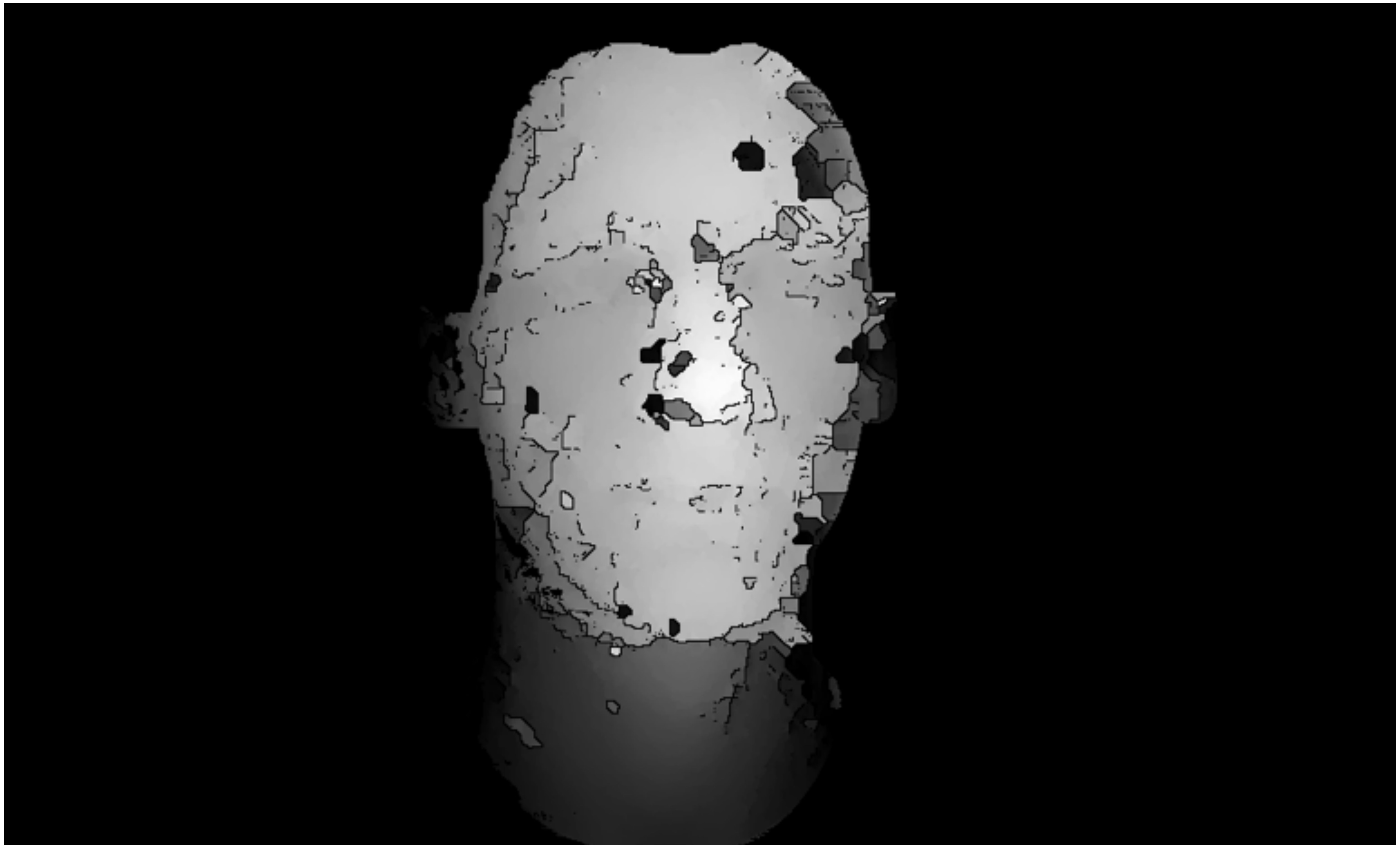


# Stereo Vision


16-385 Computer Vision (Kris Kitani)
**Carnegie Mellon University**

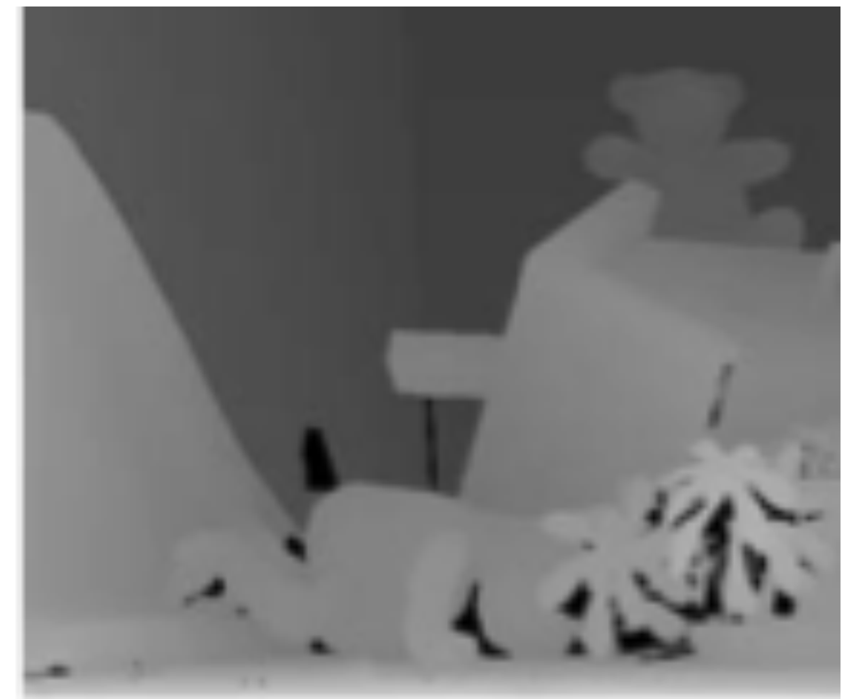
 

*What's different between these two images?*

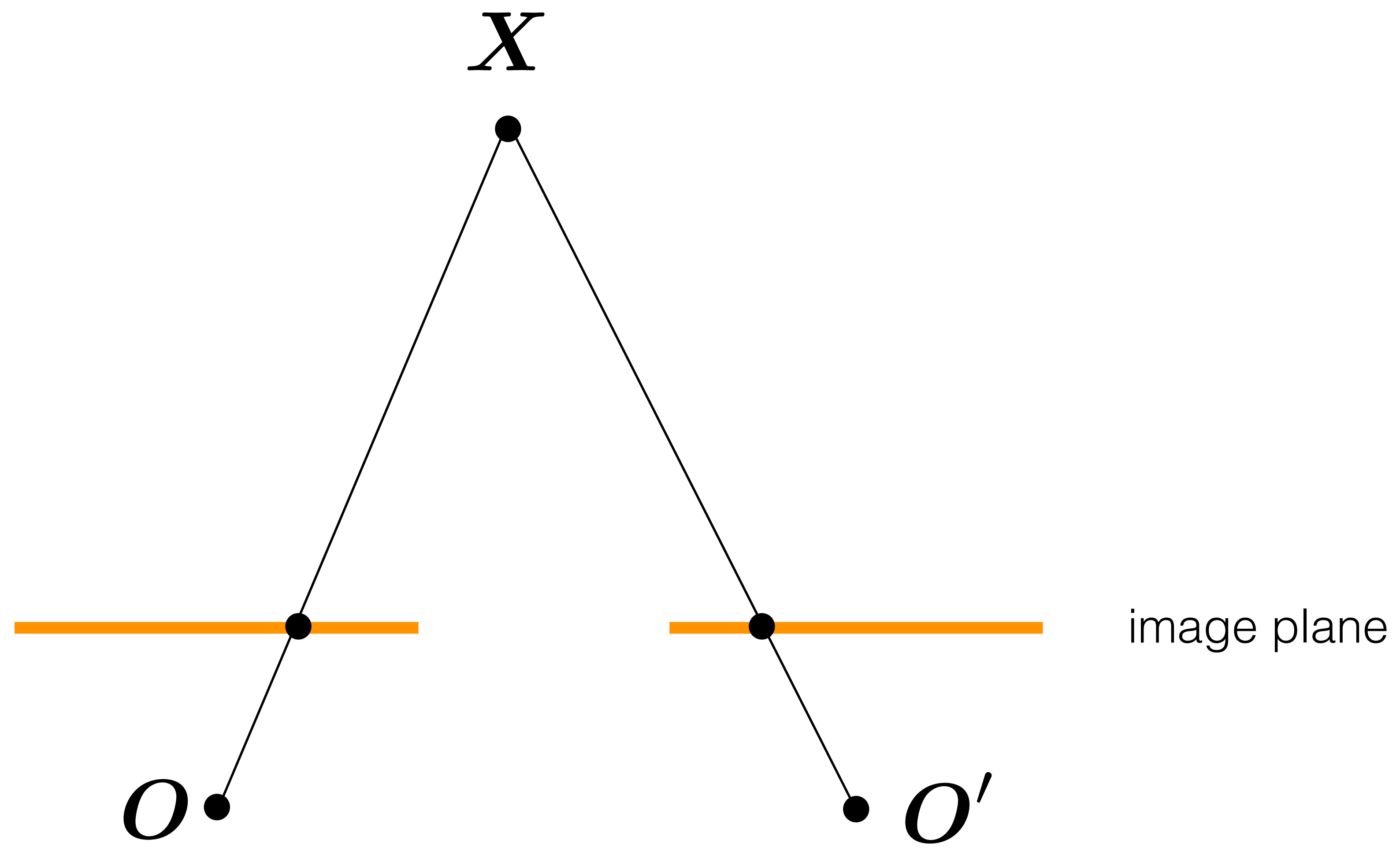
*Objects that are close move more or less?*

The amount of horizontal movement is inversely proportional to …

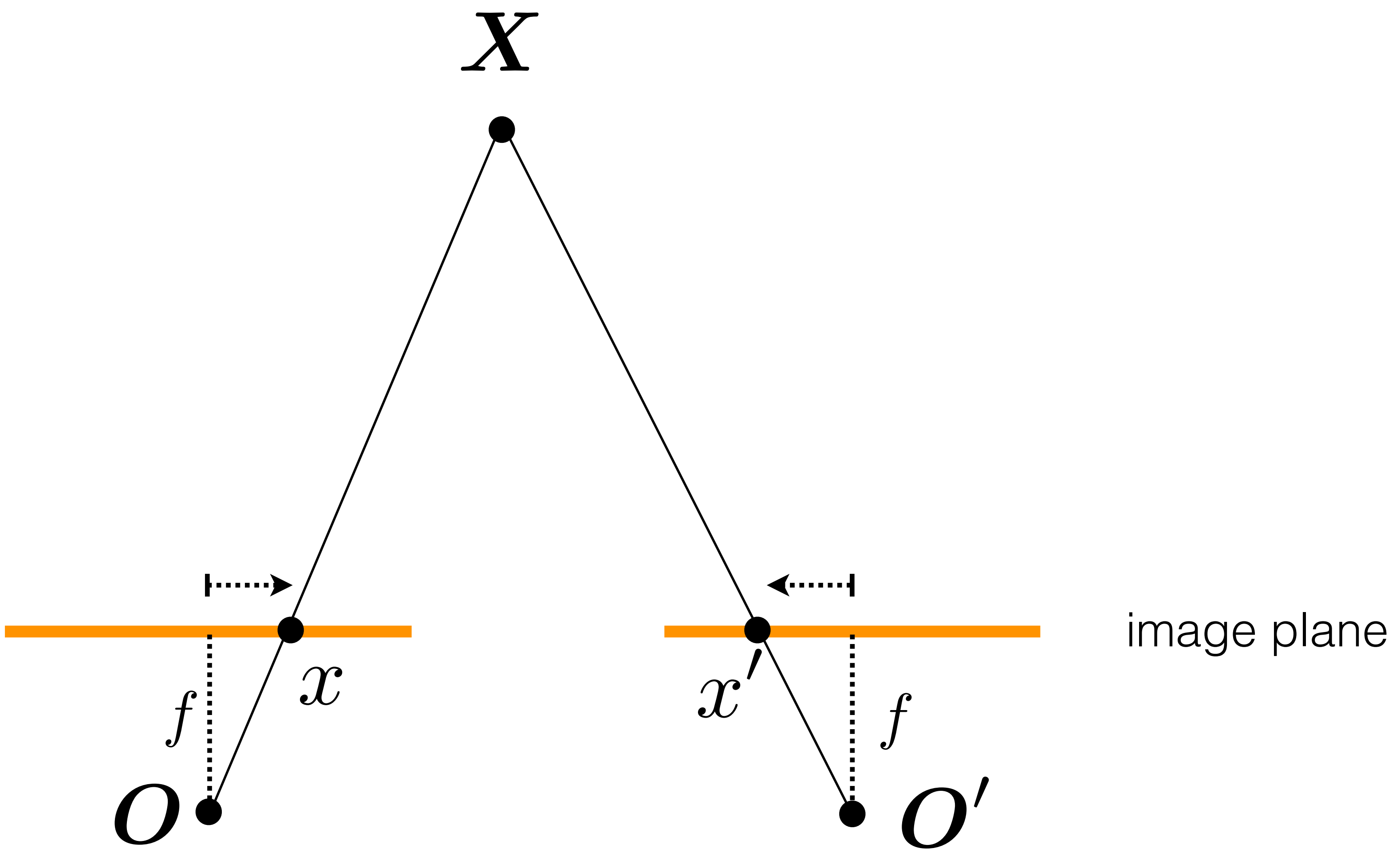


The amount of horizontal movement is inversely proportional to …

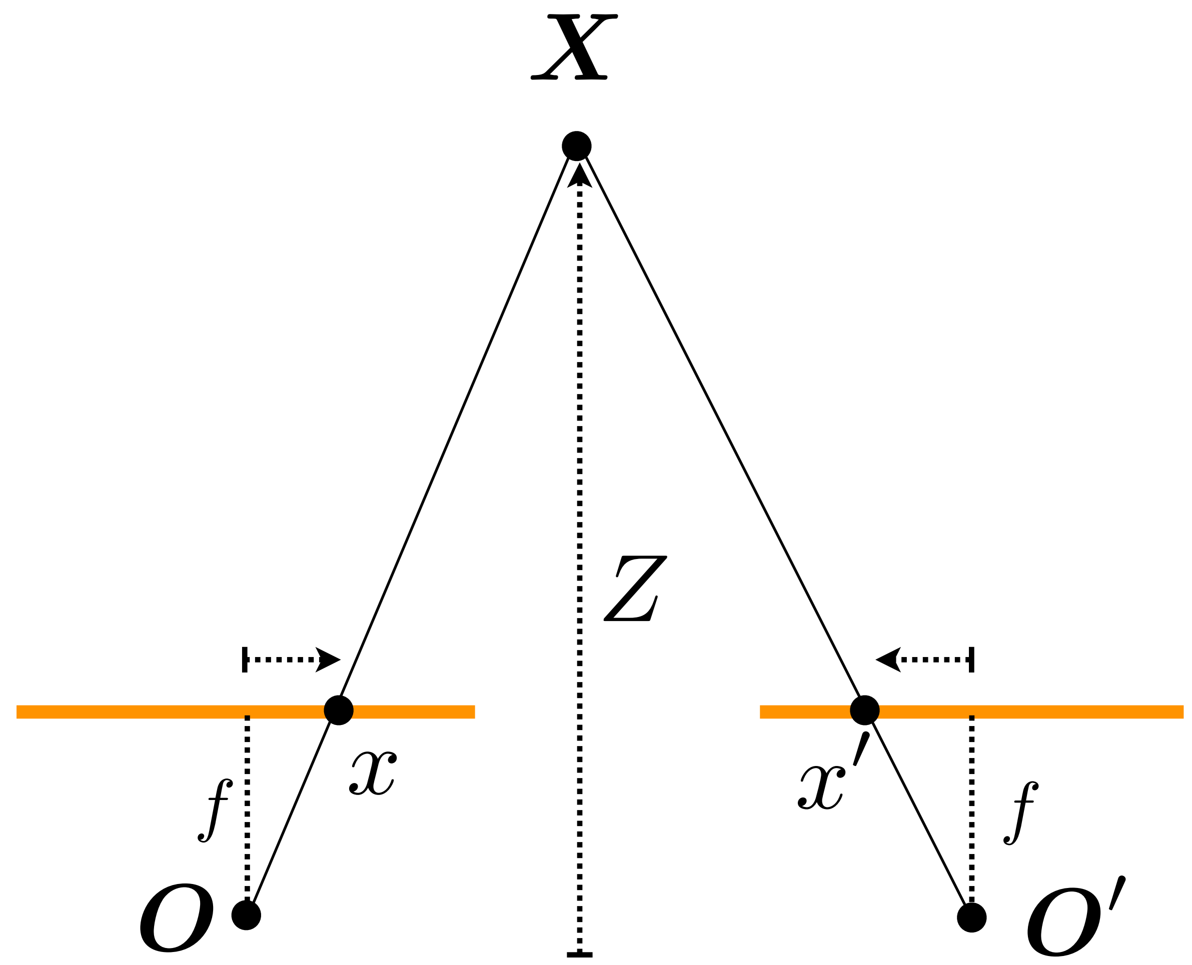
… the distance from the camera.



