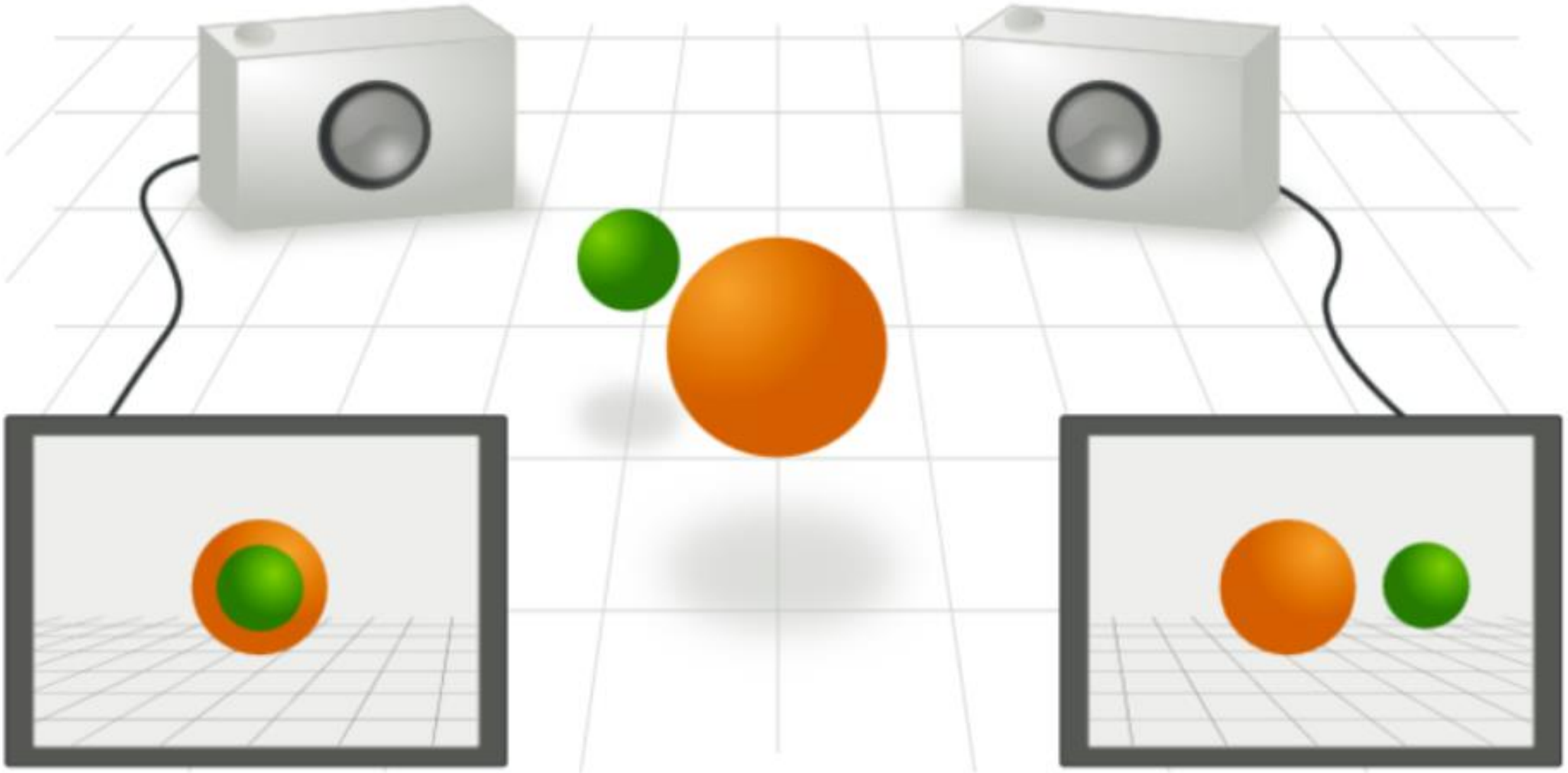
EBU7240

Computer Vision

- Stereo: Epipolar geometry -

Semester 1, 2021

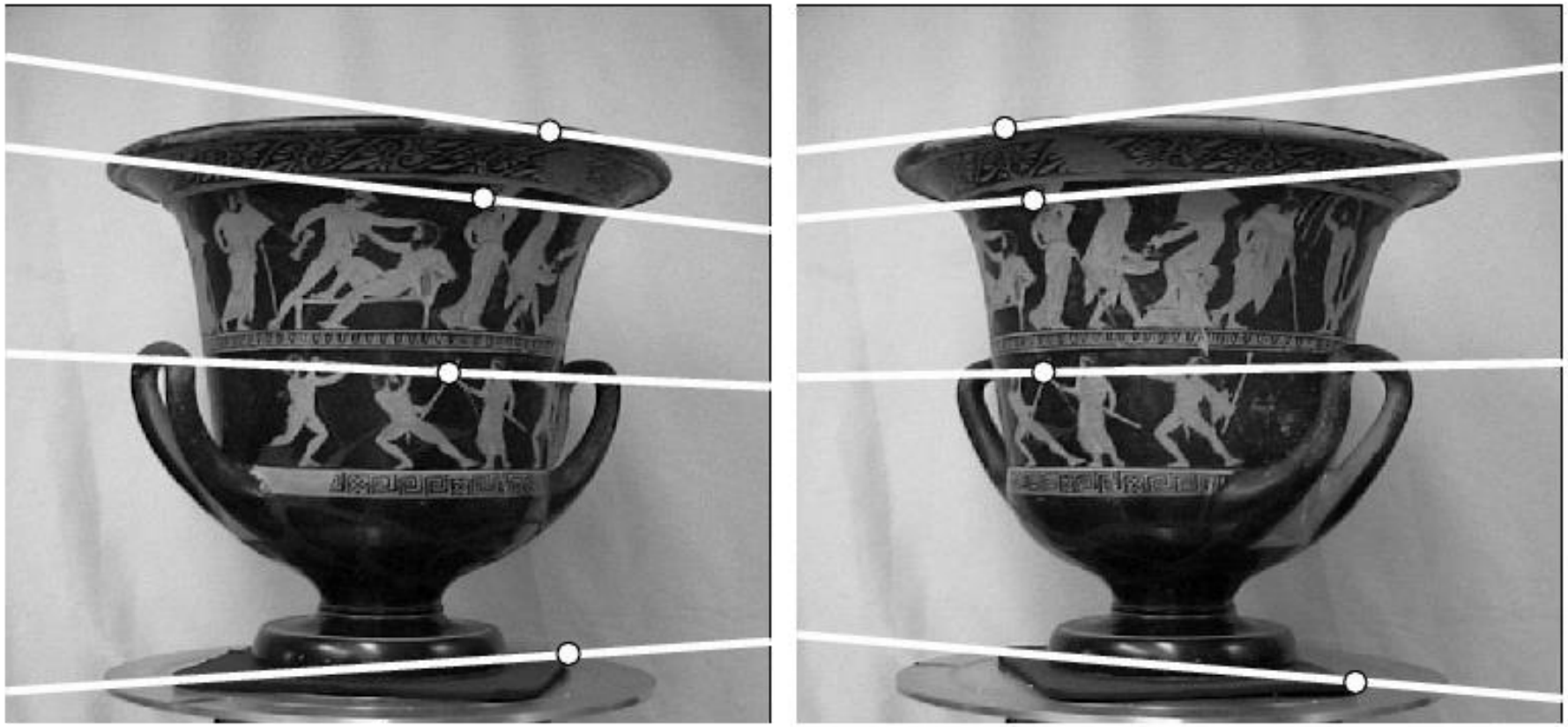
Changjae Oh



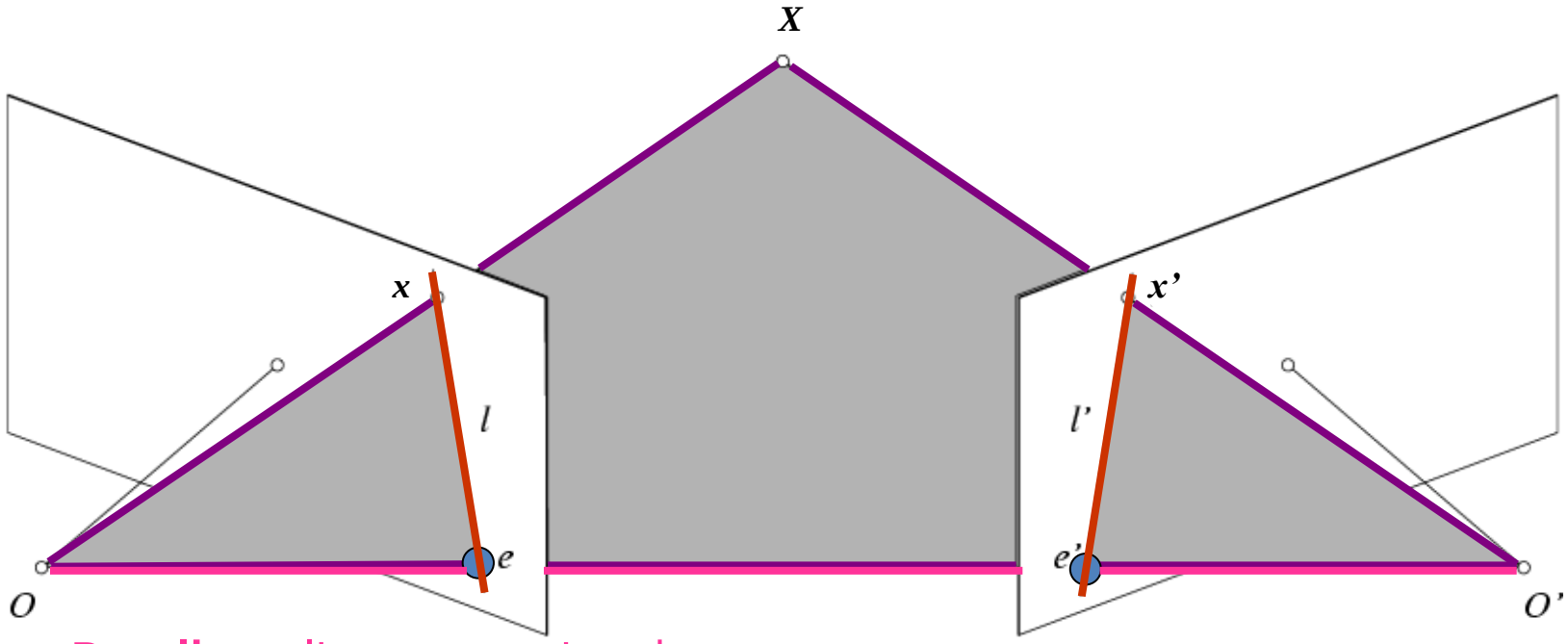
Objectives

- To understand epipolar geometry for stereo vision
- To understand epipolar constraint in calibrated/uncalibrated case

