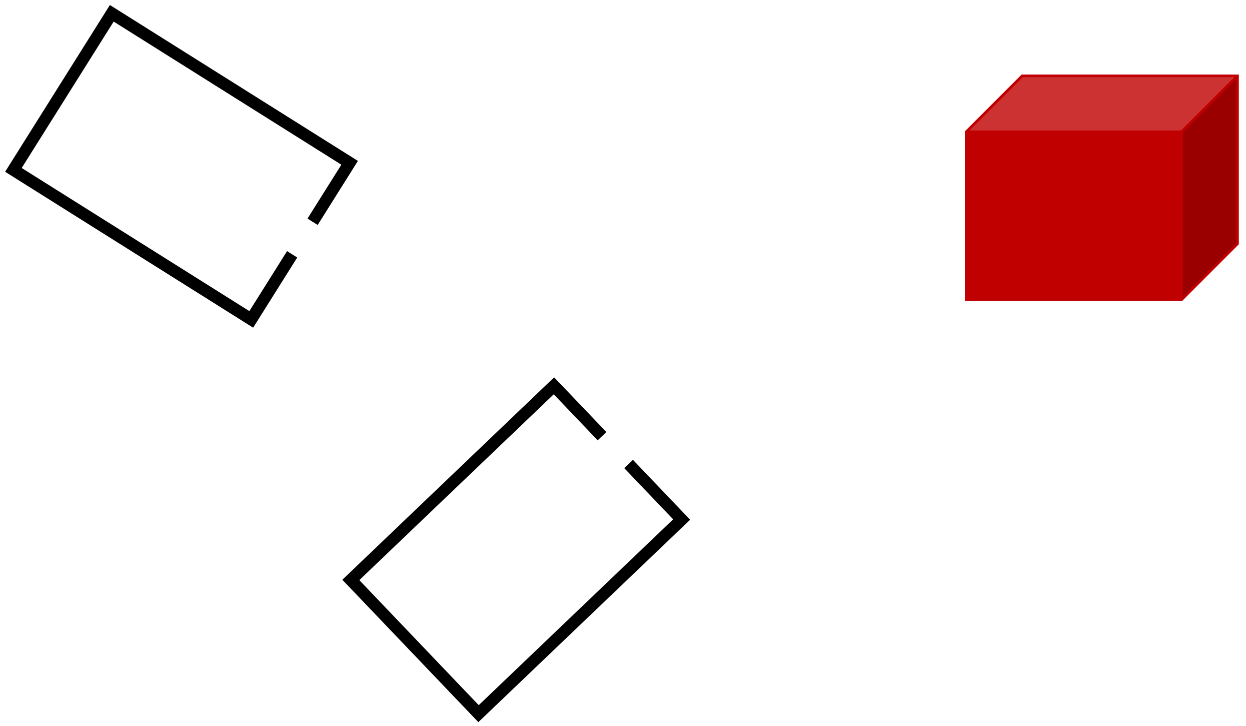
Binocular stereo

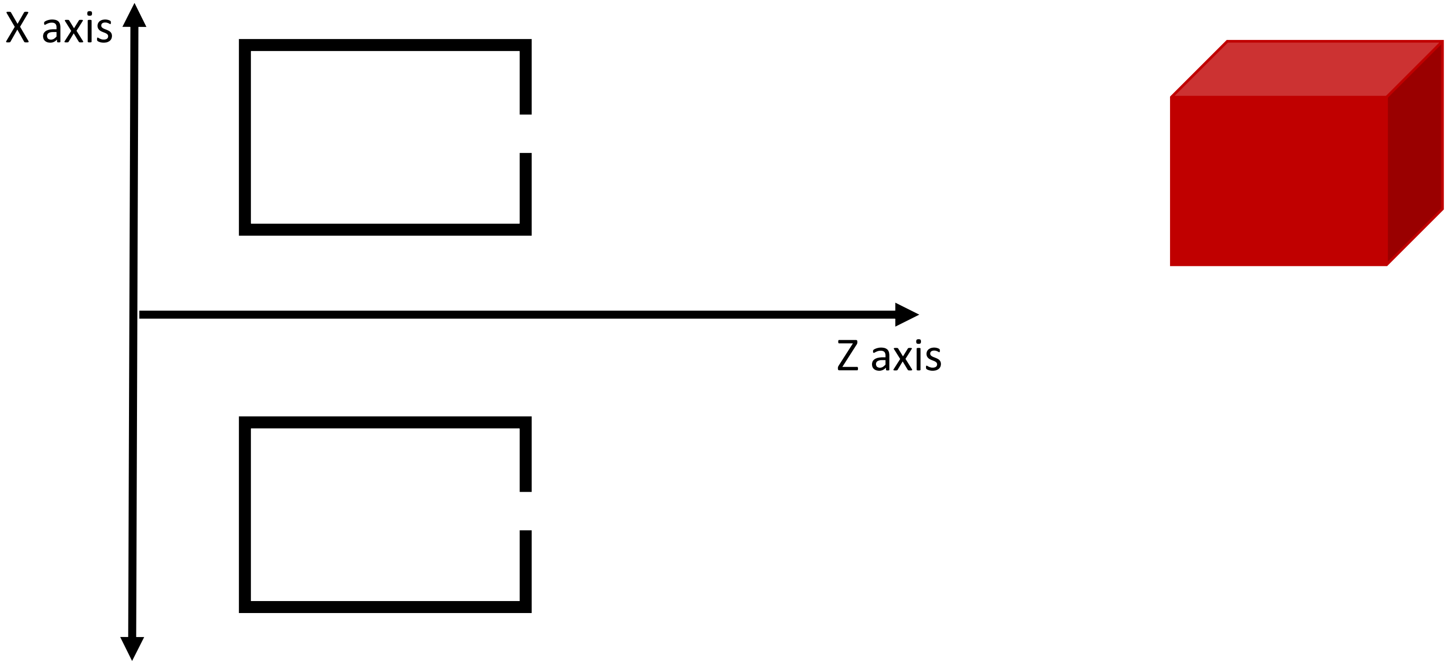
Binocular stereo

- General case: cameras can be arbitrary locations and orientations



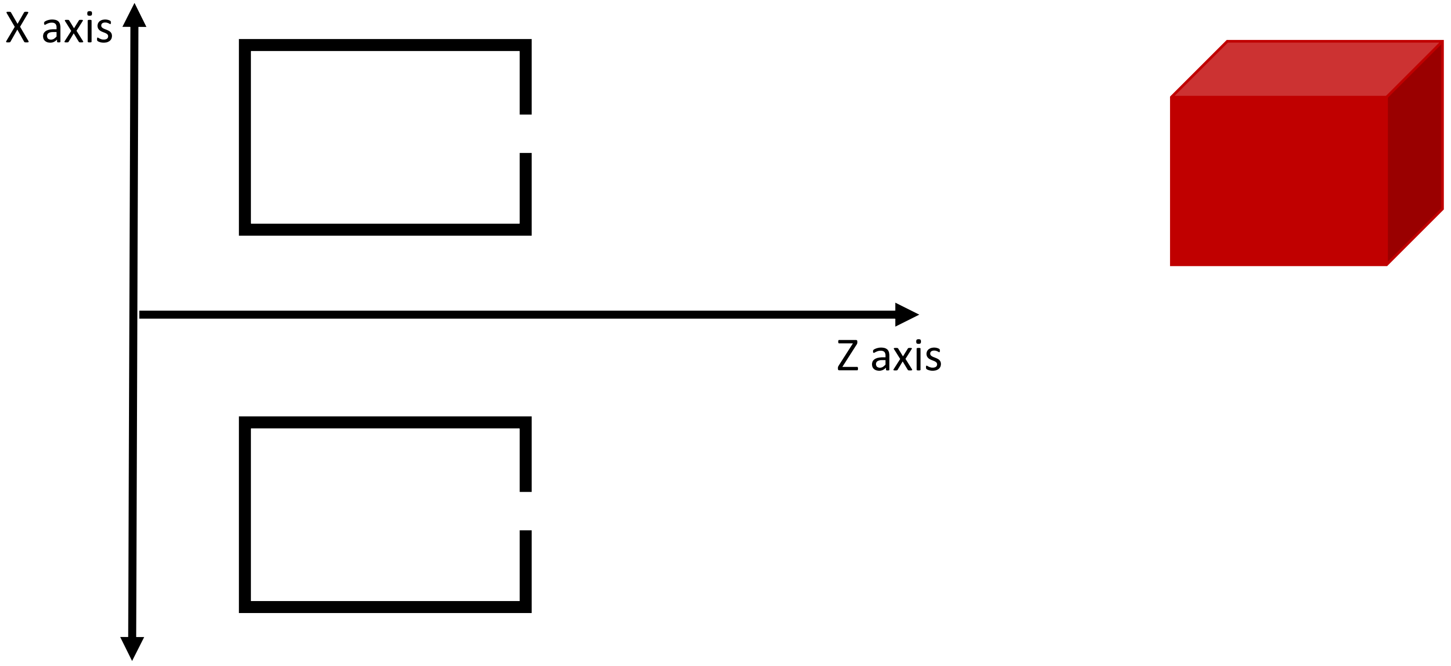
Binocular stereo

- Special case: cameras are parallel to each other and translated along X axis

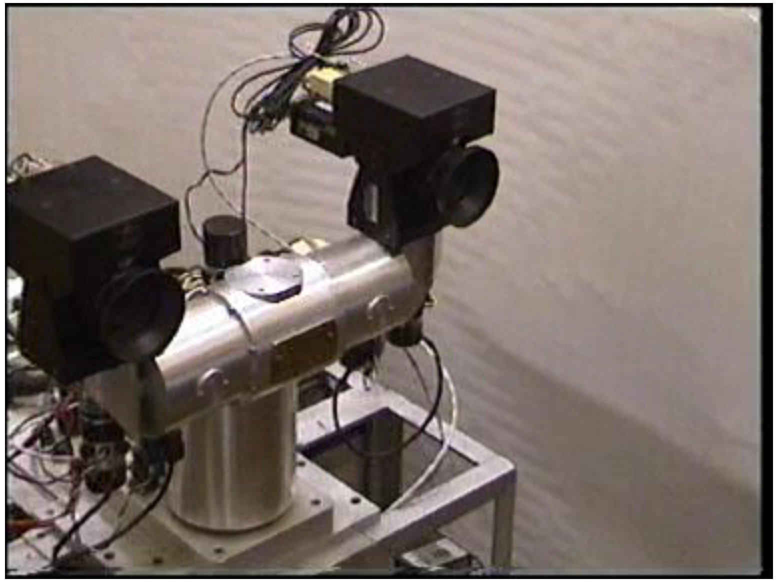


Stereo with *rectified cameras*

- Special case: cameras are parallel to each other and translated along X axis

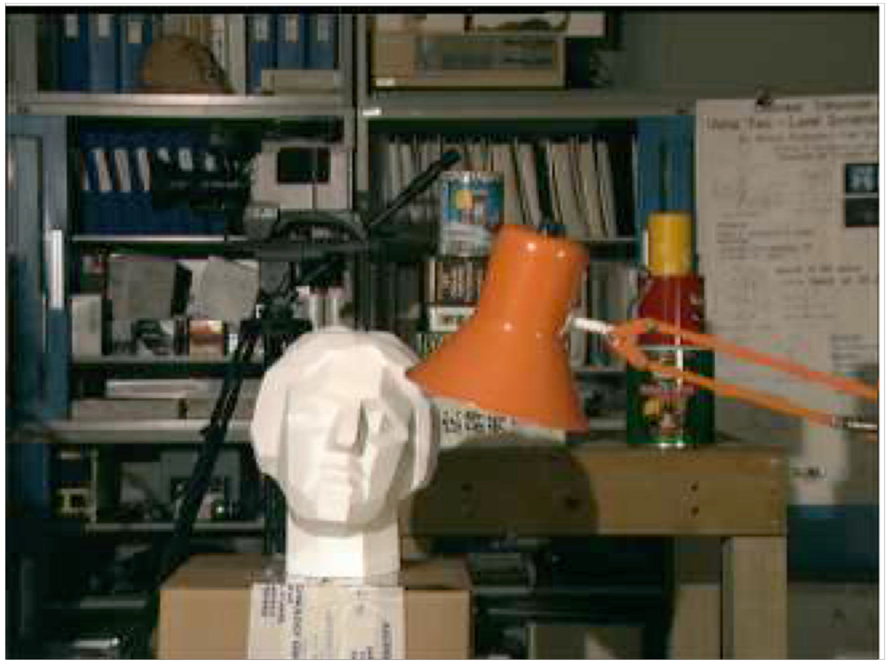


Stereo head



Kinect / depth cameras

Stereo with “rectified cameras”



Perspective projection in rectified cameras

- Without loss of generality, assume origin is at pinhole of 1st camera

$$\vec{\mathbf{x}}_{img}^{(1)} \equiv \begin{bmatrix} I & \mathbf{0} \end{bmatrix} \vec{\mathbf{x}}_w$$

$$\vec{\mathbf{x}}_{img}^{(2)} \equiv \begin{bmatrix} I & \mathbf{t} \end{bmatrix} \vec{\mathbf{x}}_w$$

$$\mathbf{t} = \begin{bmatrix} t_x \\ 0 \\ 0 \end{bmatrix}$$

Perspective projection in rectified cameras

- Without loss of generality, assume origin is at pinhole of 1st camera

$$\vec{\mathbf{x}}_{img}^{(1)} \equiv \begin{bmatrix} I & \mathbf{0} \end{bmatrix} \vec{\mathbf{x}}_w$$

$$\vec{\mathbf{x}}_{img}^{(2)} \equiv \begin{bmatrix} I & \mathbf{t} \end{bmatrix} \vec{\mathbf{x}}_w$$

$$\mathbf{t} = \begin{bmatrix} t_x \\ 0 \\ 0 \end{bmatrix} \quad \vec{\mathbf{x}}_w = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{x}_w \\ 1 \end{bmatrix}$$

