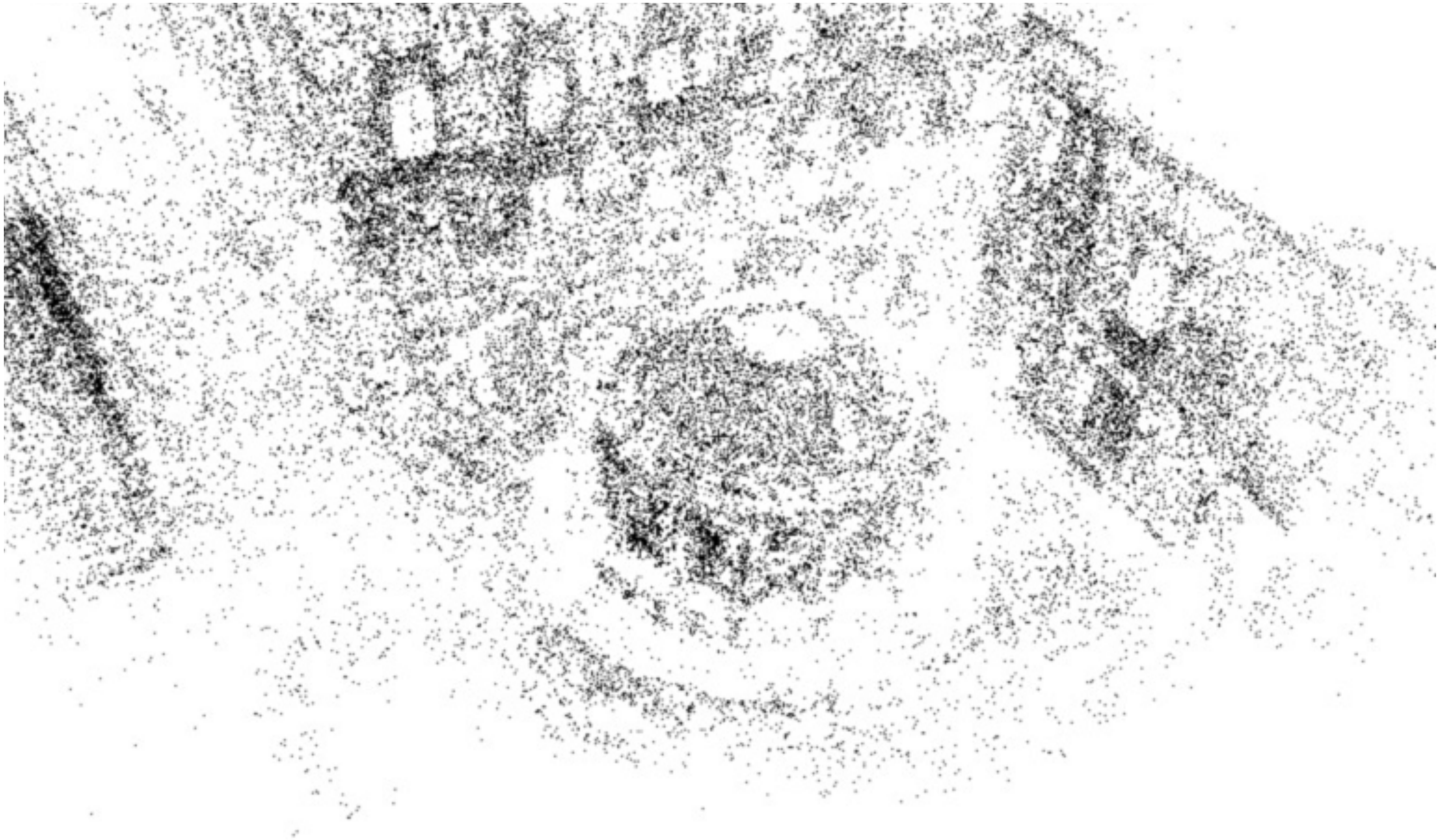


Reconstruction

16-385 Computer Vision

Carnegie Mellon University (Kris Kitani)

	Structure (scene geometry)	Motion (c	Measurements
Pose Estimation	known	estimate	3D to 2D correspondences
Triangulation	estimate	known	2D to 2D coorespondences
Reconstruction	estimate	estimate	2D to 2D coorespondences

Reconstruction

(2 view structure from motion)

Given a set of matched points

$$\{\boldsymbol{x}_i, \boldsymbol{x}'_i\}$$

Estimate the camera matrices

$$\mathbf{P}, \mathbf{P}'$$

Estimate the 3D point

$$\mathbf{X}$$

Procedure for Reconstruction

1. Compute the Fundamental Matrix **F** from points correspondences
   8-point algorithm

$$\boldsymbol{x}'^{\top}_m \mathbf{F} \boldsymbol{x}_m = 0$$

Procedure for Reconstruction

1. Compute the Fundamental Matrix **F** from points correspondences
   8-point algorithm

2. Compute the camera matrices **P** from the Fundamental matrix
   $\mathbf{P} = [\ \mathbf{I} \mid \mathbf{0}\ ]$ and $\mathbf{P}' = [\ [\mathbf{e}'_{\times}]\mathbf{F} \mid \mathbf{e}'\ ]$

Camera matrices corresponding to the fundamental matrix **F** may be chosen as

$$\mathbf{P} = [\mathbf{I}|\mathbf{0}] \quad \mathbf{P}' = [[\boldsymbol{e}_{\times}]\mathbf{F}|\boldsymbol{e}']$$

(See Hartley and Zisserman C.9 for proof)

Decomposing **F** into **R** and **T**

If we have calibrated cameras we have $\mathbf{K}$ and $\mathbf{K}'$

Essential matrix: $\mathbf{E} = \mathbf{K}'^{\top}\mathbf{F}\mathbf{K}$

Decomposing **F** into **R** and **T**

If we have calibrated cameras we have $\mathbf{K}$ and $\mathbf{K}'$

Essential matrix: $\mathbf{E} = \mathbf{K}'^\top \mathbf{F} \mathbf{K}$

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

Decomposing **F** into **R** and **T**

If we have calibrated cameras we have $\mathbf{K}$ and $\mathbf{K}'$

Essential matrix: $\mathbf{E} = \mathbf{K}'^\top \mathbf{F} \mathbf{K}$

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:

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

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

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

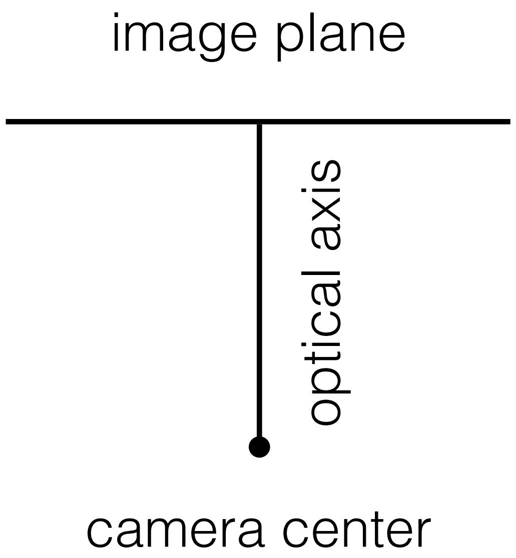
(note: det(R) = -1 means rotation and reflection)

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

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

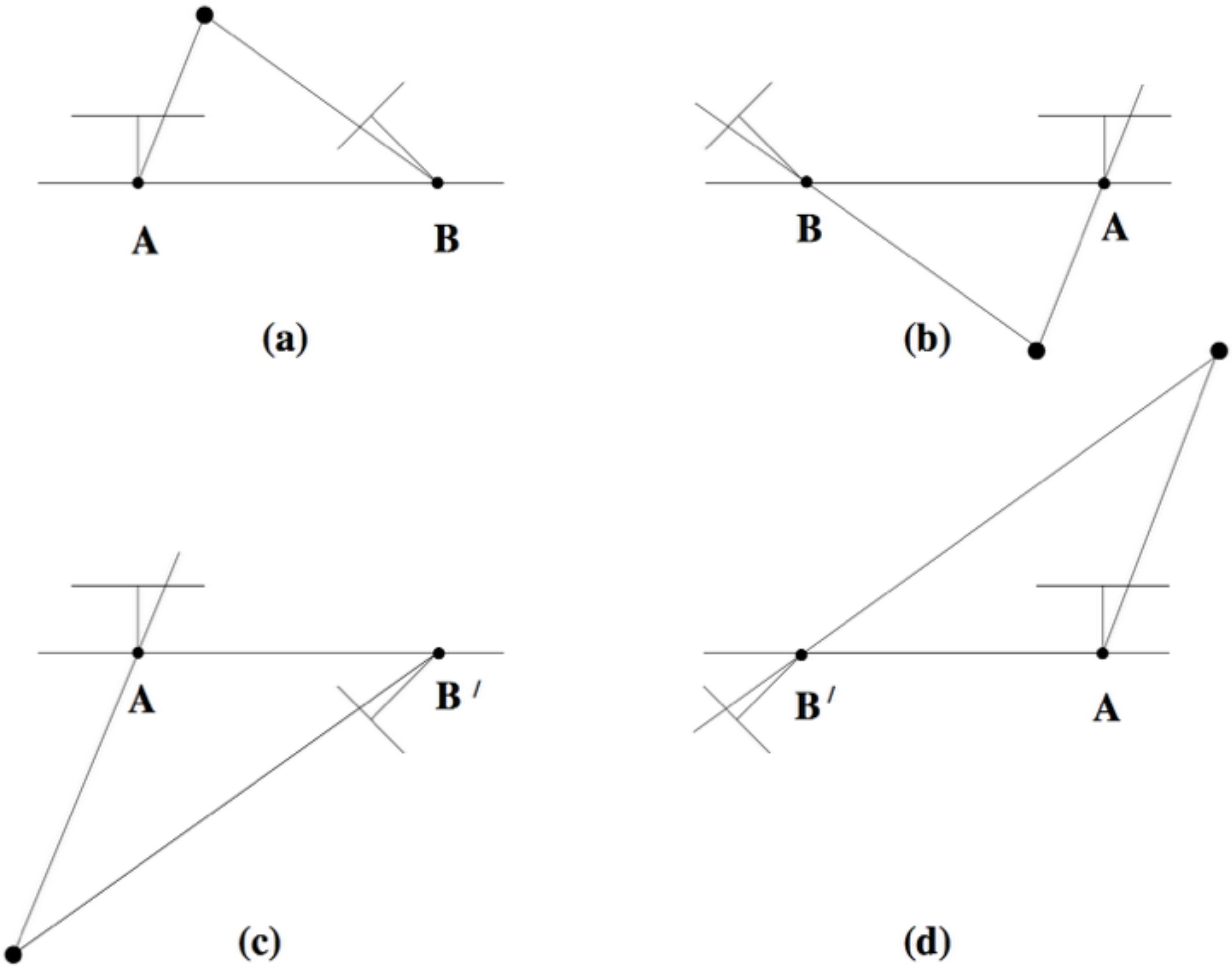
Let’s visualize the four configurations…

Camera Icon



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

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



From points correspondences to camera displacement

1. Normalize the image points **x**,**x′** using **K**,**K′**

2. Use the 8-point algorithm to find an approximation of **E** (SVD!)

3. Project **E** to essential space (SVD!!)

4. Recover possible solutions for **R** and **T** (SVD!!!)

5. Use point correspondence to find the correct **R**,**T** pair (don’t use SVD…)

Procedure for Reconstruction

1. Compute the Fundamental Matrix **F** from points correspondences

   8-point algorithm

2. Compute the camera matrices **P** from the Fundamental matrix

   $\mathbf{P} = [\ \mathbf{I} \mid \mathbf{0}\ ]$ and $\mathbf{P}' = [\ [\mathbf{e}'_{x}]\mathbf{F} \mid \mathbf{e}'\ ]$

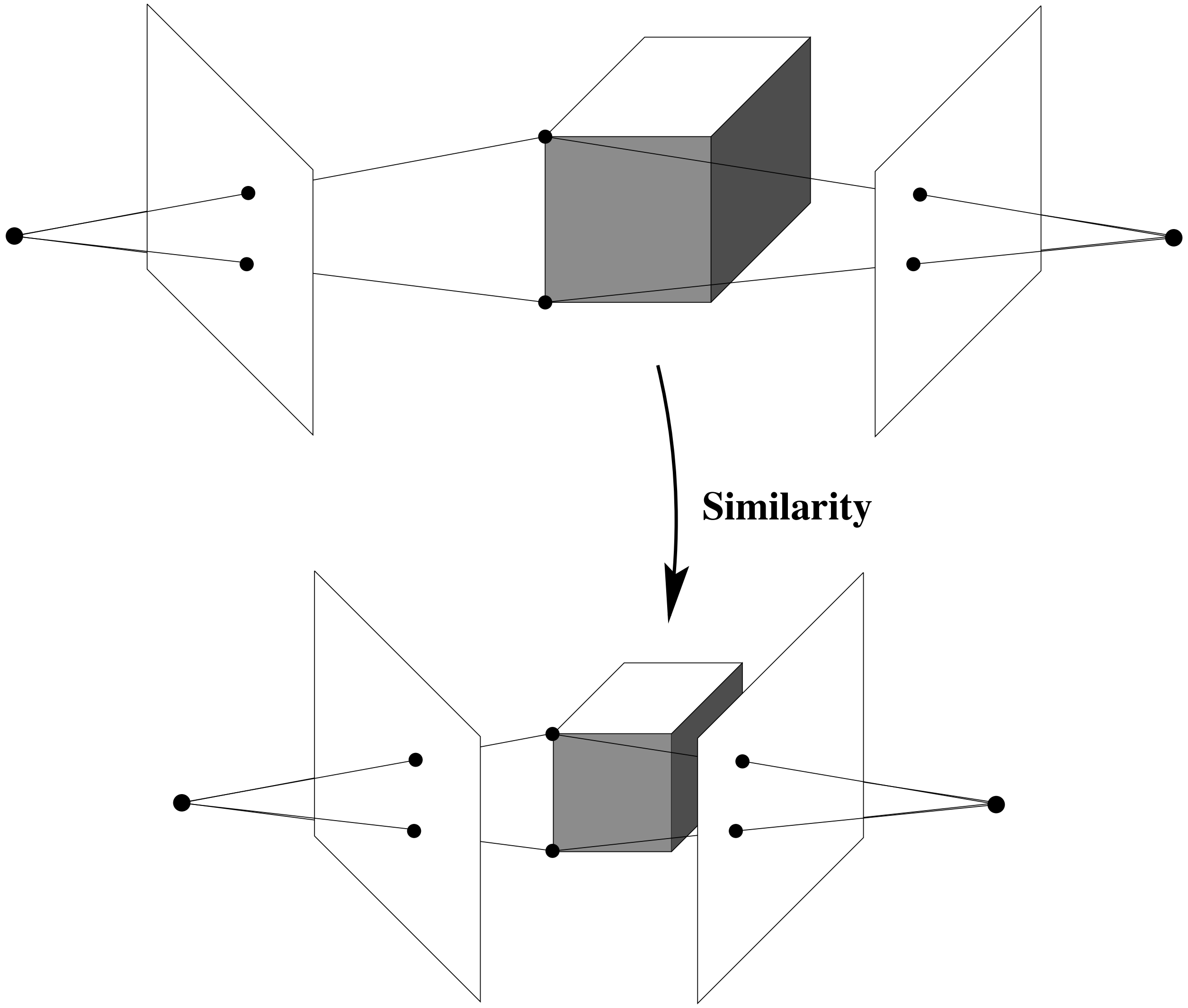
3. For each point correspondence, compute the point **X** in 3D space (triangularization)

   DLT with $x = \mathbf{P}\, X$ and $x' = \mathbf{P}'\, X$

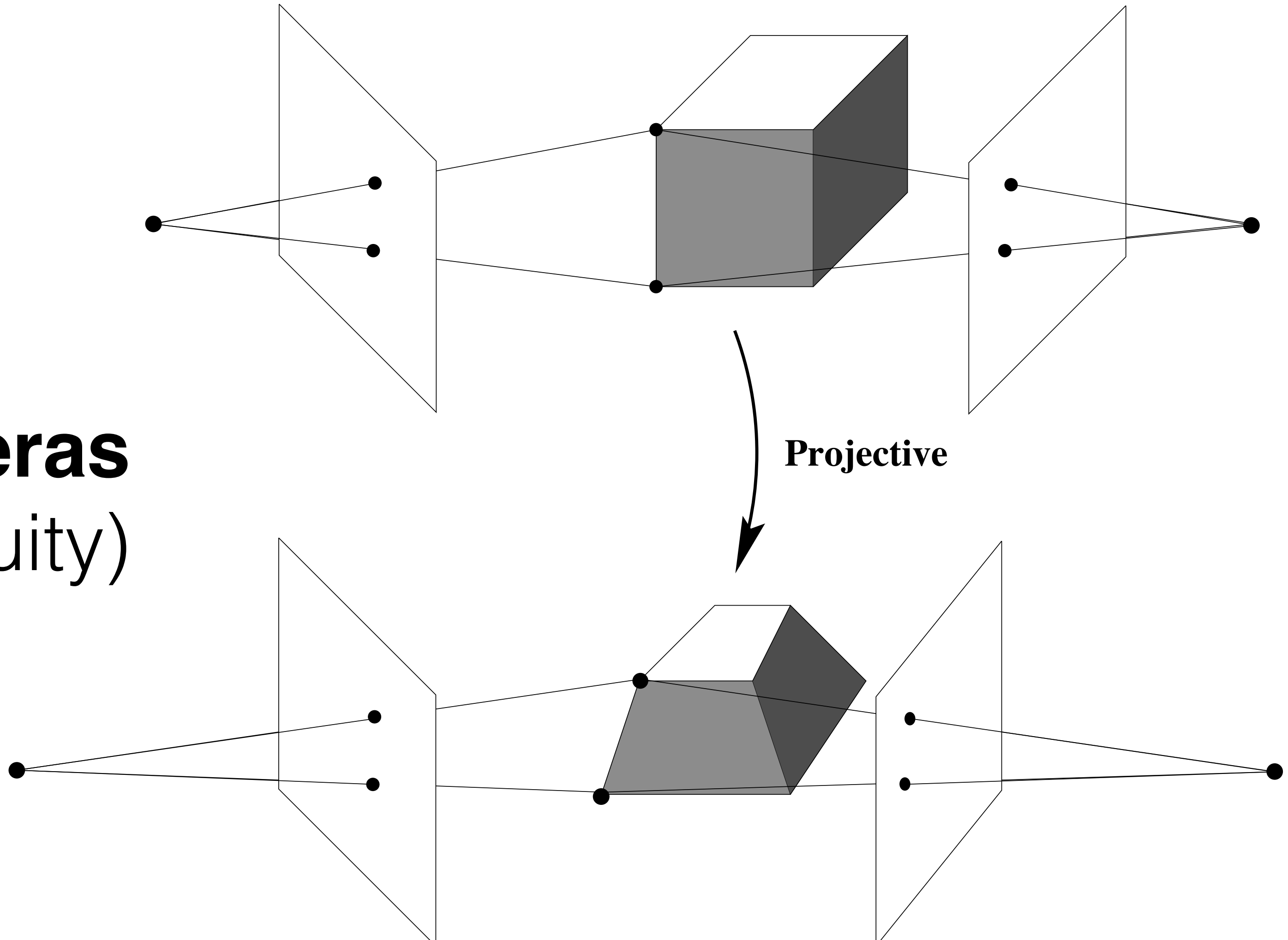
Projective Ambiguity

- Reconstruction is ambiguous by an arbitrary 3D projective transformation without prior knowledge of camera parameters

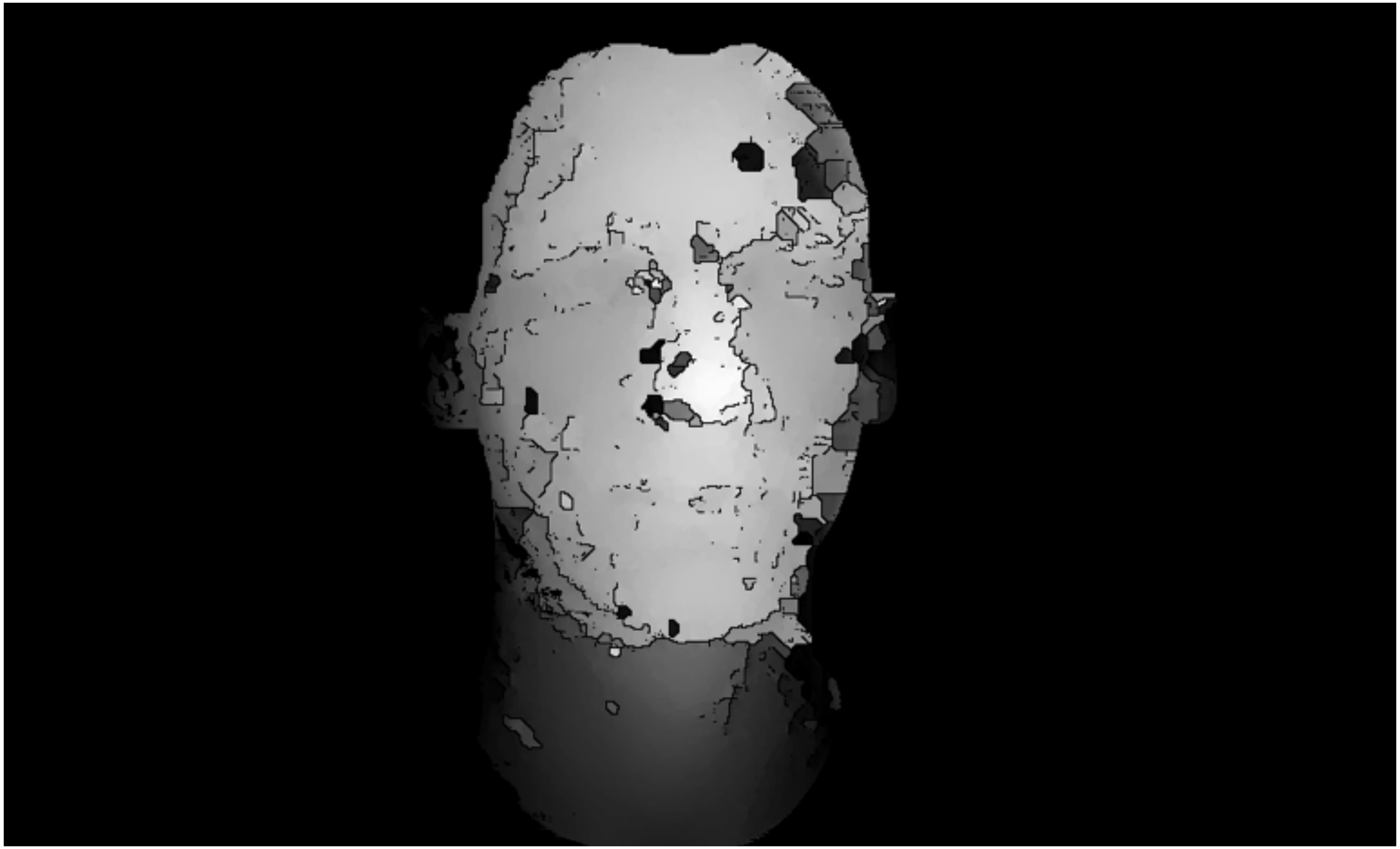
Calibrated cameras
(similarity projection ambiguity)



Uncalibrated cameras
(projective projection ambiguity)



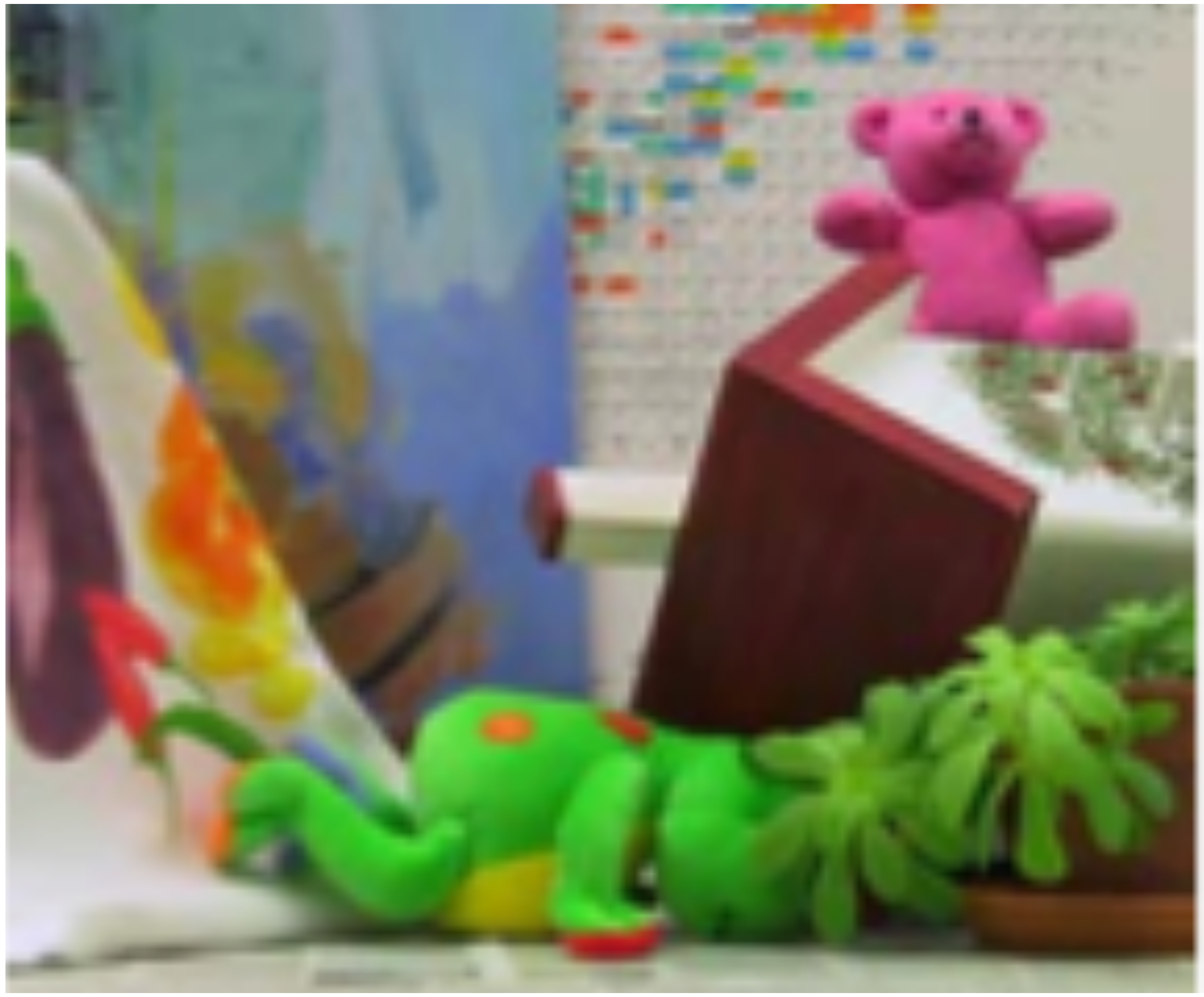
	Structure (scene geometry)	Motion (c	Measurements
Pose Estimation	known	estimate	3D to 2D correspondences
Triangulation	estimate	known	2D to 2D coorespondences
Reconstruction	estimate	estimate	2D to 2D coorespondences

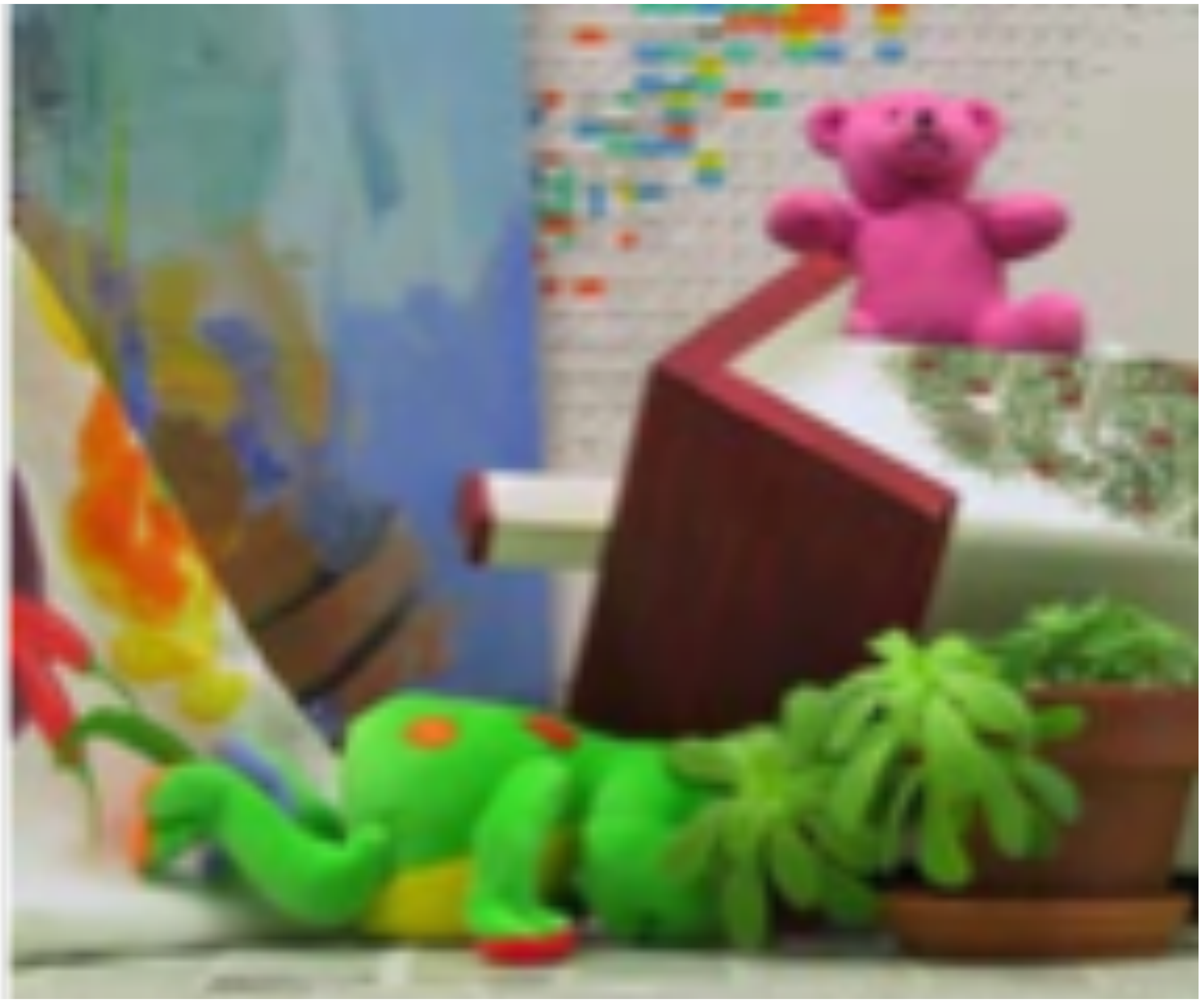


Stereo Vision

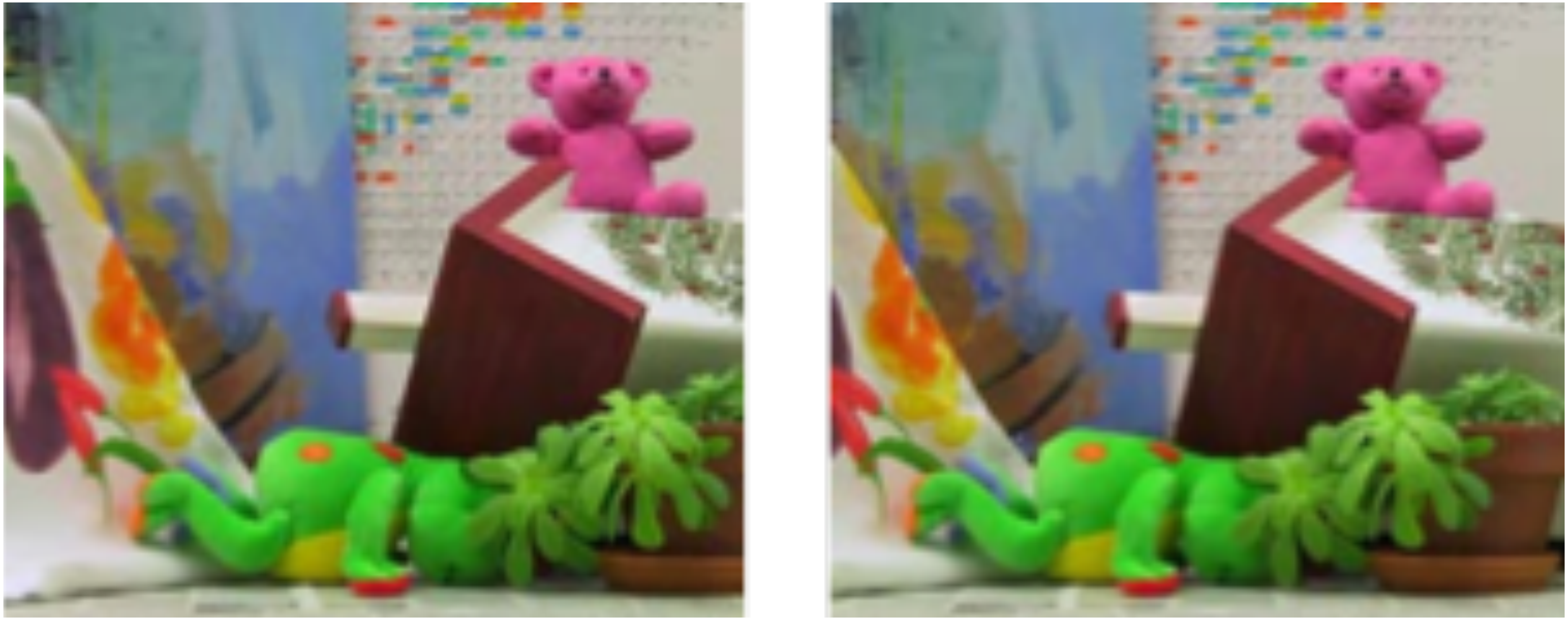
16-385 Computer Vision

Carnegie Mellon University (Kris Kitani)



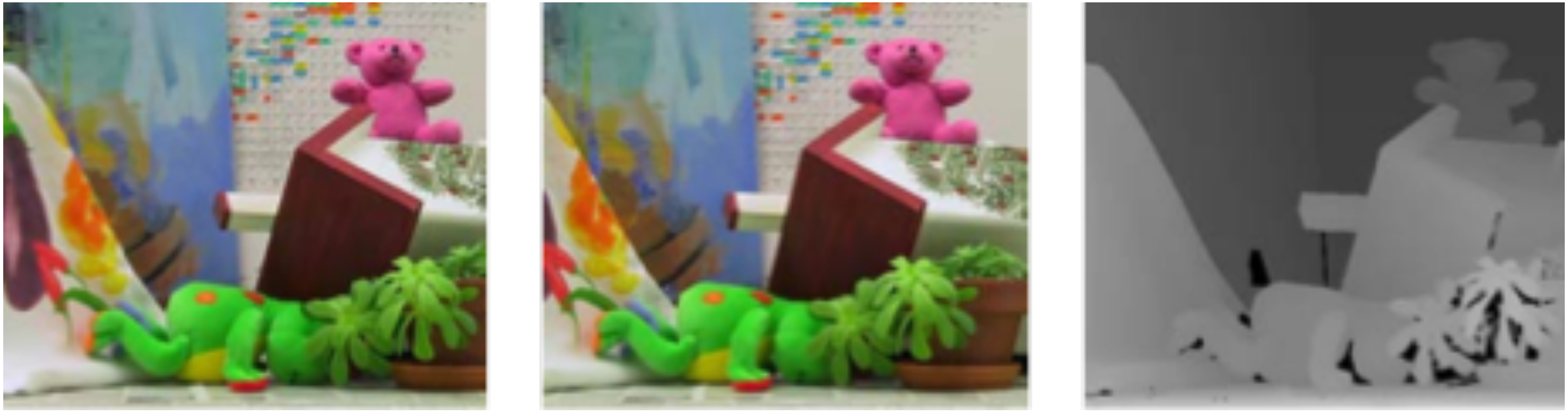


What's different between these two images?

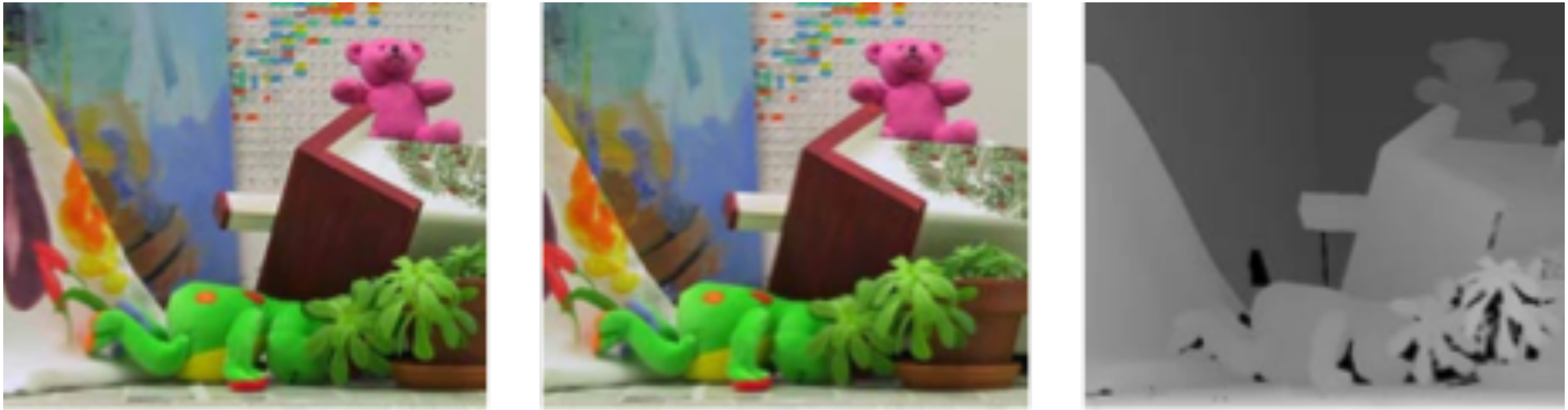


Objects that are close move more or less?

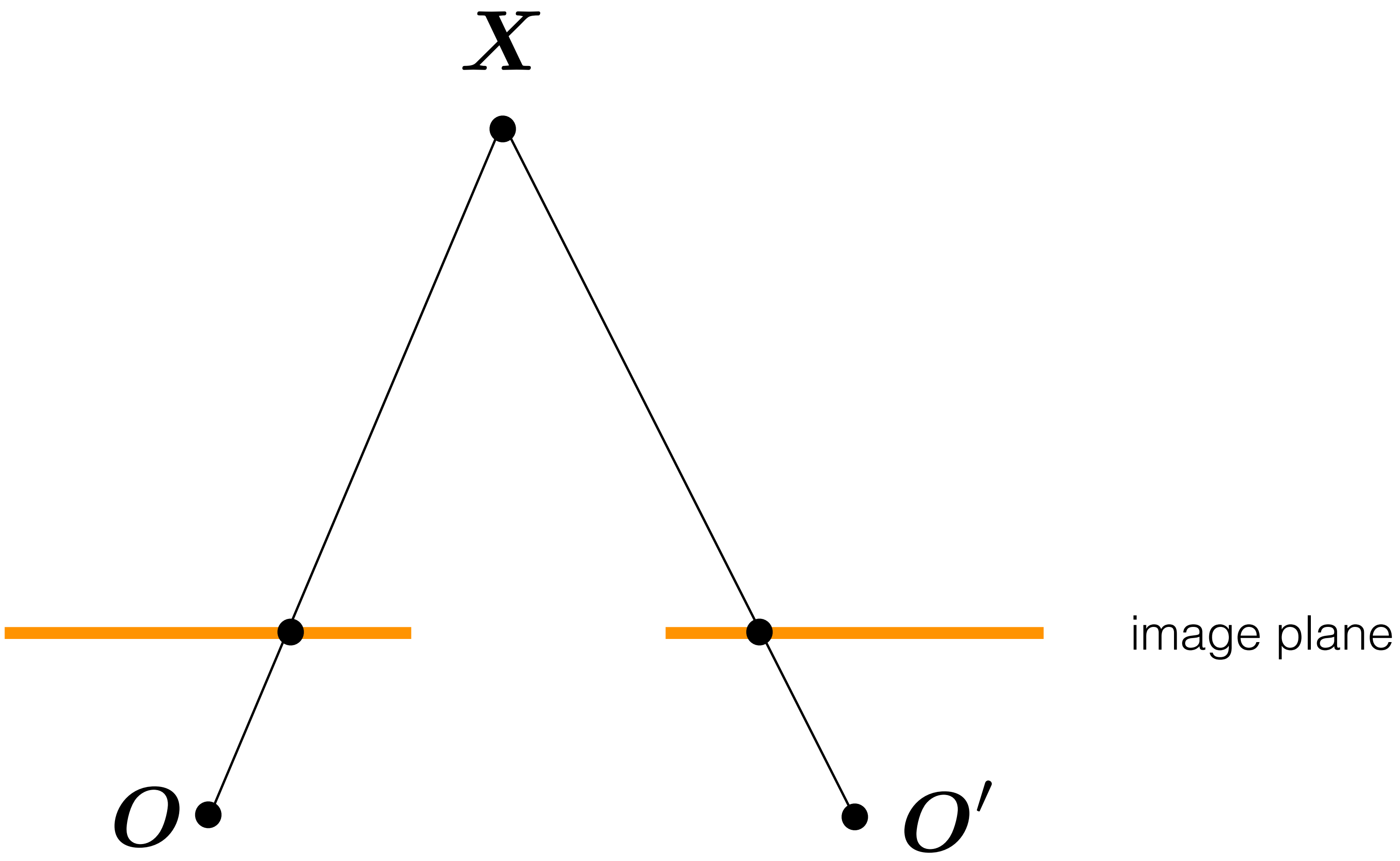
The amount of horizontal movement is inversely proportional to …

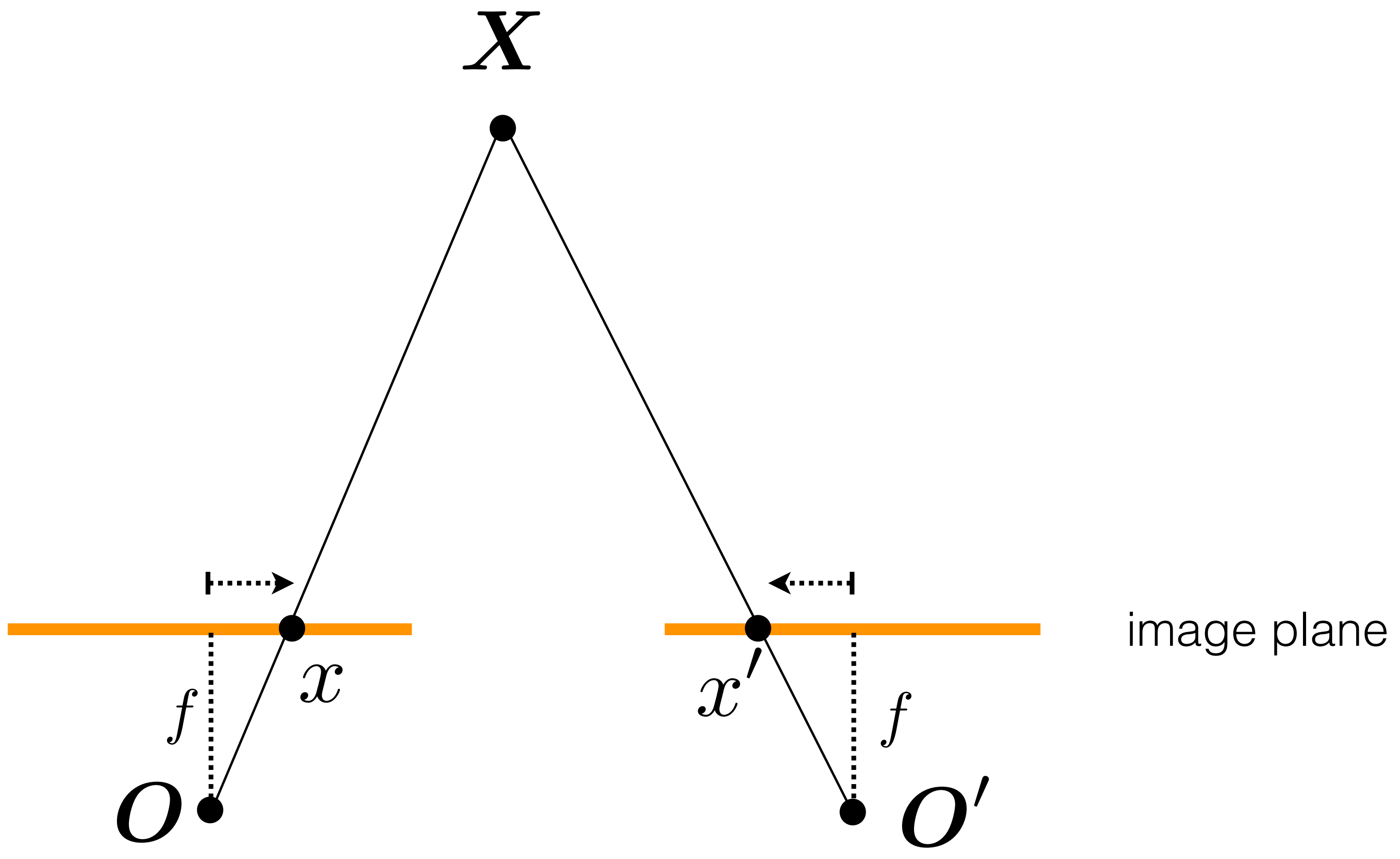


The amount of horizontal movement is inversely proportional to …

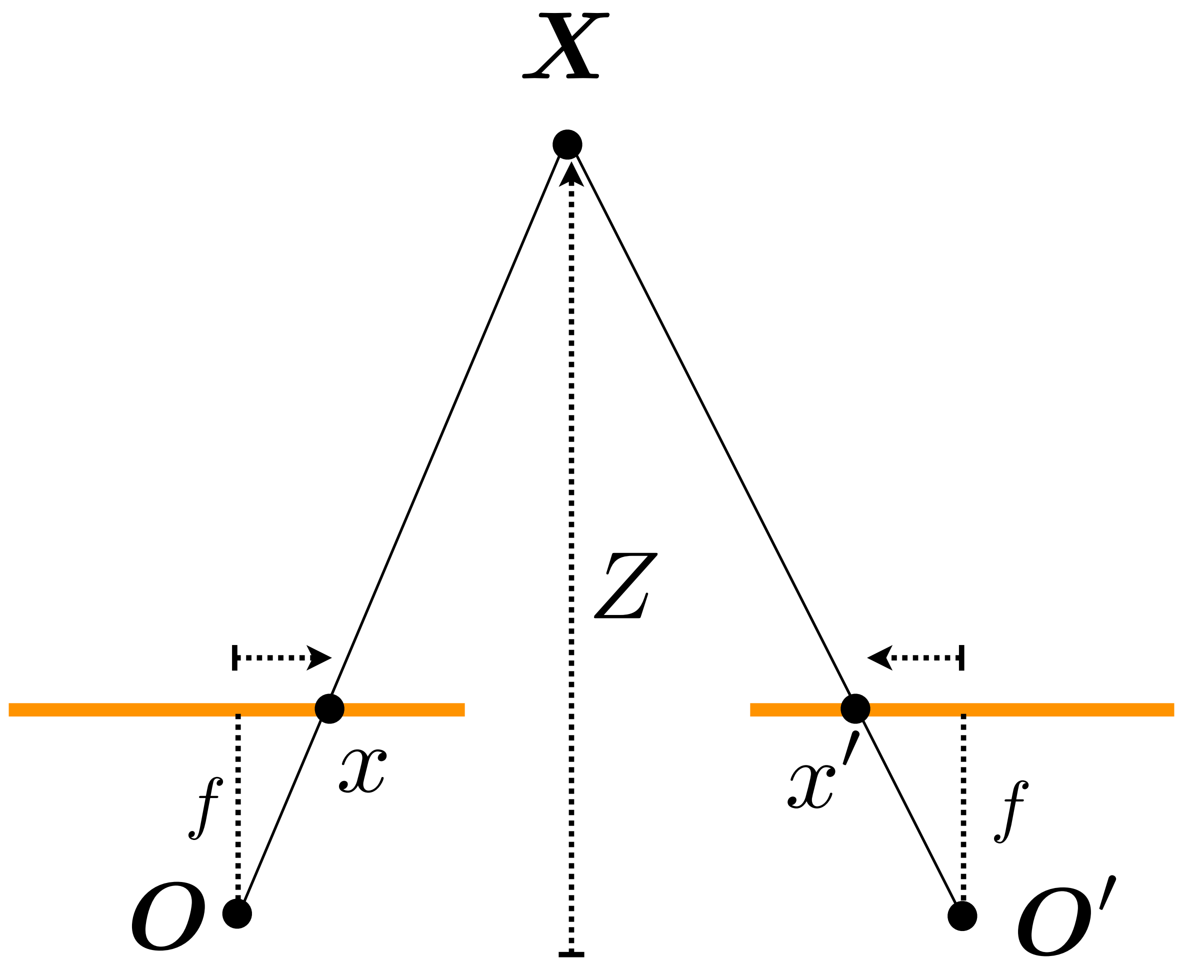


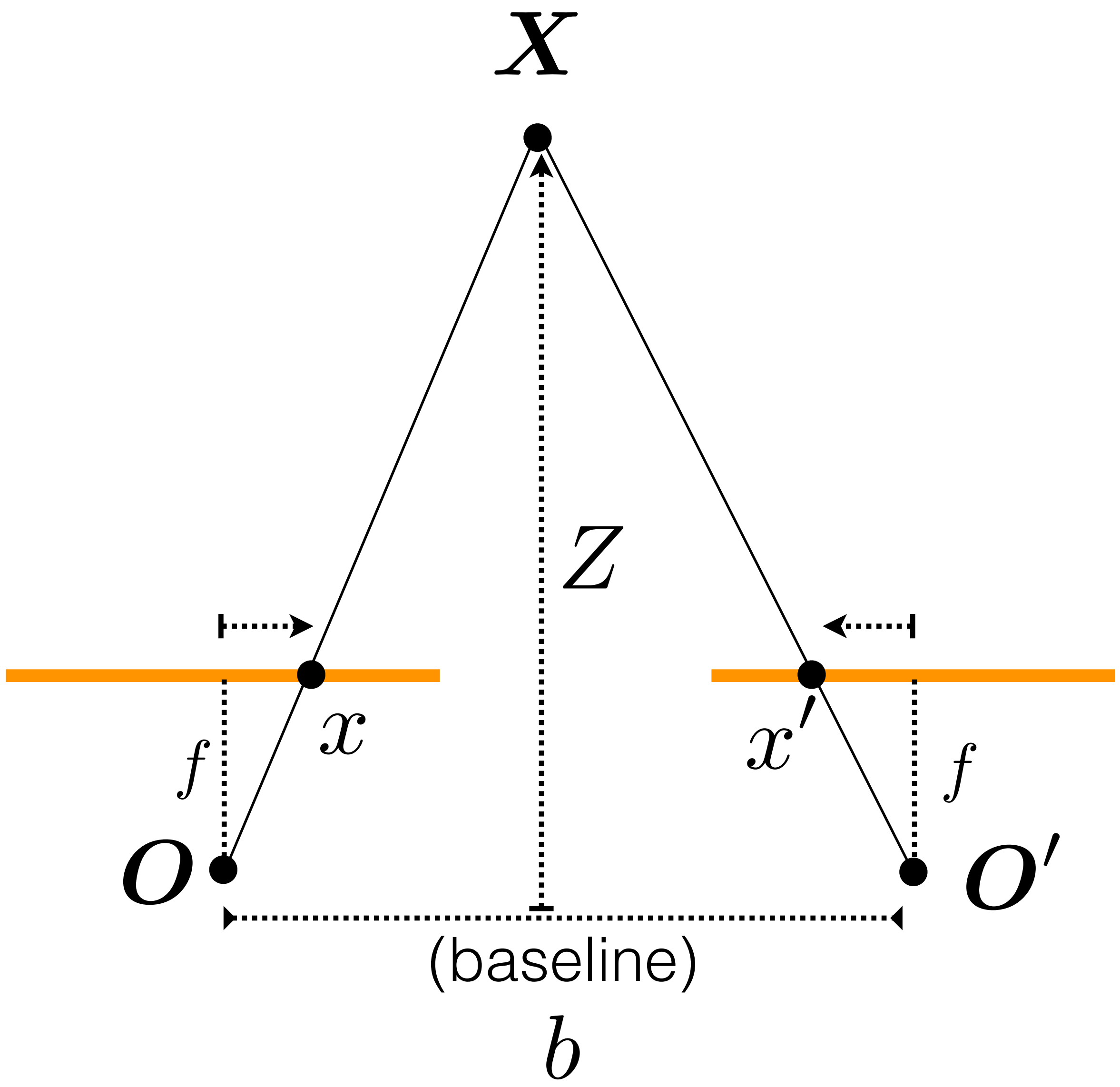
… the distance from the camera.