Two-view geometry



Epipolar geometry

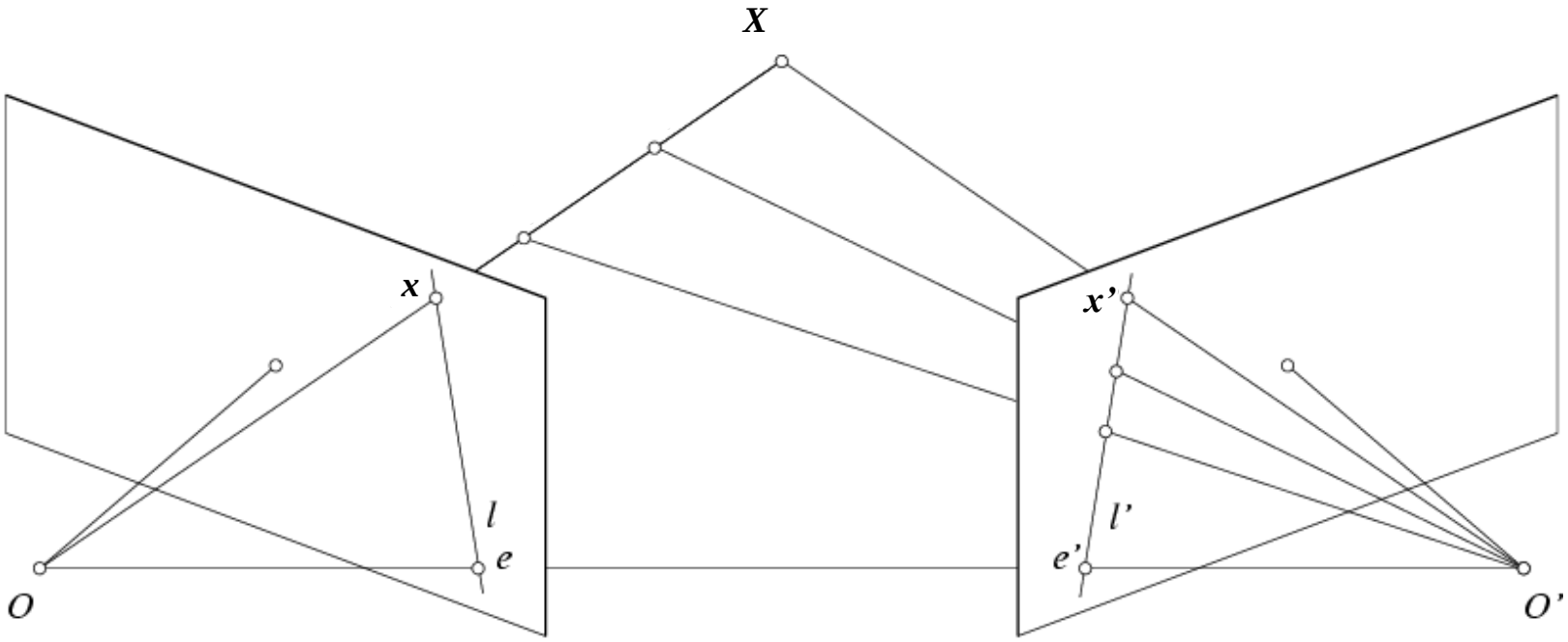


- **Baseline** – line connecting the two camera centers
- **Epipolar Plane** – plane containing baseline (1D family)
- **Epipoles**

= intersections of baseline with image planes
= projections of the other camera center
= vanishing points of the motion direction

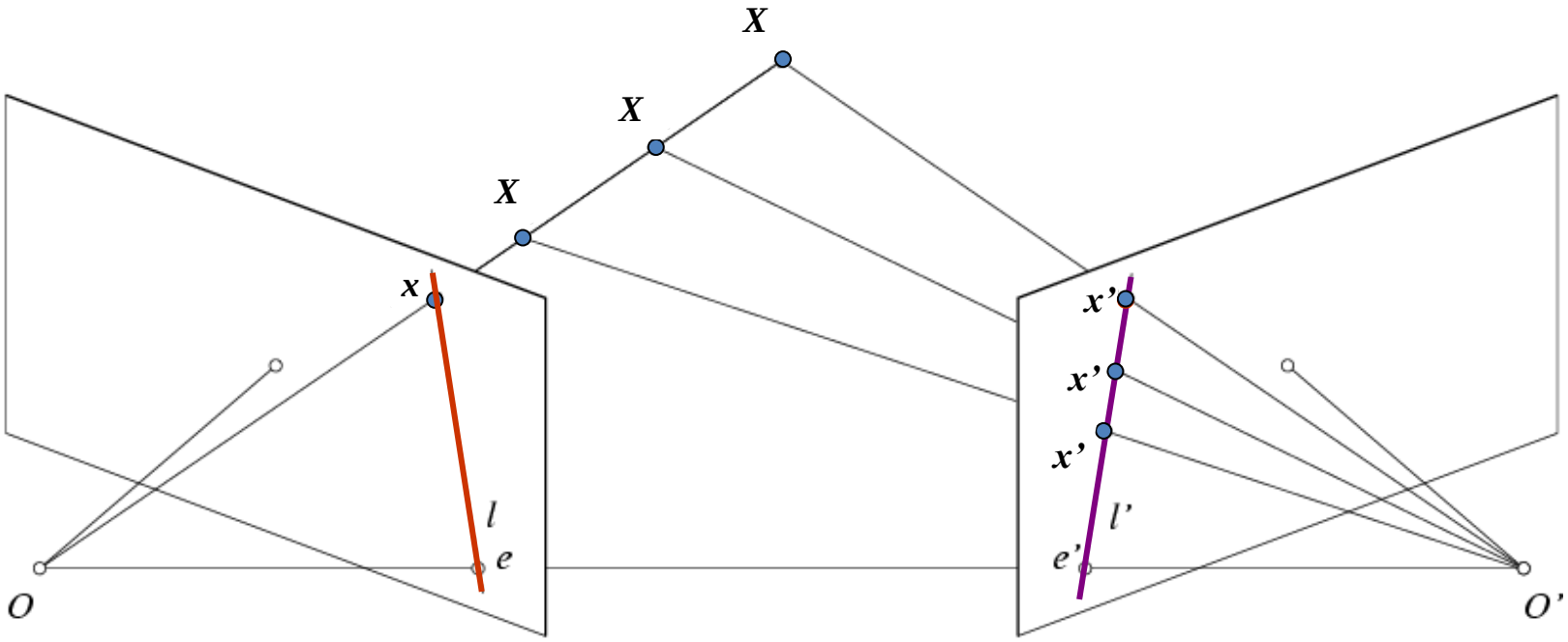
- **Epipolar Lines** - intersections of epipolar plane with image planes (always come in corresponding pairs)

Epipolar constraint



- If we observe a point ***x*** in one image, where can the corresponding point ***x'*** be in the other image?

Epipolar constraint



- Potential matches for ***x*** have to lie on the corresponding epipolar line ***l'***.

- Potential matches for ***x'*** have to lie on the corresponding epipolar line ***l***.

Epipolar constraint

- **An important concept for stereo vision**

Task: Match point in the left image to point in the right image

How would you do it?
Block matching to the entire image? (EBU6230)

Credit: K. Kitani

Epipolar constraint

- **An important concept for stereo vision**

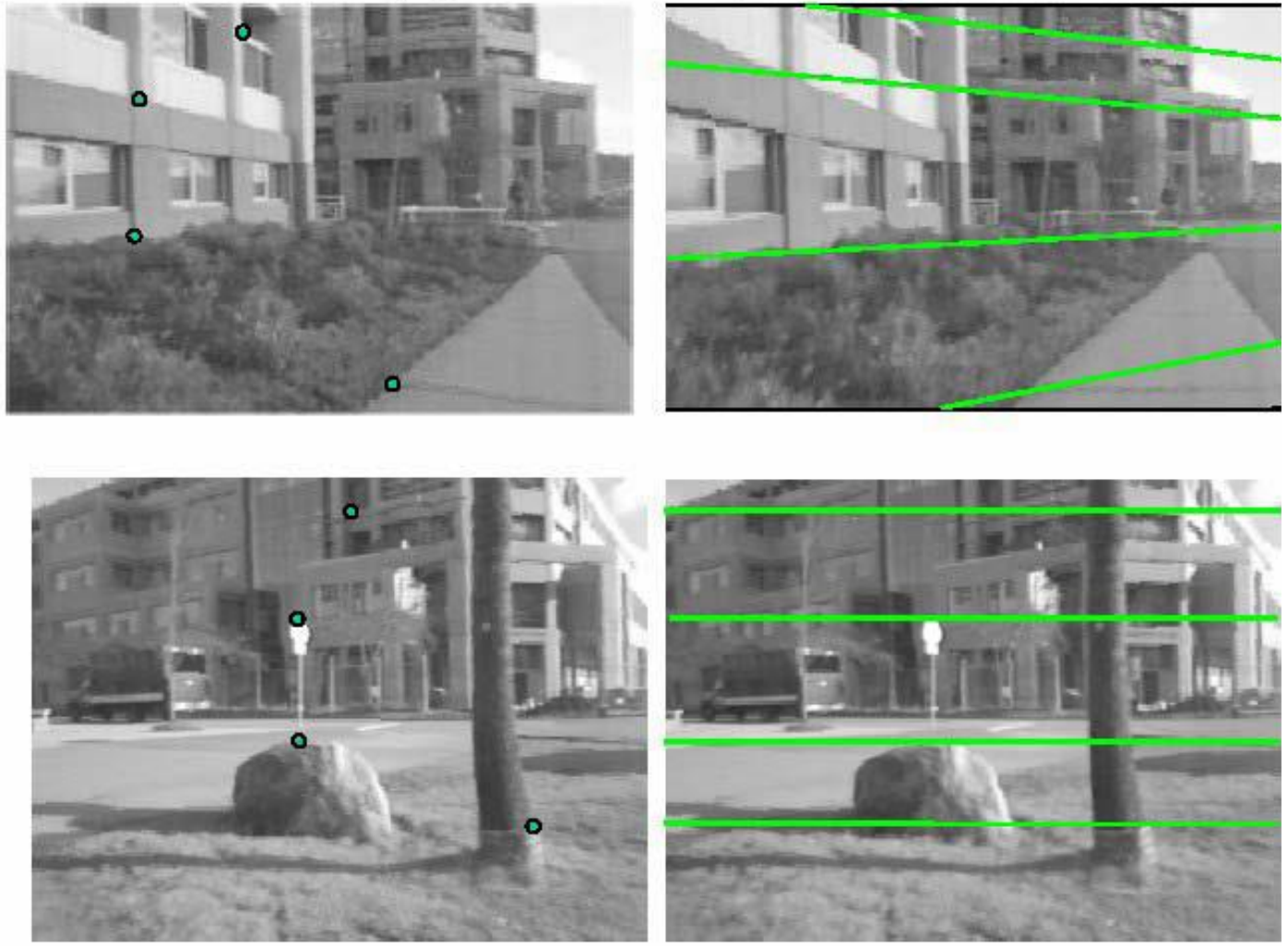
Task: Match point in the left image to point in the right image

