

Stereo



Overview of today's lecture

- 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?

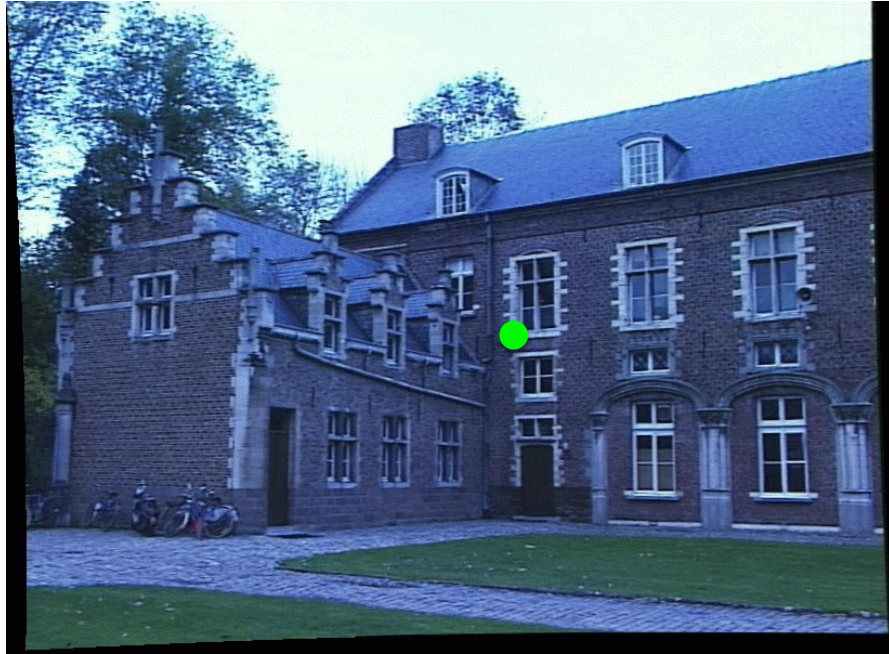


Left image



Right image

How would you reconstruct 3D points?



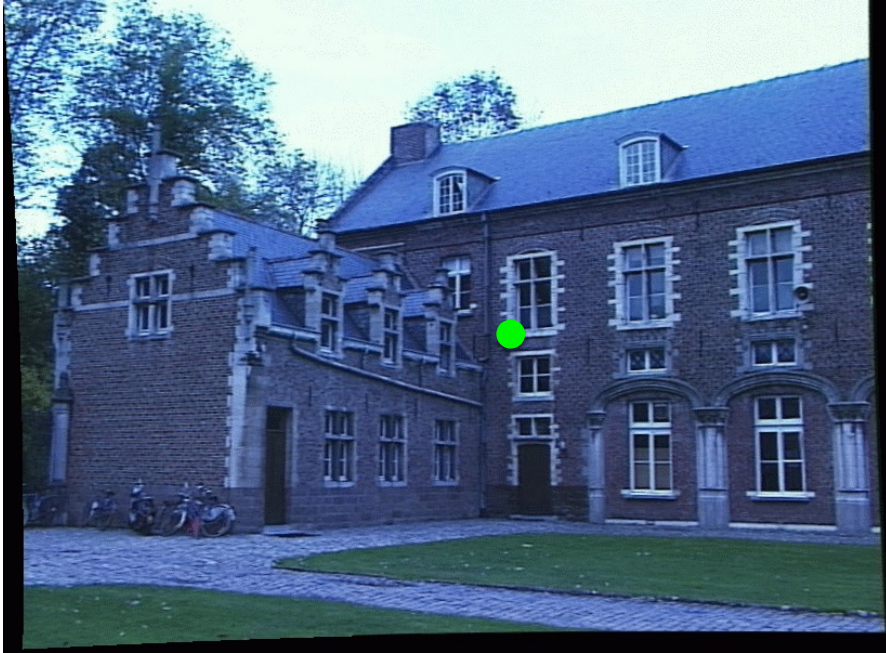
Left image



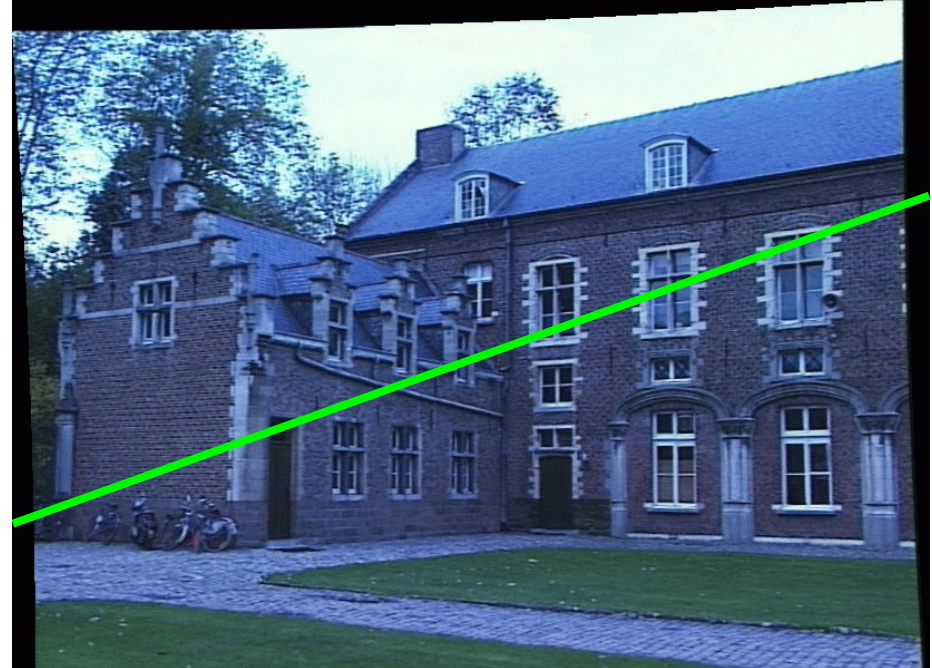
Right image

1. Select point in one image (how?)

How would you reconstruct 3D points?



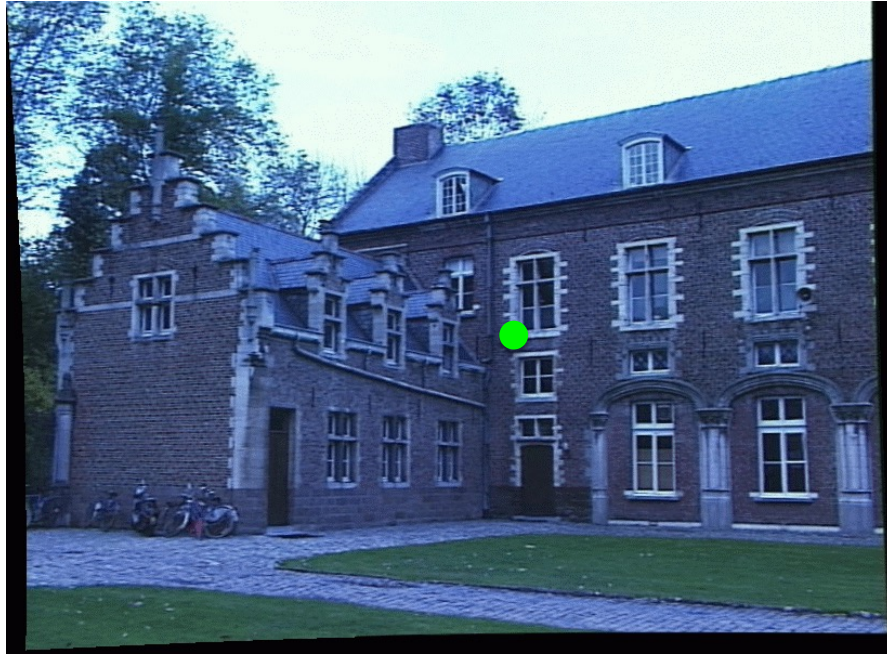
Left image



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?



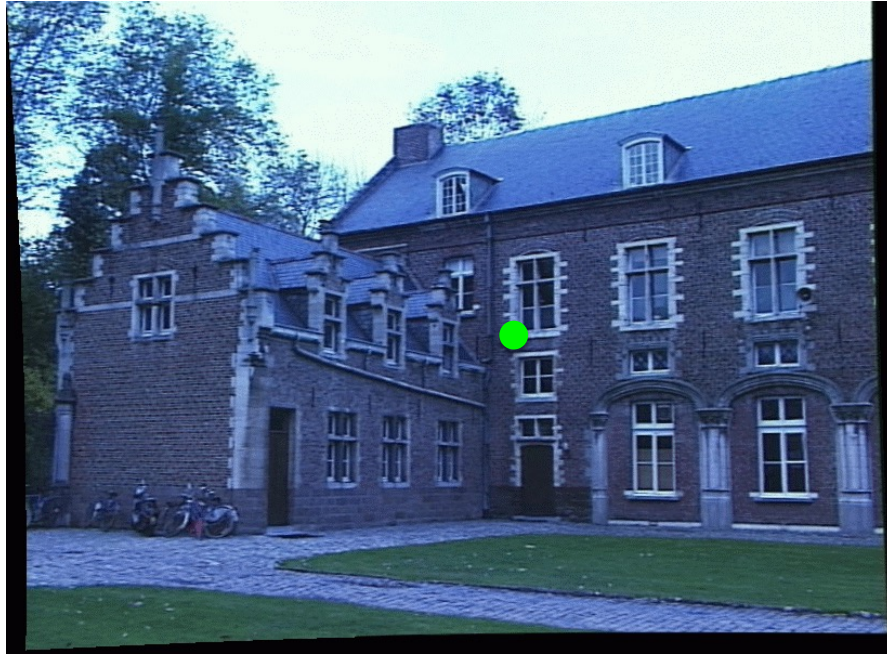
Left image



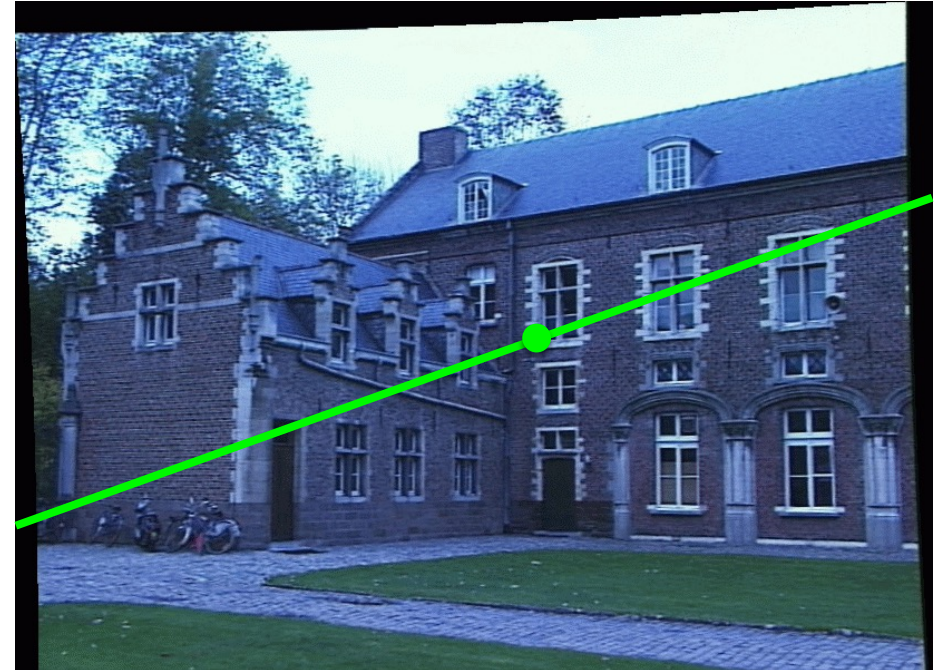
Right image

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?



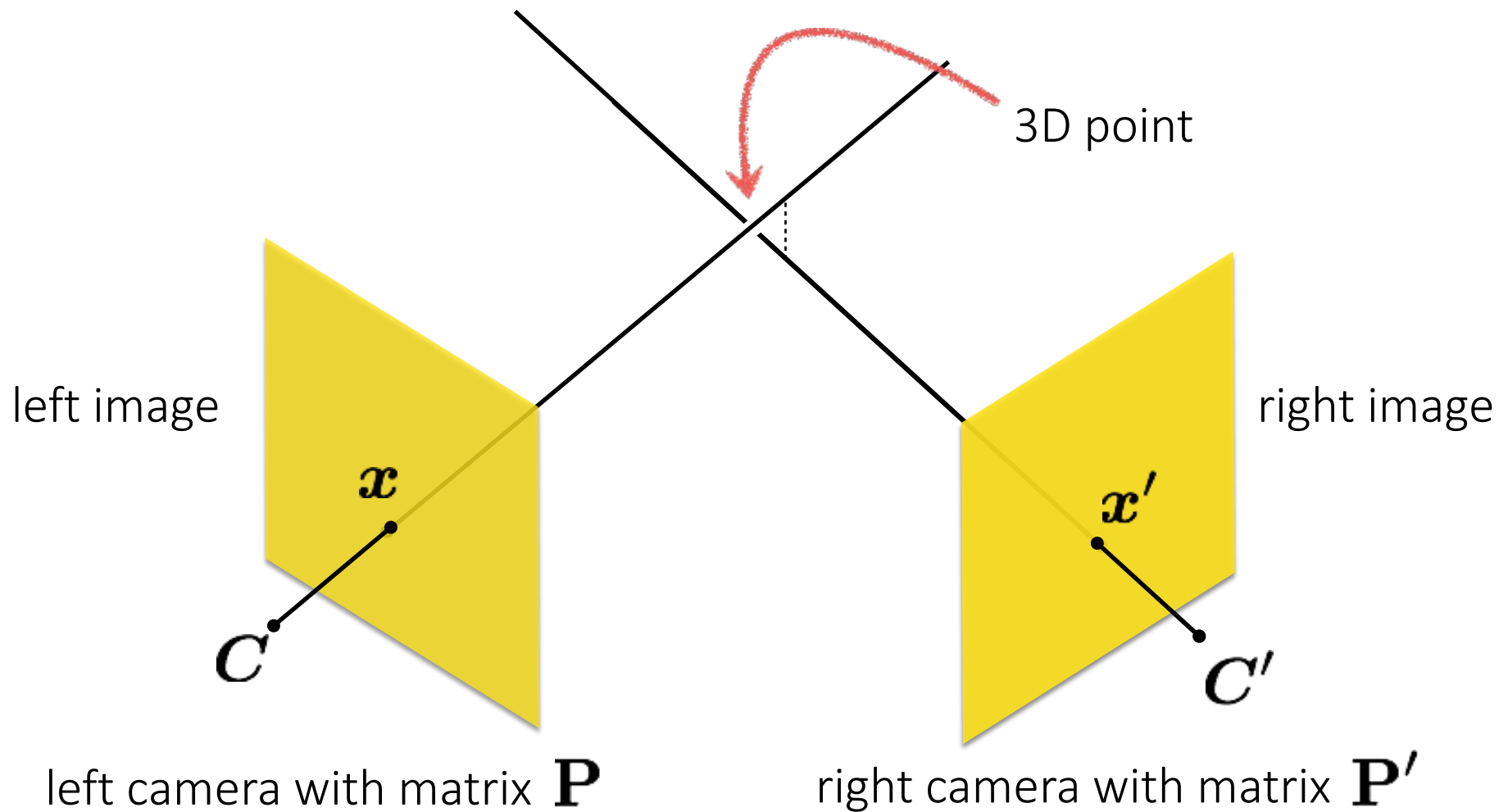
Left image



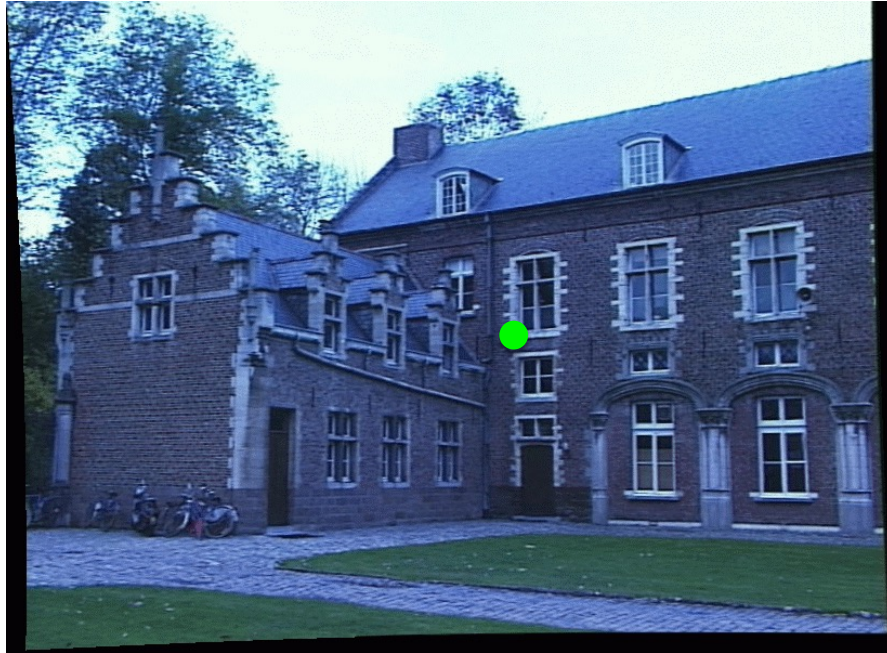
Right image

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?)

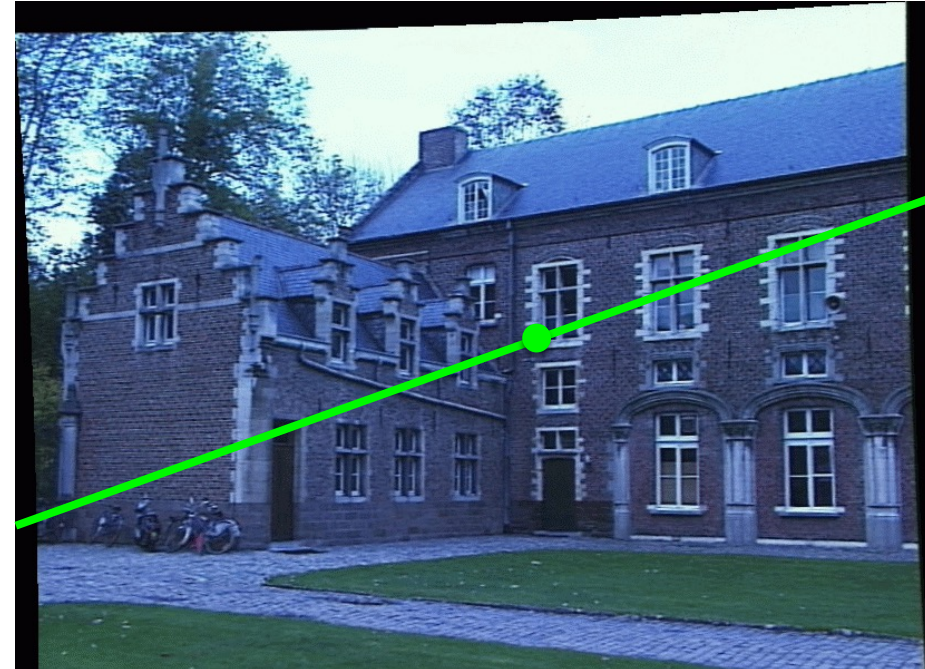
Triangulation



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?

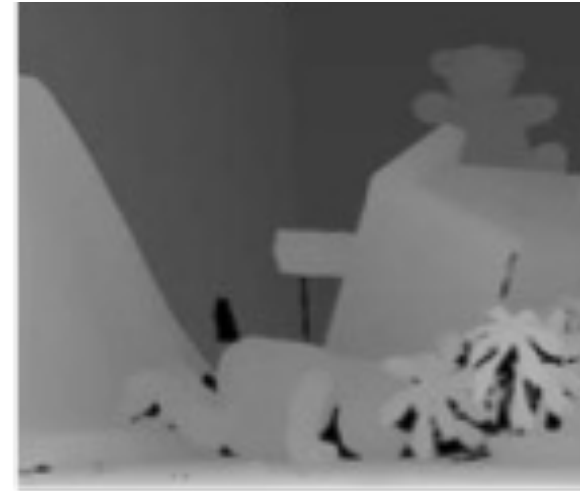




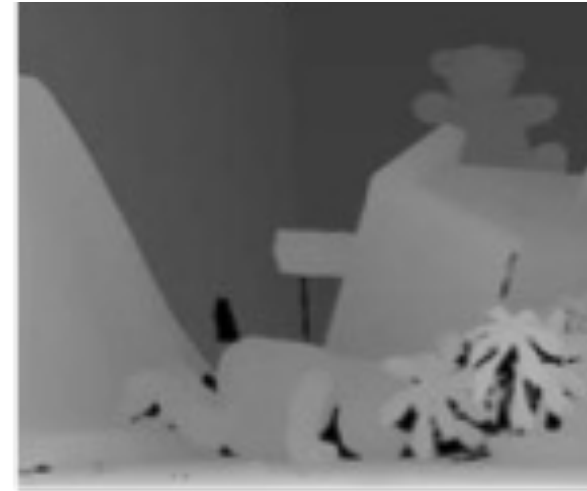


Objects that are close move more or less?

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

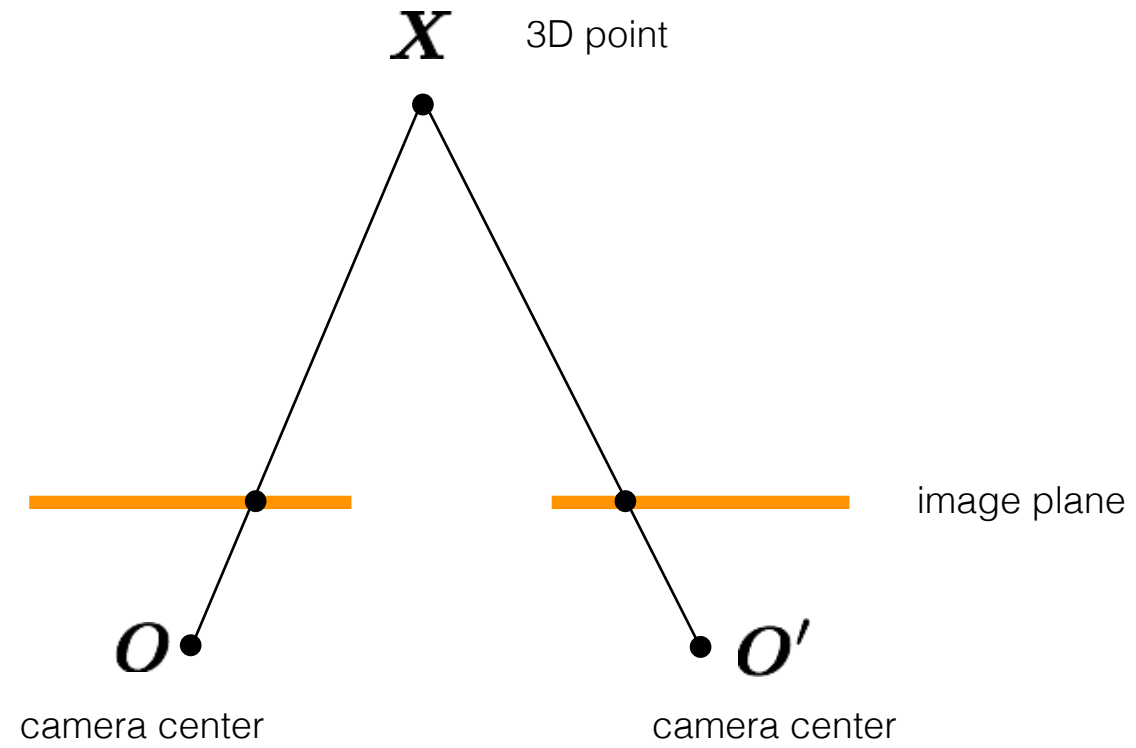


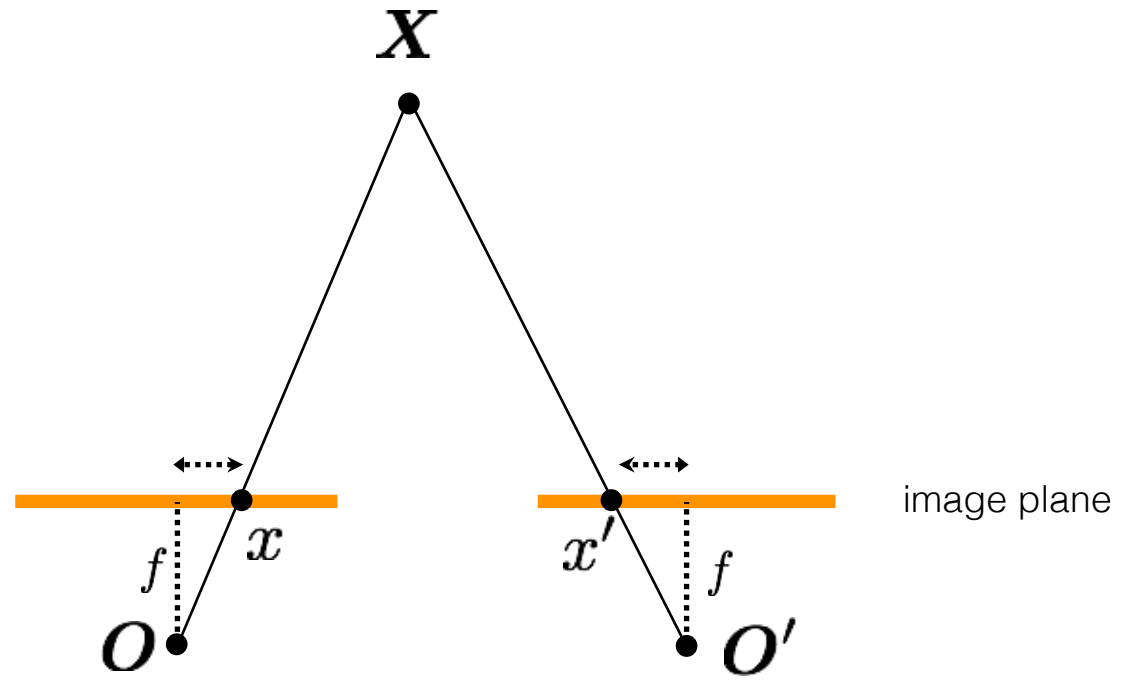
The amount of horizontal movement is
inversely proportional to ...