X
image plane
O
O'

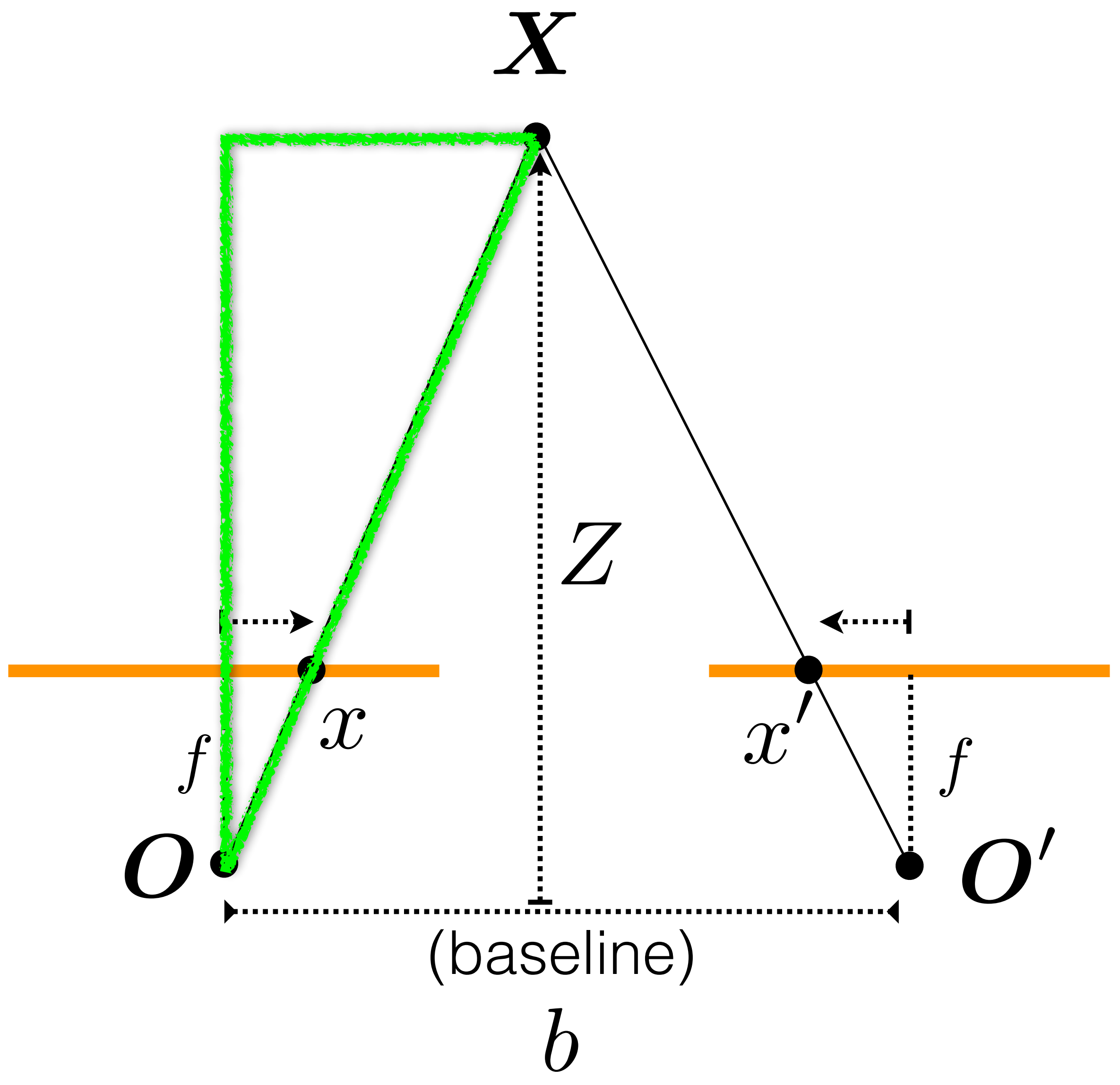


$\boldsymbol{X}$
image plane
x
x'
f
f
$\boldsymbol{O}$
$\boldsymbol{O}'$

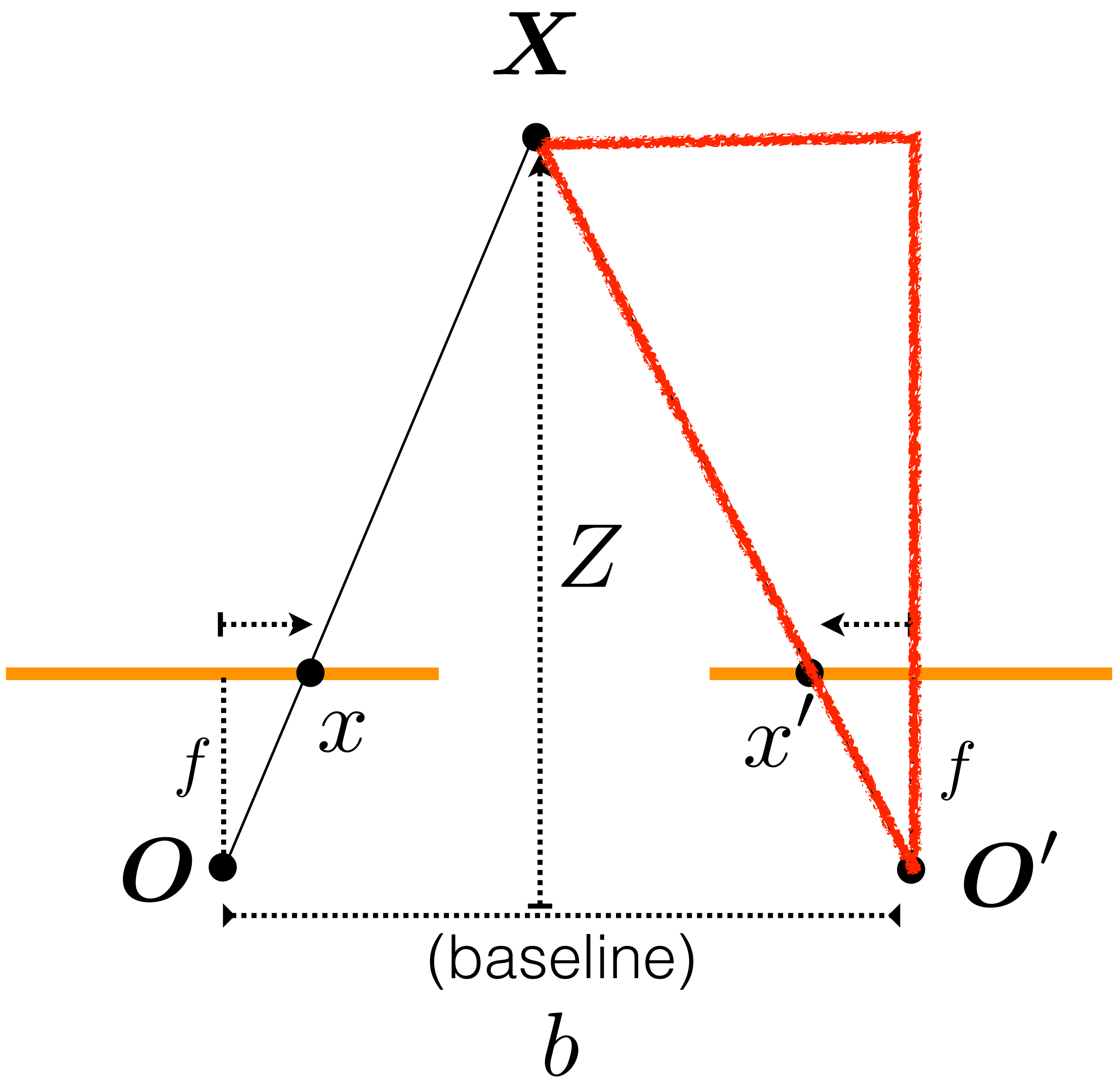

$\boldsymbol{X}$
Z
x
x'
f
f
$\boldsymbol{O}$
$\boldsymbol{O}'$


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

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

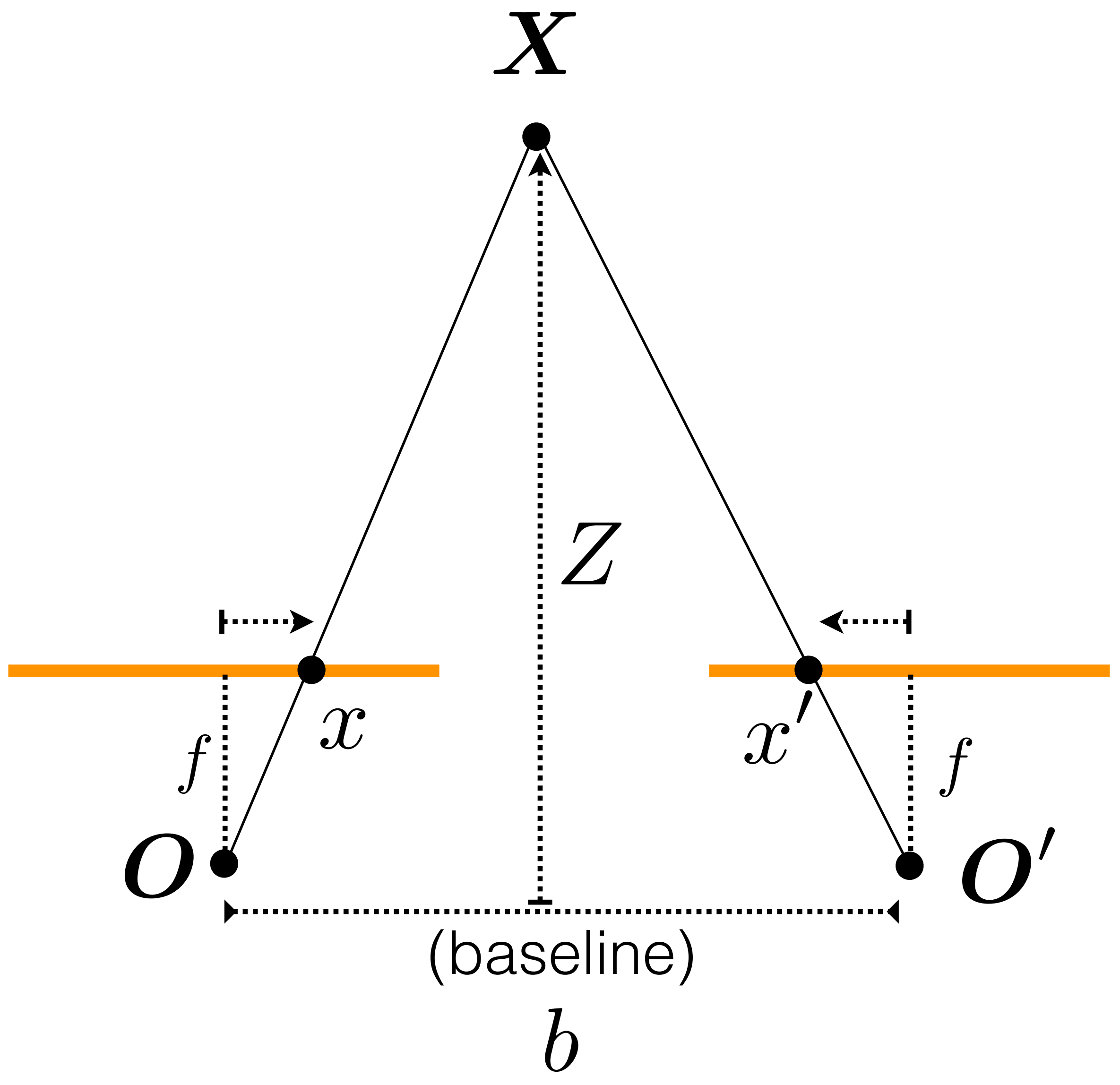


$$\frac{X}{Z} = \frac{x}{f}$$



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

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



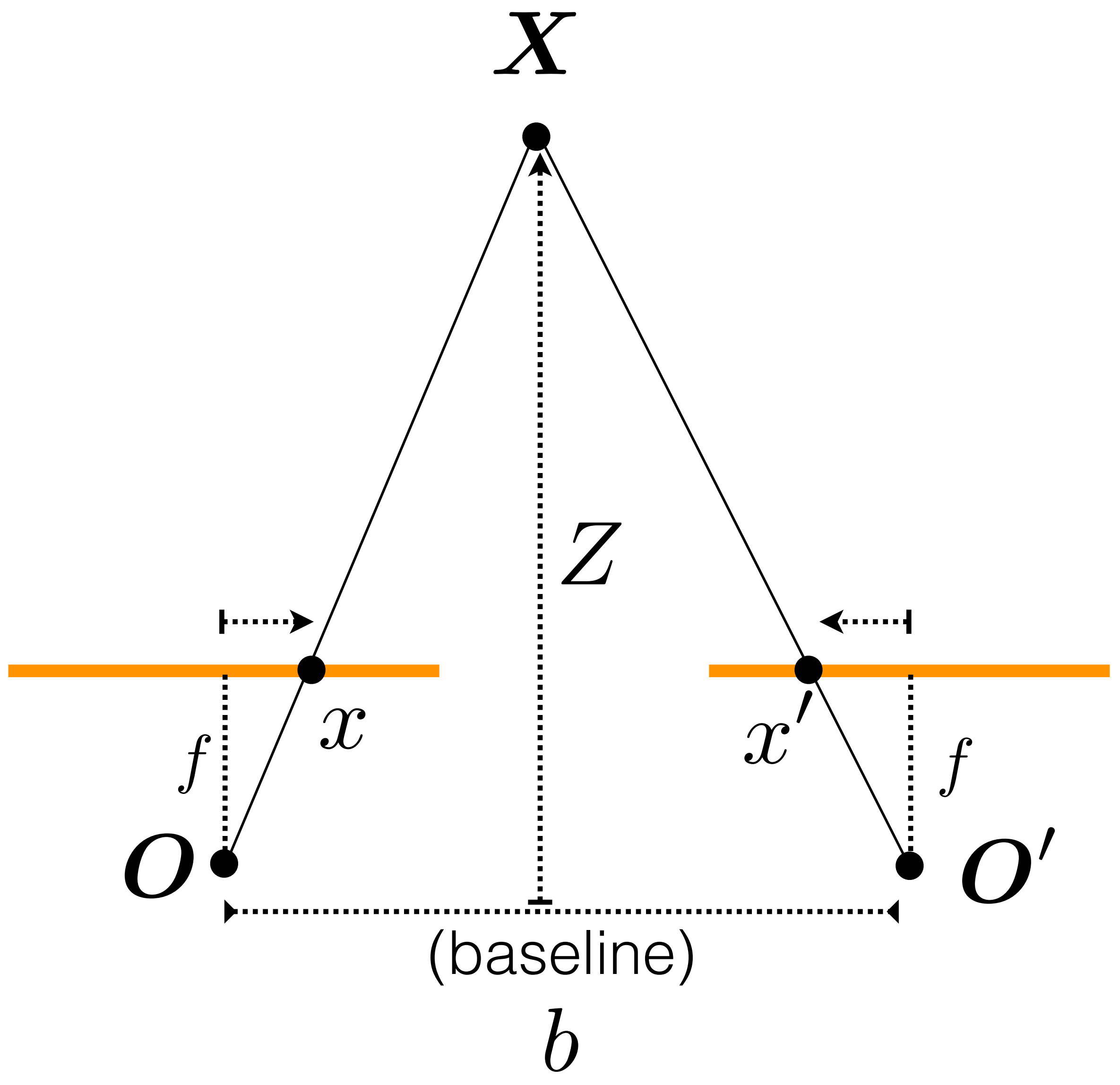
$$\frac{b - X}{Z} = \frac{x'}{f}$$

Disparity

$$d = x - x'$$

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

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



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

Disparity

$$d = x - x'$$

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

inversely proportional to depth

Real-time stereo sensing

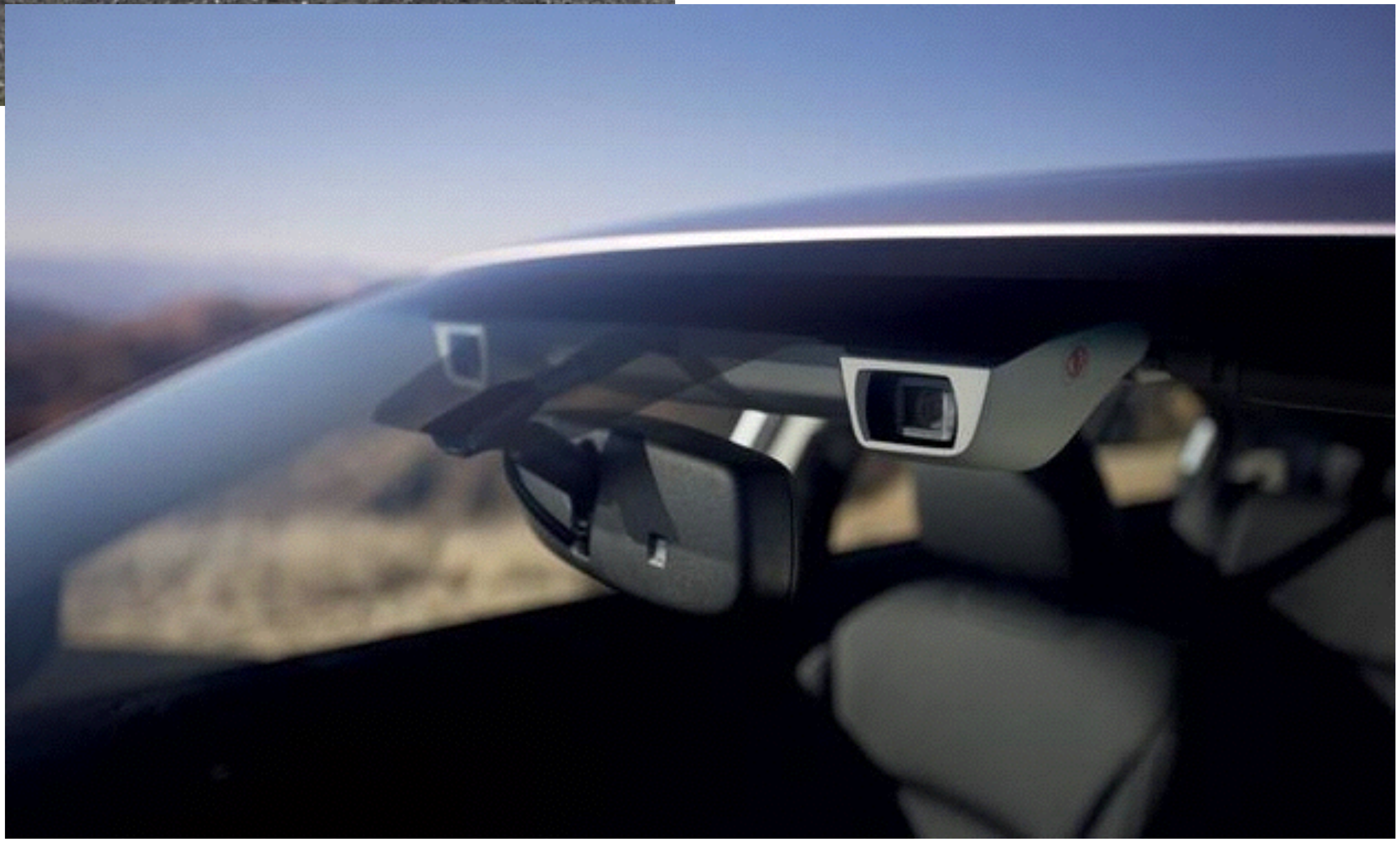
Nomad robot searches for meteorites in Antartica

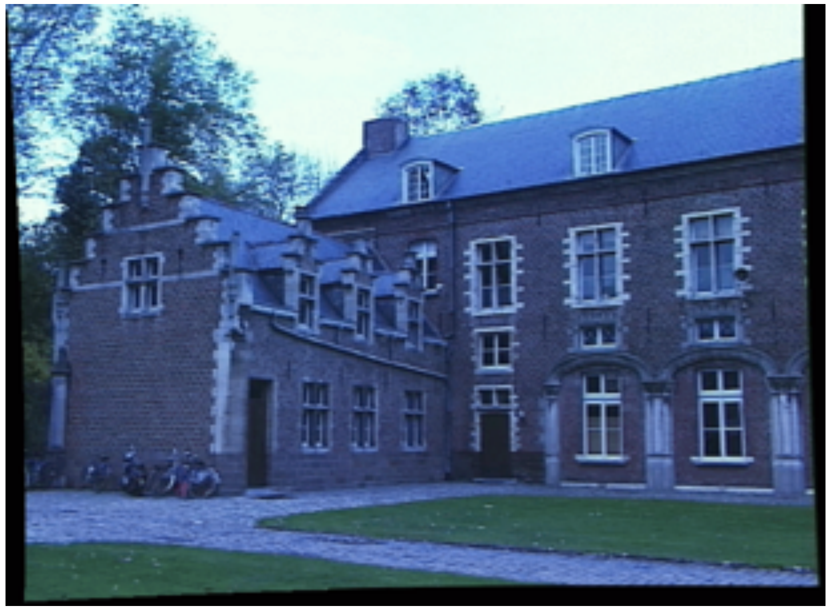
http://www.frc.ri.cmu.edu/projects/meteorobot/index.html

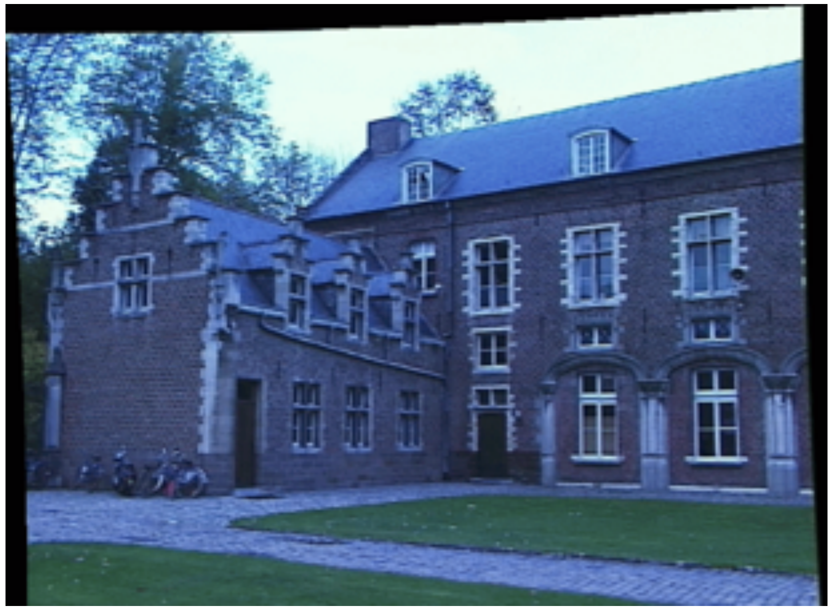
Navigabilty Map
VFH

Subaru
Eyesight system

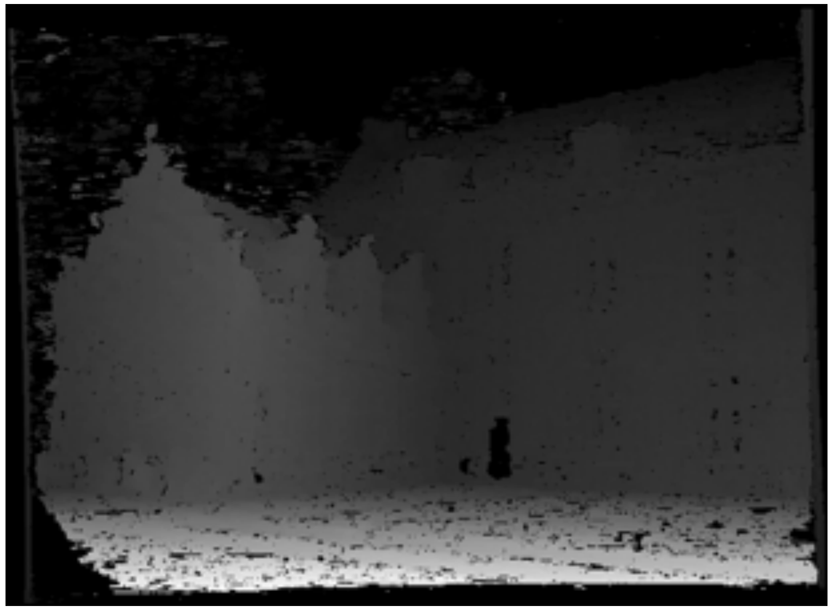
Pre-collision
braking

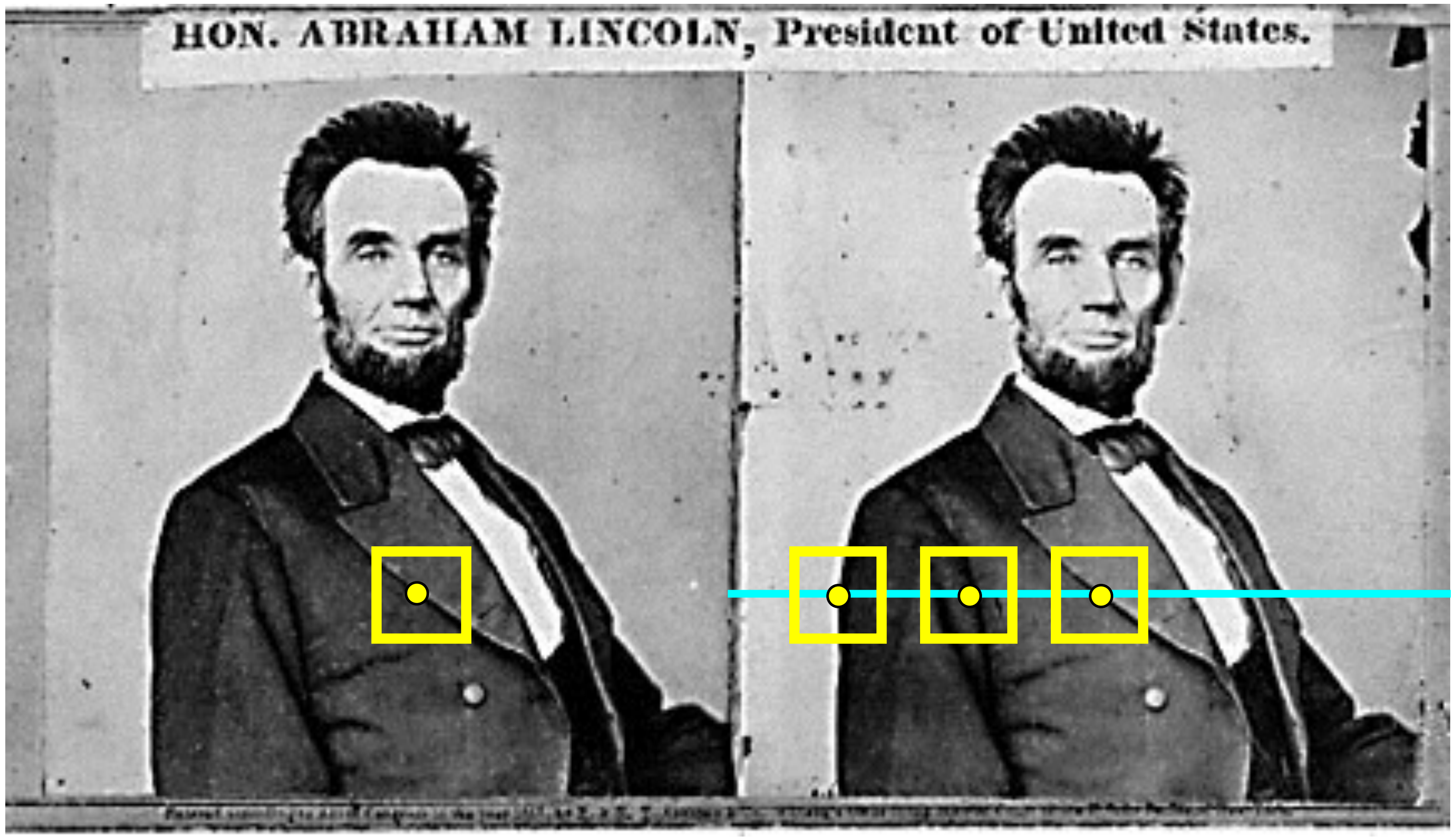






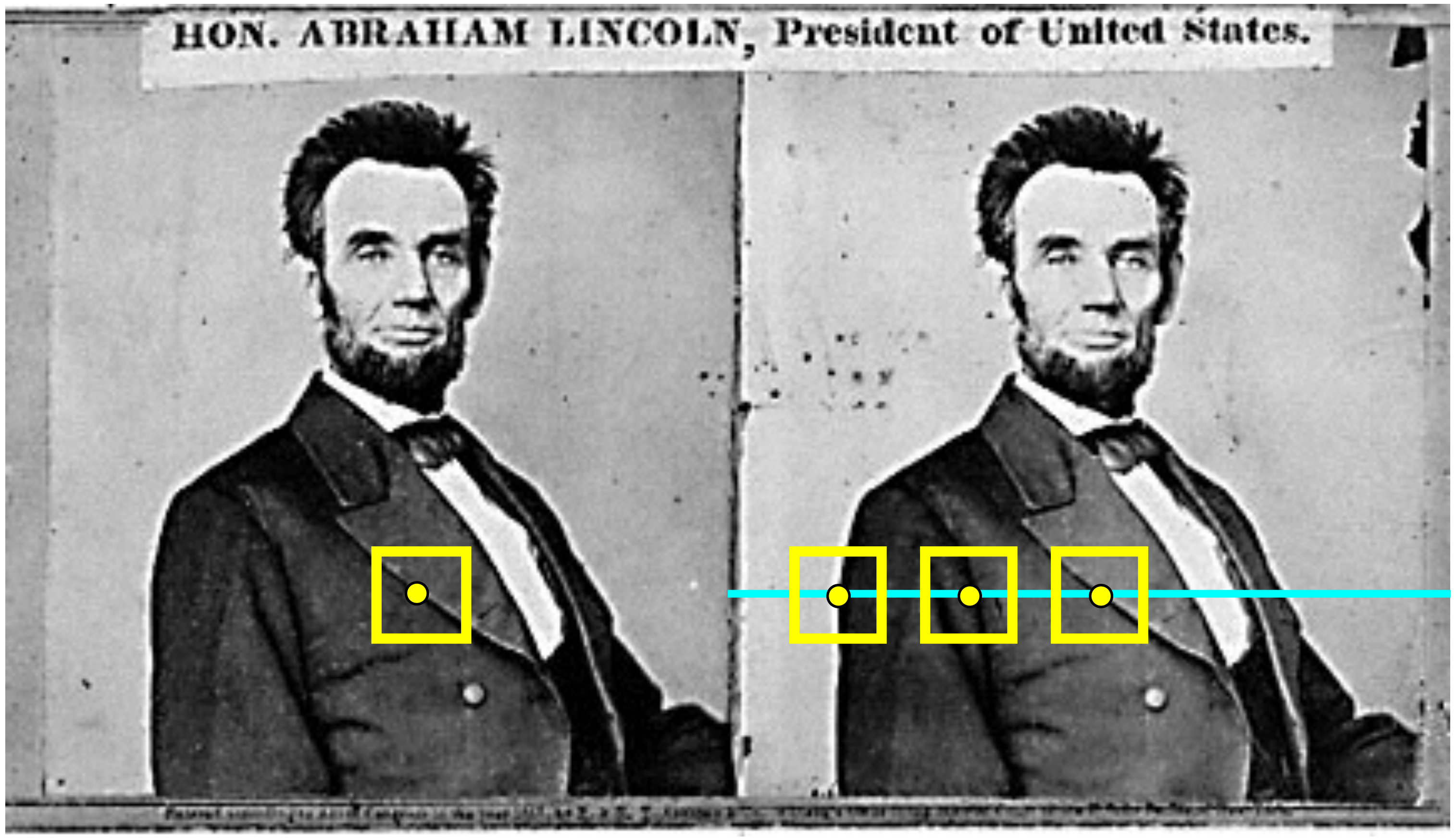
How so you compute depth
from a stereo pair?



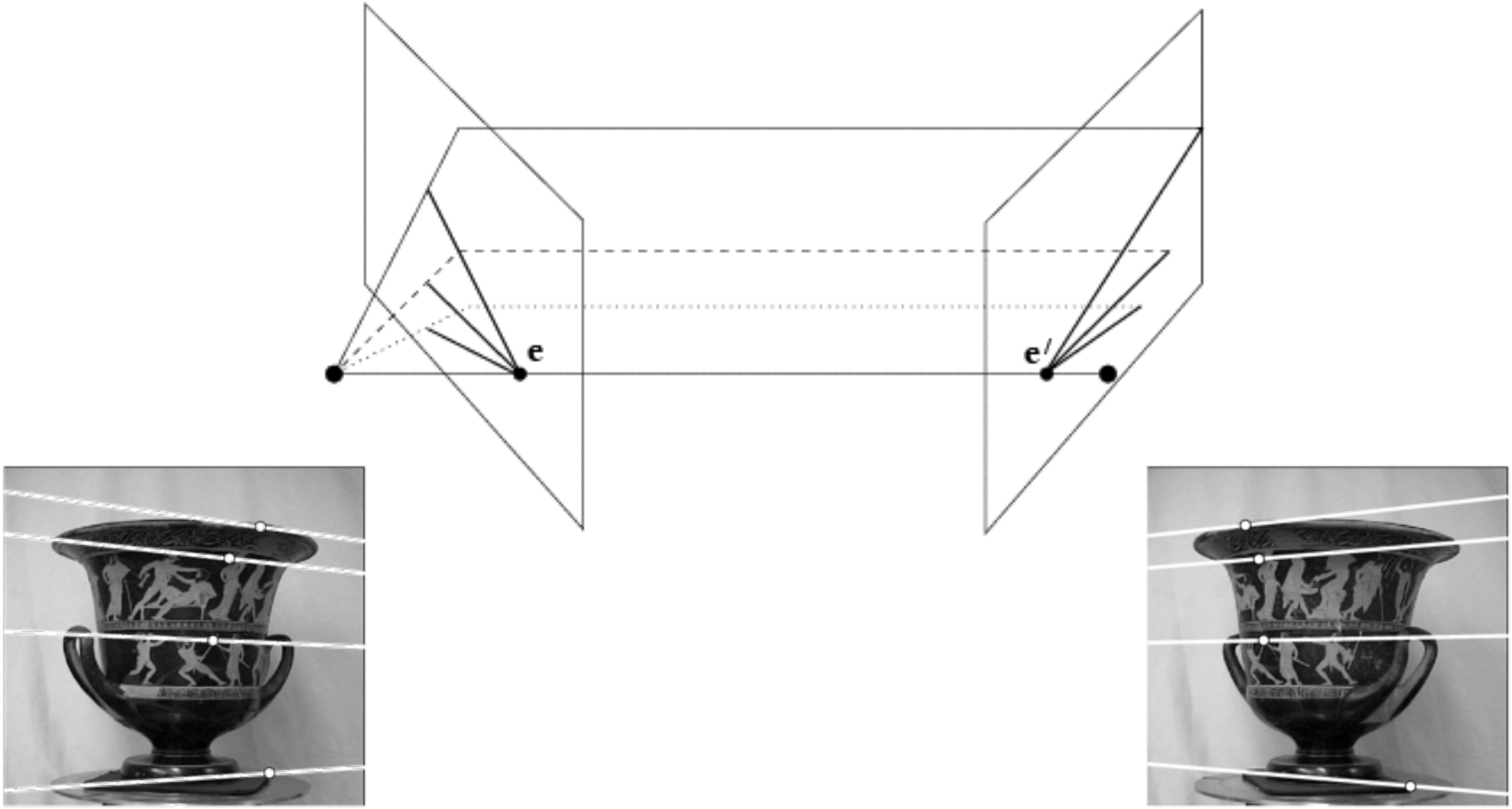


1. Rectify images
   (make epipolar lines horizontal)
2. For each pixel
   a. Find epipolar line
   b. Scan line for best match
   c. Compute depth from disparity

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

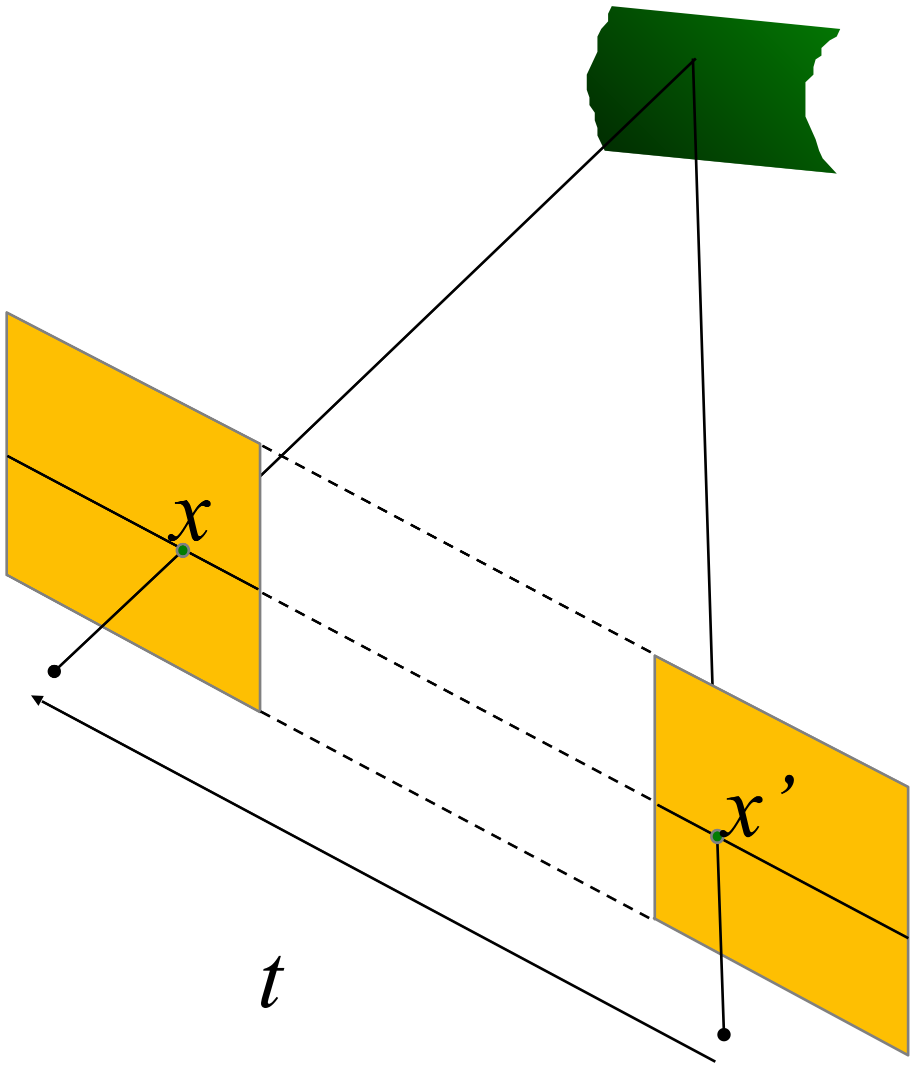


How can you make the epipolar lines horizontal?



It’s hard to make the image planes exactly parallel

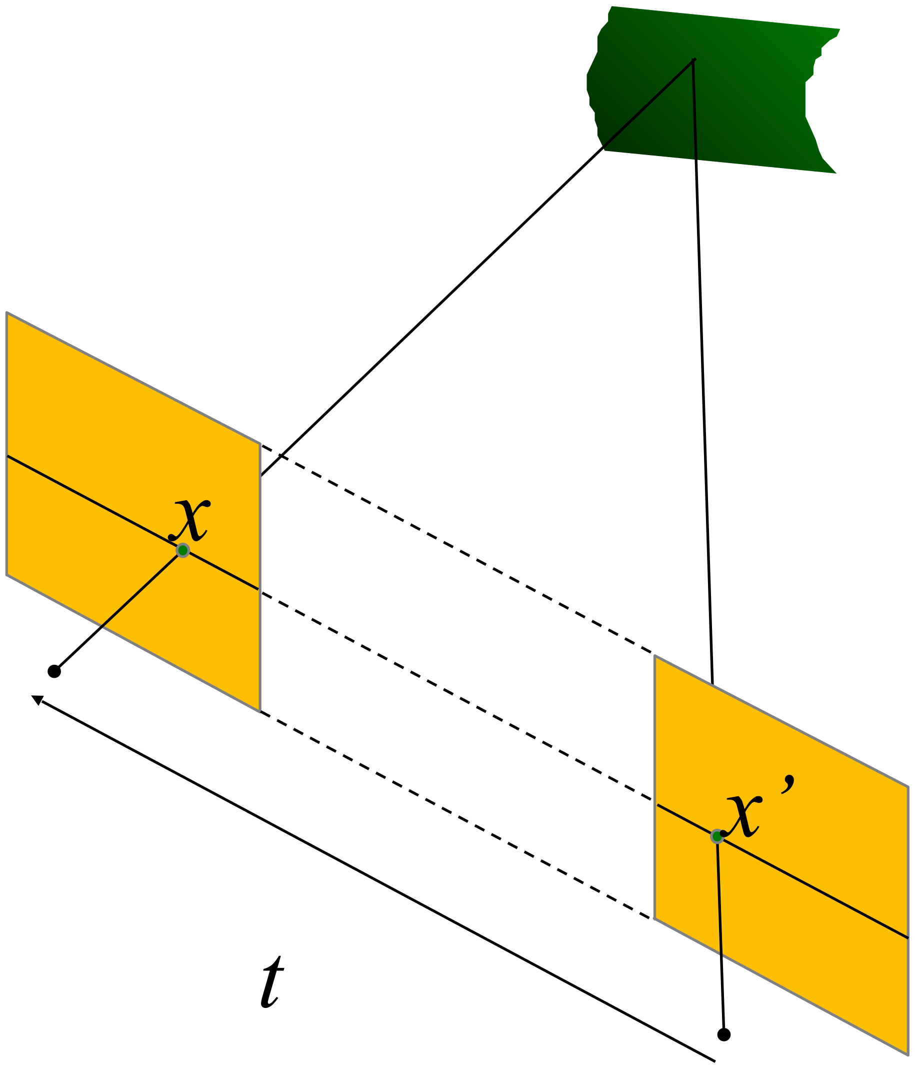
How can you make the epipolar lines horizontal?



When this relationship holds

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

How can you make the epipolar lines horizontal?



When this relationship holds

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

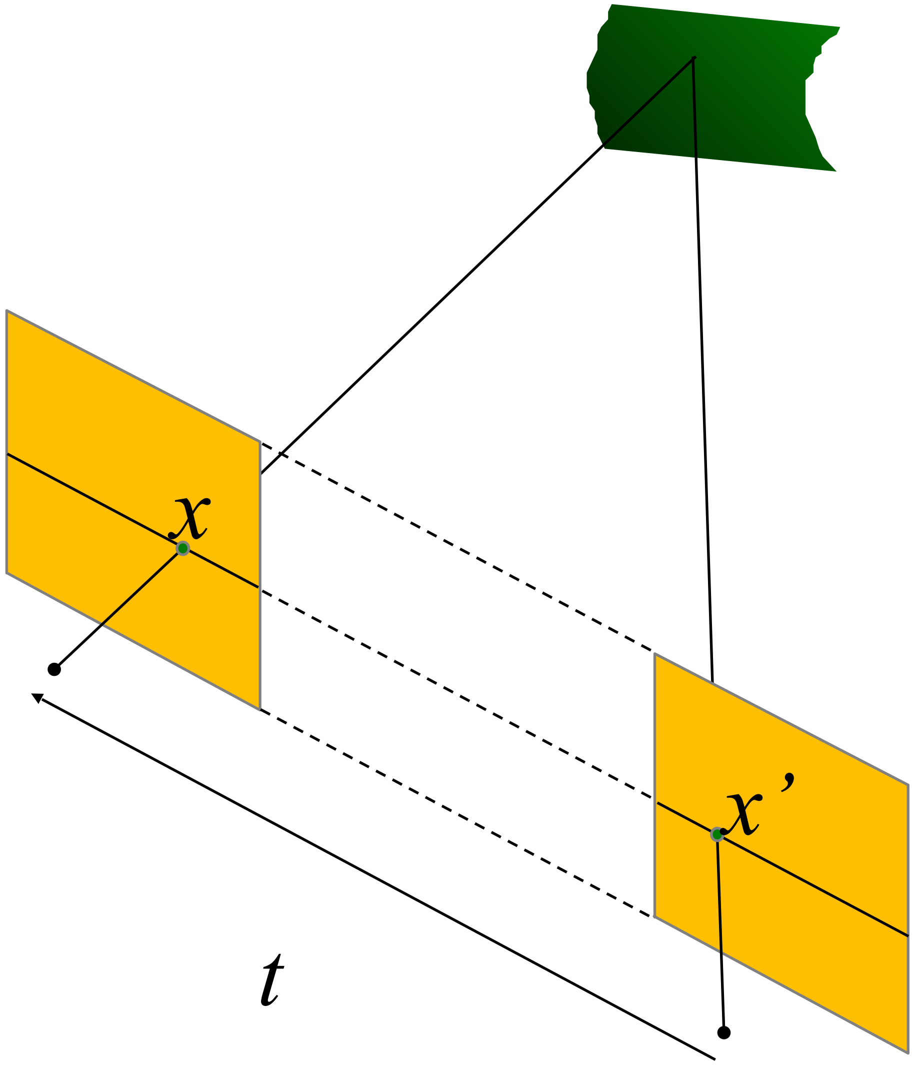
Let's try this out…

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

This always has to hold

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

How can you make the epipolar lines horizontal?