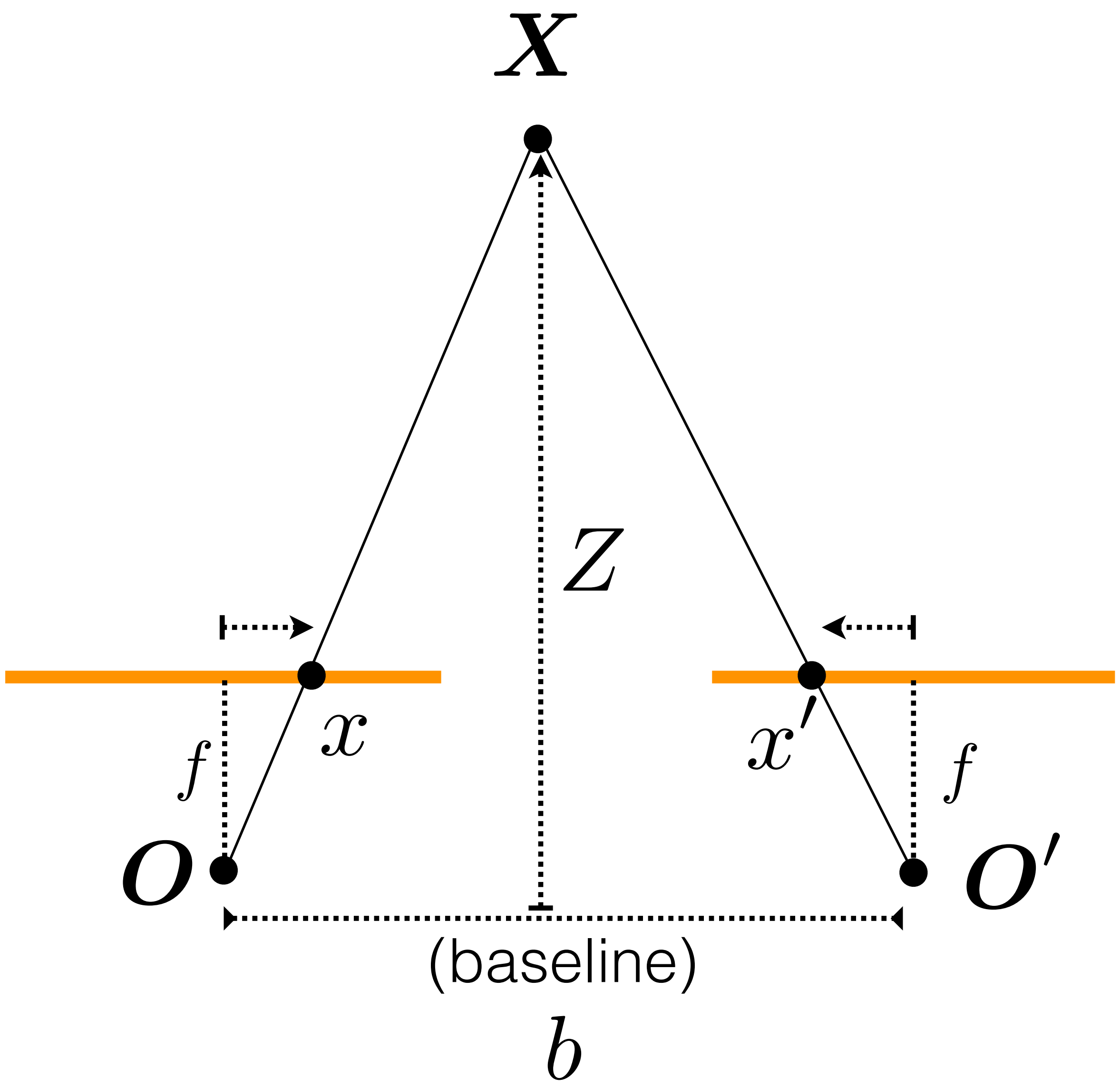
$X$
image plane
$O$
$O'$



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

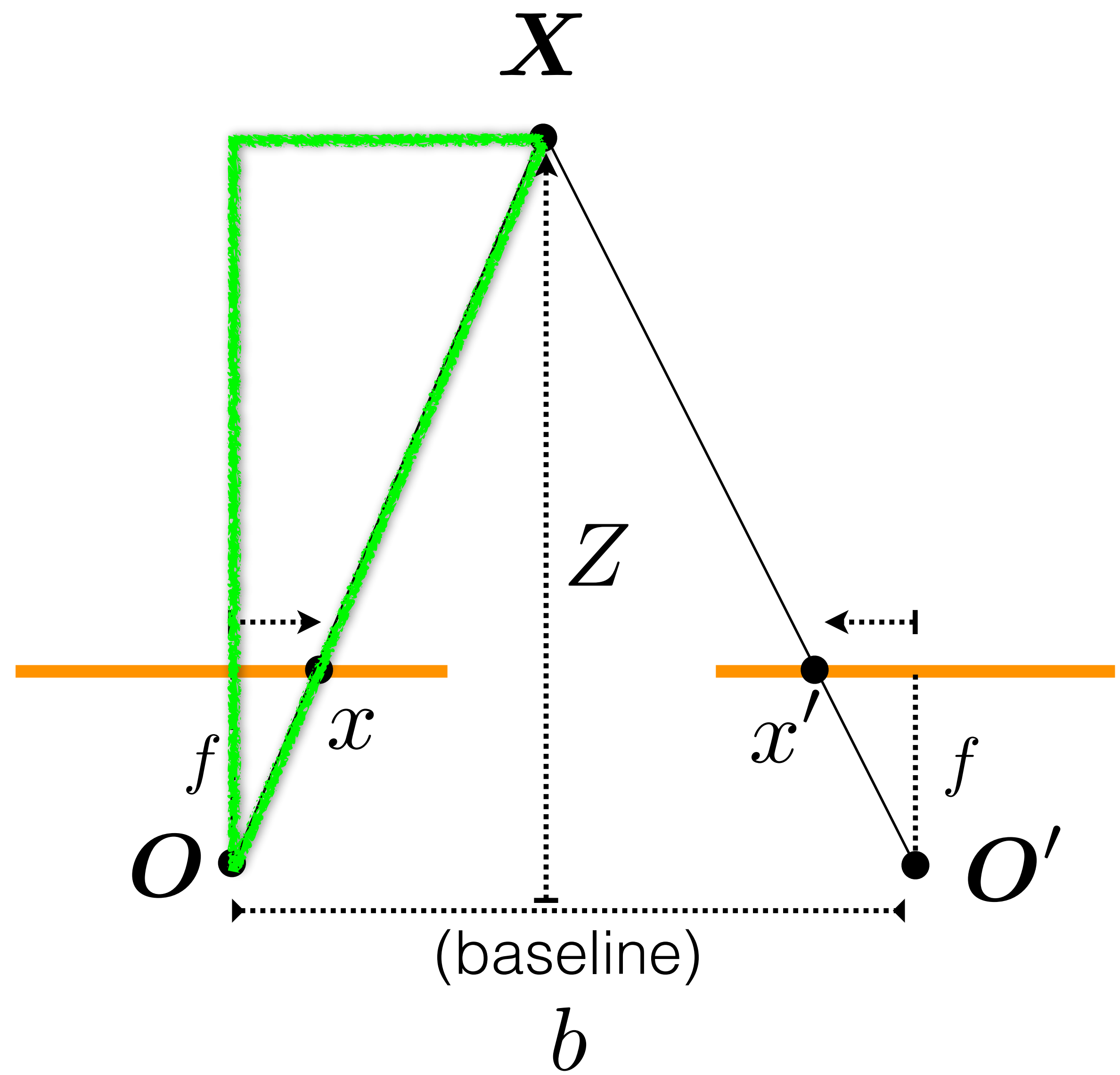


X
Z
x
x'
f
f
O
O'