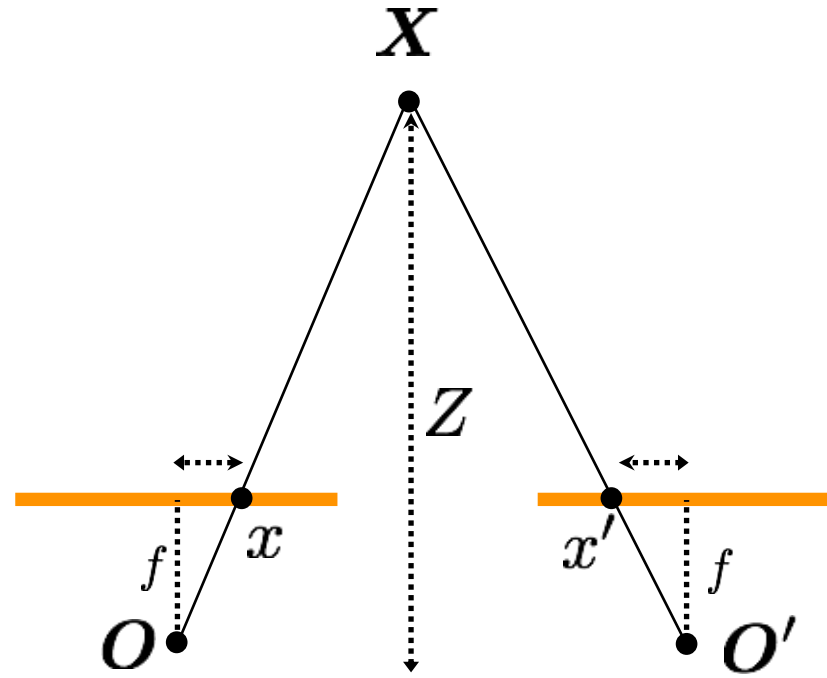


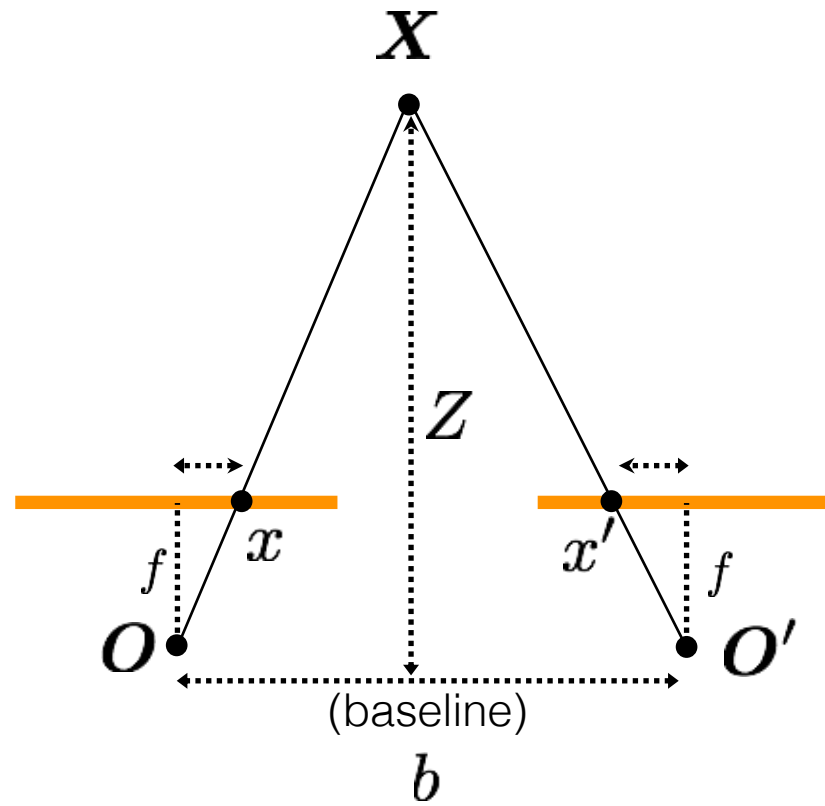
... the distance from the camera.

More formally...

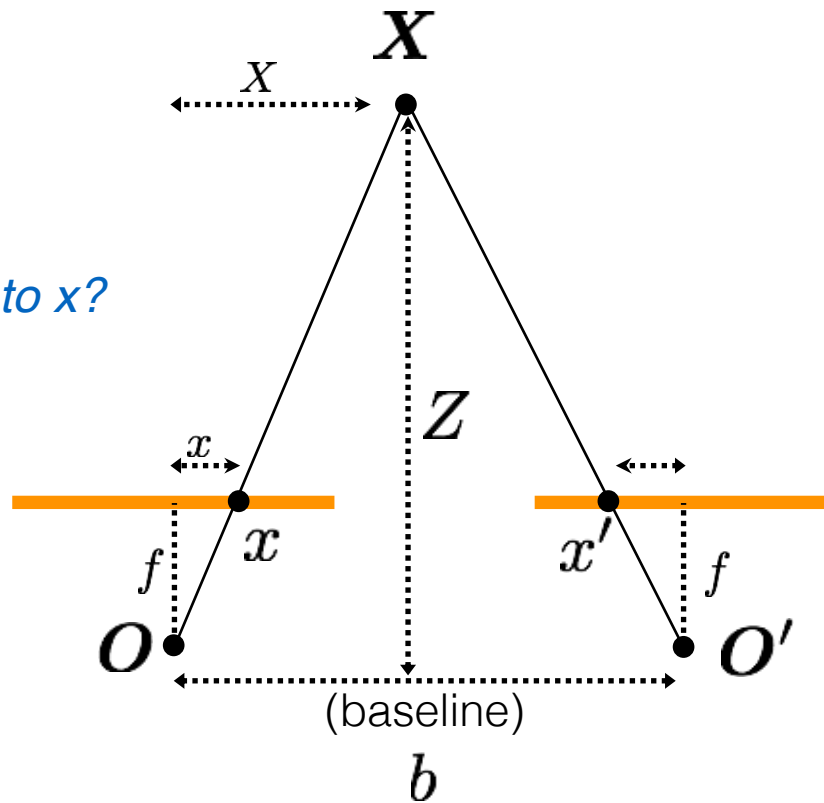




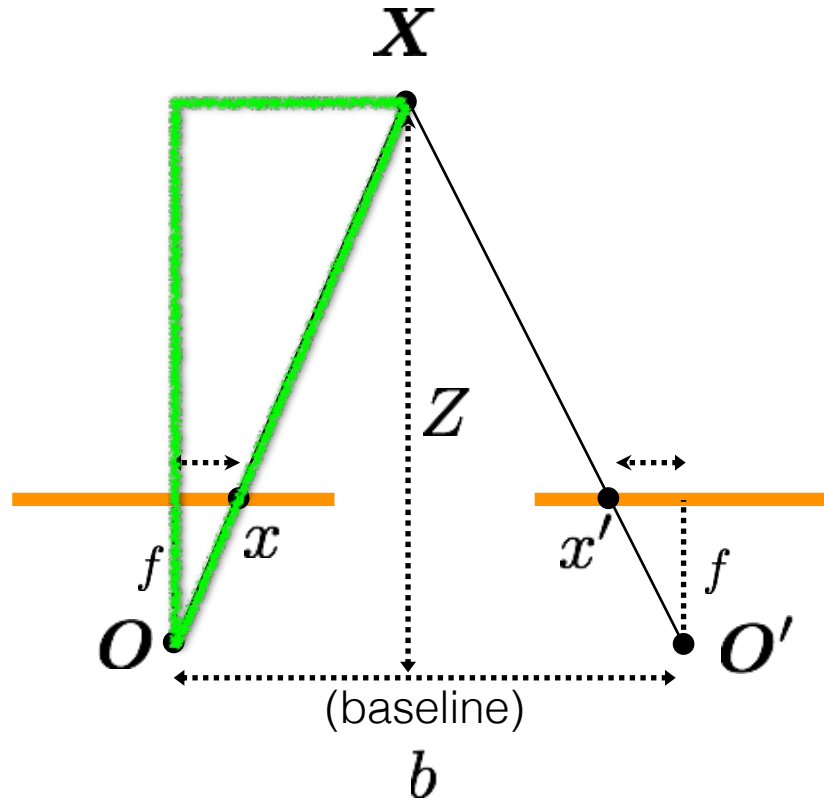




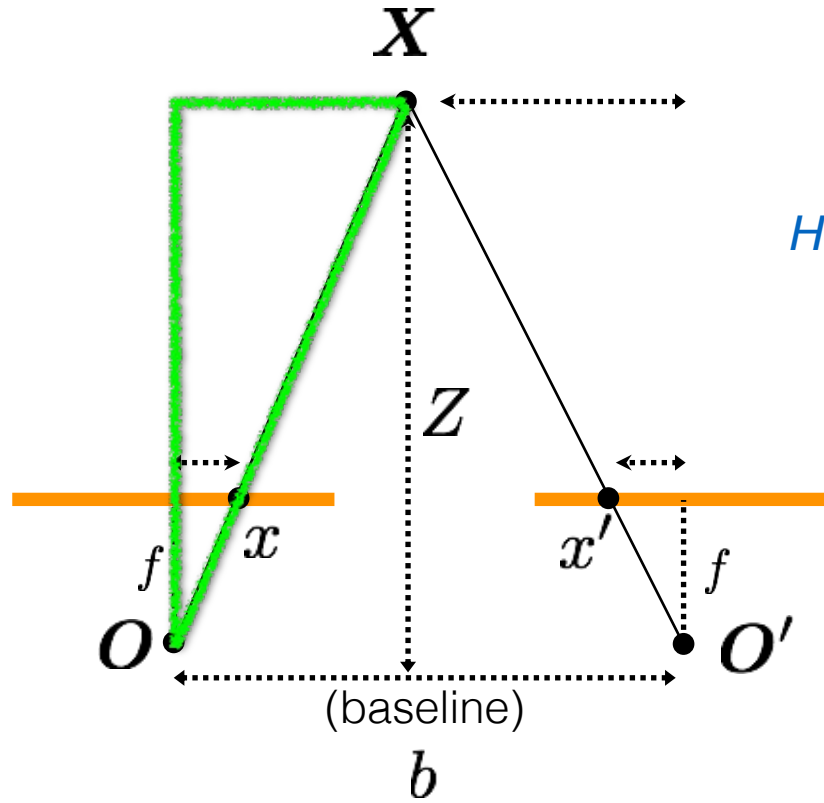
How is X related to x ?



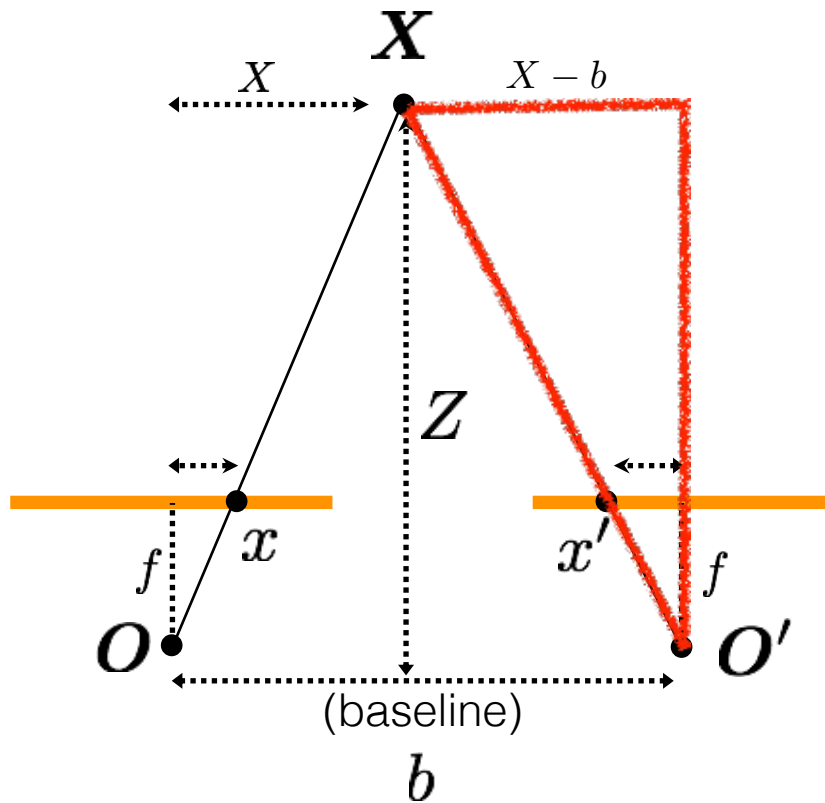
$$\frac{X}{Z} = \frac{x}{f}$$



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

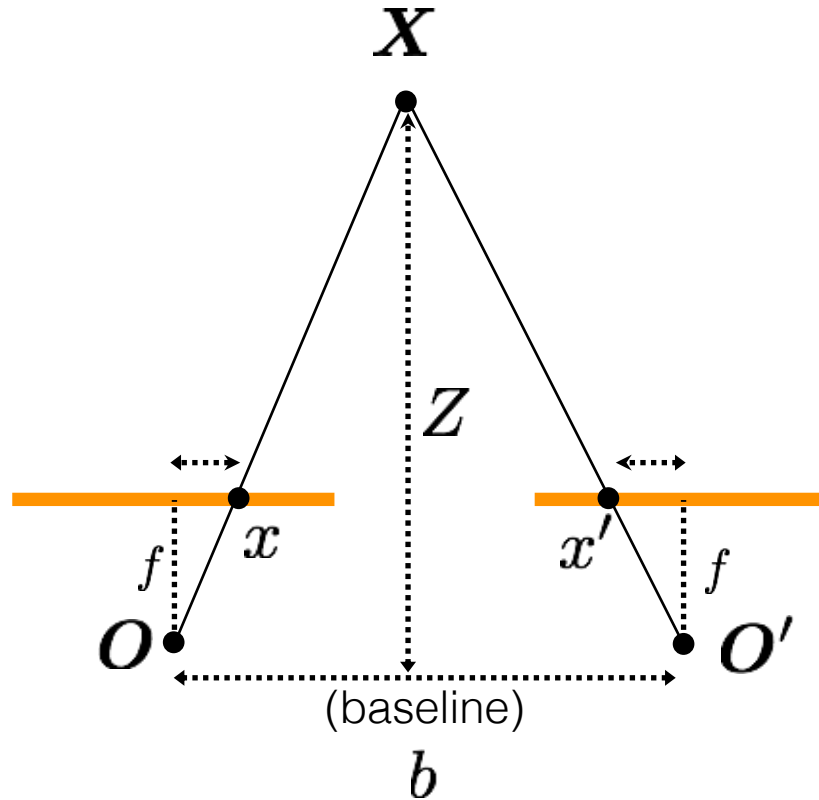


$$\frac{X}{Z} = \frac{x}{f}$$



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

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



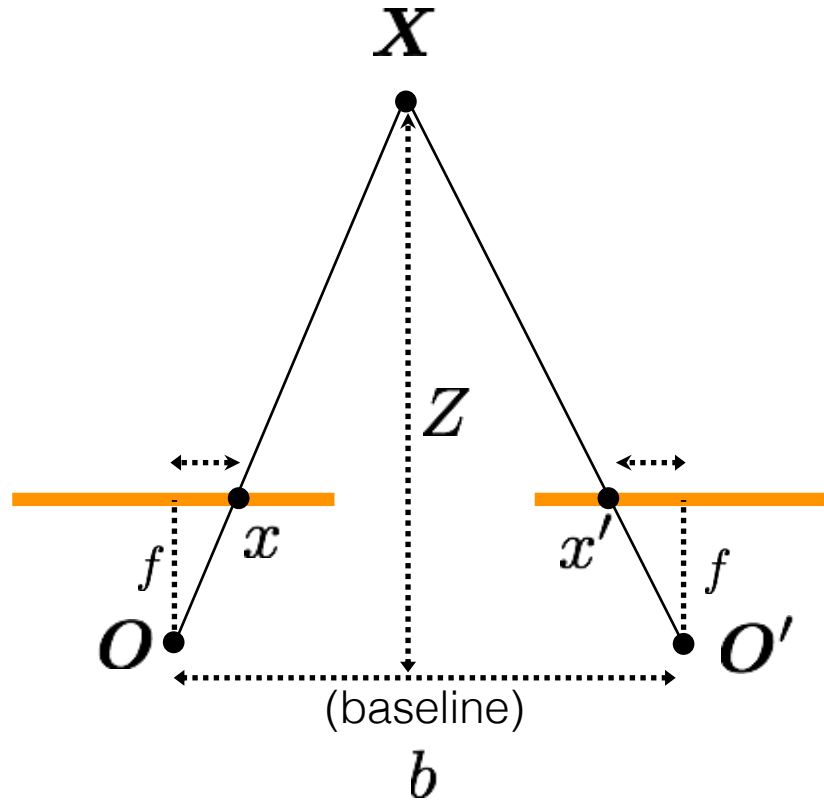
$$\frac{X - b}{Z} = \frac{x'}{f}$$

Disparity

$$d = x - x' \quad (\text{wrt to camera origin of image plane})$$

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

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



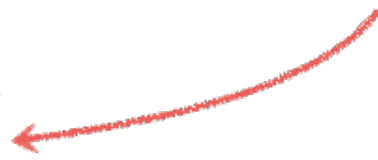
$$\frac{X - b}{Z} = \frac{x'}{f}$$

Disparity

$$d = x - x'$$

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

inversely proportional
to depth

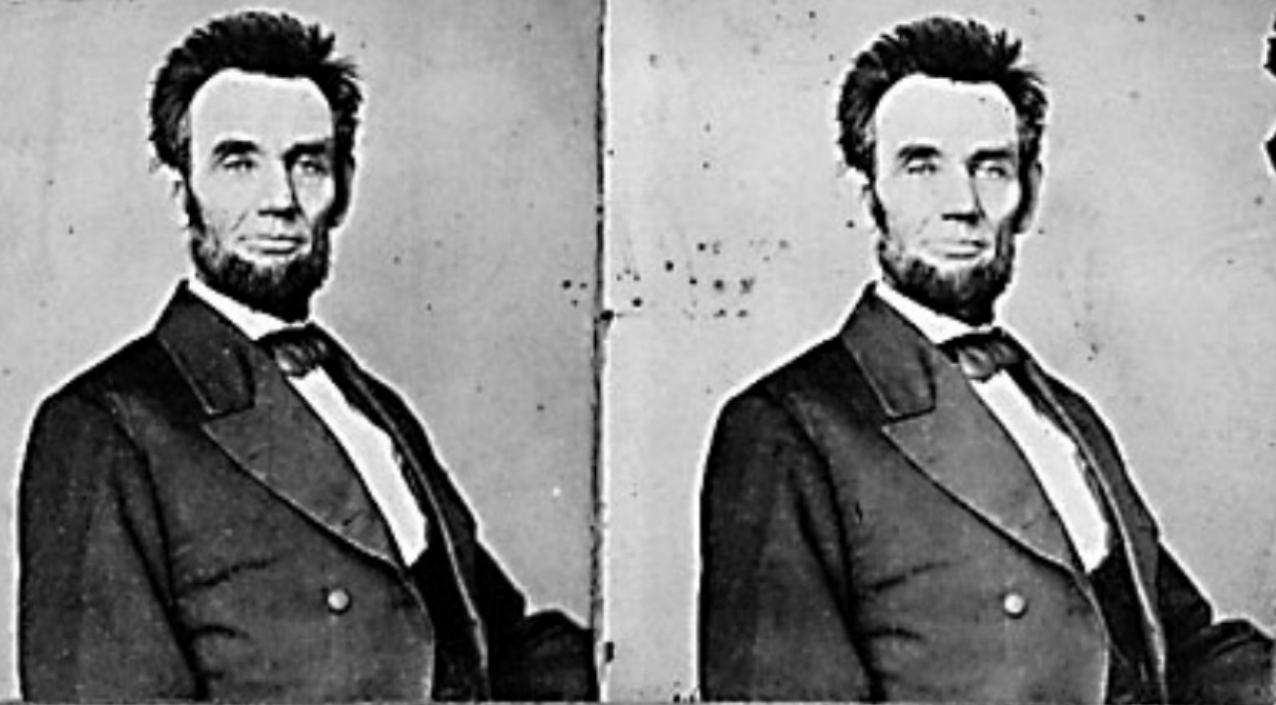


What other vision system uses disparity for depth sensing?

Stereoscopes: A 19th Century Pastime



HON. ABRAHAM LINCOLN, President of United States.



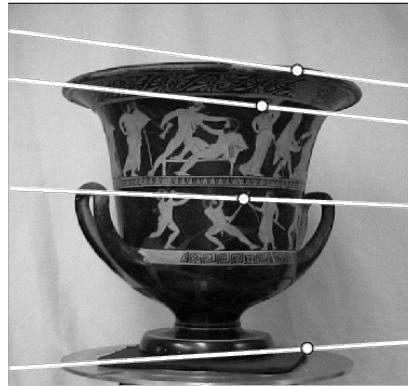


Public Library, Stereoscopic Looking Room, Chicago, by Phillips, 1923

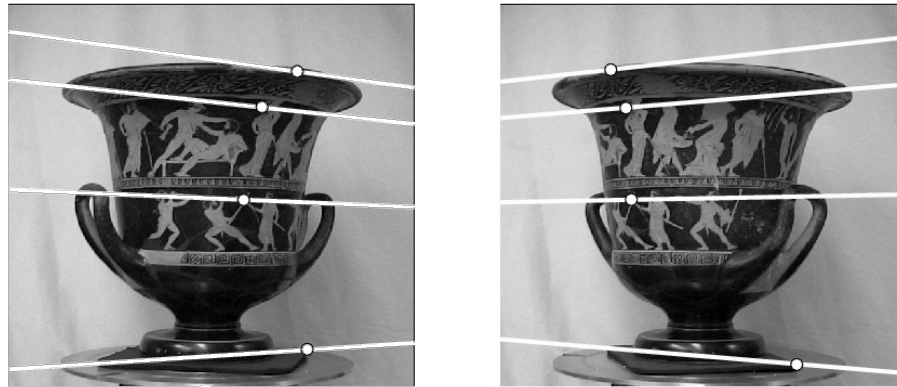


*Is disparity the only depth cue
the human visual system uses?*

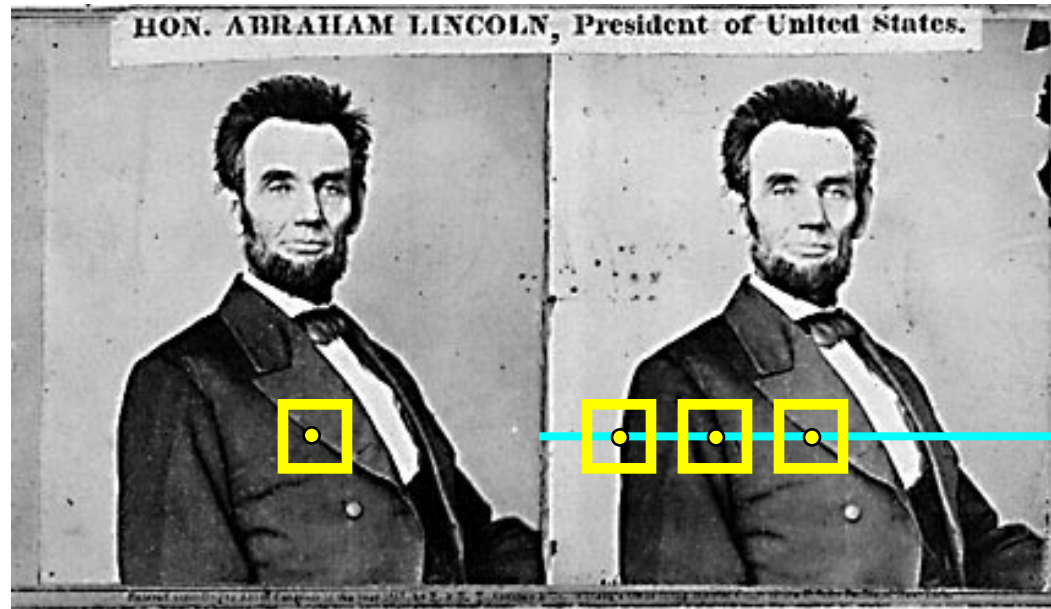
So can I compute depth from any two images of the same object?