When this relationship holds

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

Let's try this out…

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

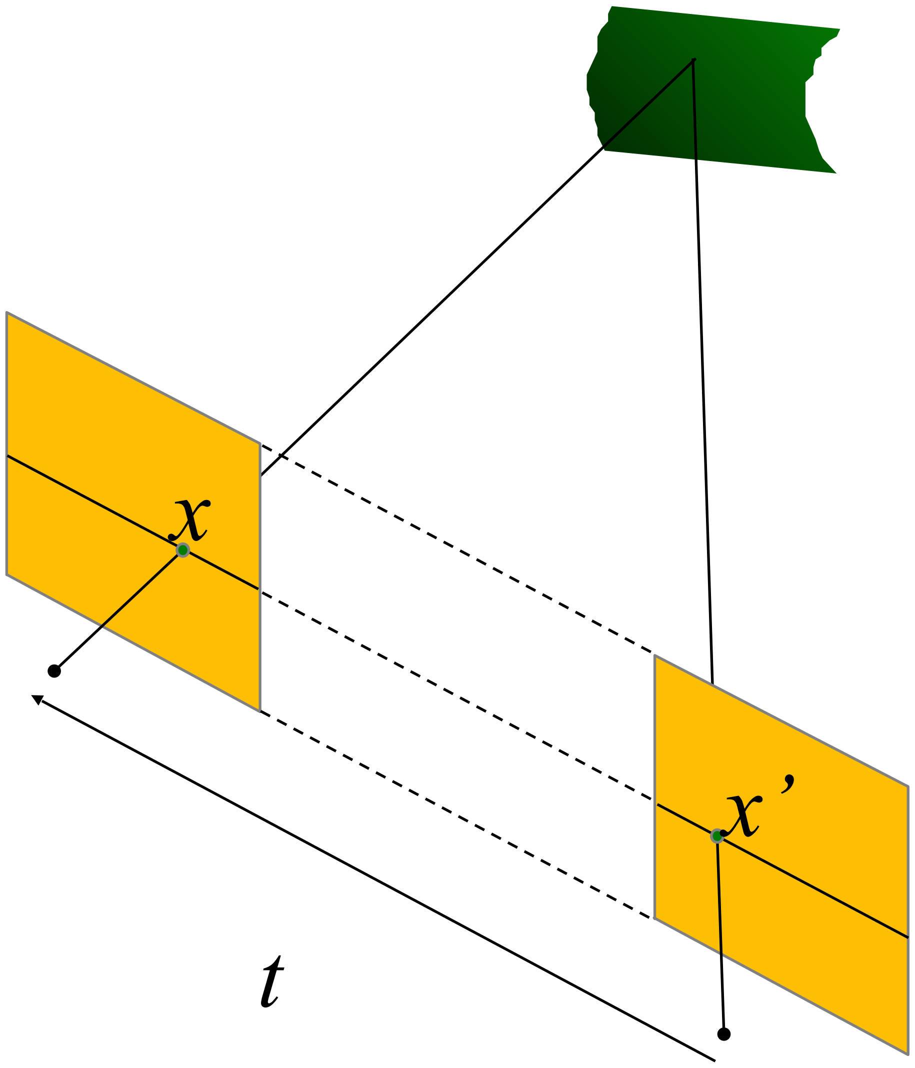
This always has to hold

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

Write out the constraint

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

How can you make the epipolar lines horizontal?



When this relationship holds

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

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

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

Write out the constraint

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

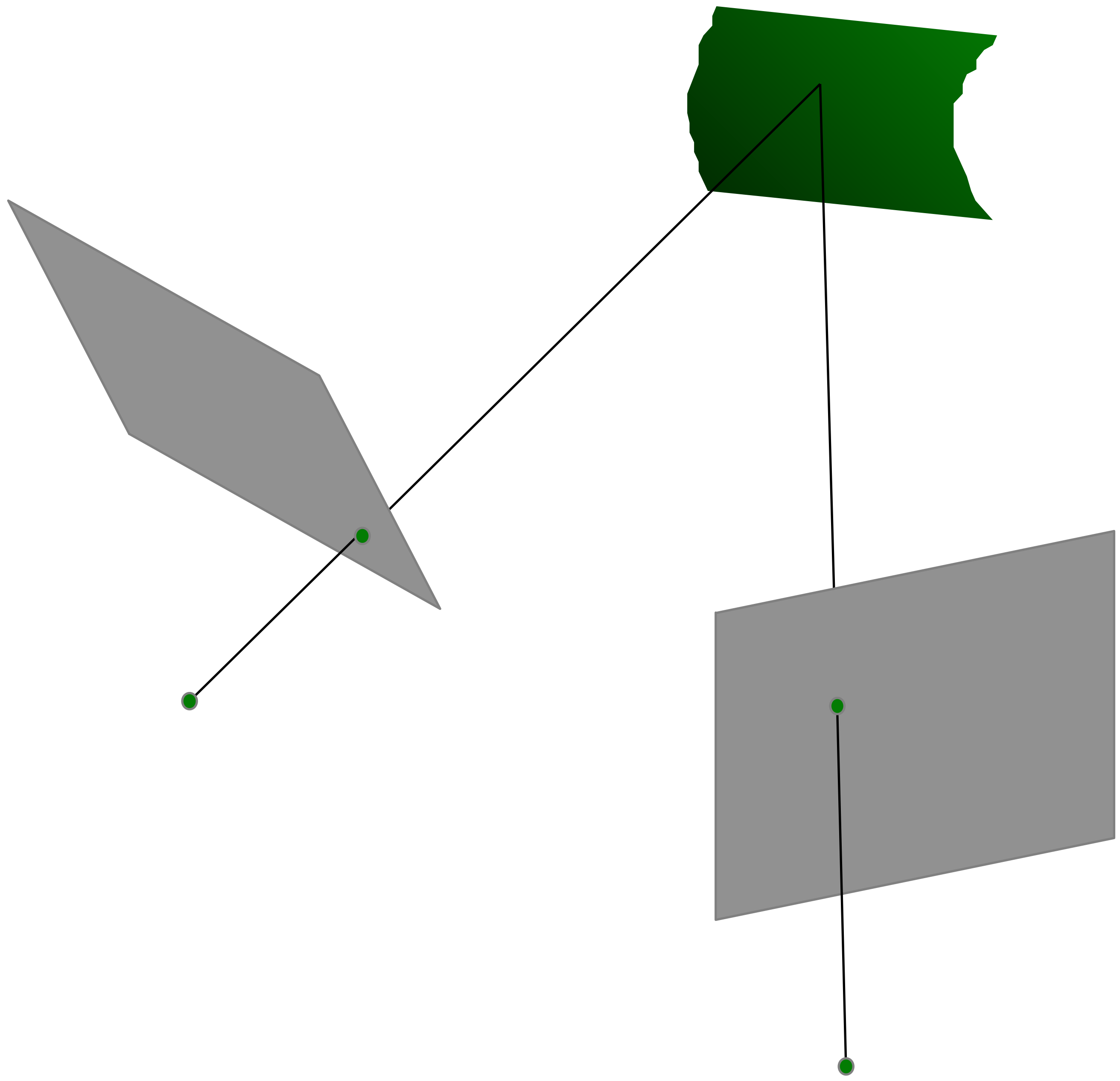
$$\begin{pmatrix} u & v & 1 \end{pmatrix} \begin{pmatrix} 0 \\ -T \\ Tv' \end{pmatrix} = 0$$

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

$$\boxed{Tv = Tv'}$$

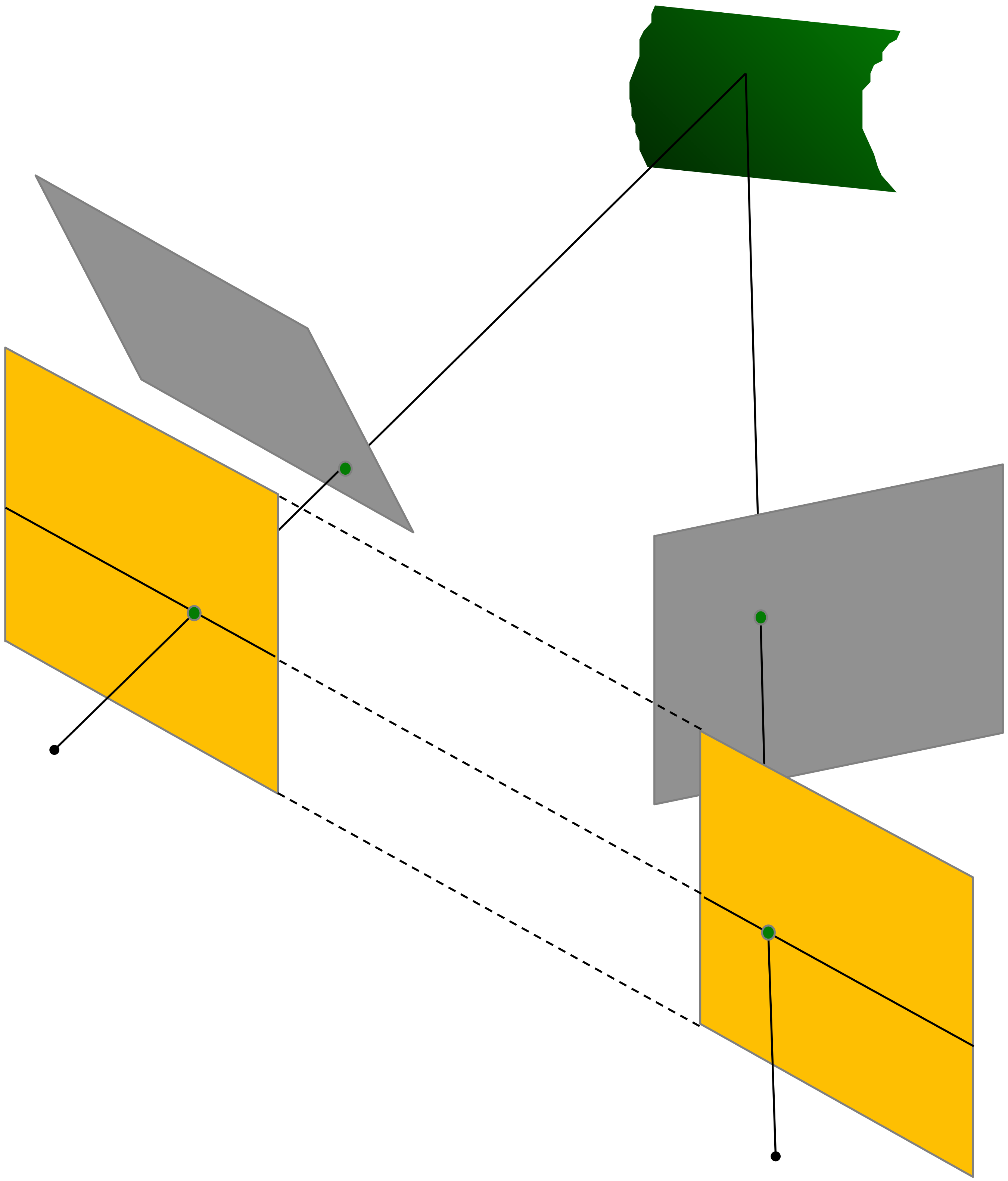
y coordinate is always the same!

What is stereo rectification?



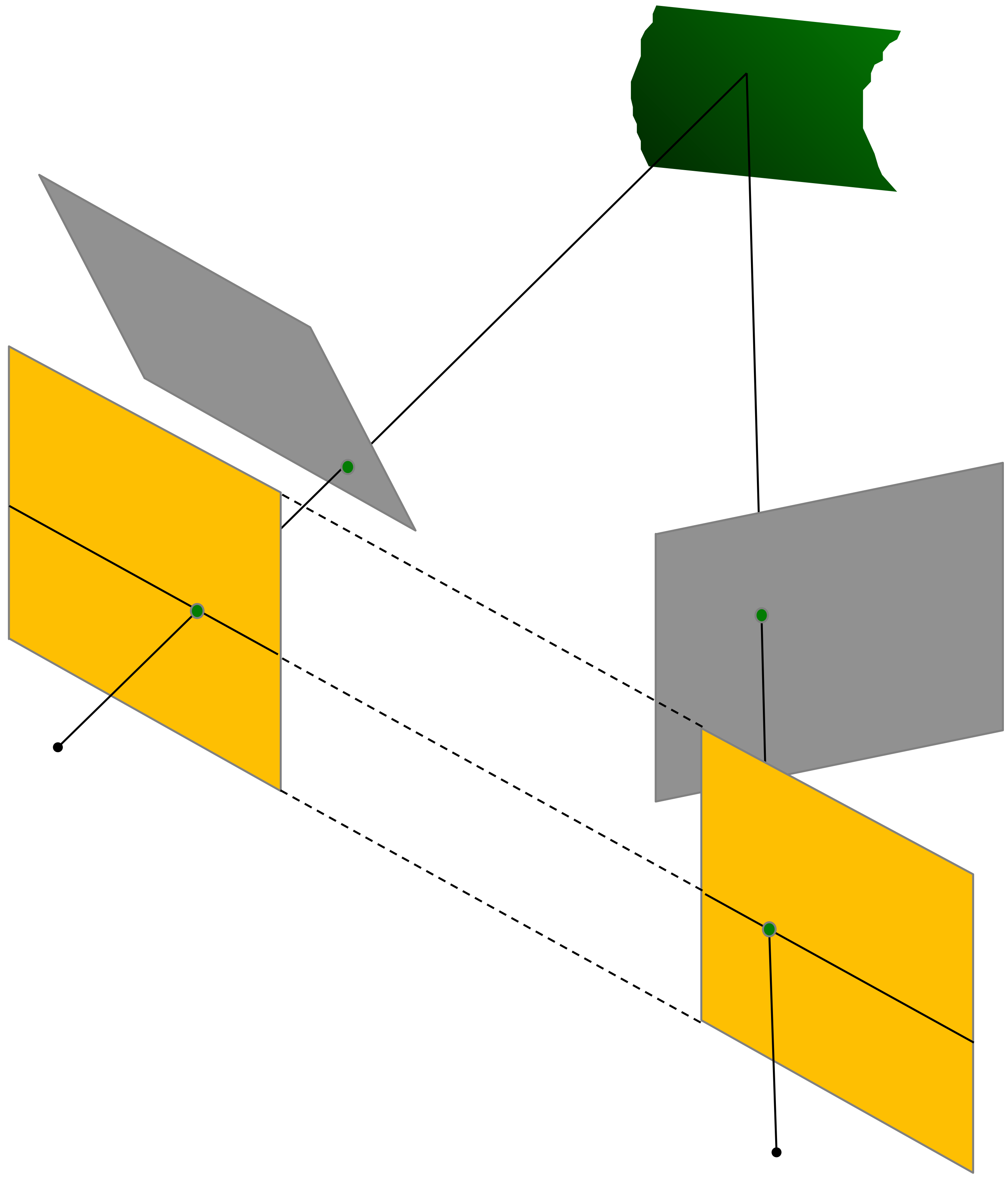
What is stereo rectification?

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



What is stereo rectification?

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



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 left camera so that the epipole is at infinity

2. Apply the same rotation to the right camera

3. Rotate the right camera by $\mathbf{R}^{-1}$

4. Adjust the scale

Setting the epipole to infinity

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

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

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

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

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

Where is this point located on the image plane?

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 $\mathbf{R}_{\text{rect}}$ 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 **E** using the 8 point algorithm

2. Estimate the epipole **e** (solve **Ee**=0)

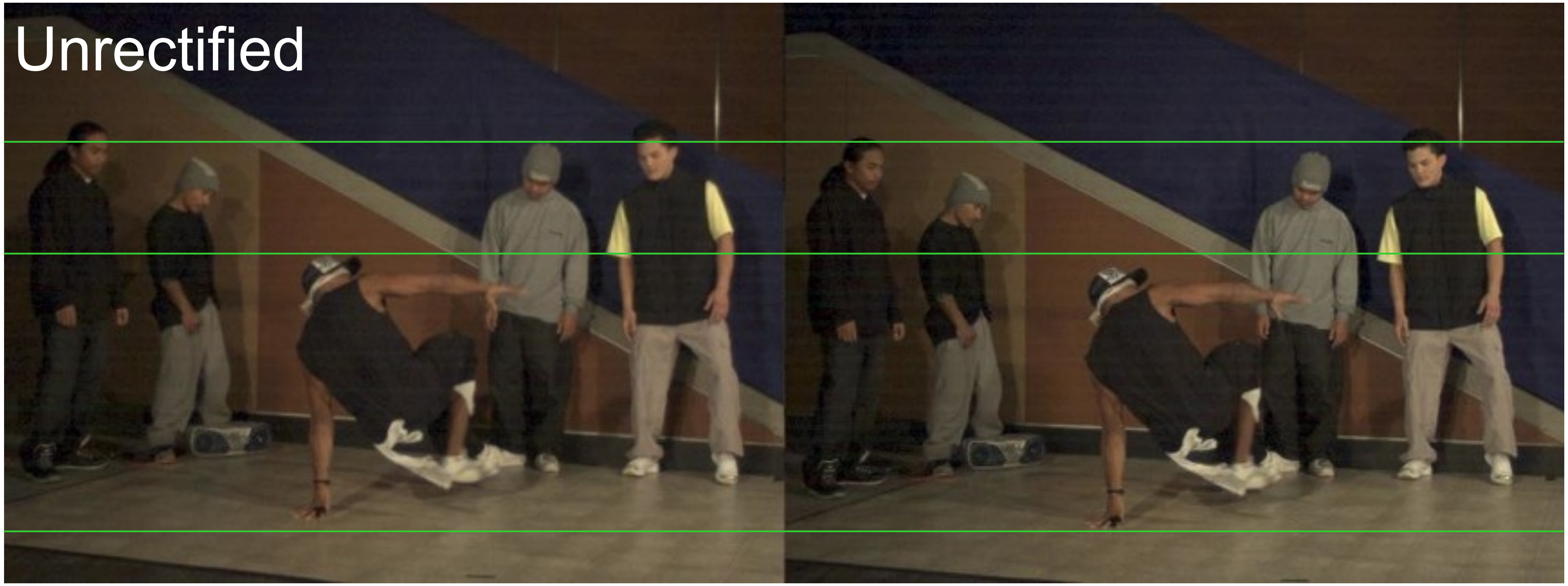
3. Build $\mathbf{R}_{\text{rect}}$ from **e**

4. Decompose **E** into **R** and **T**

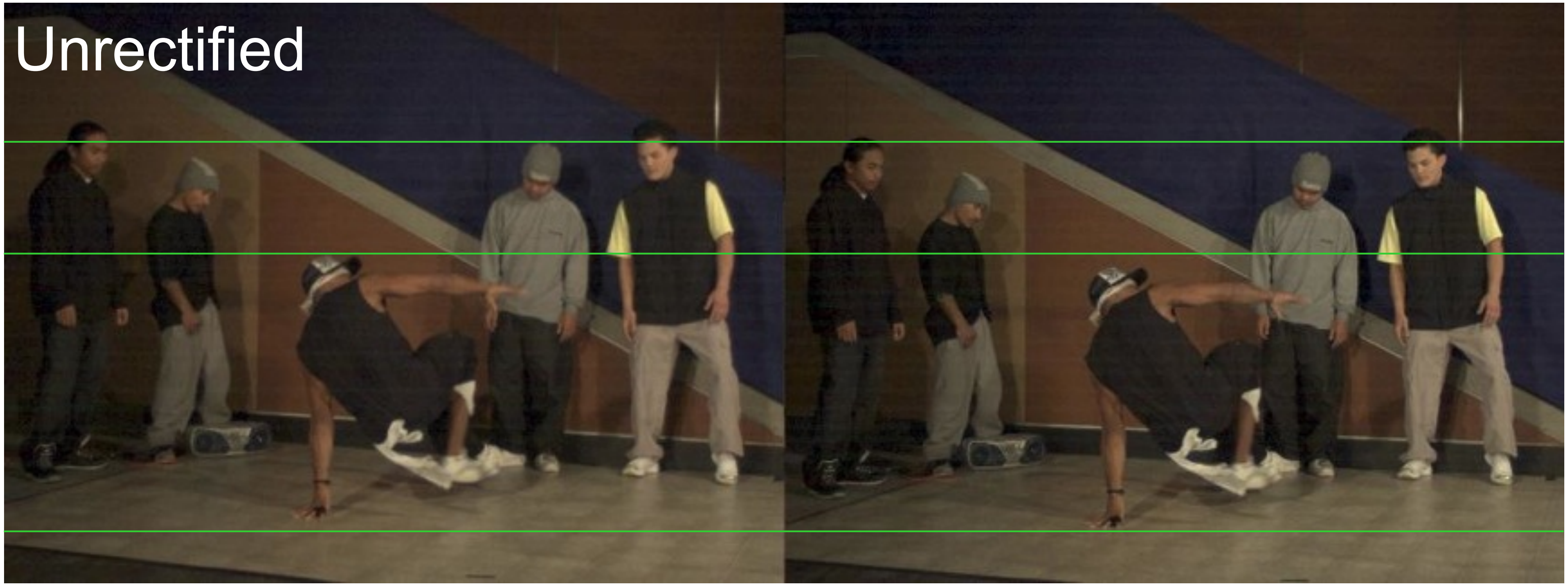
5. Set $\mathbf{R}_1=\mathbf{R}_{\text{rect}}$ and $\mathbf{R}_2 = \mathbf{R}\mathbf{R}_{\text{rect}}$

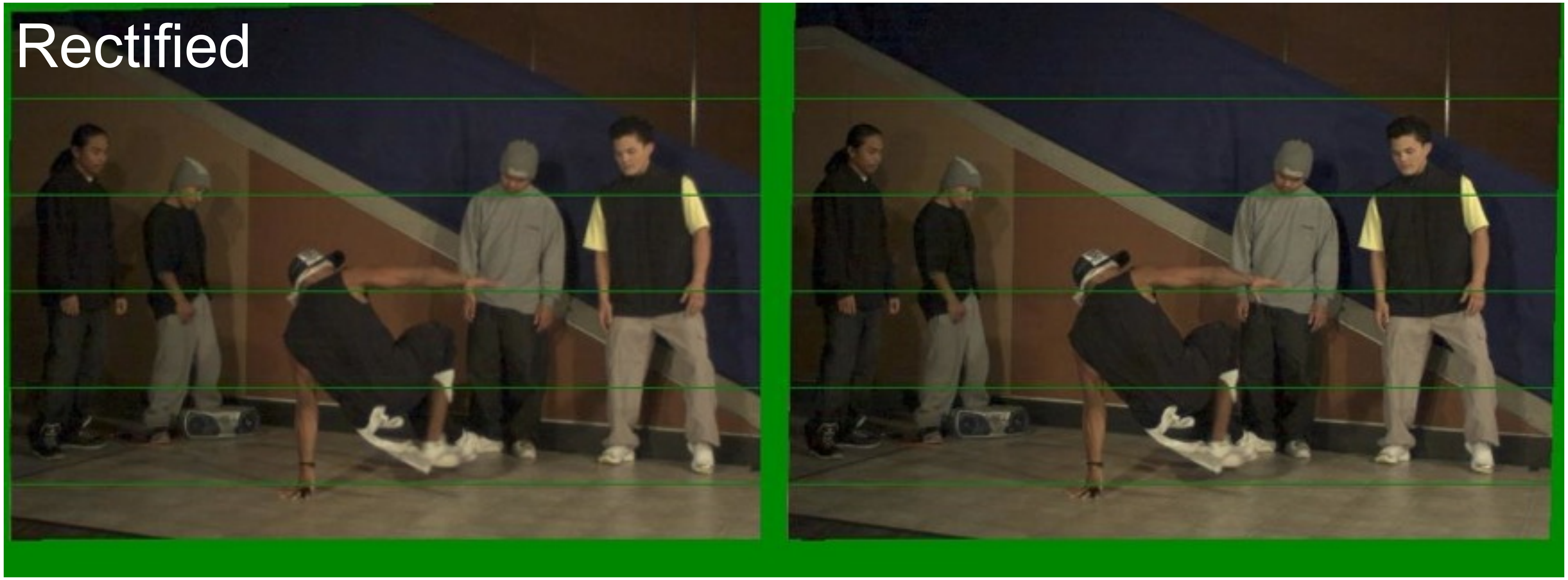
6. Rotate each left camera point **x'**~ **Hx** where $\mathbf{H} = \mathbf{K}\mathbf{R}_1$
   *You may need to alter the focal length (inside **K**) to keep points within the original image size

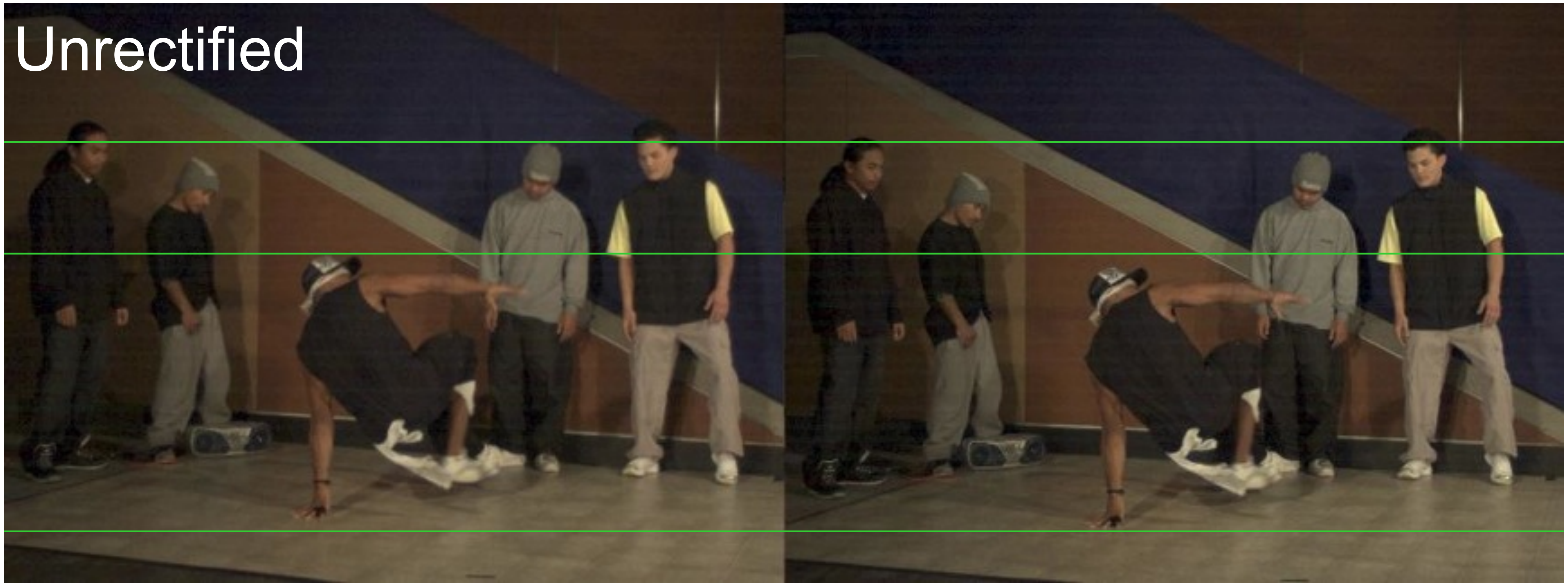
7. Repeat 6 and 7 for right camera points using $\mathbf{R}_2$

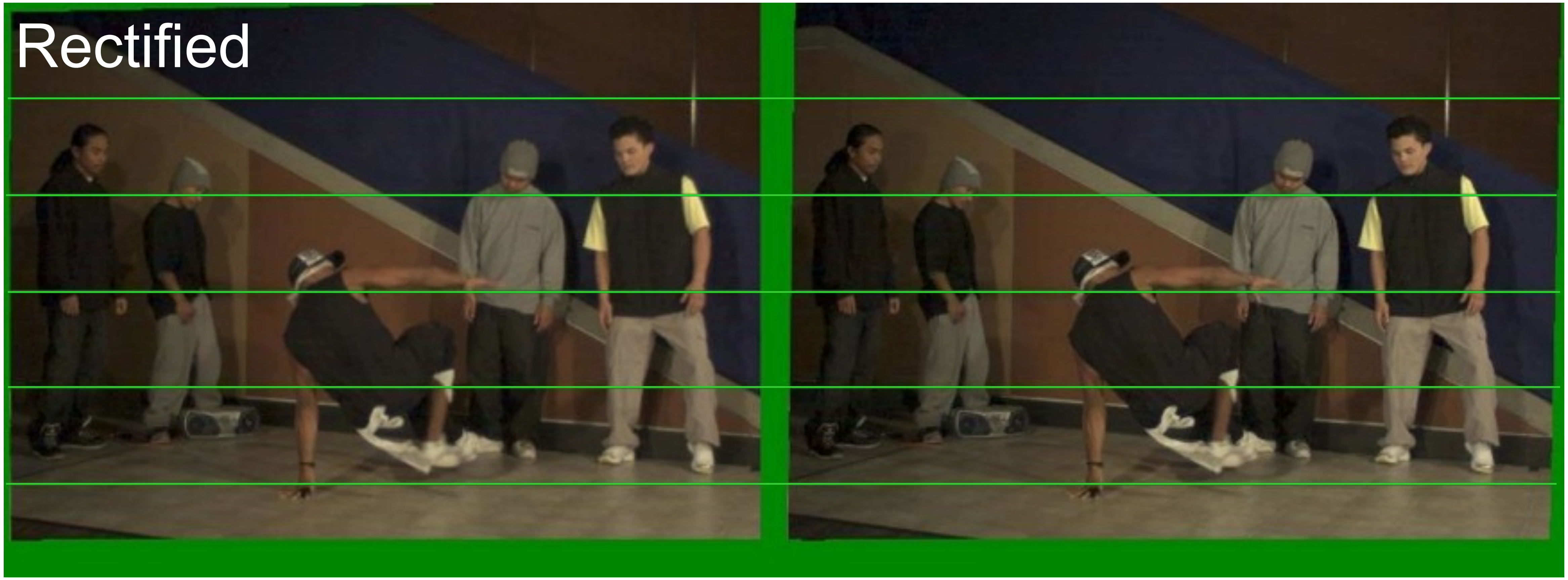


Unrectified

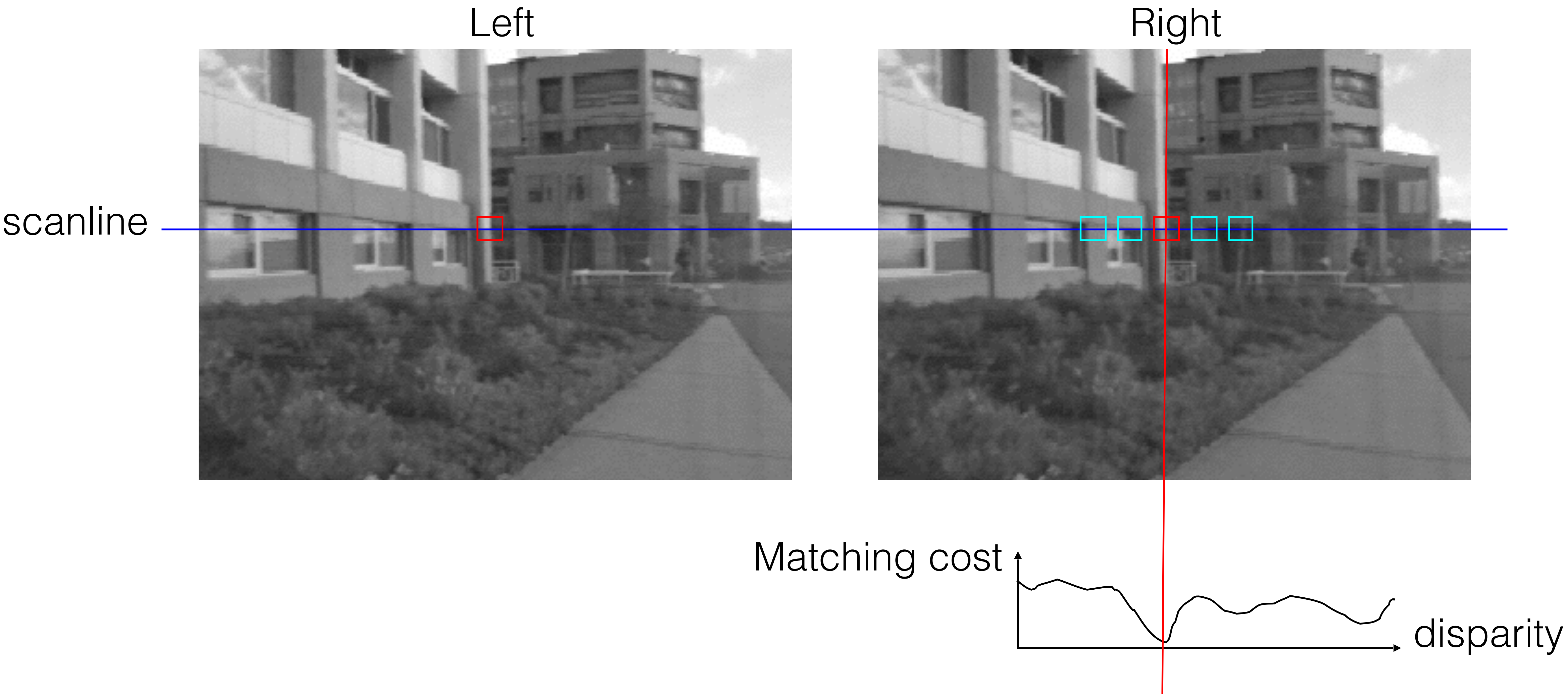

Unrectified


Rectified

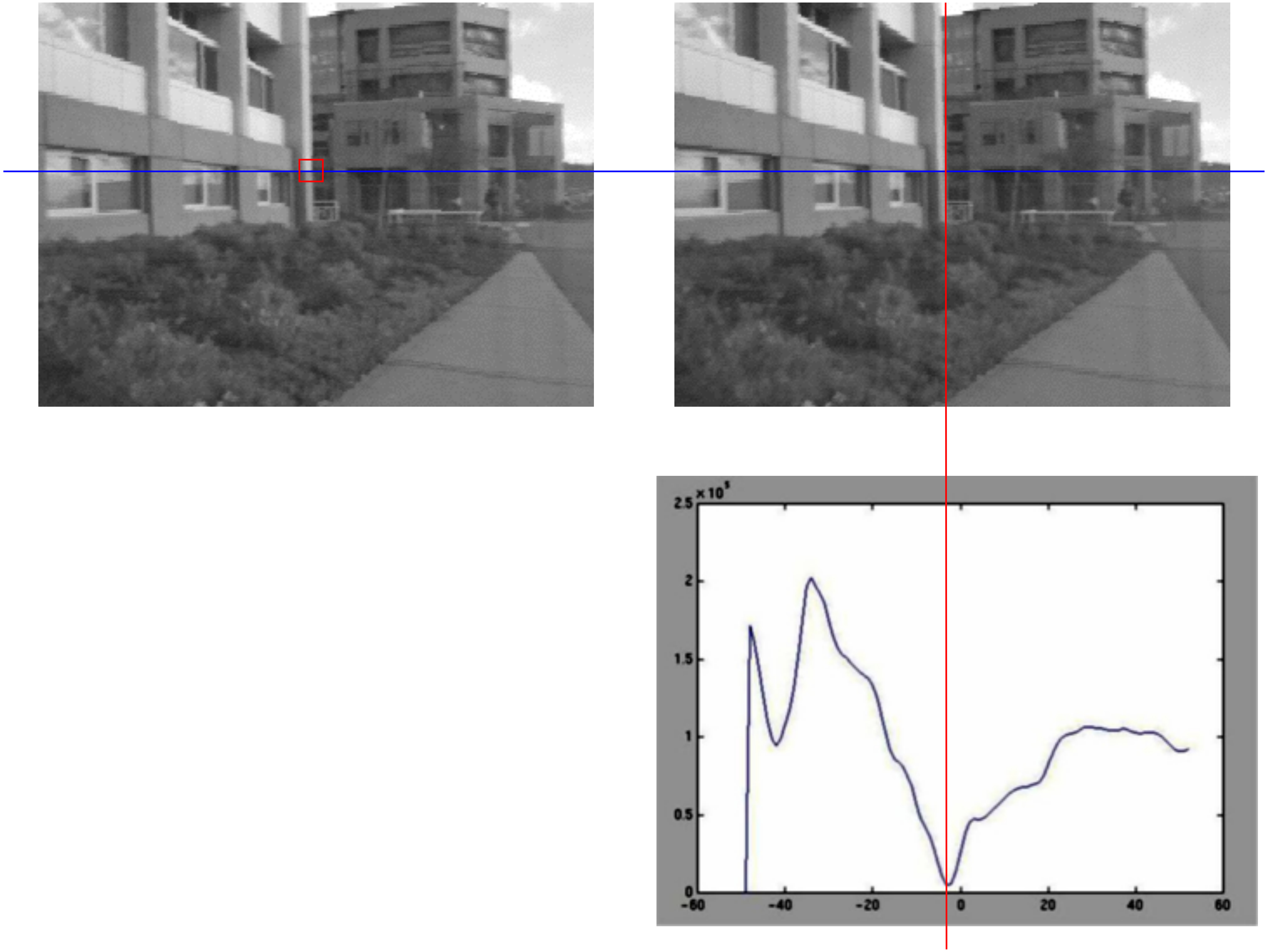

Unrectified


Rectified

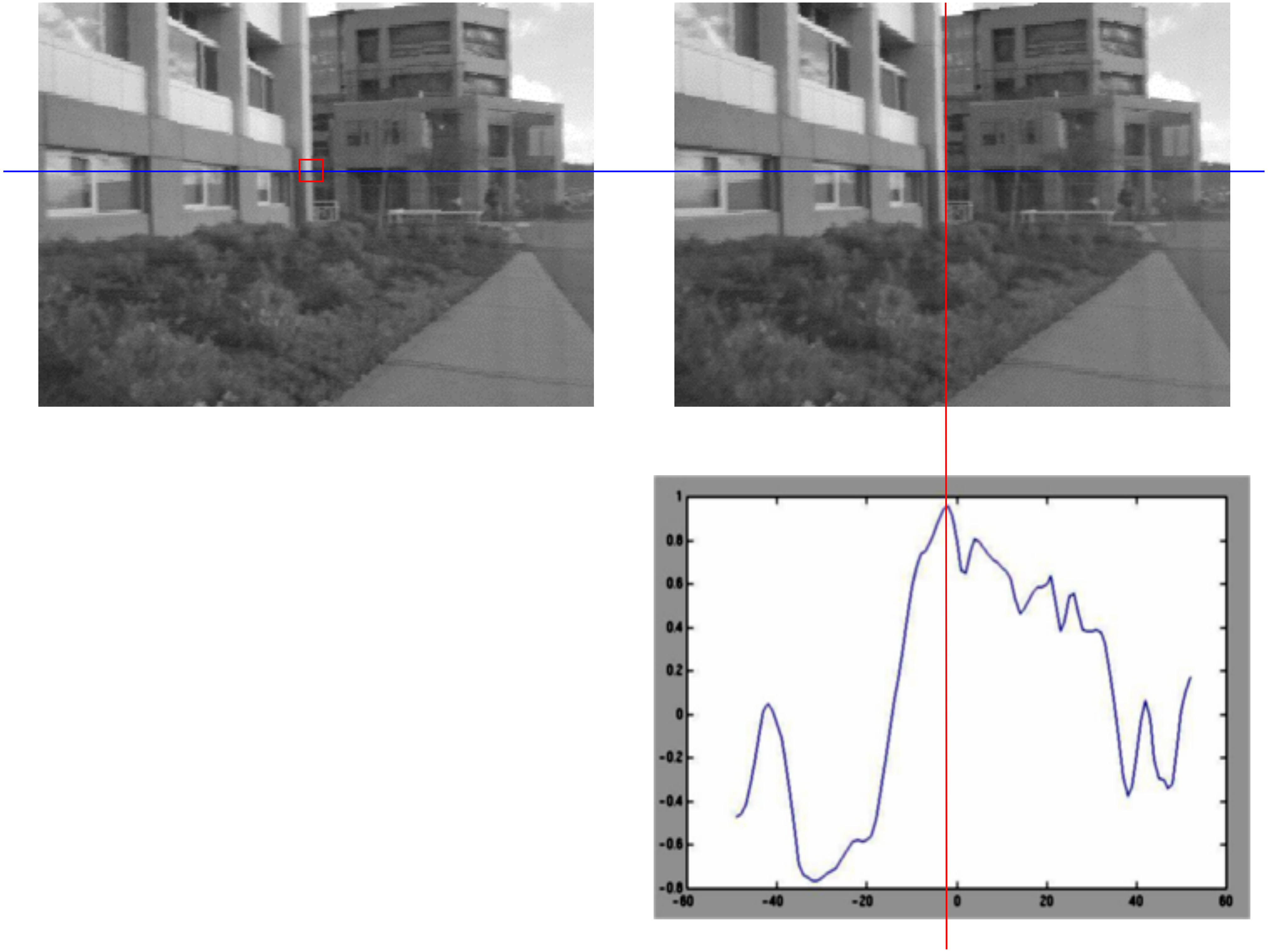
Finding the best match



- 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

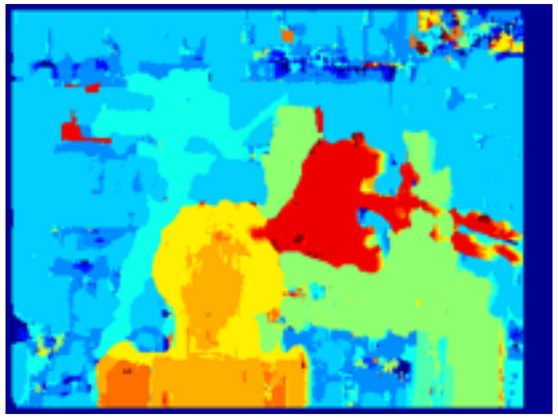


SSD

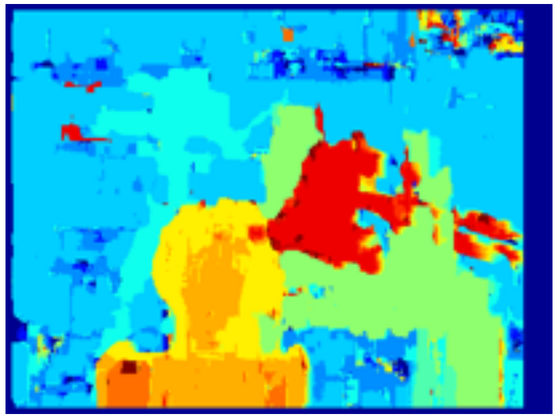


Normalized cross-correlation

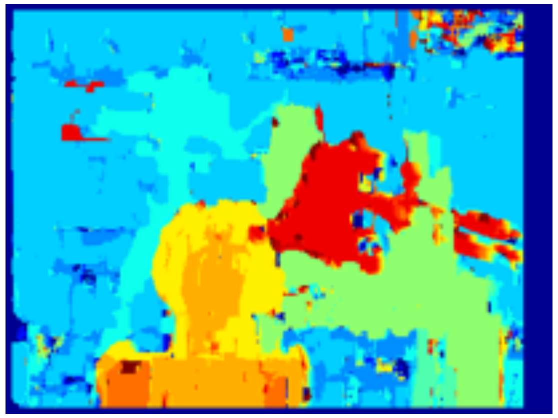
Similarity Measure	Formula
Sum of Absolute Differences (SAD)	$\sum_{(i,j)\in W} \lvert I_1(i,j) - I_2(x+i,y+j)\rvert$
Sum of Squared Differences (SSD)	$\sum_{(i,j)\in W} \left(I_1(i,j) - I_2(x+i,y+j)\right)^2$
Zero-mean SAD	$\sum_{(i,j)\in W} \lvert I_1(i,j) - \bar{I}_1(i,j) - I_2(x+i,y+j) + \bar{I}_2(x+i,y+j)\rvert$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