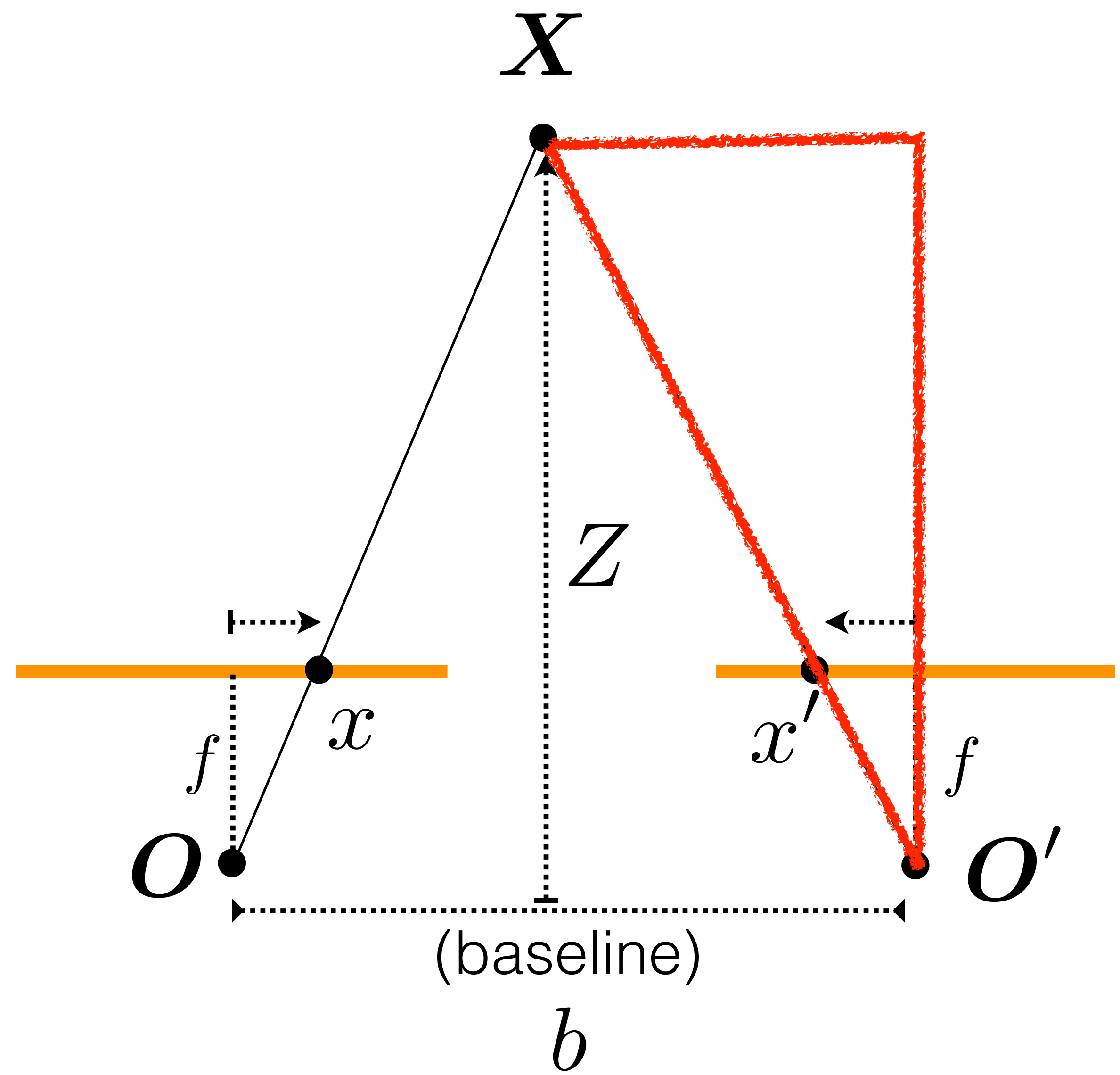


X
Z
x
x'
f
f
O
O'
(baseline)
b

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

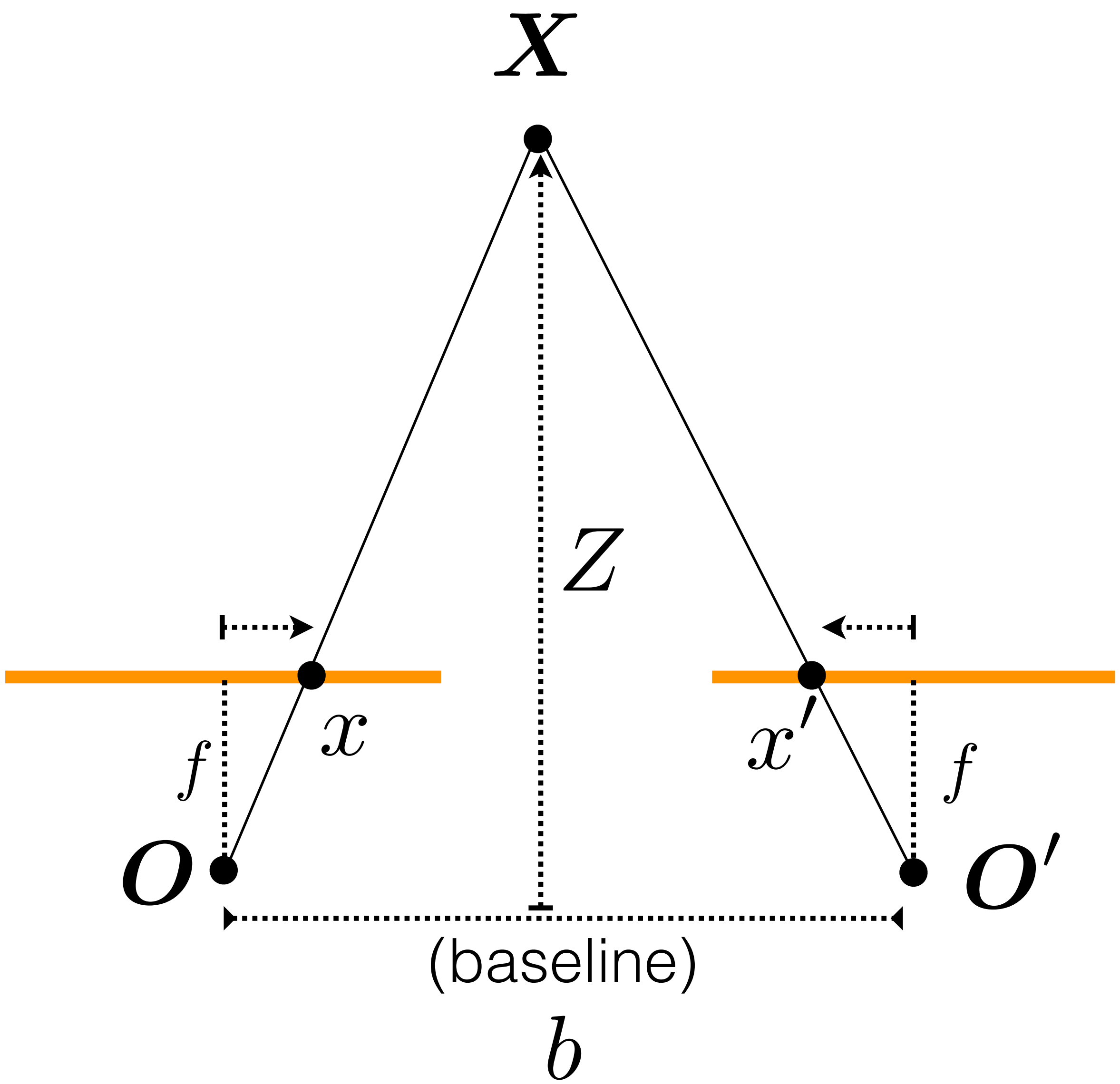


$$\frac{X}{Z} = \frac{x}{f}$$




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

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



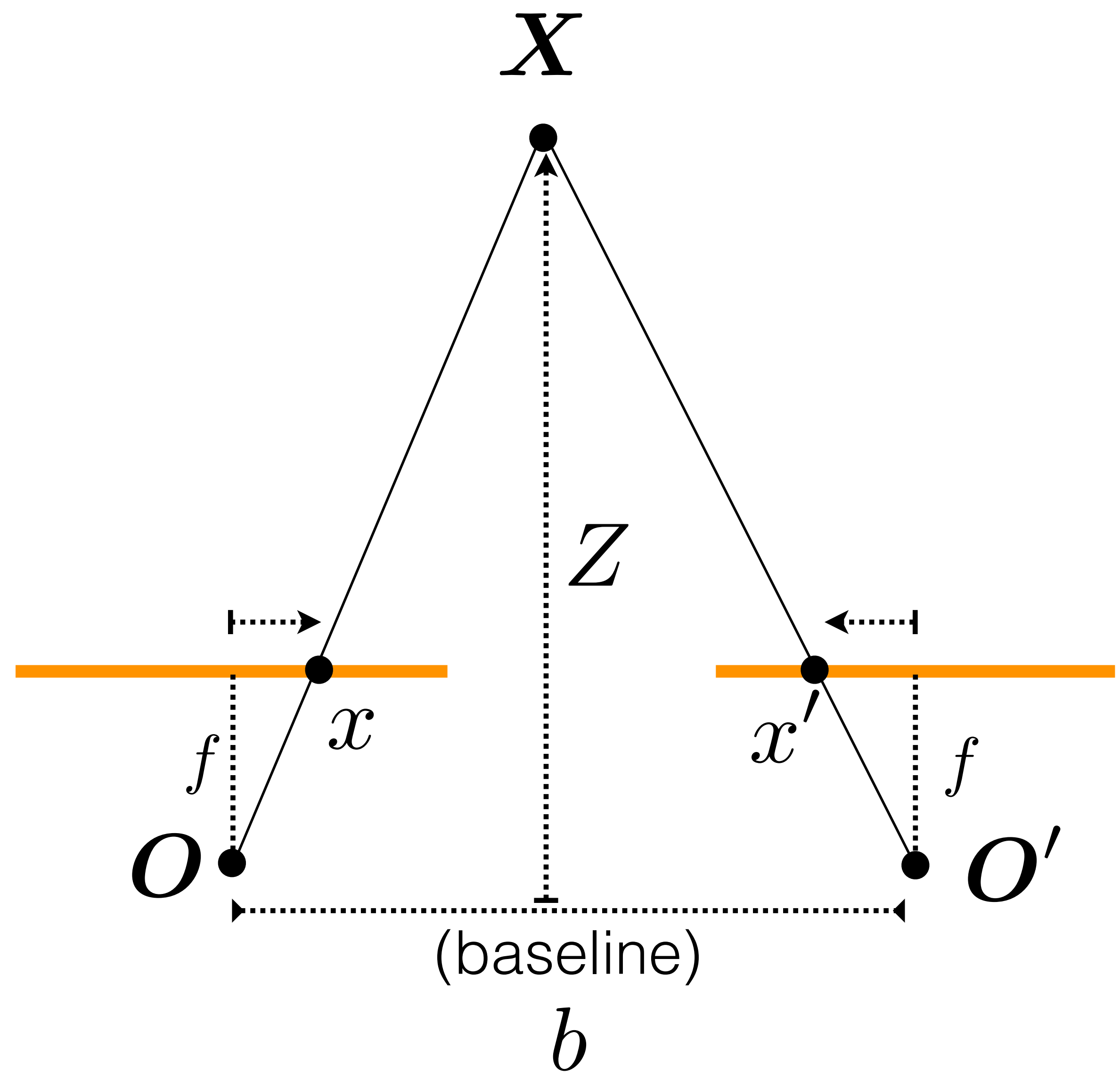

$$\frac{b - X}{Z} = \frac{x'}{f}$$

# Disparity

$$d = x - x'$$

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

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




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

# Disparity

$$d = x - x'$$

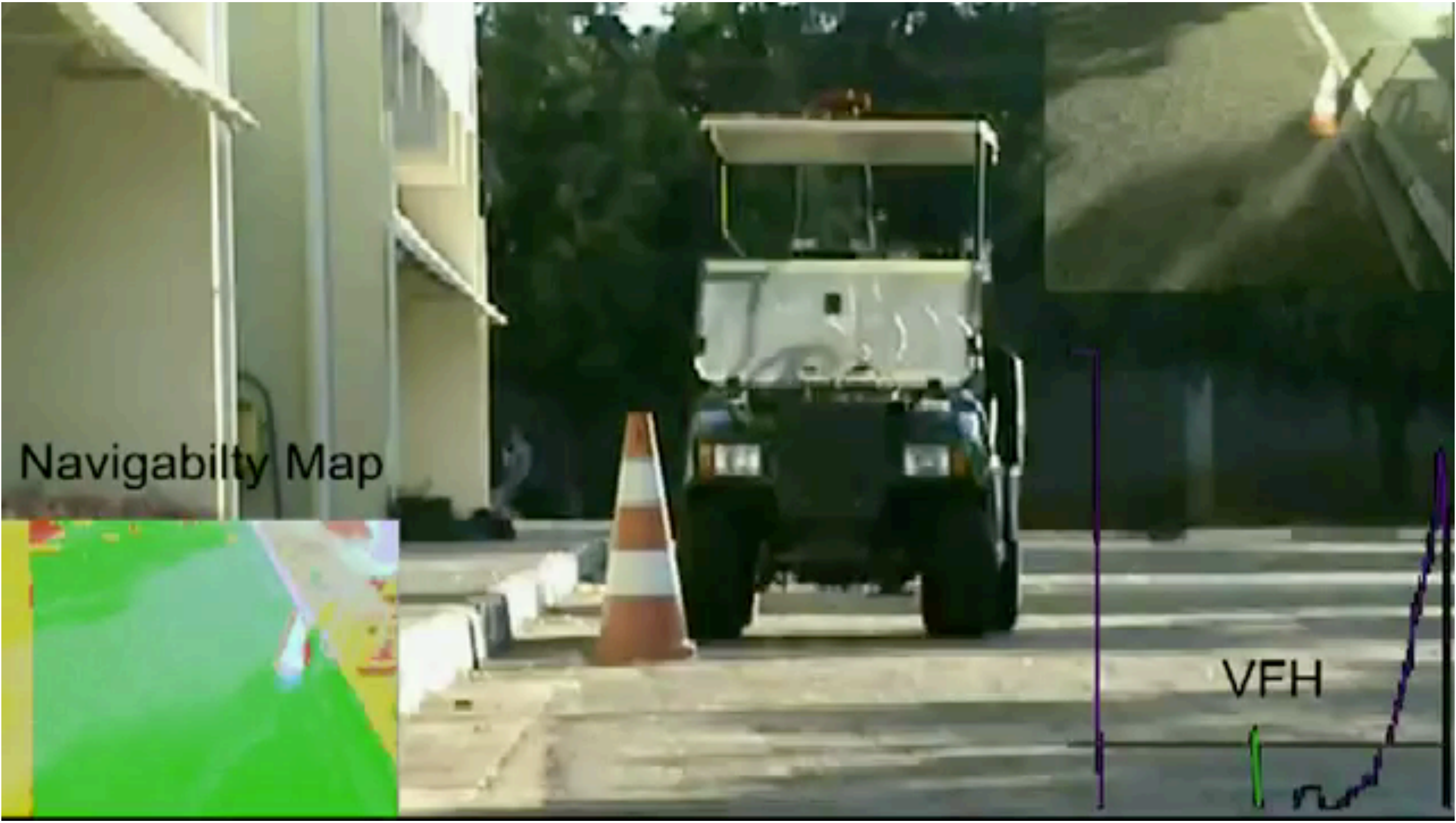
$$= \frac{bf}{Z}$$

inversely proportional to depth

# Real-time stereo sensing

Nomad robot searches for meteorites in Antartica

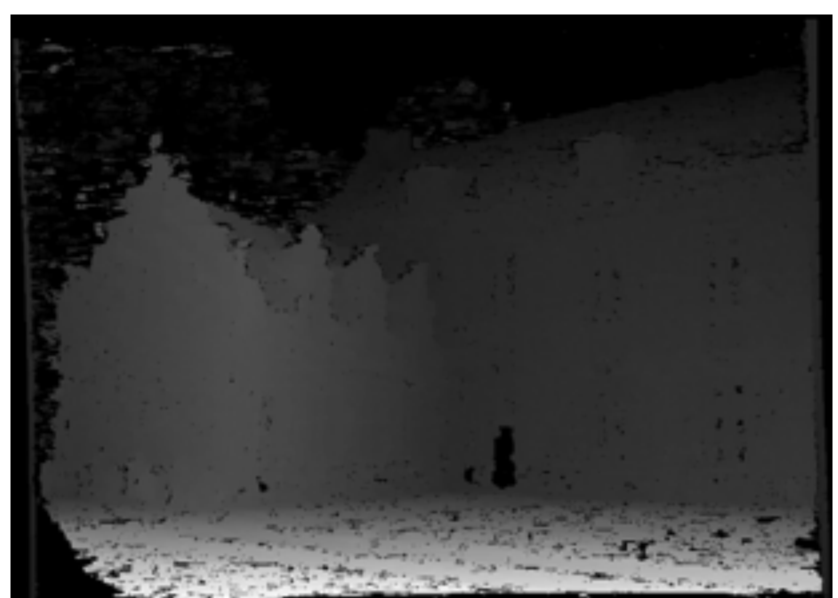
http://www.frc.ri.cmu.edu/projects/meteorobot/index.html

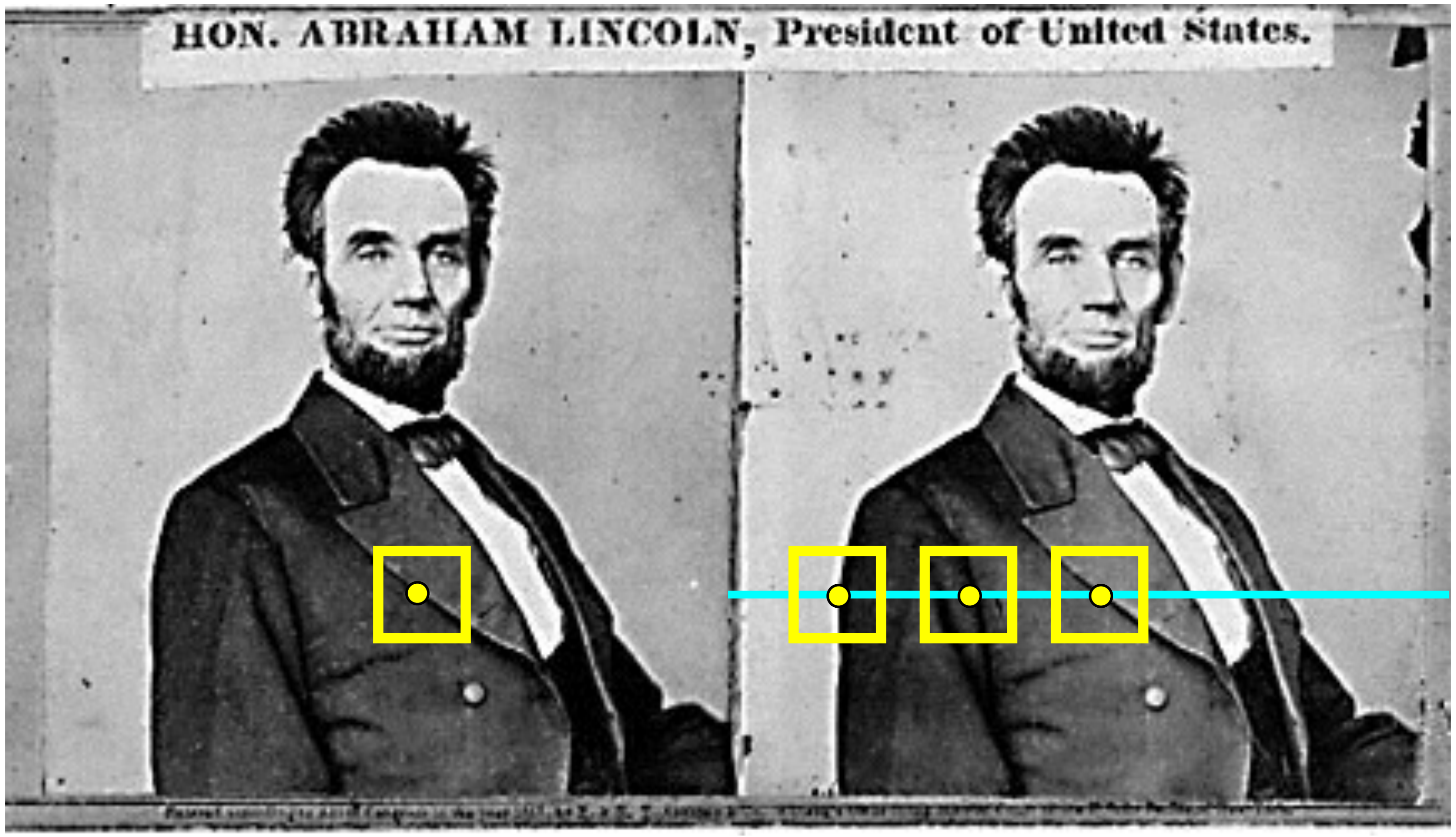

Navigabilty Map
VFH

Subaru
Eyesight system

Pre-collision
braking

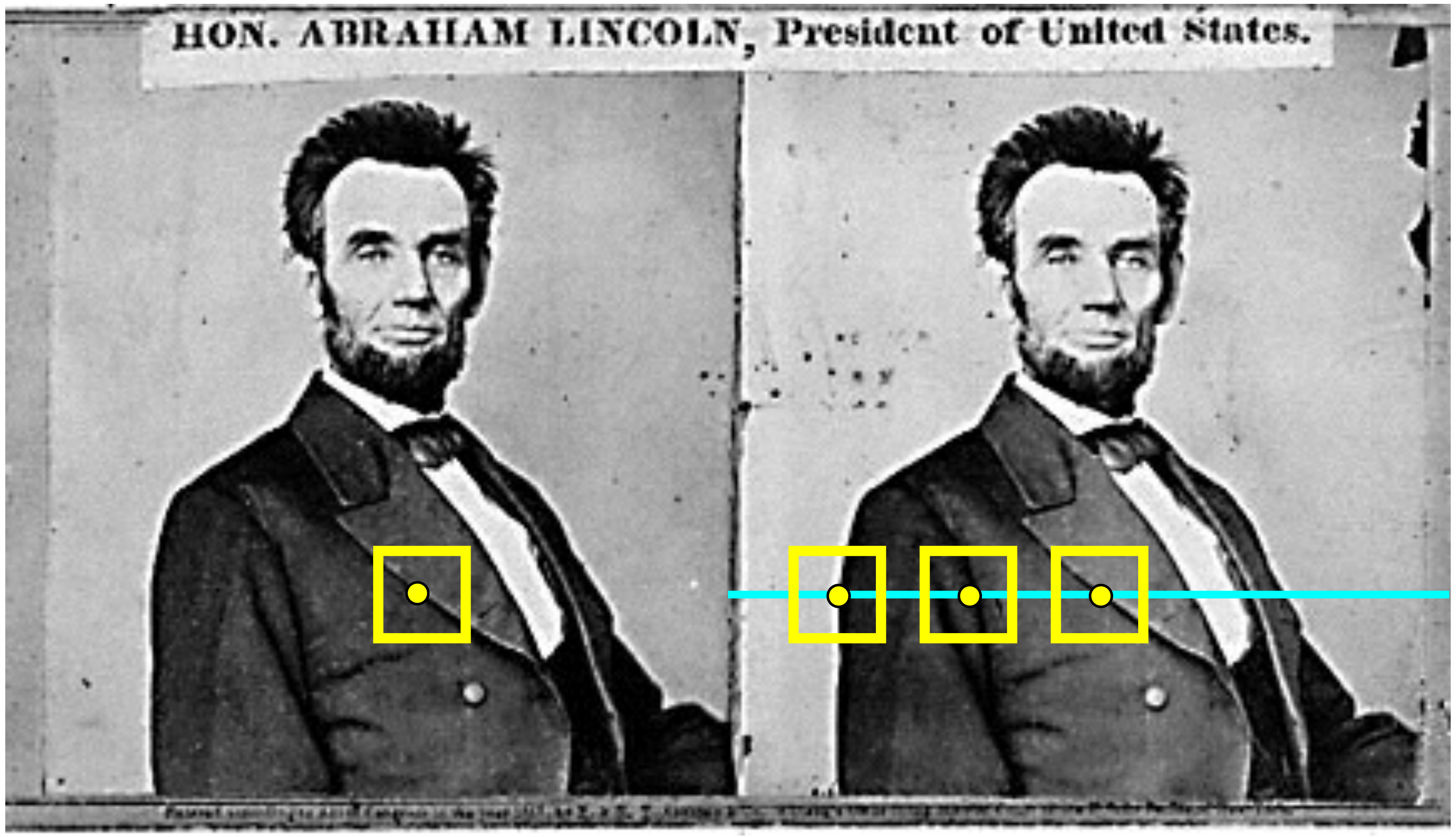
How so you compute depth
from a stereo pair?