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

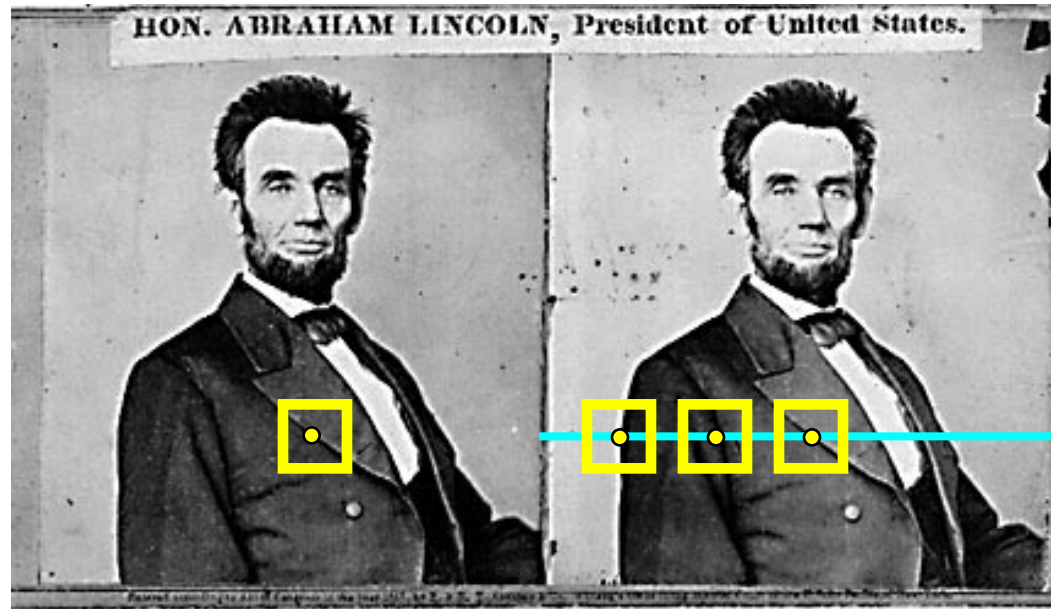


1. Need sufficient baseline
2. Images need to be 'rectified' first (make epipolar lines horizontal)

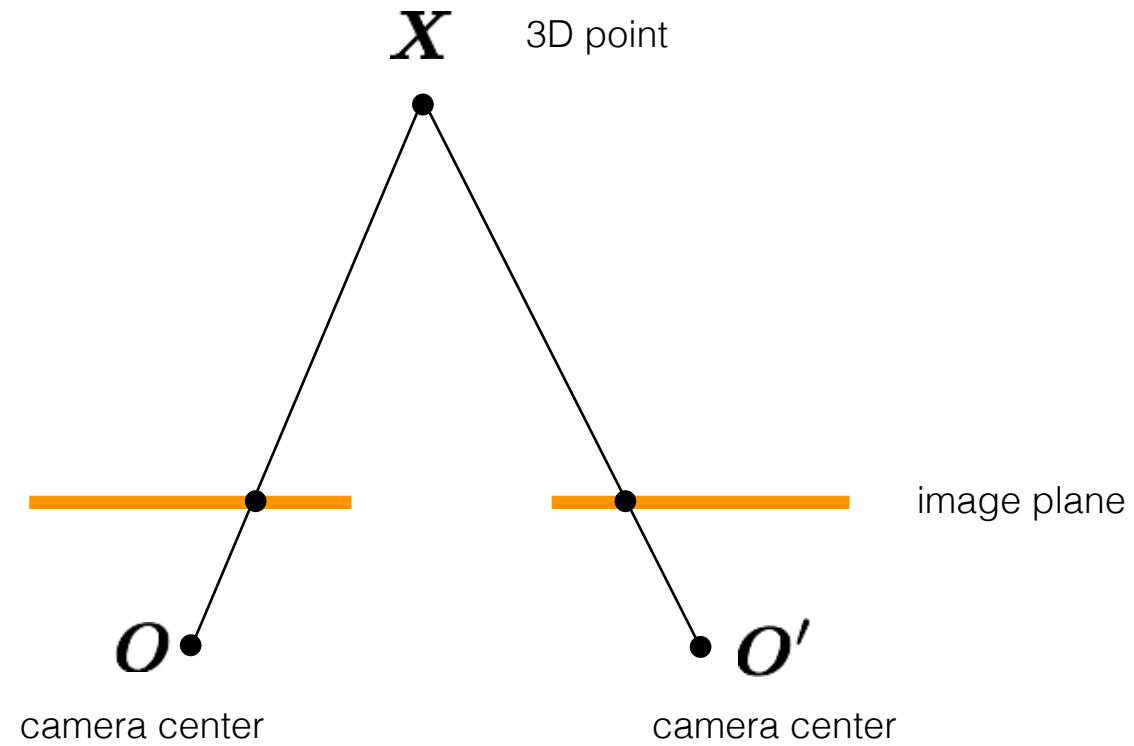


1. Rectify images
(make epipolar lines horizontal)
2. For each pixel
 - a. Find epipolar line
 - b. Scan line for best match
 - c. Compute depth from disparity

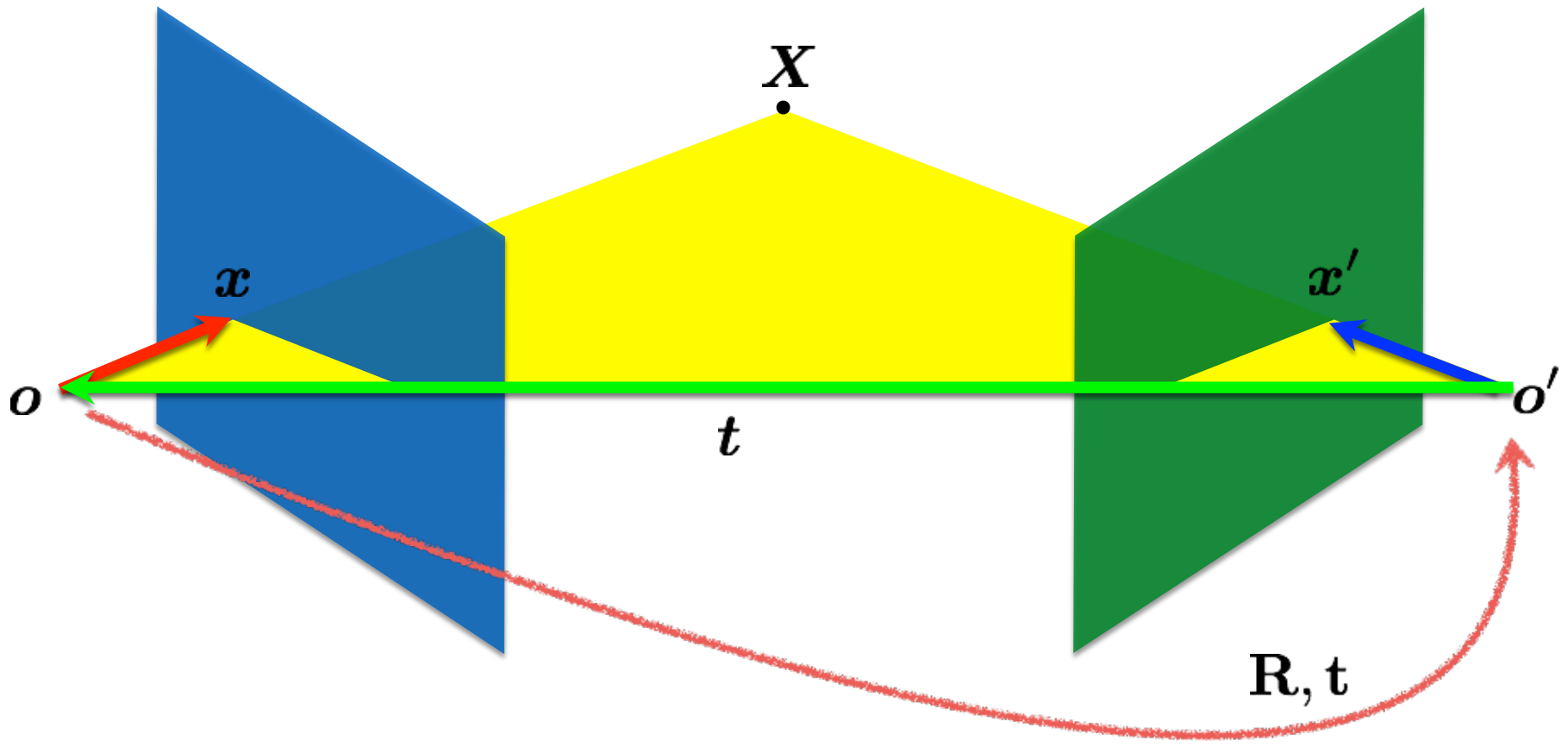
$$Z = \frac{bf}{d}$$



How can you make the epipolar lines horizontal?

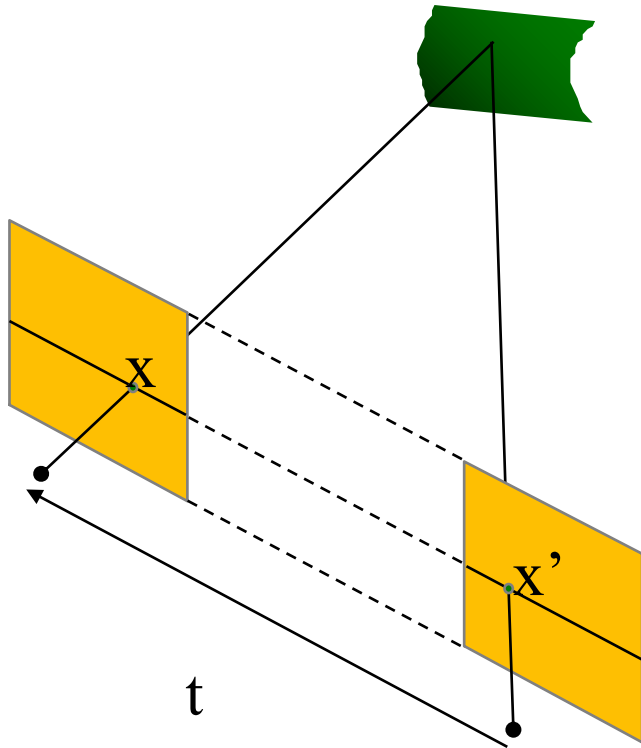


What's special about these two cameras?



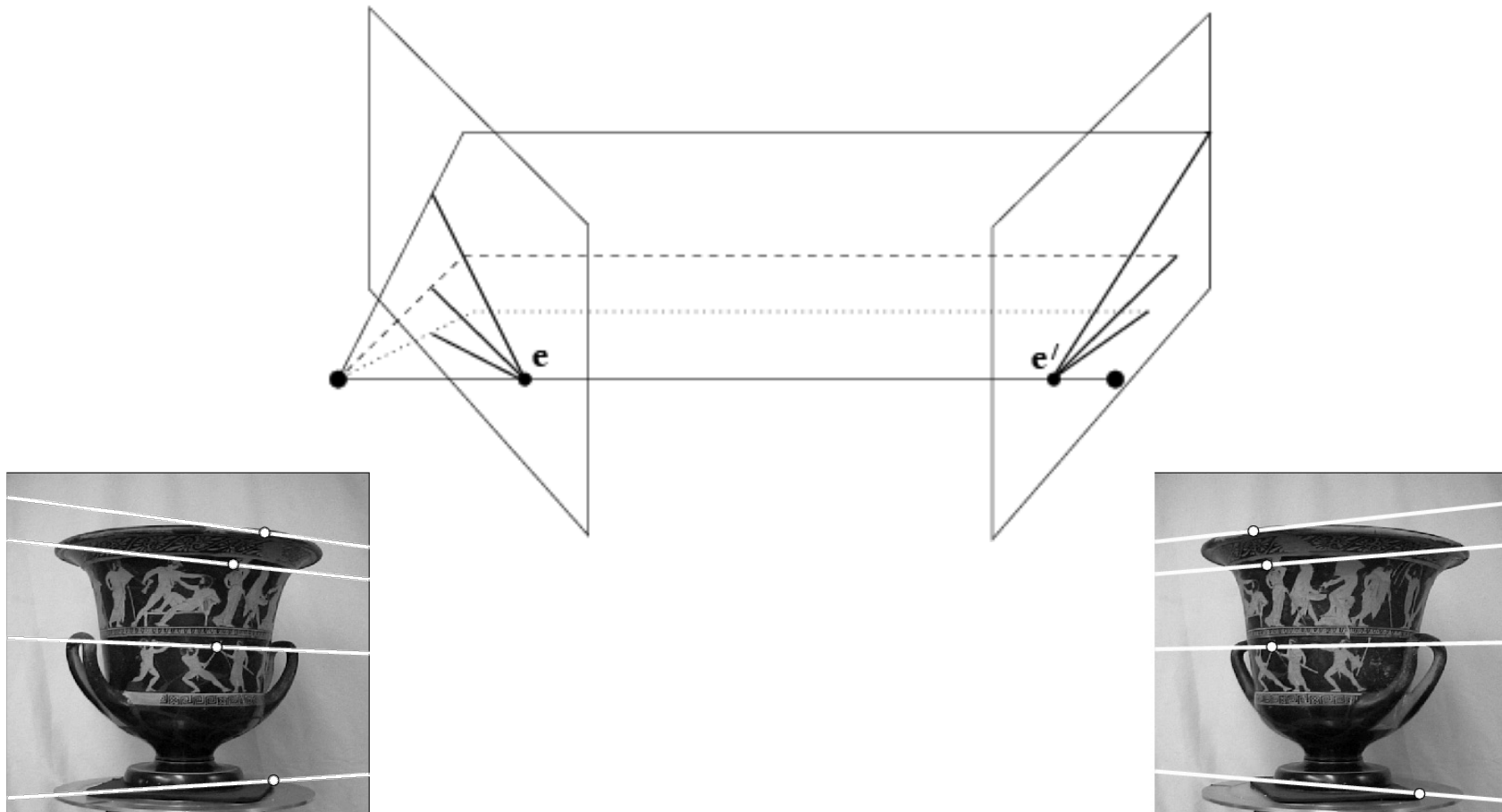
$$x' = \mathbf{R}(x - t)$$

When are epipolar lines horizontal?



When this relationship holds:

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



It's hard to make the image planes exactly parallel



How can you make the epipolar lines horizontal?

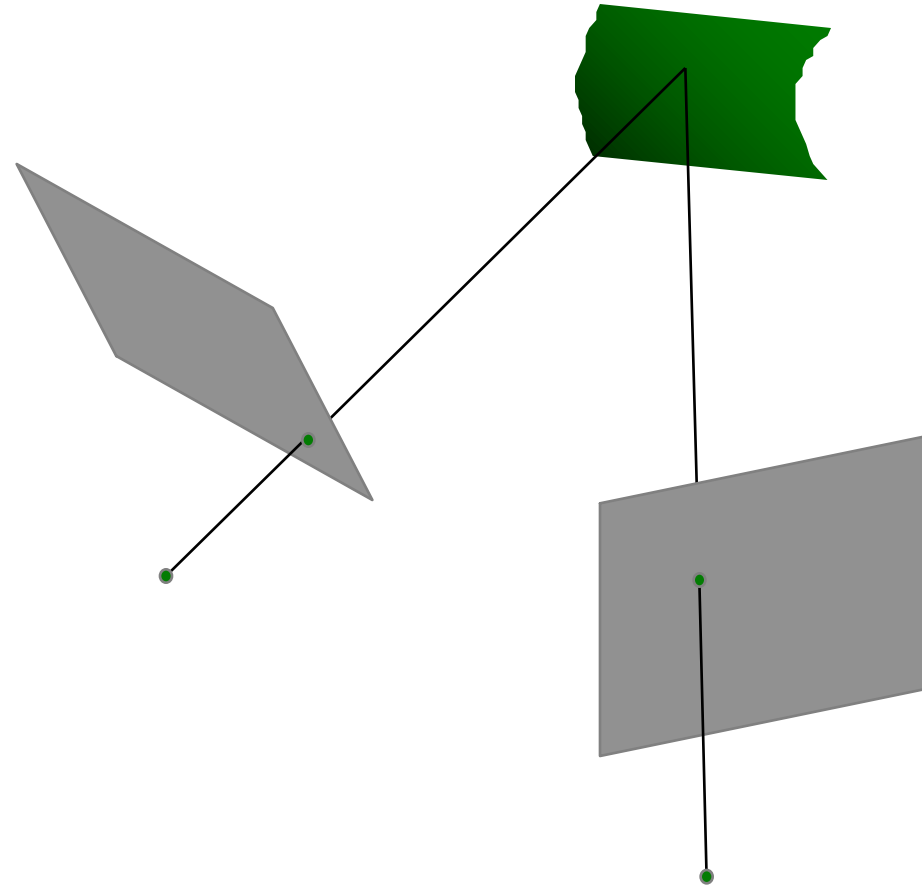




Use stereo rectification?

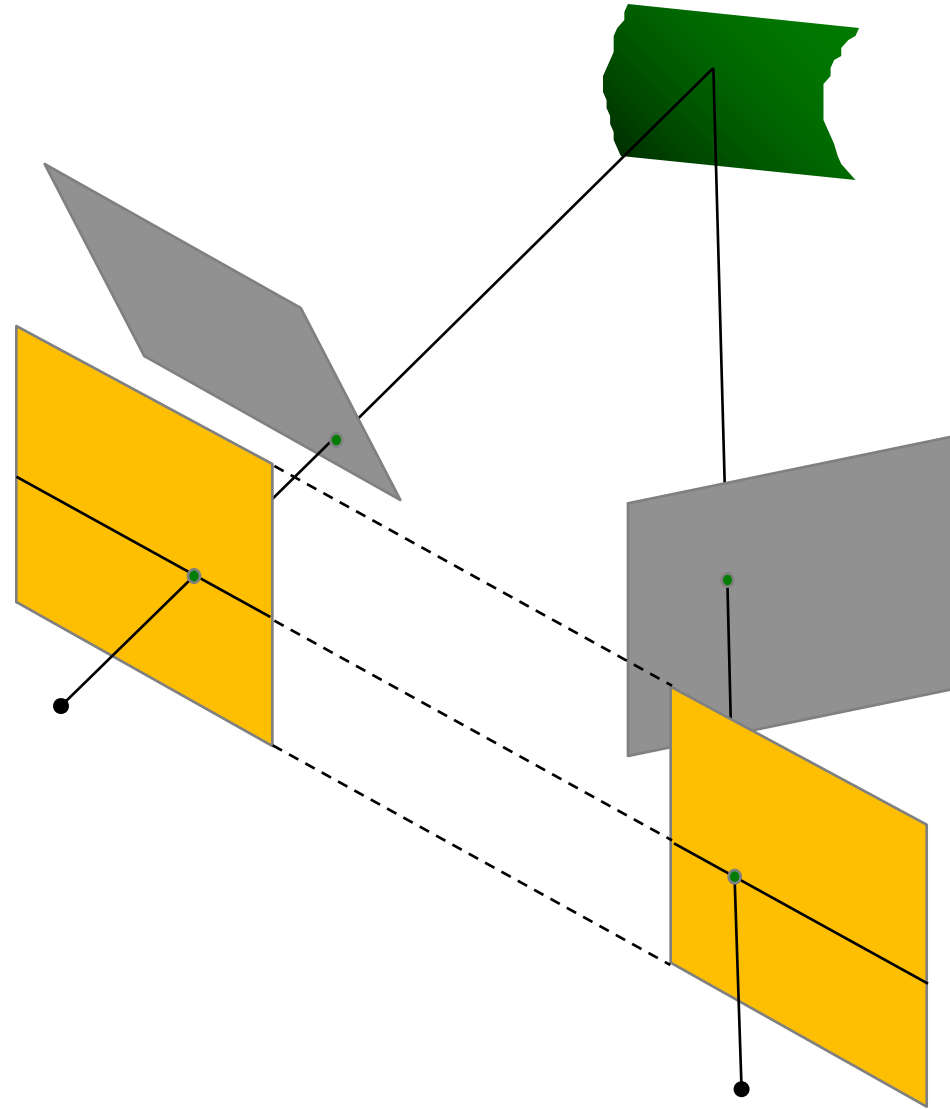


What is stereo rectification?



What is stereo rectification?

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

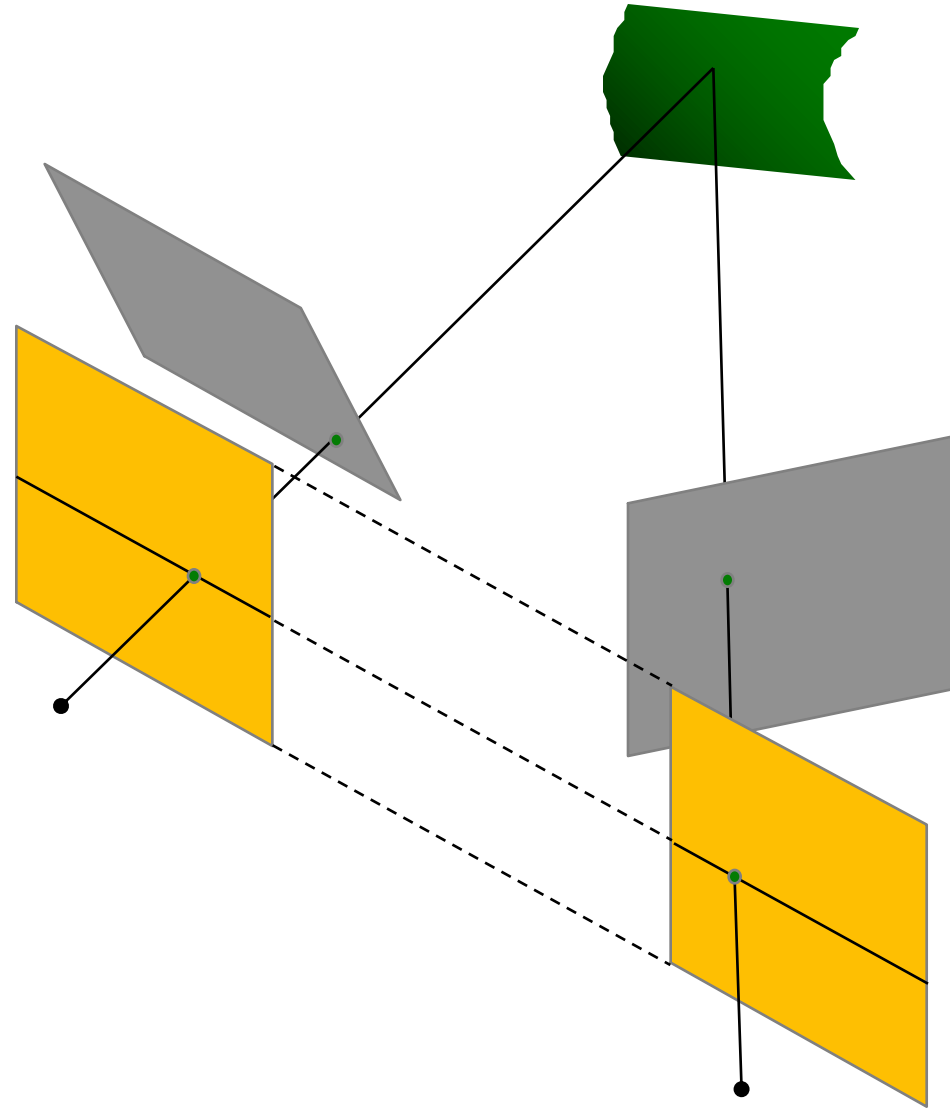


How can you do this?

What is stereo rectification?