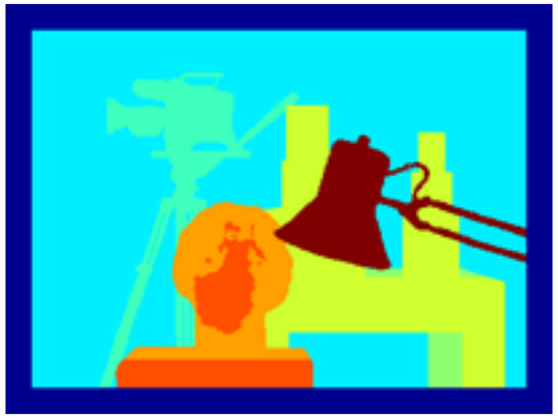
SAD



SSD

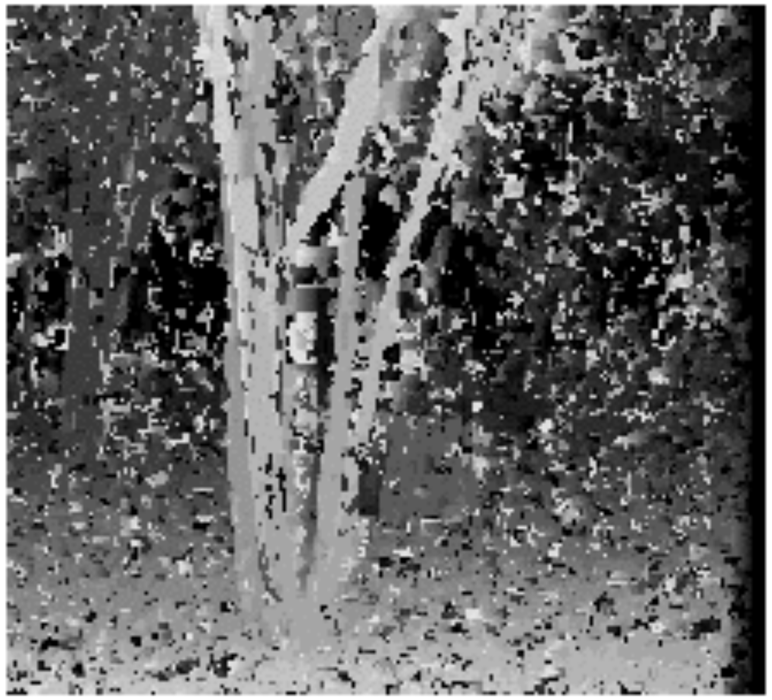


NCC

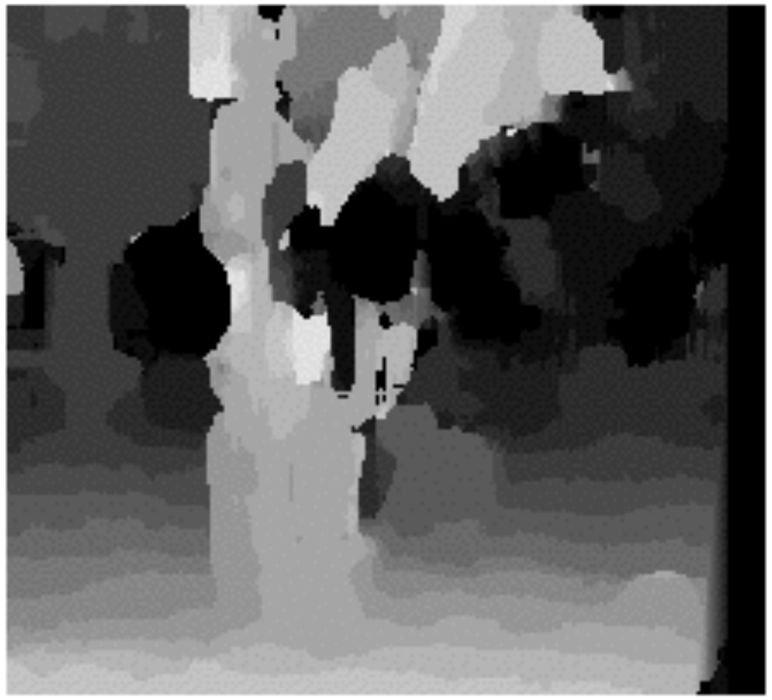


Ground truth

Effect of window size

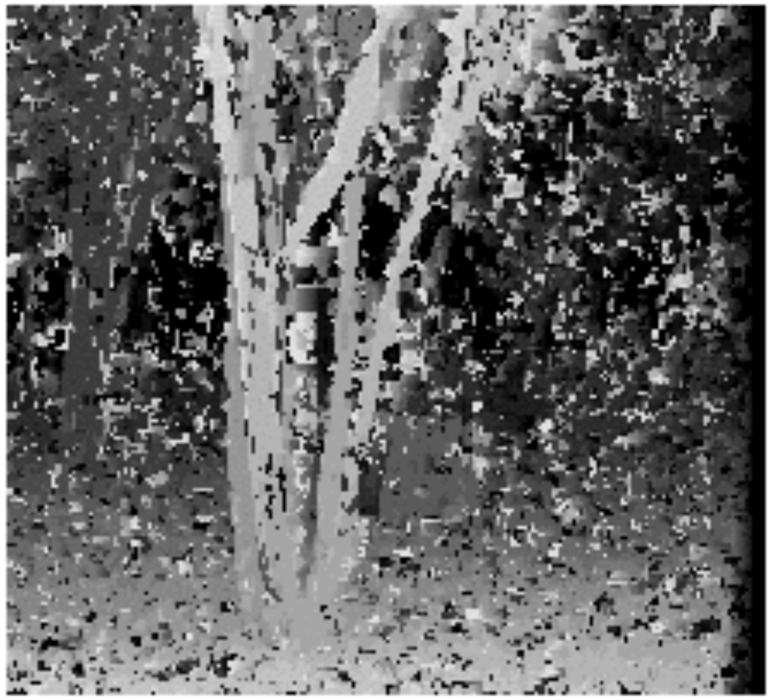


W = 3

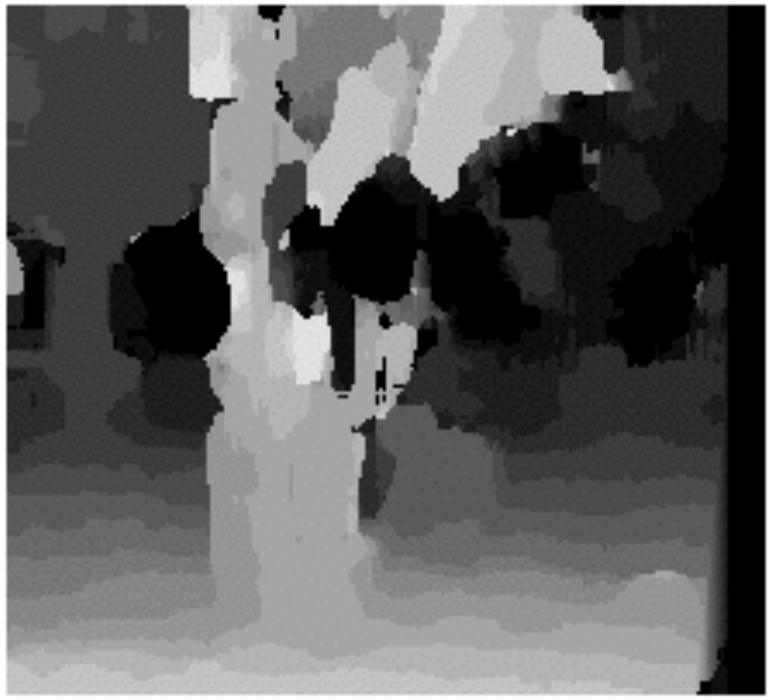


W = 20

Effect of window size



W = 3



W = 20

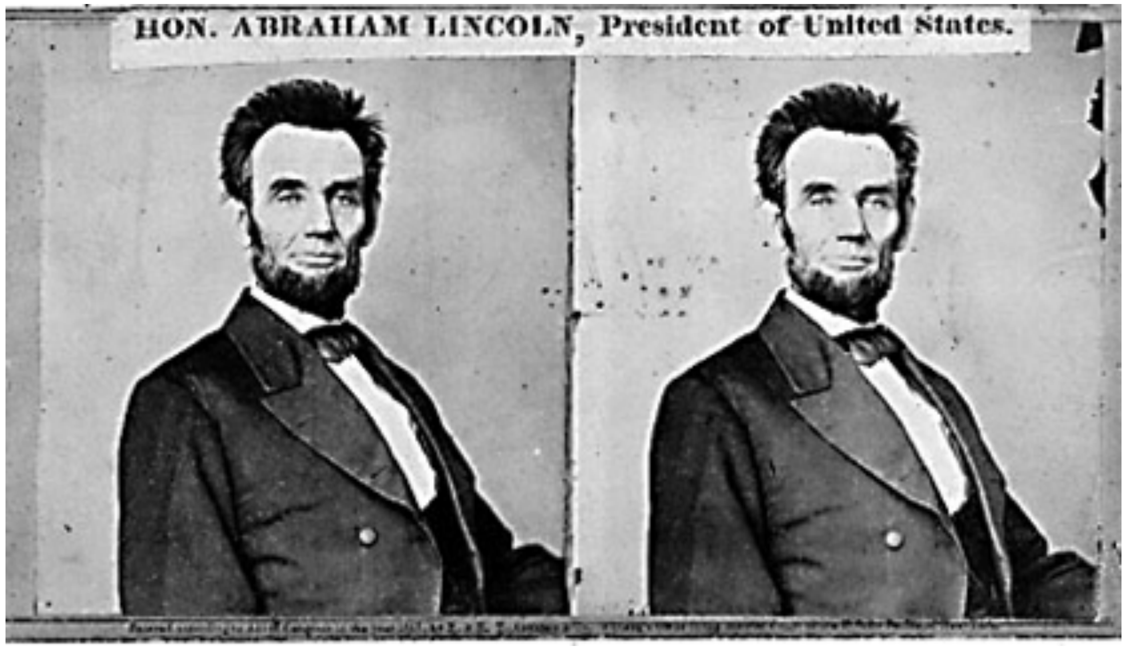
Smaller window

- + More detail
- - More noise

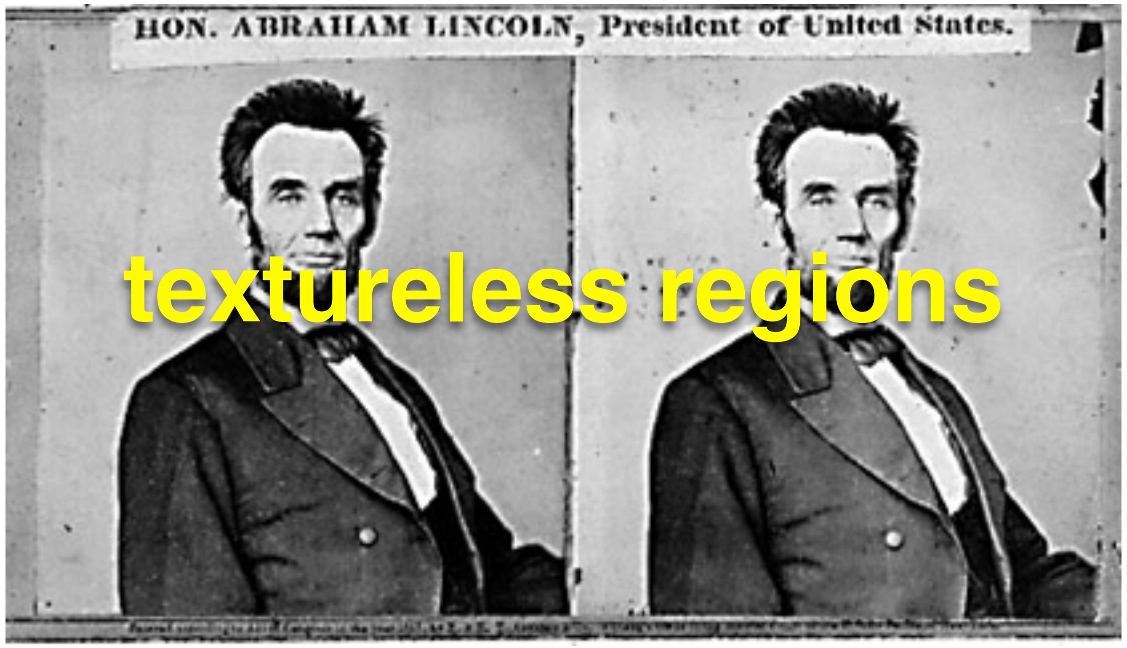
Larger window

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

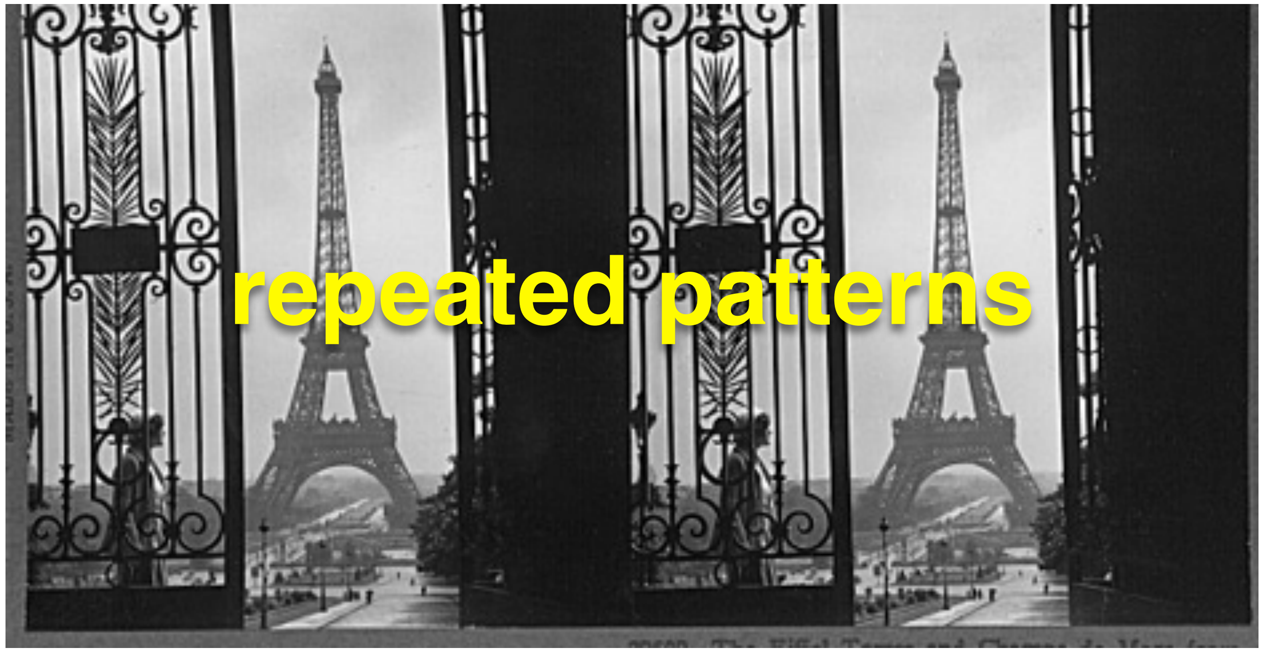
When will stereo block matching fail?



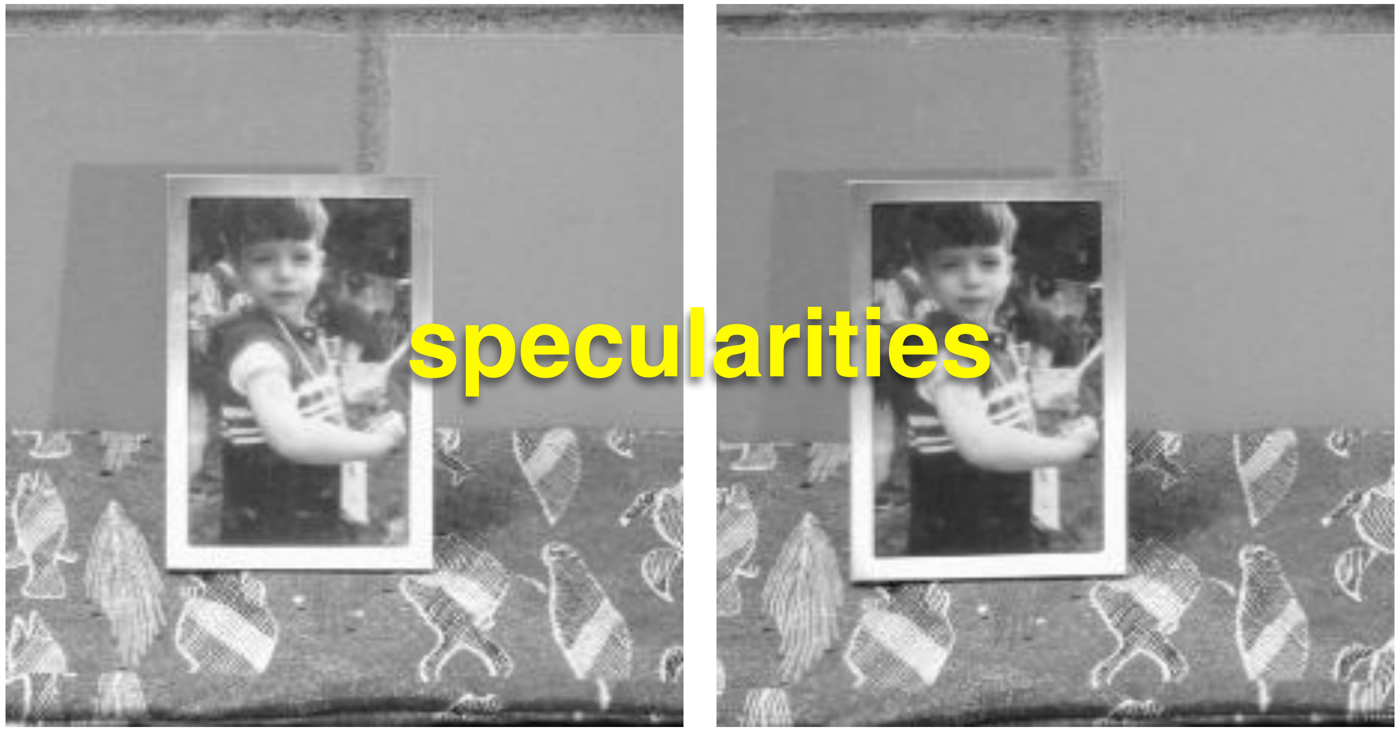
When will stereo block matching fail?



textureless regions



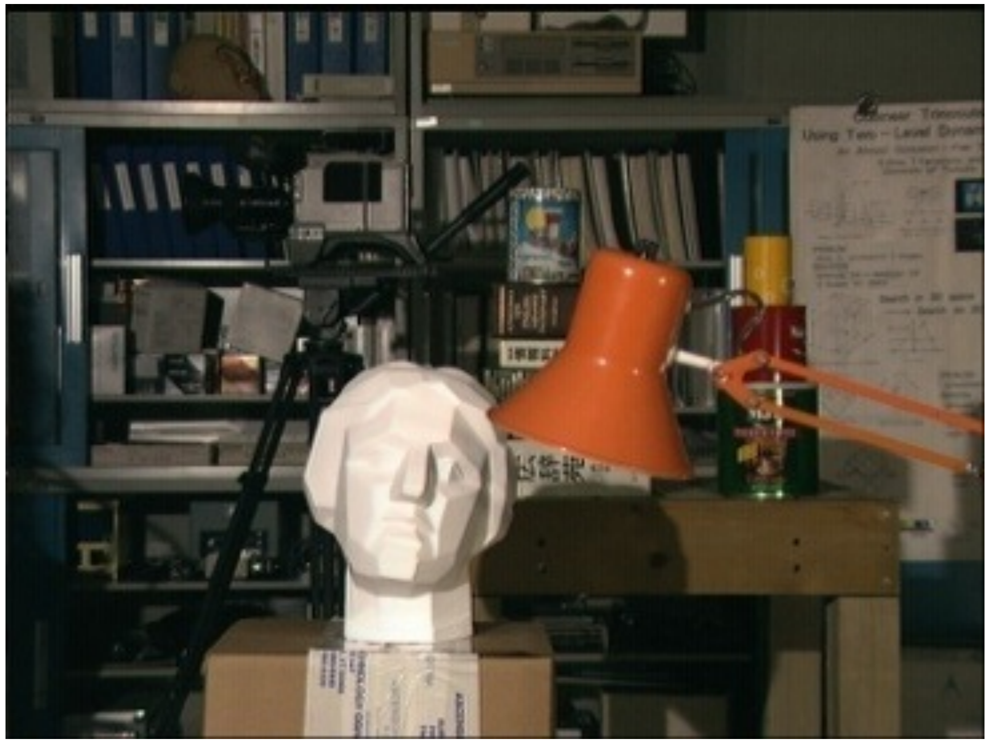
repeated patterns



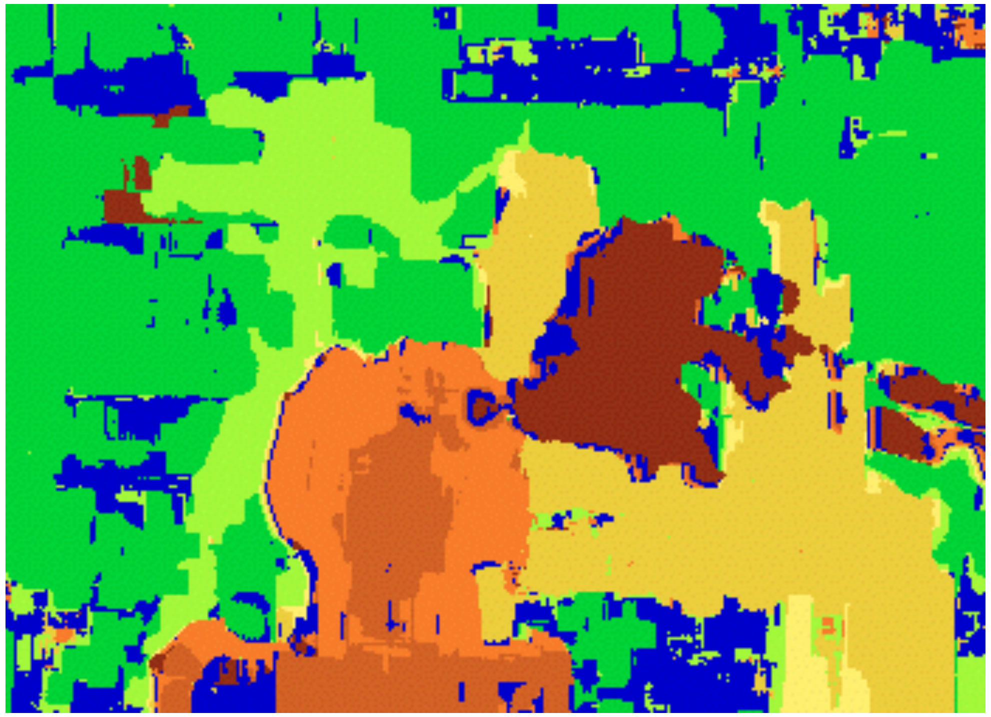
specularities

(break)

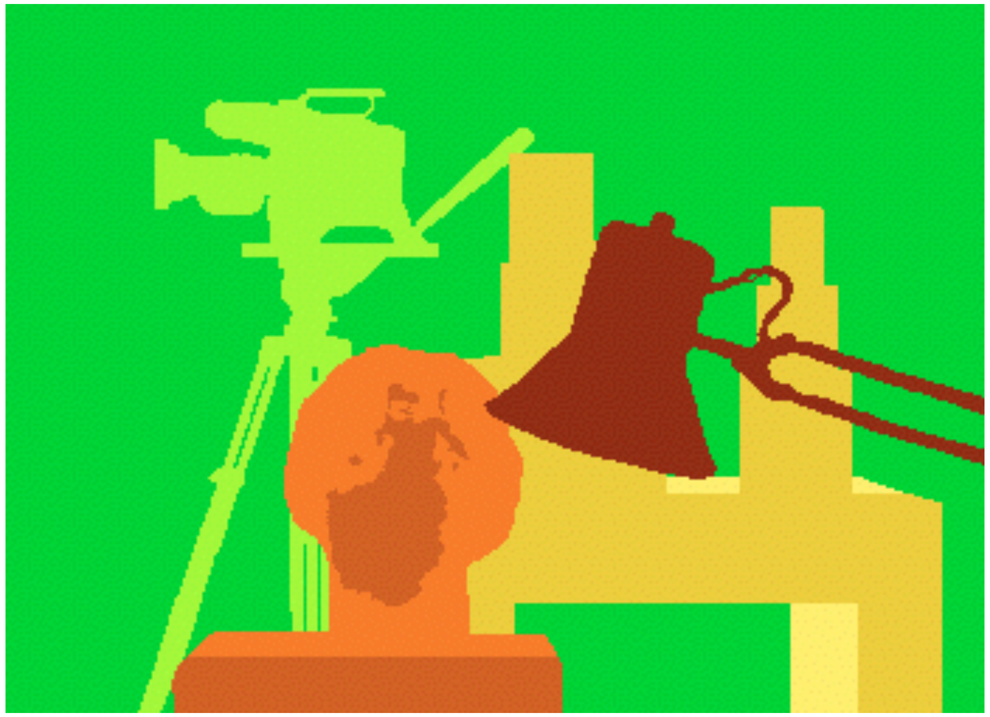
Improving Stereo Block Matching



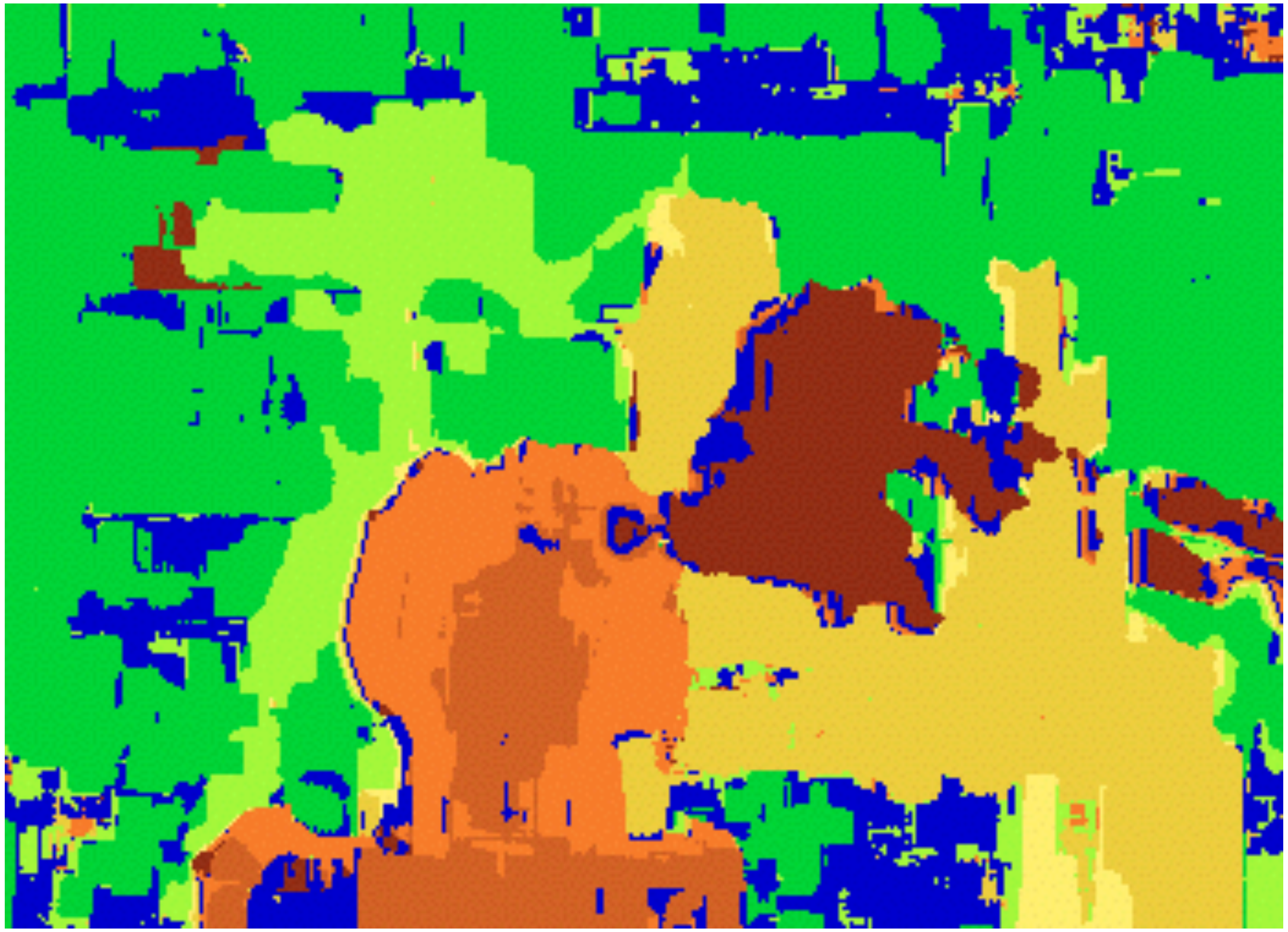
Block matching



Ground truth



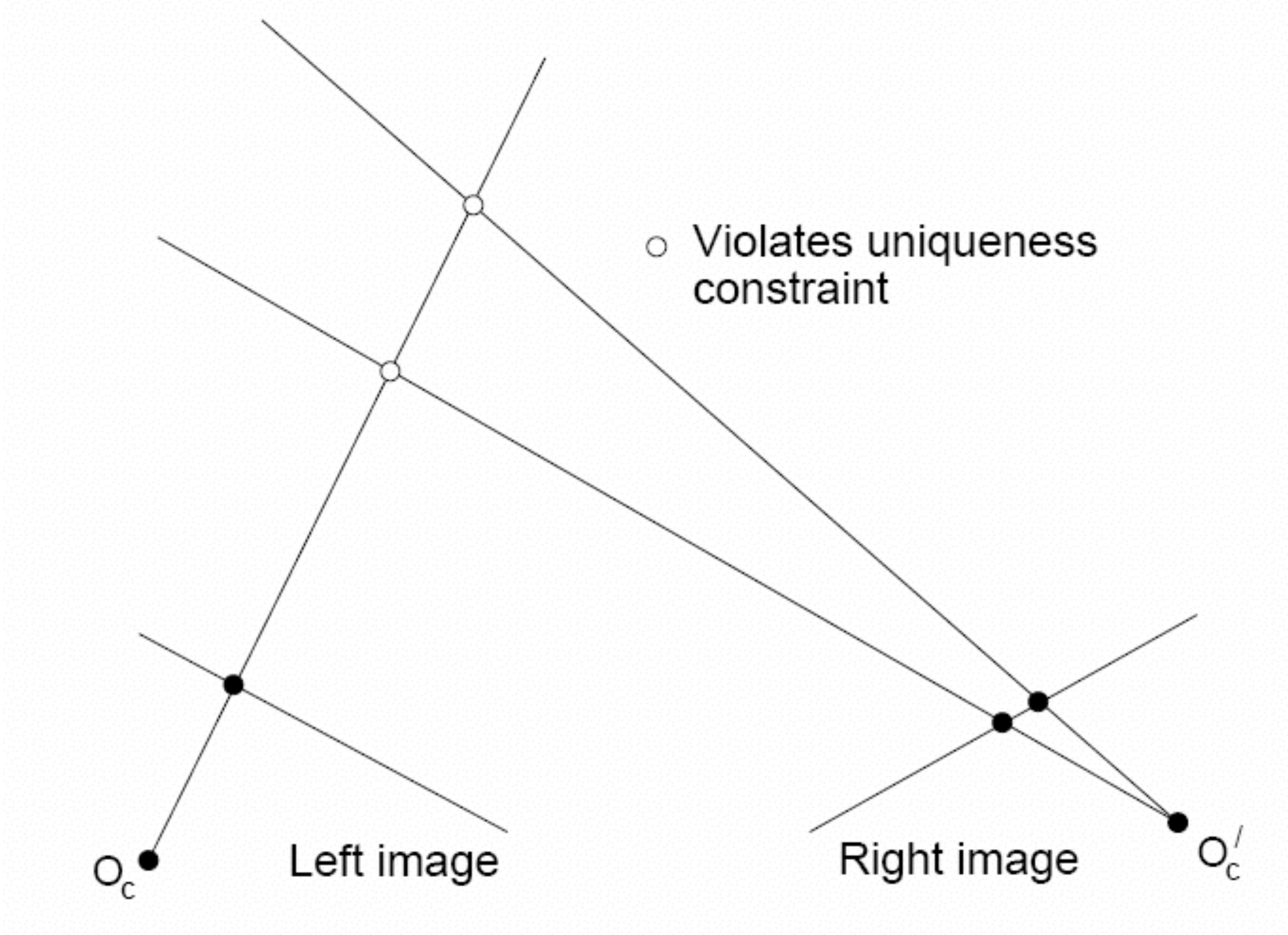
What are some problems with the result?



How can we improve depth estimation?

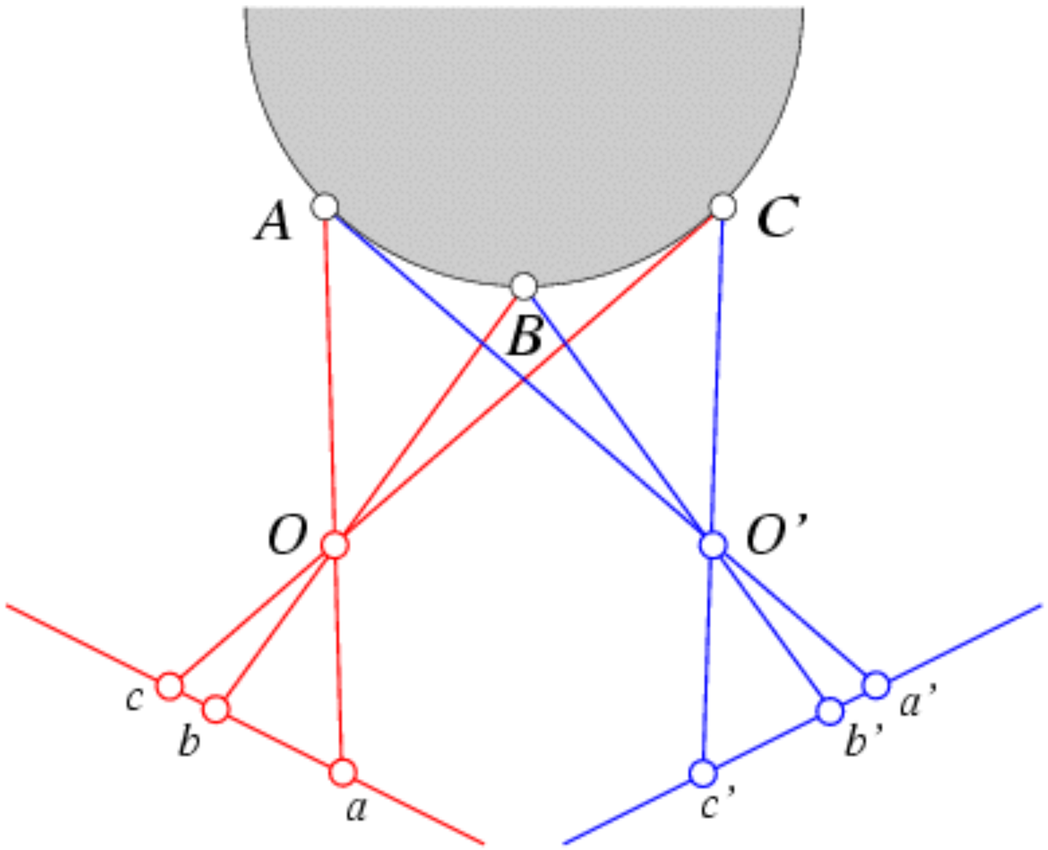
Uniqueness

For any point in one image, there should be at most one matching point in the other image



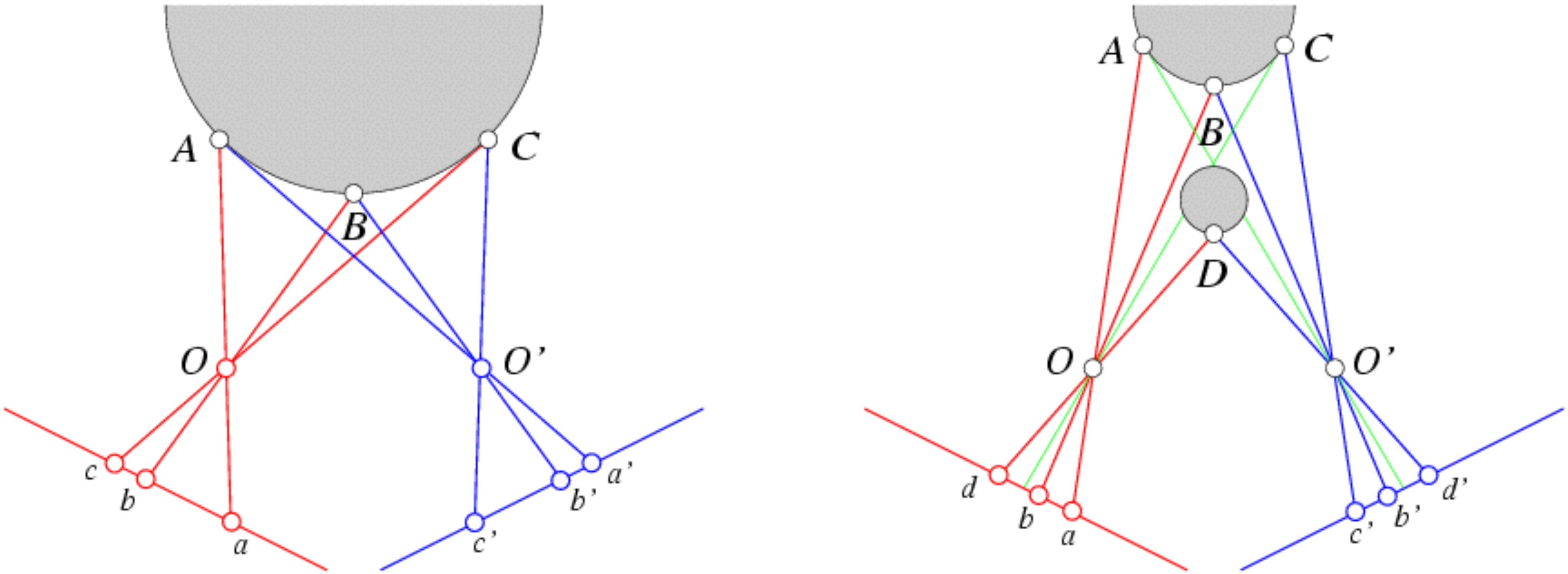
Ordering

Corresponding points should be in the same order in both views



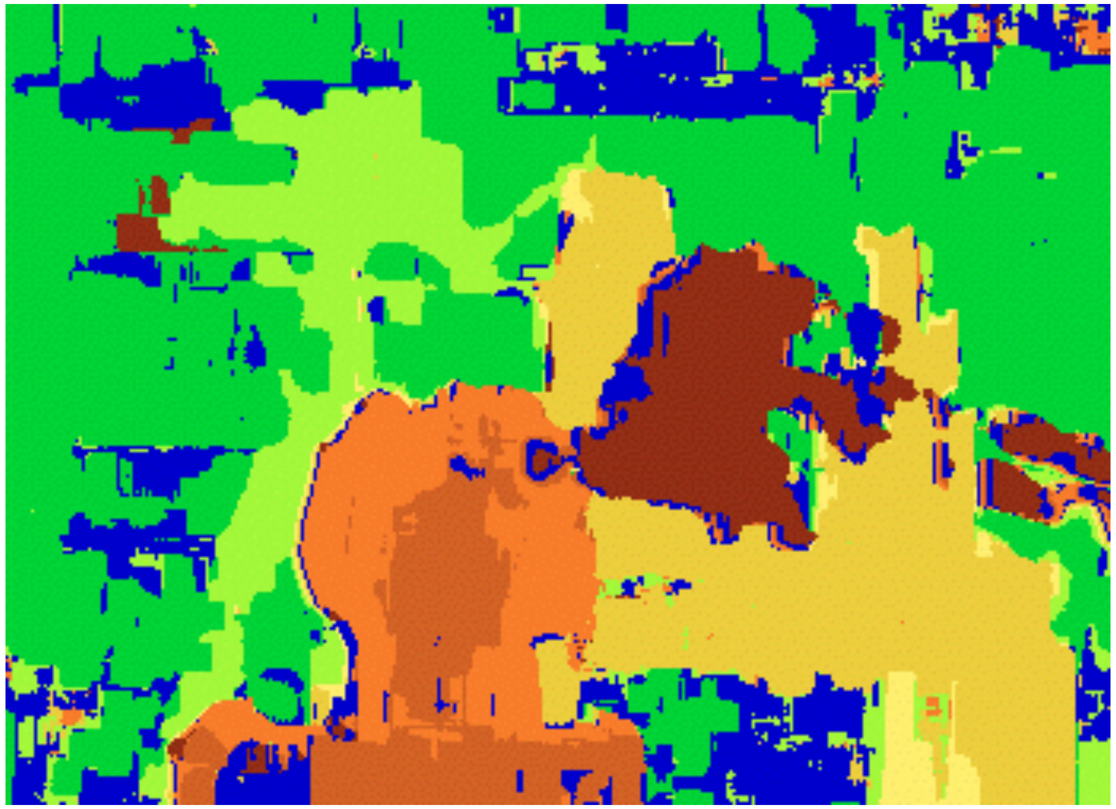
Ordering

Corresponding points should be in the same order in both views

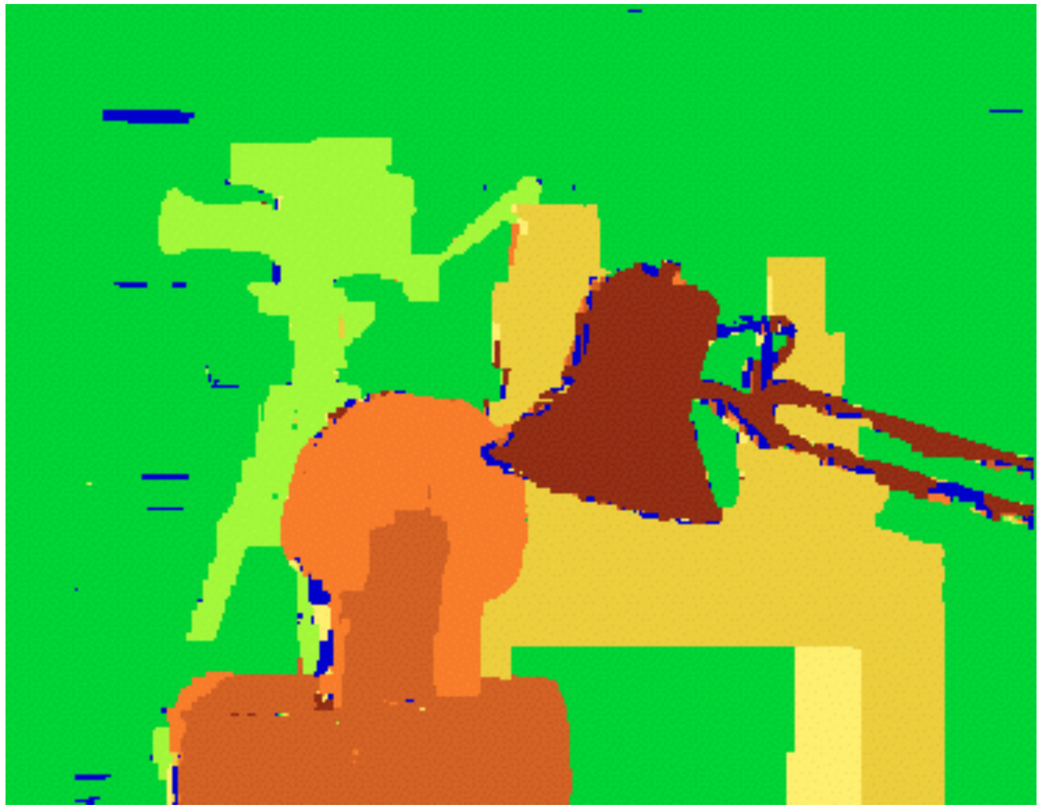


Smoothness

We expect disparity values to change slowly (for the most part)



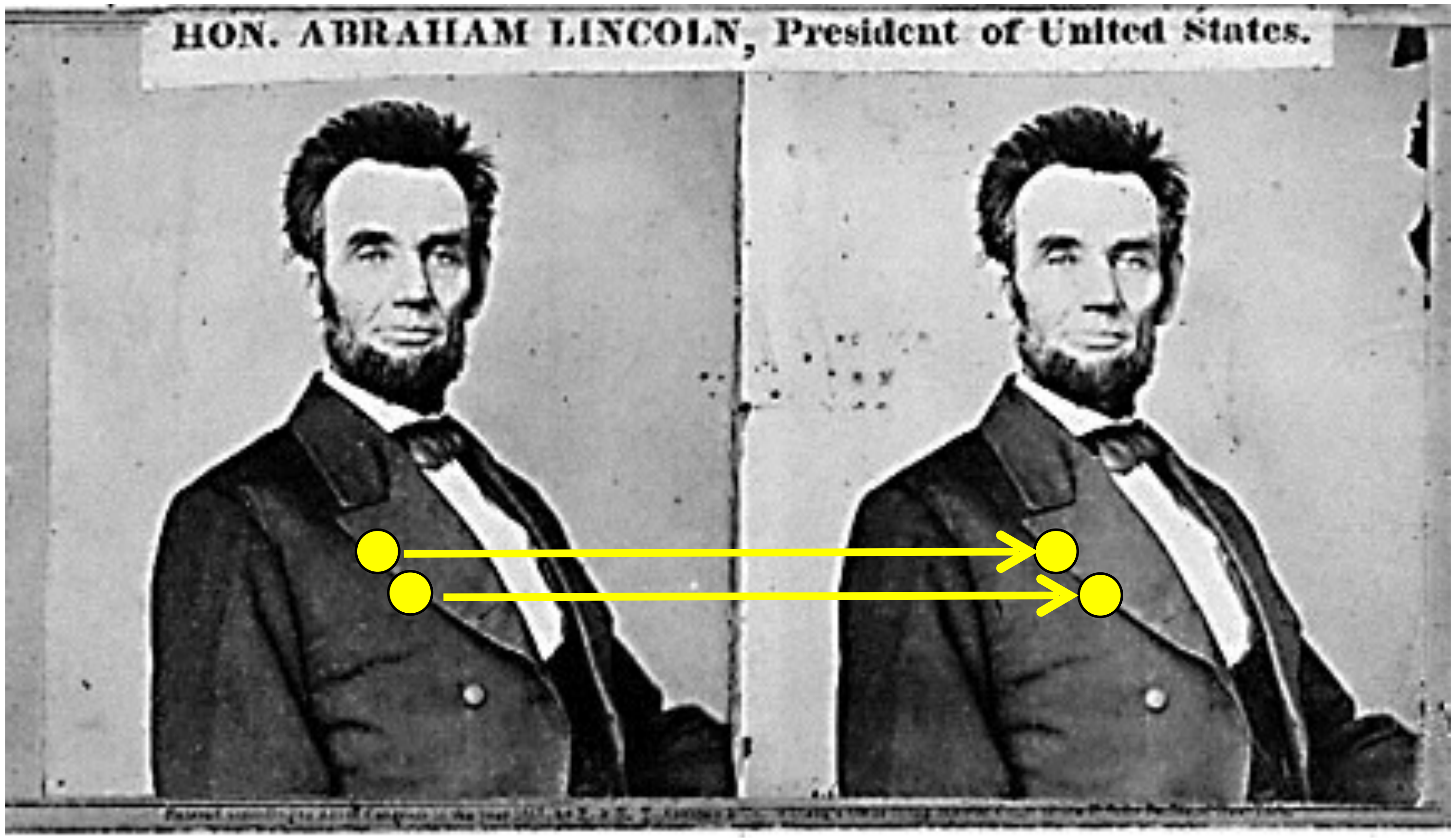
Too many discontinuities



Better

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

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

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

(match cost)

Adjacent pixels should (usually) move about the same amount

(smoothness cost)

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

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

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

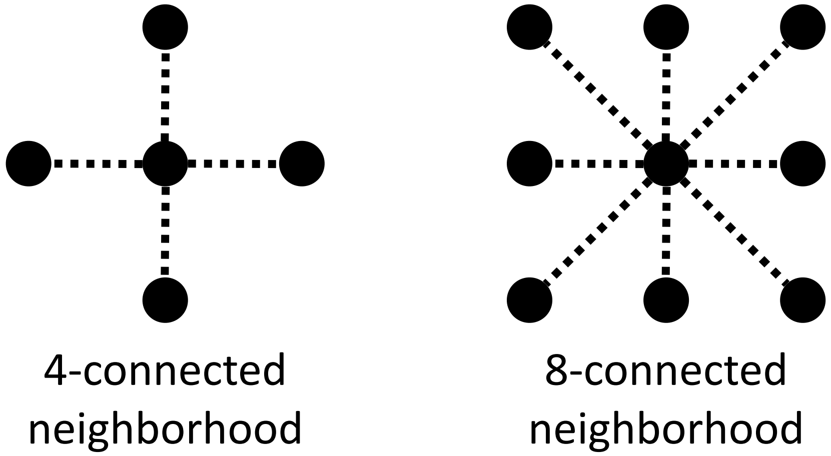
$$E(d) = E_d(d) + \lambda E_s(d)$$

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

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

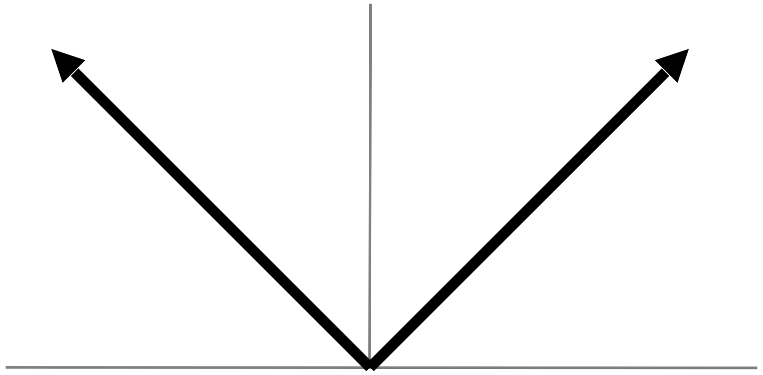
$$E_s(d) = \sum_{(p,q) \in \mathcal{E}} V(d_p, d_q)$$

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



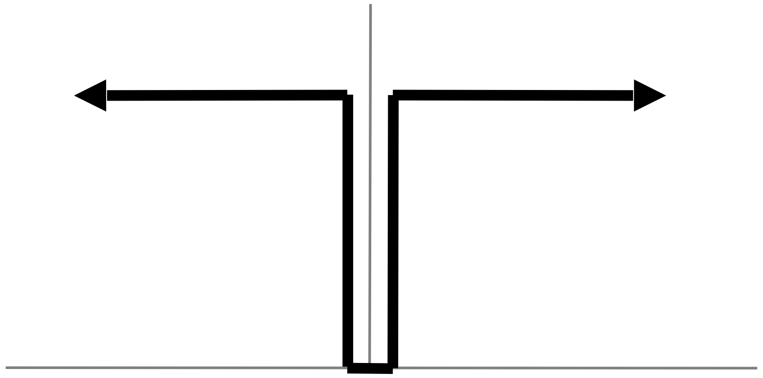
$$E_s(d) = \sum_{(p,q)\in\mathcal{E}} V(d_p, d_q)$$

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



L$_1$ distance

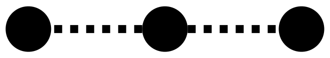
$$V(d_p, d_q) = \begin{cases} 0 & \text{if } d_p = d_q \\ 1 & \text{if } d_p \neq d_q \end{cases}$$



“Potts model”