Perspective projection in rectified cameras

- Without loss of generality, assume origin is at pinhole of 1st camera

$$\vec{\mathbf{x}}_{img}^{(1)} \equiv \begin{bmatrix} I & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x}_w \\ 1 \end{bmatrix} = \mathbf{x}_w = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$\vec{\mathbf{x}}_{img}^{(2)} \equiv \begin{bmatrix} I & \mathbf{t} \end{bmatrix} \begin{bmatrix} \mathbf{x}_w \\ 1 \end{bmatrix} = \mathbf{x}_w + \mathbf{t} = \begin{bmatrix} X + t_x \\ Y \\ Z \end{bmatrix}$$

Perspective projection in rectified cameras

- Without loss of generality, assume origin is at pinhole of 1st camera

$$\vec{\mathbf{x}}_{img}^{(1)} \equiv \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$\vec{\mathbf{x}}_{img}^{(2)} \equiv \begin{bmatrix} X + t_x \\ Y \\ Z \end{bmatrix}$$

Perspective projection in rectified cameras

- Without loss of generality, assume origin is at pinhole of 1st camera

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \equiv \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} \equiv \begin{bmatrix} X + t_x \\ Y \\ Z \end{bmatrix}$$

Perspective projection in rectified cameras

- Without loss of generality, assume origin is at pinhole of 1st camera

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \equiv \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} \equiv \begin{bmatrix} X + t_x \\ Y \\ Z \end{bmatrix}$$

Perspective projection in rectified cameras

- Without loss of generality, assume origin is at pinhole of 1st camera

$$\begin{bmatrix} \lambda x_1 \\ \lambda y_1 \\ \lambda \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$\begin{bmatrix} \lambda x_2 \\ \lambda y_2 \\ \lambda \end{bmatrix} = \begin{bmatrix} X + t_x \\ Y \\ Z \end{bmatrix}$$

Perspective projection in rectified cameras

- Without loss of generality, assume origin is at pinhole of 1st camera

X coordinate differs by t_x/Z

$$x_1 = \frac{X}{Z} \qquad x_2 = \frac{X + t_x}{Z}$$

$$y_1 = \frac{Y}{Z} \qquad y_2 = \frac{Y}{Z}$$

Y coordinate is the same!

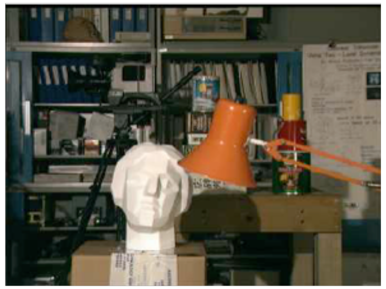
Perspective projection in rectified cameras

- X coordinate differs by t_x/Z
- That is, difference in X coordinate is *inversely proportional to depth*
- Difference in X coordinate is called *disparity*
- Translation between cameras (tx) is called *baseline*

- *disparity = baseline / depth*

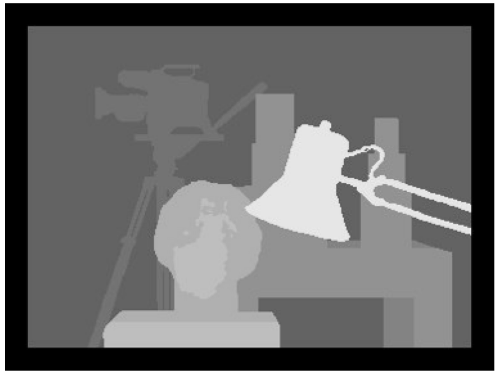
The disparity image

- For pixel (x,y) in one image, only need to know disparity to get correspondence
- Create an image with color at (x,y) = disparity