Want to avoid search over entire image

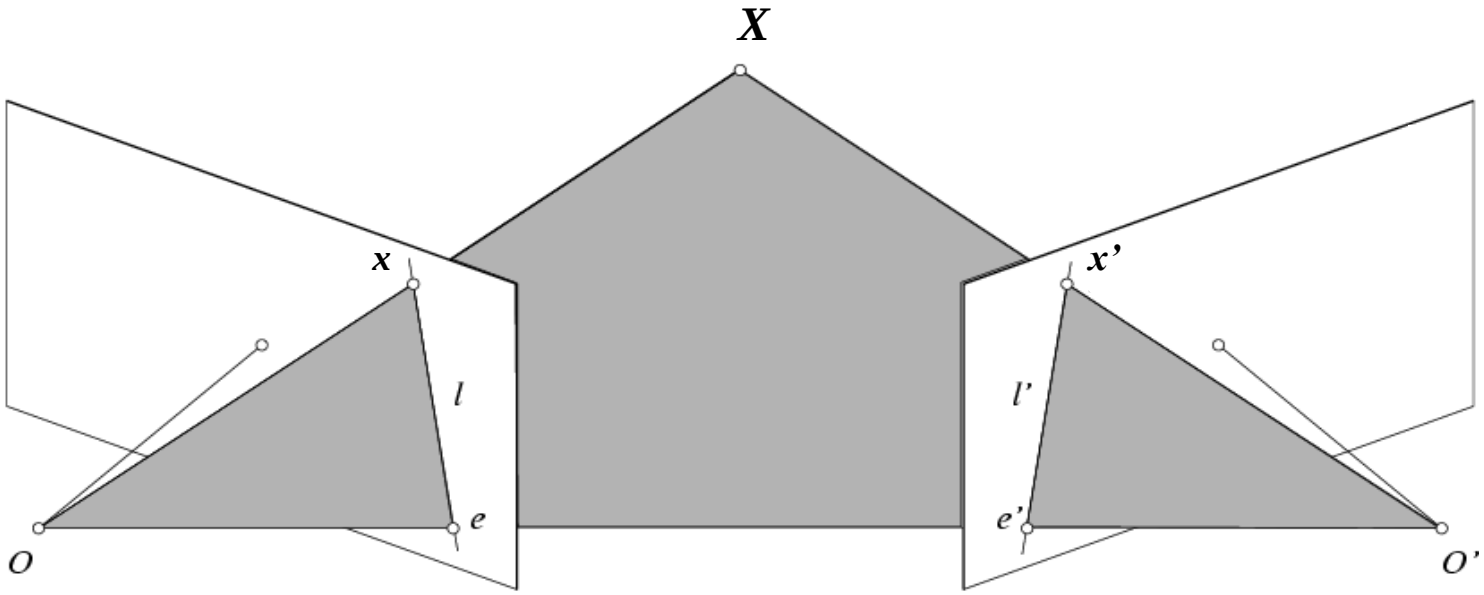
(if the images have been rectified)
Epipolar constraint reduces search to a single line

Epipolar constraint example

- Epipolar constraint reduces search range to a single line



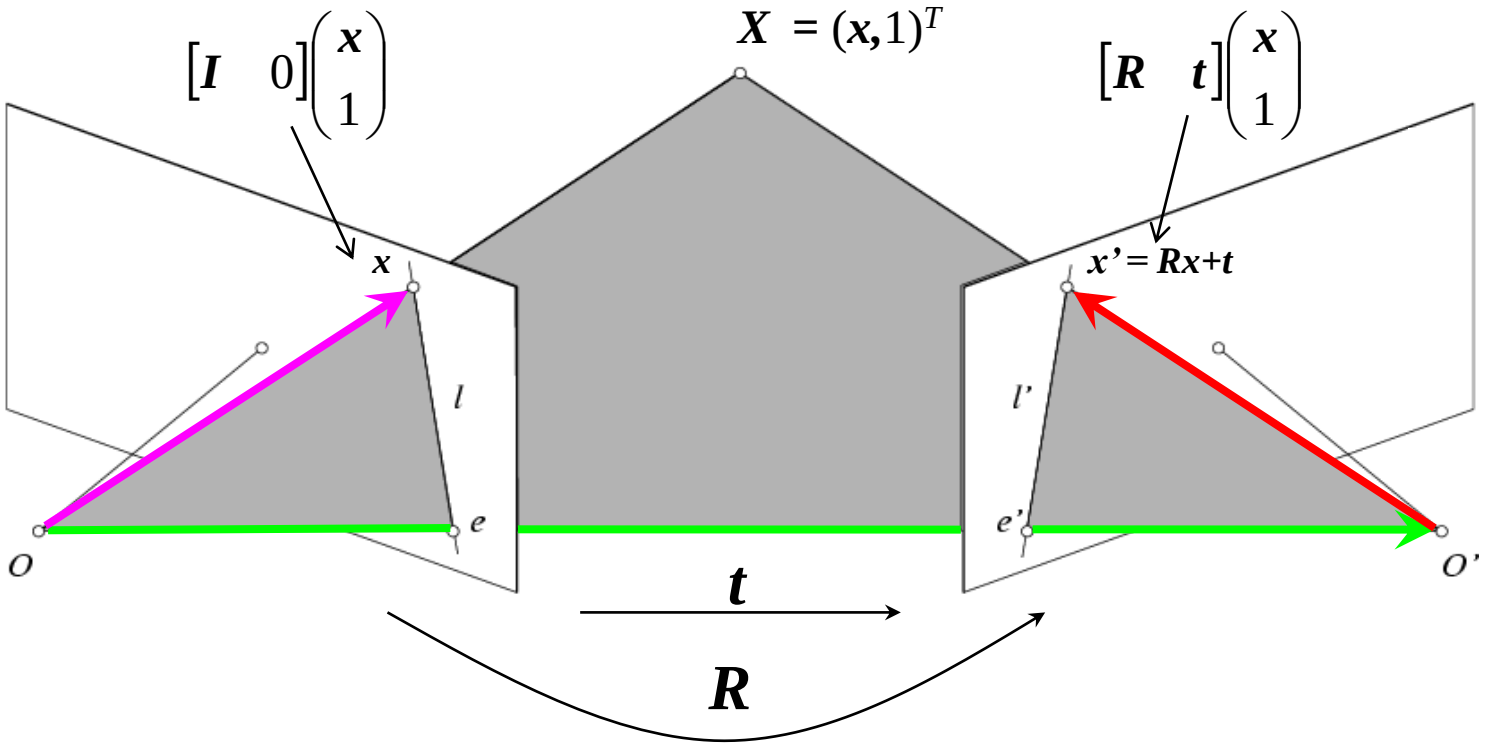
Epipolar constraint: Calibrated case



- Intrinsic and extrinsic parameters of the cameras are known, world coordin ate system is set to that of the first camera
- Then the projection matrices are given by $\boldsymbol{K}[\boldsymbol{I} \mid \boldsymbol{0}]$ and $\boldsymbol{K}'[\boldsymbol{R} \mid \boldsymbol{t}]$
- We can multiply the projection matrices (and the image points) by the inve rse of the calibration matrices to get *normalized* image coordinates:

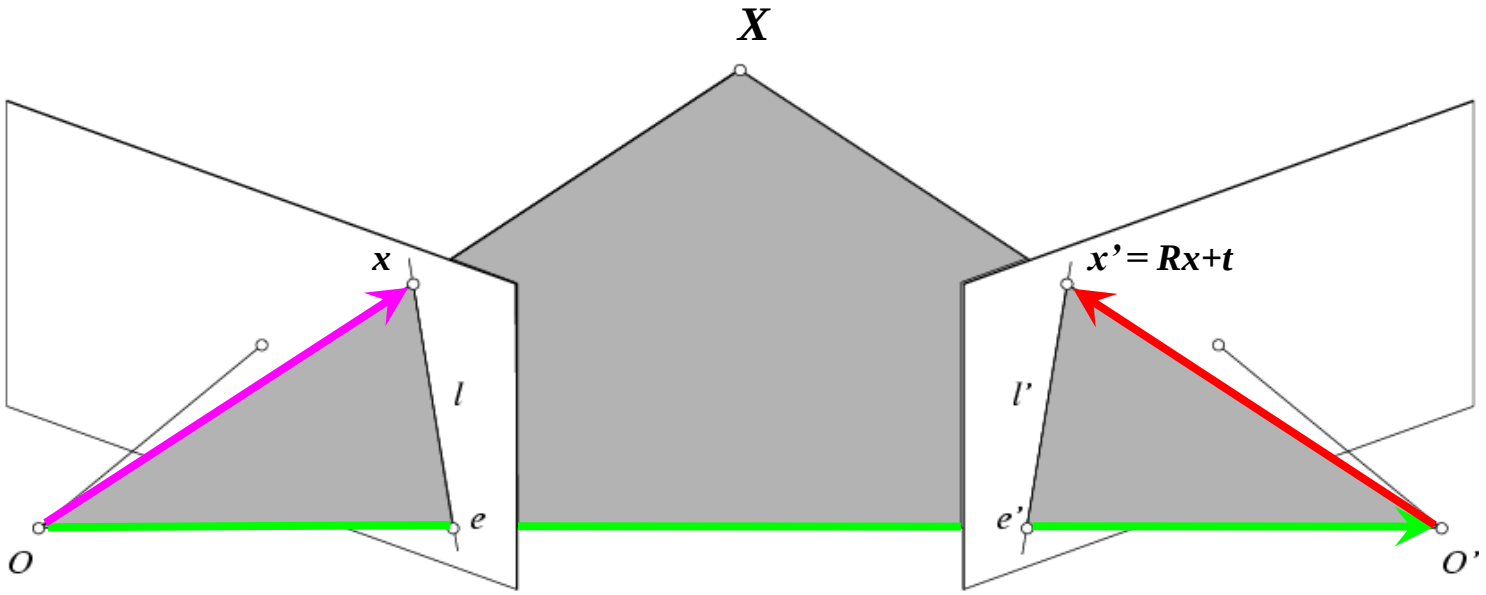
$$\boldsymbol{x}_{\text{norm}} = \boldsymbol{K}^{-1}\boldsymbol{x}_{\text{pixel}} = [\boldsymbol{I} \;\; 0]\,\boldsymbol{X}, \qquad \boldsymbol{x}'_{\text{norm}} = \boldsymbol{K}'^{-1}\boldsymbol{x}'_{\text{pixel}} = [\boldsymbol{R} \;\; \boldsymbol{t}]\,\boldsymbol{X}$$

Epipolar constraint: Calibrated case



The vectors ***Rx***, ***t***, and ***x'*** are coplanar

Epipolar constraint: Calibrated case

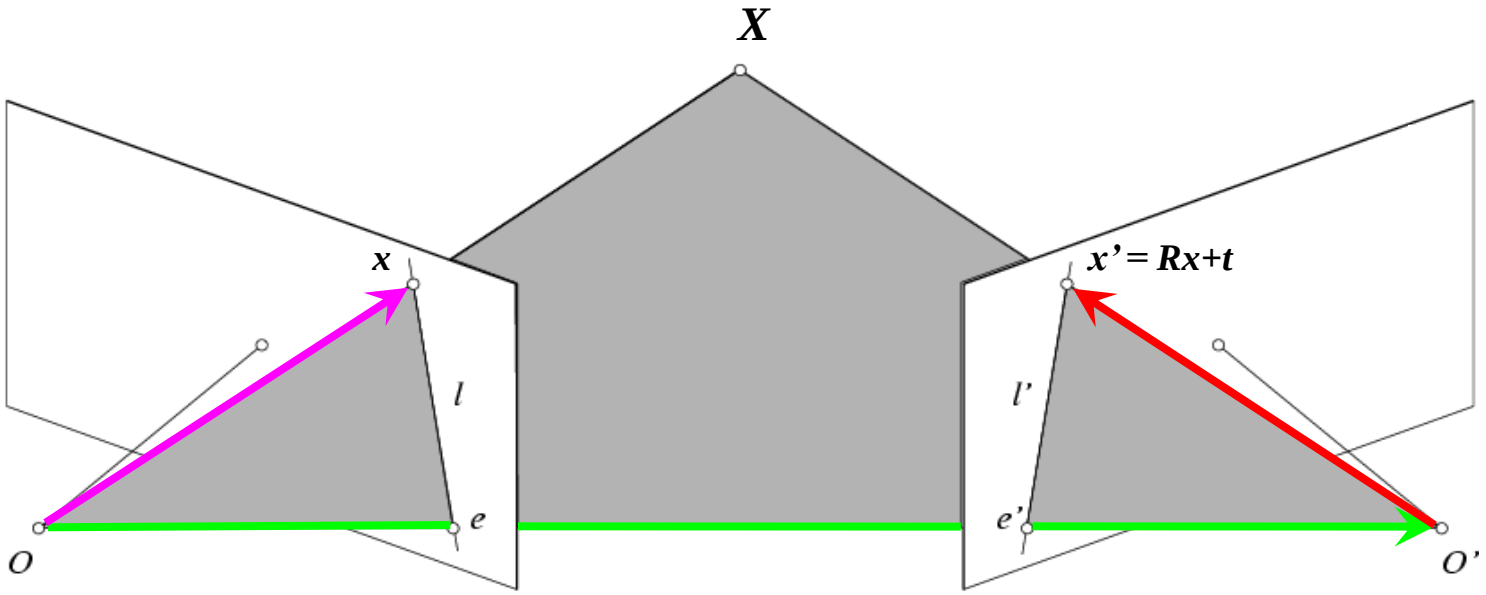


$$x' \cdot [t \times (Rx)] = 0 \quad \Rightarrow \quad x'^{T}[t_{\times}]Rx = 0$$