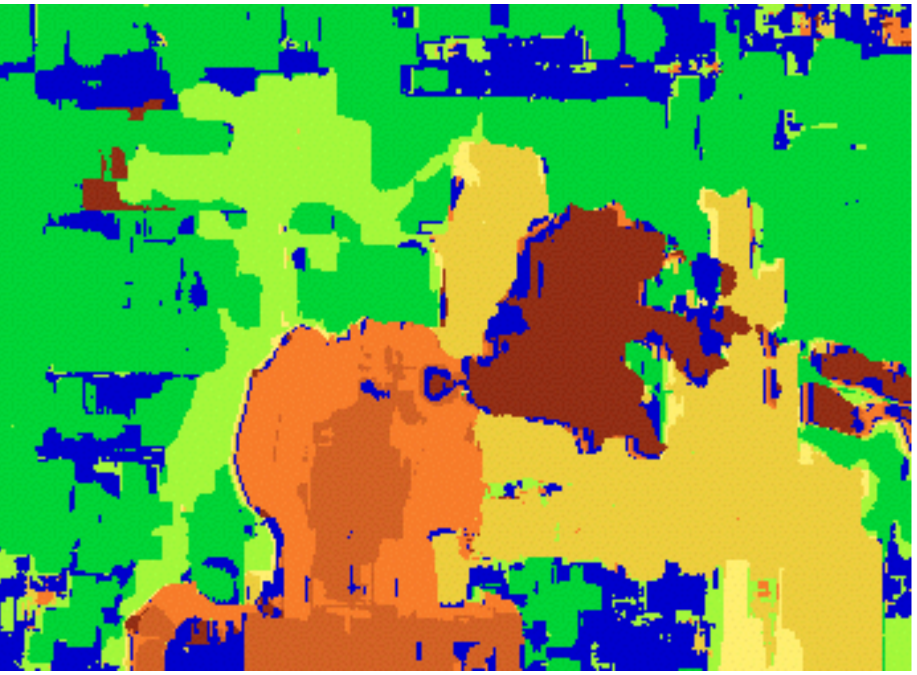
Dynamic Programming

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

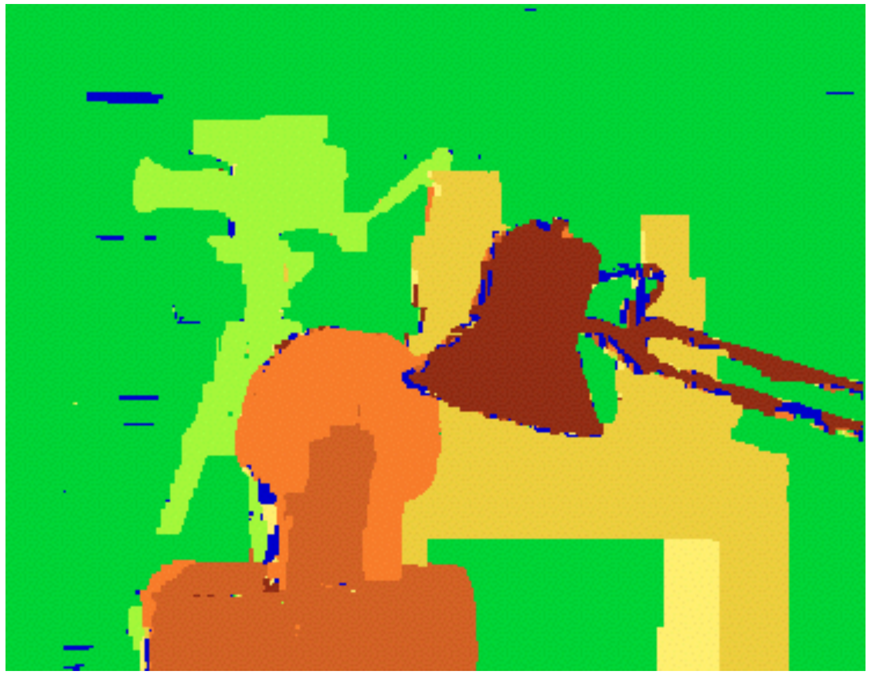
Can minimize this independently per scanline
using dynamic programming (DP) 

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

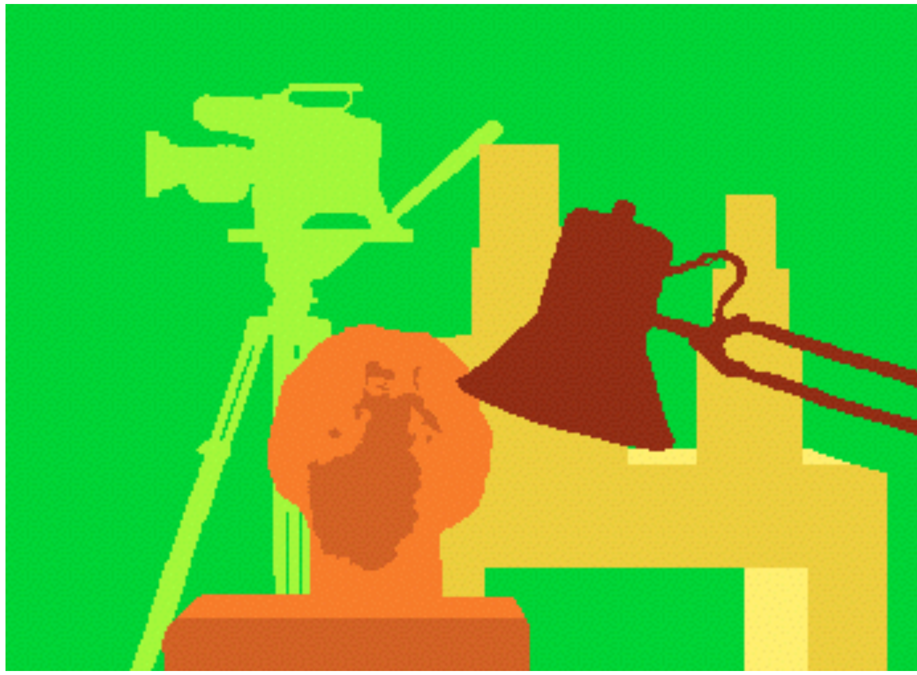
$$D(x, y, d) = C(x, y, d) + \min_{d'} \left\{ D(x-1, y, d') + \lambda \left| d - d' \right| \right\}$$



Match only

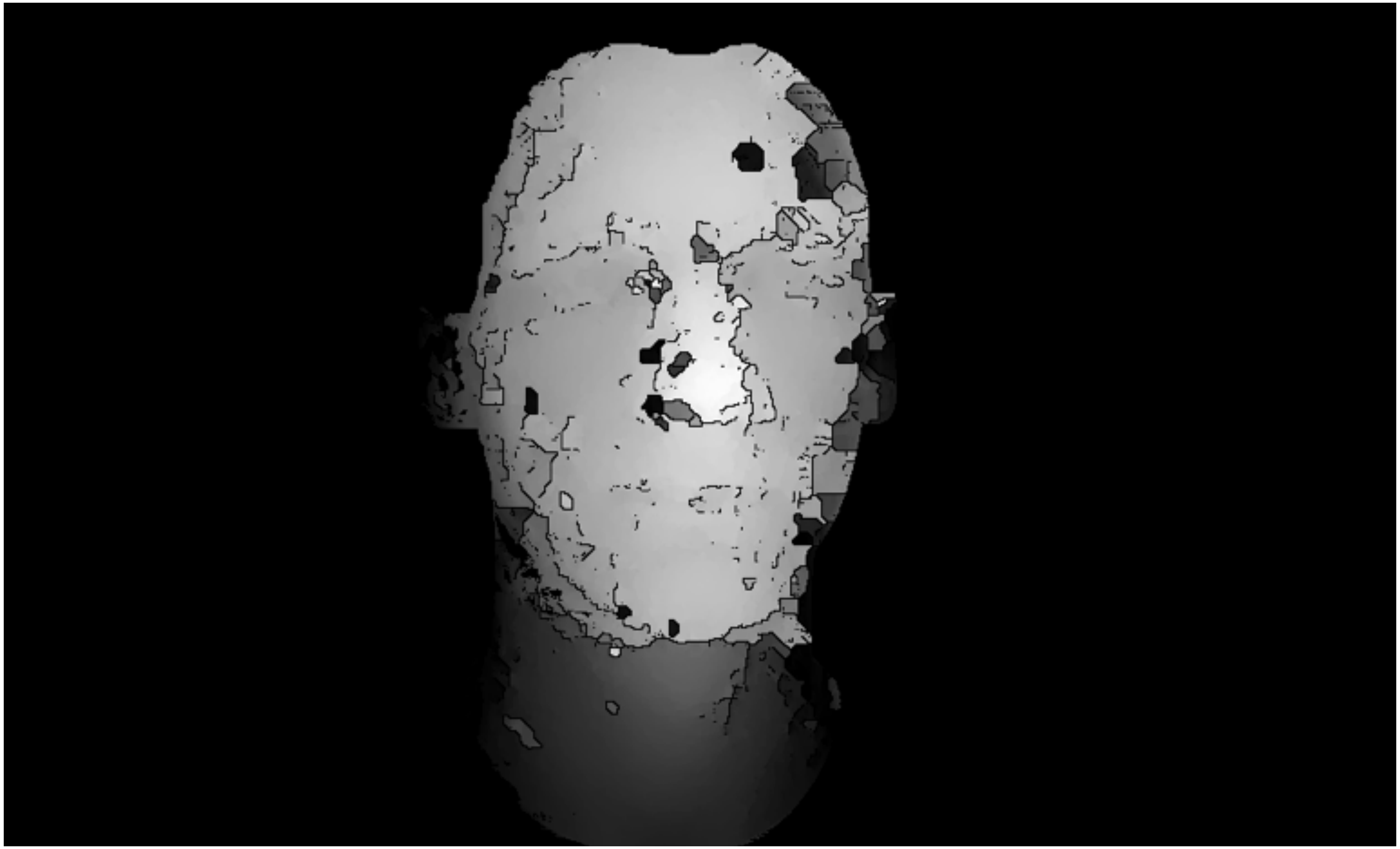


Match & smoothness
(graph cuts)



Ground Truth

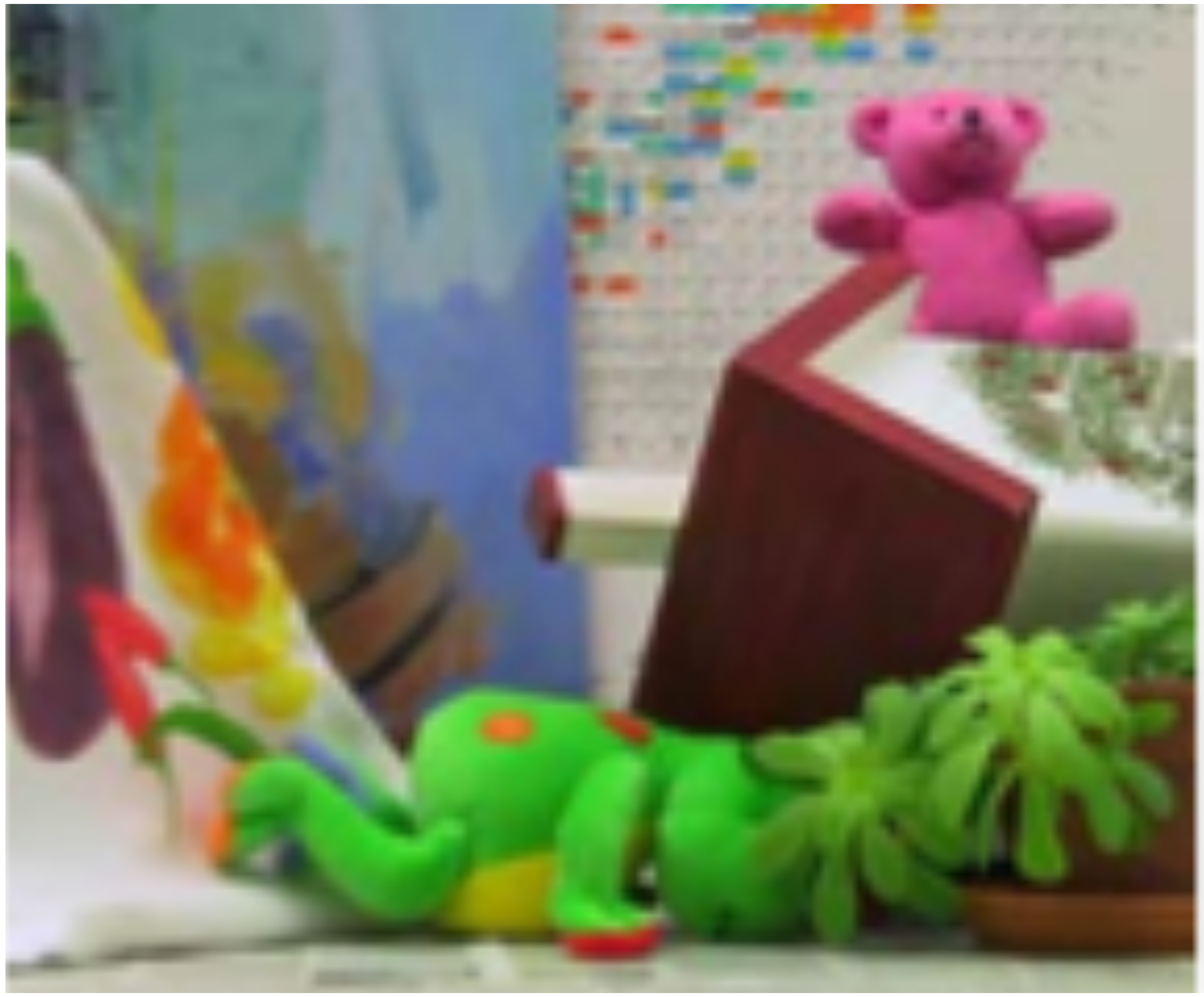
Y. Boykov, O. Veksler, and R. Zabih, Fast Approximate Energy Minimization via Graph Cuts, PAMI 2001

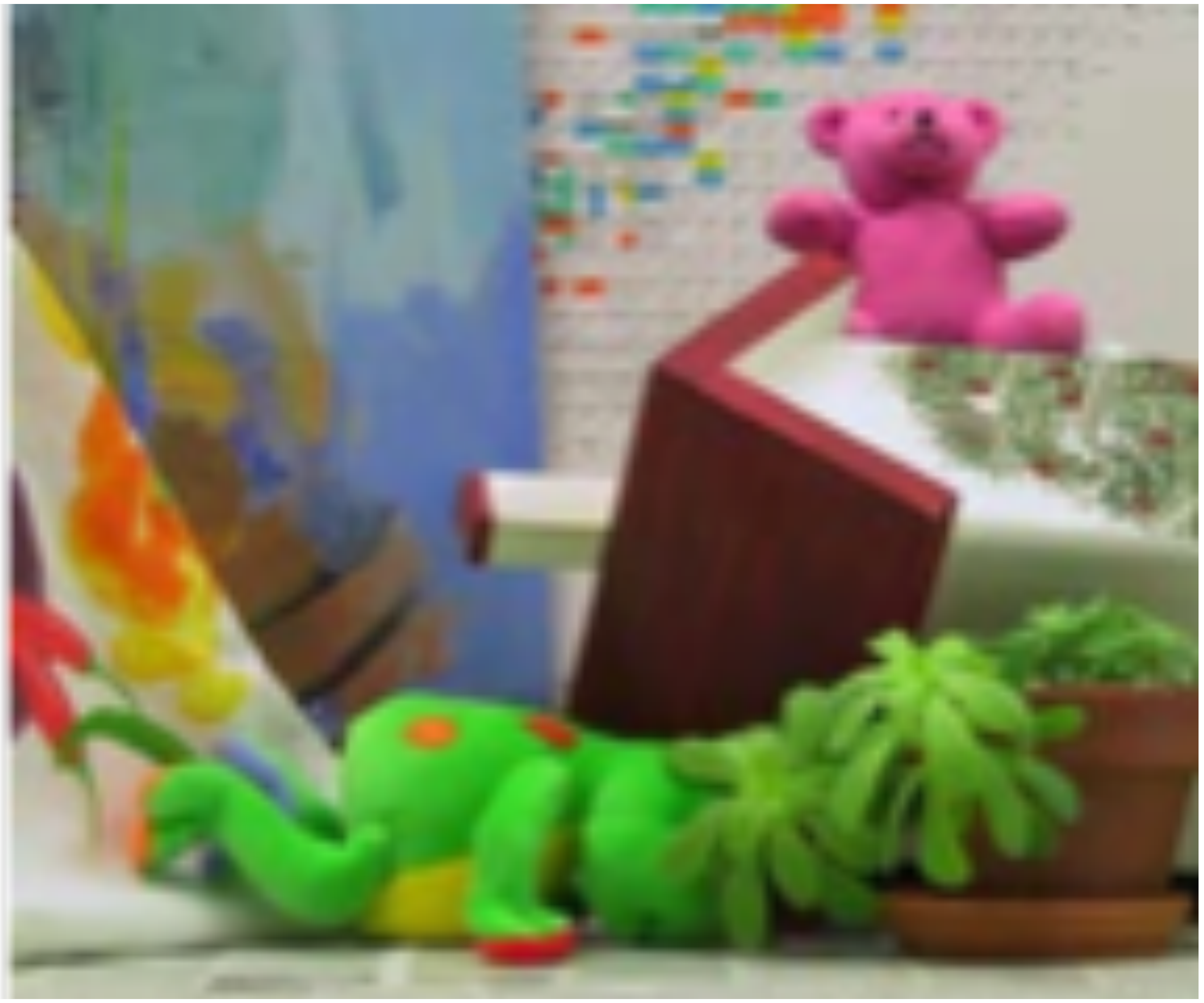


Stereo Vision

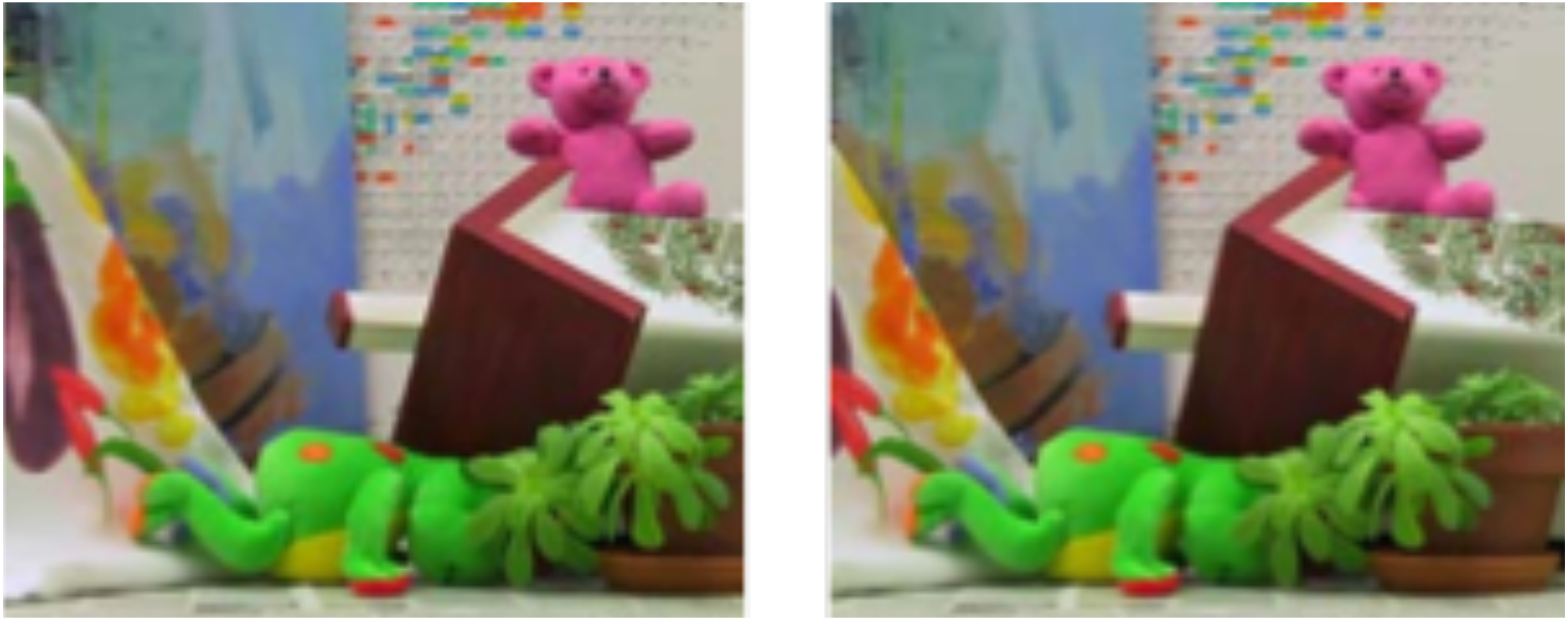
16-385 Computer Vision

Carnegie Mellon University (Kris Kitani)



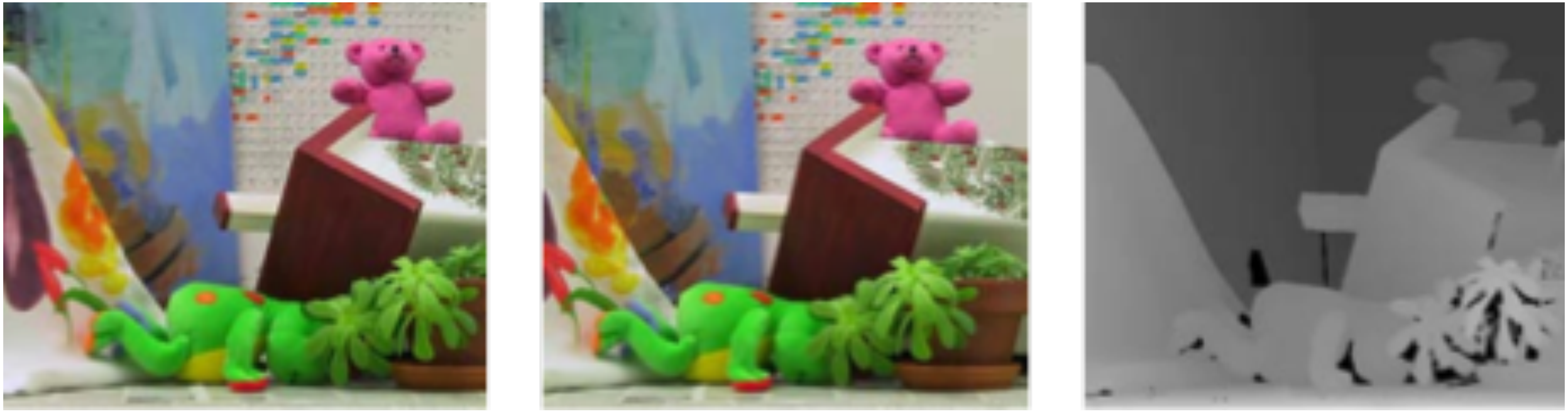


What's different between these two images?

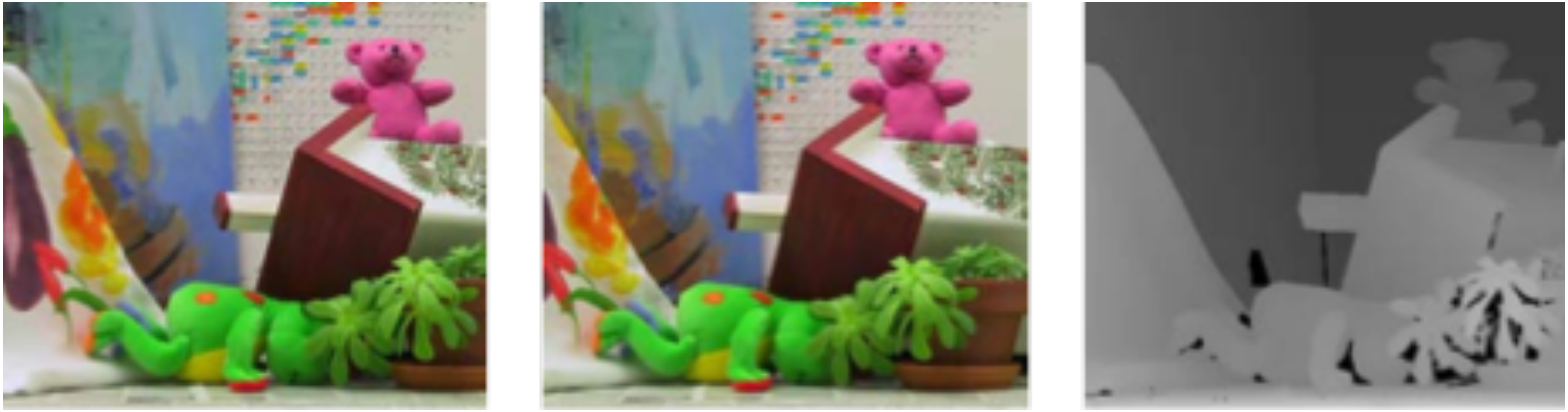


Objects that are close move more or less?

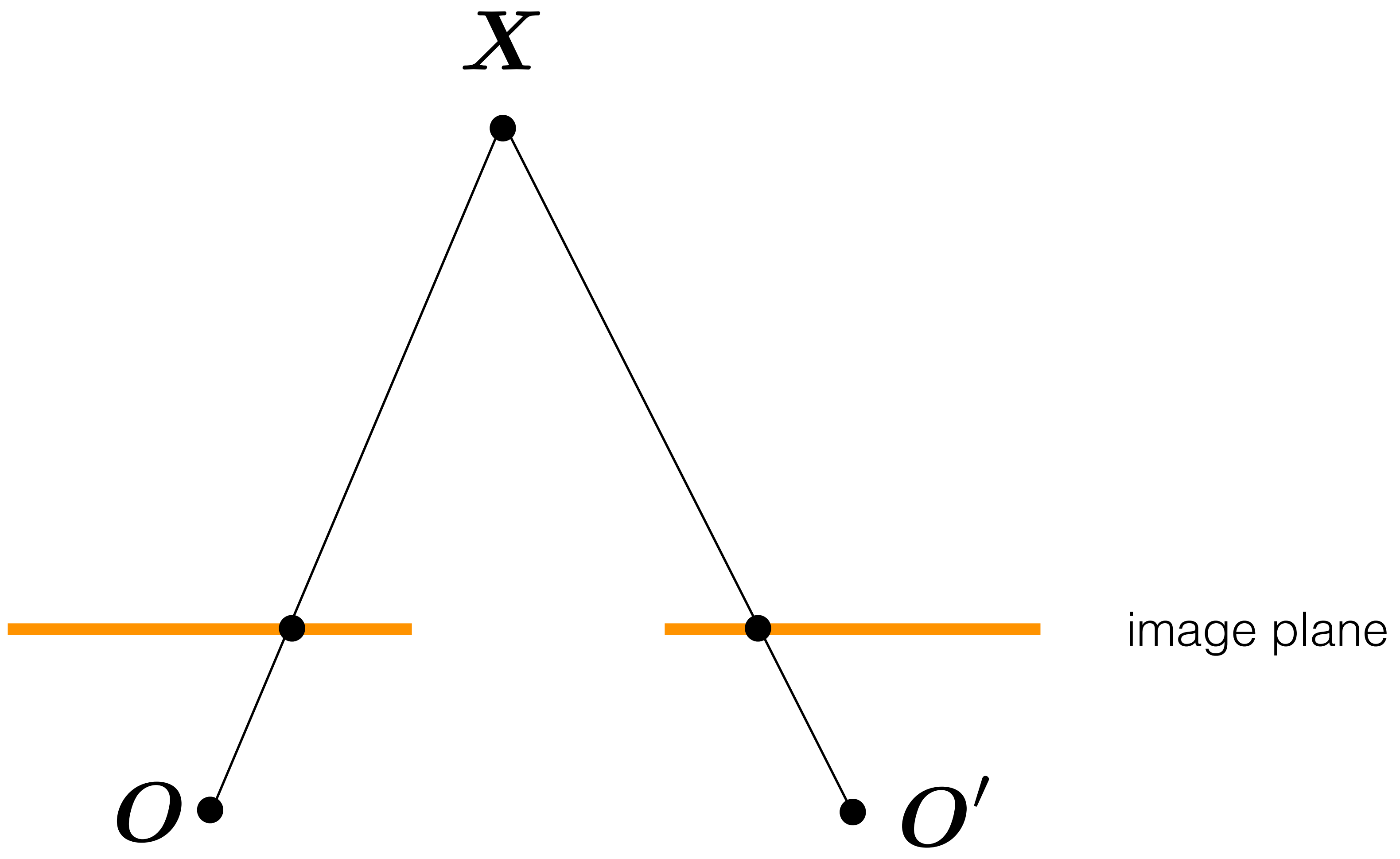
The amount of horizontal movement is inversely proportional to …

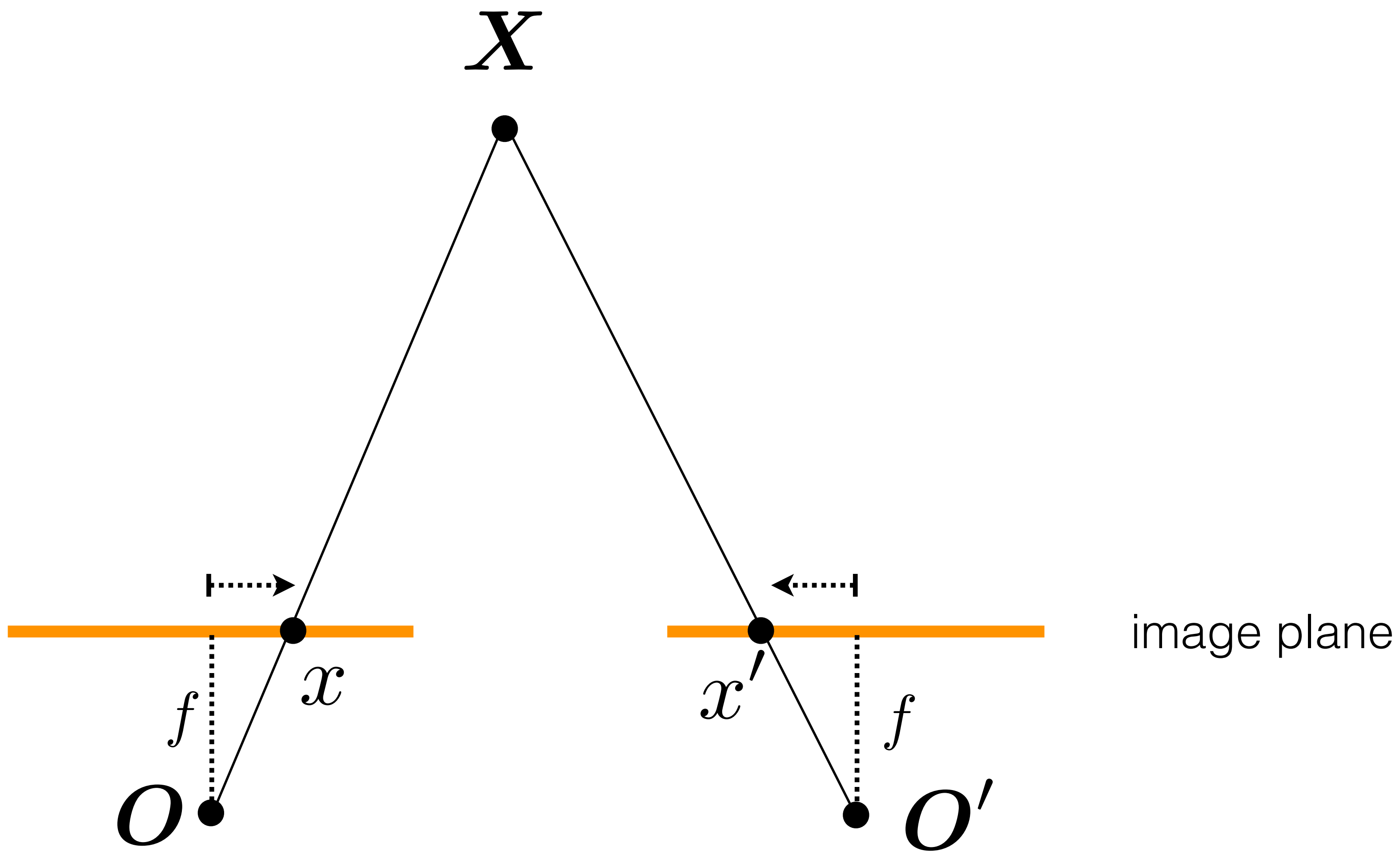


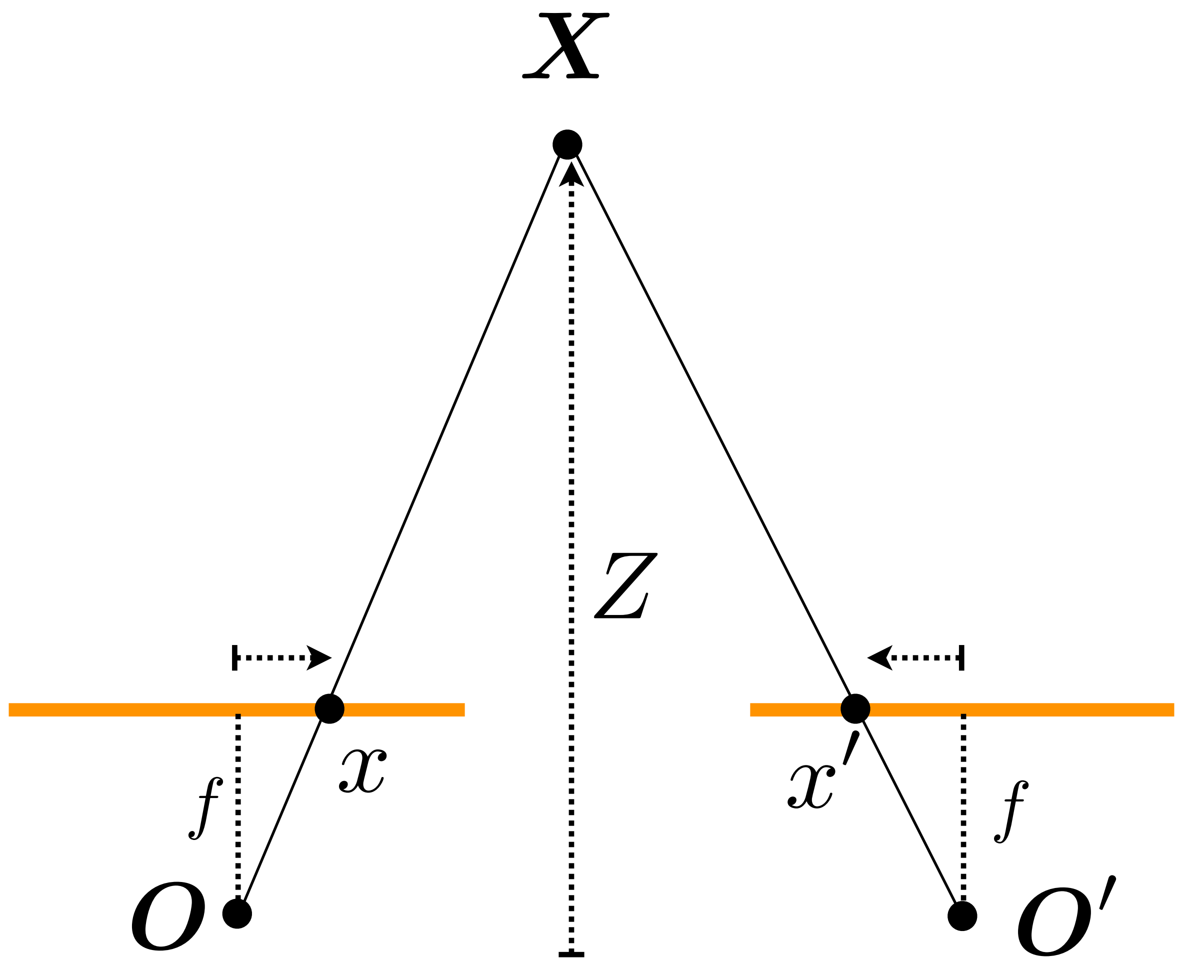
The amount of horizontal movement is inversely proportional to …

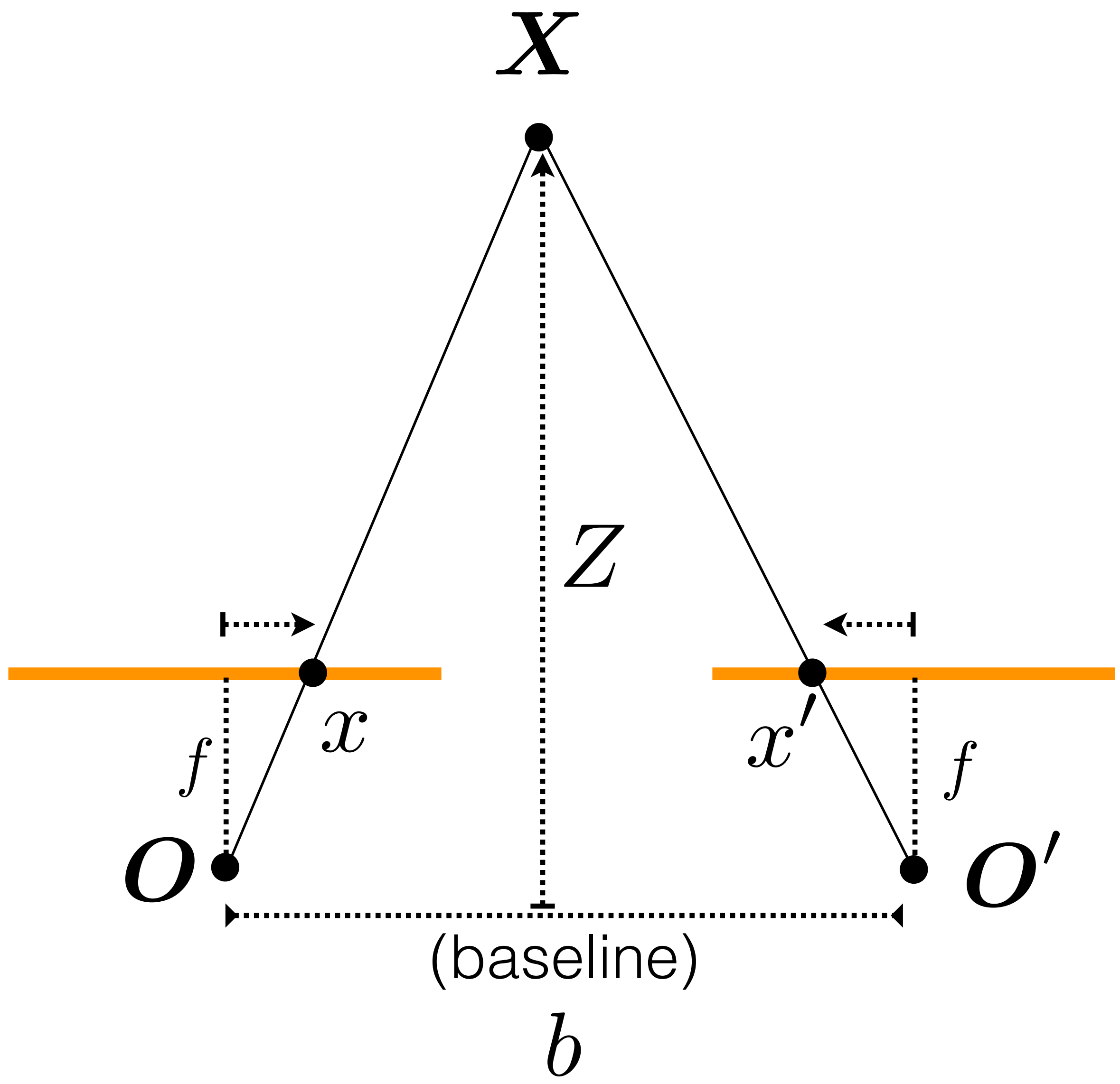


… the distance from the camera.

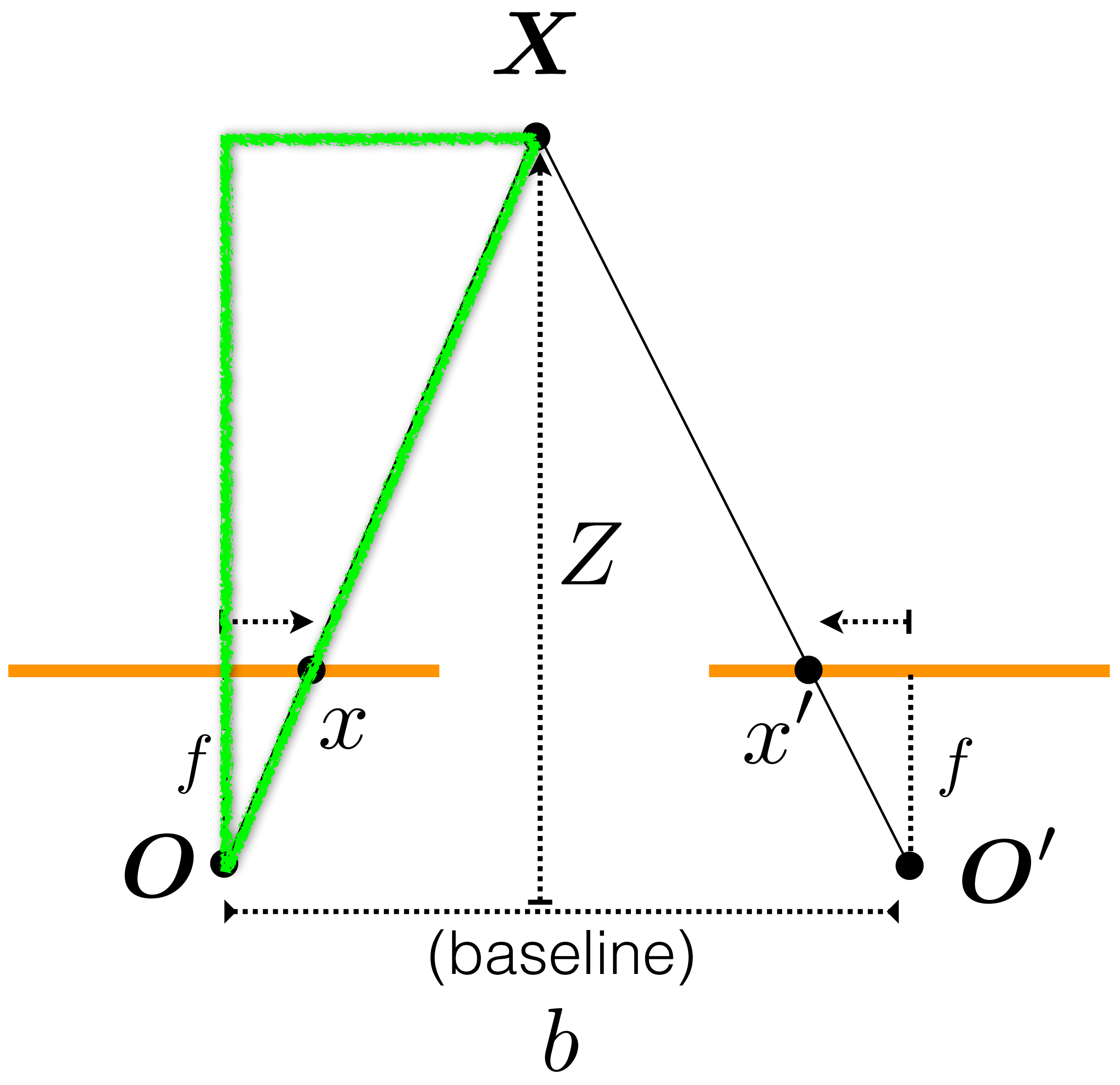

$\boldsymbol{X}$
image plane
$\boldsymbol{O}$
$\boldsymbol{O'}$


$\boldsymbol{X}$
image plane
x
x'
f
f
$\boldsymbol{O}$
$\boldsymbol{O}'$

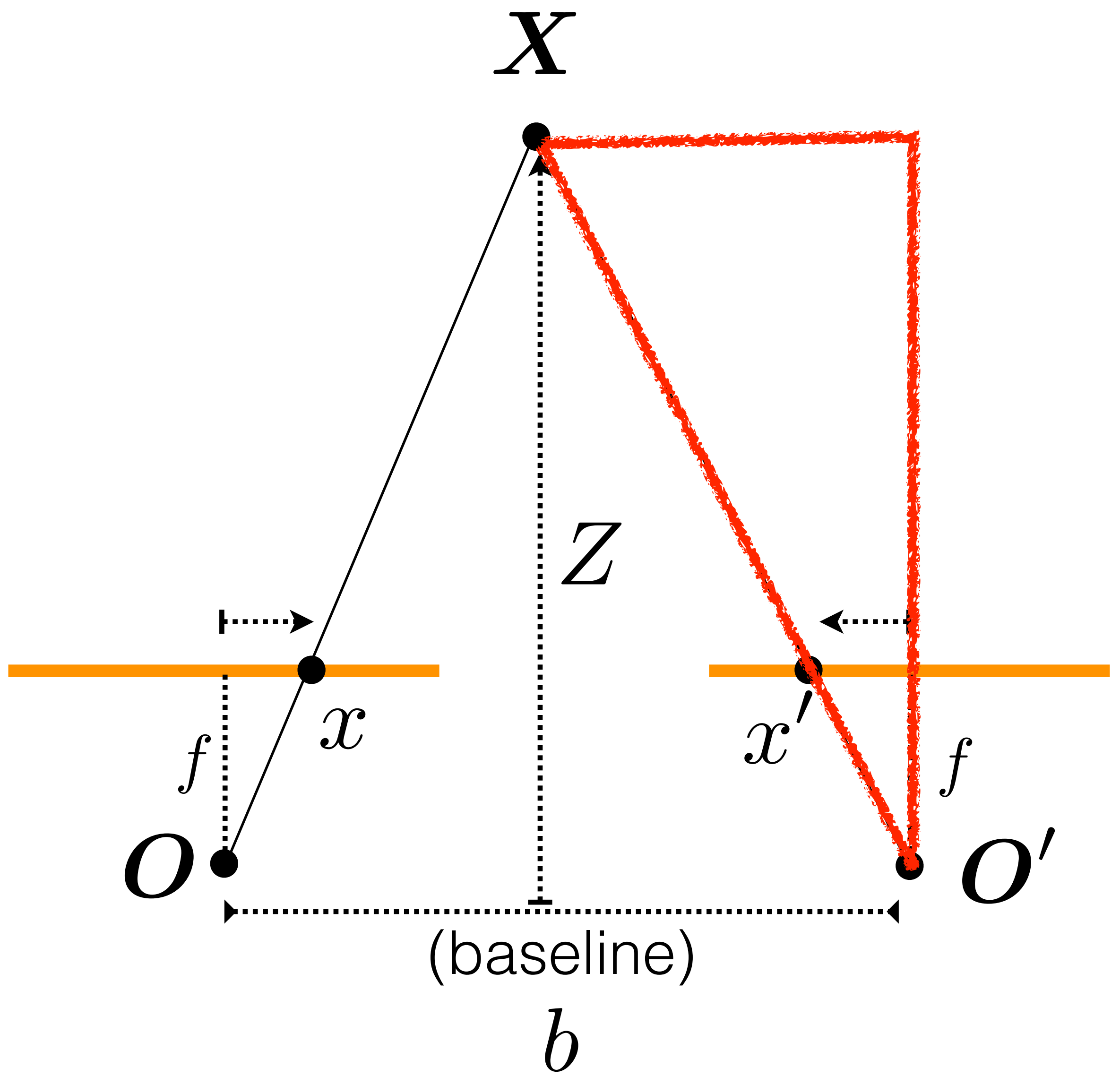

X
Z
x
x'
f
f
O
O'