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

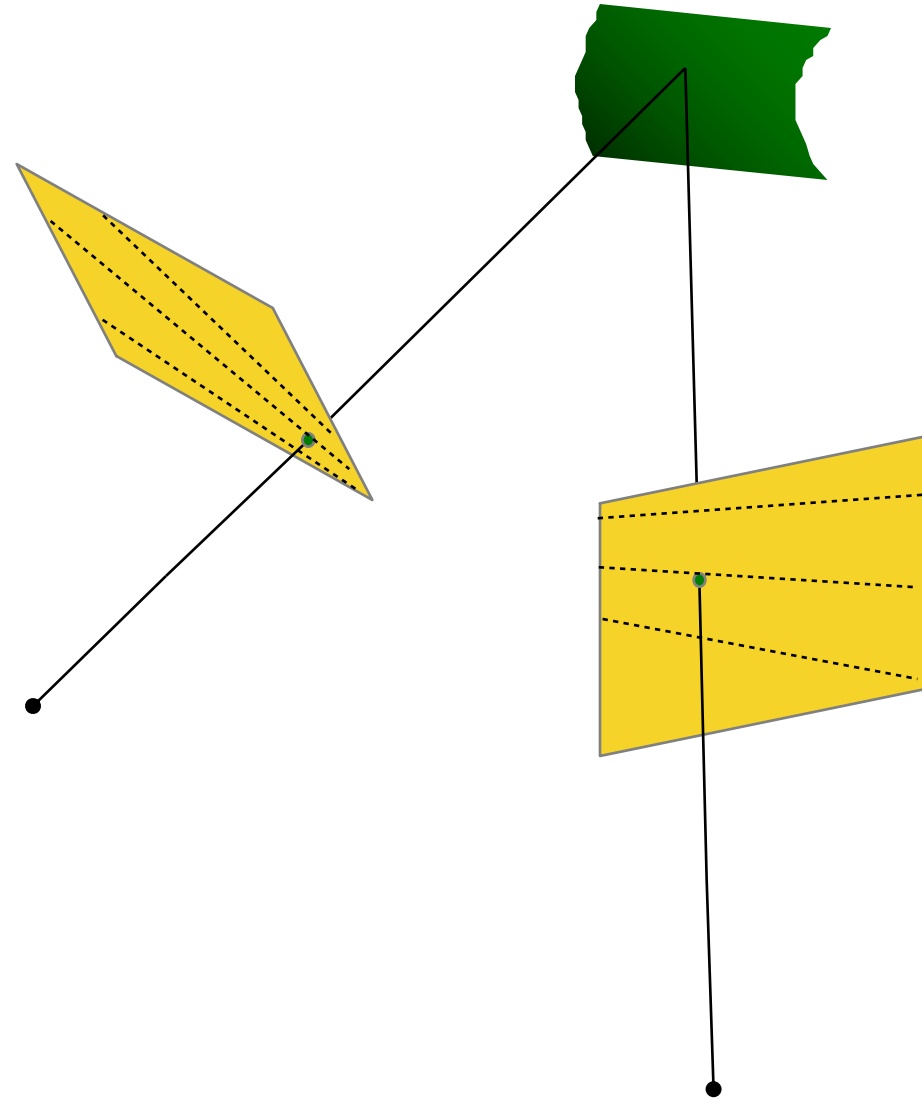
Need two homographies (3x3 transform), one for each input image reprojection



Stereo Rectification

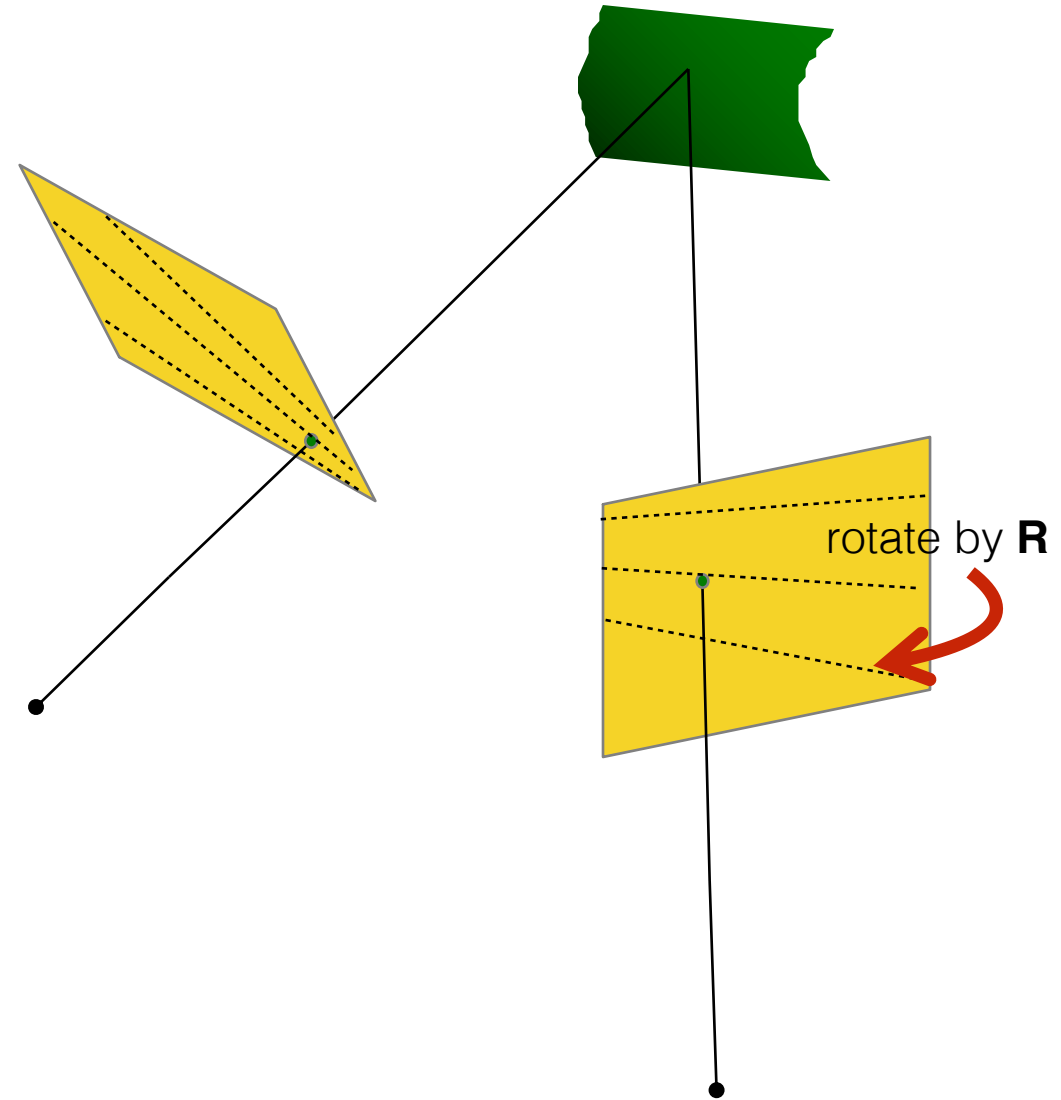
1. **Rotate** the right camera by **R**
(aligns camera coordinate system orientation only)
2. Rotate (**rectify**) the left camera so that the epipole is at infinity
3. Rotate (**rectify**) the right camera so that the epipole is at infinity
4. Adjust the **scale**

Stereo Rectification:



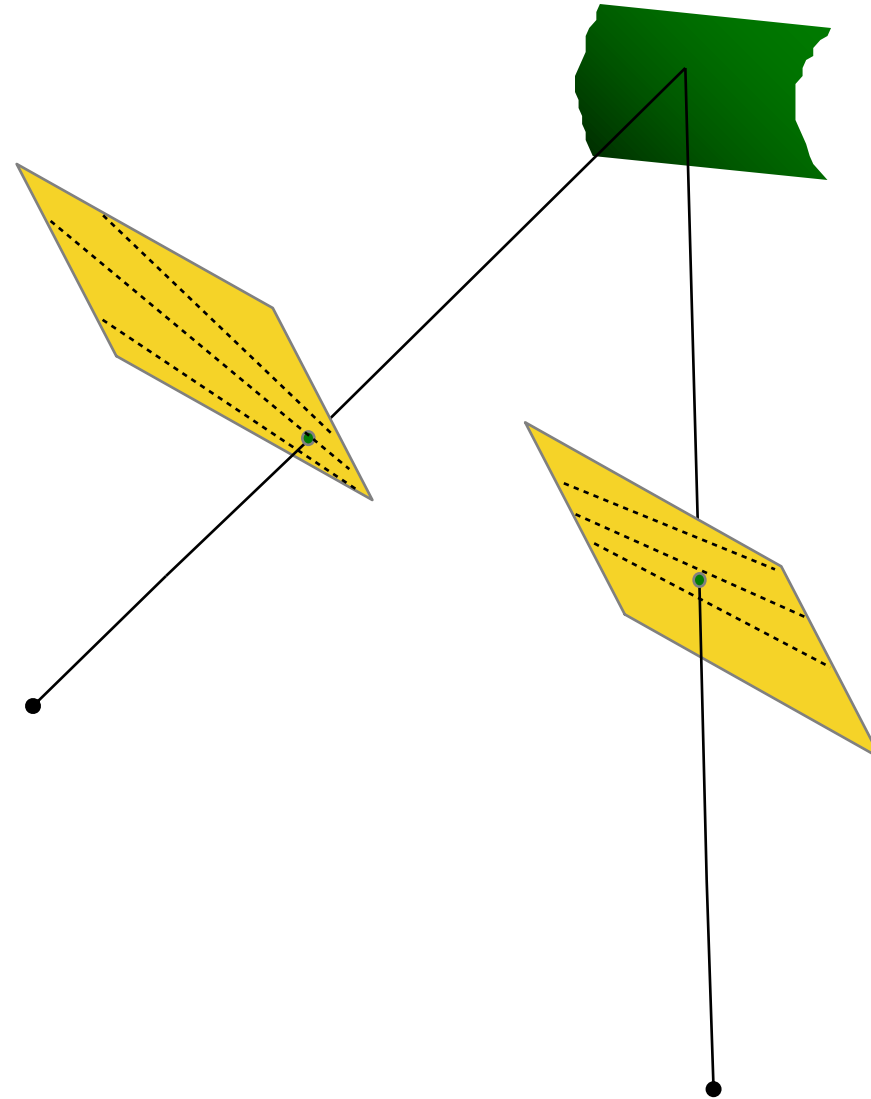
1. Compute \mathbf{E} to get \mathbf{R}
2. Rotate right image by \mathbf{R}
3. Rotate both images by \mathbf{R}_{rect}
4. Scale both images by \mathbf{H}

Stereo Rectification:



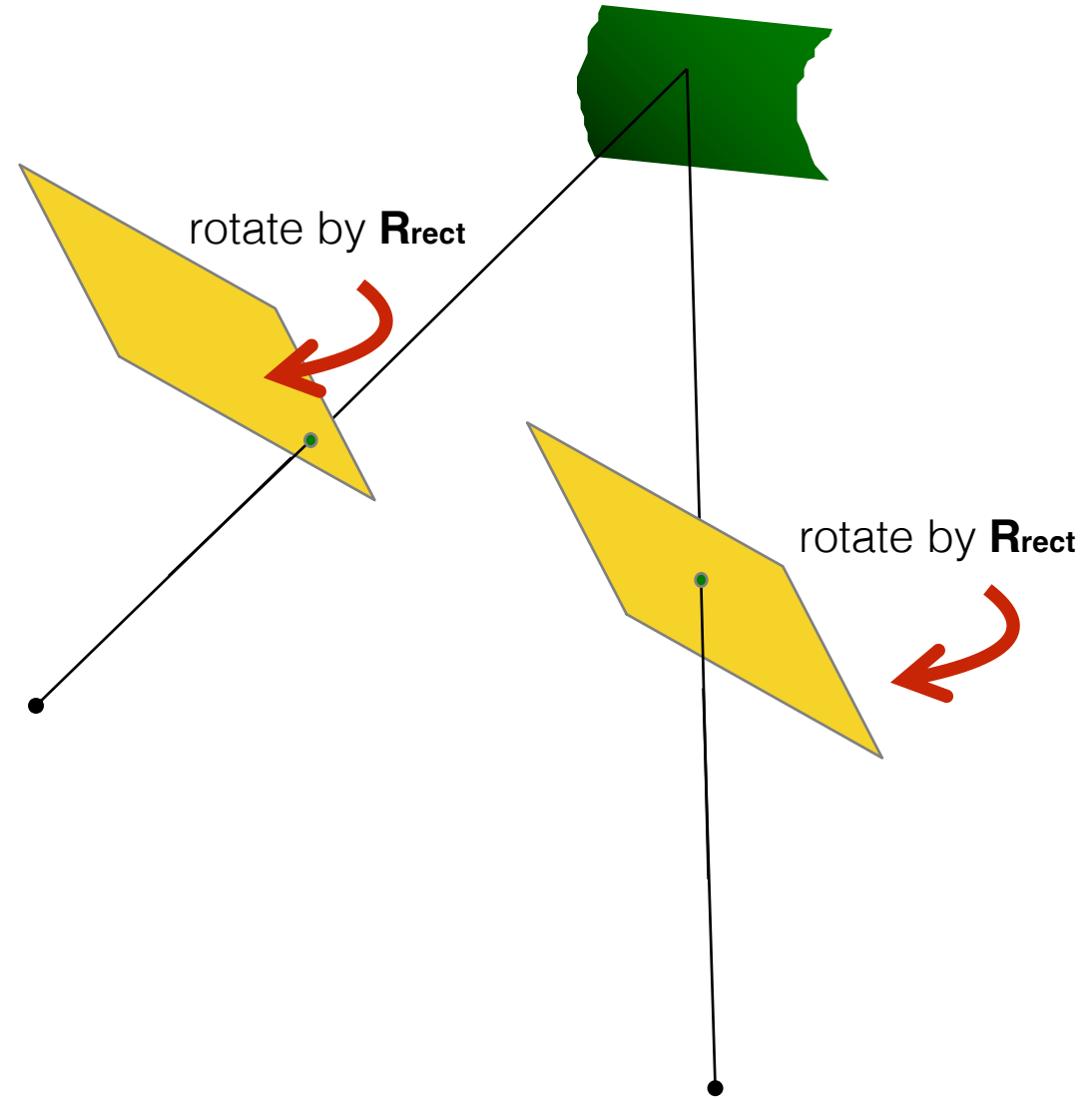
1. Compute \mathbf{E} to get \mathbf{R}
2. Rotate right image by \mathbf{R}
3. Rotate both images by \mathbf{R}_{rect}
4. Scale both images by \mathbf{H}

Stereo Rectification:



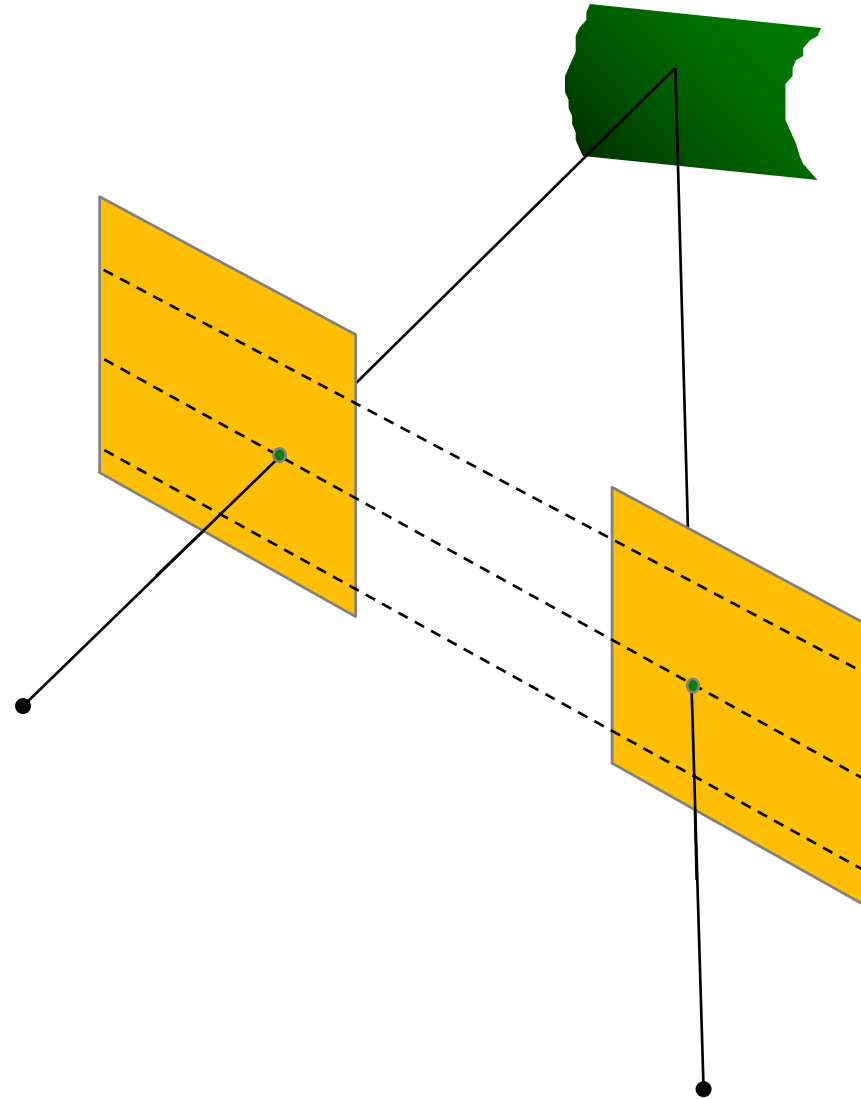
1. Compute \mathbf{E} to get \mathbf{R}
2. Rotate right image by \mathbf{R}
3. Rotate both images by \mathbf{R}_{rect}
4. Scale both images by \mathbf{H}

Stereo Rectification:



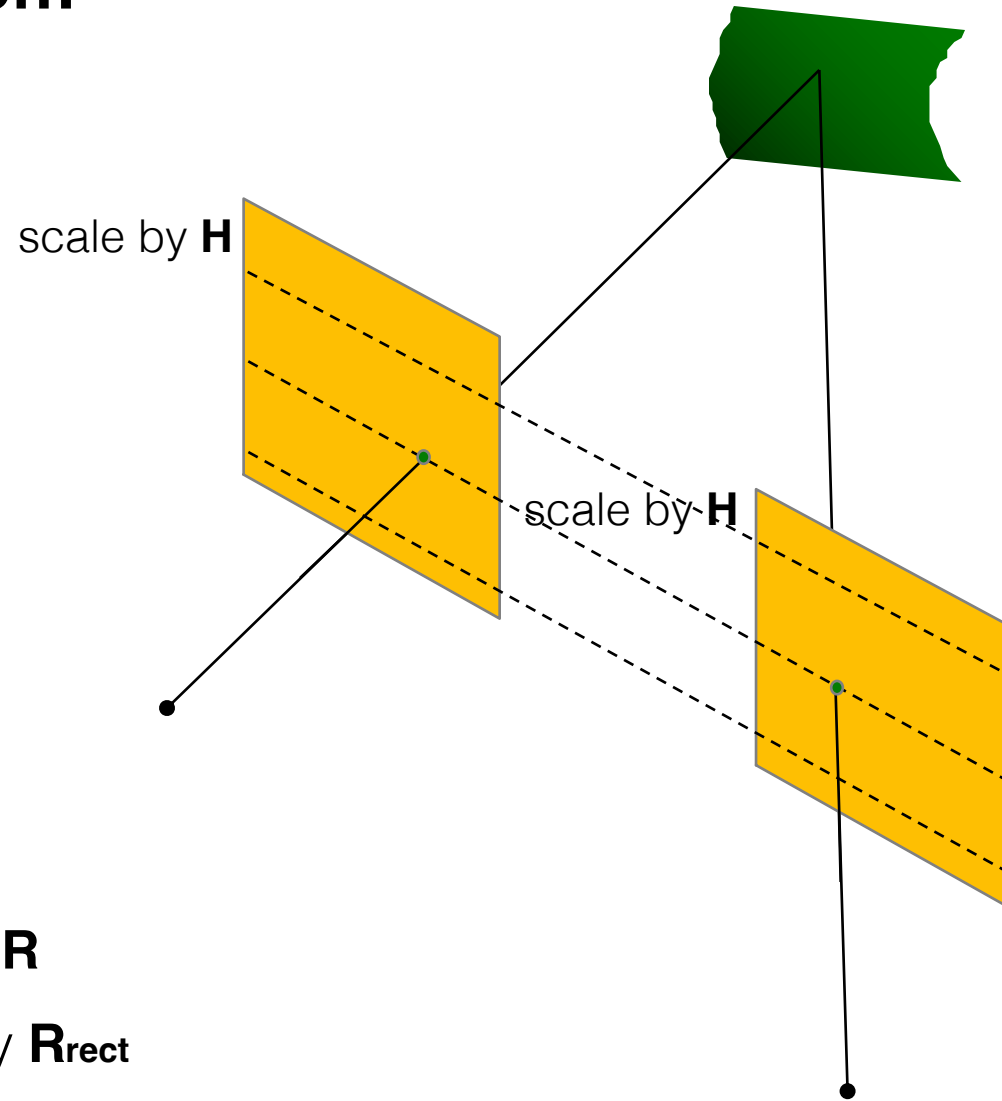
1. Compute \mathbf{E} to get \mathbf{R}
2. Rotate right image by \mathbf{R}
3. Rotate both images by \mathbf{R}_{rect}
4. Scale both images by \mathbf{H}

Stereo Rectification:



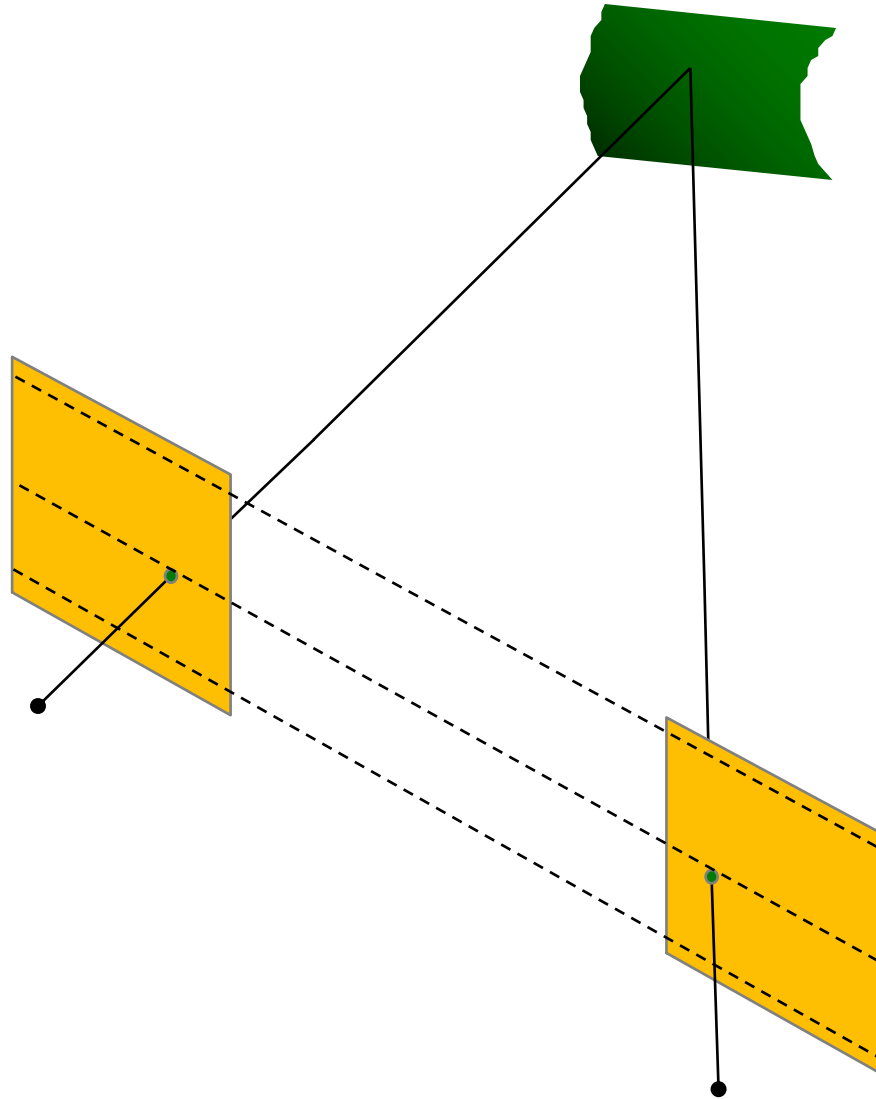
1. Compute \mathbf{E} to get \mathbf{R}
2. Rotate right image by \mathbf{R}
3. Rotate both images by \mathbf{R}_{rect}
4. Scale both images by \mathbf{H}

Stereo Rectification:



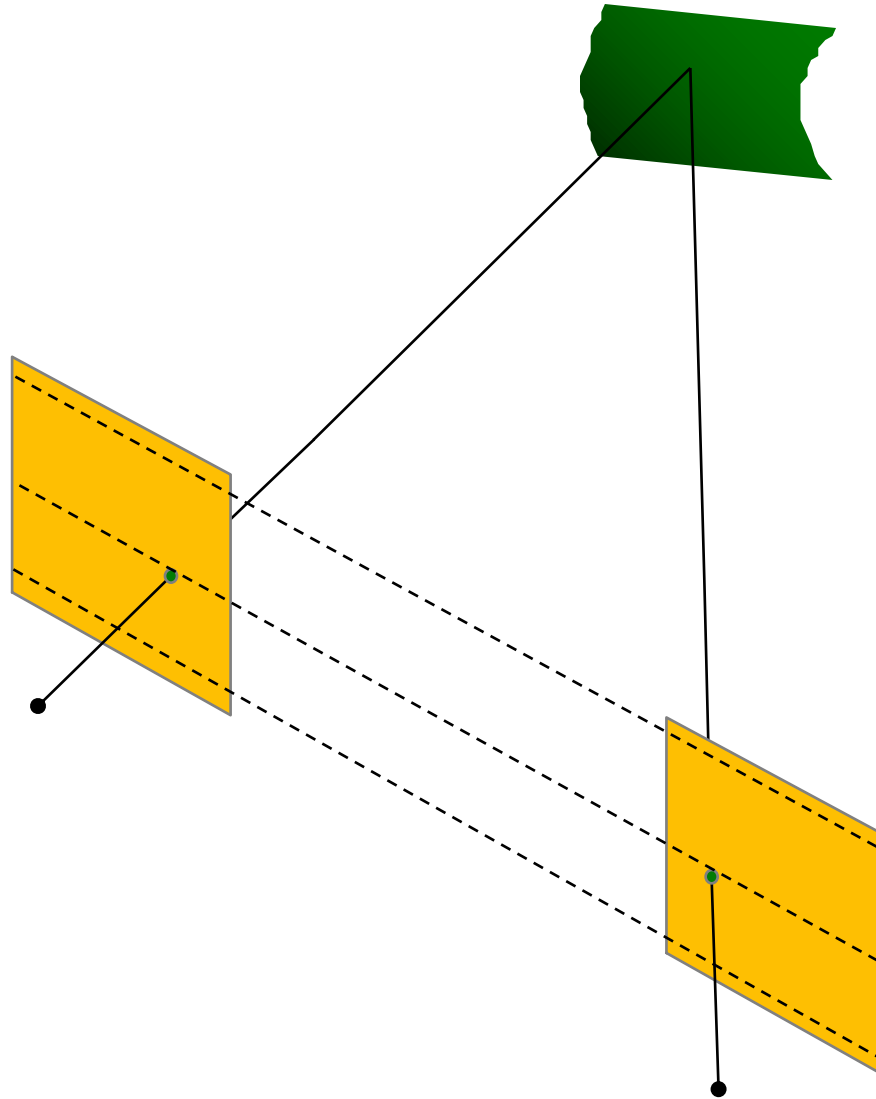
1. Compute \mathbf{E} to get \mathbf{R}
2. Rotate right image by \mathbf{R}
3. Rotate both images by \mathbf{R}_{rect}
4. Scale both images by \mathbf{H}

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}

Suppose $\mathbf{E} = [\mathbf{t}]_{\times} \mathbf{R}$

Step 1: Use \mathbf{E} to get \mathbf{R}

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

We get FOUR solutions:

$$\mathbf{R}_1 = \mathbf{U}\mathbf{W}\mathbf{V}^{\top} \quad \mathbf{R}_2 = \mathbf{U}\mathbf{W}^{\top}\mathbf{V}^{\top} \quad [\mathbf{t}]_{\times} = \pm \mathbf{U}\mathbf{W}\mathbf{\Sigma}\mathbf{U}^{\top}$$

two possible rotations two possible translations

We get FOUR solutions:

$$\mathbf{R}_1 = \mathbf{U}\mathbf{W}\mathbf{V}^\top$$
$$[\mathbf{t}]_{\times} = \mathbf{U}\mathbf{W}\Sigma\mathbf{U}^\top$$

$$\mathbf{R}_1 = \mathbf{U}\mathbf{W}\mathbf{V}^\top$$
$$[\mathbf{t}]_{\times} = -\mathbf{U}\mathbf{W}\Sigma\mathbf{U}^\top$$

$$\mathbf{R}_2 = \mathbf{U}\mathbf{W}^\top\mathbf{V}^\top$$
$$[\mathbf{t}]_{\times} = \mathbf{U}\mathbf{W}\Sigma\mathbf{U}^\top$$

$$\mathbf{R}_2 = \mathbf{U}\mathbf{W}^\top\mathbf{V}^\top$$
$$[\mathbf{t}]_{\times} = -\mathbf{U}\mathbf{W}\Sigma\mathbf{U}^\top$$

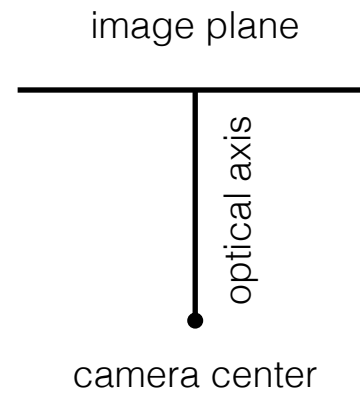
Which one do we choose?