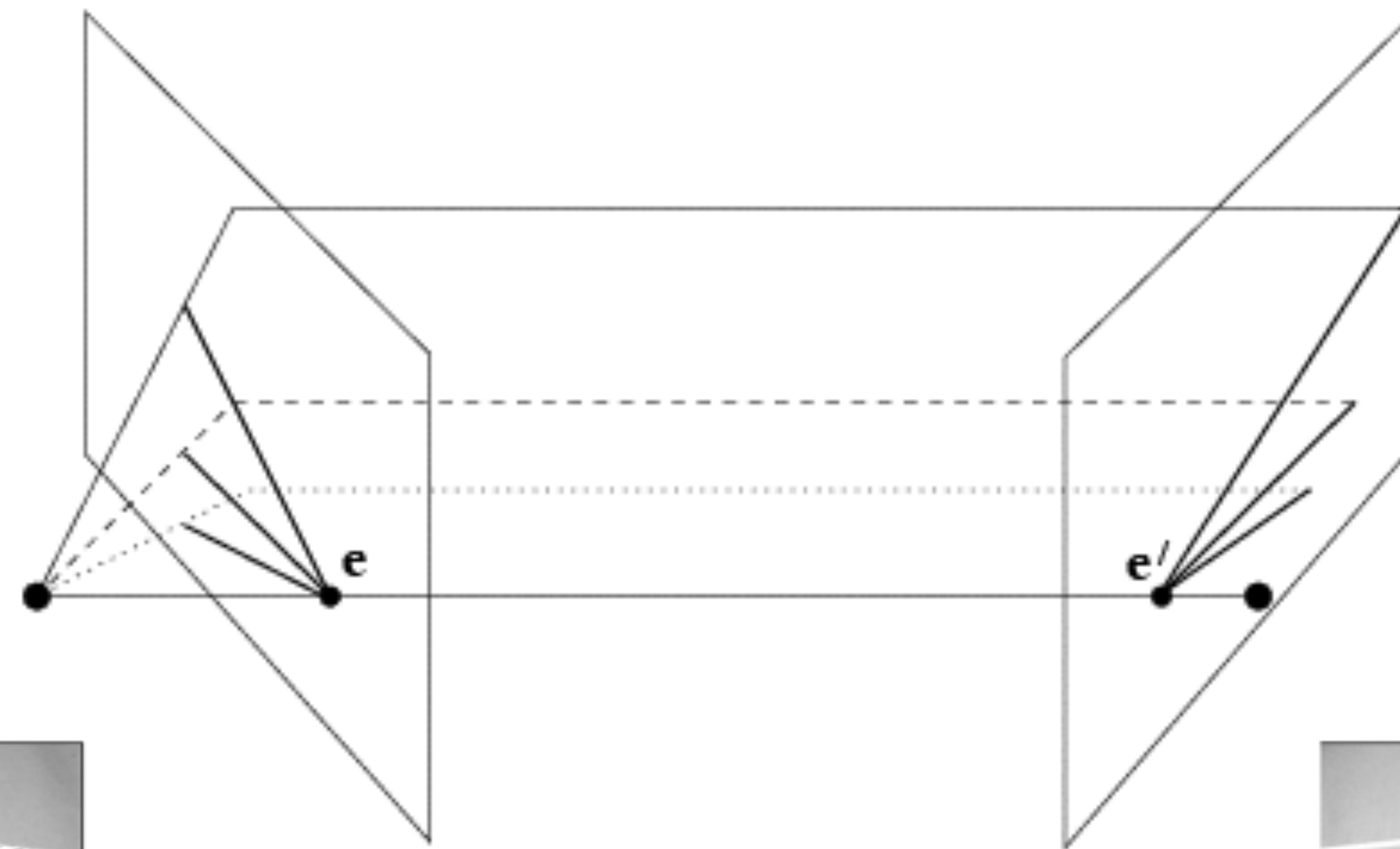



1. Rectify images
   (make epipolar lines horizontal)
2. For each pixel
   a. Find epipolar line
   b. Scan line for best match
   c. Compute depth from disparity

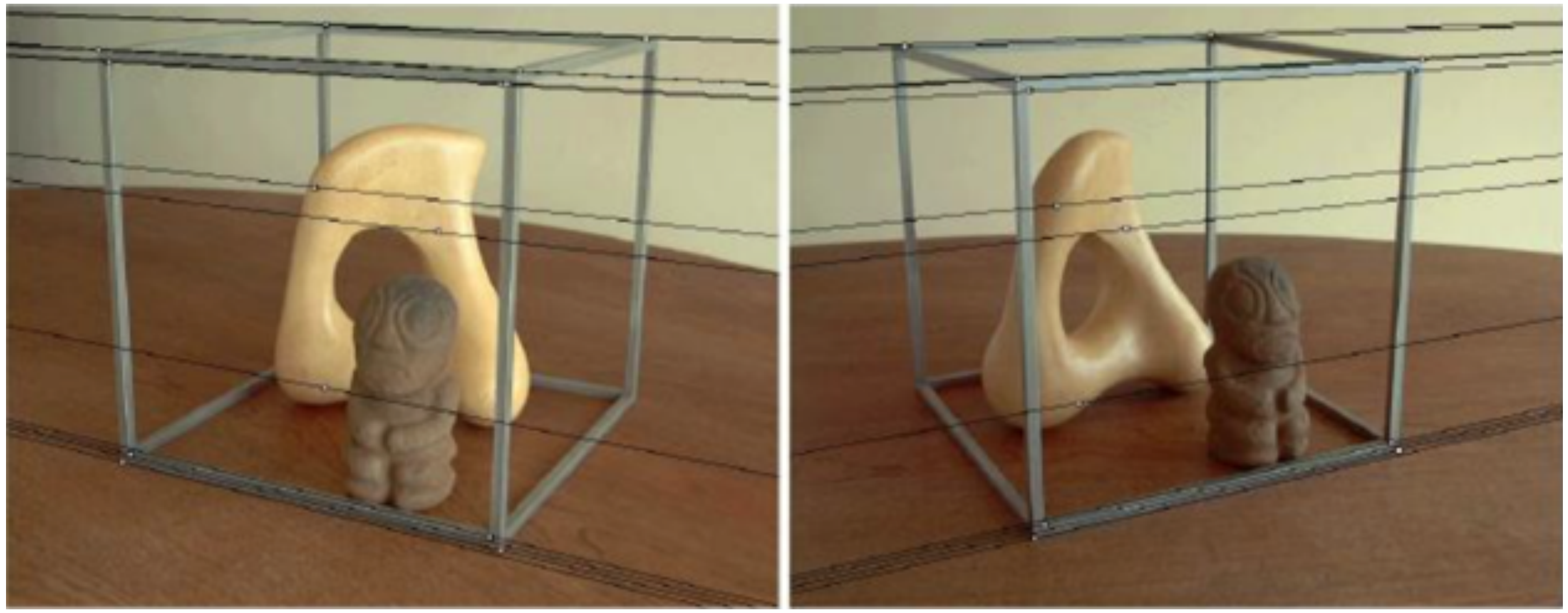
$$Z = \frac{bf}{d}$$



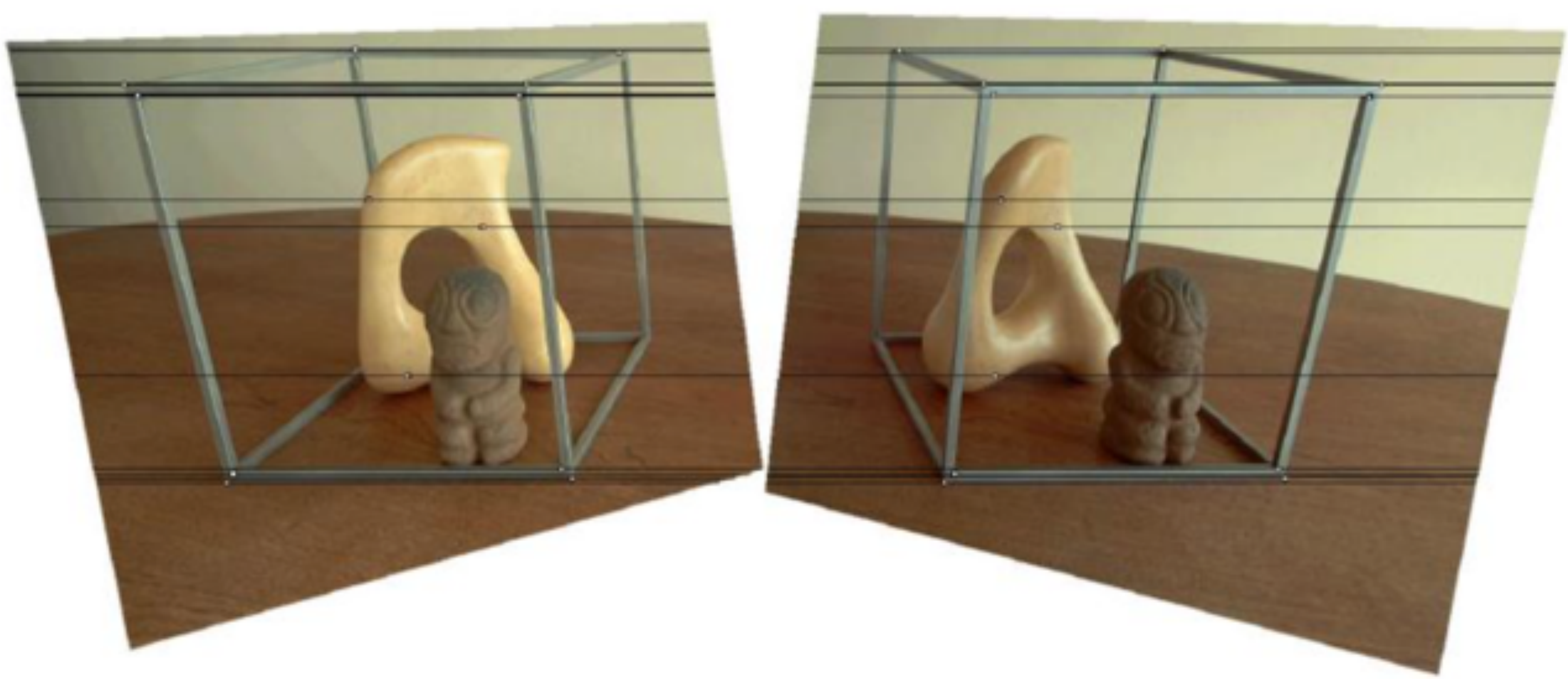

*How can you make the epipolar lines horizontal?*



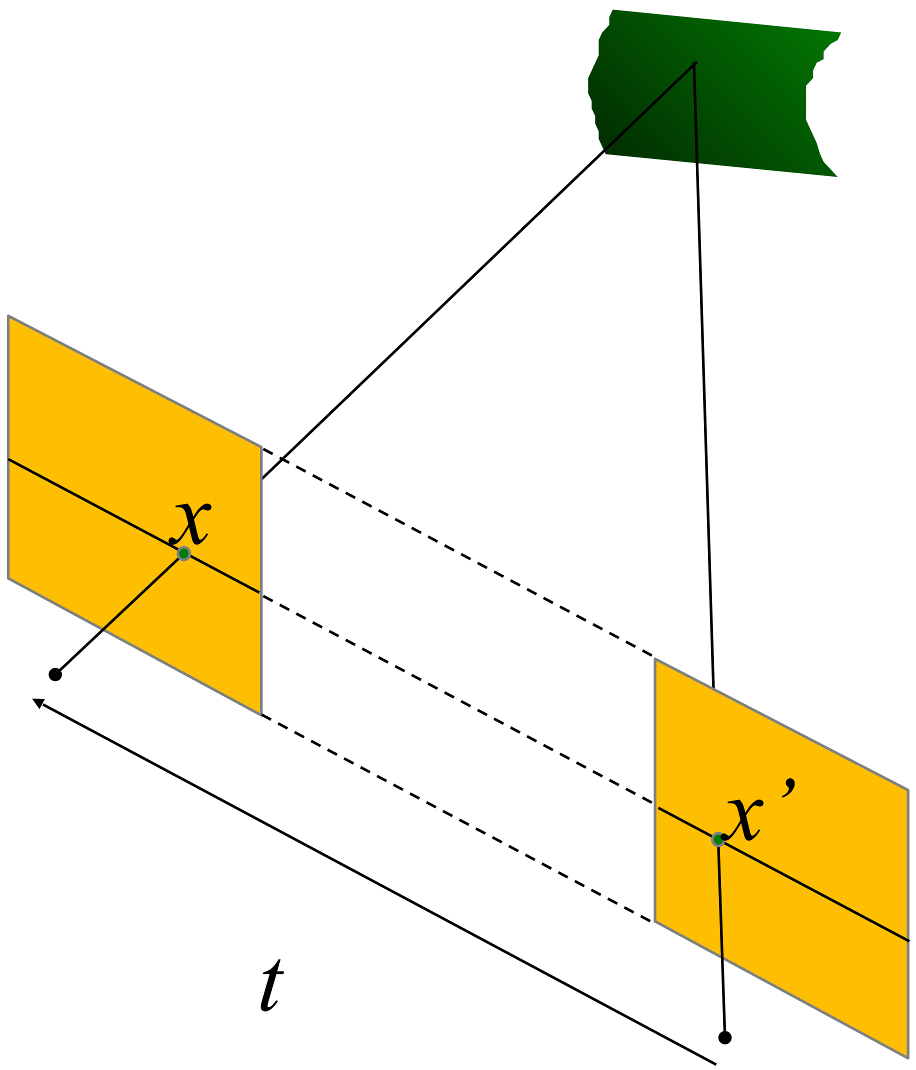

It’s hard to make the image planes exactly parallel



*How can you make the epipolar lines horizontal?*



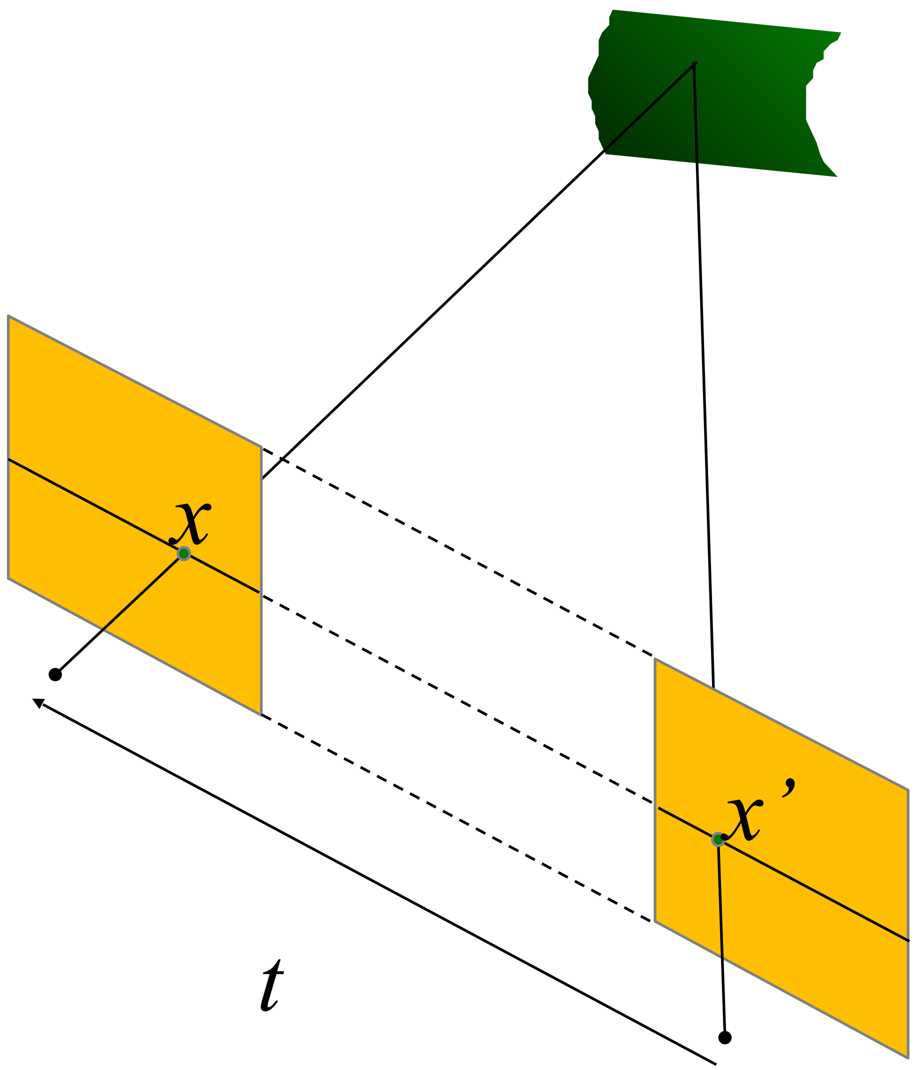
# *How can you make the epipolar lines horizontal?*