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

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

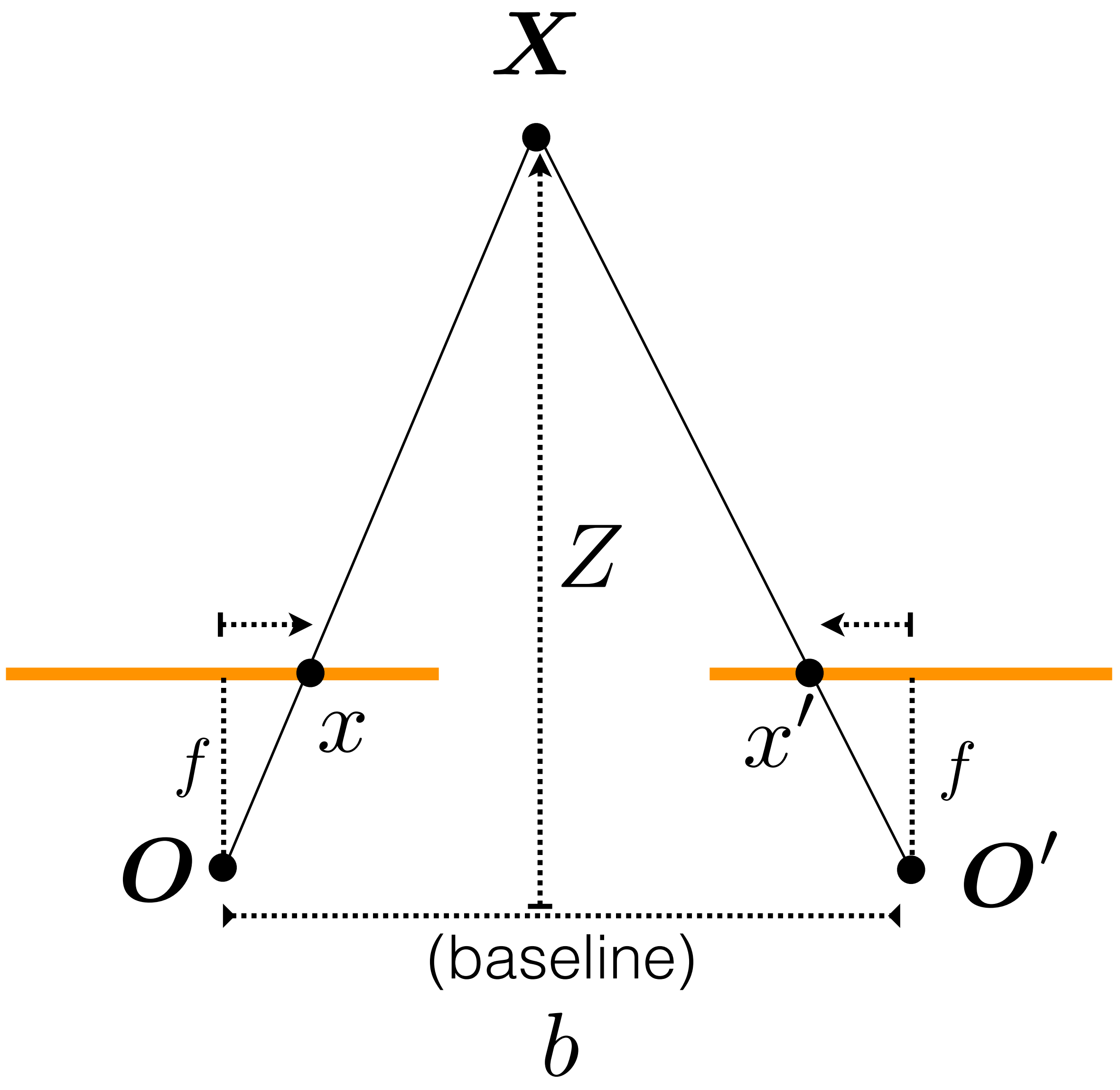


$$\frac{X}{Z} = \frac{x}{f}$$



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

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



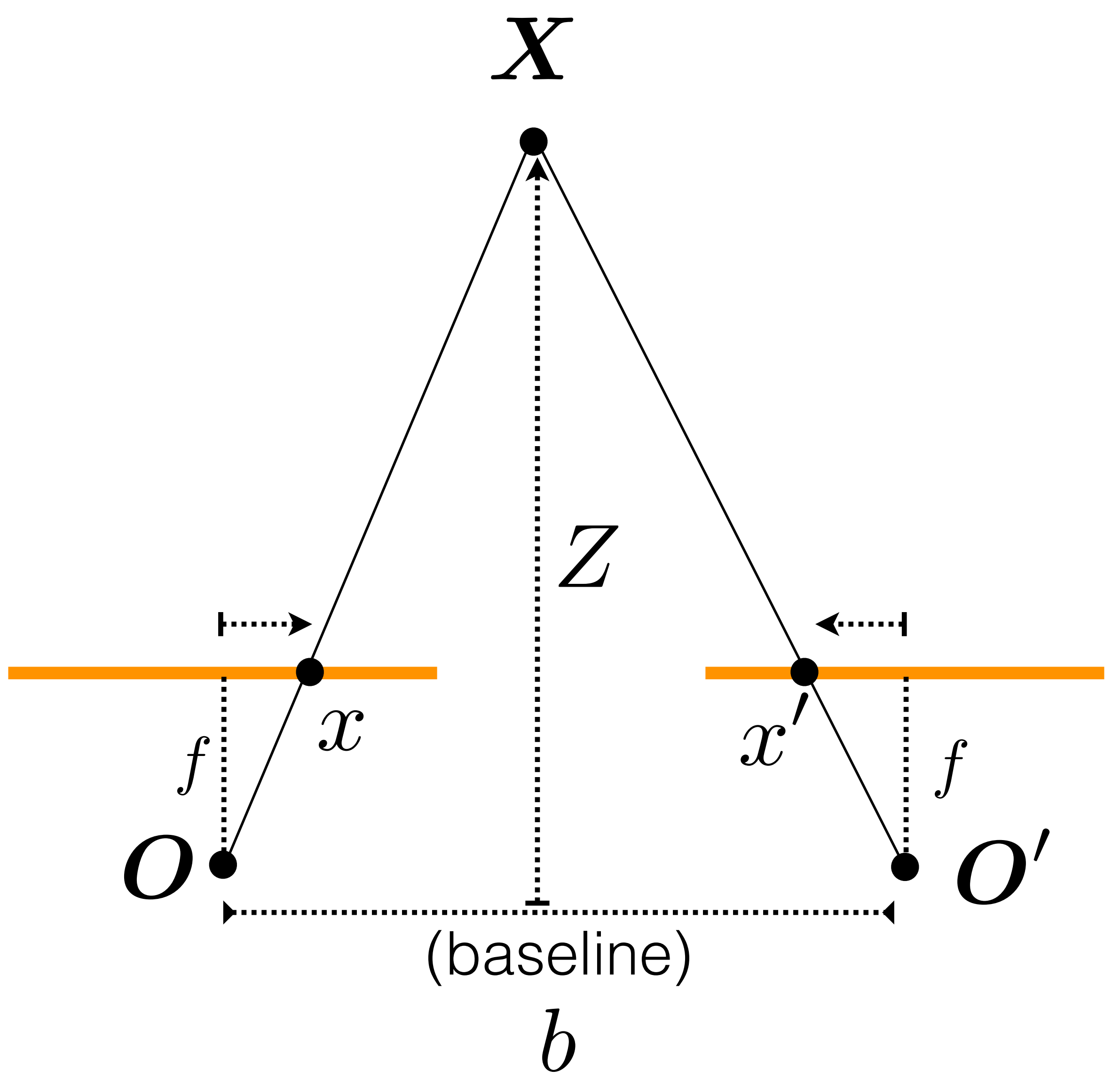
$$\frac{b - X}{Z} = \frac{x'}{f}$$

Disparity

$$d = x - x'$$

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

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



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

Disparity

$$d = x - x'$$

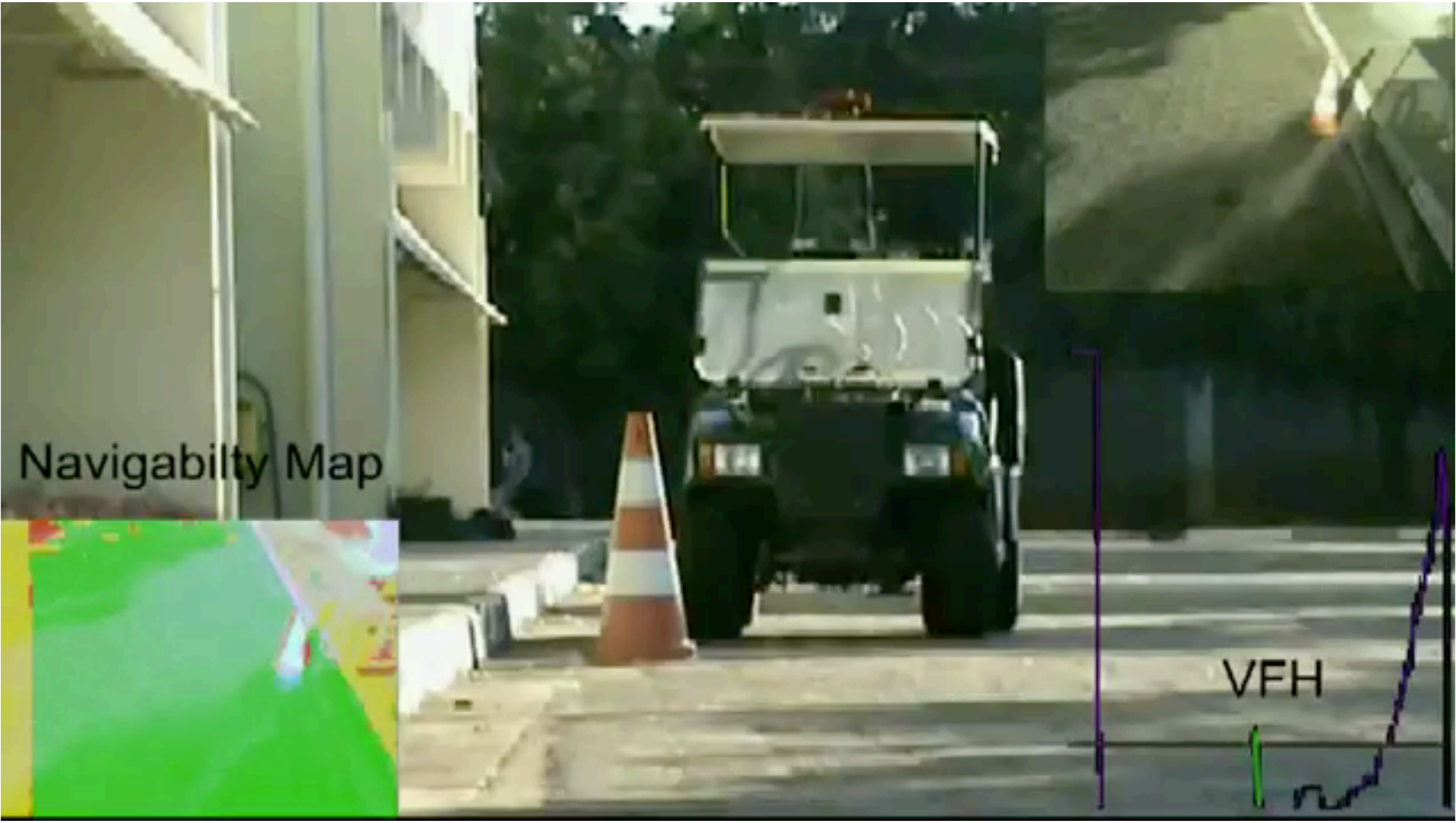
$$= \frac{bf}{Z}$$

inversely proportional to depth

Real-time stereo sensing

Nomad robot searches for meteorites in Antartica

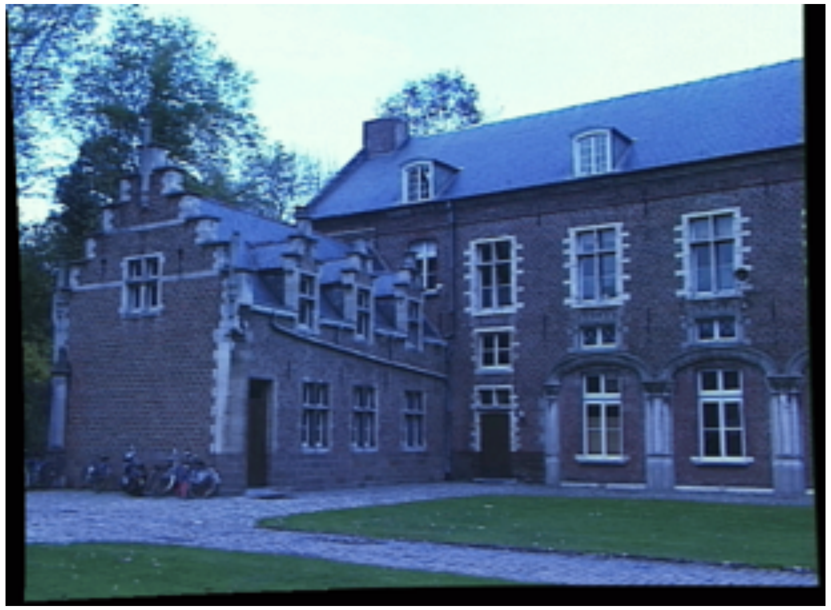
http://www.frc.ri.cmu.edu/projects/meteorobot/index.html

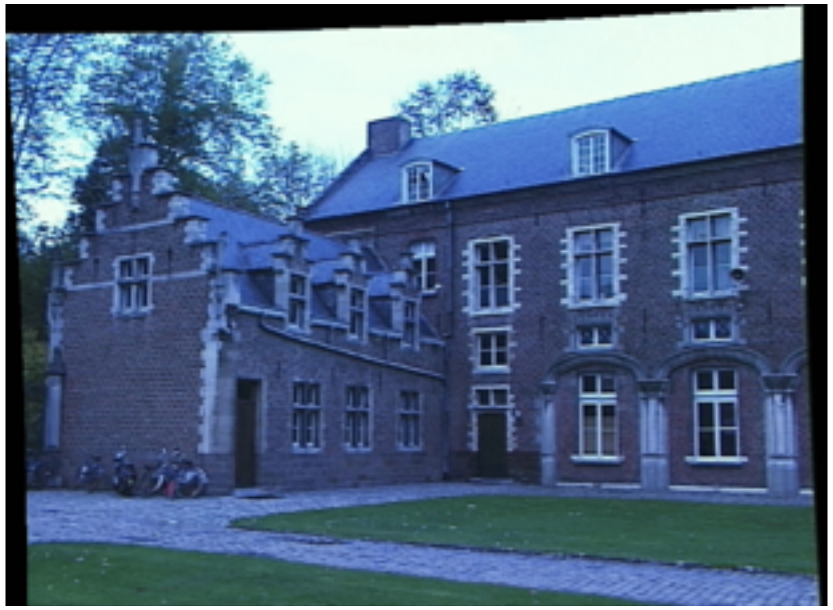


Navigabilty Map
VFH

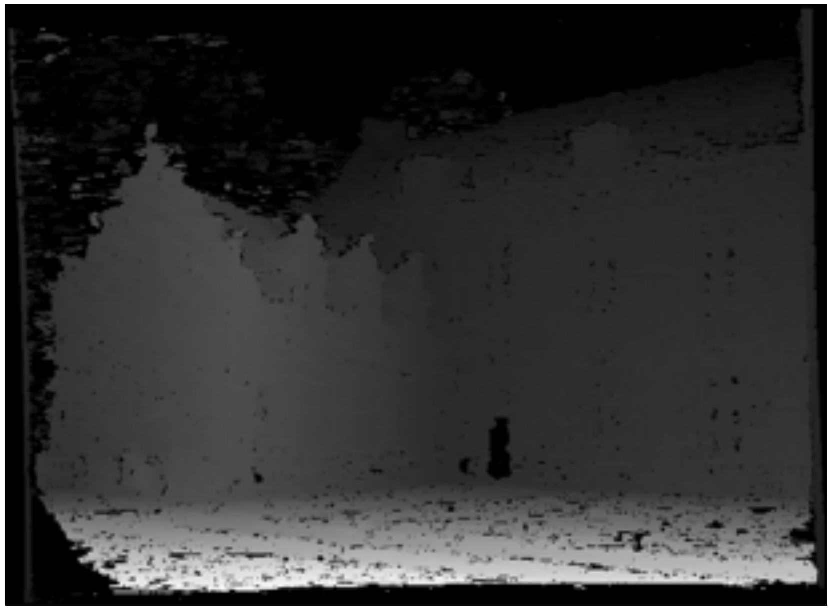
Subaru
Eyesight system

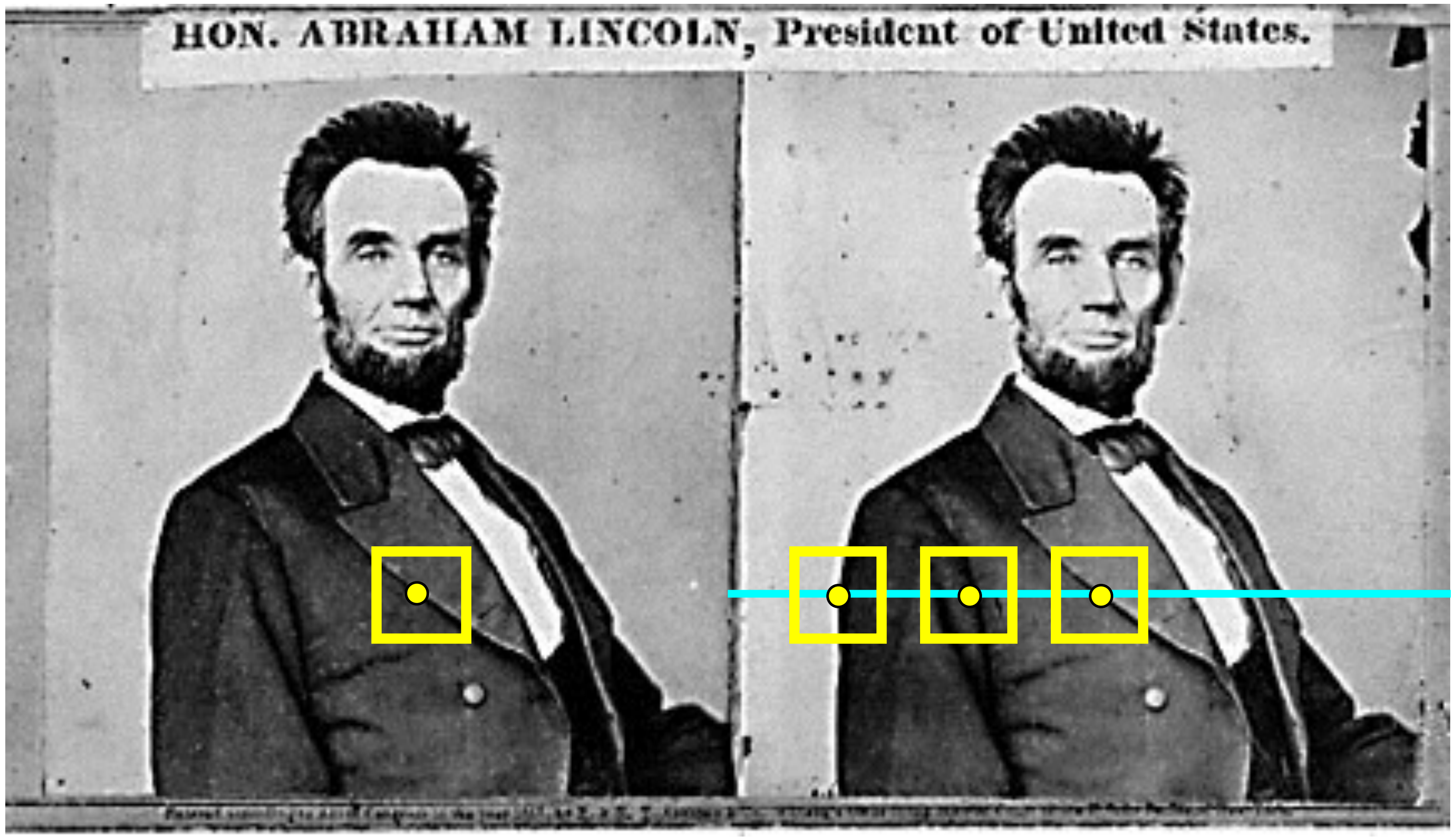
Pre-collision
braking





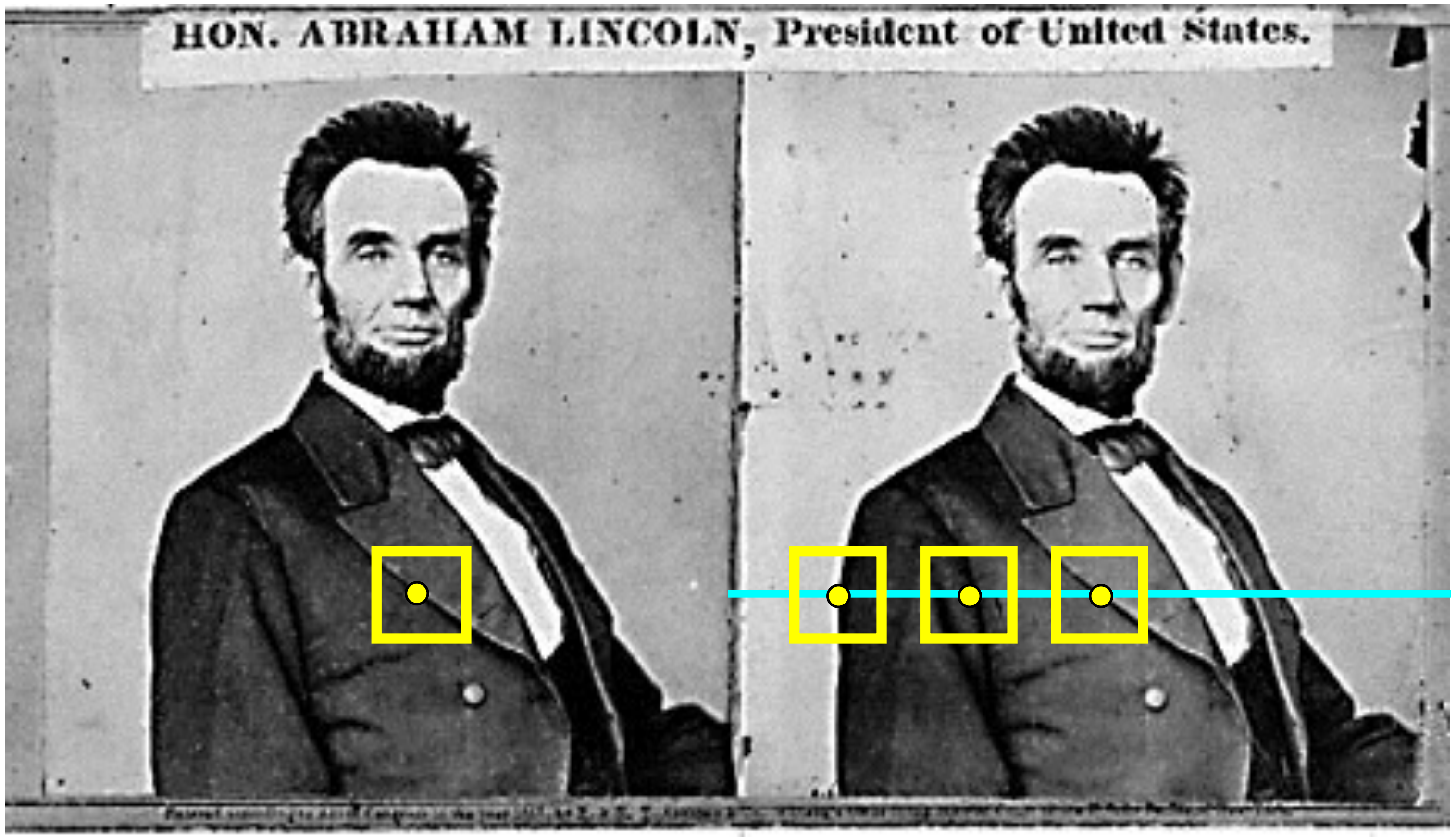
How so you compute depth
from a stereo pair?



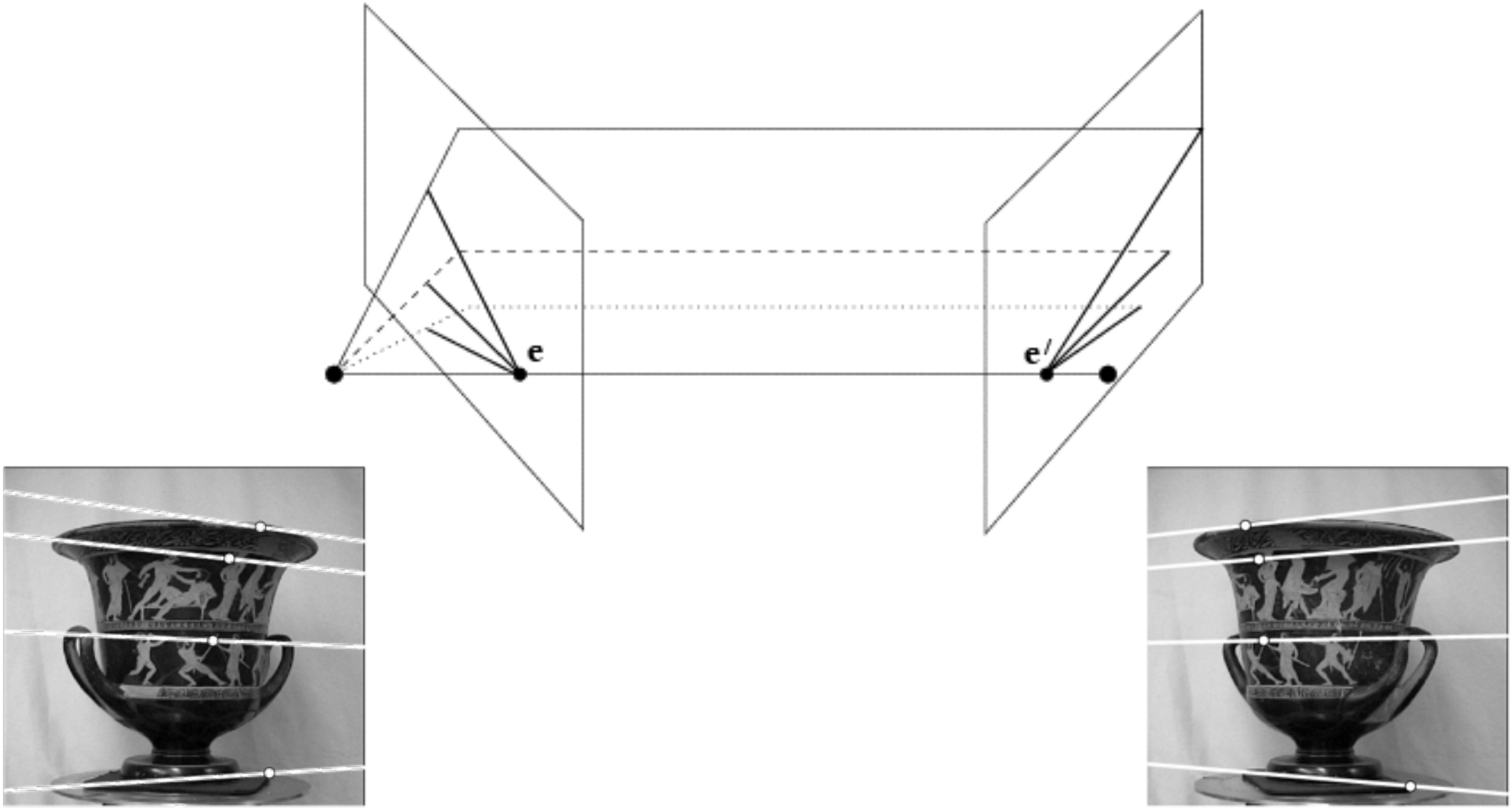


1. Rectify images
   (make epipolar lines horizontal)
2. For each pixel
   a. Find epipolar line
   b. Scan line for best match
   c. Compute depth from disparity

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

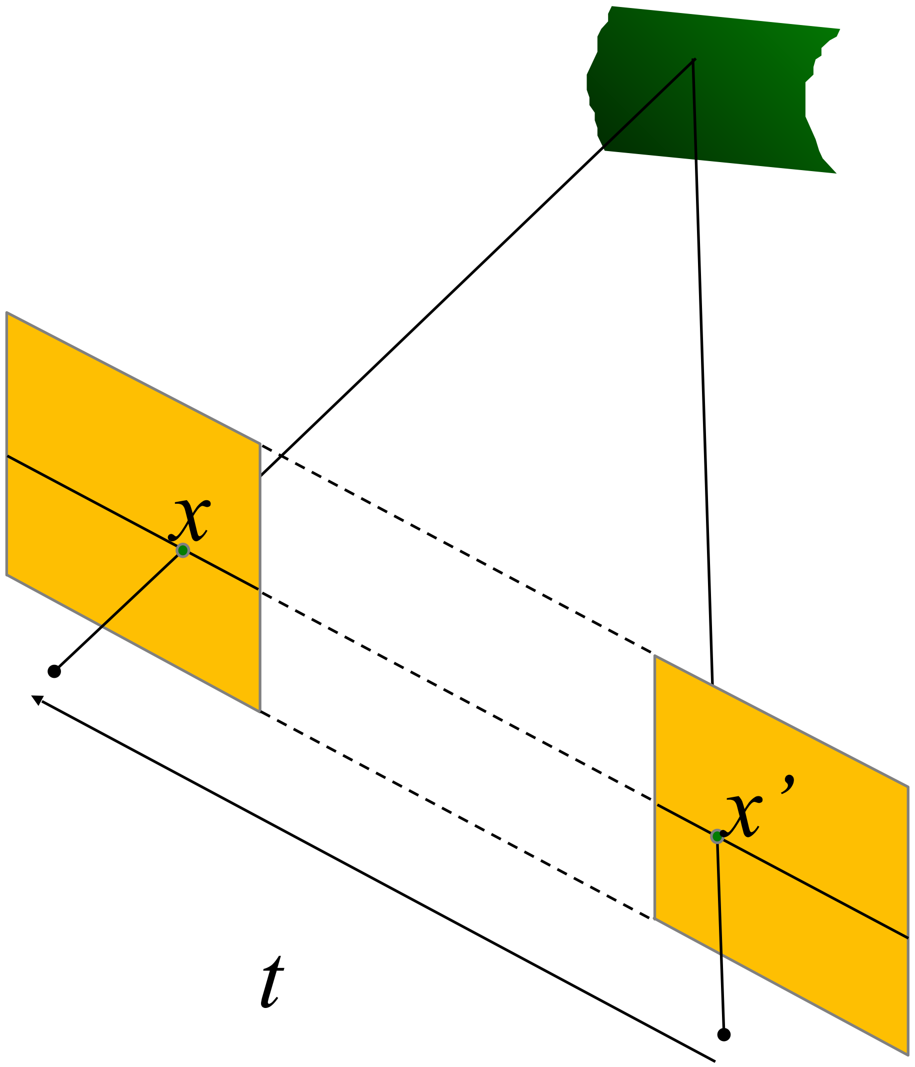


How can you make the epipolar lines horizontal?



It’s hard to make the image planes exactly parallel

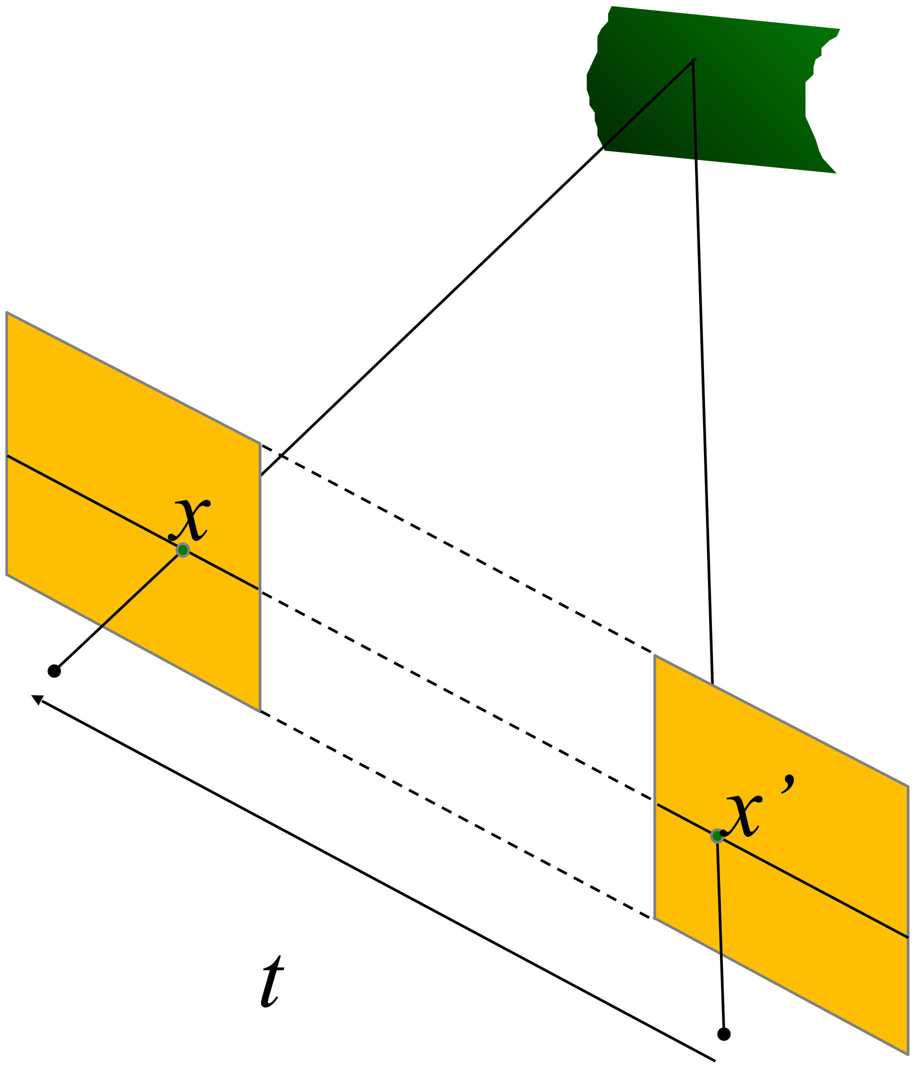
How can you make the epipolar lines horizontal?



When this relationship holds

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

How can you make the epipolar lines horizontal?



When this relationship holds

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

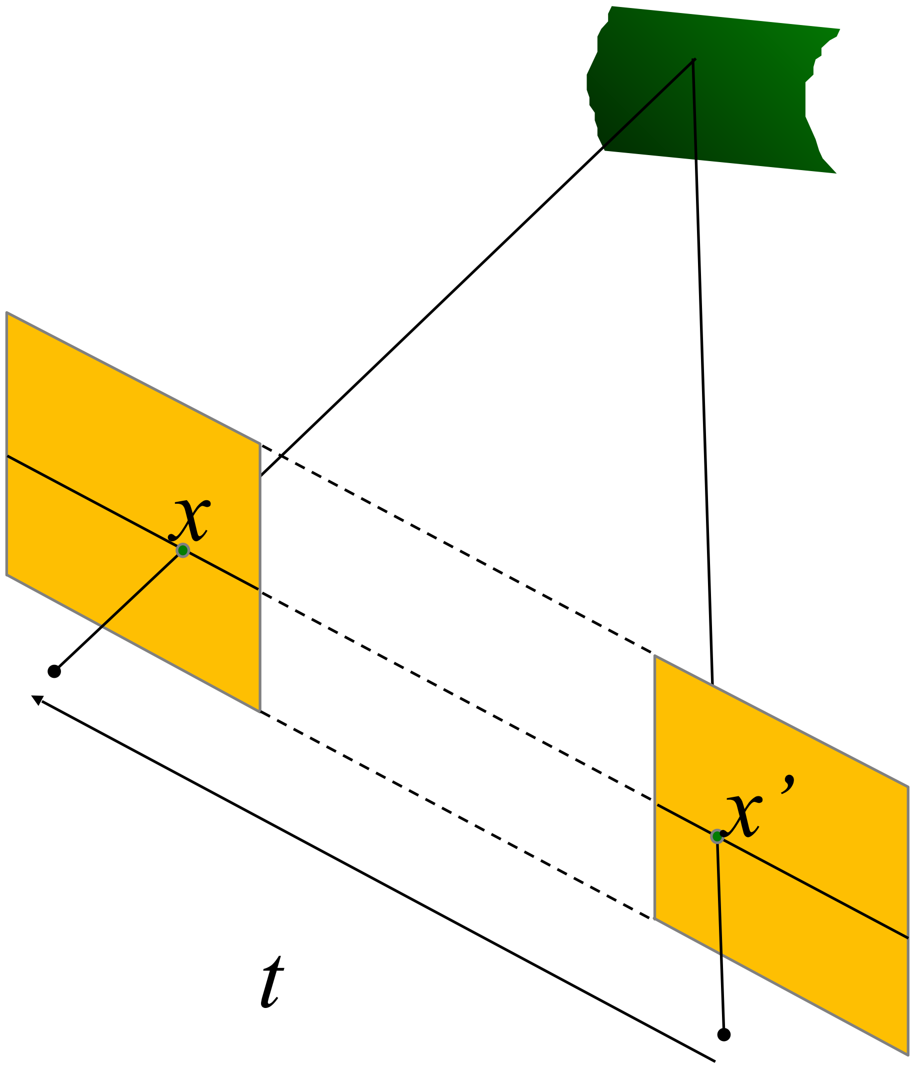
Let's try this out…

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

This always has to hold

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

How can you make the epipolar lines horizontal?



When this relationship holds

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

Let's try this out…

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

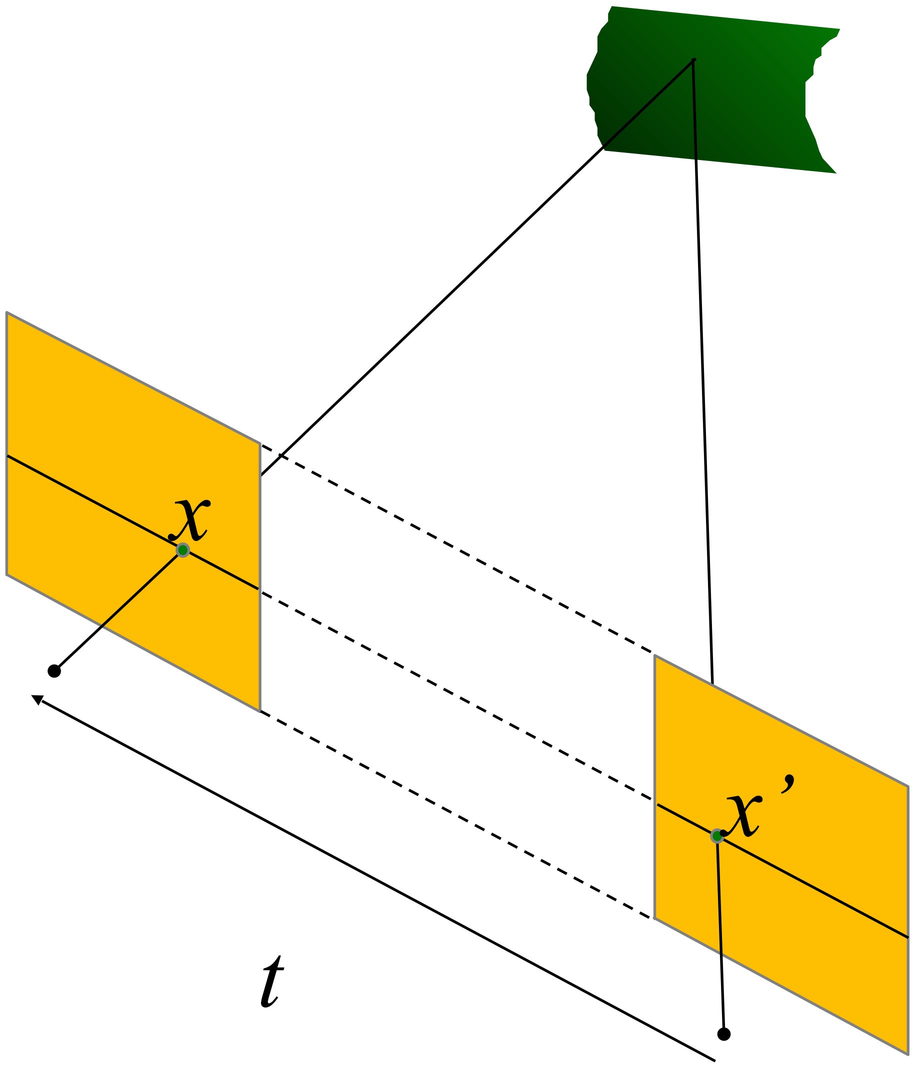
This always has to hold

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

Write out the constraint

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

How can you make the epipolar lines horizontal?



When this relationship holds

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

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

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

Write out the constraint

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

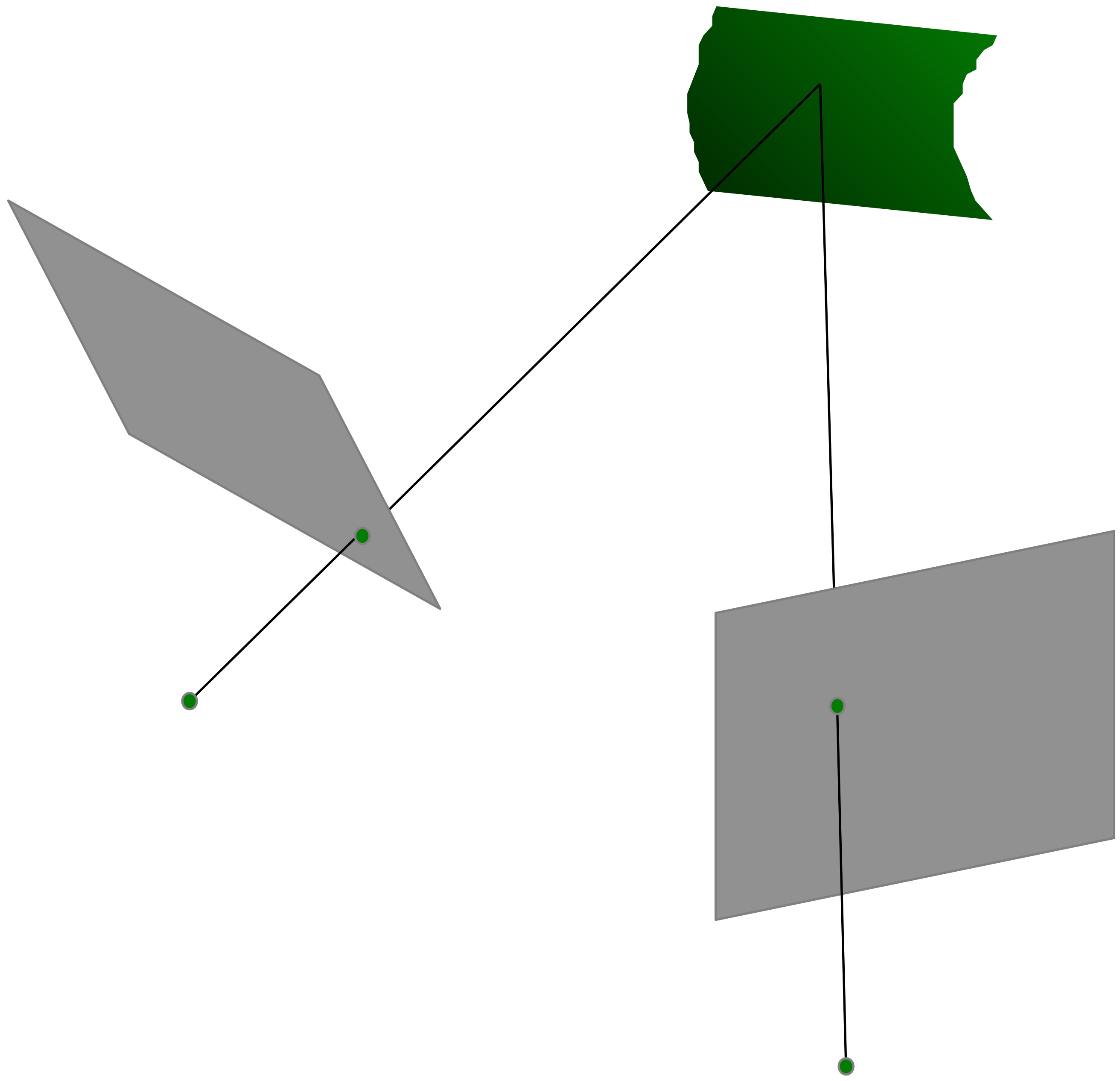
$$\begin{pmatrix} u & v & 1 \end{pmatrix} \begin{pmatrix} 0 \\ -T \\ Tv' \end{pmatrix} = 0$$

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

$$\boxed{Tv = Tv'}$$

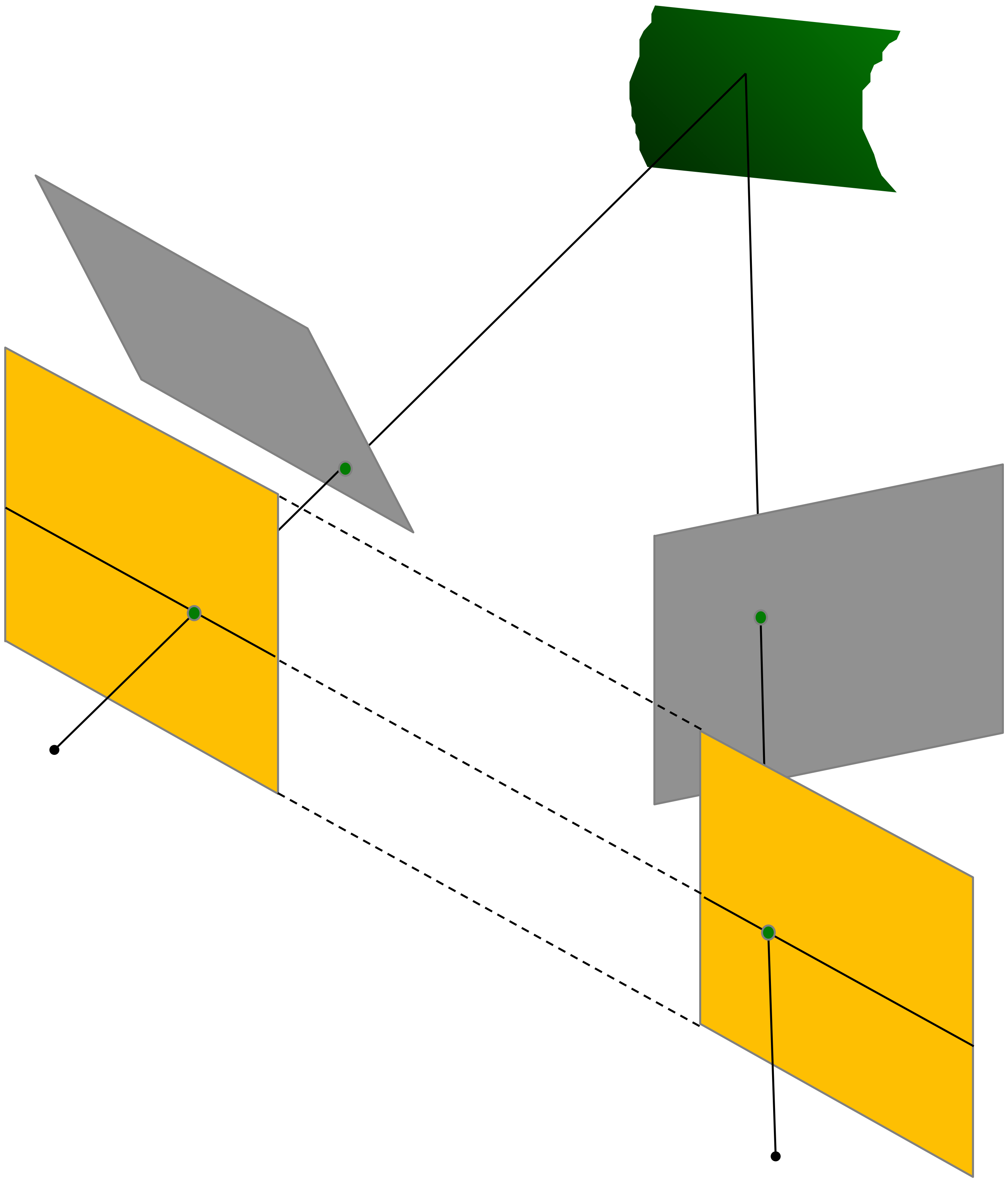
y coordinate is always the same!

What is stereo rectification?



What is stereo rectification?

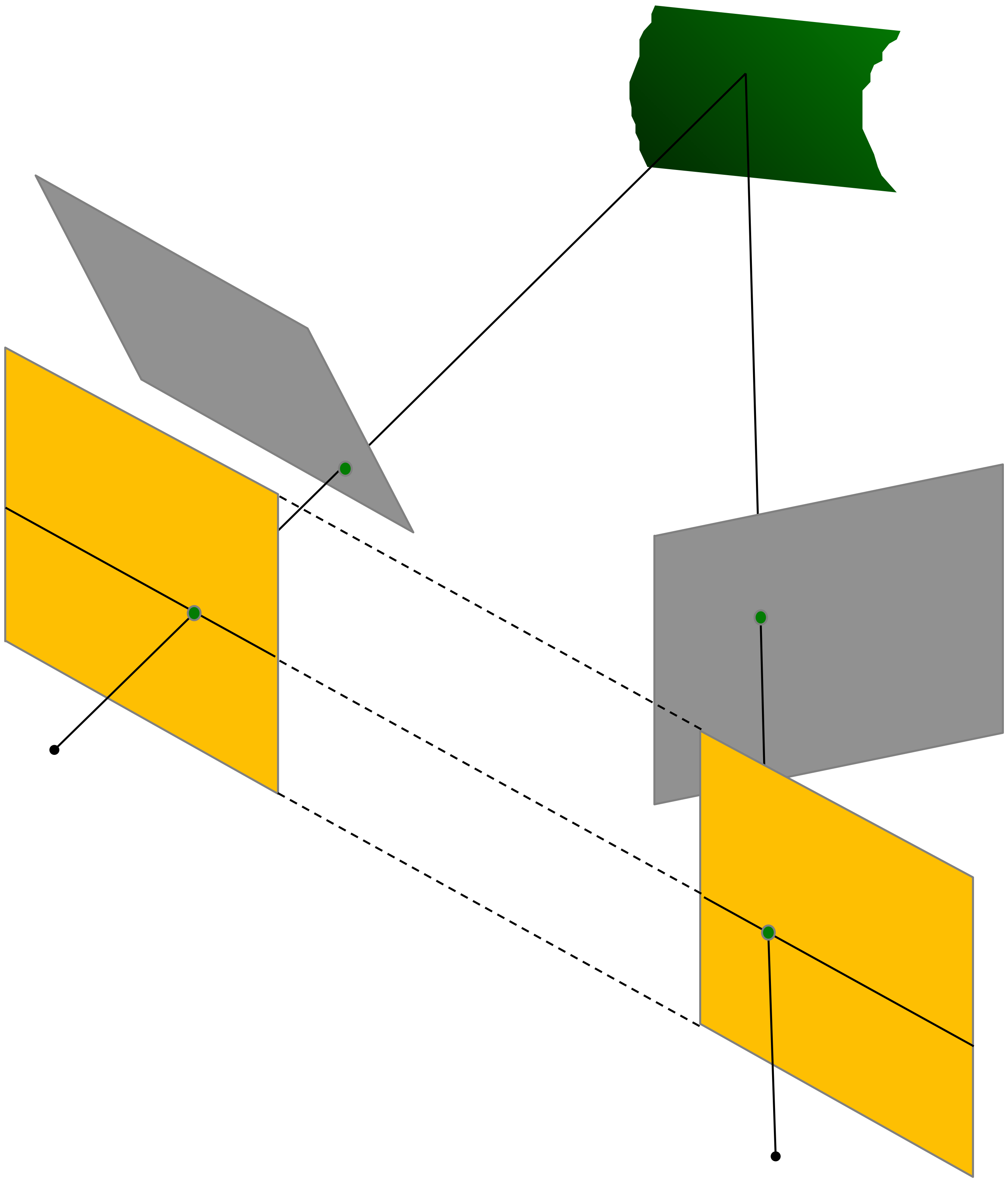
Reproject image planes onto a common plane parallel to the line between camera centers



What is stereo rectification?