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)

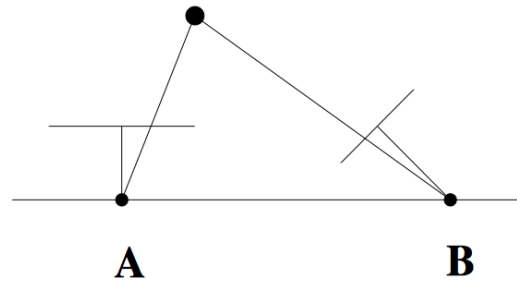
Let's visualize the four configurations...

Camera Icon

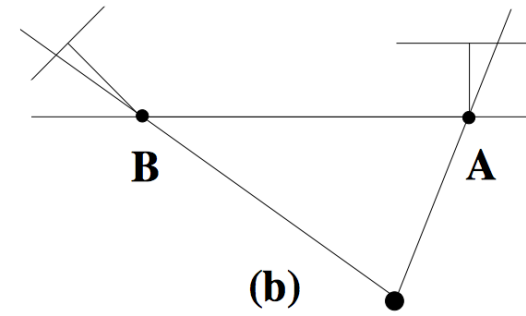


Find the configuration where the point is in front of both cameras

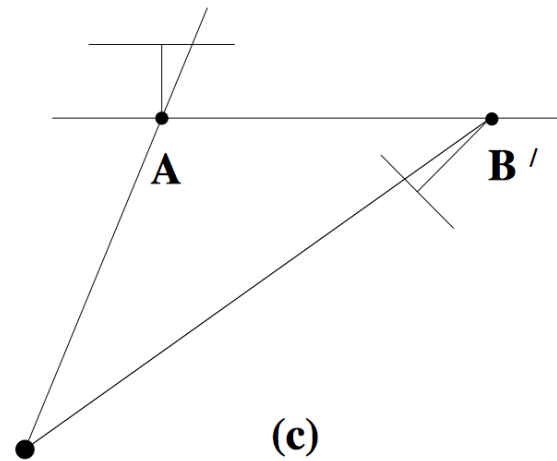
Find the configuration where the points is in front of both cameras



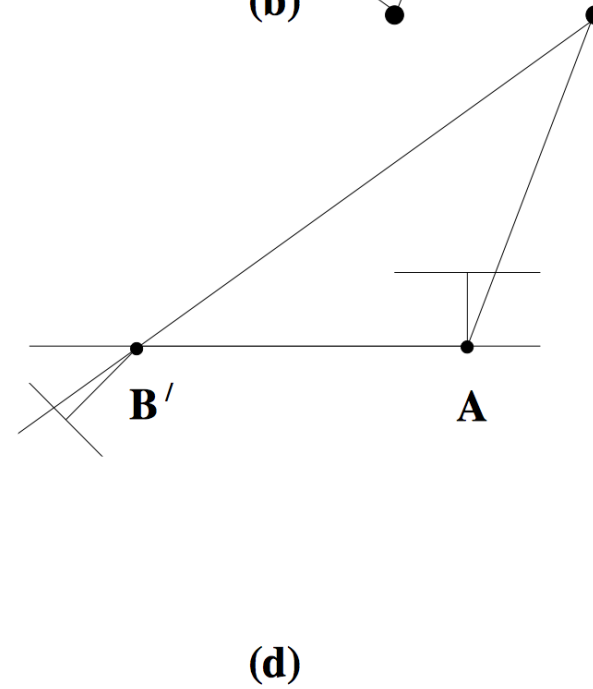
(a)



(b)

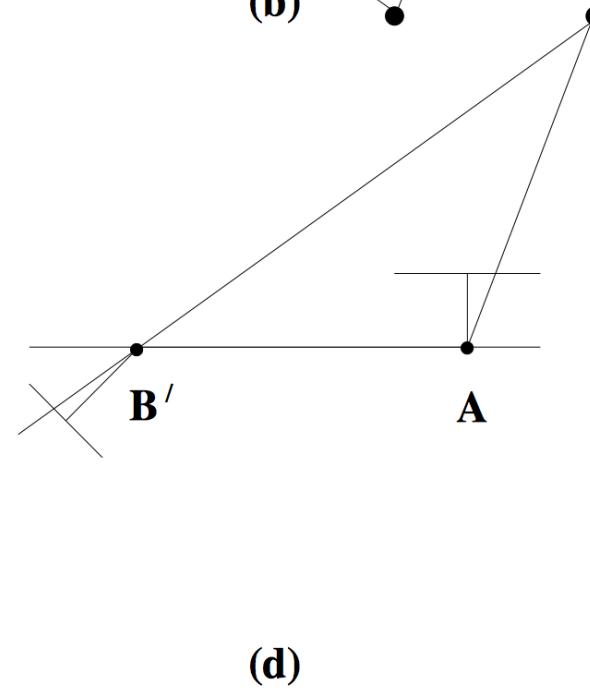
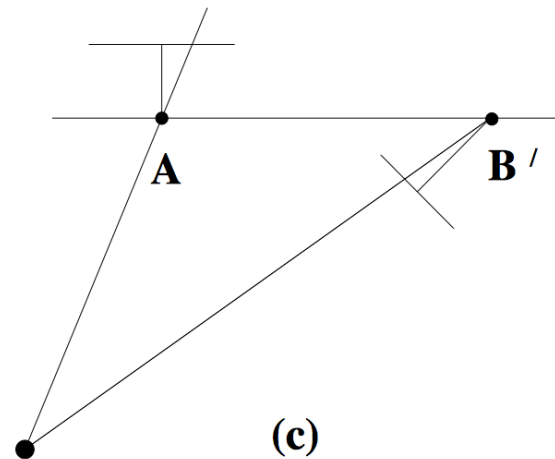
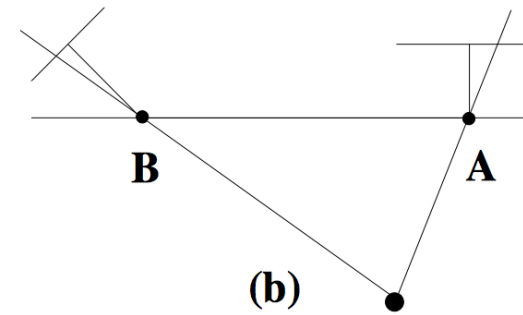
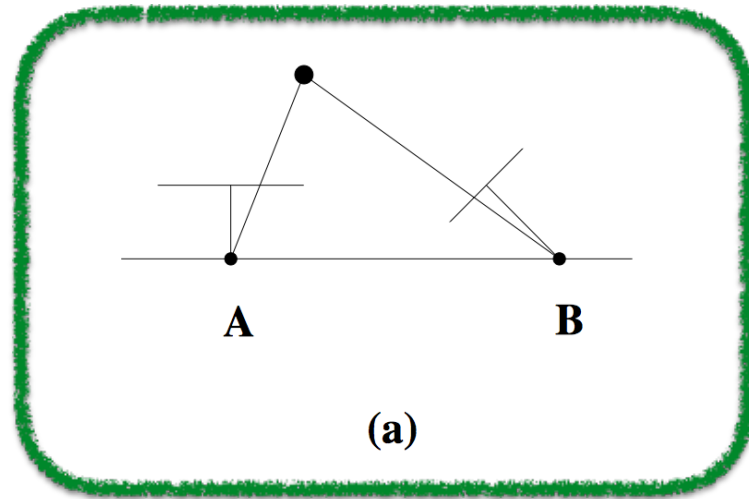


(c)

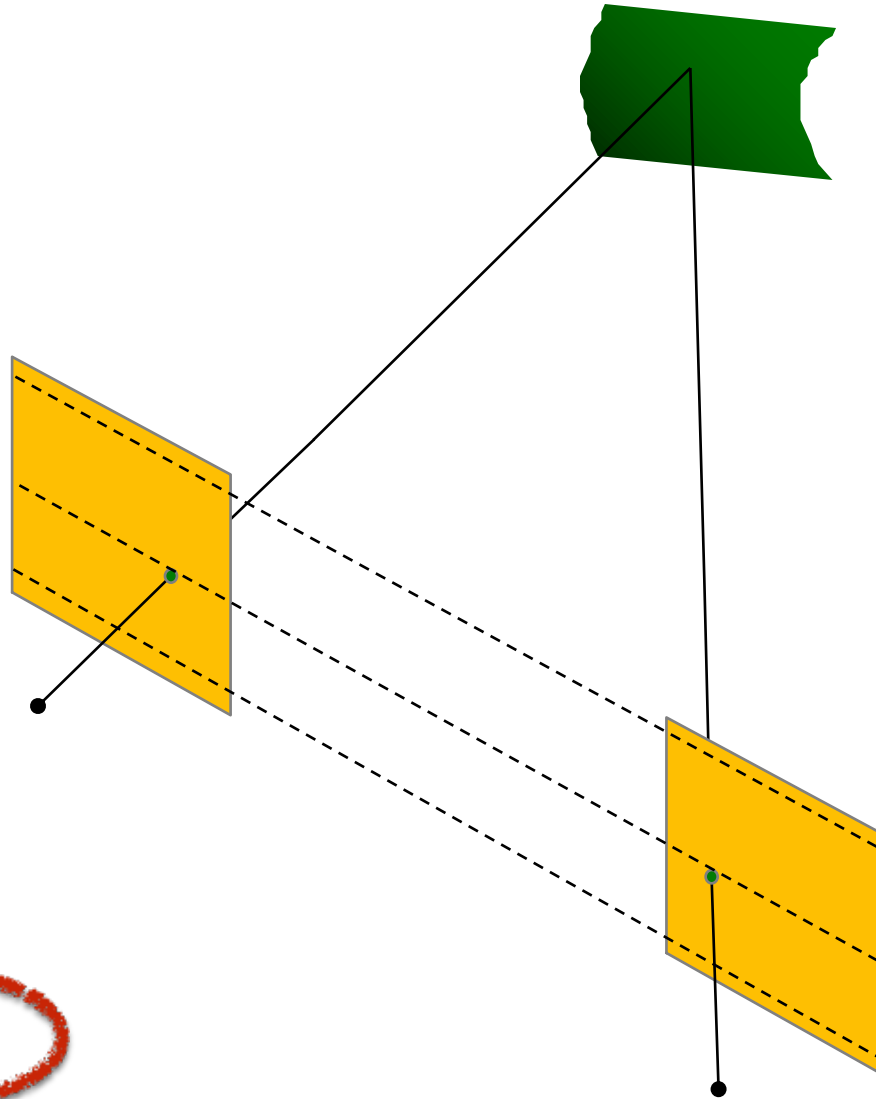


(d)

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



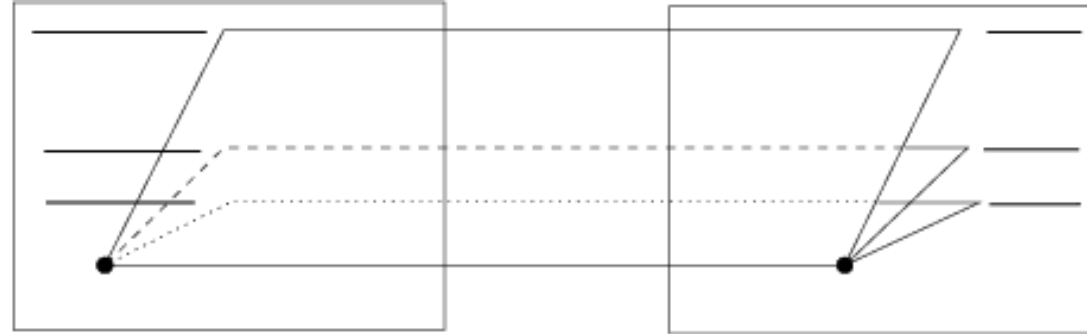
Stereo Rectification:



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}

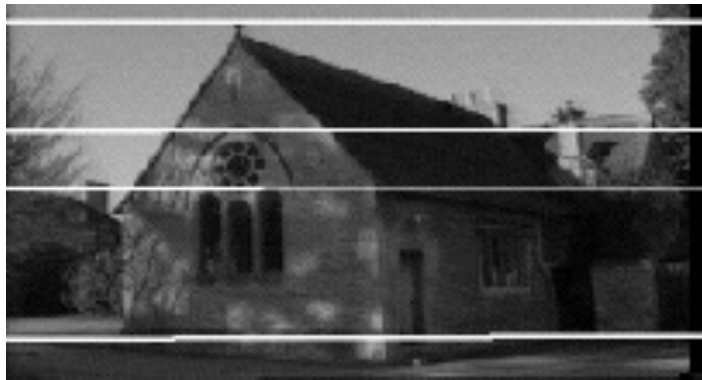
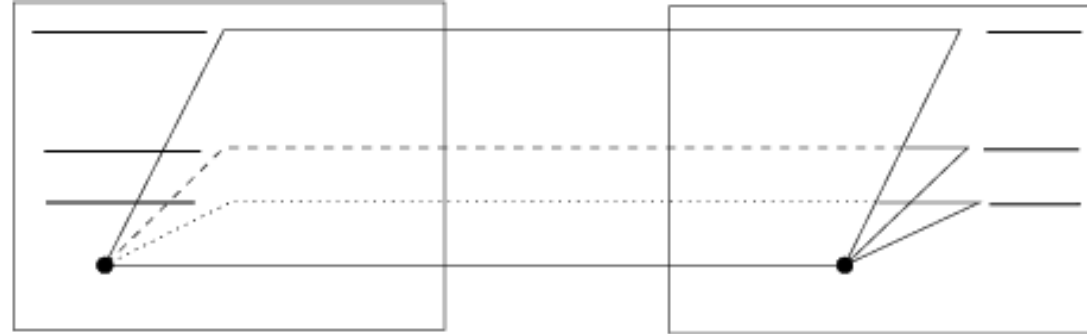
*When do epipolar
lines become
horizontal?*

Parallel cameras



Where is the epipole?

Parallel cameras



epipole at infinity

Setting the epipole to infinity

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

Let $\mathbf{R}_{\text{rect}} = \begin{bmatrix} \mathbf{r}_1^\top \\ \mathbf{r}_2^\top \\ \mathbf{r}_3^\top \end{bmatrix}$ Given: epipole \mathbf{e}
(using SVD on \mathbf{E})
(translation from \mathbf{E})

$$\mathbf{r}_1 = \mathbf{e}_1 = \frac{\mathbf{T}}{\|\mathbf{T}\|} \quad \text{epipole coincides with translation vector}$$

$$\mathbf{r}_2 = \frac{1}{\sqrt{T_x^2 + T_y^2}} \begin{bmatrix} -T_y & T_x & 0 \end{bmatrix} \quad \text{cross product of } \mathbf{e} \text{ and the direction vector of the optical axis}$$

$$\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2 \quad \text{orthogonal vector}$$

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

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

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

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

Where is this point located on the image plane?

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

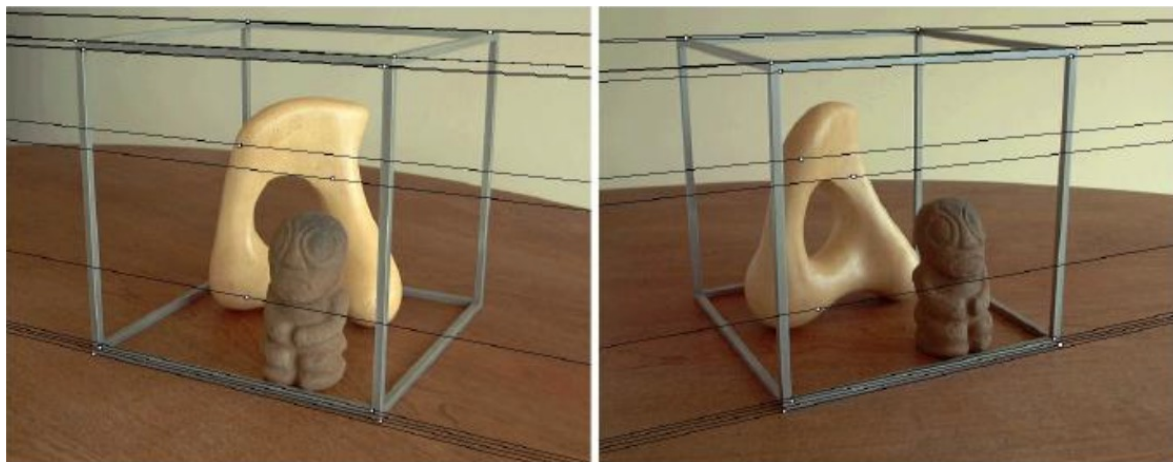
then $R_{\text{rect}} \mathbf{e}_1 = \begin{bmatrix} \mathbf{r}_1^\top \mathbf{e}_1 \\ \mathbf{r}_2^\top \mathbf{e}_1 \\ \mathbf{r}_3^\top \mathbf{e}_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

Where is this point located on the image plane?

At x-infinity

Stereo Rectification Algorithm

1. Estimate \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



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

How would you do this?

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

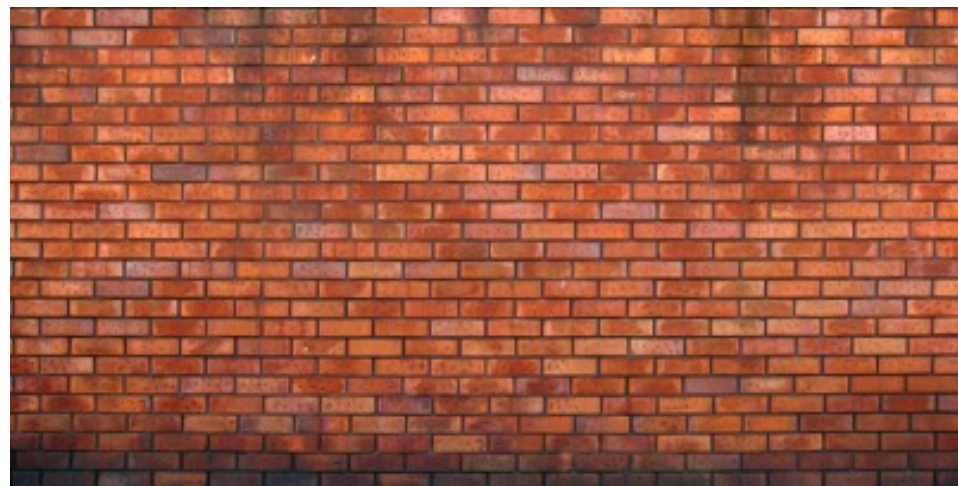
Reminder from filtering

How do we detect an edge?

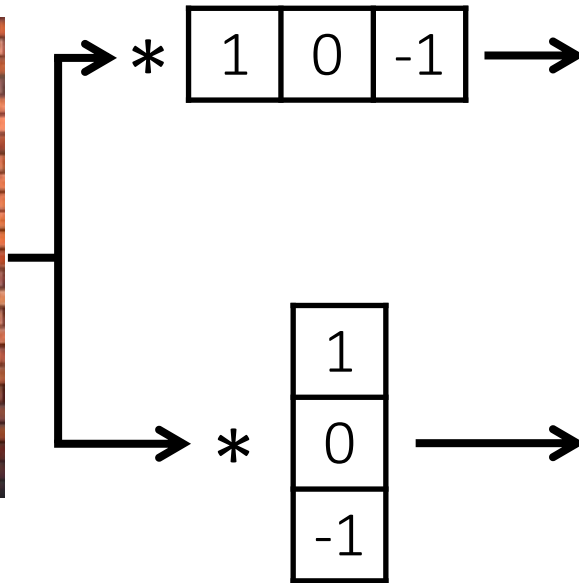
Reminder from filtering

How do we detect an edge?

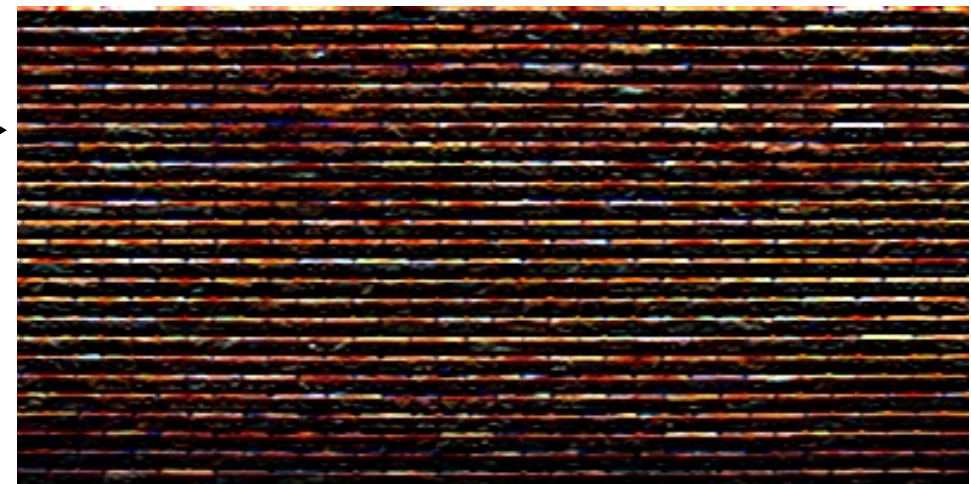
- We filter with something that looks like an edge.



original



horizontal edge filter

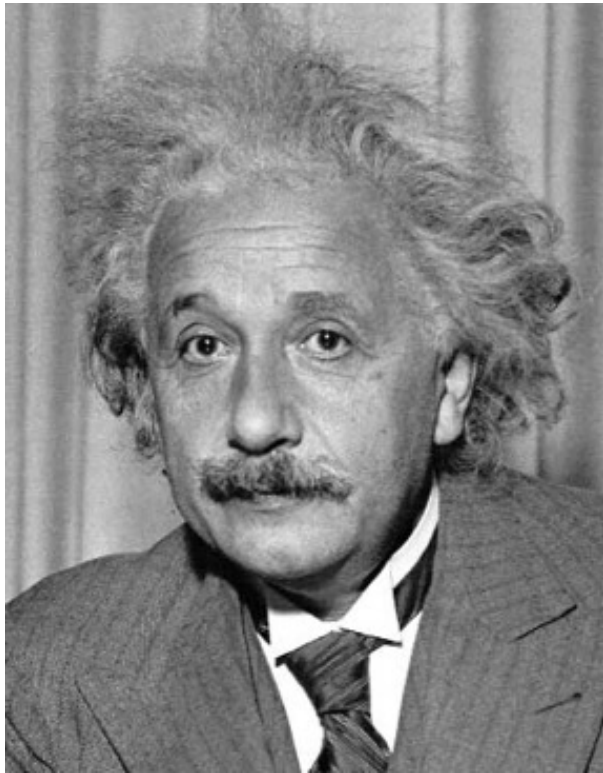


vertical edge filter

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

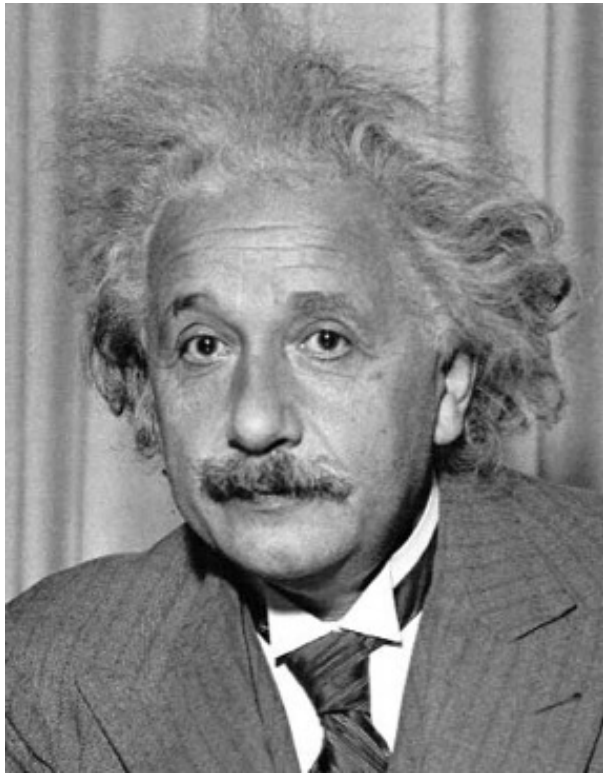
Find this template

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

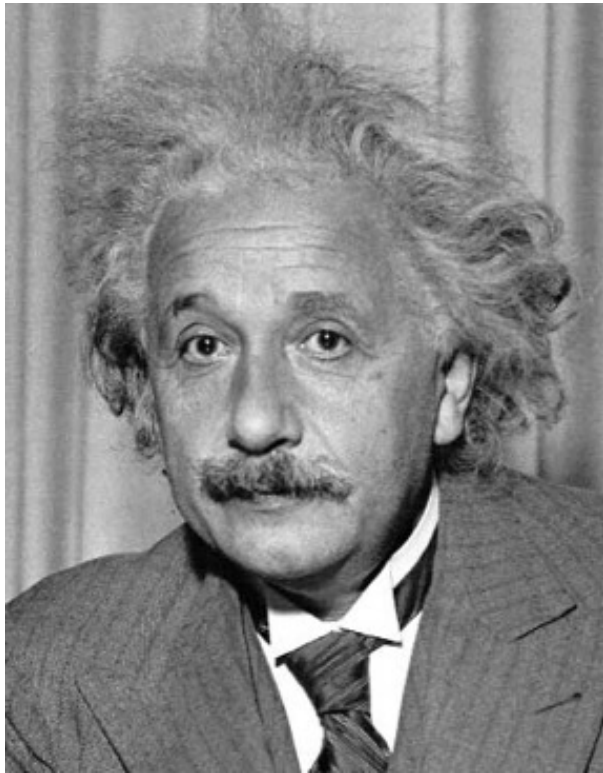
The diagram shows a large arrow pointing from the word 'image' to the right side of the equation. A smaller arrow points from the word 'filter' and the eye icon to the $g[k, l]$ term in the equation.