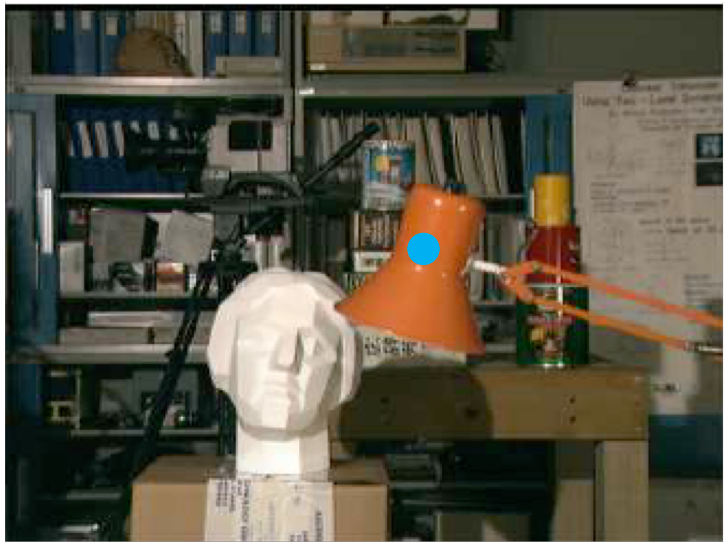
right image

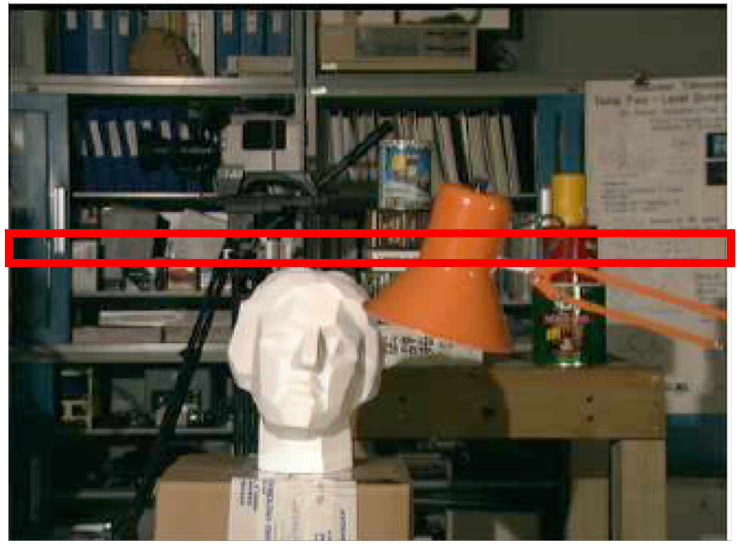
left image



disparity

Perspective projection in rectified cameras





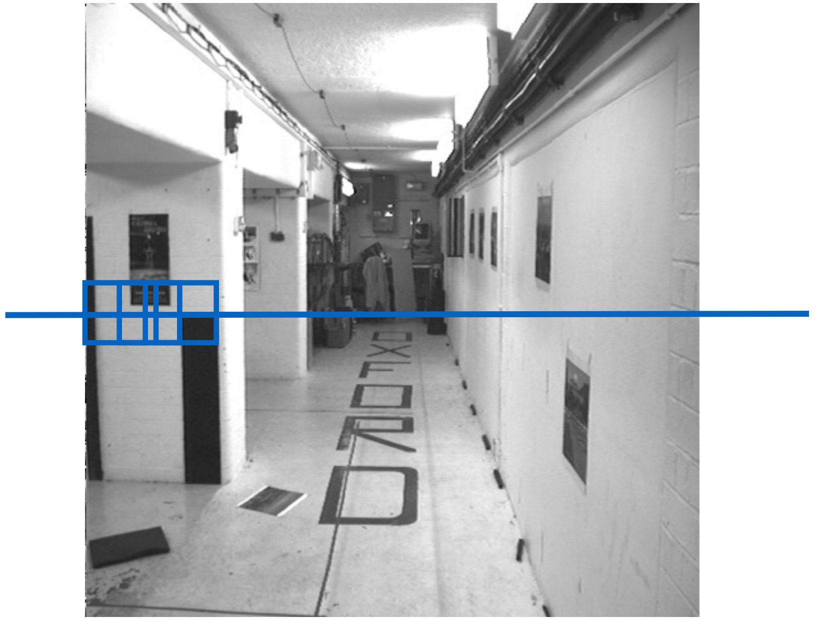
- For rectified cameras, correspondence problem is easier
- Only requires searching along a particular *row.*

NCC - Normalized Cross Correlation

- Lighting and color change pixel intensities
- Example: increase brightness / contrast
- $I' = \alpha I + \beta$
- Subtract patch mean: invariance to β
- Divide by norm of vector: invariance to α
- $x' = x - < x >$
- $x'' = \frac{x'}{||x'||}$
- *similarity* = $x'' \cdot y''$

Why not SIFT?

Cross-correlation of neighborhood



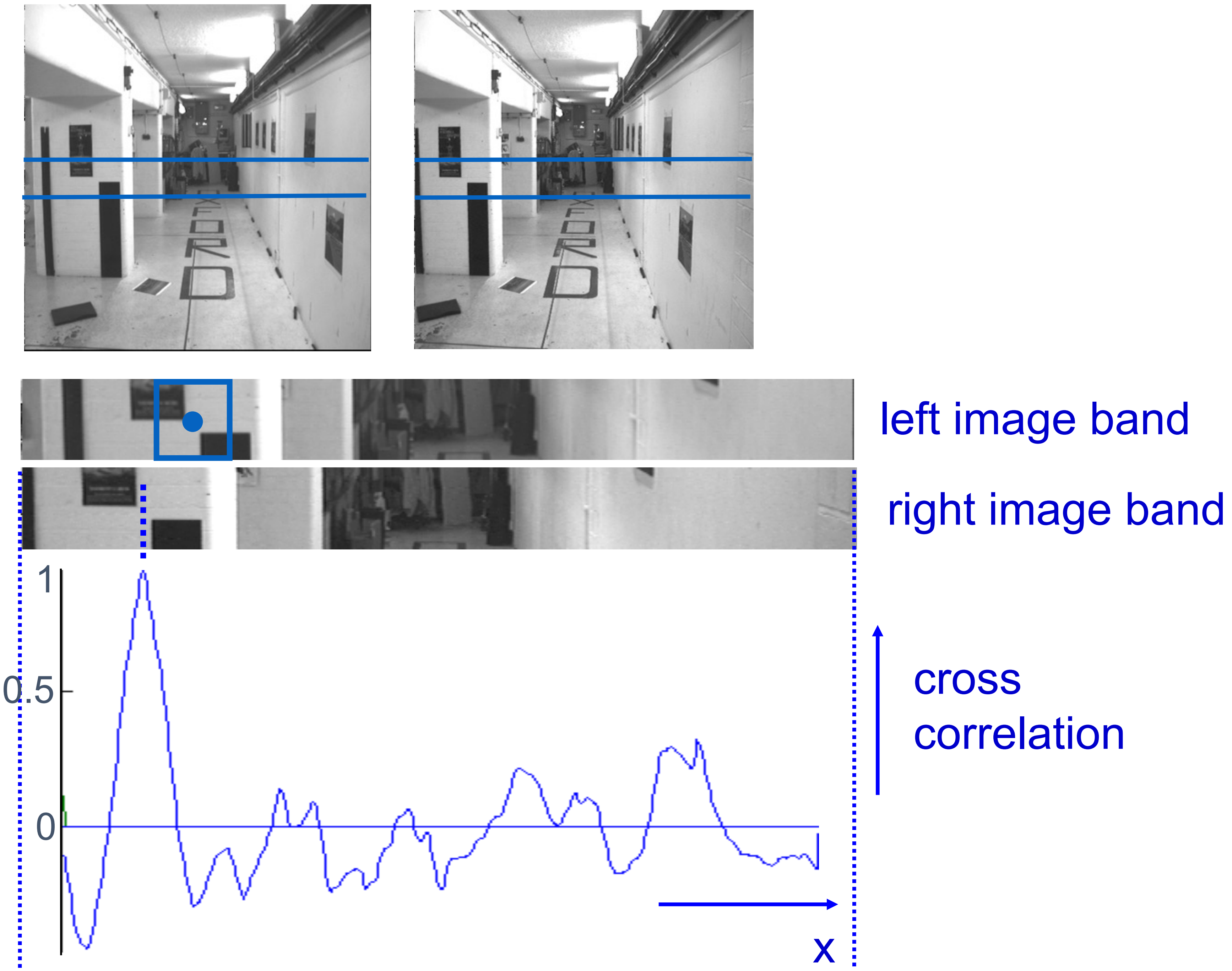
regions $\mathtt{A}, \mathtt{B}$, write as vectors $\mathbf{a}, \mathbf{b}$