Recall: $\mathbf{a} \times \mathbf{b} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = [\mathbf{a}_{\times}]\mathbf{b}$

The vectors *Rx*, *t*, and *x'* are coplanar

Epipolar constraint: Calibrated case

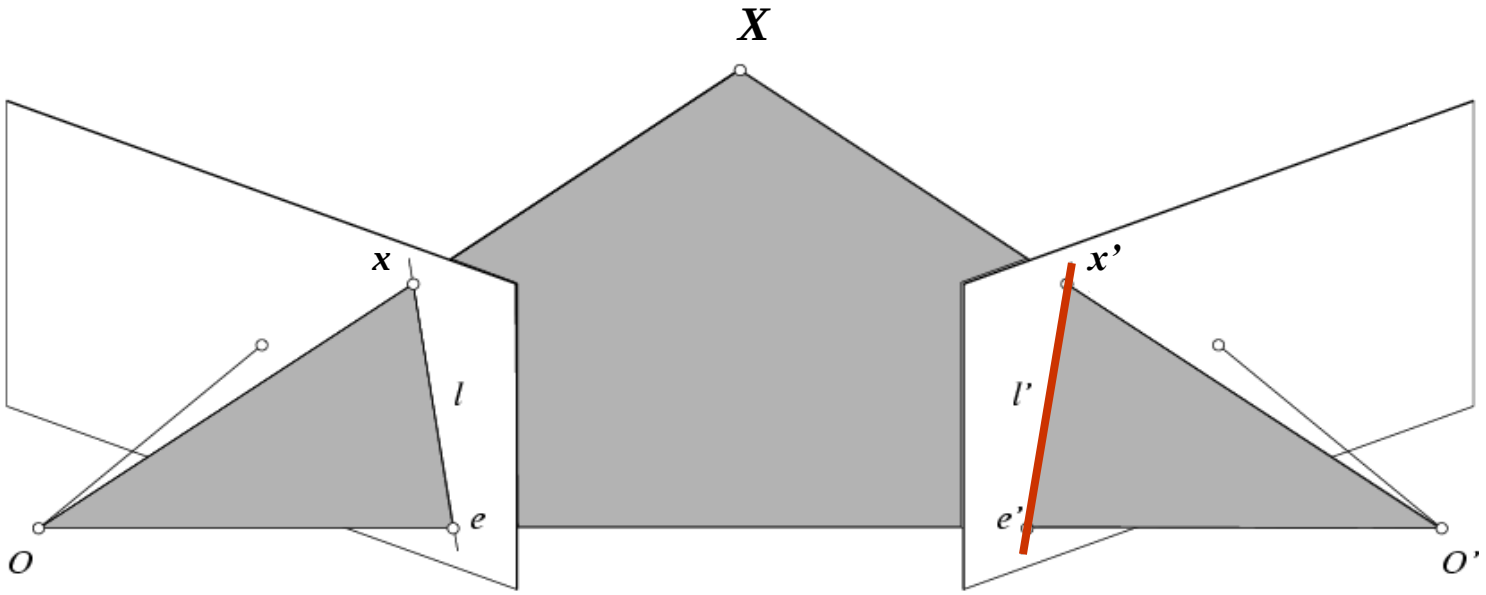


$$x' \cdot [t \times (Rx)] = 0 \quad \Rightarrow \quad x'^T [t_\times] R x = 0 \quad \Rightarrow \quad x'^T E x = 0$$

Essential Matrix
(Longuet-Higgins, 1981)

The vectors ***Rx***, ***t***, and ***x'*** are coplanar

Epipolar constraint: Calibrated case

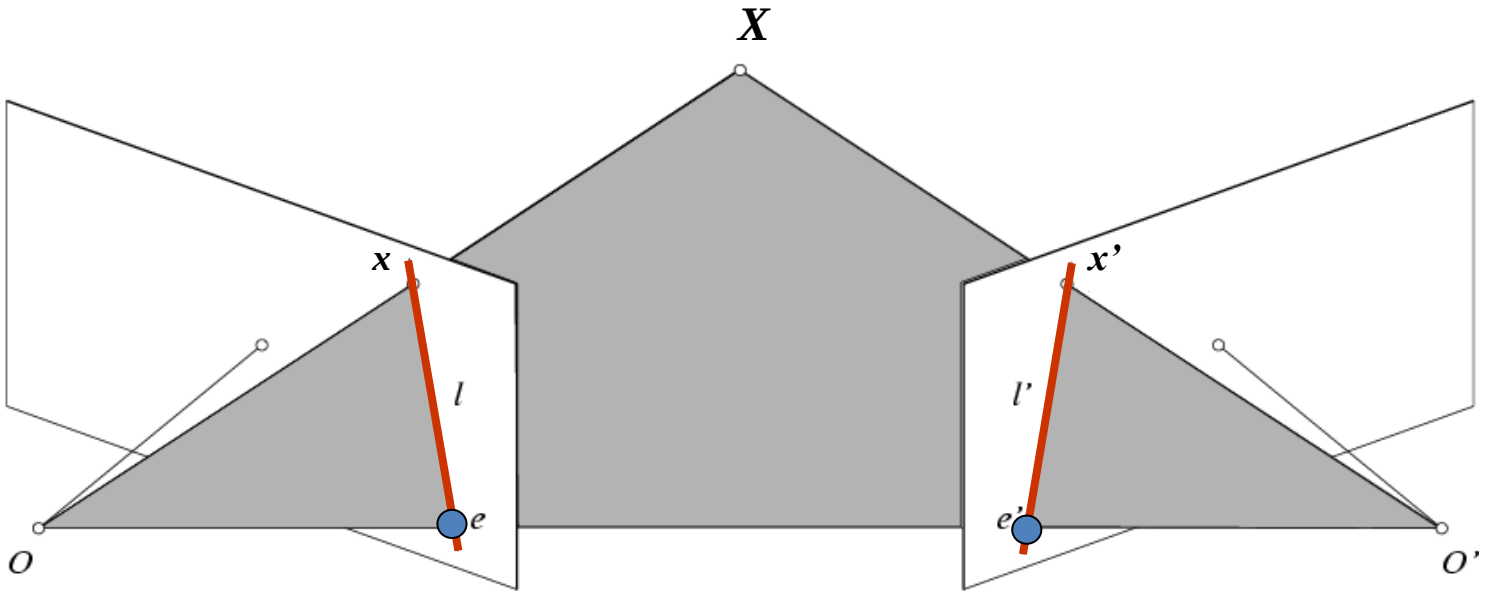


$$x'^T E x = 0$$

- Ex **is the epipolar line associated with** x **(**$l' = Ex$ **)**
  - Recall: a line is given by $ax + by + c = 0$ or

$$\mathbf{l}^T\mathbf{x} = 0 \quad \text{where} \quad \mathbf{l} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

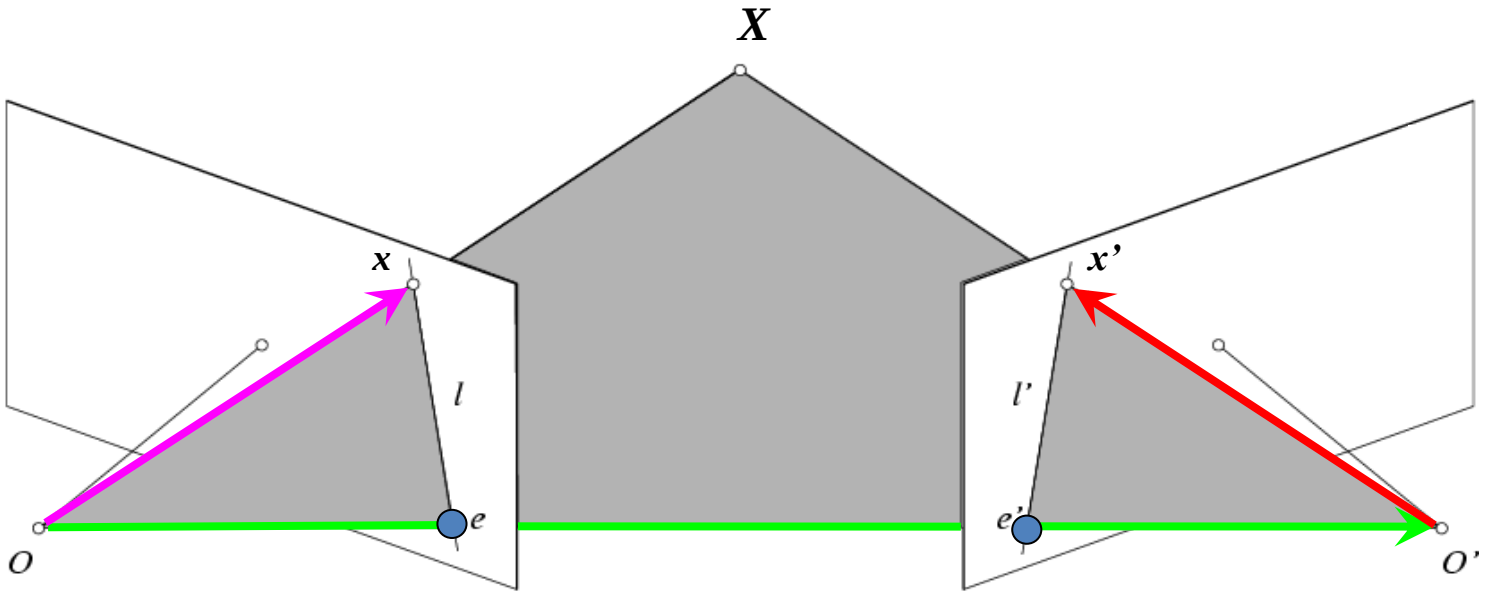
Epipolar constraint: Calibrated case



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

- $\boldsymbol{Ex}$ is the epipolar line associated with $\boldsymbol{x}$ ($l' = \boldsymbol{Ex}$)
- $\boldsymbol{E}^T\boldsymbol{x}'$ is the epipolar line associated with $\boldsymbol{x}'$ ($l = \boldsymbol{E}^T\boldsymbol{x}'$)
- $\boldsymbol{Ee} = 0$ and $E^T\boldsymbol{e}' = 0$
- $\boldsymbol{E}$ is singular (rank two)
- $\boldsymbol{E}$ has five degrees of freedom