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

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


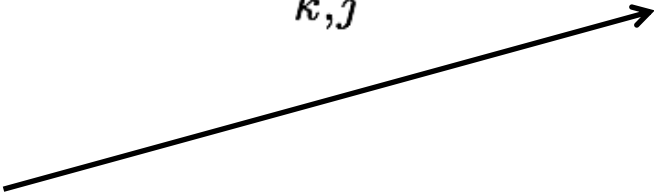
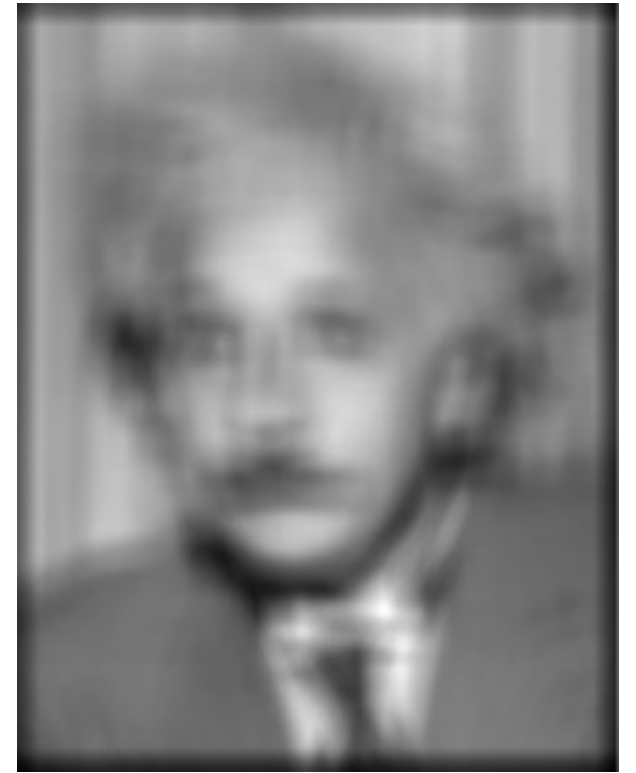
filter 

image 

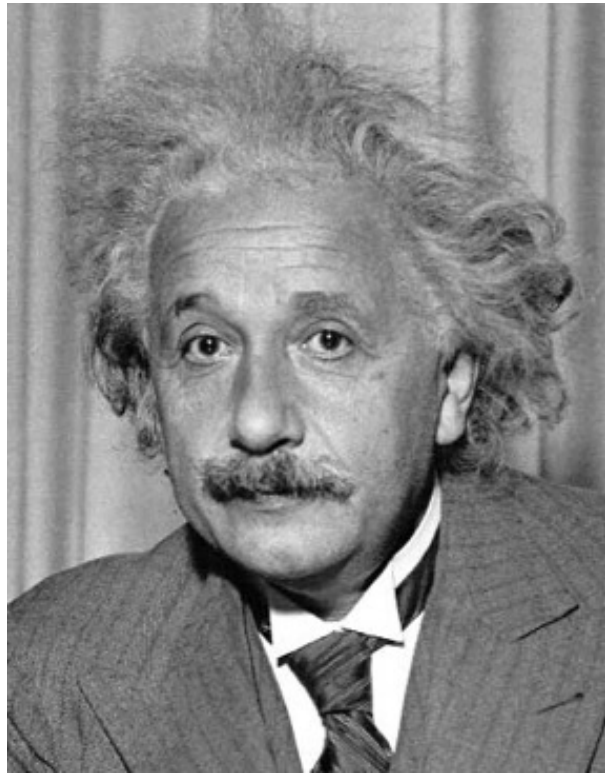


Solution 1: Filter the image using the template as filter kernel.

What went wrong?


Find this template

How do we detect the template  in the following image?

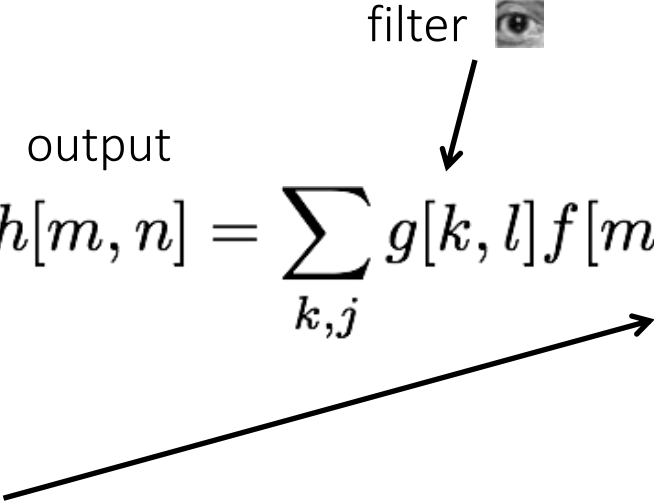


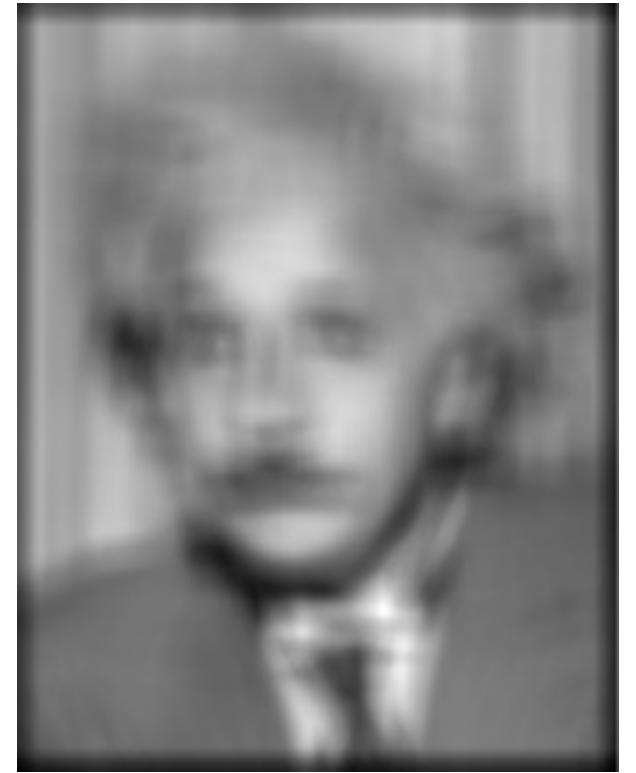
output

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

filter 

image



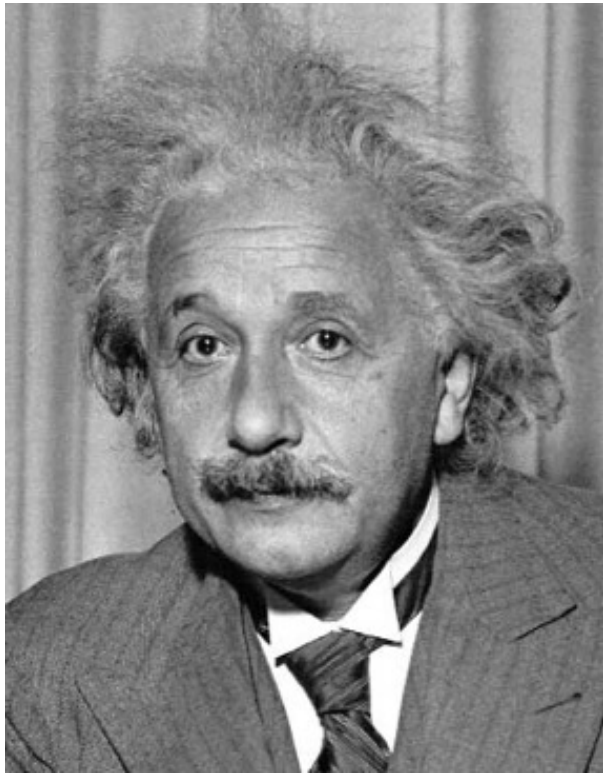


Increases for higher local intensities.

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


Find this template

How do we detect the template  in the following image?



output

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

filter  template mean


image

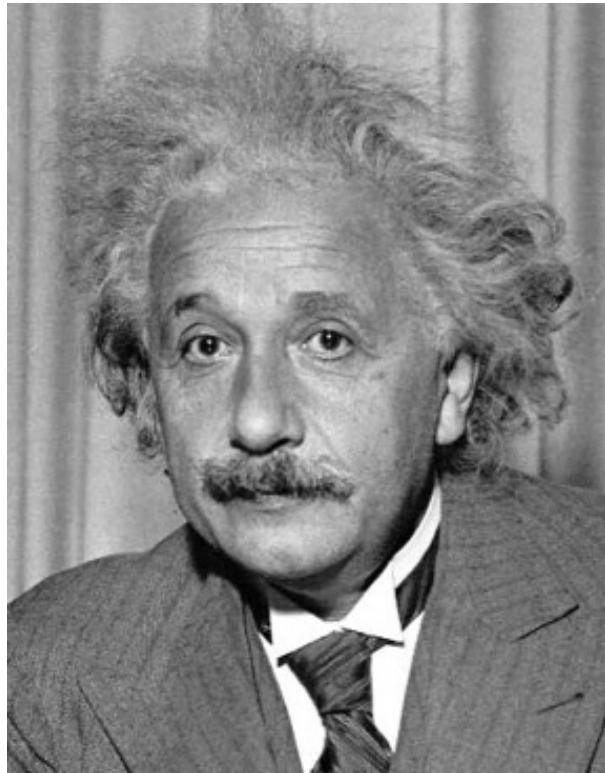
The diagram shows the mathematical formula for template matching. An arrow labeled 'filter' points to the term $g[k, l]$. Another arrow labeled 'template mean' points to the term \bar{g} . A long arrow labeled 'image' points from the Einstein portrait to the term $f[m + k, n + l]$.

What will the output look like?

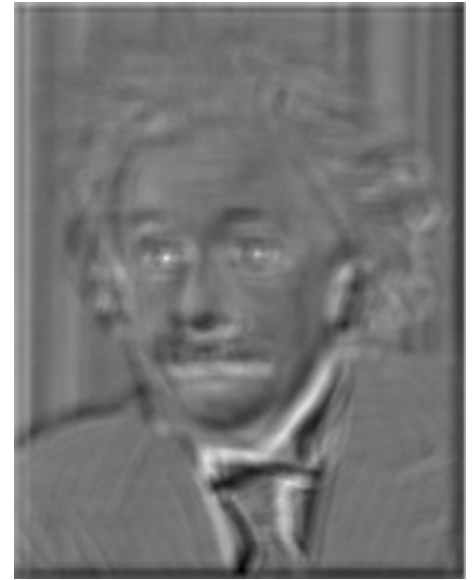
Solution 2: Filter the image using a *zero-mean* template.


Find this template

How do we detect the template  in the following image?



output



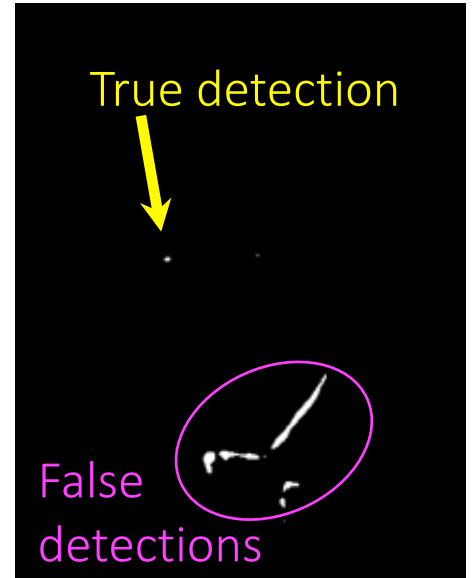
filter  template mean

output

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

image

thresholding

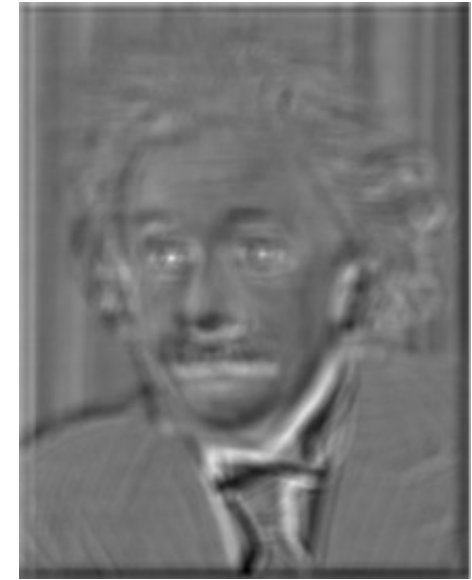
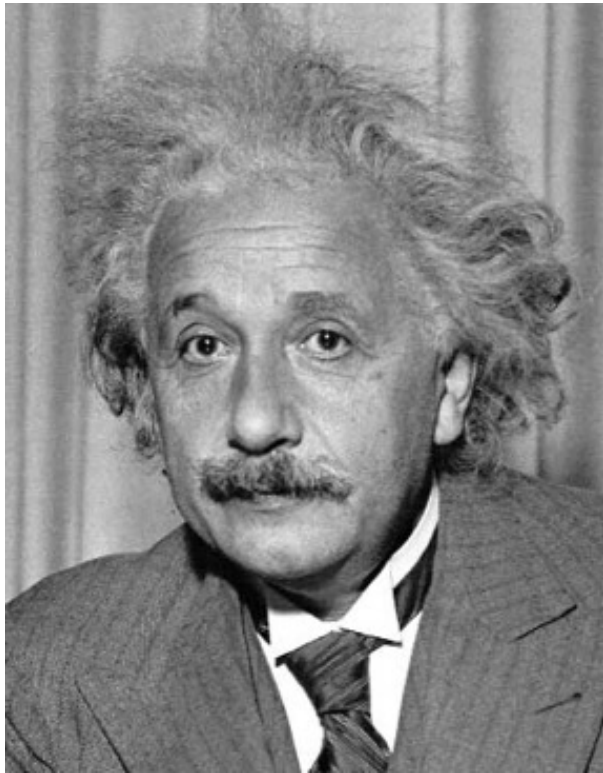


Solution 2: Filter the image using a *zero-mean* template.

What went wrong?


Find this template

How do we detect the template  in the following image?



output

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

filter  template mean

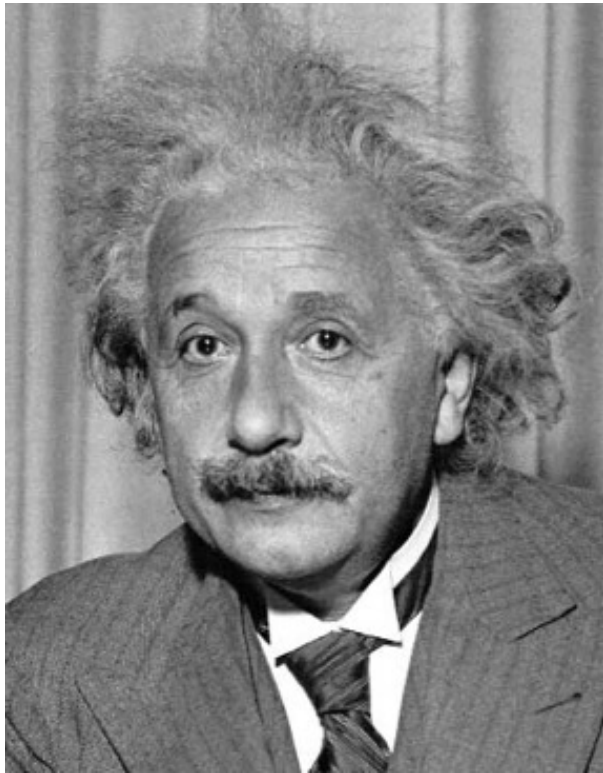
image

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

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

filter 

image

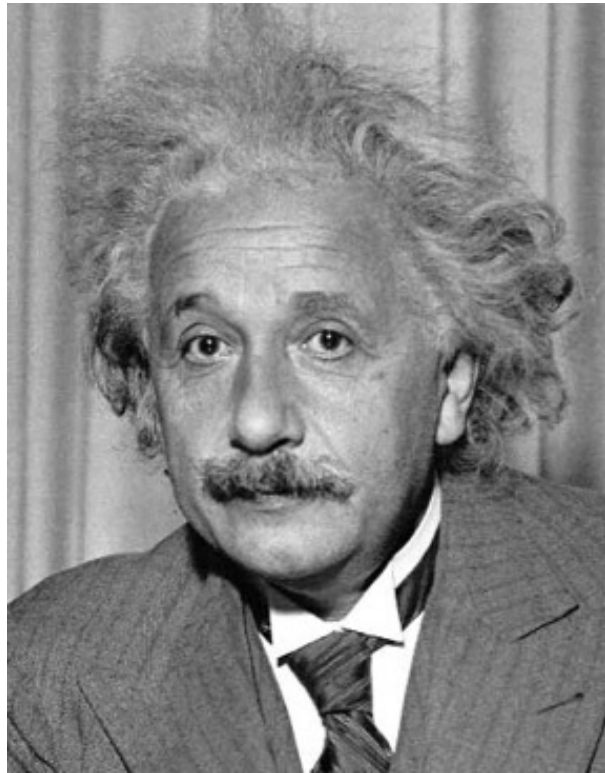
The diagram shows the word 'output' above the equation. An arrow labeled 'filter' with an eye icon points to the $g[k, l]$ term in the equation. An arrow labeled 'image' points from the Einstein portrait to the $f[m + k, n + l]$ term in the equation.

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?




1-output



output

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

filter 

image

thresholding

True detection

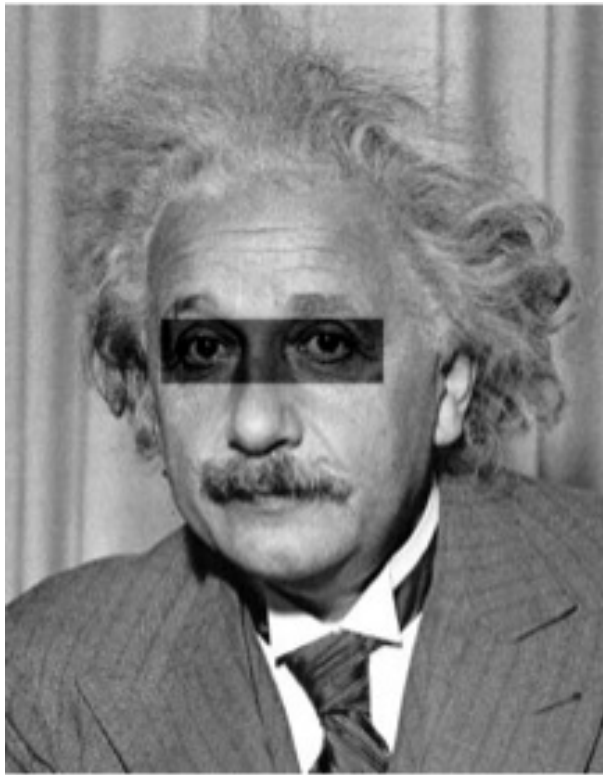


Solution 3: Use sum of squared differences (SSD).


What could go wrong?

Find this template

How do we detect the template  in the following image?



1-output

filter 

output

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

image

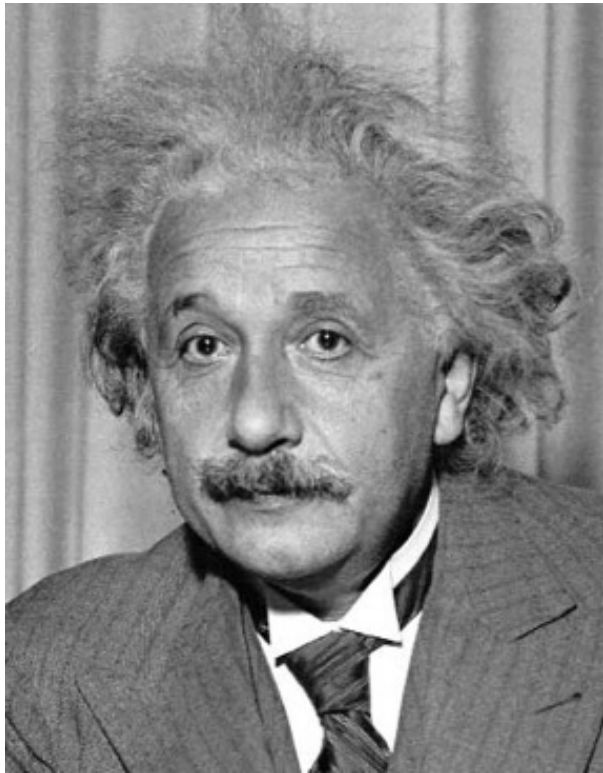
The diagram shows a large arrow pointing from the 'image' label to the equation, and a smaller arrow pointing from the 'filter' label to the equation.

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?



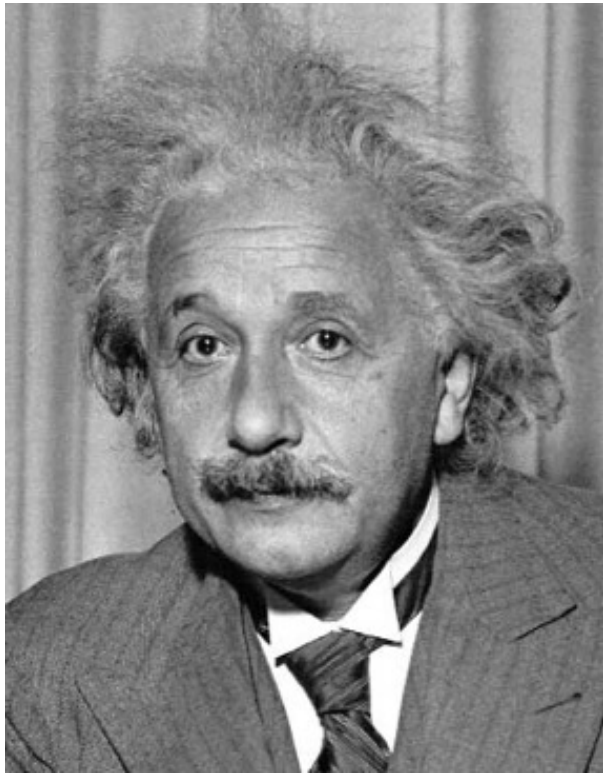
Observations so far:

- 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

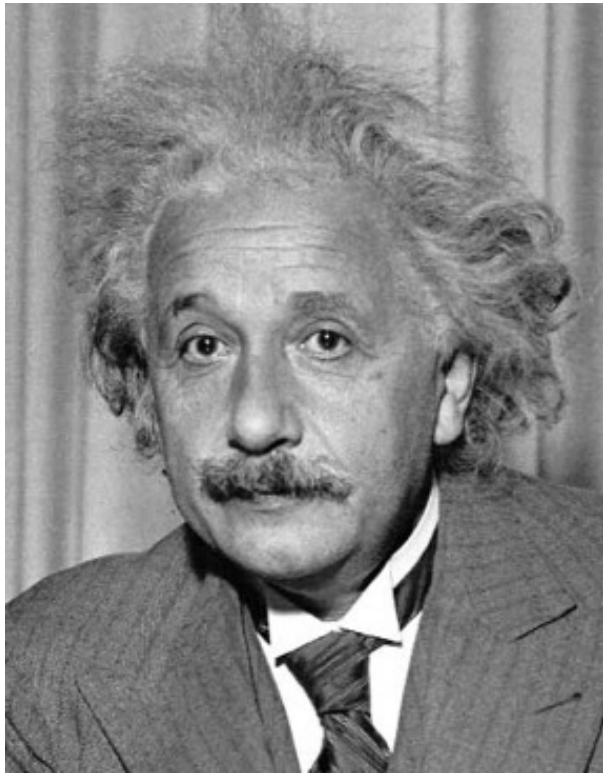
$$h[m, n] = \frac{\sum_{k,l} (g[k, l] - \bar{g})(f[m + k, n + l] - \bar{f}_{m,n})}{\sqrt{(\sum_{k,l} (g[k, l] - \bar{g})^2 \sum_{k,l} (f[m + k, n + l] - \bar{f}_{m,n})^2)}}$$

image local patch mean

Solution 4: Normalized cross-correlation (NCC).

Find this template

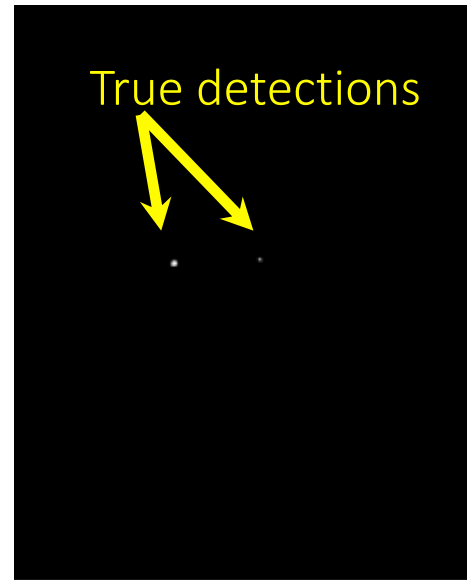
How do we detect the template  in the following image?