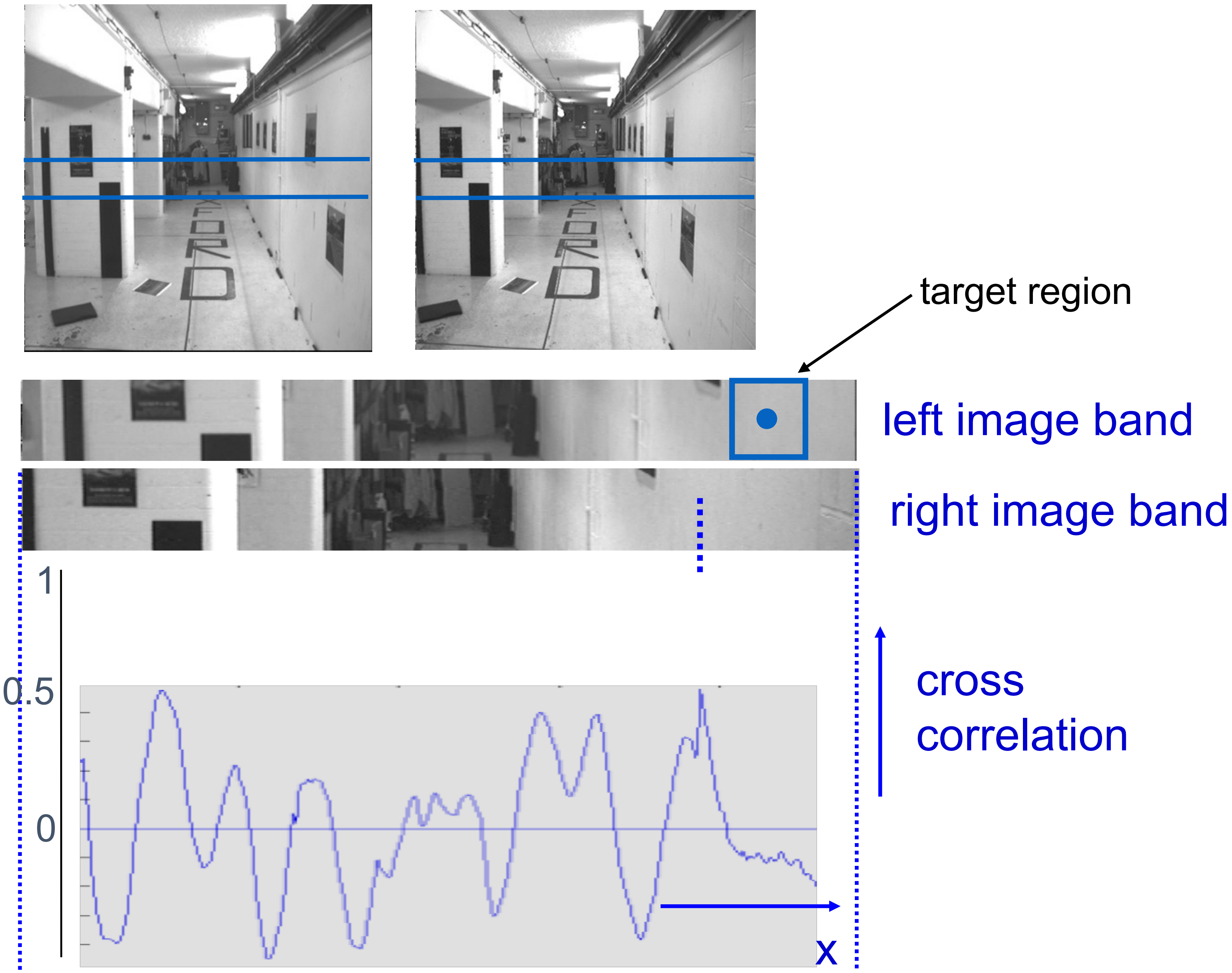
translate so that mean is zero

$$\mathtt{a} \rightarrow \mathtt{a} - \langle \mathbf{a} \rangle, \quad \mathtt{b} \rightarrow \mathtt{b} - \langle \mathbf{b} \rangle$$

$$\text{cross correlation} = \frac{\mathbf{a}.\mathbf{b}}{|\mathbf{a}||\mathbf{b}|}$$

Invariant to $I \rightarrow \alpha I + \beta$


left image band
right image band
1
0.5
0
cross
correlation
x


target region
left image band
right image band
1
0.5
0
x
cross
correlation

The NCC cost volume

- Consider M x N image
- Suppose there are D possible disparities.
- For every pixel, D possible scores
- Can be written as an M x N x D array
- To get disparity, take max along 3rd axis

Computing the NCC volume

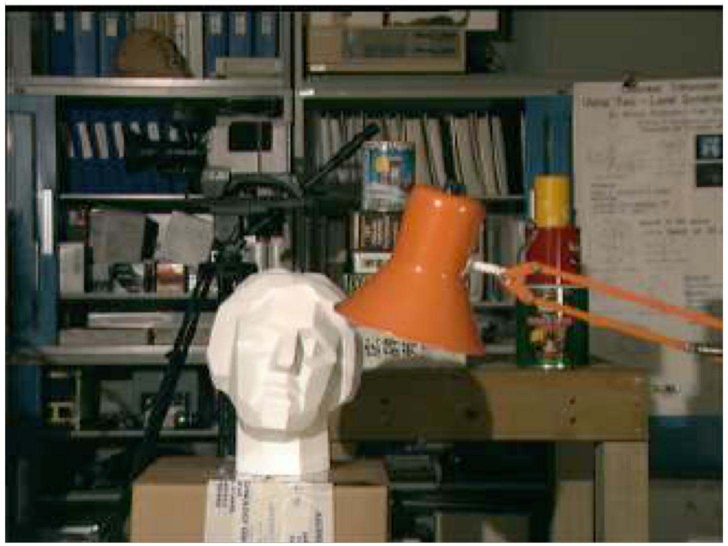
1. For every pixel (x, y)
    1. For every disparity d
        1. Get normalized patch from image 1 at (x, y)
        2. Get normalized patch from image 2 at (x + d, y)
        3. Compute NCC

Computing the NCC volume

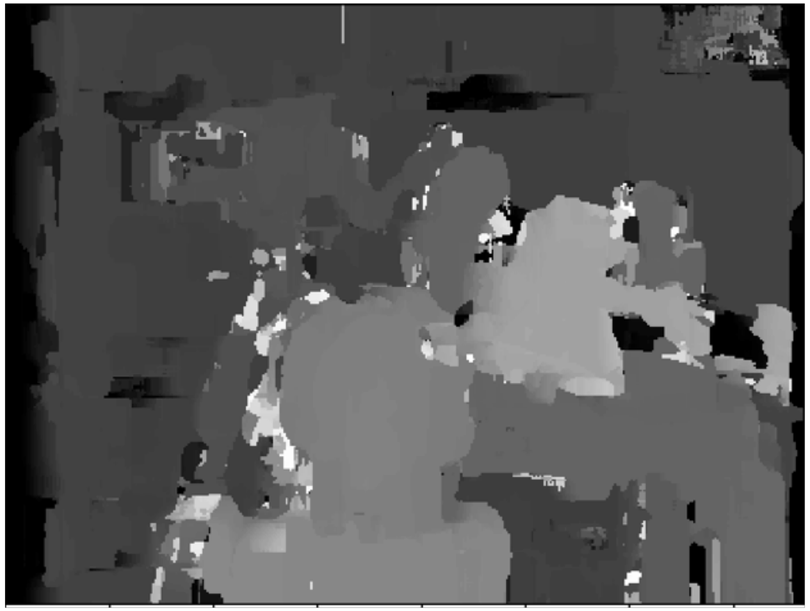
1. For every disparity d
    1. For every pixel (x, y)
        1. Get normalized patch from image 1 at (x, y)
        2. Get normalized patch from image 2 at (x + d, y)
        3. Compute NCC