**When this relationship holds:**

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

# *How can you make the epipolar lines horizontal?*




**When this relationship holds:**

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

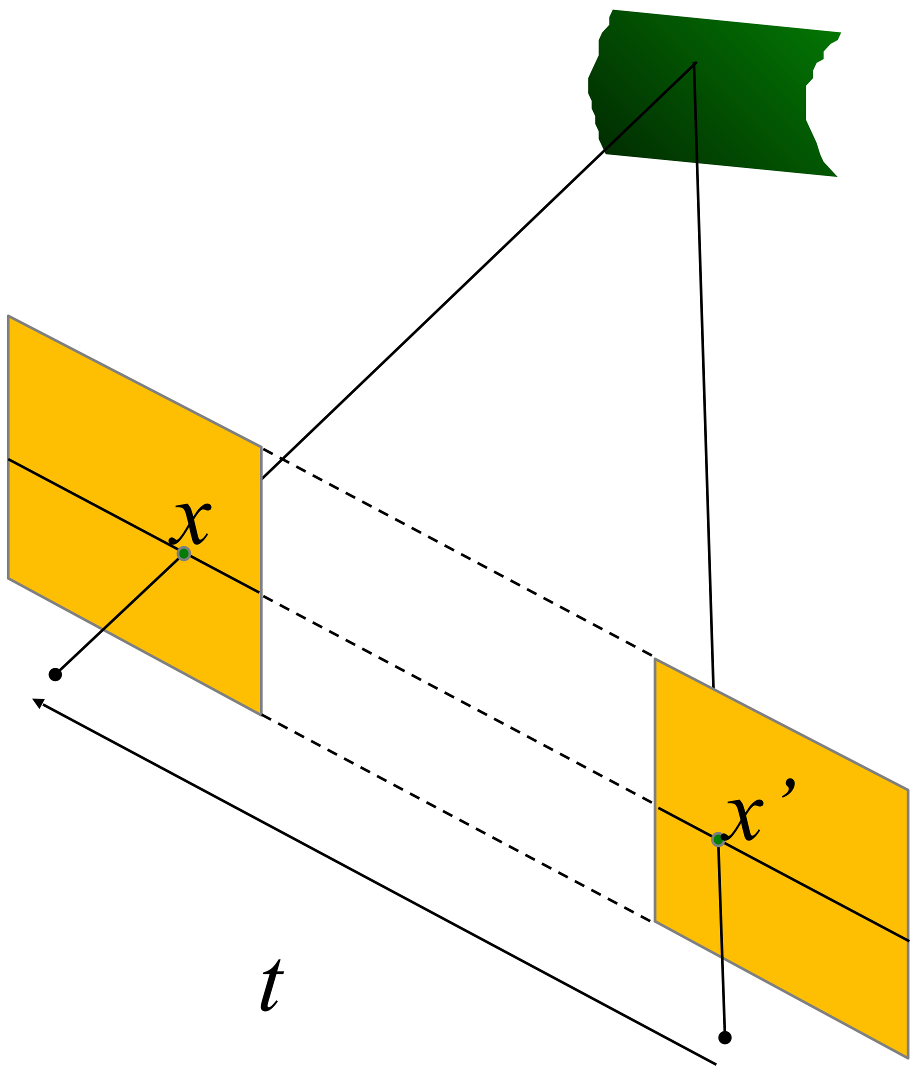
Let's try this out…

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

This always has to hold

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

# *How can you make the epipolar lines horizontal?*




**When this relationship holds:**

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

Let's try this out…

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

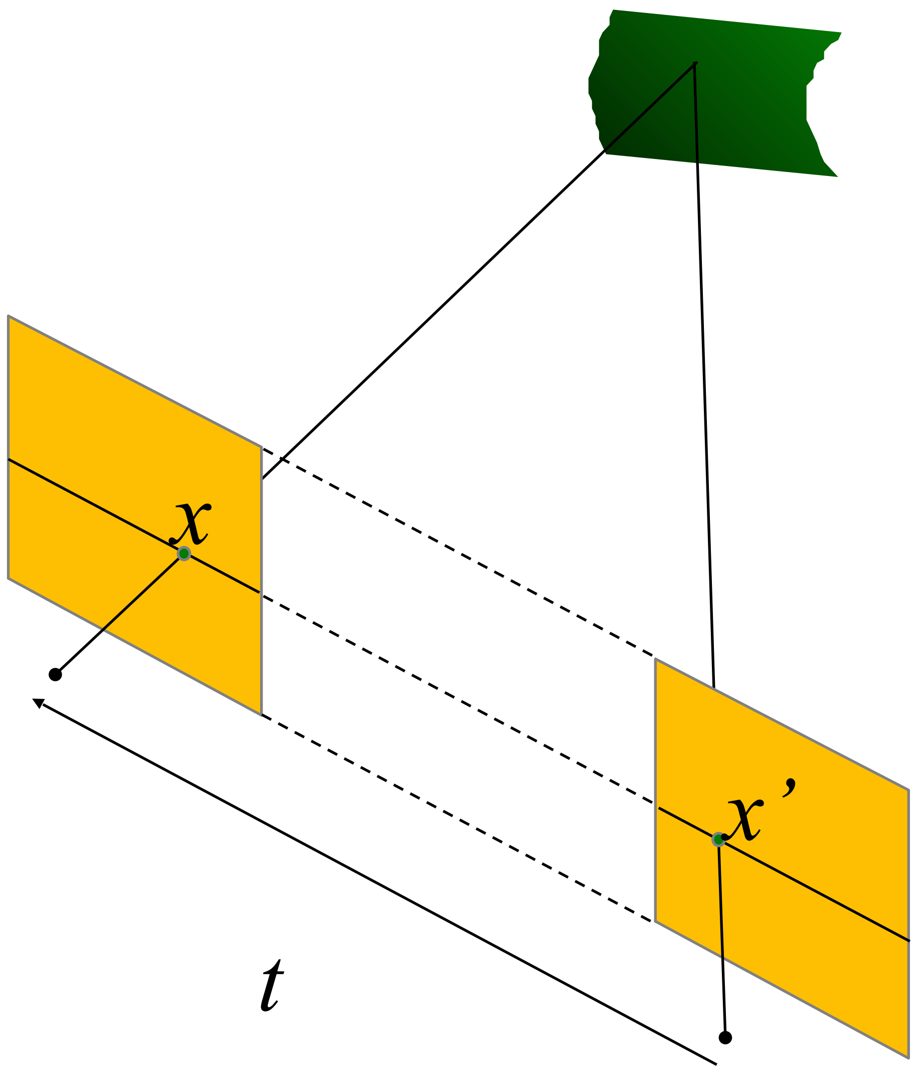
This always has to hold

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

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:**

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

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

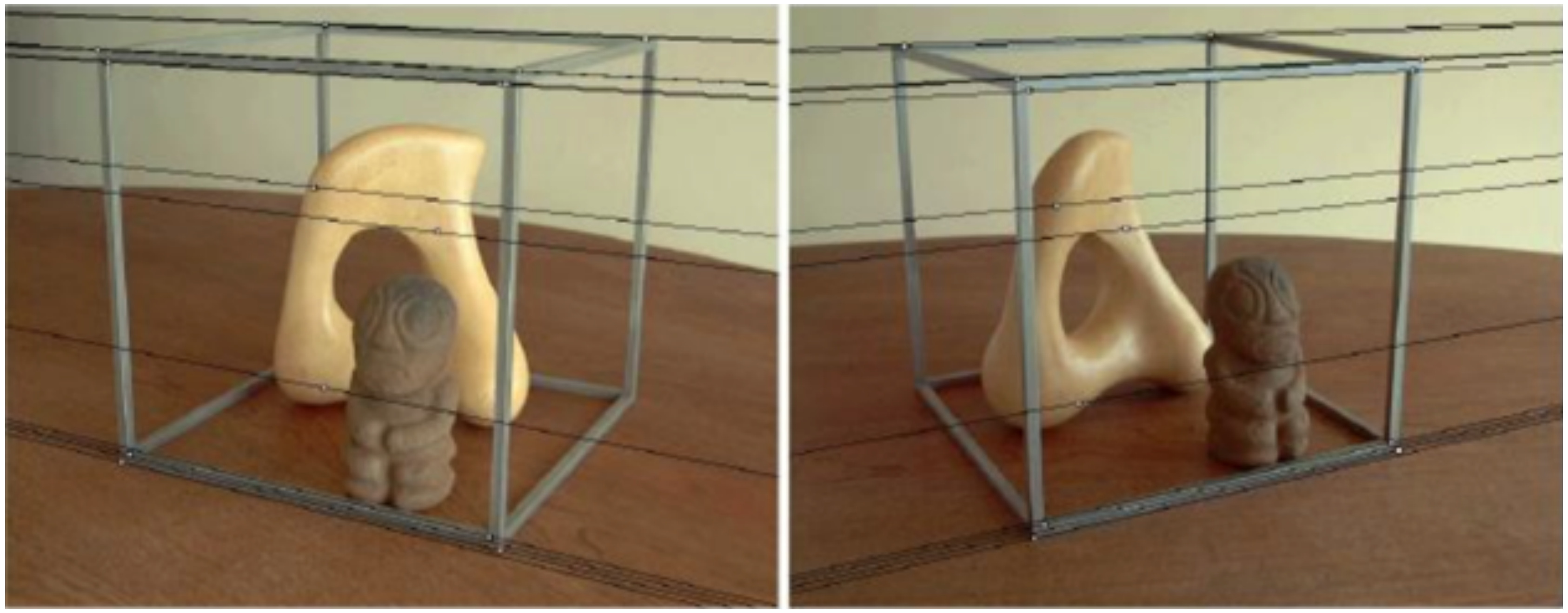
$$\begin{pmatrix} u & v & 1 \end{pmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T \\ 0 & T & 0 \end{bmatrix} \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = 0$$

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

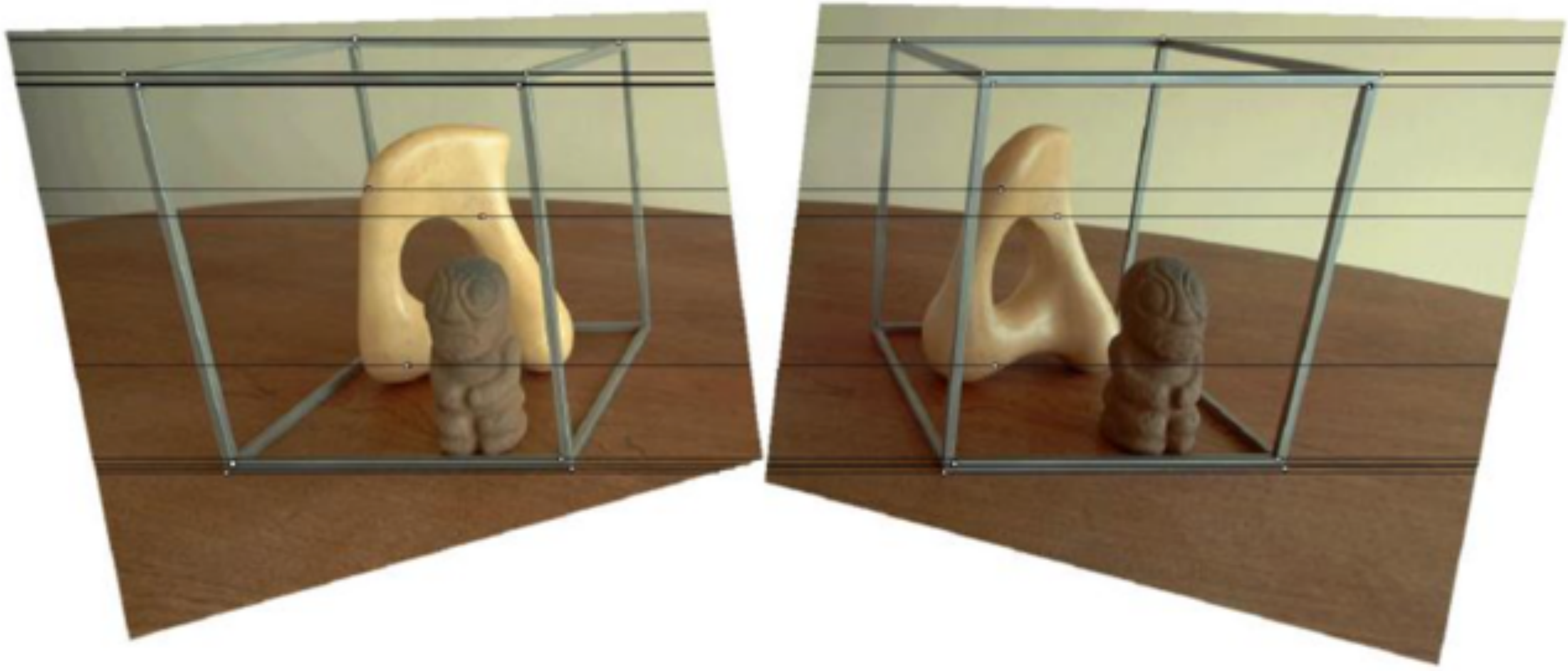
The image of a 3D point will always be on the same horizontal line