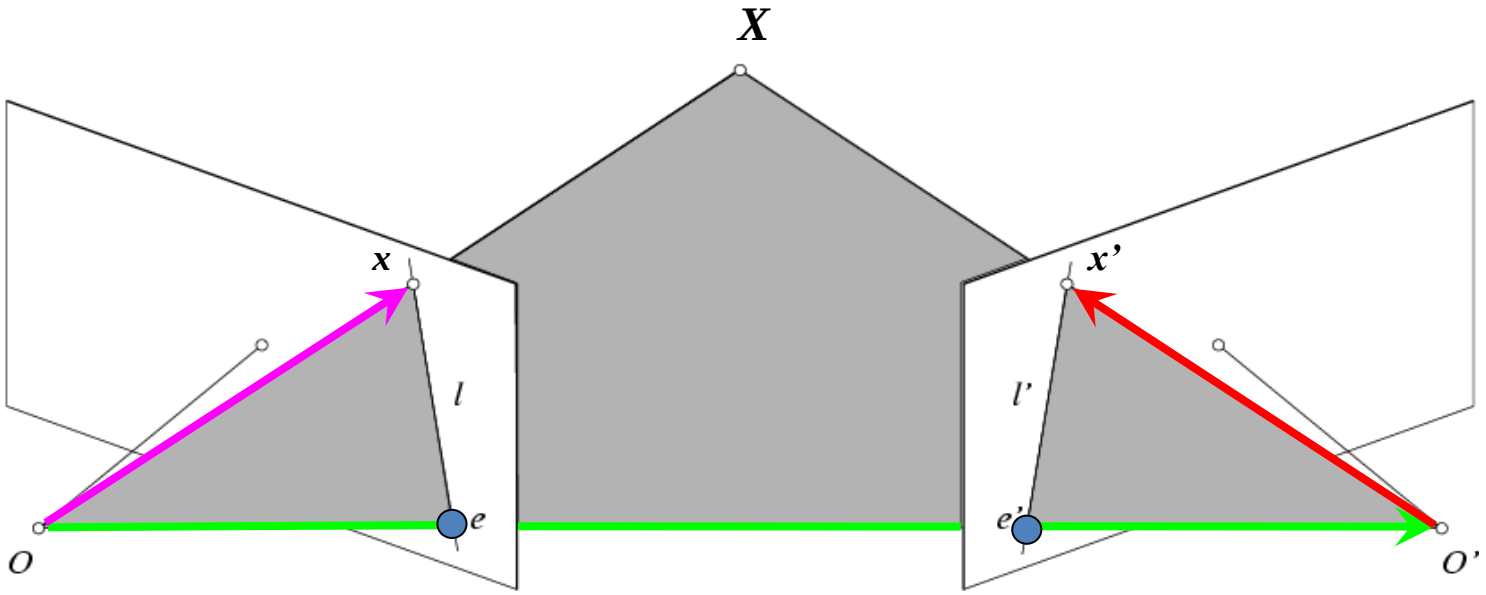
Epipolar constraint: Uncalibrated case



- The calibration matrices ***K*** and ***K′*** of the two cam eras are unknown
- We can write the epipolar constraint in terms of *unknown* normalized coordinates:

$$\hat{x}'^T E \hat{x} = 0 \qquad \hat{x} = K^{-1}x, \quad \hat{x}' = K'^{-1}x'$$

Epipolar constraint: Uncalibrated case



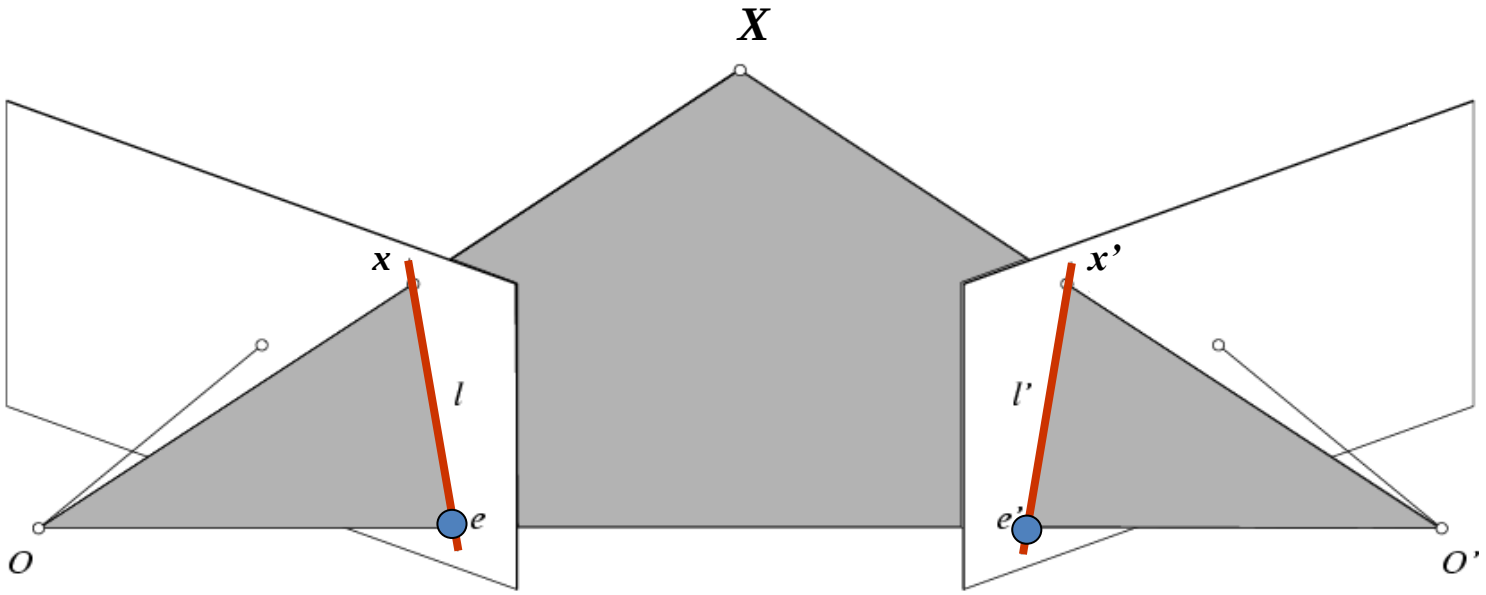
$$\hat{x}'^T E \hat{x} = 0 \quad \Rightarrow \quad x'^T F x = 0 \quad \text{with} \quad F = K'^{-T} E K^{-1}$$

$$\hat{x} = K^{-1} x$$

$$\hat{x}' = K'^{-1} x'$$

Fundamental Matrix
(Faugeras and Luong, 1992)

Epipolar constraint: Uncalibrated case



$$\hat{\boldsymbol{x}}'^{\boldsymbol{T}}\boldsymbol{E}\,\hat{\boldsymbol{x}}=0 \quad \Rightarrow \quad \boldsymbol{x}'^{\boldsymbol{T}}\boldsymbol{F}\,\boldsymbol{x}=0 \quad \text{with} \quad \boldsymbol{F}=\boldsymbol{K}'^{-\boldsymbol{T}}\boldsymbol{E}\,\boldsymbol{K}^{-1}$$

- $\boldsymbol{Fx}$ is the epipolar line associated with $\boldsymbol{x}(\boldsymbol{l}'=\boldsymbol{Fx})$
- $\boldsymbol{F}^T\boldsymbol{x}'$ is the epipolar line associated with $\boldsymbol{x}'(\boldsymbol{l}=\boldsymbol{F}^T\boldsymbol{x}')$
- $\boldsymbol{Fe}=0$ and $\boldsymbol{F}^T\boldsymbol{e}'=0$
- $\boldsymbol{F}$ is singular (rank two)
- $\boldsymbol{F}$ has *seven* degrees of freedom

Estimating the fundamental matrix

The eight-point algorithm

$$\boldsymbol{x} = (u, v, 1)^T, \quad \boldsymbol{x}' = (u', v', 1)$$

$$\begin{bmatrix} u' & v' & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 0$$

$$\begin{bmatrix} u'u & u'v & u' & v'u & v'v & v' & u & v & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

Solve homogeneous linear system using eight or more matches

Enforce rank-2 constraint (take SVD of ***F*** and throw out the smallest singular value)

Problem with eight-point algorithm

$$\begin{bmatrix} u'u & u'v & u' & v'u & v'v & v' & u & v \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \end{bmatrix} = -1$$

Problem with eight-point algorithm

250906.36	183269.57	921.81	200931.10	146766.13	738.21	272.19	198.81
2692.28	131633.03	176.27	6196.73	302975.59	405.71	15.27	746.79
416374.23	871684.30	935.47	408110.89	854384.92	916.90	445.10	931.81
191183.60	171759.40	410.27	416435.62	374125.90	893.65	465.99	418.65
48988.86	30401.76	57.89	298604.57	185309.58	352.87	846.22	525.15
164786.04	546559.67	813.17	1998.37	6628.15	9.86	202.65	672.14
116407.01	2727.75	138.89	169941.27	3982.21	202.77	838.12	19.64
135384.58	75411.13	198.72	411350.03	229127.78	603.79	681.28	379.48

$$\begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \end{bmatrix} = -1$$

- Poor numerical conditioning
- Can be fixed by rescaling the data

EBU7240

Computer Vision

- Stereo-

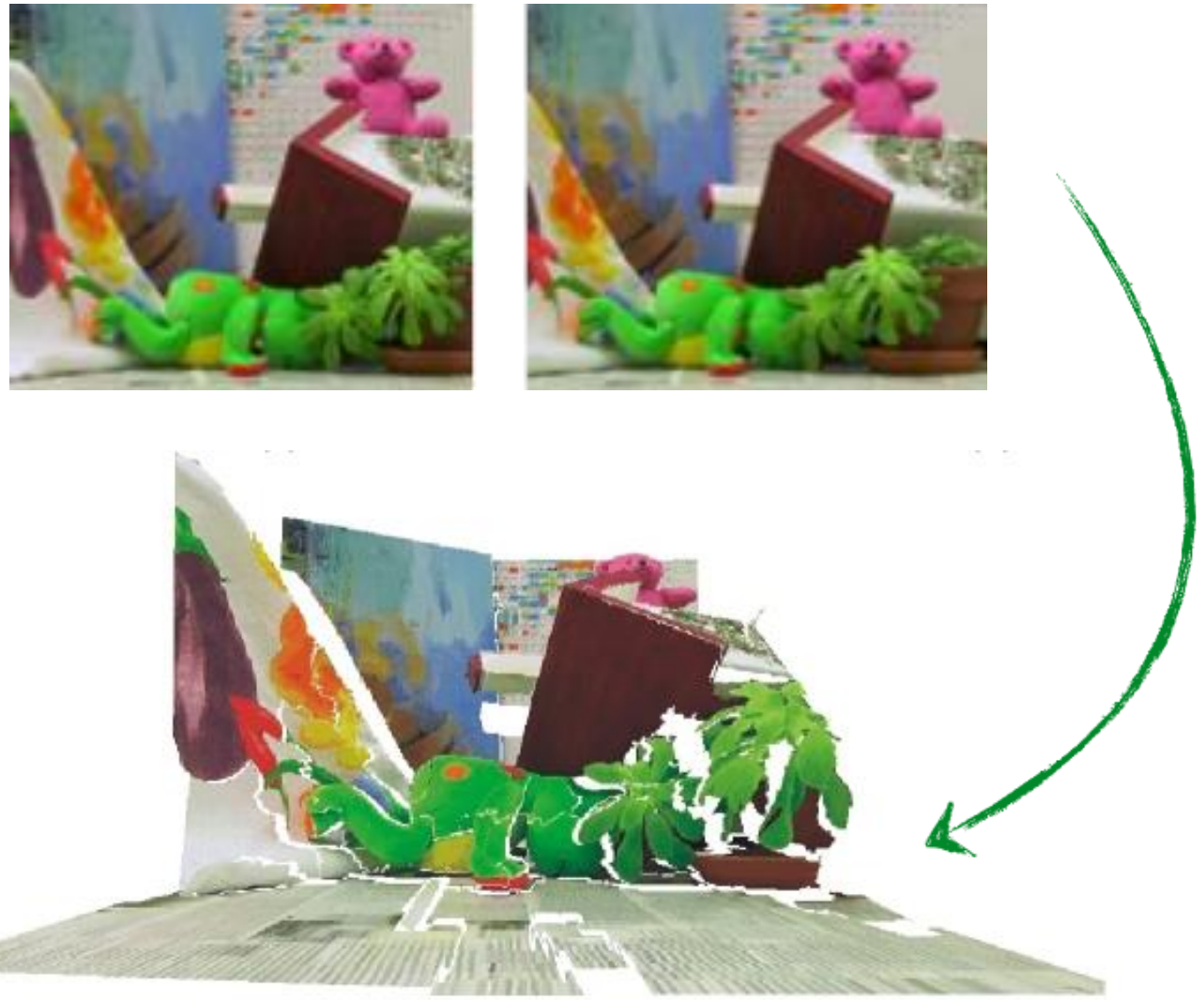
Semester 1, 2021

Changjae Oh

Objectives

- To understand the computational approach to depth estimation from stereo images
- To understand active method for depth estimation