Assume all pixels lie at same disparity d (i.e., lie on same plane) and compute cost for each

Plane sweep stereo



NCC volume



Disparity

Perspective projection in rectified cameras

- Without loss of generality, assume origin is at pinhole of 1st camera

X coordinate differs by t_x/Z

$$x_1 = \frac{X}{Z} \qquad x_2 = \frac{X + t_x}{Z}$$

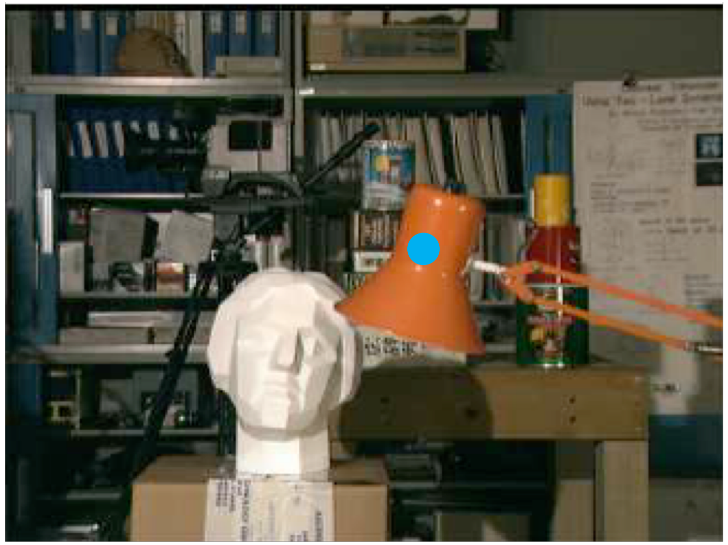
$$y_1 = \frac{Y}{Z} \qquad y_2 = \frac{Y}{Z}$$

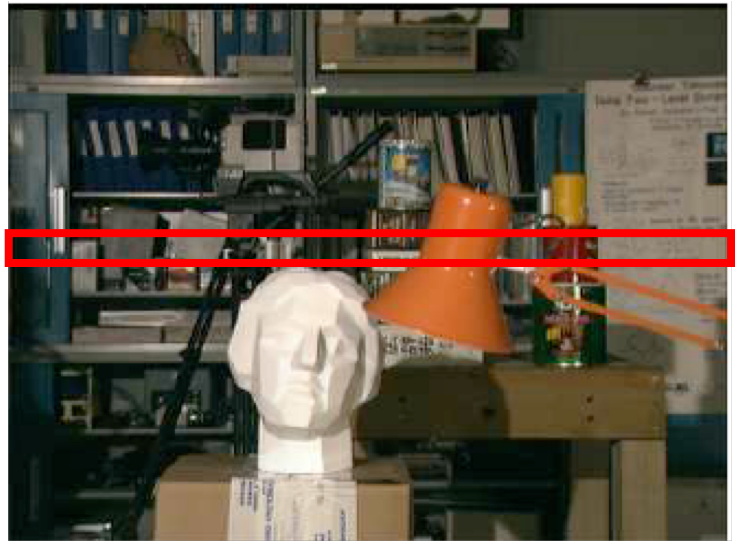
Y coordinate is the same!

Perspective projection in rectified cameras

- disparity = t_x/Z
- If t_x is known, disparity gives Z
- Otherwise, disparity gives Z in units of t_x
  - Small-baseline, near depth = large-baseline, far depth

Perspective projection in rectified cameras

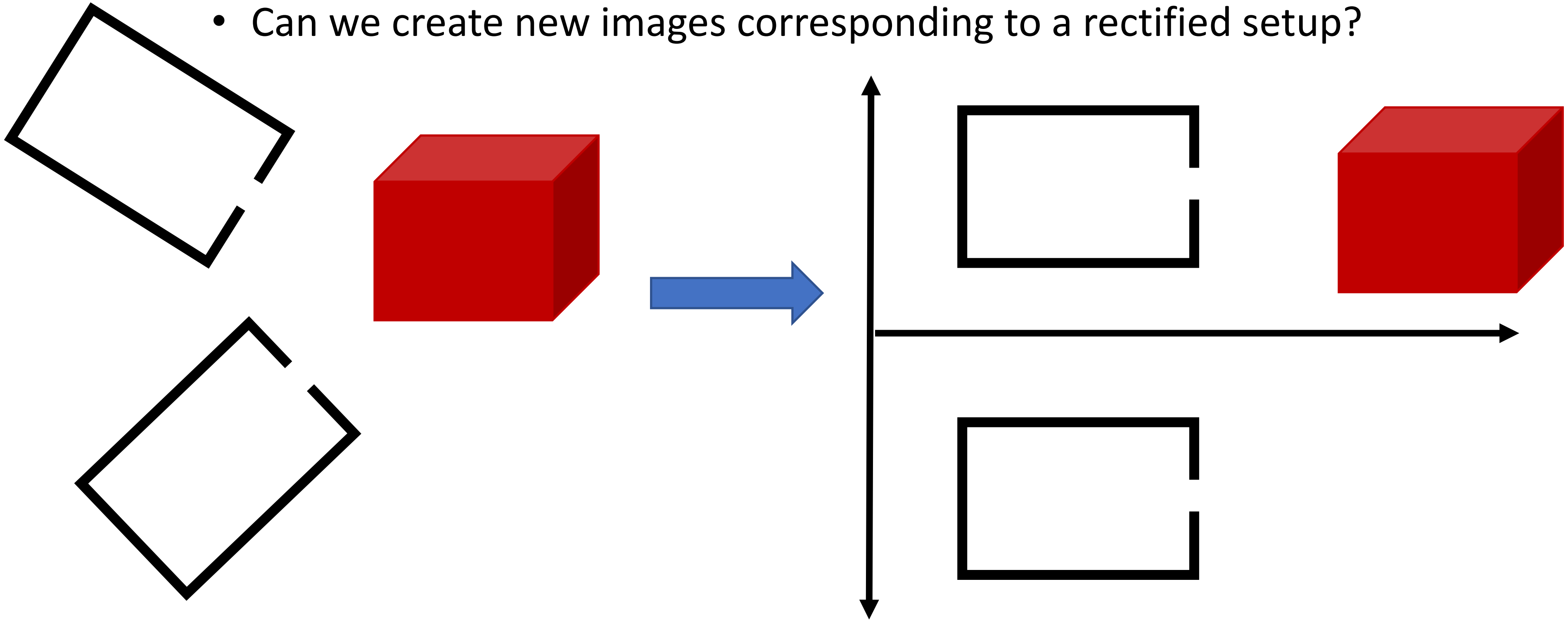




- For rectified cameras, correspondence problem is easier
- Only requires searching along a particular *row.*