$$Tv = Tv'$$

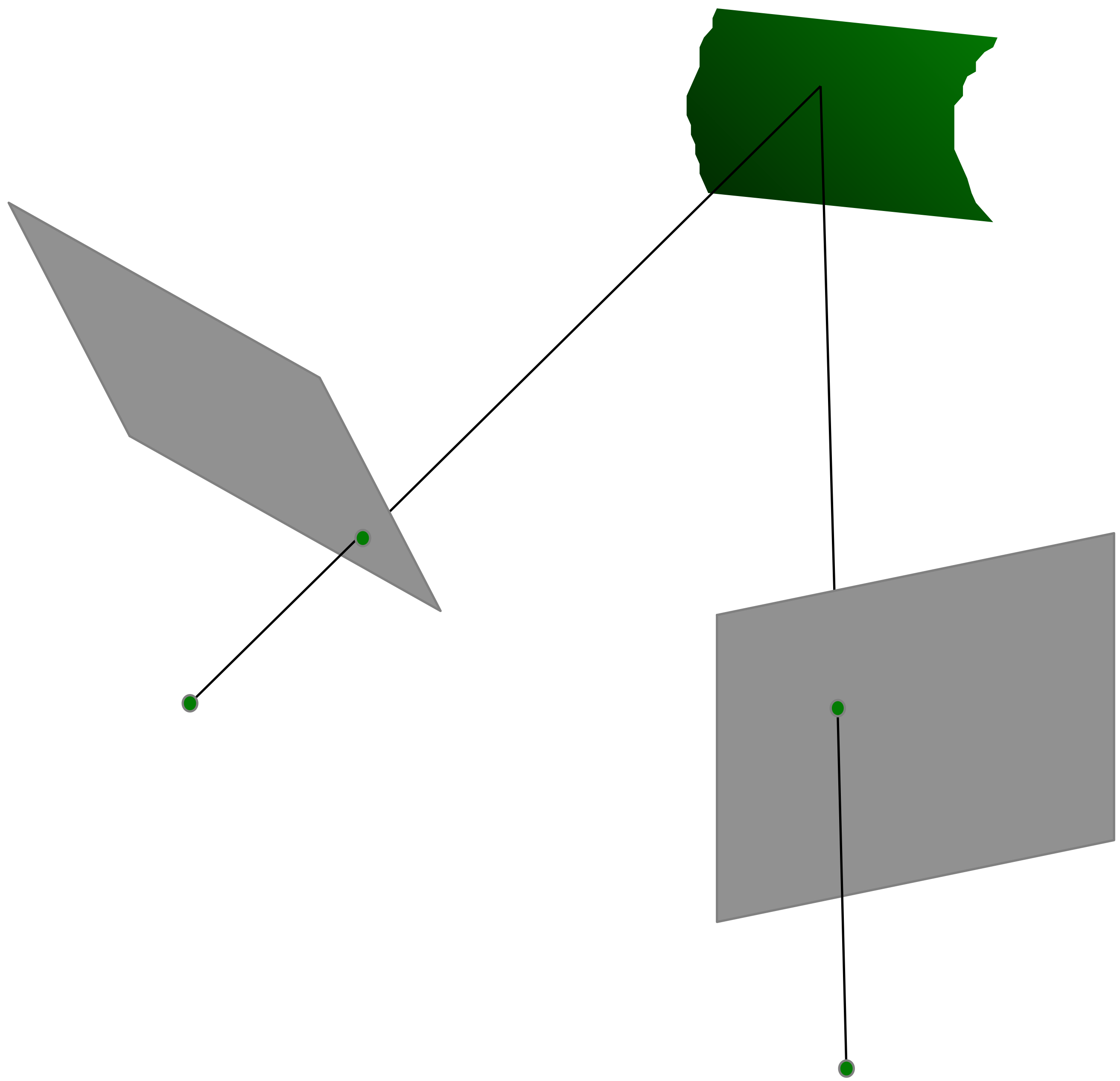
y coordinate is always the same!



What is stereo rectification?

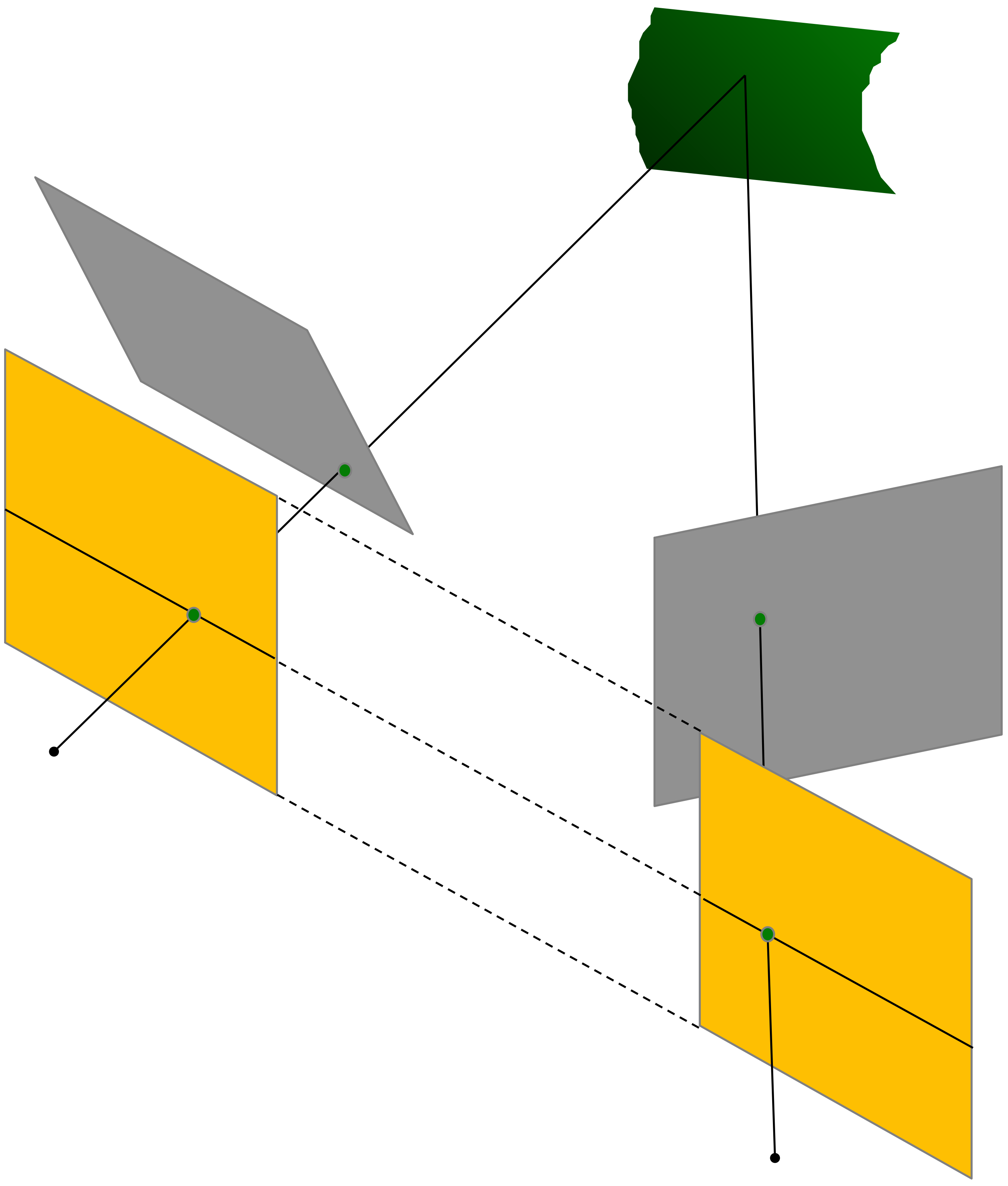


# *What is stereo rectification?*



## *What is stereo rectification?*

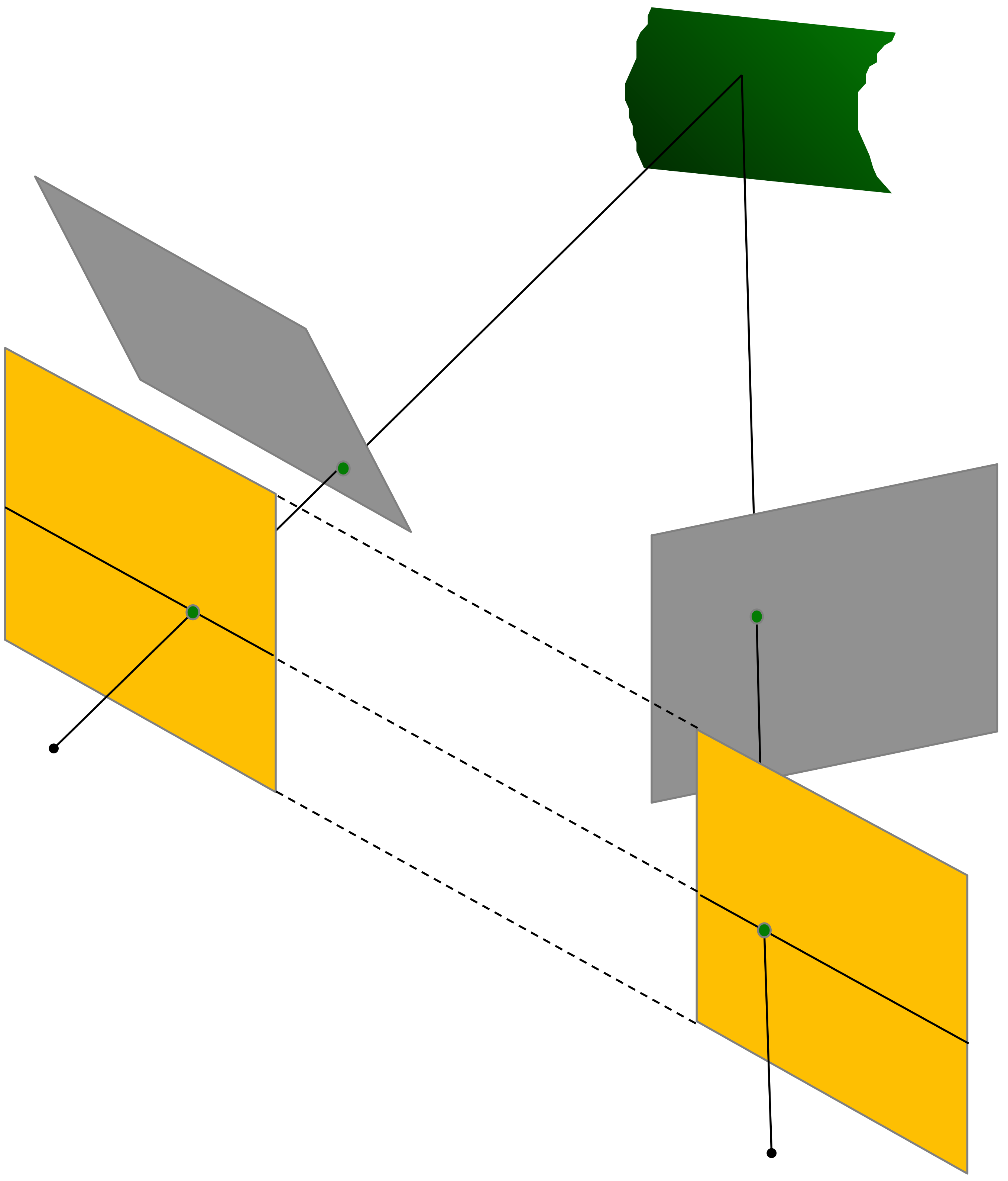
Reproject image planes onto a common plane parallel to the line between camera centers



## *What is stereo rectification?*

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

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



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

# Stereo Rectification

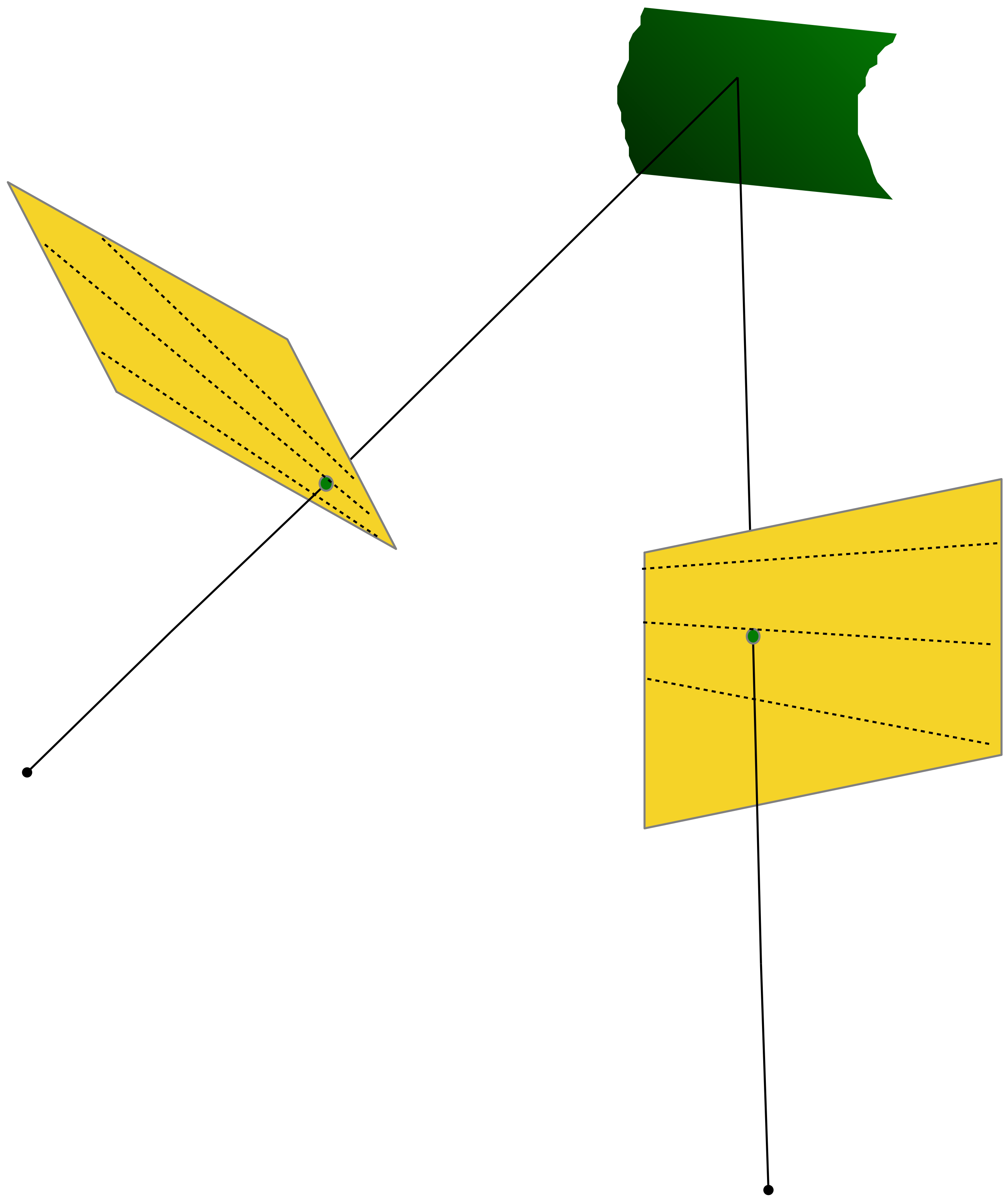
1. **Rotate** the right camera by **R**
   (aligns camera coordinate system orientation only)