1-output



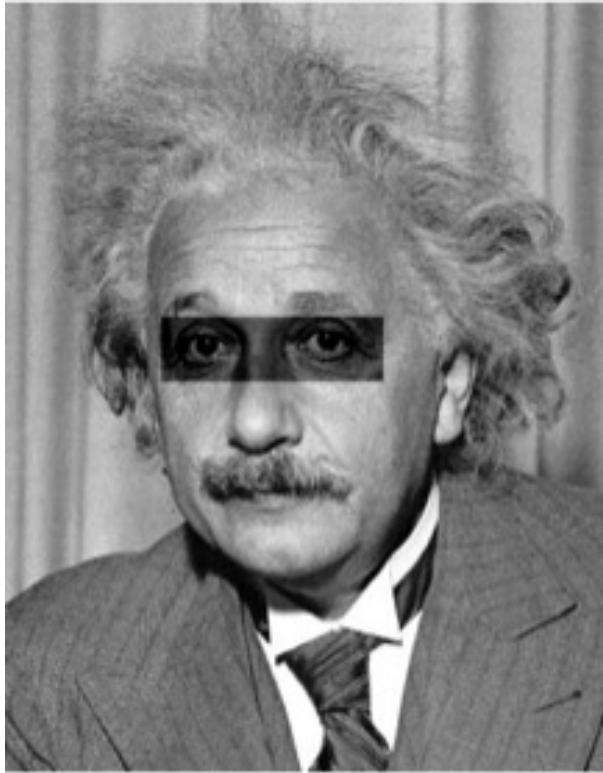
thresholding



Solution 4: Normalized cross-correlation (NCC).

Find this template

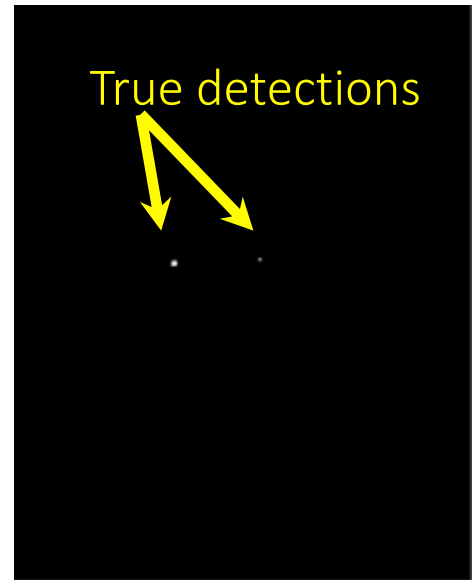
How do we detect the template  in the following image?



1-output



thresholding



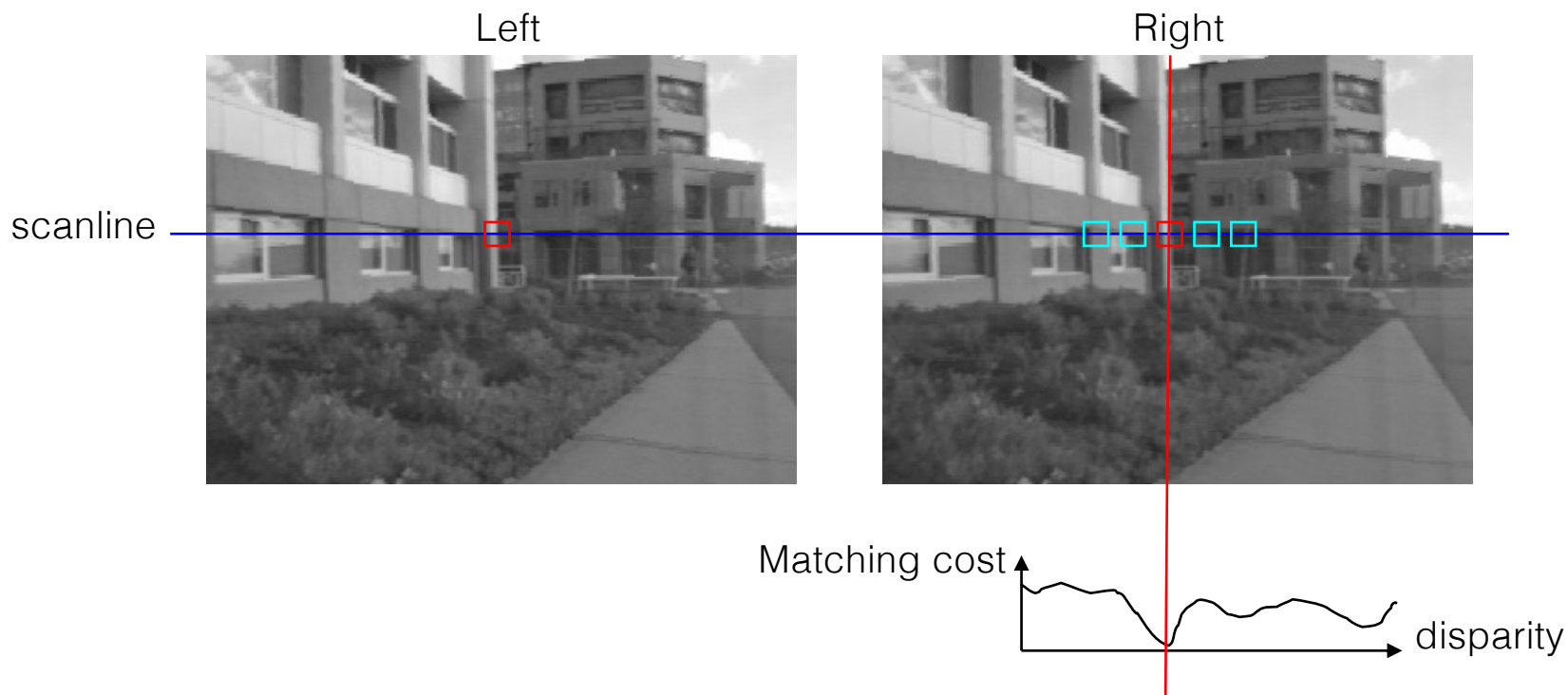
Solution 4: Normalized cross-correlation (NCC).

What is the best method?

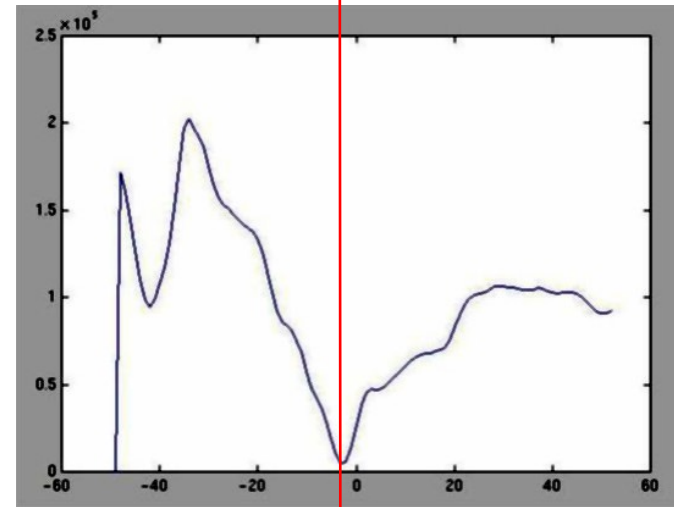
It depends on whether you care about speed or invariance.

- 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.

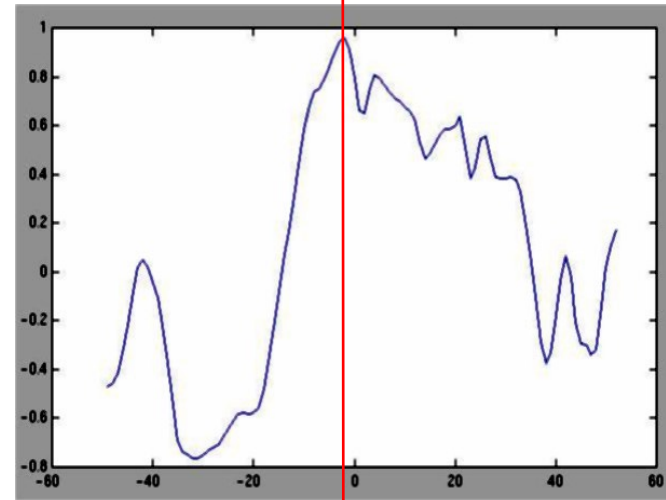
Stereo Block Matching



- 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

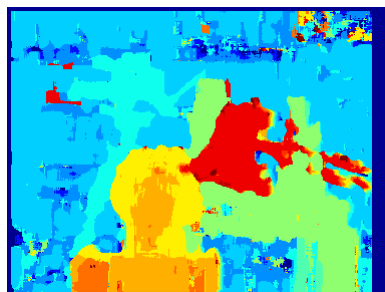


SSD

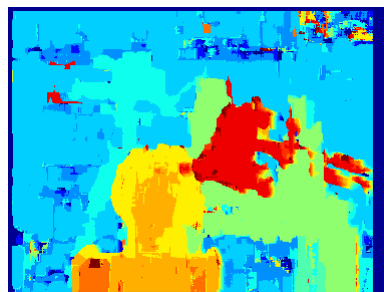


Normalized cross-correlation

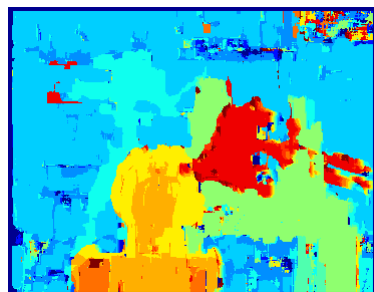
| Similarity Measure | Formula |
|------------------------------------|---|
| Sum of Absolute Differences (SAD) | $\sum_{(i,j) \in W} I_1(i,j) - I_2(x+i, y+j) $ |
| Sum of Squared Differences (SSD) | $\sum_{(i,j) \in W} (I_1(i,j) - I_2(x+i, y+j))^2$ |
| Zero-mean SAD | $\sum_{(i,j) \in W} I_1(i,j) - \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

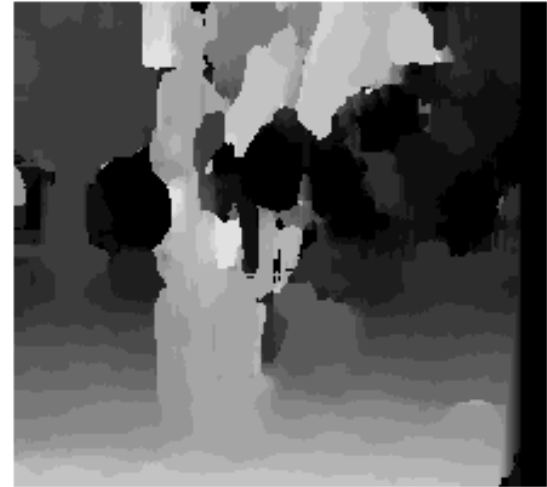


Ground truth

Effect of window size



$W = 3$



$W = 20$

Effect of window size



$W = 3$

Smaller window

- + More detail
- More noise

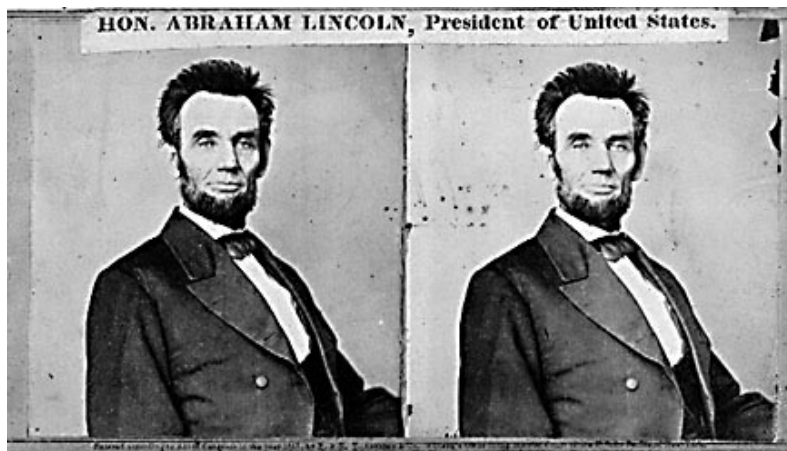


$W = 20$

Larger window

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

When will stereo block matching fail?



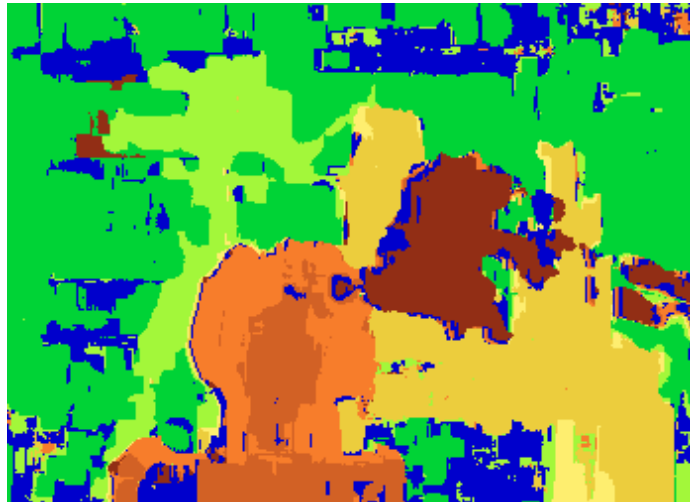
When will stereo block matching fail?



Improving stereo matching



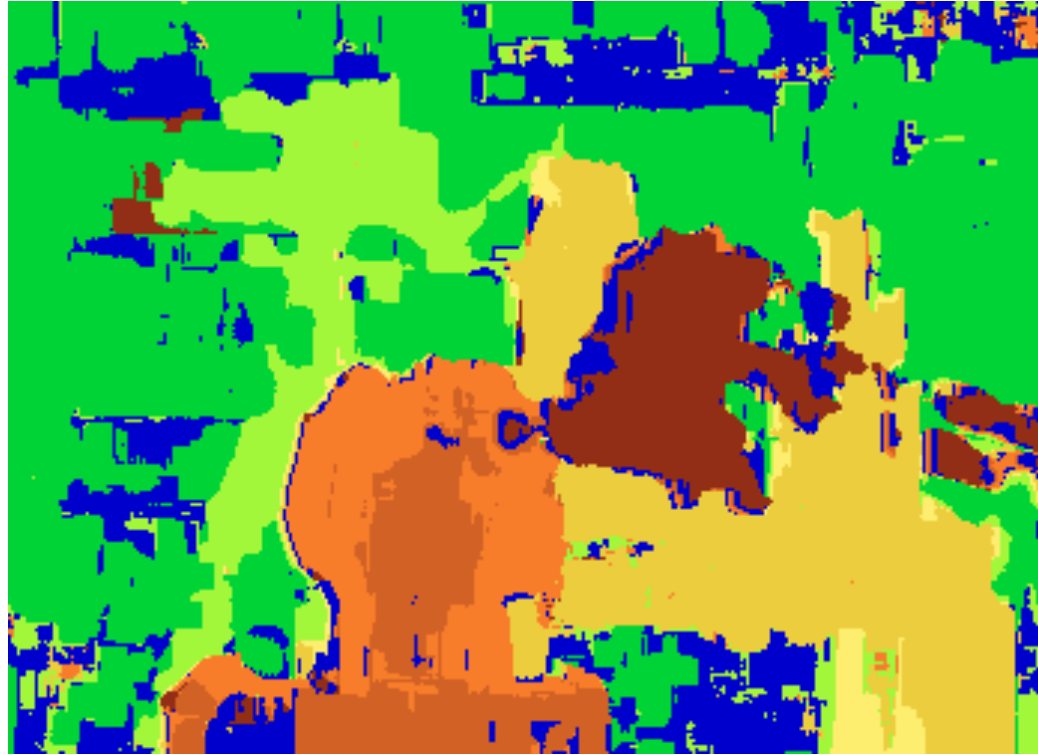
Block matching



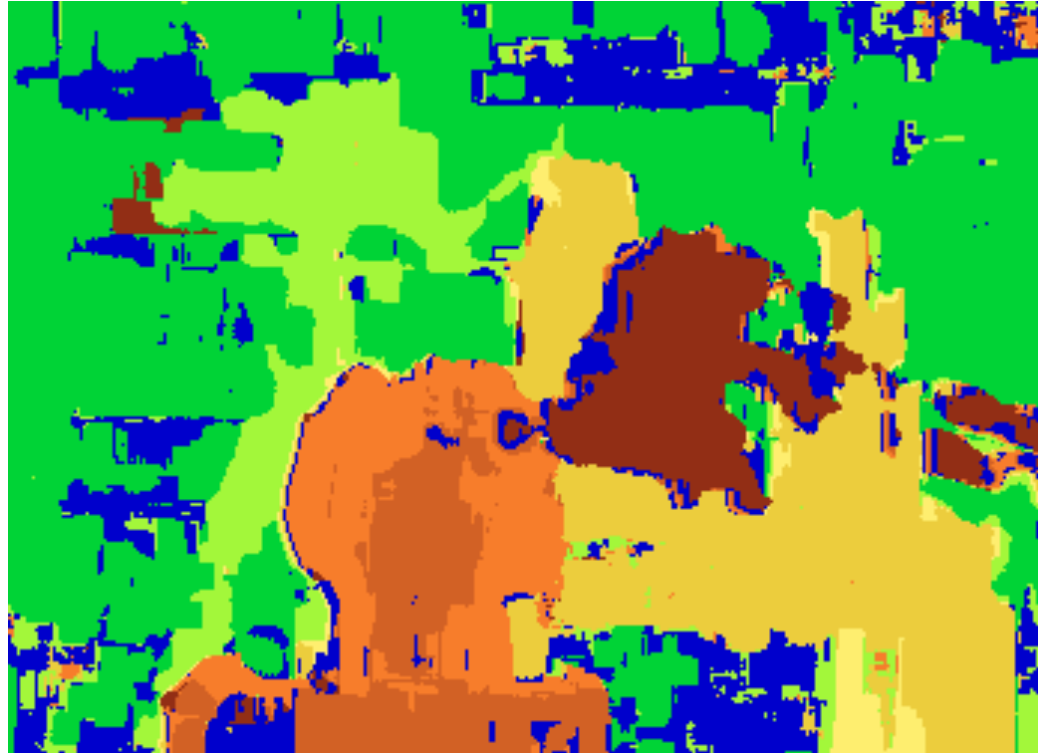
Ground truth



What are some problems with the result?



How can we improve depth estimation?



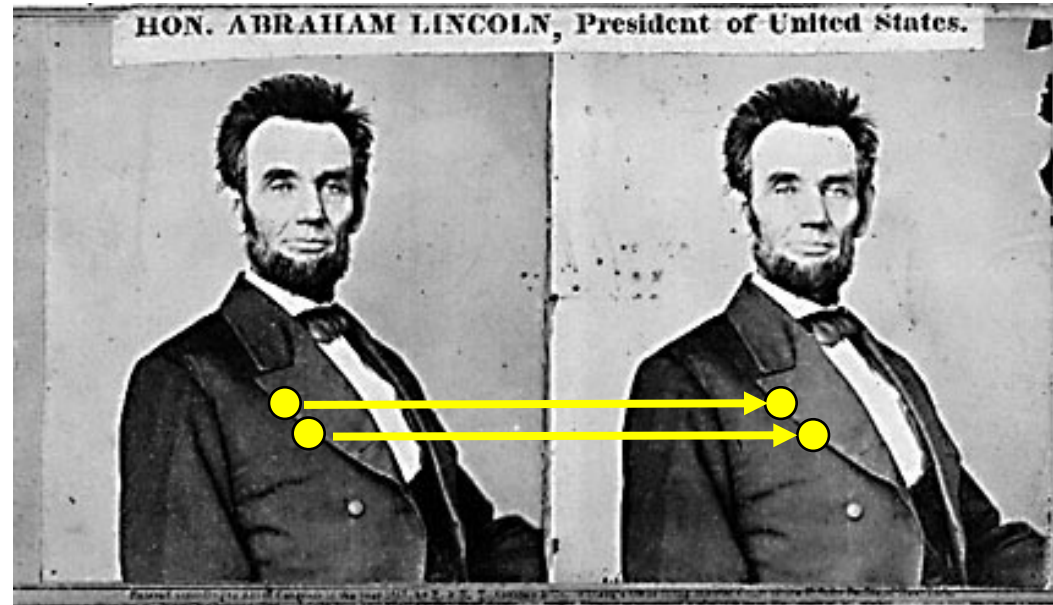
How can we improve depth estimation?

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)

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

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

data term

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

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

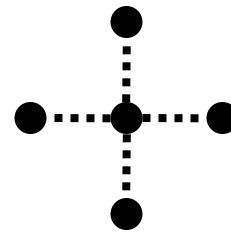
$$E_d(d) = \sum_{(x,y) \in I} C(x, y, d(x, y))$$

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

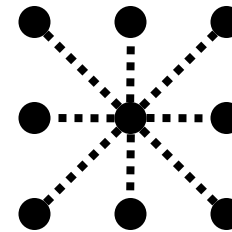
$$E_s(d) = \sum_{(p,q) \in \mathcal{E}} V(d_p, d_q)$$

smoothness term

\mathcal{E} : set of neighboring pixels



4-connected neighborhood



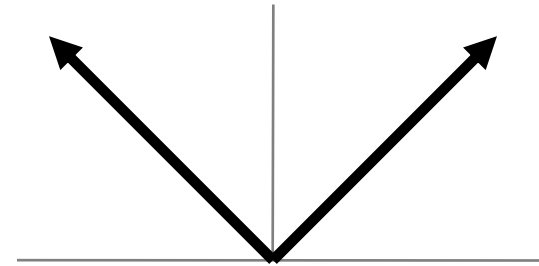
8-connected neighborhood

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

smoothness term

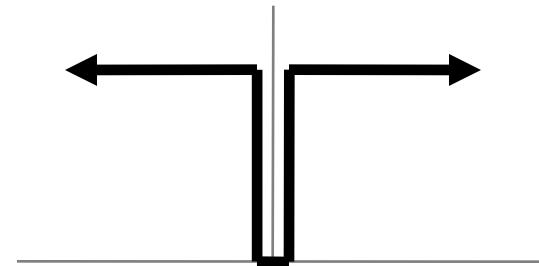
$$V(d_p, d_q) = |d_p - d_q|$$

L₁ 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...