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

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 left camera so that the epipole is at infinity

2. Apply the same rotation to the right camera

3. Rotate the right camera by $\mathbf{R}^{-1}$

4. Adjust the scale

Setting the epipole to infinity

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

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

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

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

then $$R_{\text{rect}}\boldsymbol{e}_1 = \left[\begin{array}{c} \boldsymbol{r}_1^\top \boldsymbol{e}_1 \\ \boldsymbol{r}_2^\top \boldsymbol{e}_1 \\ \boldsymbol{r}_3^\top \boldsymbol{e}_1 \end{array}\right] = \left[\begin{array}{c} ? \\ ? \\ ? \end{array}\right]$$

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

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

Where is this point located on the image plane?

Stereo Rectification Algorithm

1. Estimate **E** using the 8 point algorithm (SVD)
2. Estimate the epipole **e** (SVD of **E**)
3. Build $\mathbf{R}_{\text{rect}}$ from **e**
4. Decompose **E** into **R** and **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 $\mathbf{Ee}=0$)

3. Build $\mathbf{R}_{\mathrm{rect}}$ from $\mathbf{e}$

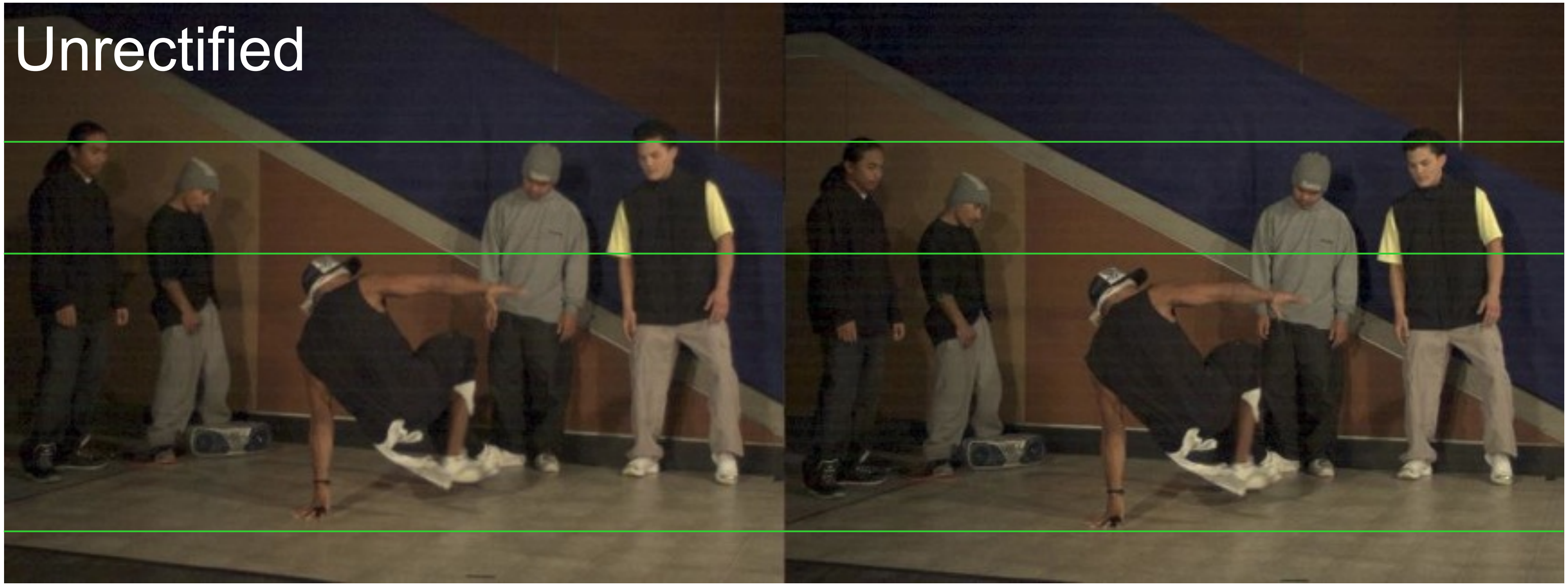
4. Decompose $\mathbf{E}$ into $\mathbf{R}$ and $\mathbf{T}$

5. Set $\mathbf{R}_1=\mathbf{R}_{\mathrm{rect}}$ and $\mathbf{R}_2 = \mathbf{R}\mathbf{R}_{\mathrm{rect}}$

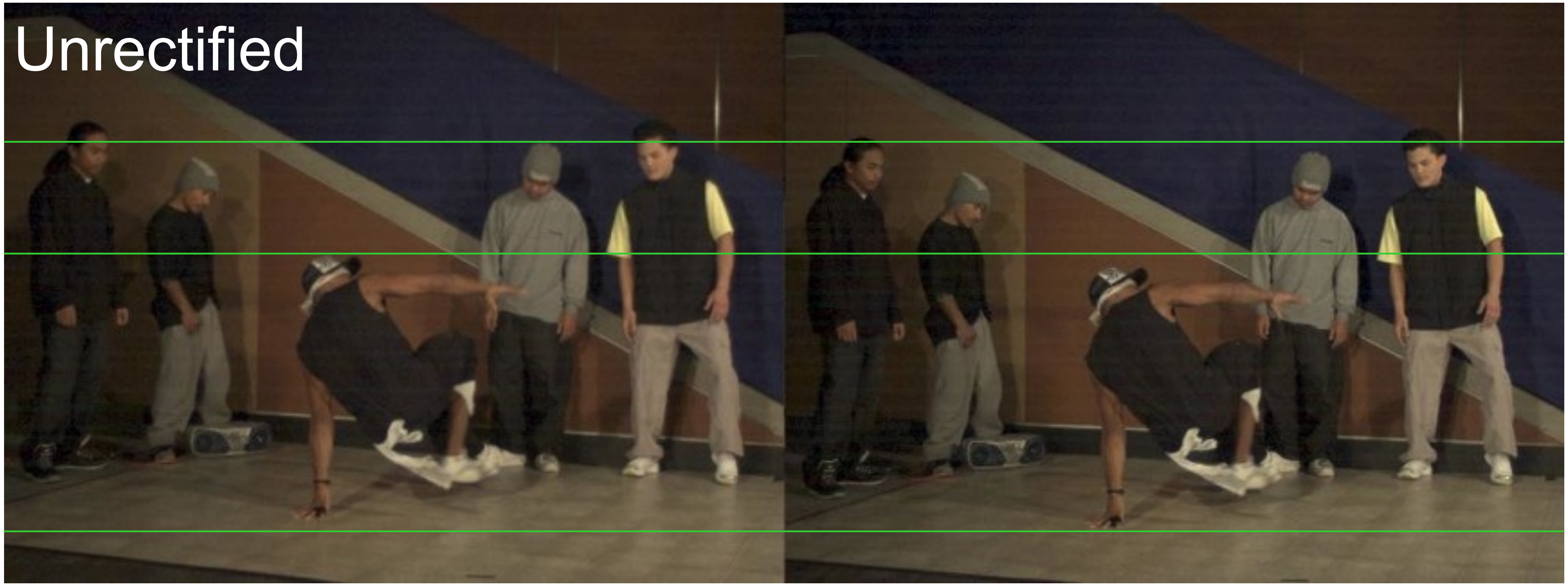
6. Rotate each left camera point $\mathbf{x}' \sim \mathbf{Hx}$ where $\mathbf{H} = \mathbf{K}\mathbf{R}_1$

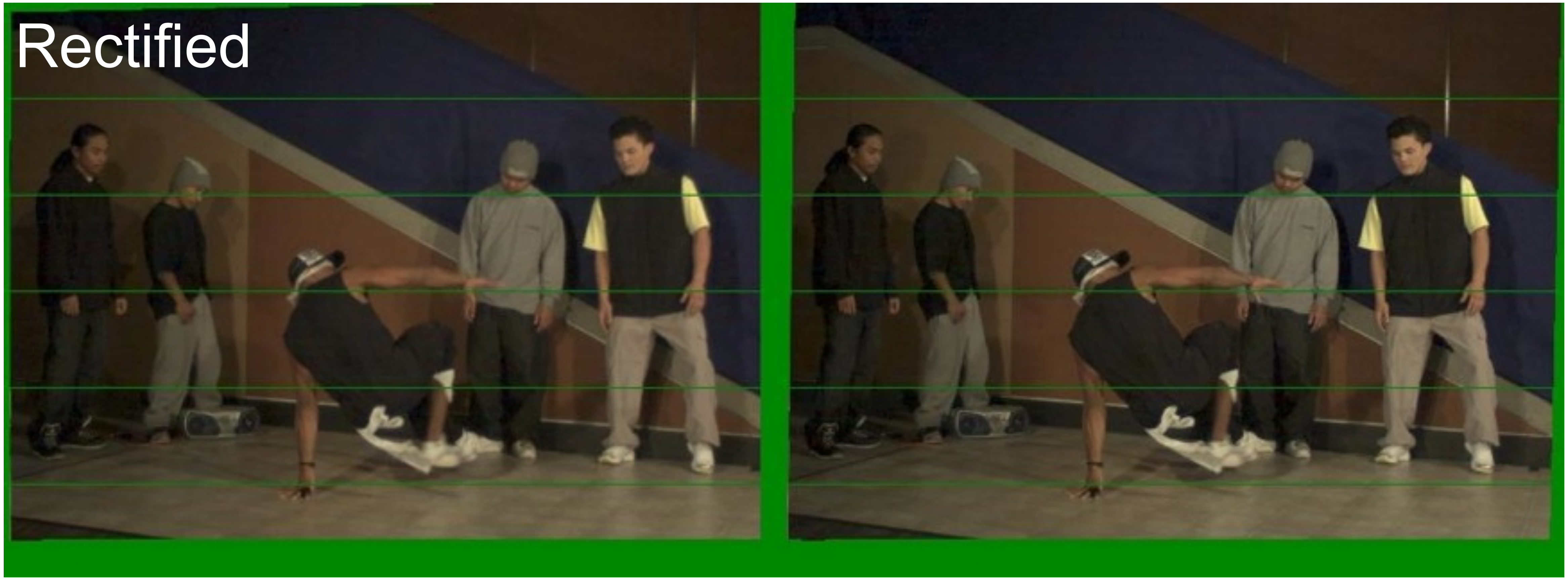
   *You may need to alter the focal length (inside **K**) to keep points within the original image size

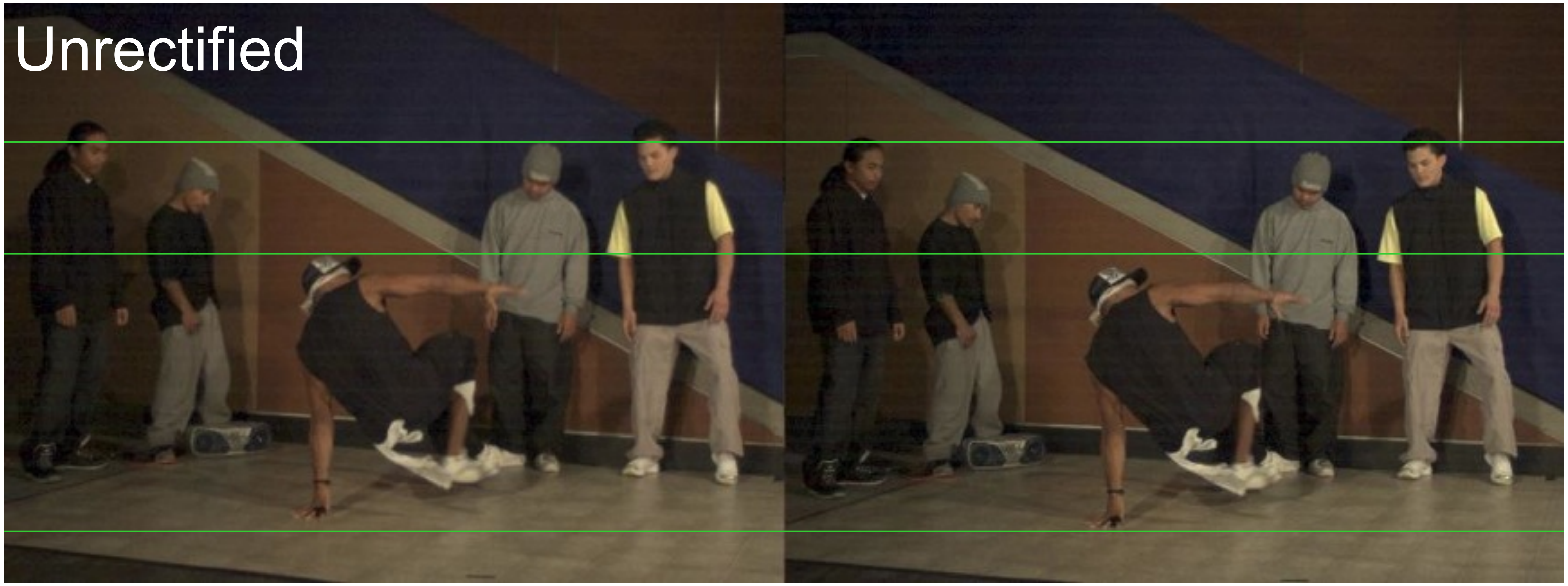
7. Repeat 6 and 7 for right camera points using $\mathbf{R}_2$



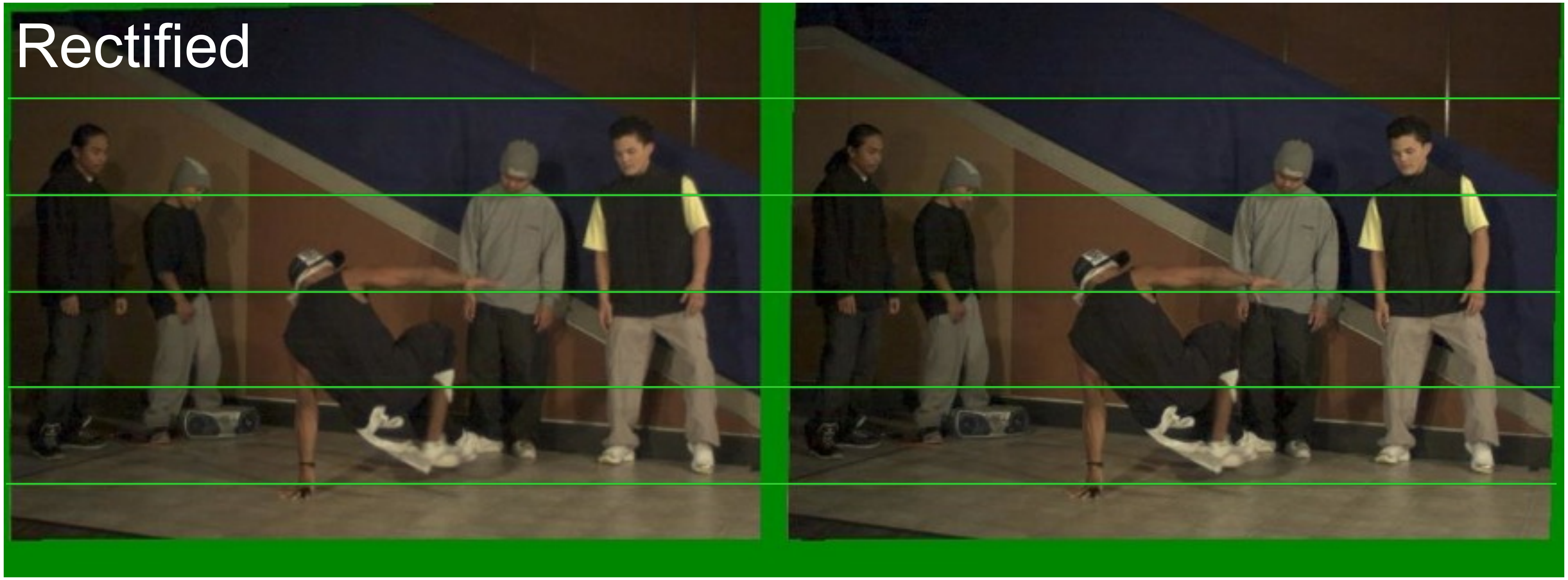
Unrectified


Unrectified


Rectified

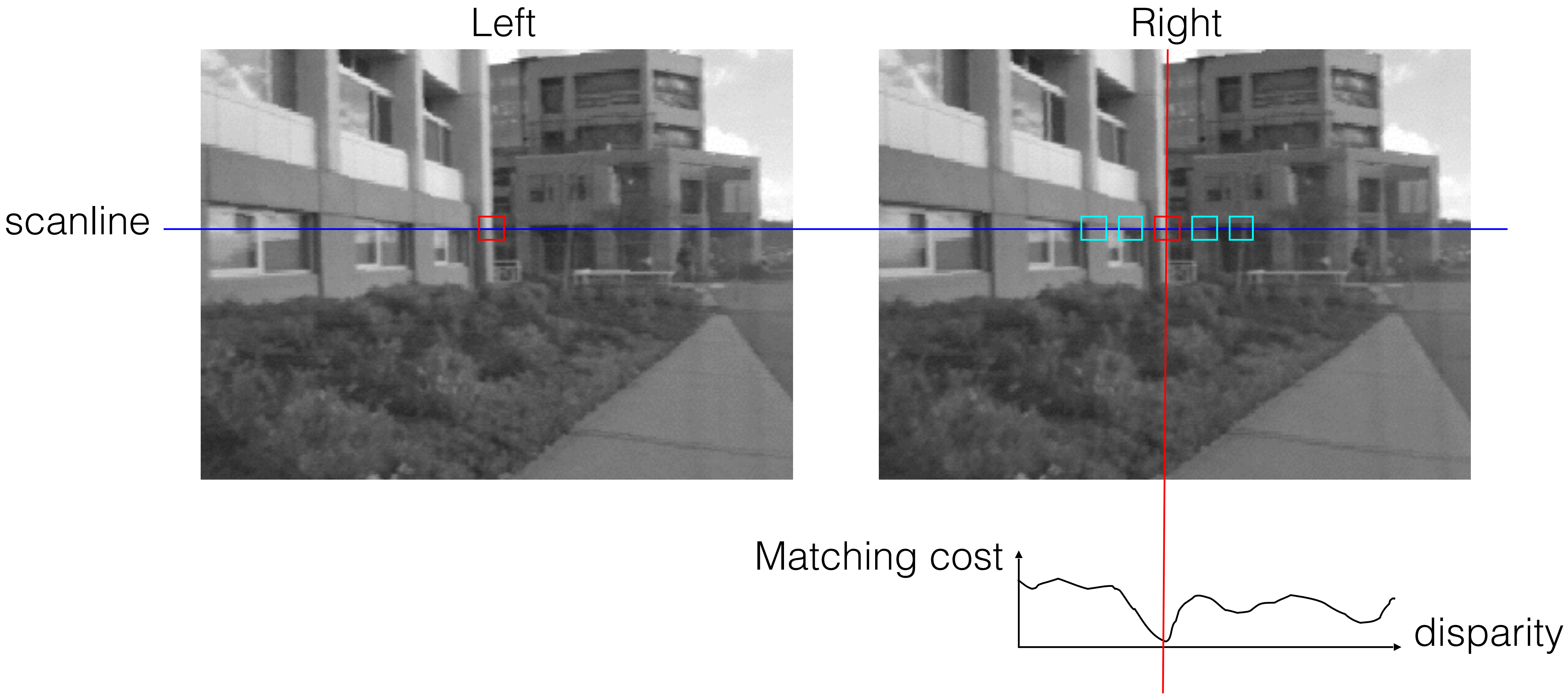


Unrectified

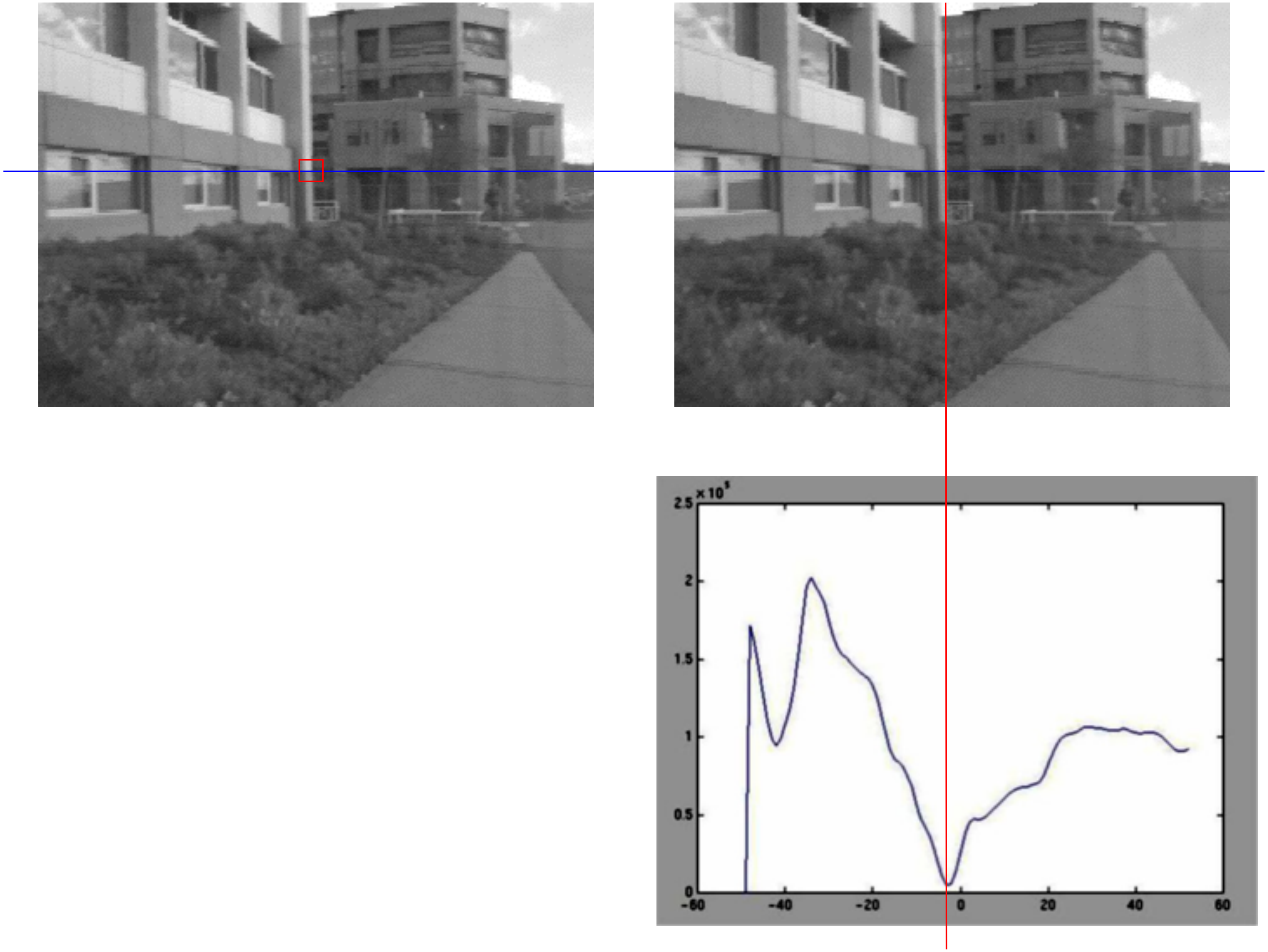


Rectified

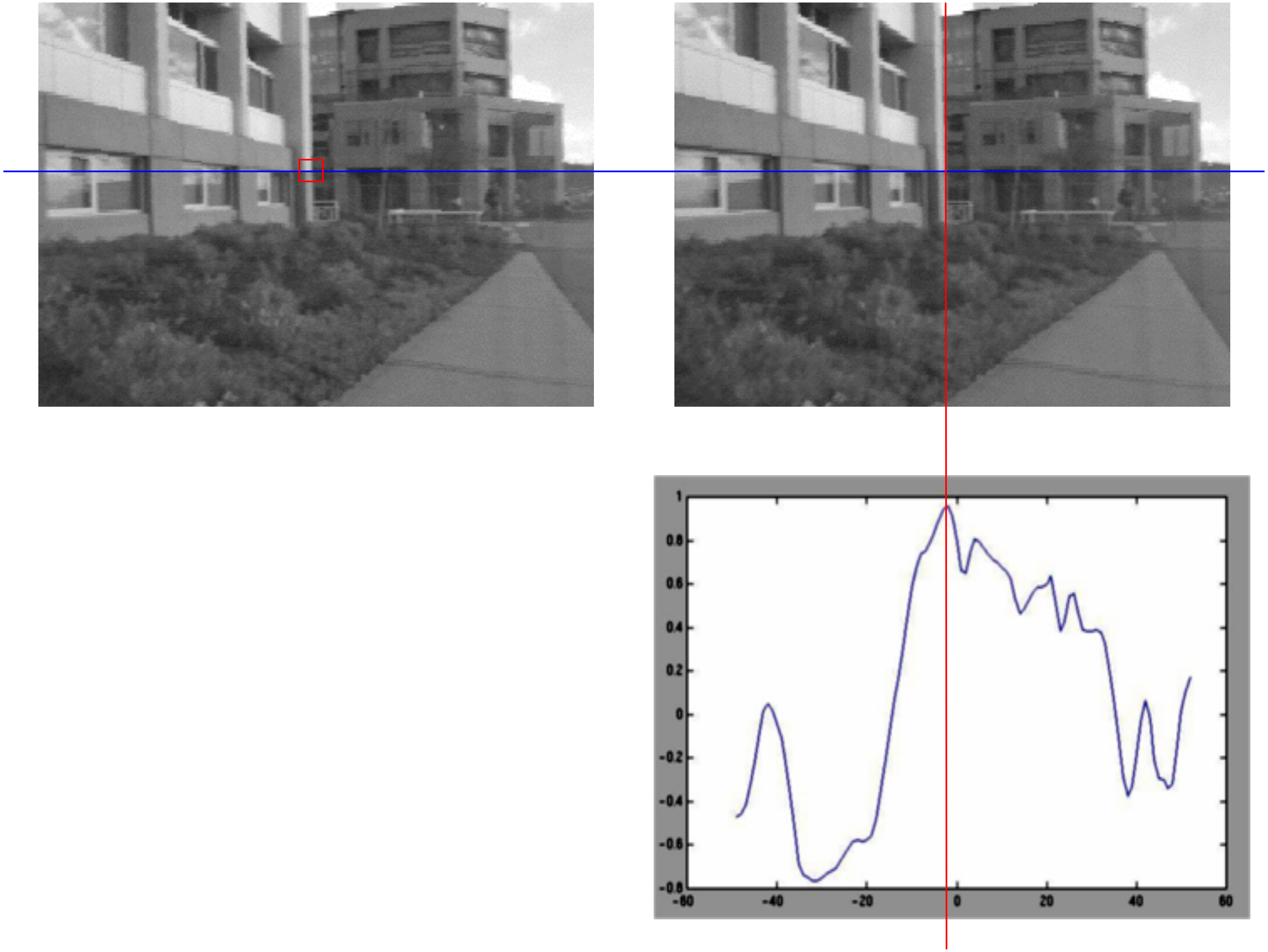
Finding the best match



- 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

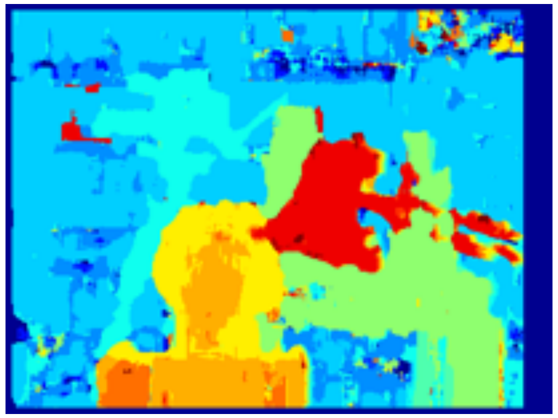


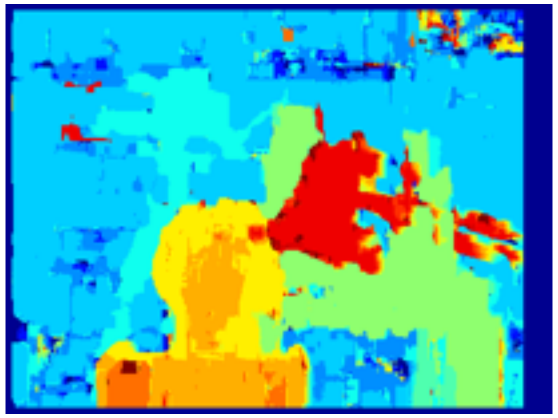
SSD

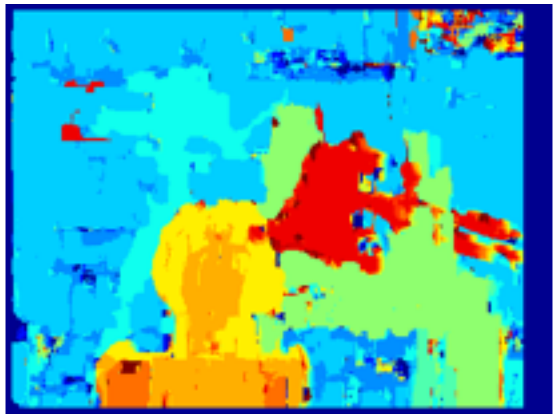


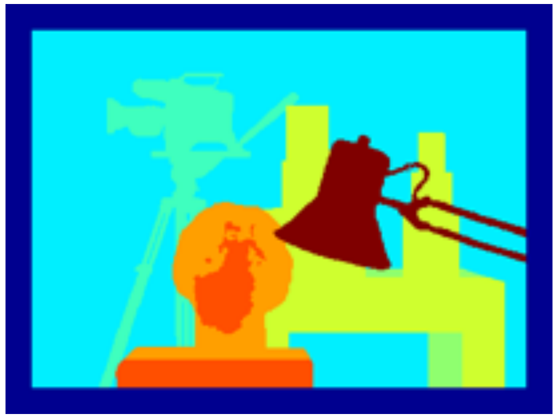
Normalized cross-correlation

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)	$\frac{\sum_{(i,j)\in W} I_1(i,j).I_2(x+i,y+j)}{\sqrt[2]{\sum_{(i,j)\in W} I_1^2(i,j).\sum_{(i,j)\in W} I_2^2(x+i,y+j)}}$

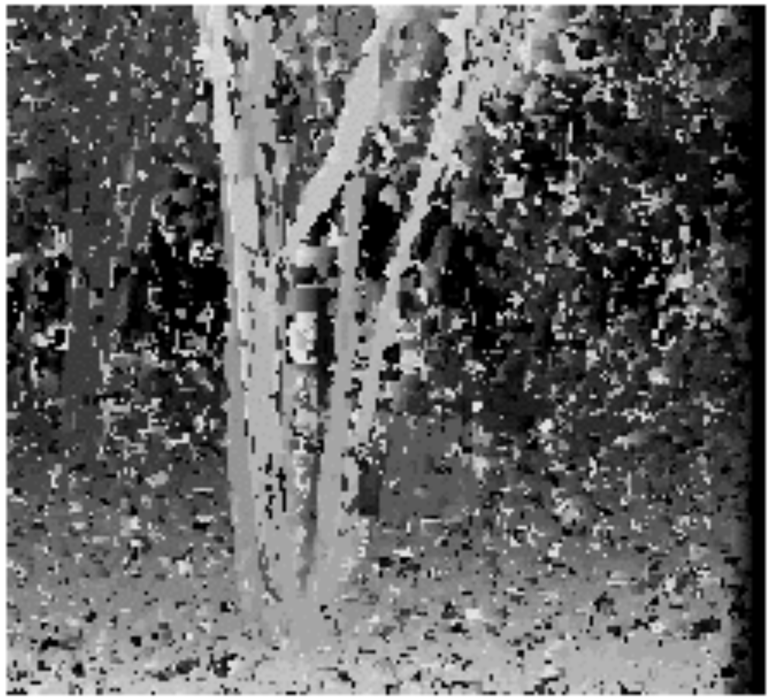

SAD


SSD

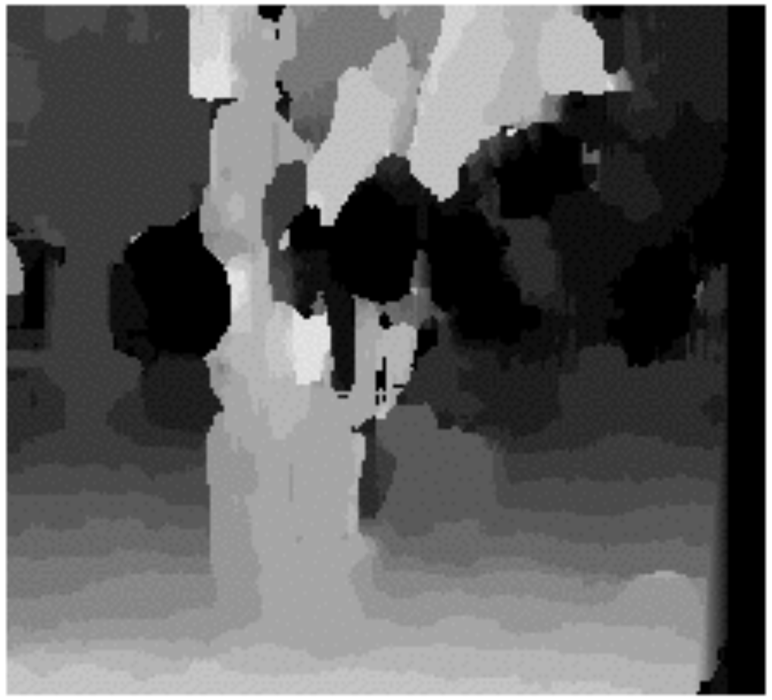

NCC


Ground truth

Effect of window size

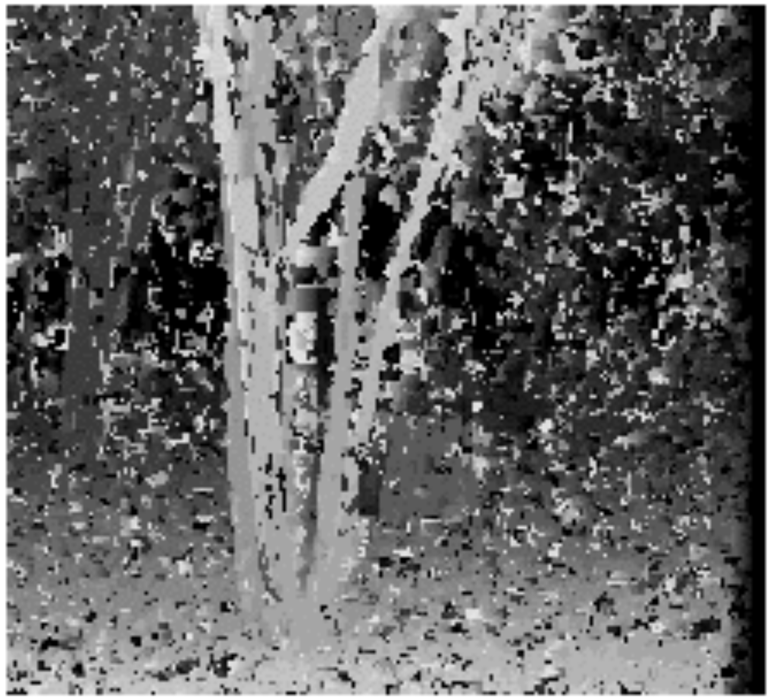


W = 3

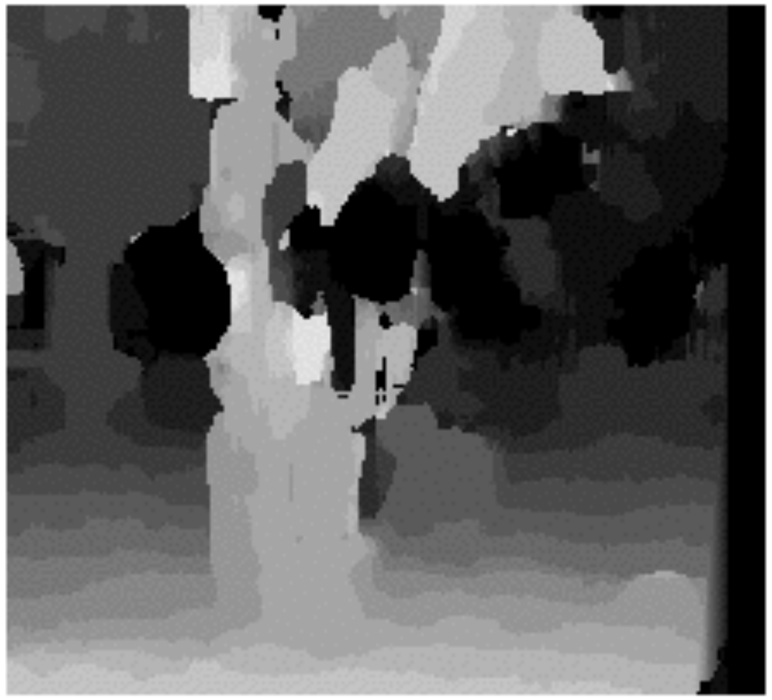


W = 20

Effect of window size



W = 3



W = 20

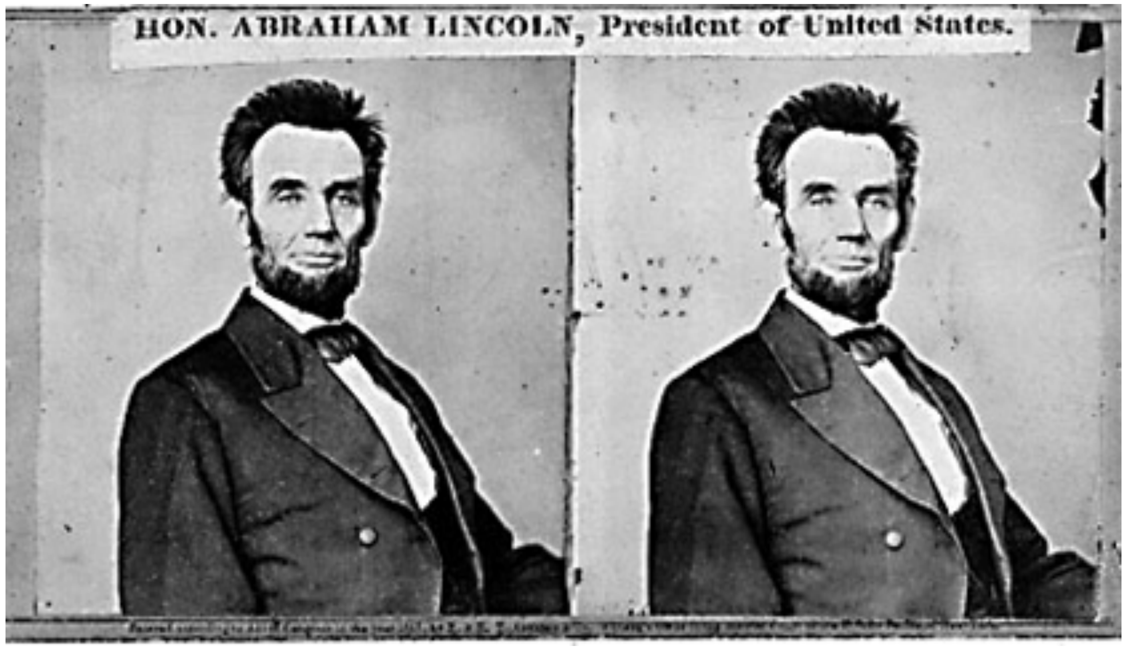
Smaller window

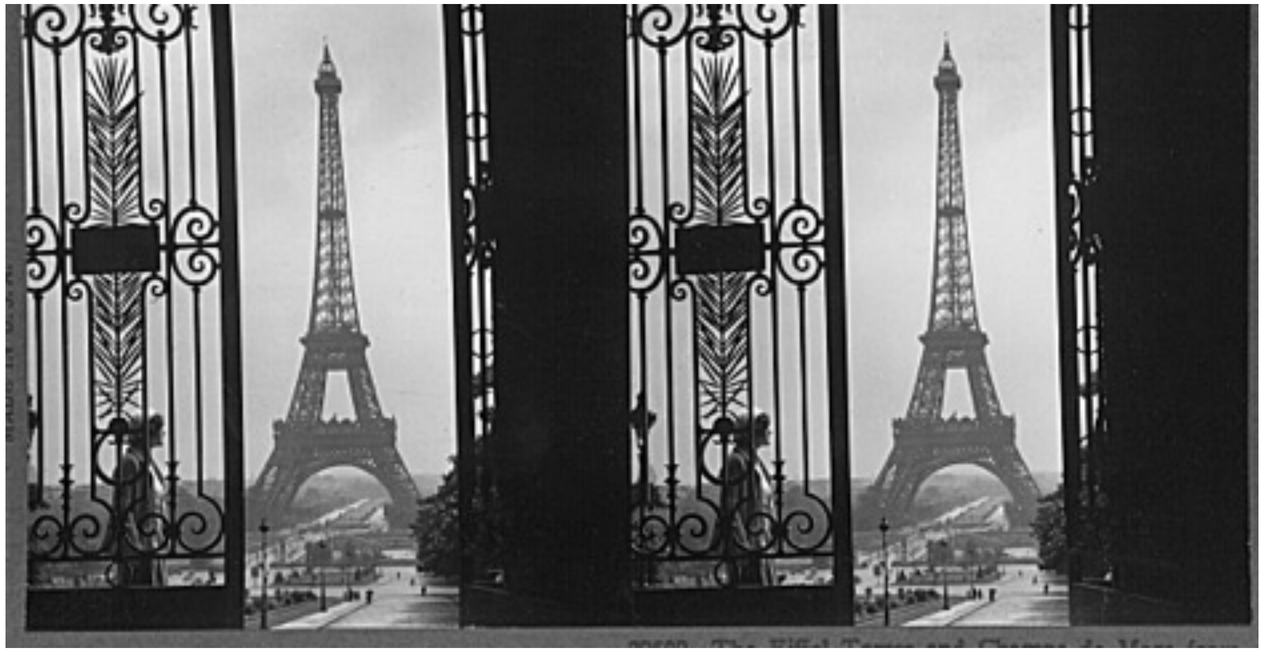
- + More detail
- - More noise

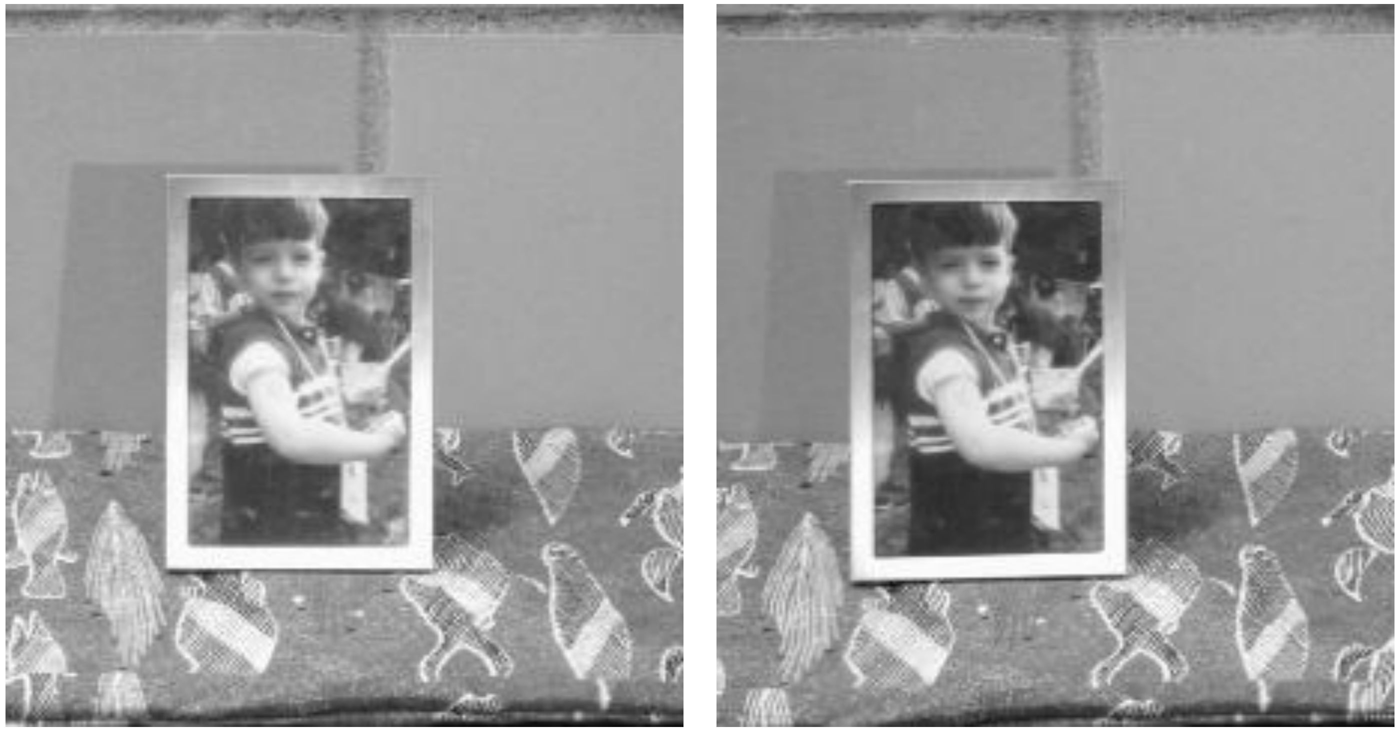
Larger window

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

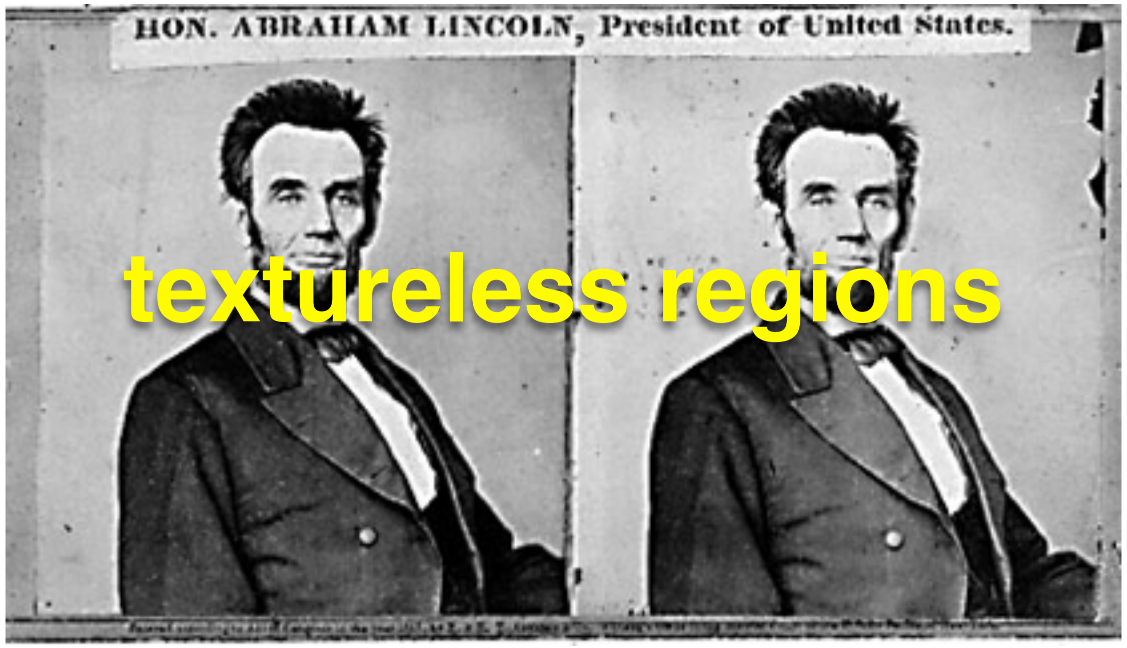
When will stereo block matching fail?