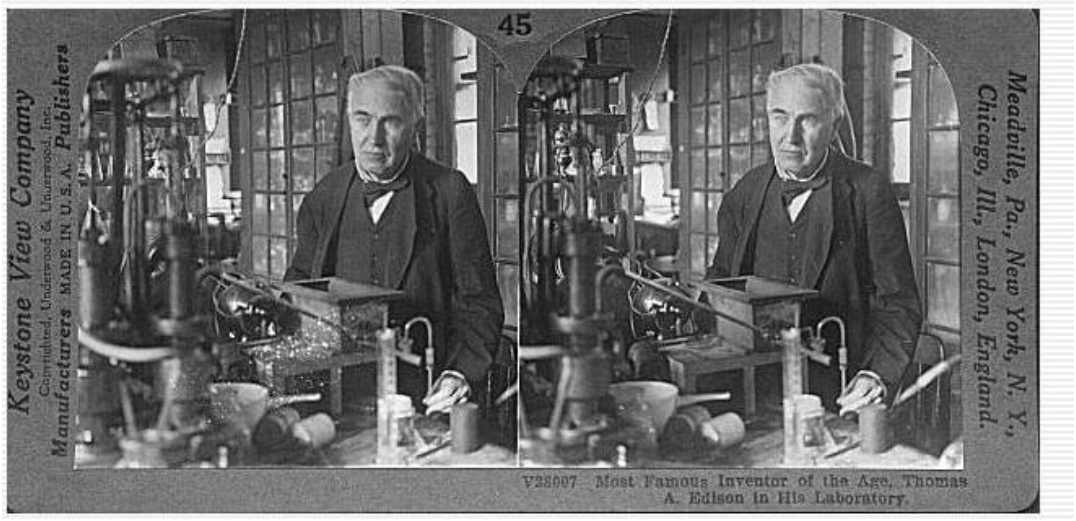
Two-View Stereo



Many slides adapted from Steve Seitz

Stereograms

- Humans can fuse pairs of images to get a sensation of depth



Stereograms: Invented by Sir Charles Wheatstone, 1838

Stereograms

Stereograms

- Humans can fuse pairs of images to get a sensation of depth

Autostereograms: www.magiceye.com

Stereograms

- Humans can fuse pairs of images to get a sensation of depth

Autostereograms: www.magiceye.com

Problem formulation

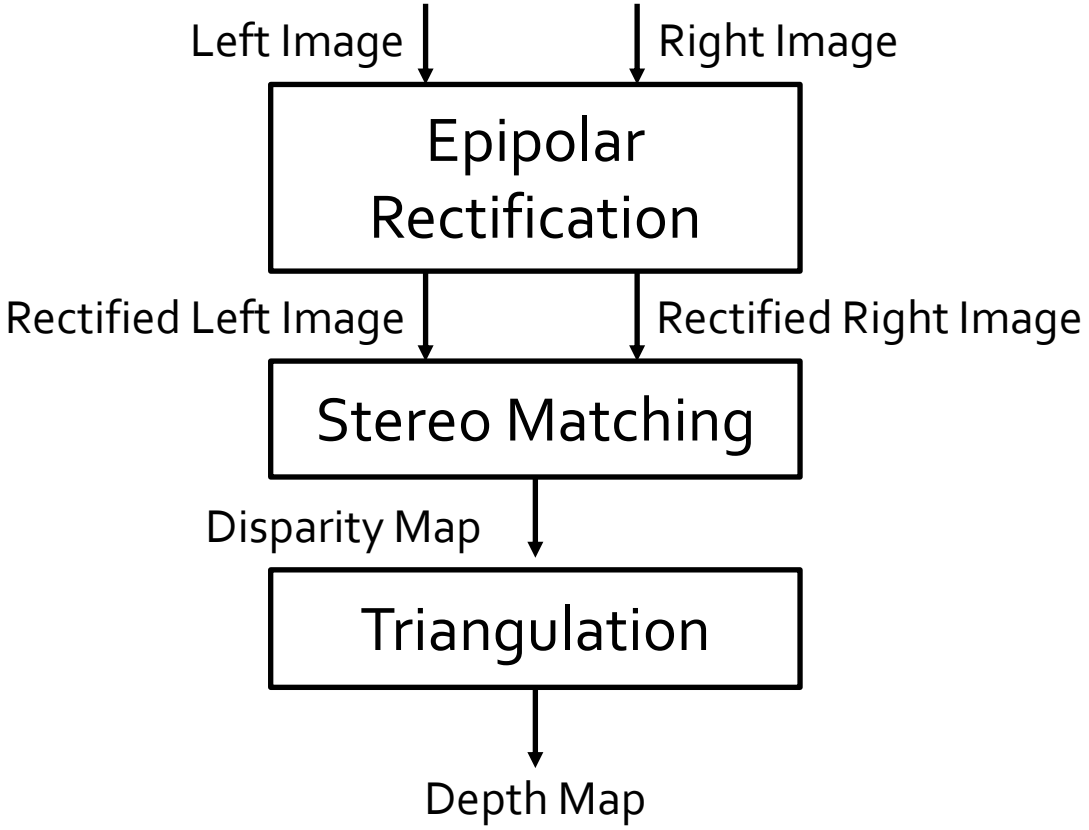
- Given a calibrated binocular stereo pair, fuse it to produce a depth image

image 1

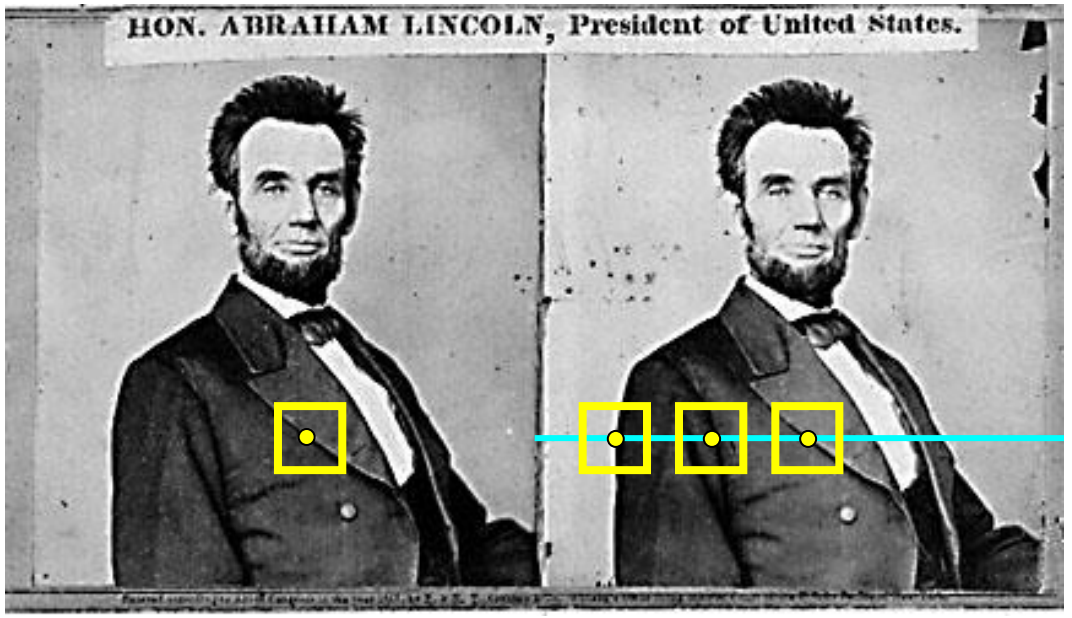
image 2

Dense depth map

Computational Stereo Pipeline



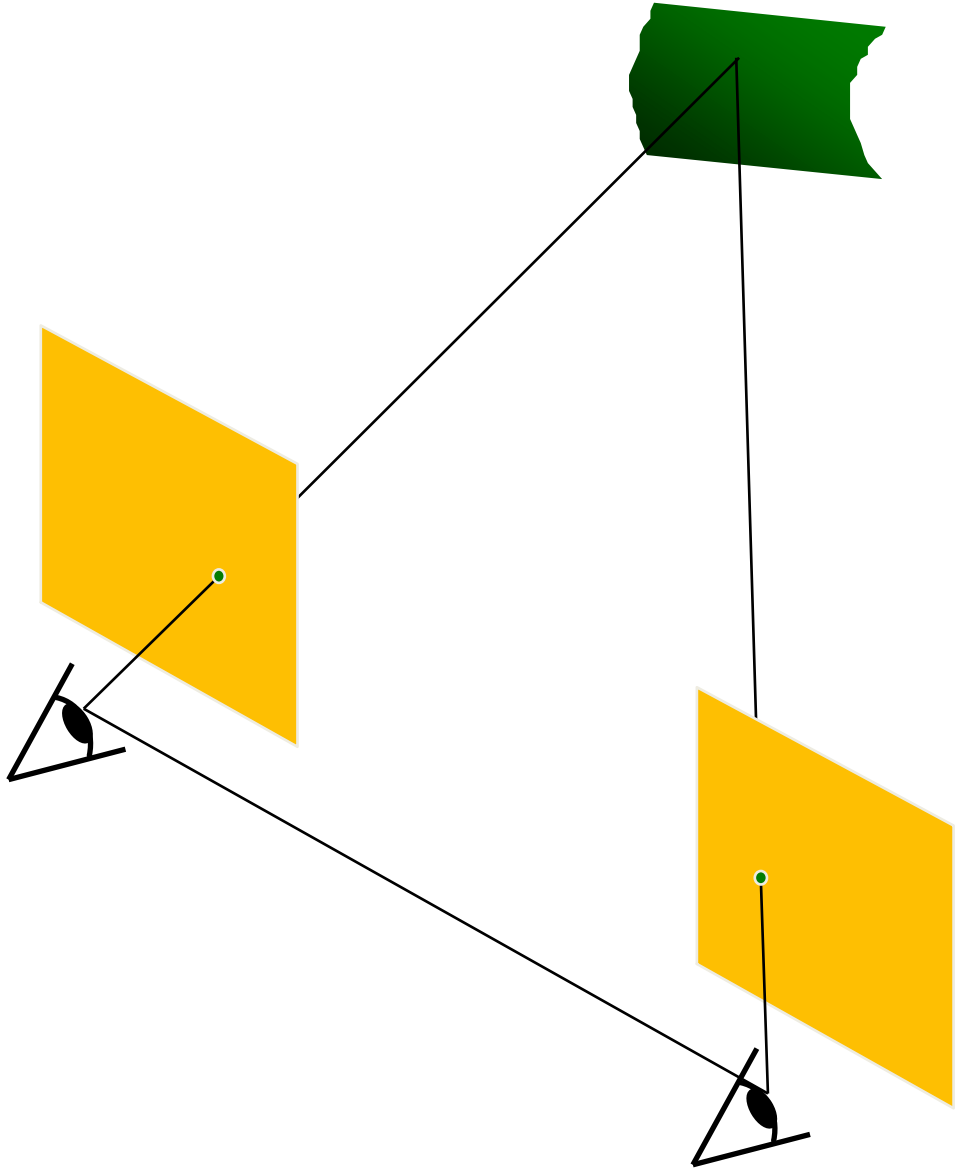
Epipolar rectification – Why needed?