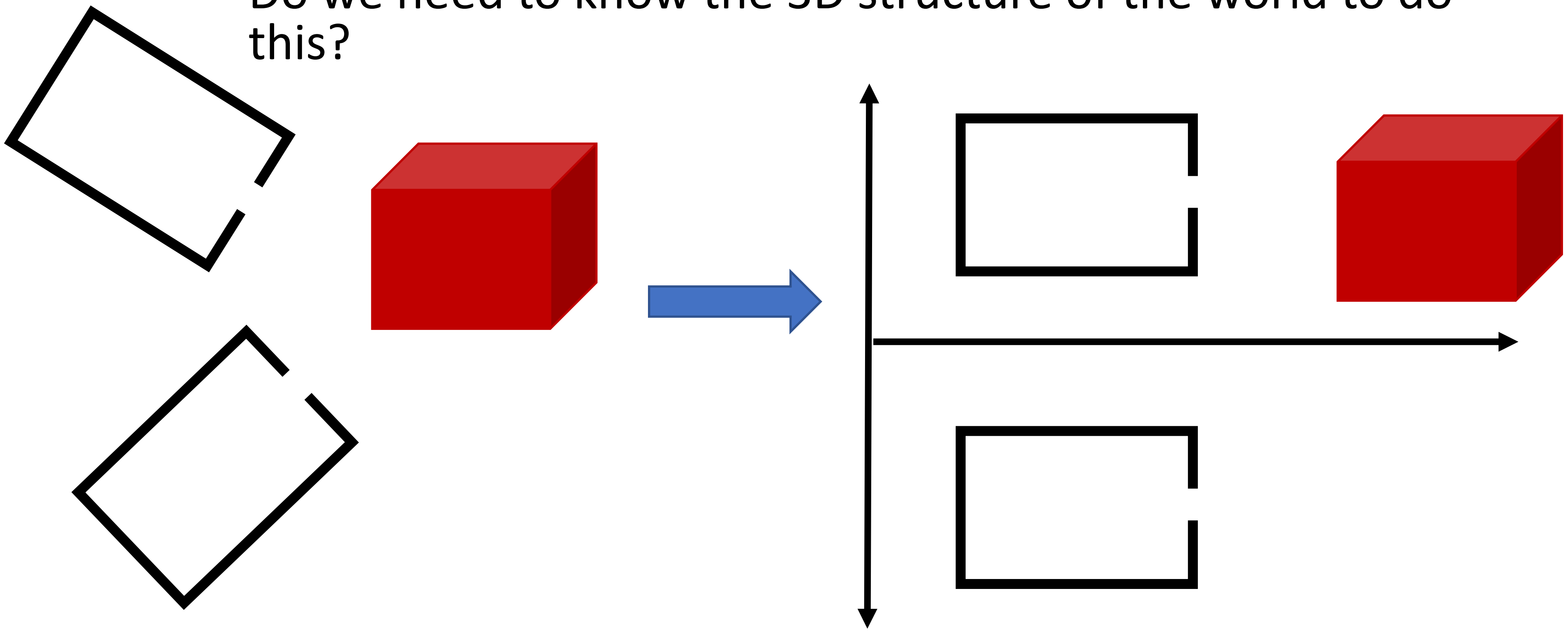
Rectifying cameras

- Given two images from two cameras with known P, can we rectify them?
  - Can we create new images corresponding to a rectified setup?



Rectifying cameras

- Can we rotate / translate cameras?
  - Do we need to know the 3D structure of the world to do this?



Rotating cameras

$$\vec{\mathbf{x}}_{img} \equiv K \begin{bmatrix} R & \mathbf{t} \end{bmatrix} \vec{\mathbf{x}}_w$$

- Assume K is identity
- Assume coordinate system at camera pinhole

$$\begin{aligned} \vec{\mathbf{x}}_{img} &\equiv \begin{bmatrix} I & \mathbf{0} \end{bmatrix} \vec{\mathbf{x}}_w \\ &\equiv \begin{bmatrix} I & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x}_w \\ 1 \end{bmatrix} \\ &\equiv \mathbf{x}_w \end{aligned}$$

Rotating cameras

$$\vec{\mathbf{x}}_{img} \equiv K \begin{bmatrix} R & \mathbf{t} \end{bmatrix} \vec{\mathbf{x}}_w$$

- Assume K is identity
- Assume coordinate system at camera pinhole

$$\begin{aligned} \vec{\mathbf{x}}_{img} &\equiv \begin{bmatrix} I & \mathbf{0} \end{bmatrix} \vec{\mathbf{x}}_w \\ &\equiv \begin{bmatrix} I & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x}_w \\ 1 \end{bmatrix} \\ &\equiv \mathbf{x}_w \end{aligned}$$

Rotating cameras

$$\vec{\mathbf{x}}_{img} \equiv \begin{bmatrix} I & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x}_w \\ 1 \end{bmatrix}$$

$$\vec{\mathbf{x}}_{img} \equiv \mathbf{x}_w$$

- What happens if the camera is rotated?

$$\vec{\mathbf{x}}'_{img} \equiv \begin{bmatrix} R & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x}_w \\ 1 \end{bmatrix}$$