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

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

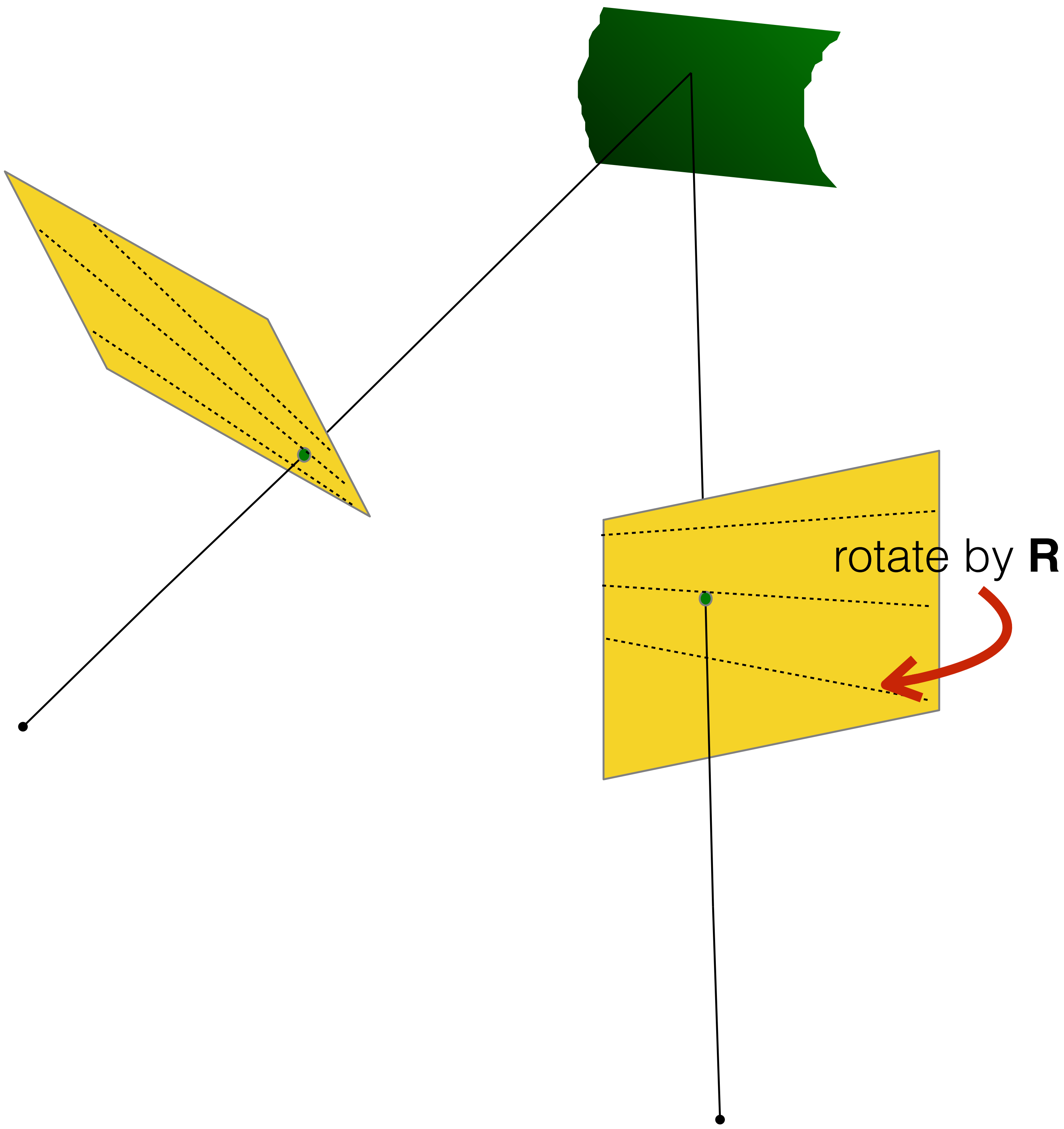
4. Adjust the **scale**

# Stereo Rectification:



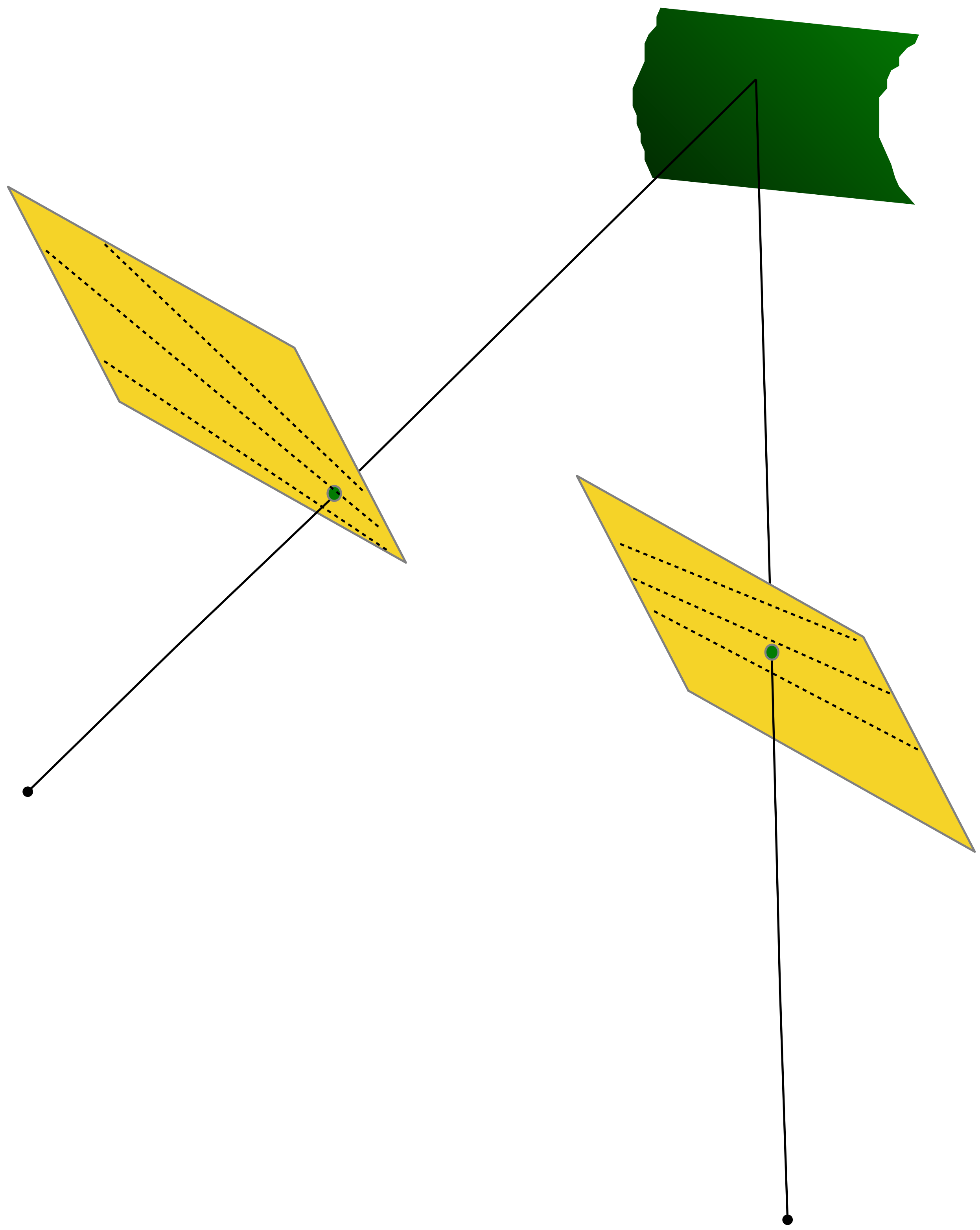
1. Compute **E** to get **R**
2. Rotate right image by **R**
3. Rotate both images by $\mathbf{R}_{\mathbf{rect}}$
4. Scale both images by **H**

# Stereo Rectification:



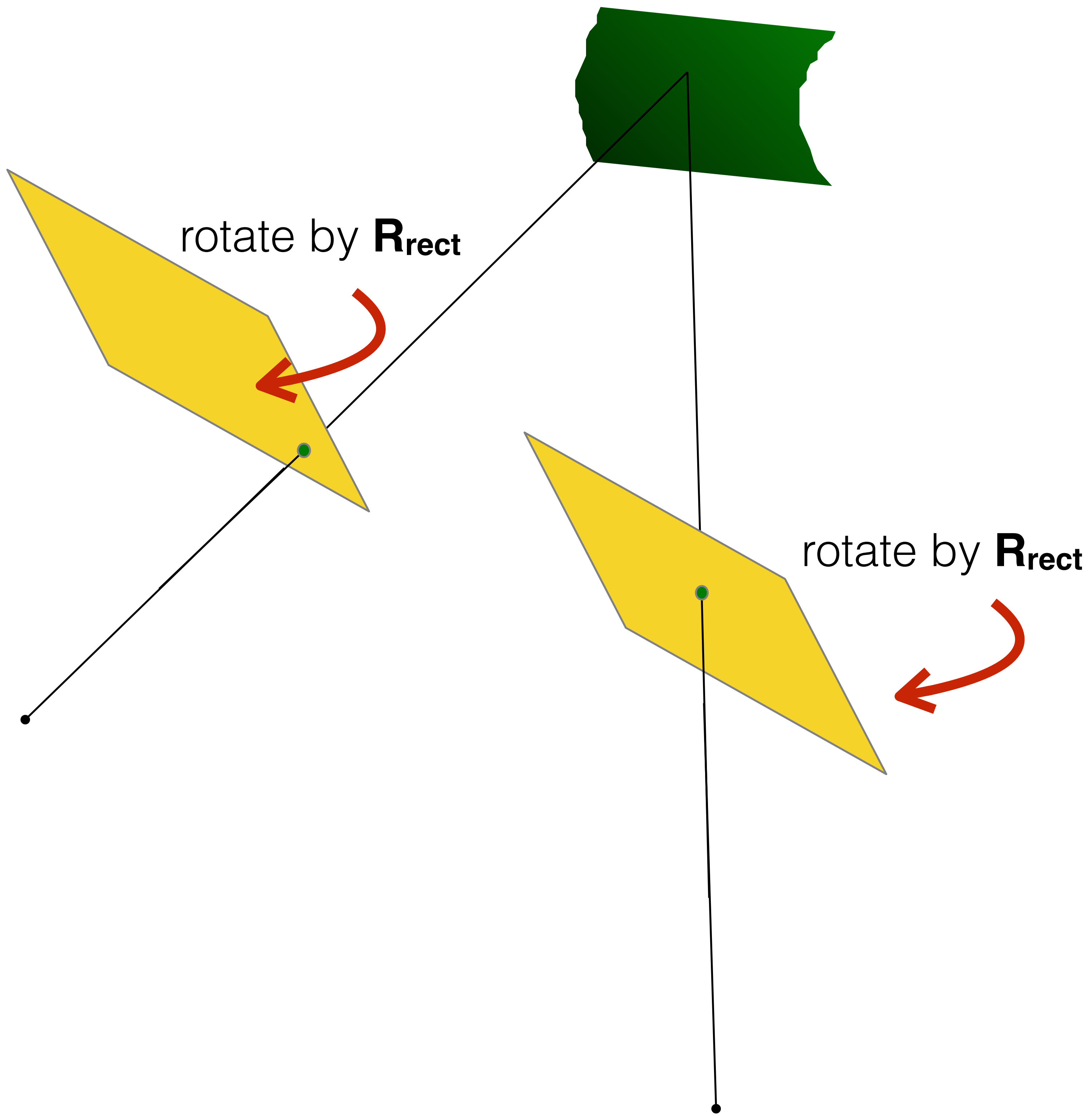

1. Compute **E** to get **R**
2. Rotate right image by **R**
3. Rotate both images by $\mathbf{R}_{rect}$
4. Scale both images by **H**

# Stereo Rectification:



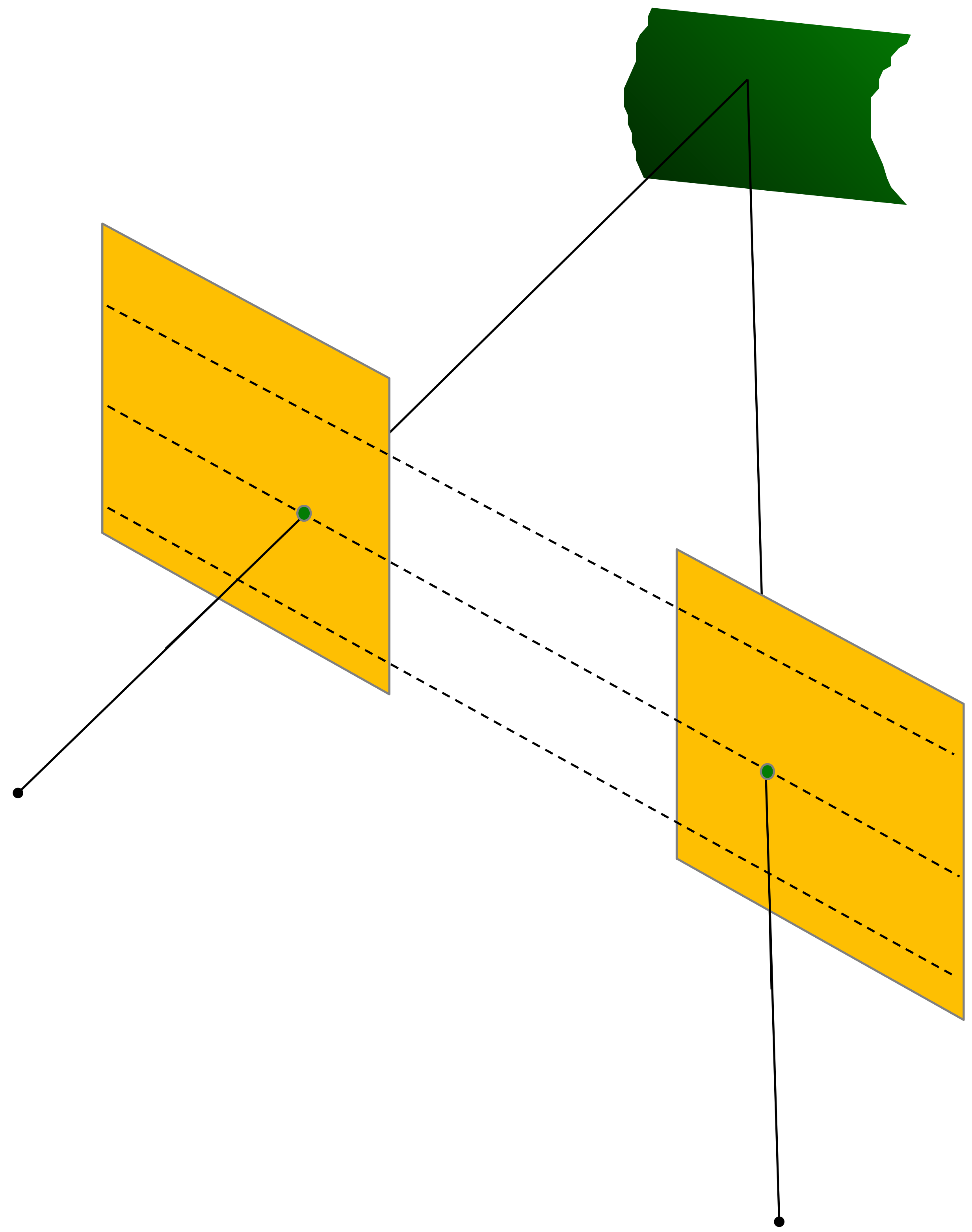
1. Compute **E** to get **R**
2. Rotate right image by **R**
3. Rotate both images by $\mathbf{R}_{\mathbf{rect}}$
4. Scale both images by **H**

# Stereo Rectification:



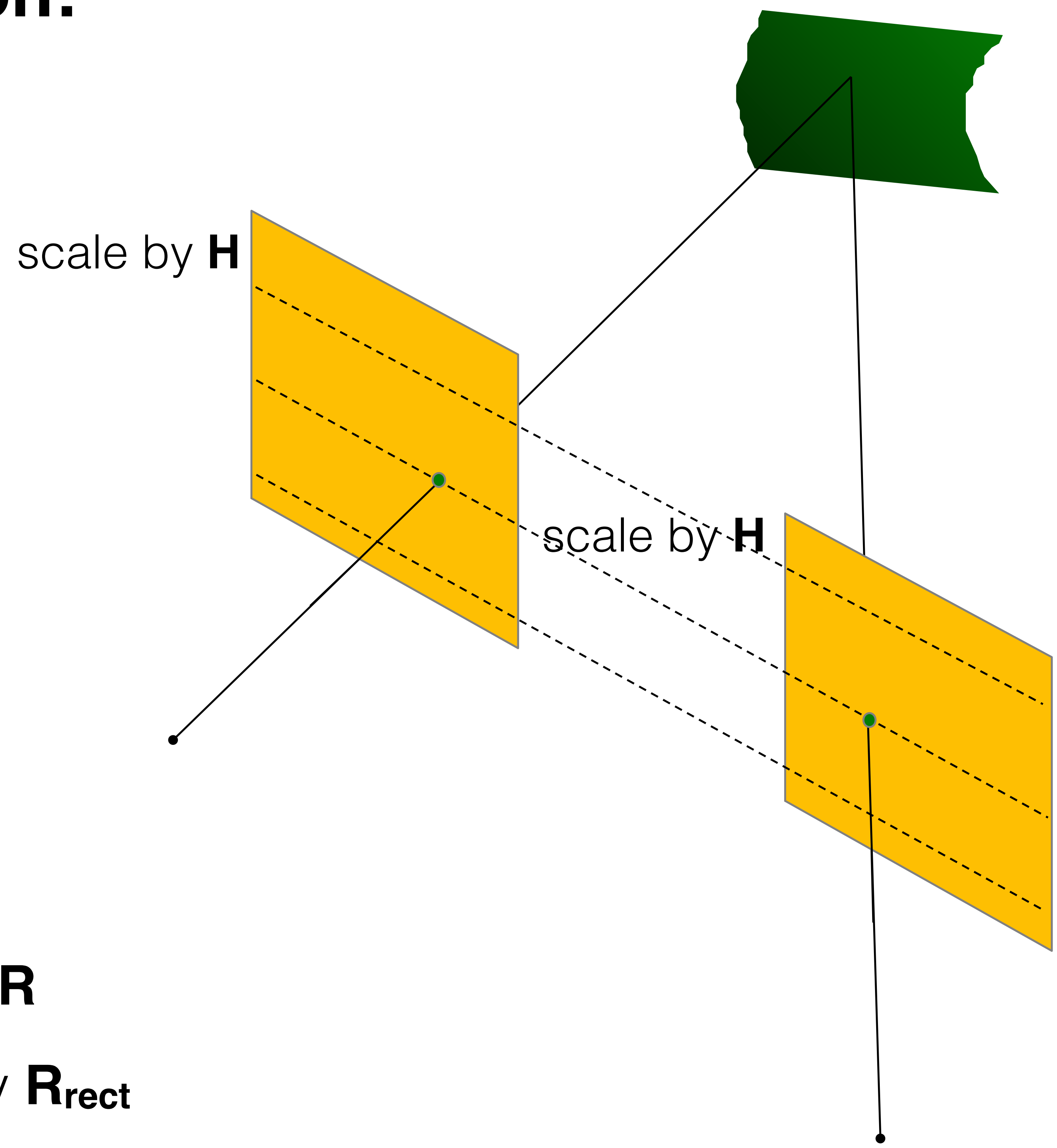

1. Compute **E** to get **R**
2. Rotate right image by **R**
3. Rotate both images by $\mathbf{R}_{\mathbf{rect}}$
4. Scale both images by **H**