- **For each pixel in the first image**
  - Find corresponding epipolar line in the right image
  - Examine all pixels on the epipolar line and pick the best match
  - Triangulate the matches to get depth information

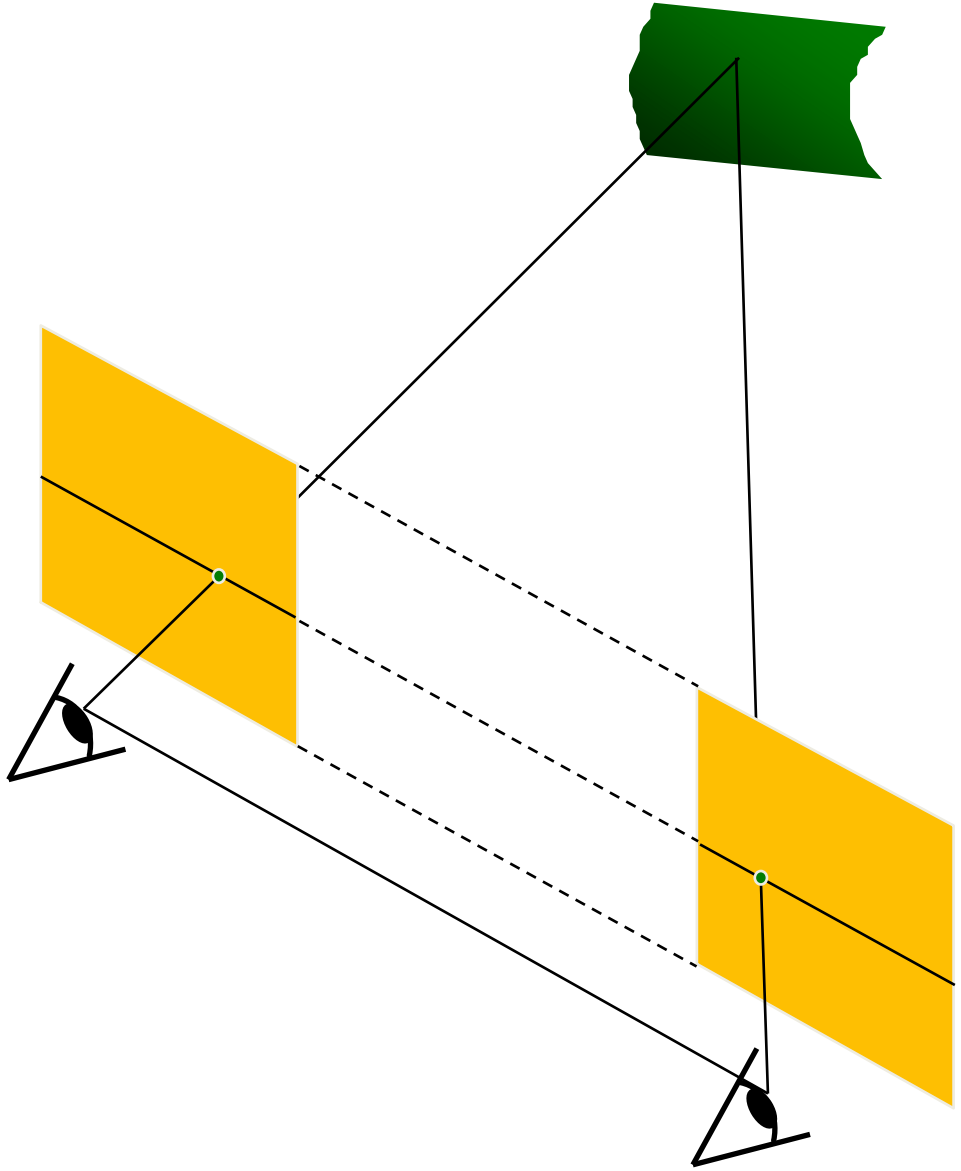
- **Simplest case: epipolar lines are corresponding scanlines**
  - When does this happen?

Simplest Case: Parallel images



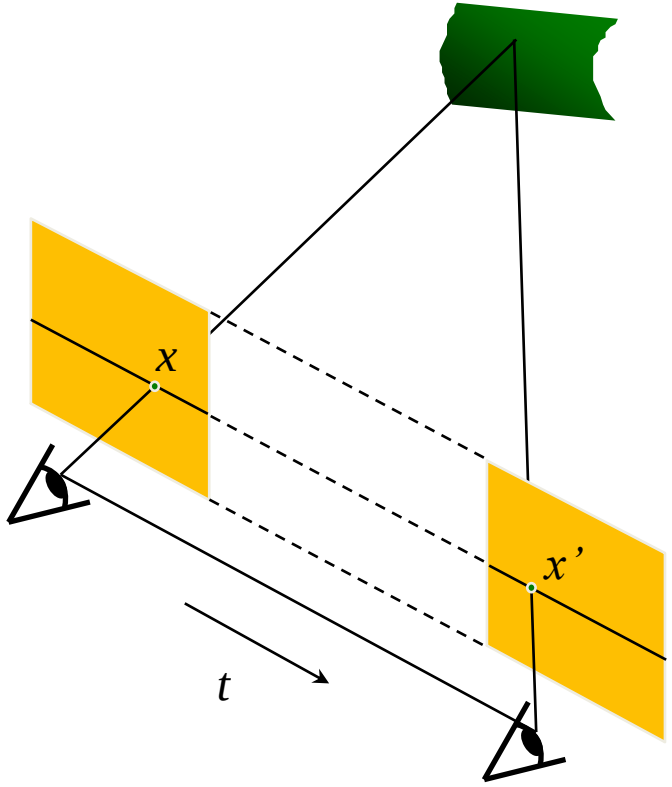
- Image planes of cameras are parallel to each other and to the baseline
- Camera centers are at same height
- Focal lengths are the same

Simplest Case: Parallel images



- Image planes of cameras are parallel to each other and to the baseline
- Camera centers are at same height
- Focal lengths are the same
- Then epipolar lines fall along the hori zontal scan lines of the images

Essential matrix for parallel images



Epipolar constraint:

$$\boldsymbol{x}'^{T}\boldsymbol{E}\,\boldsymbol{x}=0, \quad \boldsymbol{E}=[\boldsymbol{t}_{\times}]\boldsymbol{R}$$

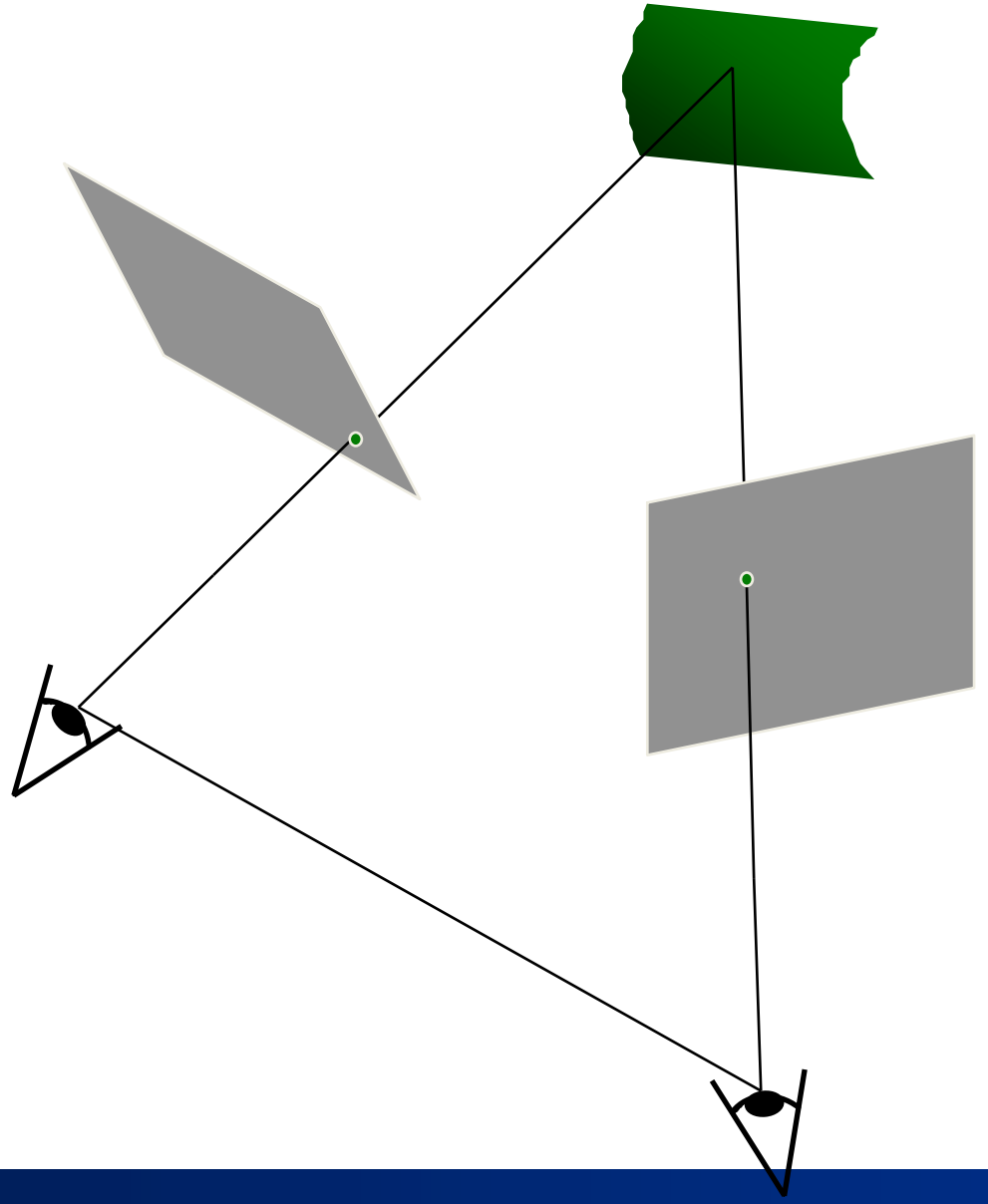
$$\boldsymbol{R}=\boldsymbol{I} \qquad \boldsymbol{t}=(T,0,0)$$