$$\equiv R\mathbf{x}_w$$

$$\equiv R\vec{\mathbf{x}}_{img}$$

Rotating cameras

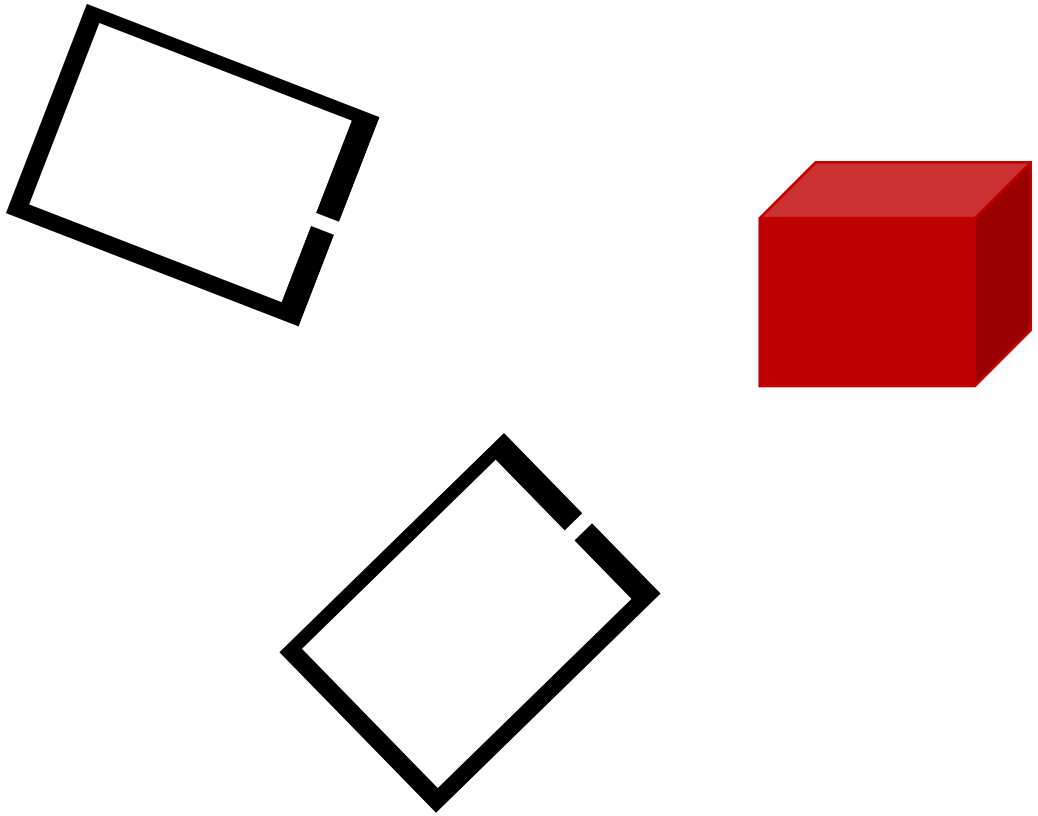
- What happens if the camera is rotated?

$$\underbrace{\vec{\mathbf{x}}'_{img}}_{\text{Homogenous coordinates of mapped pixel}} \equiv \overbrace{R}^{\text{Rotation matrix}} \underbrace{\vec{\mathbf{x}}_{img}}_{\text{Homogenous coordinates of original pixel}}$$

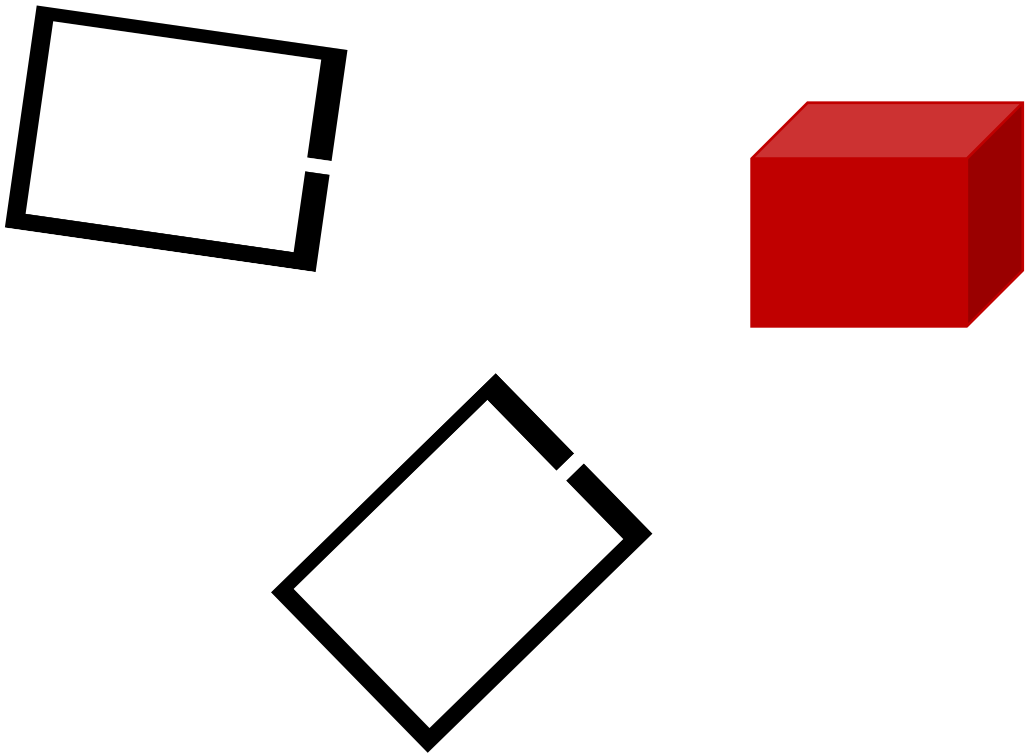
- No need to know the 3D structure

Rotating cameras

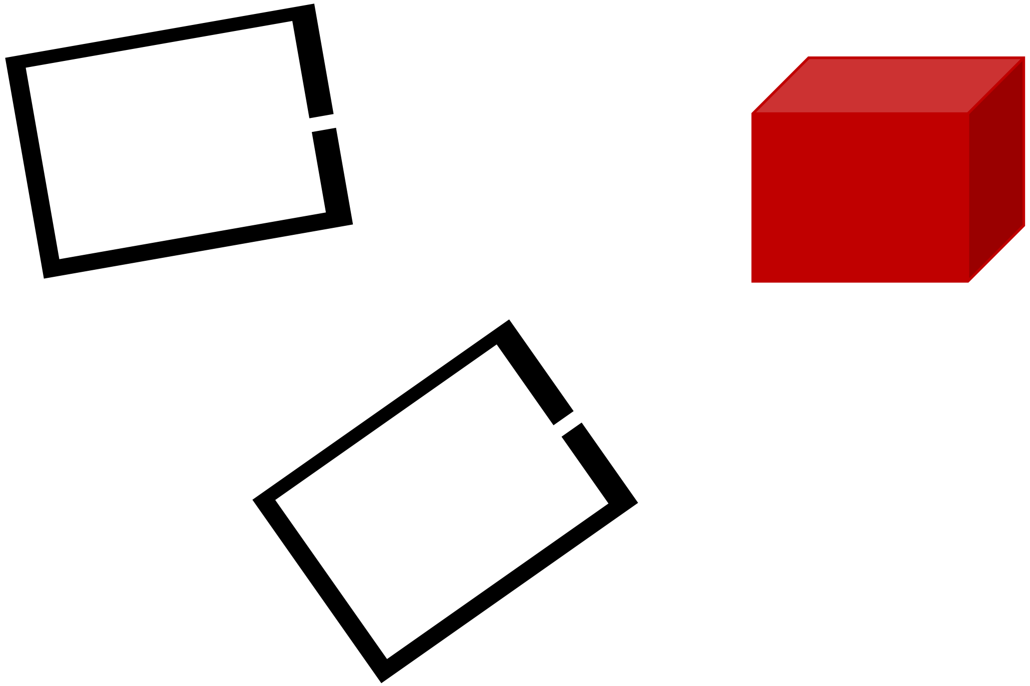
Rectifying cameras



Rectifying cameras



Rectifying cameras



Rectifying cameras