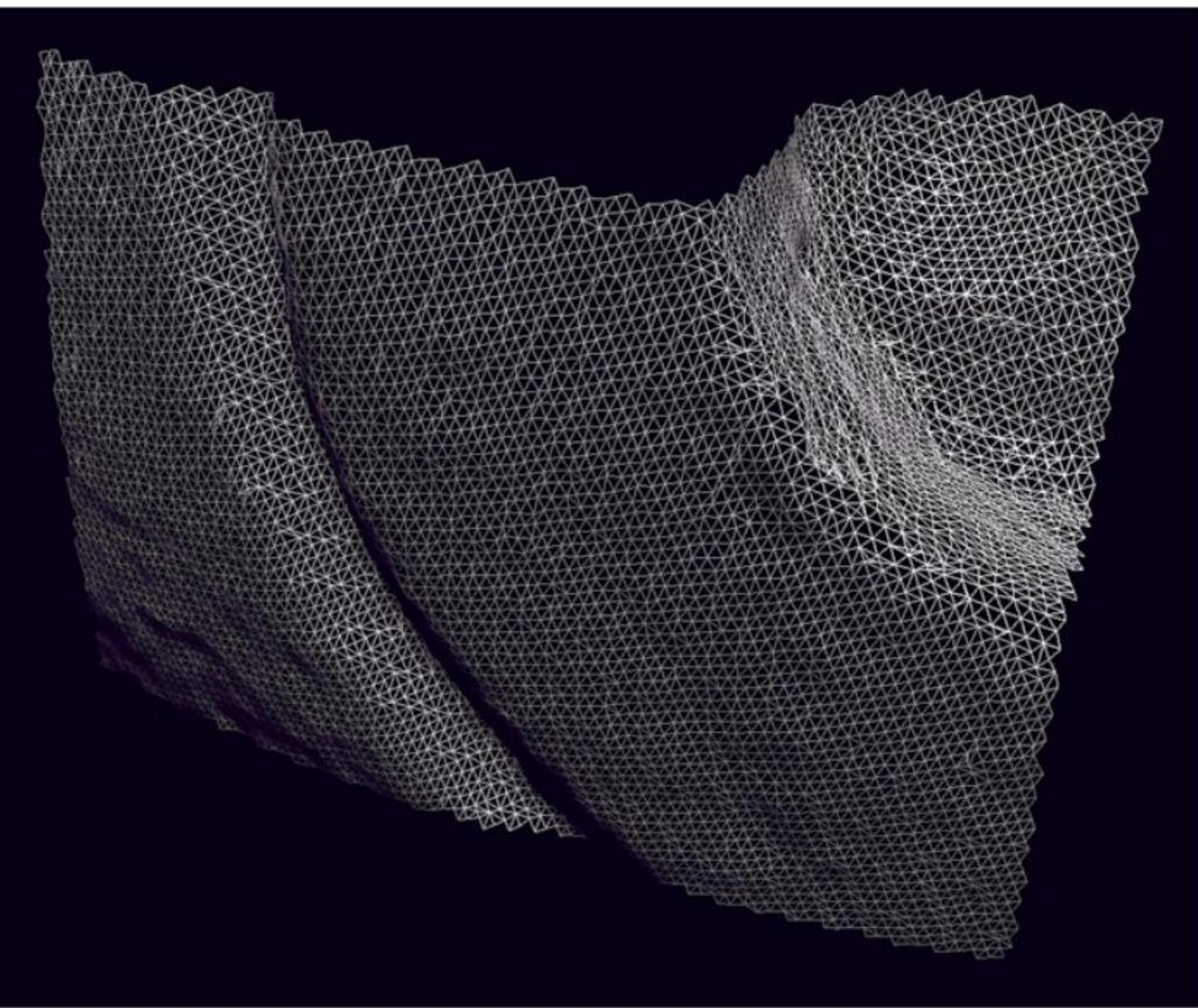
The Digital Michelangelo Project, Levoy et al.



The Digital Michelangelo Project, Levoy et al.



The Digital Michelangelo Project, Levoy et al.



The Digital Michelangelo Project, Levoy et al.

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.