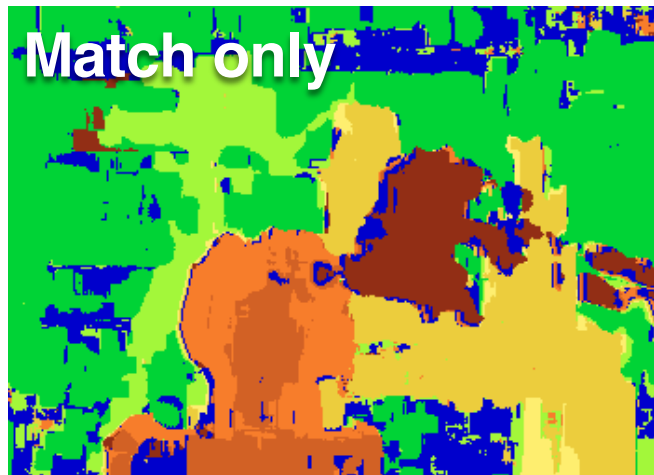
Dynamic Programming

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

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

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

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

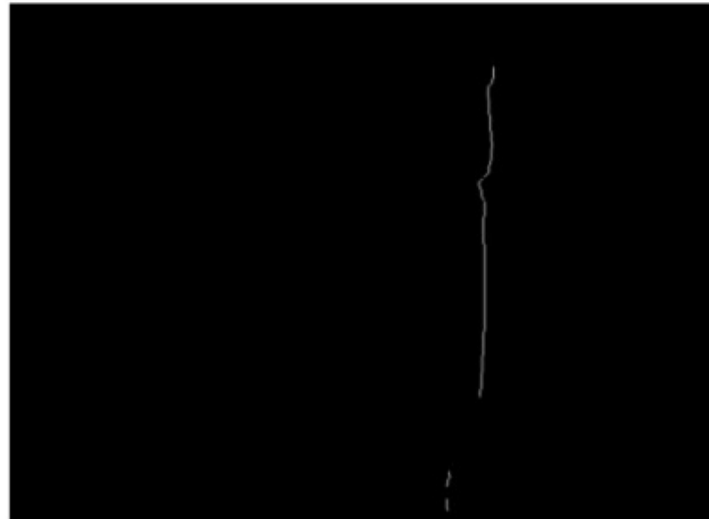


Y. Boykov, O. Veksler, and R. Zabih, [Fast Approximate Energy Minimization via Graph Cuts](#), PAMI 2001

Structured light

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

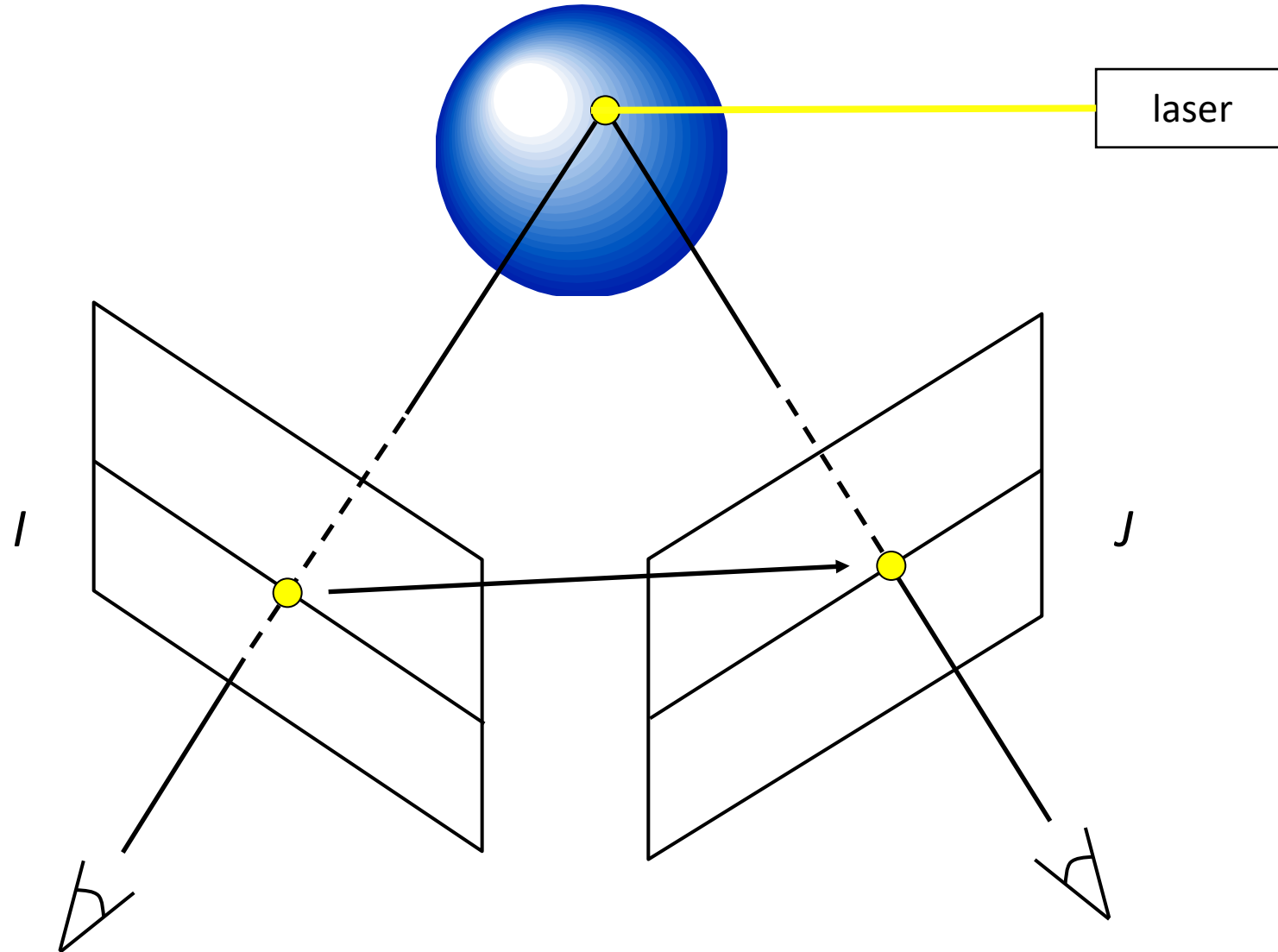
Disparity between laser points on the same scanline in the images determines the 3-D coordinates of the laser point on object



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

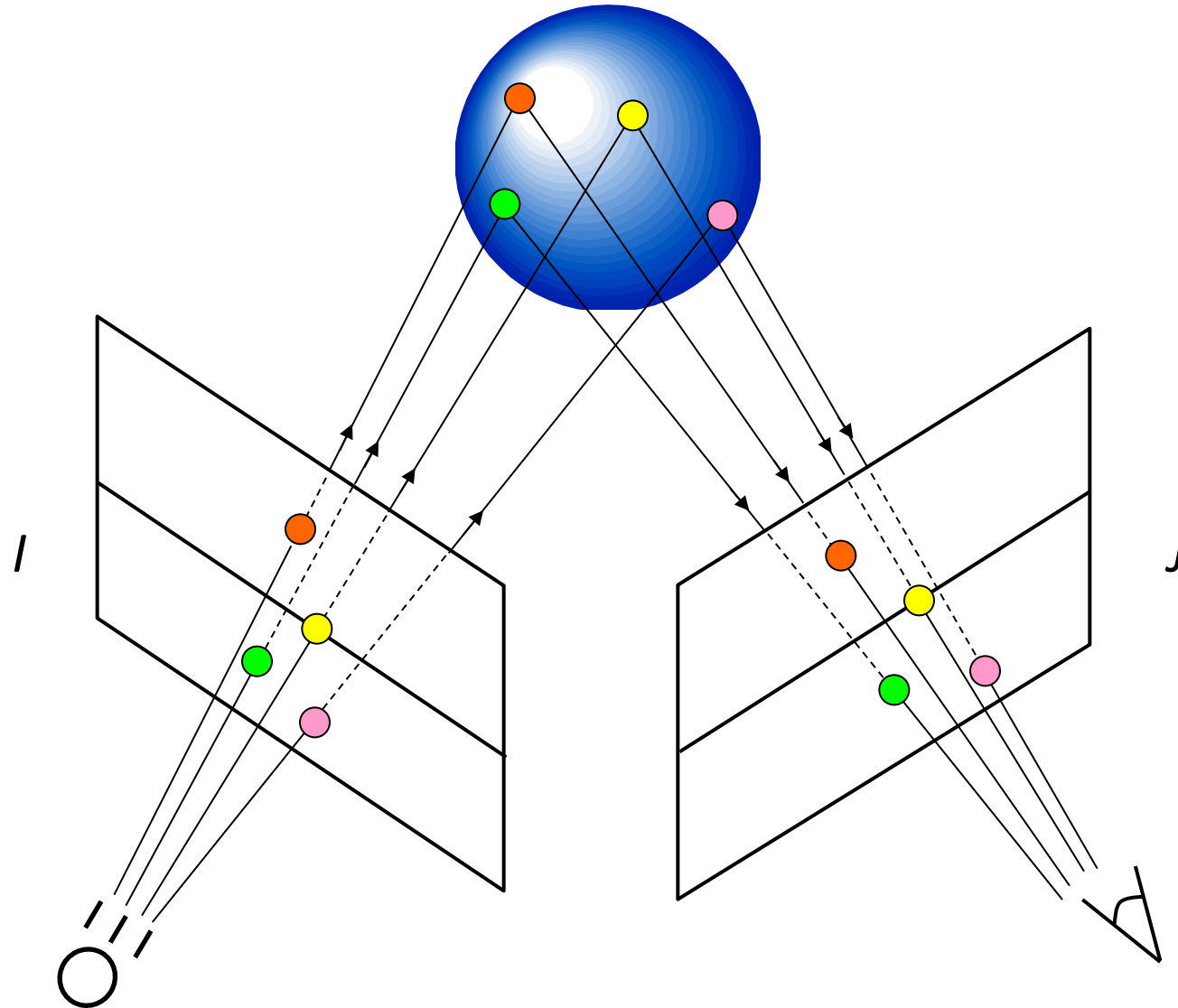


Structured light and two cameras

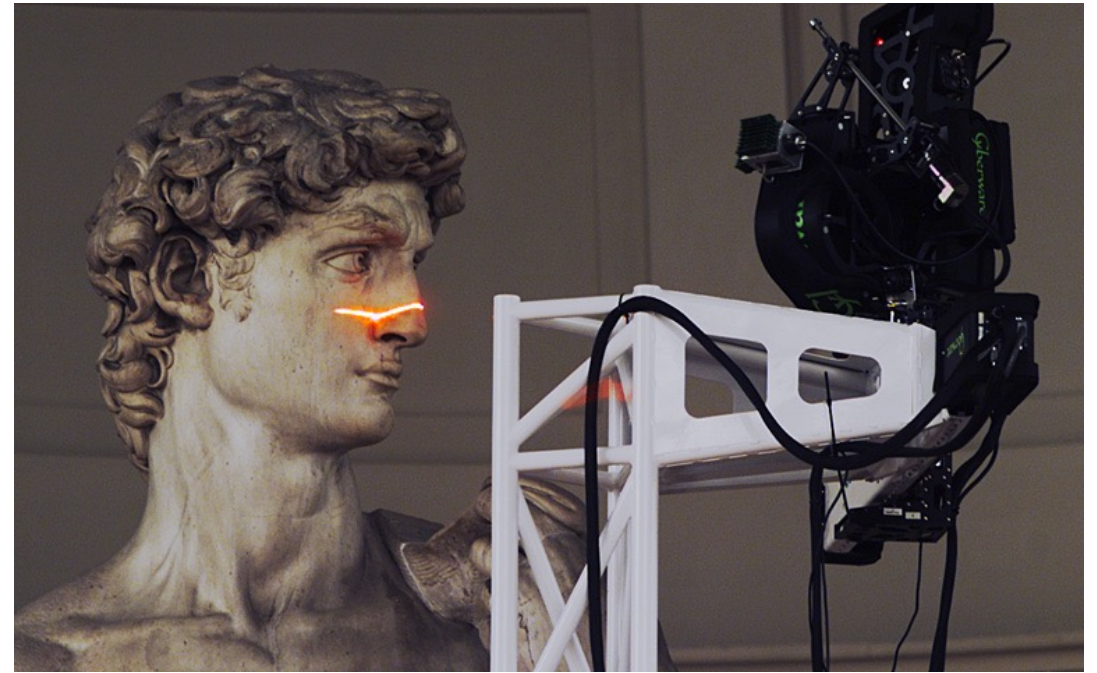
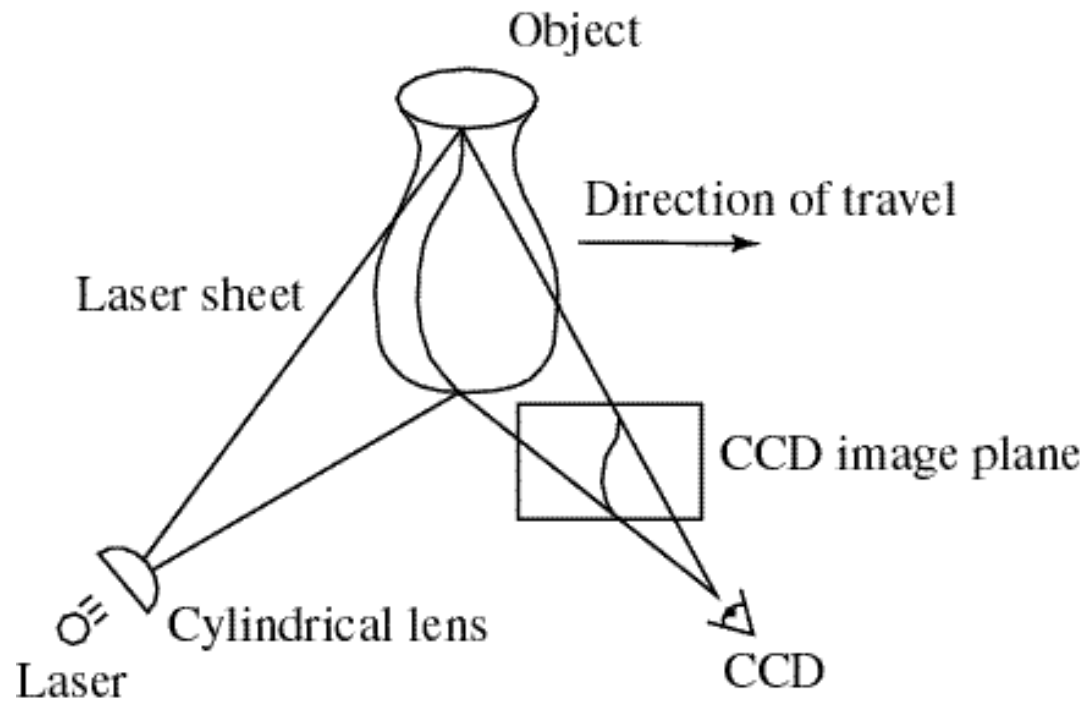


Structured light and one camera

Projector acts like
"reverse" camera



Example: Laser scanner



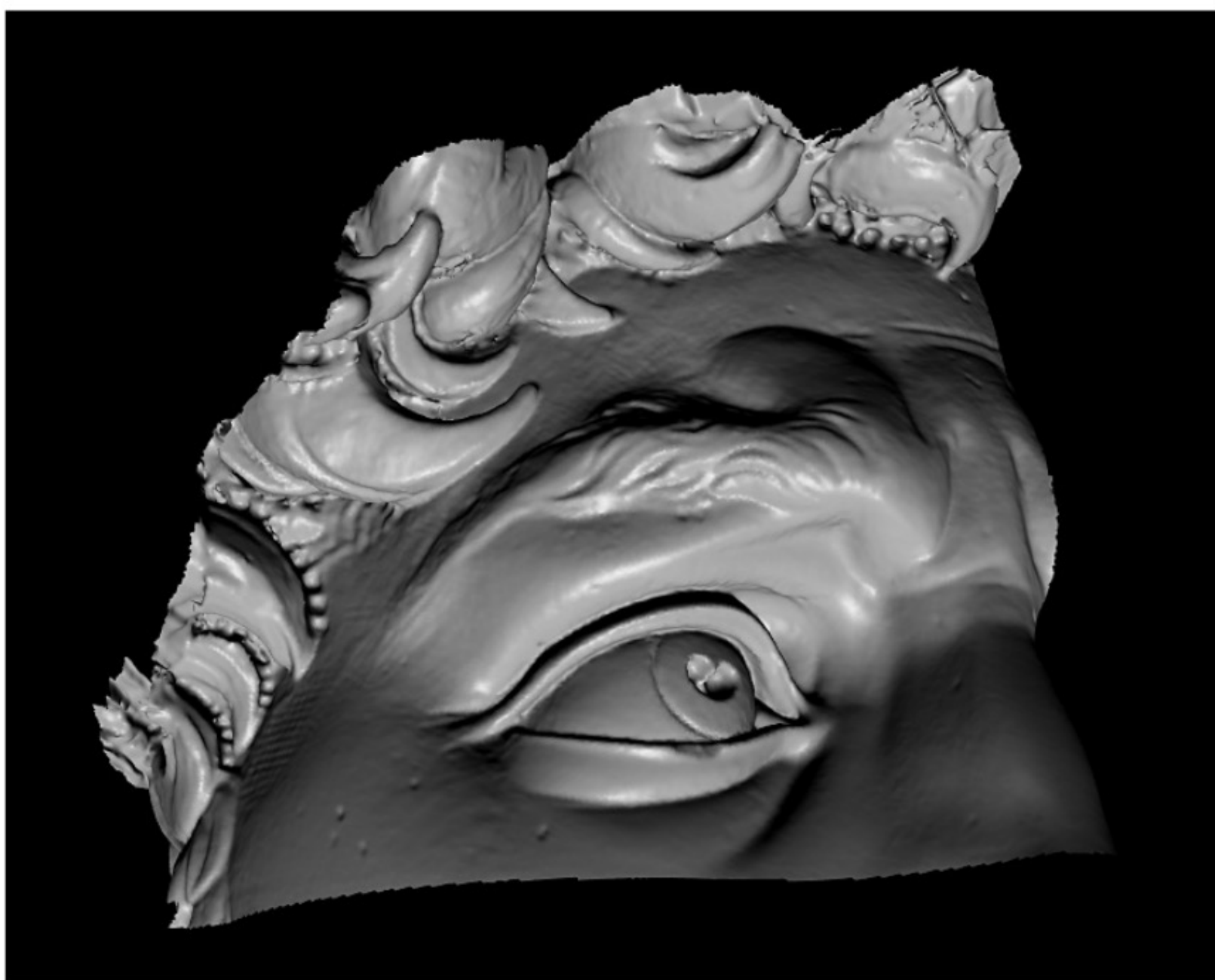
Digital Michelangelo Project
<http://graphics.stanford.edu/projects/mich/>



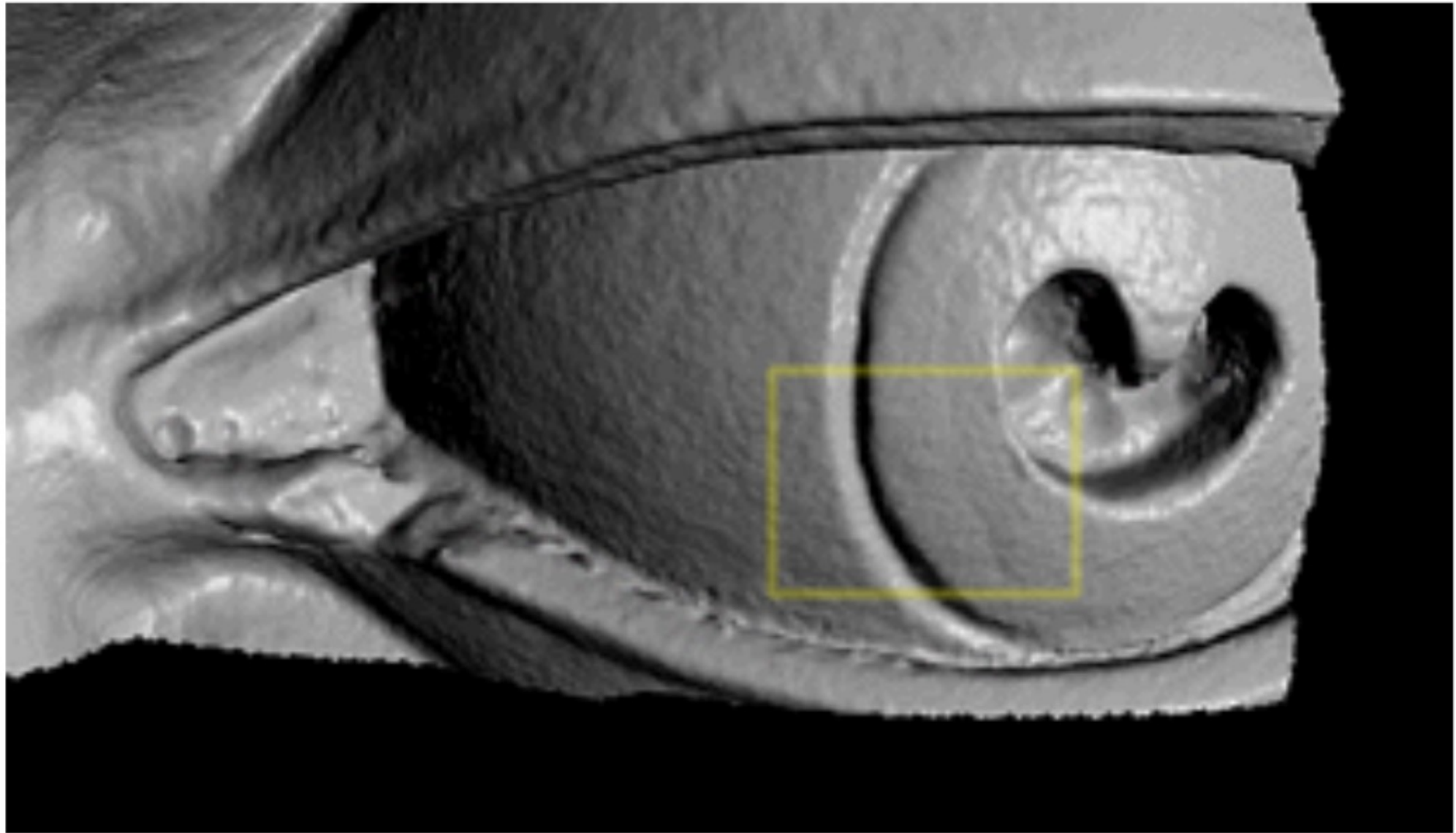
The Digital Michelangelo Project, Levoy et al.



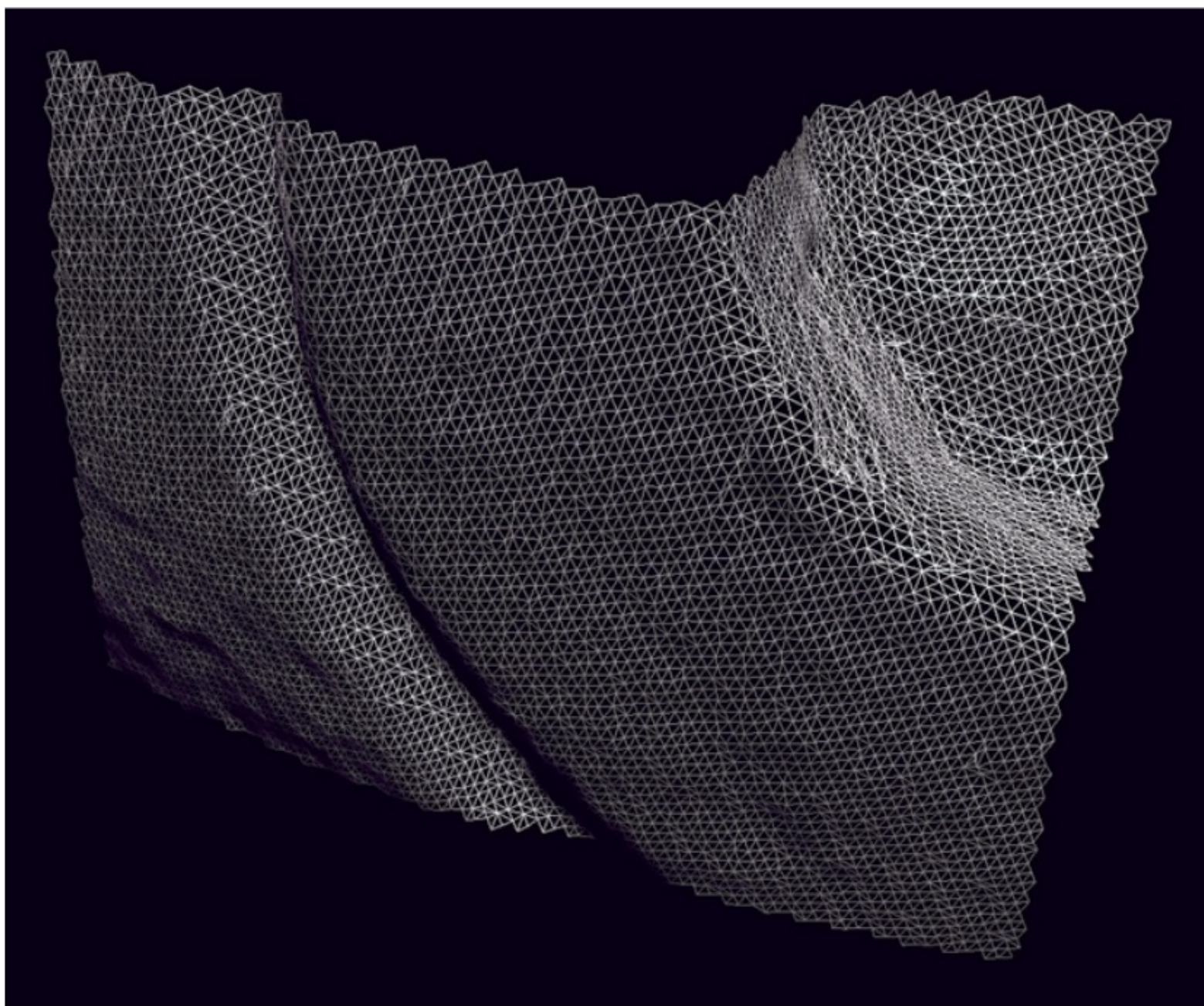
The Digital Michelangelo Project, Levoy et al.



The Digital Michelangelo Project, Levoy et al.



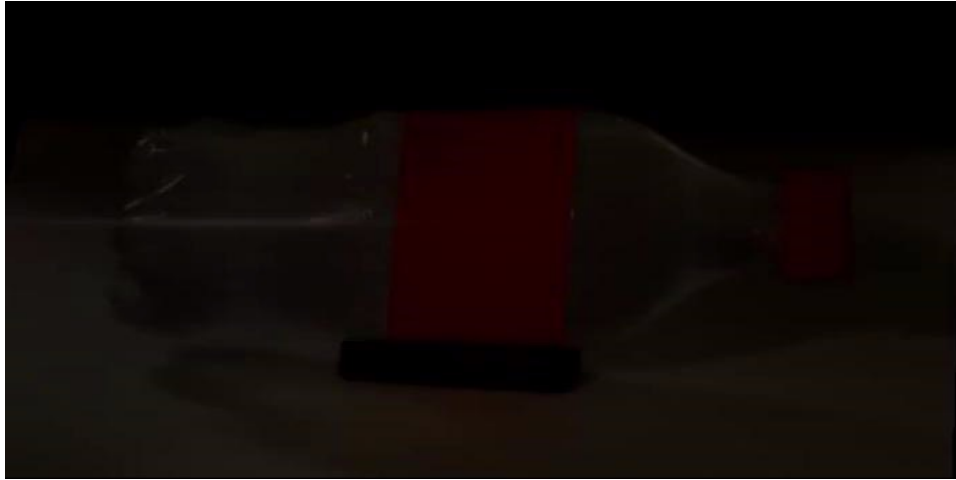
The Digital Michelangelo Project, Levoy et al.



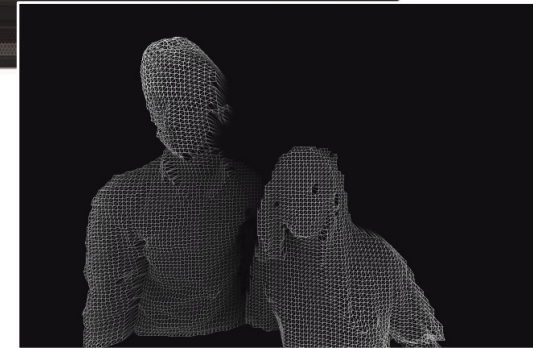
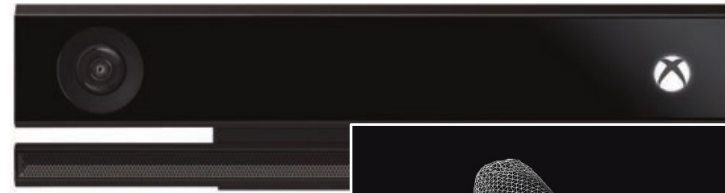
The Digital Michelangelo Project, Levoy et al.

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/>