# Stereo Rectification:



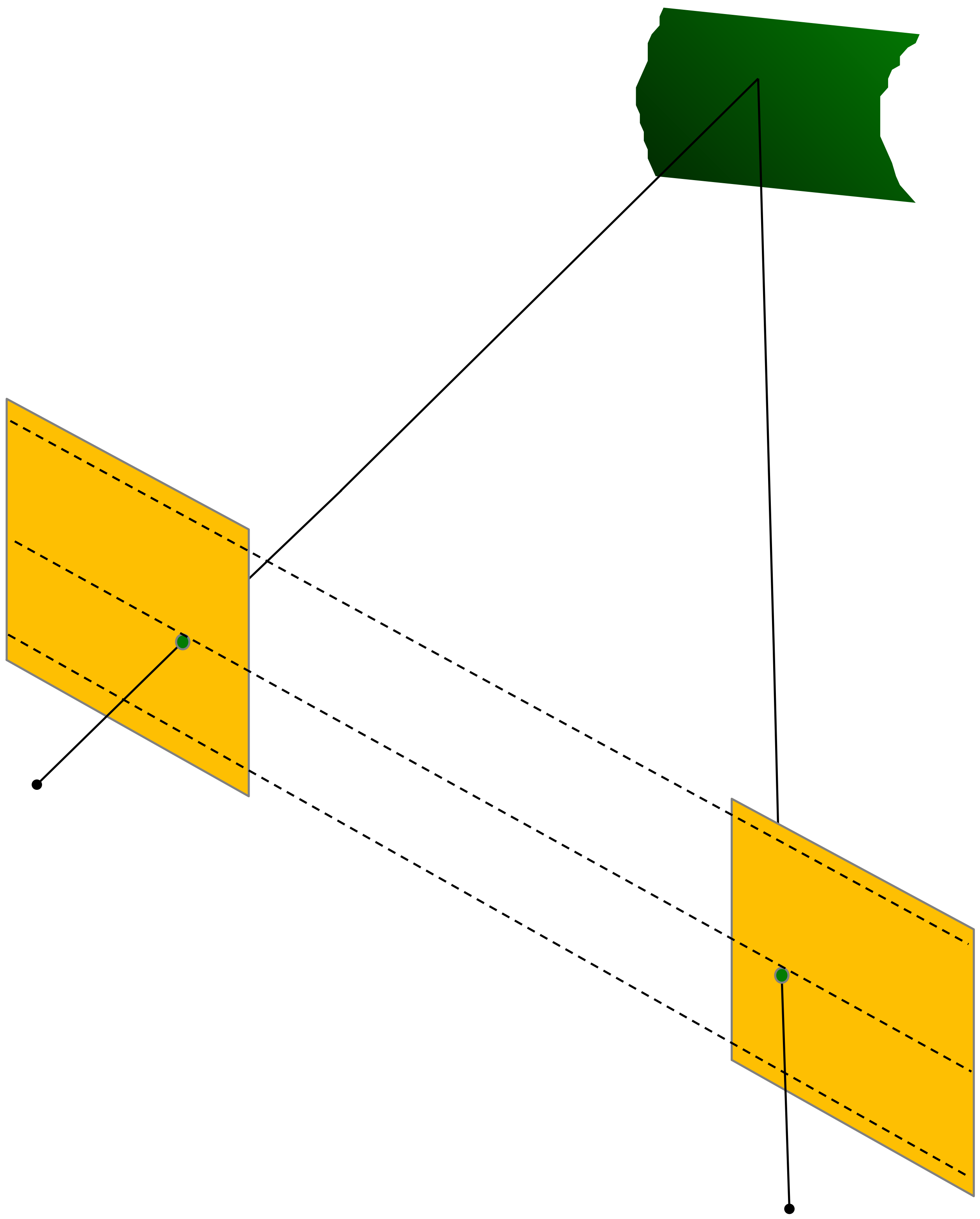
1. Compute **E** to get **R**
2. Rotate right image by **R**
3. Rotate both images by $\mathbf{R}_{\mathbf{rect}}$
4. Scale both images by **H**

# Stereo Rectification:

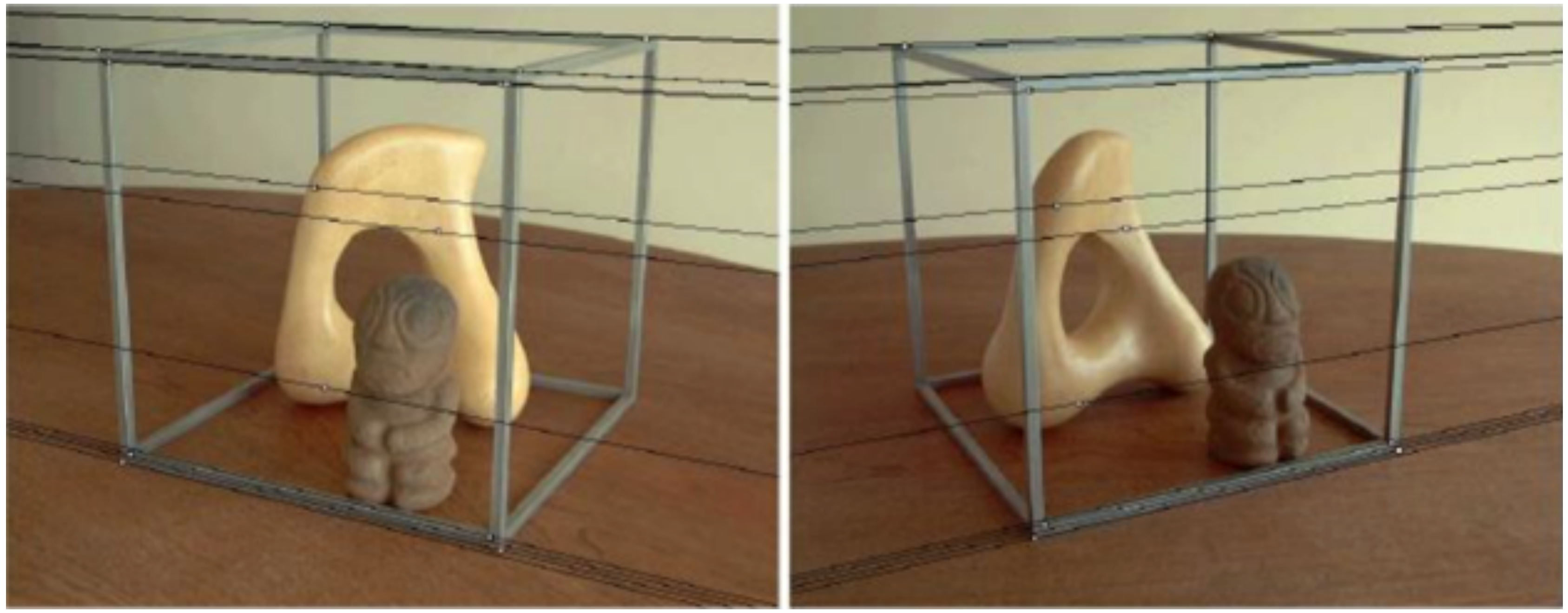



1. Compute **E** to get **R**
2. Rotate right image by **R**
3. Rotate both images by $\mathbf{R}_{\mathbf{rect}}$
4. Scale both images by **H**

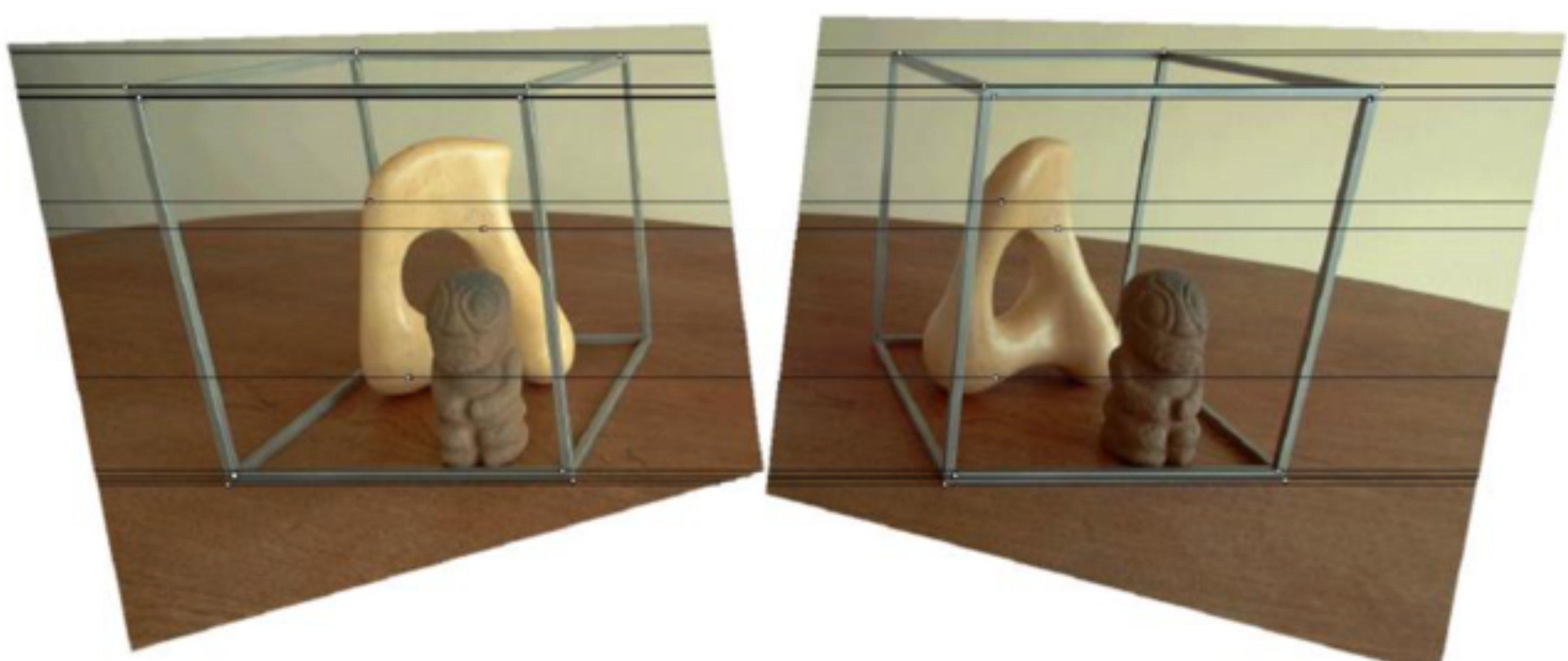
# Stereo Rectification:



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



# What can we do after rectification?



# Setting the epipole to infinity

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

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

Given:

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

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

epipole coincides with translation vector

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

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

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

orthogonal vector

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

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

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

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

*Where is this point located on the image plane?*

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

At x-infinity

# Stereo Rectification Algorithm

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

# Stereo Rectification Algorithm

1. Estimate **E** using the 8 point algorithm

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

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

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

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

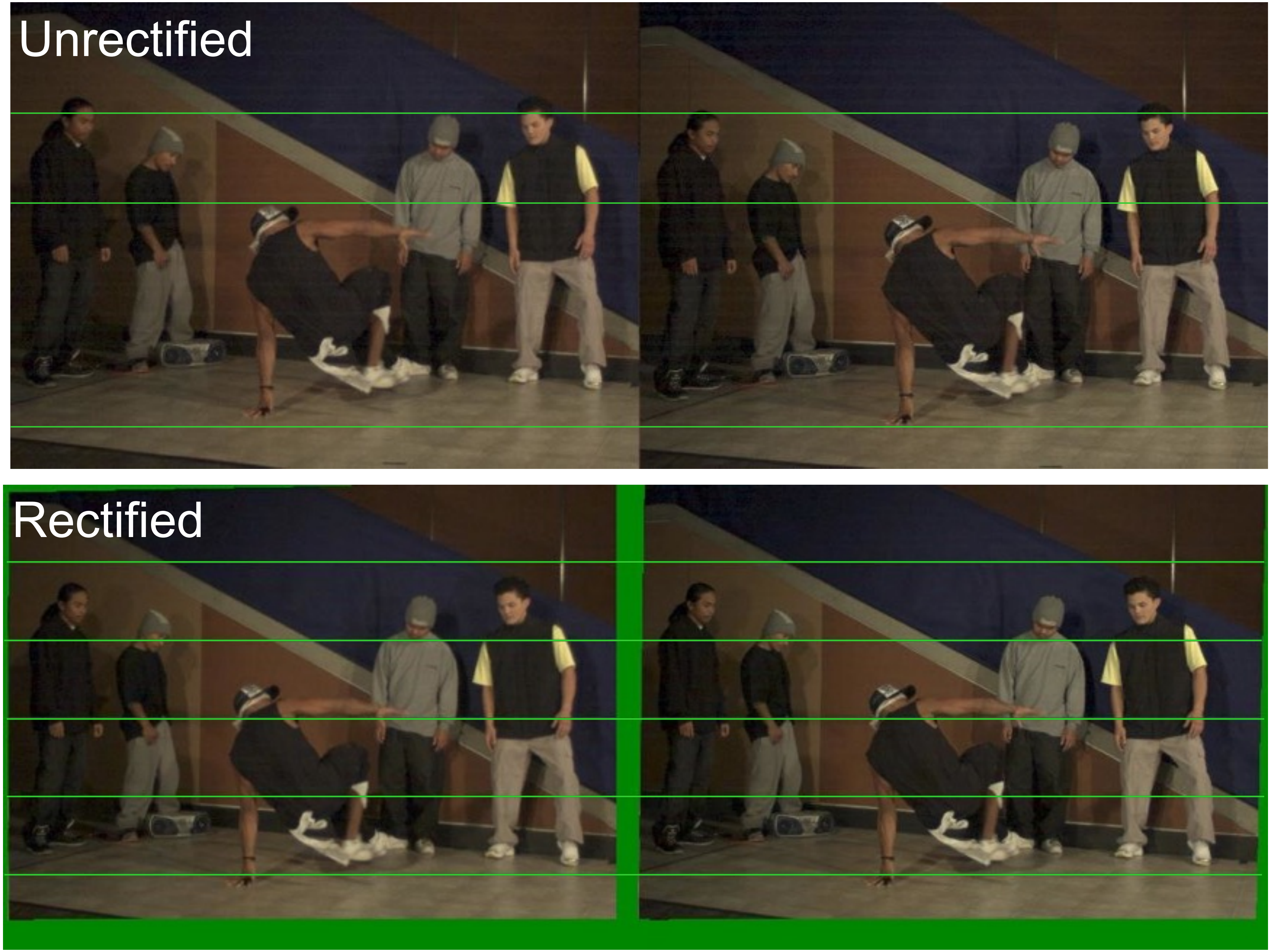
6. Rotate each left camera point $\mathbf{x}' \sim \mathbf{H}\mathbf{x}$ where $\mathbf{H} = \mathbf{K}\mathbf{R}_1$
   *You may need to alter the focal length (inside **K**) to keep points within the original image size

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


Unrectified


Unrectified


Rectified


Unrectified
Rectified