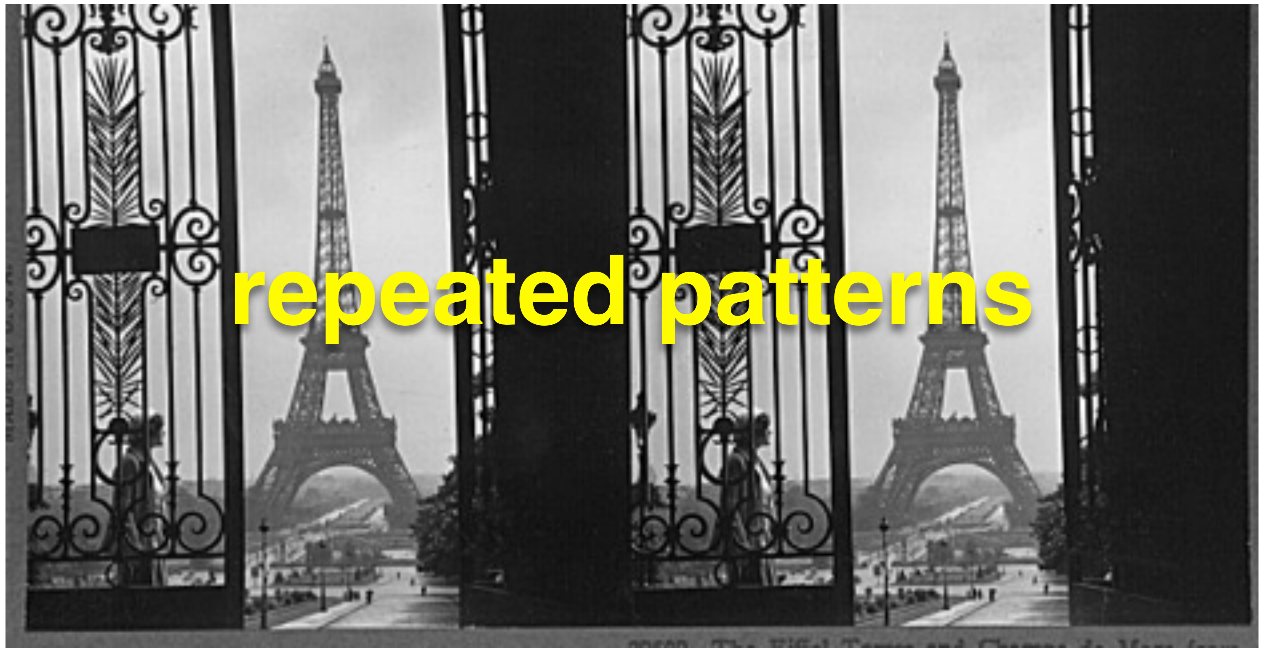


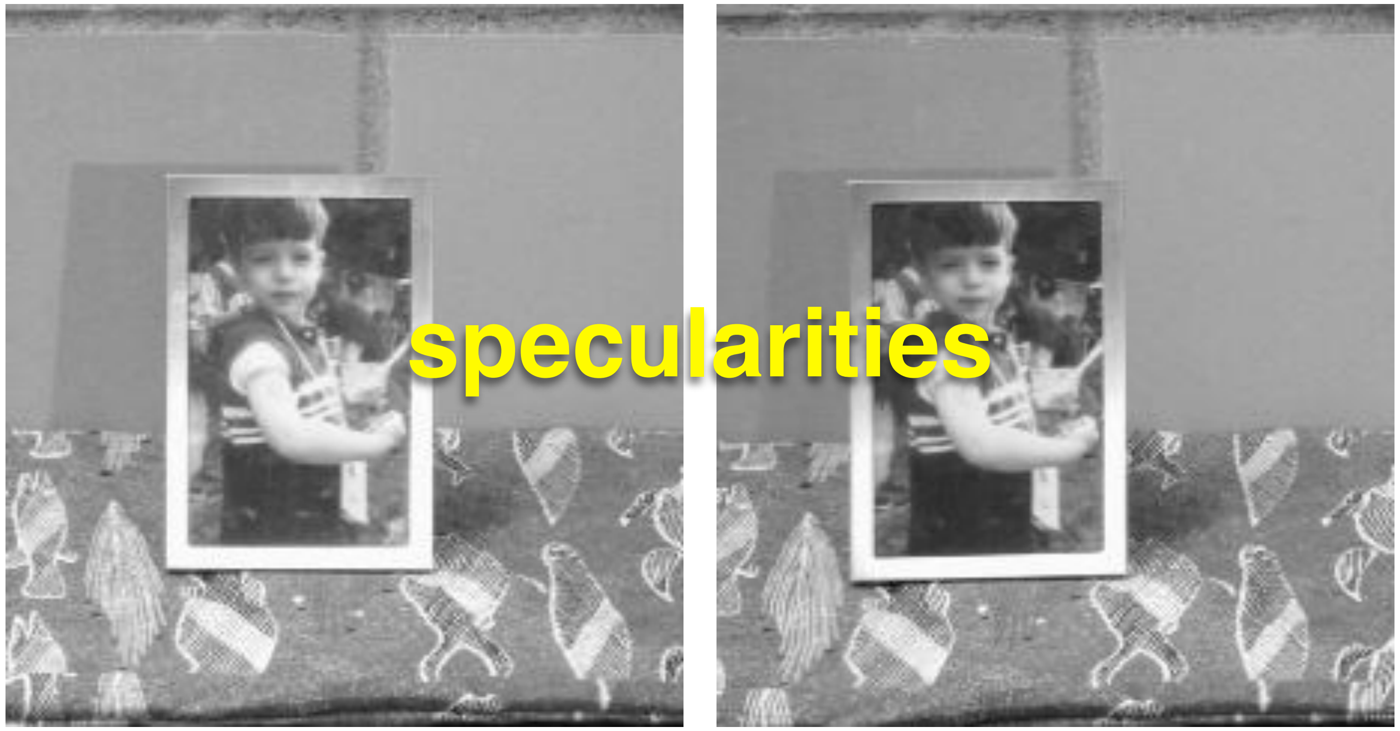




When will stereo block matching fail?

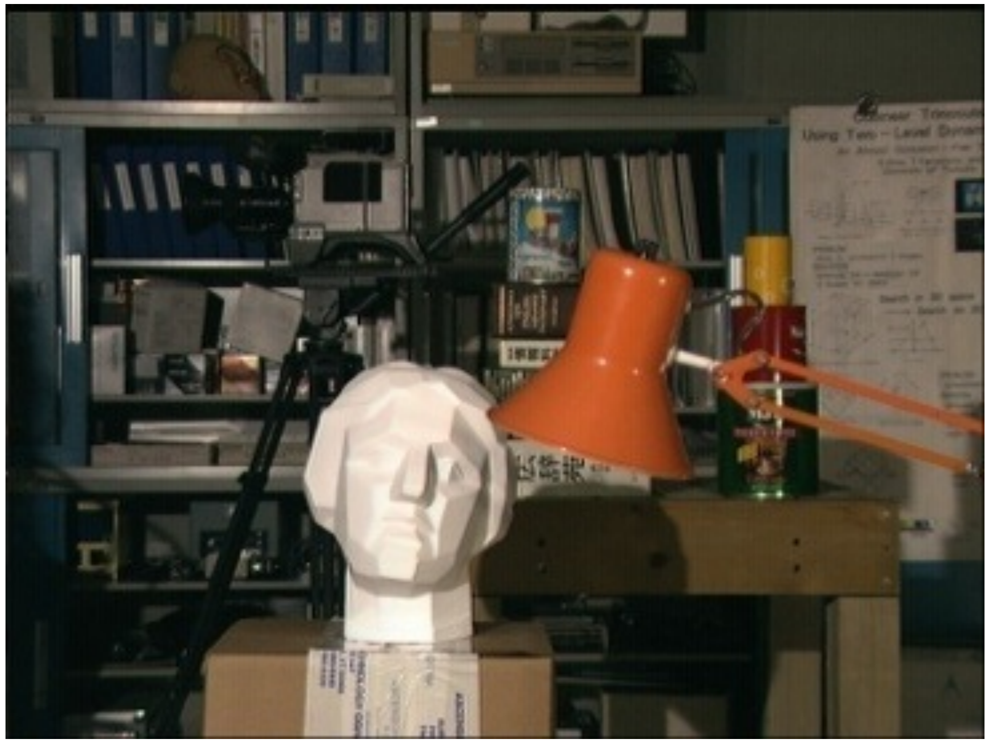




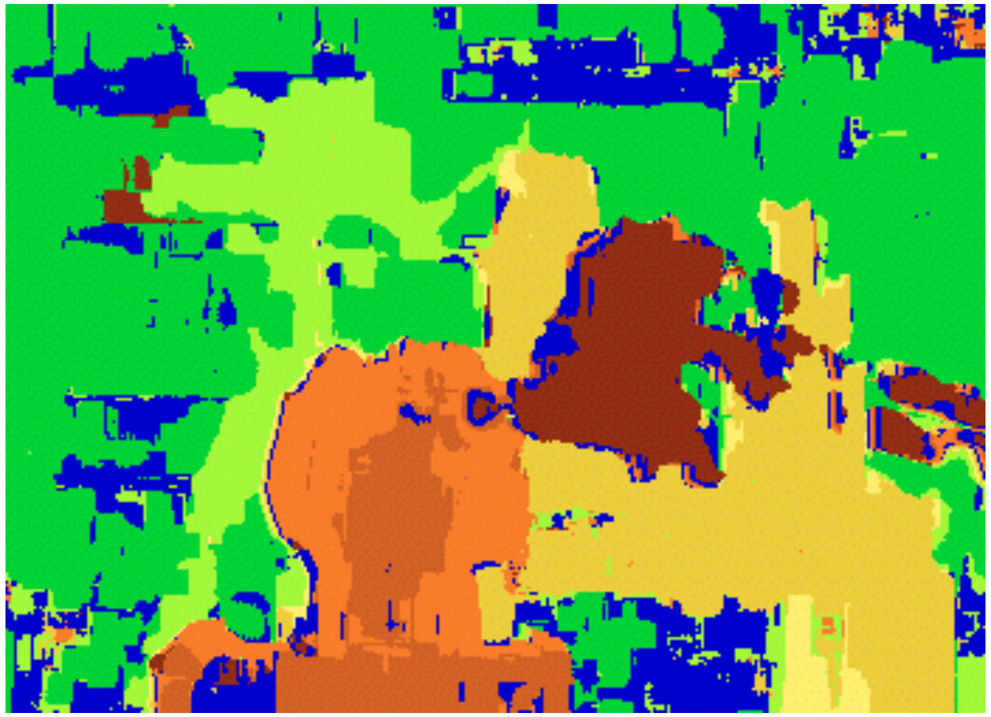


(break)

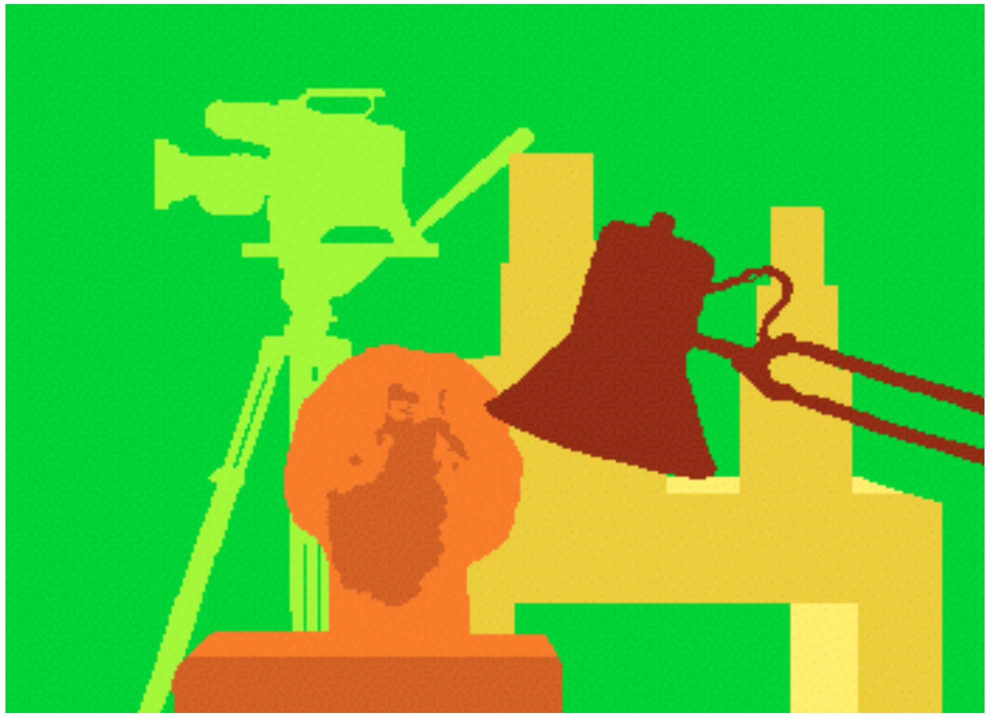
Improving Stereo Block Matching



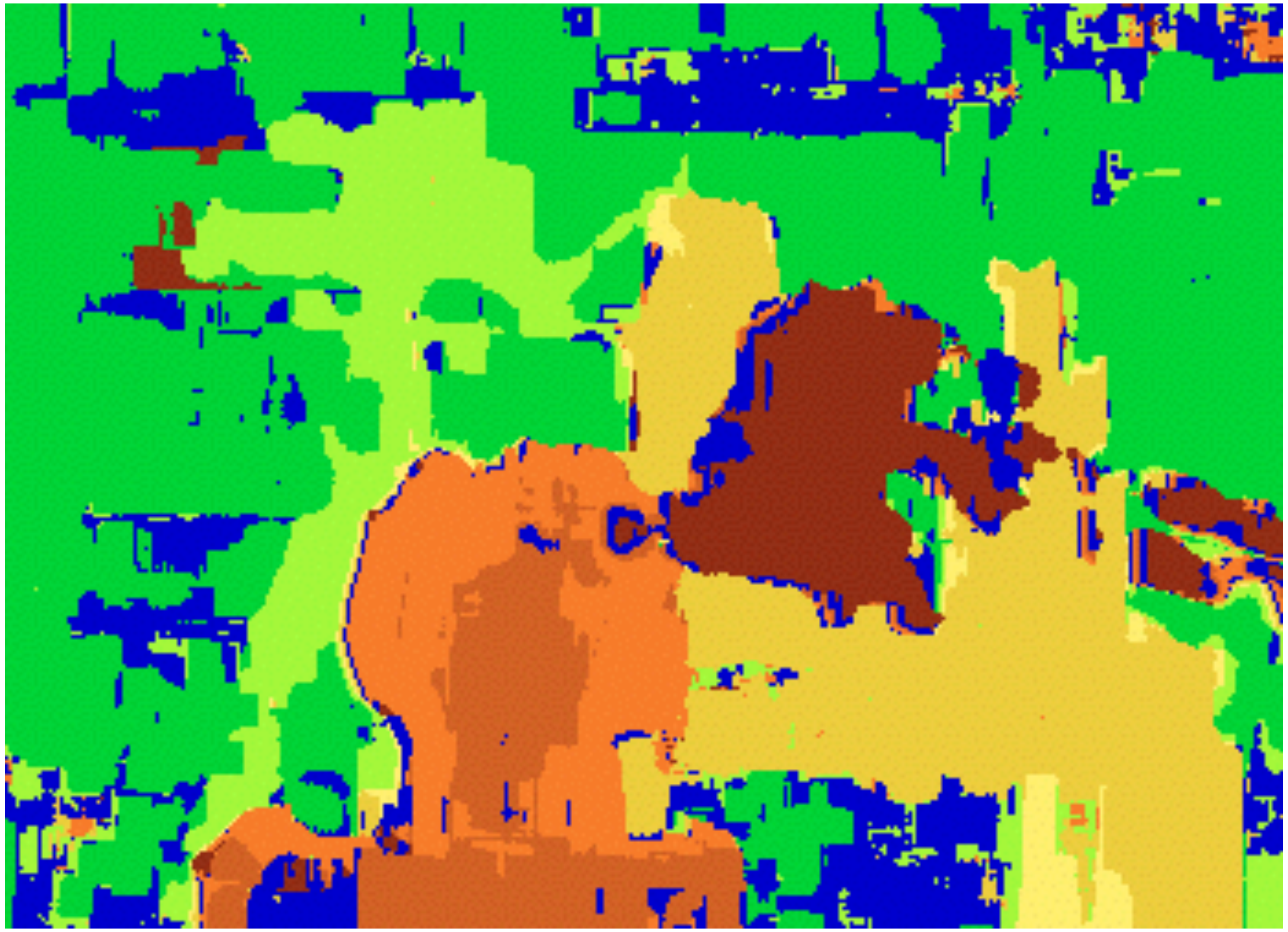
Block matching



Ground truth



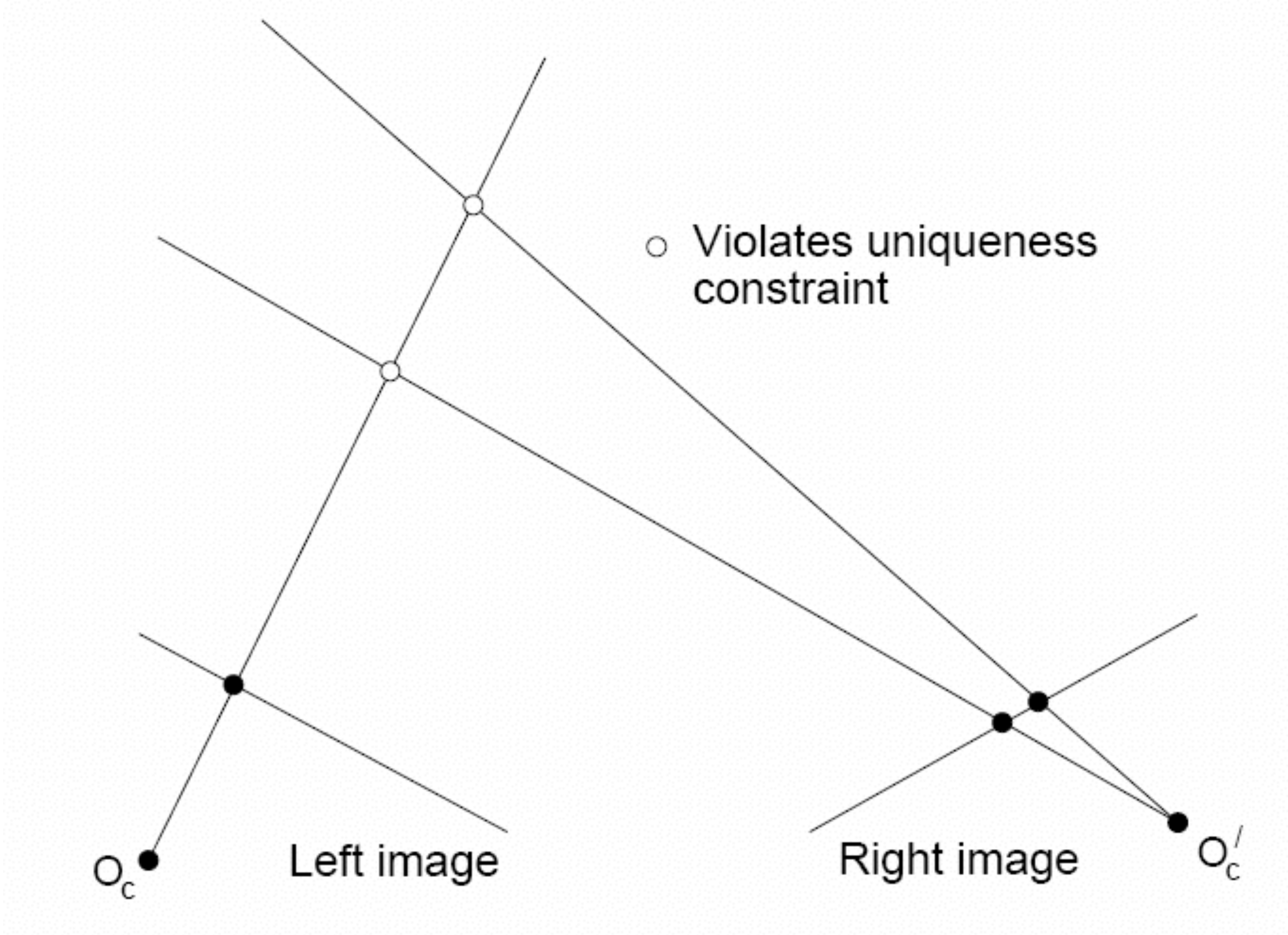
What are some problems with the result?



How can we improve depth estimation?

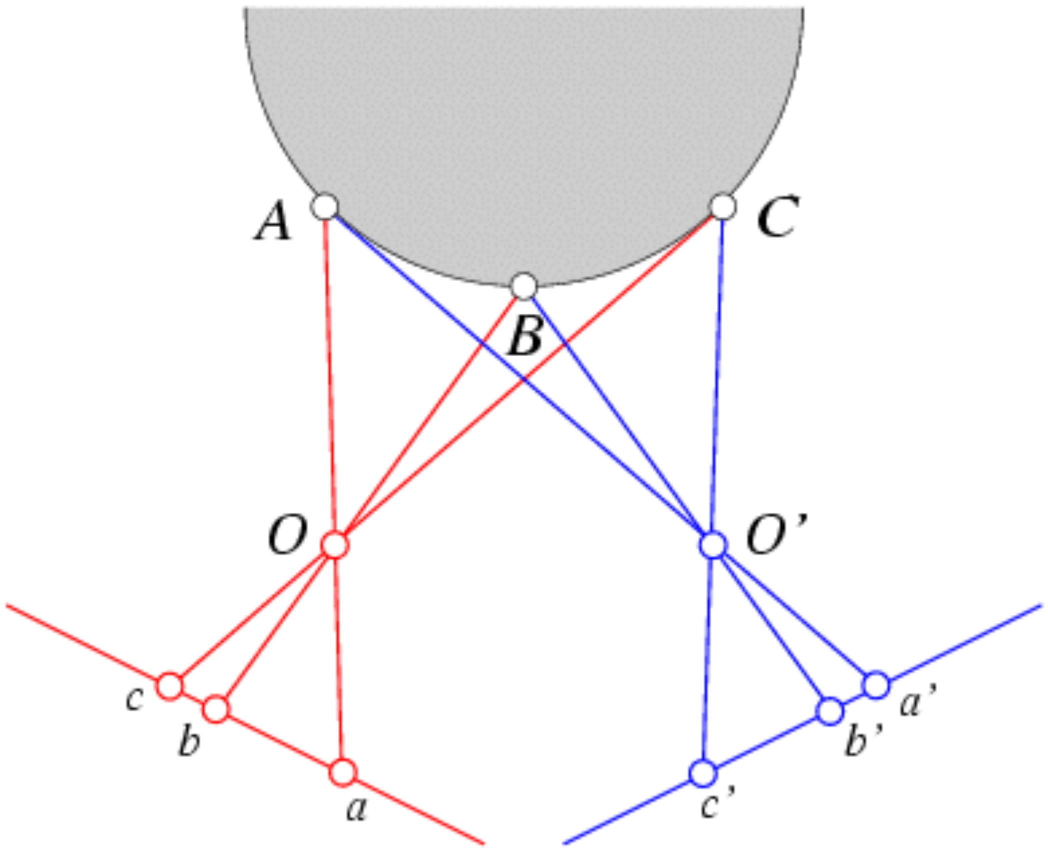
Uniqueness

For any point in one image, there should be at most one matching point in the other image



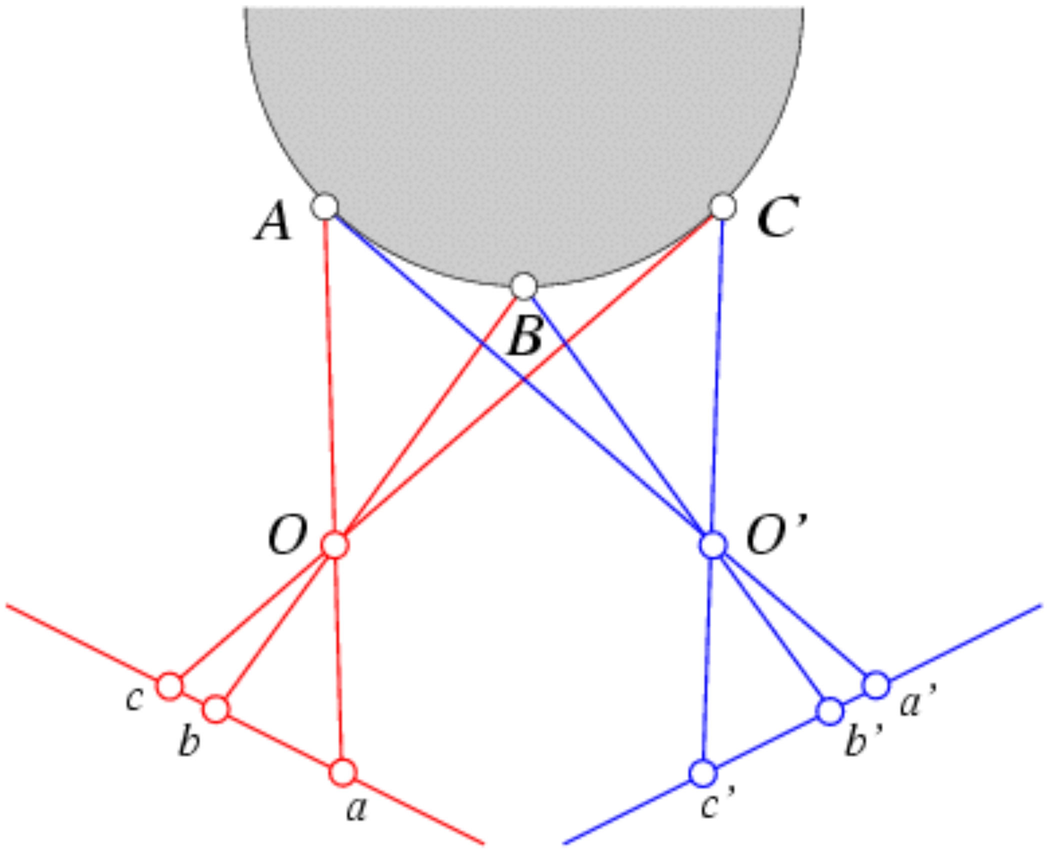
Ordering

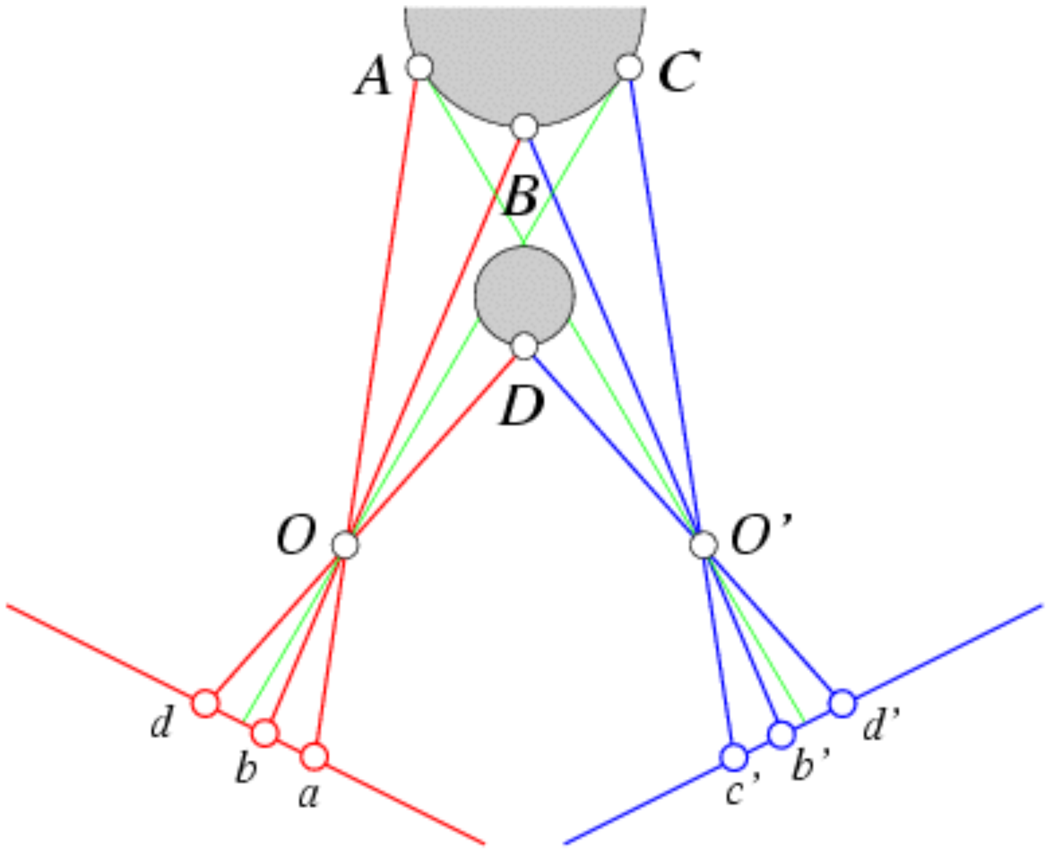
Corresponding points should be in the same order in both views



Ordering

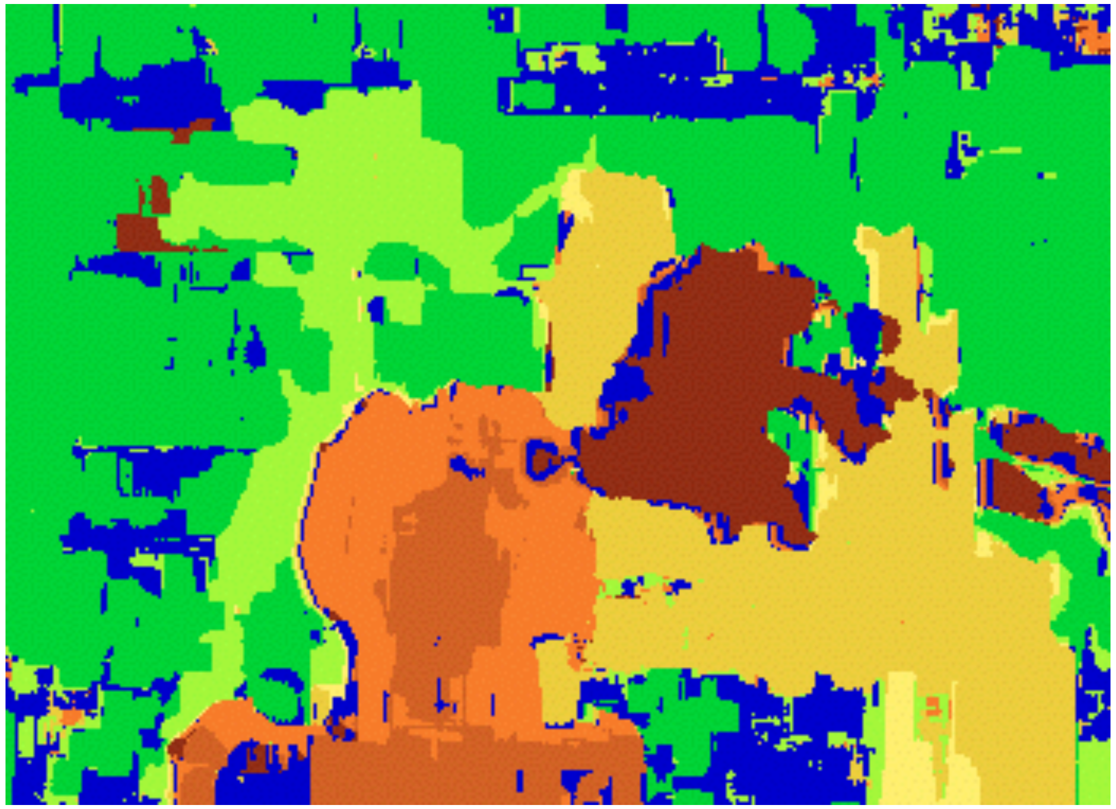
Corresponding points should be in the same order in both views



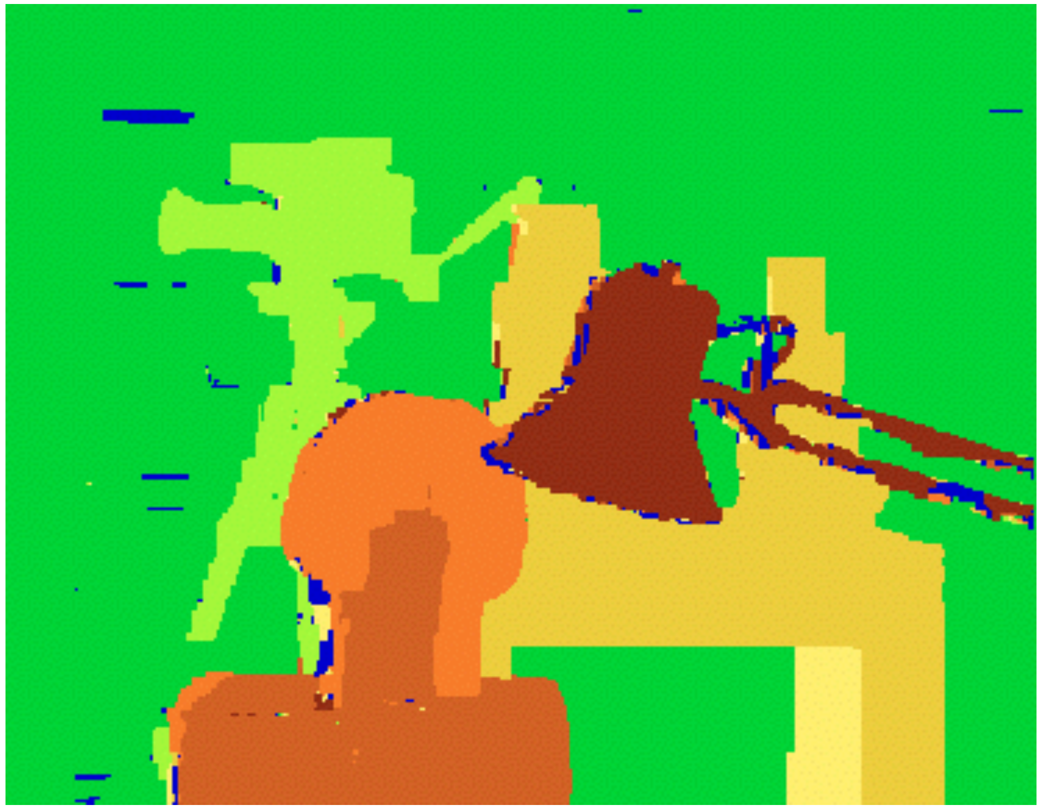


Smoothness

We expect disparity values to change slowly (for the most part)



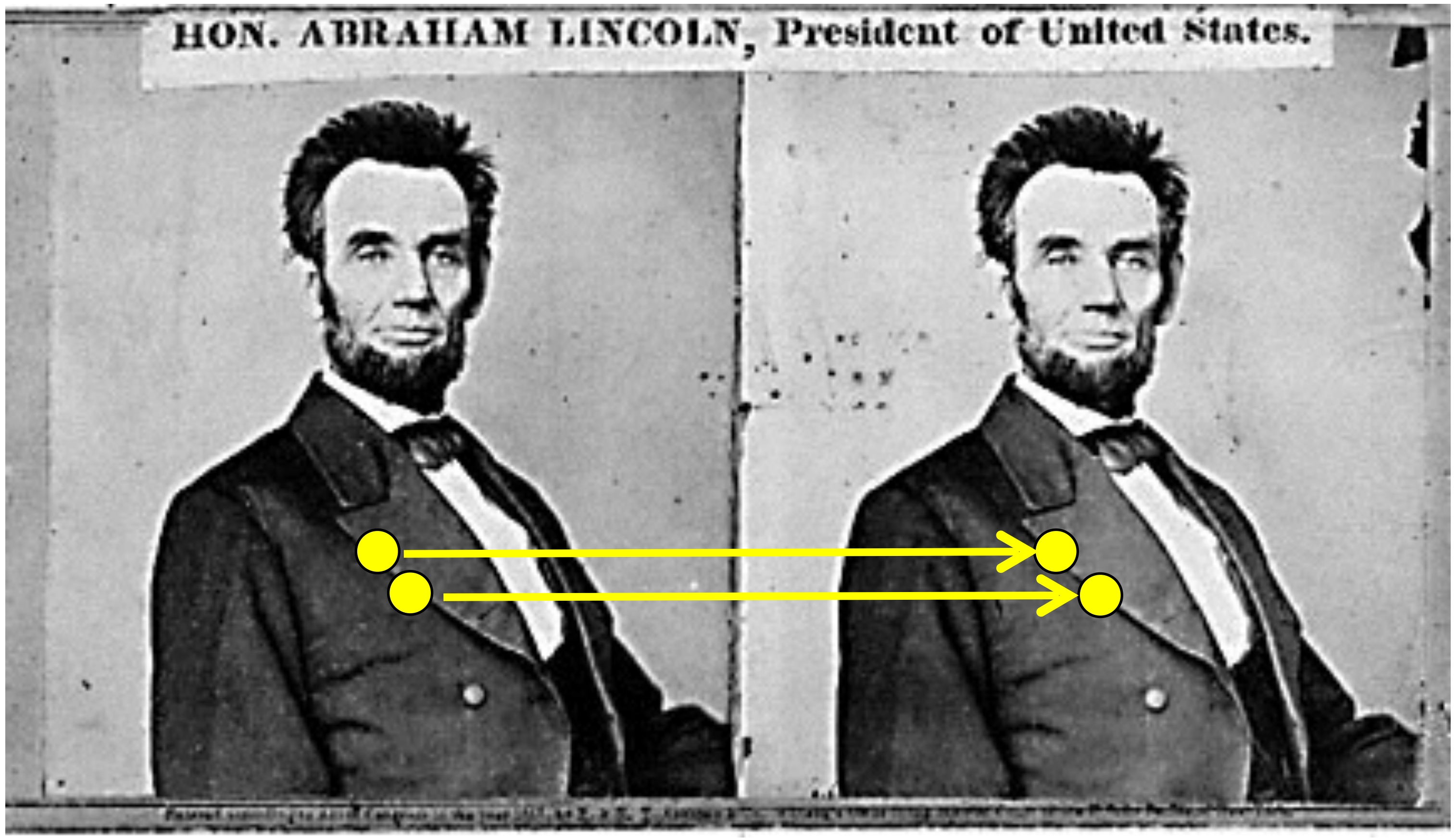
Too many discontinuities



Better

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

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

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

(match cost)

Adjacent pixels should (usually) move about the same amount

(smoothness cost)

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

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

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

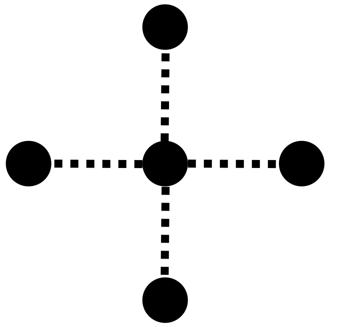
$$E(d) = E_d(d) + \lambda E_s(d)$$

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

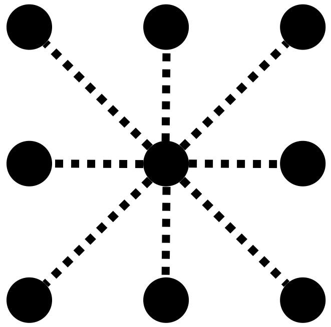
SSD distance between windows centered at I(x, y) and J(x+ d(x,y), y)

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

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



4-connected neighborhood

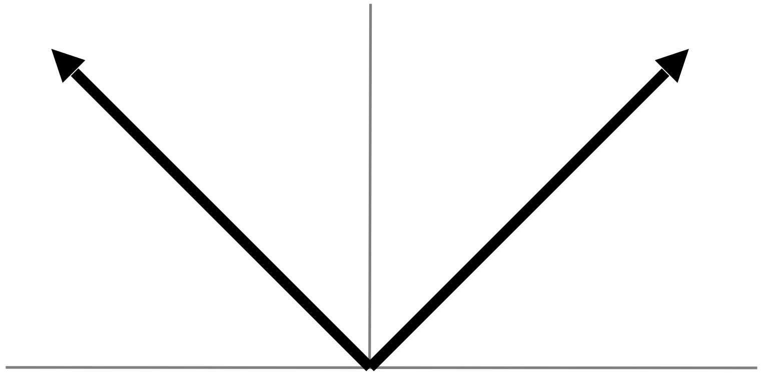


8-connected neighborhood

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

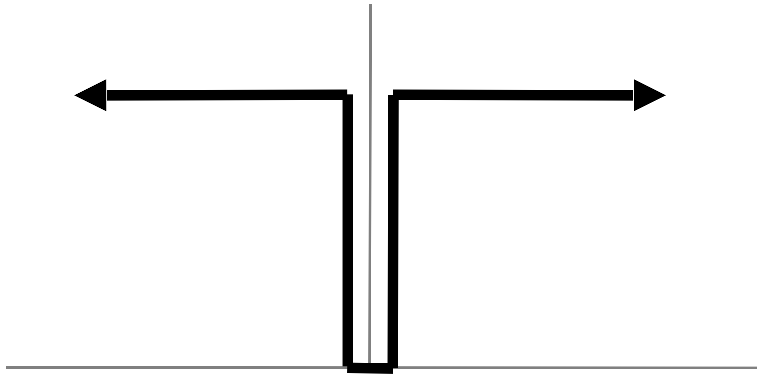
$$V(d_p, d_q) = |d_p - d_q|$$

L_1 distance



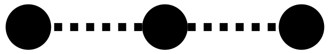
$$V(d_p, d_q) = \begin{cases} 0 & \text{if } d_p = d_q \\ 1 & \text{if } d_p \neq d_q \end{cases}$$

“Potts model”



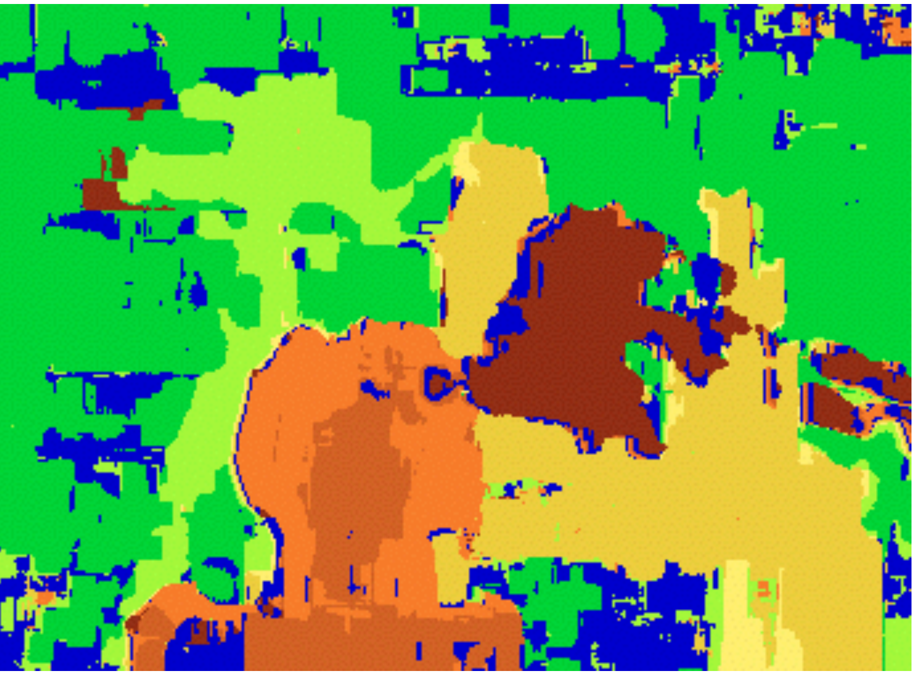
Dynamic Programming

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

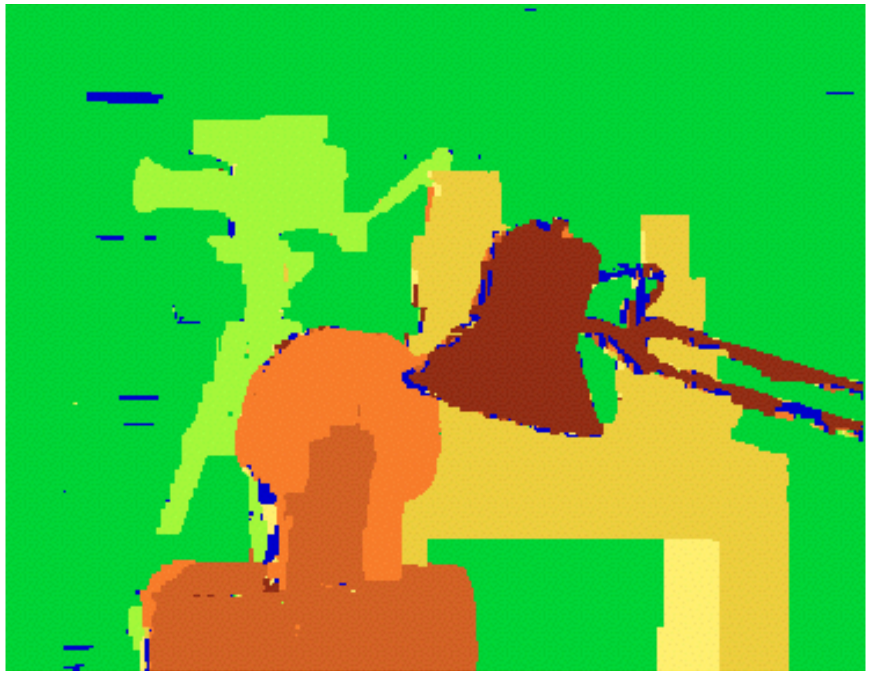
Can minimize this independently per scanline
using dynamic programming (DP) 

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

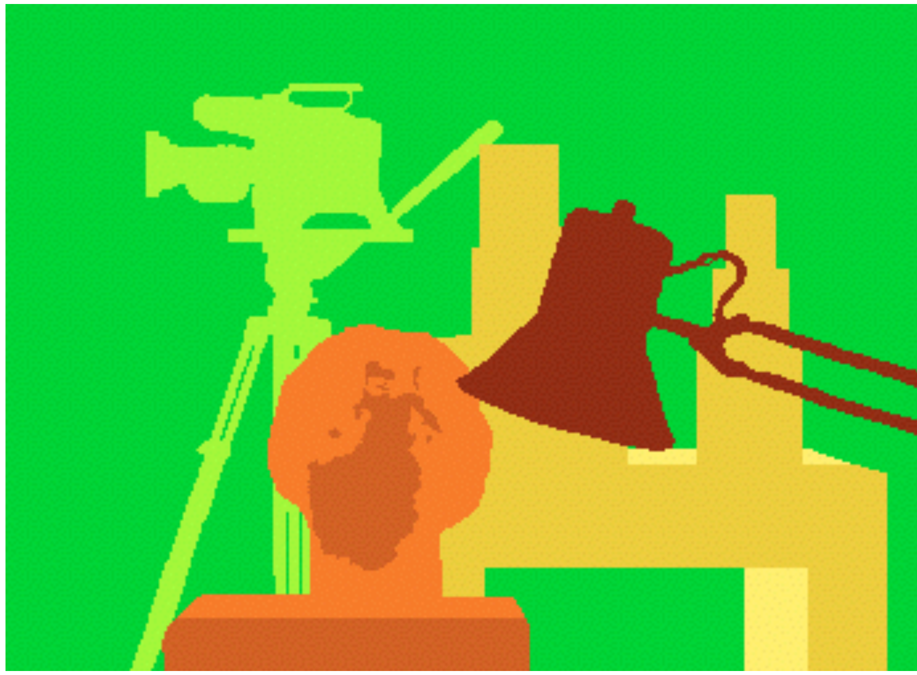
$$D(x, y, d) = C(x, y, d) + \min_{d'} \left\{ D(x-1, y, d') + \lambda \left| d - d' \right| \right\}$$



Match only



Match & smoothness
(graph cuts)



Ground Truth

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