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

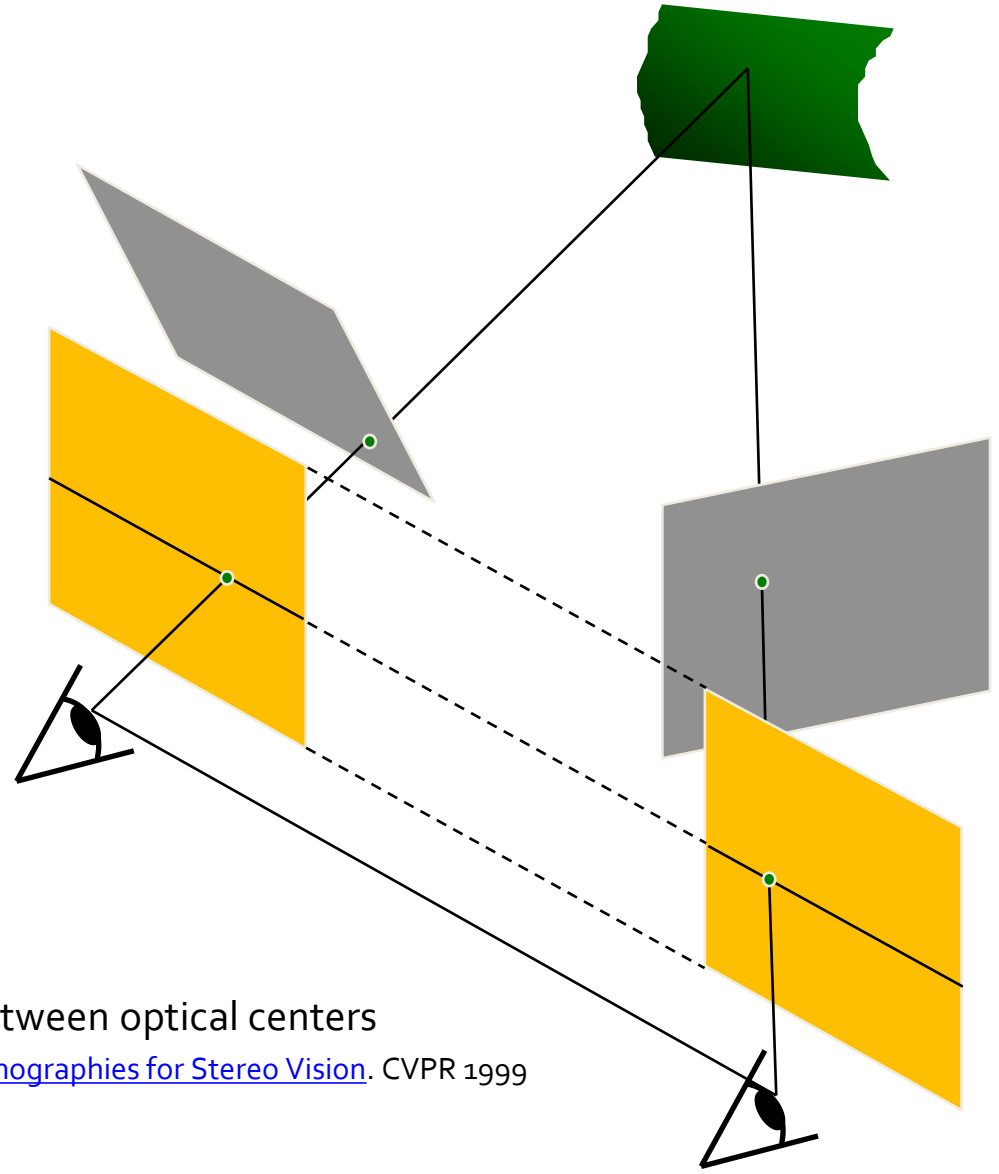
$$\begin{pmatrix}u' & v' & 1\end{pmatrix}\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & -T\\ 0 & T & 0\end{bmatrix}\begin{pmatrix}u\\ v\\ 1\end{pmatrix}=0 \qquad \begin{pmatrix}u' & v' & 1\end{pmatrix}\begin{pmatrix}0\\ -T\\ Tv\end{pmatrix}=0 \qquad Tv'=Tv$$

The y-coordinates of corresponding points are the same!

Stereo image rectification



Stereo image rectification



- **Reproject image planes**
  - onto a common plane parallel to the line between optical centers

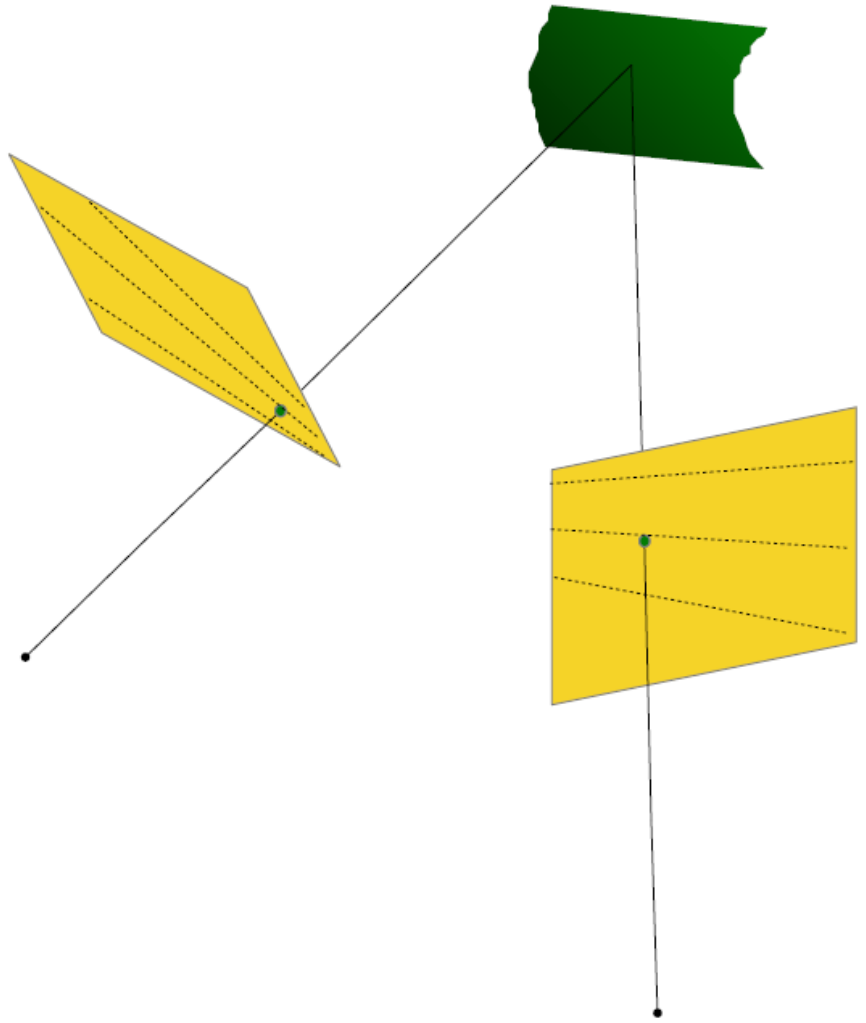
C. Loop and Z. Zhang. Computing Rectifying Homographies for Stereo Vision. CVPR 1999

Stereo image 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

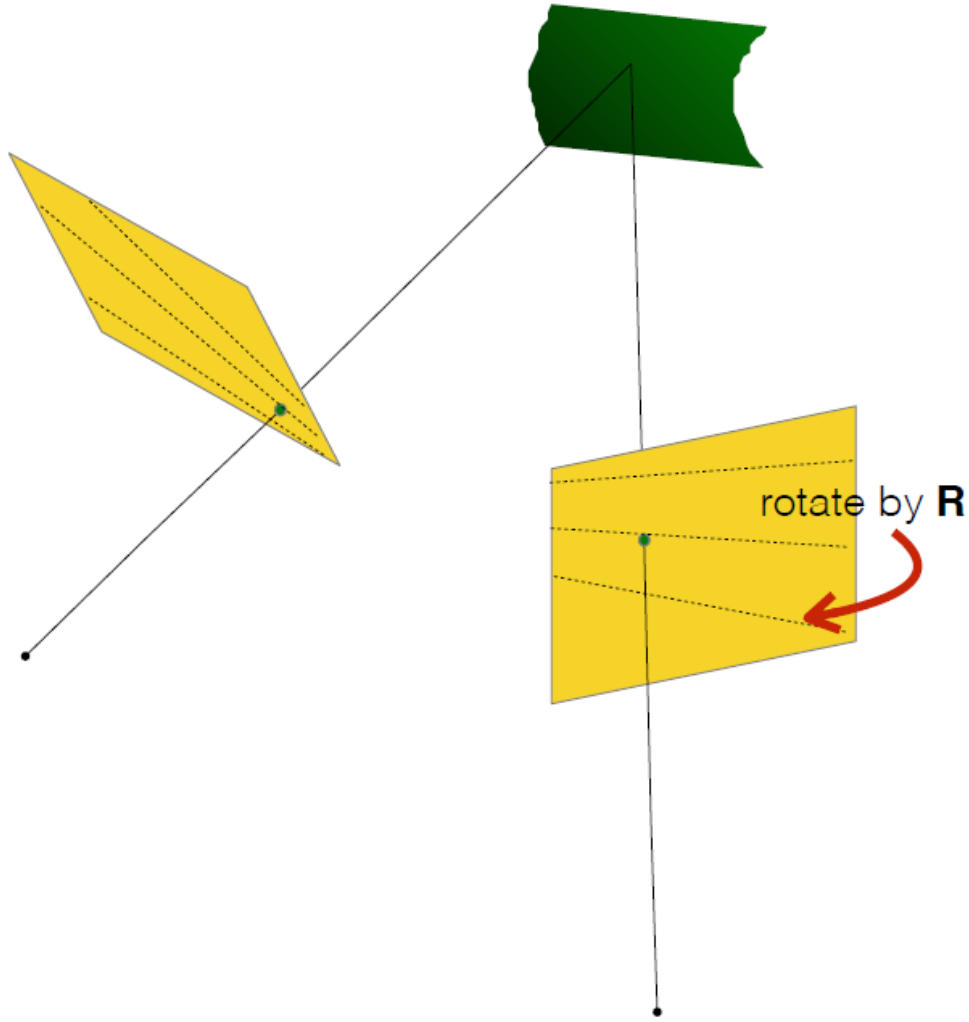
Stereo image rectification

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



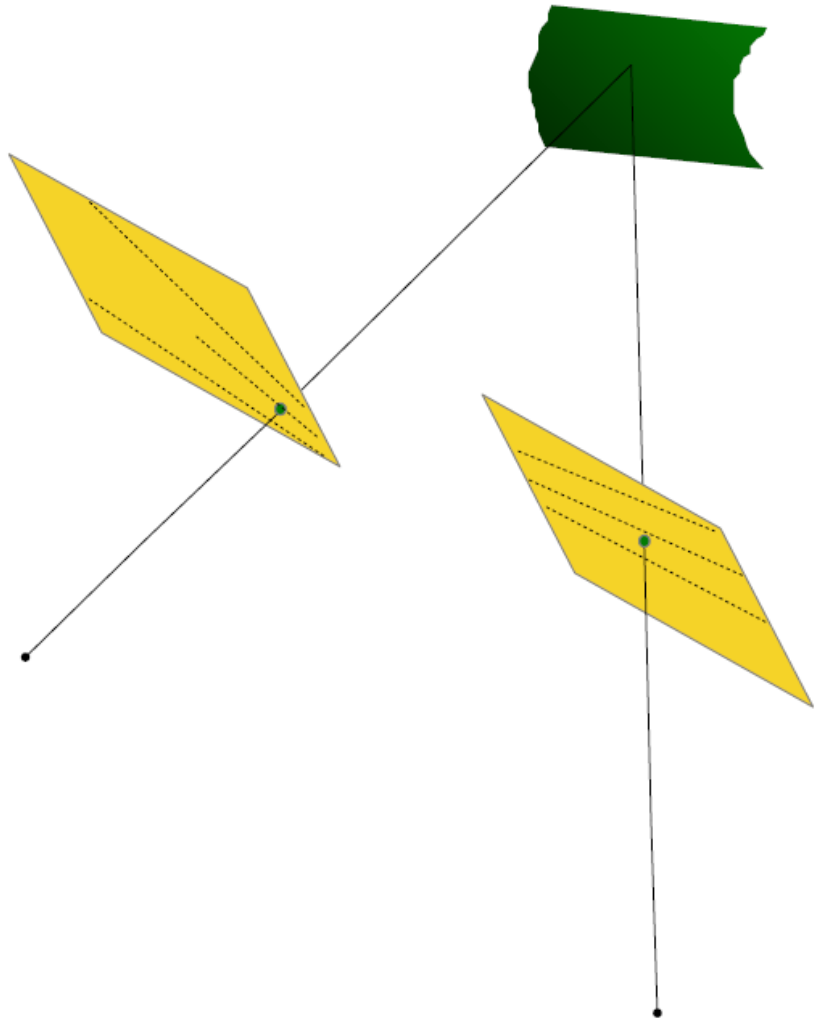
Stereo image rectification

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



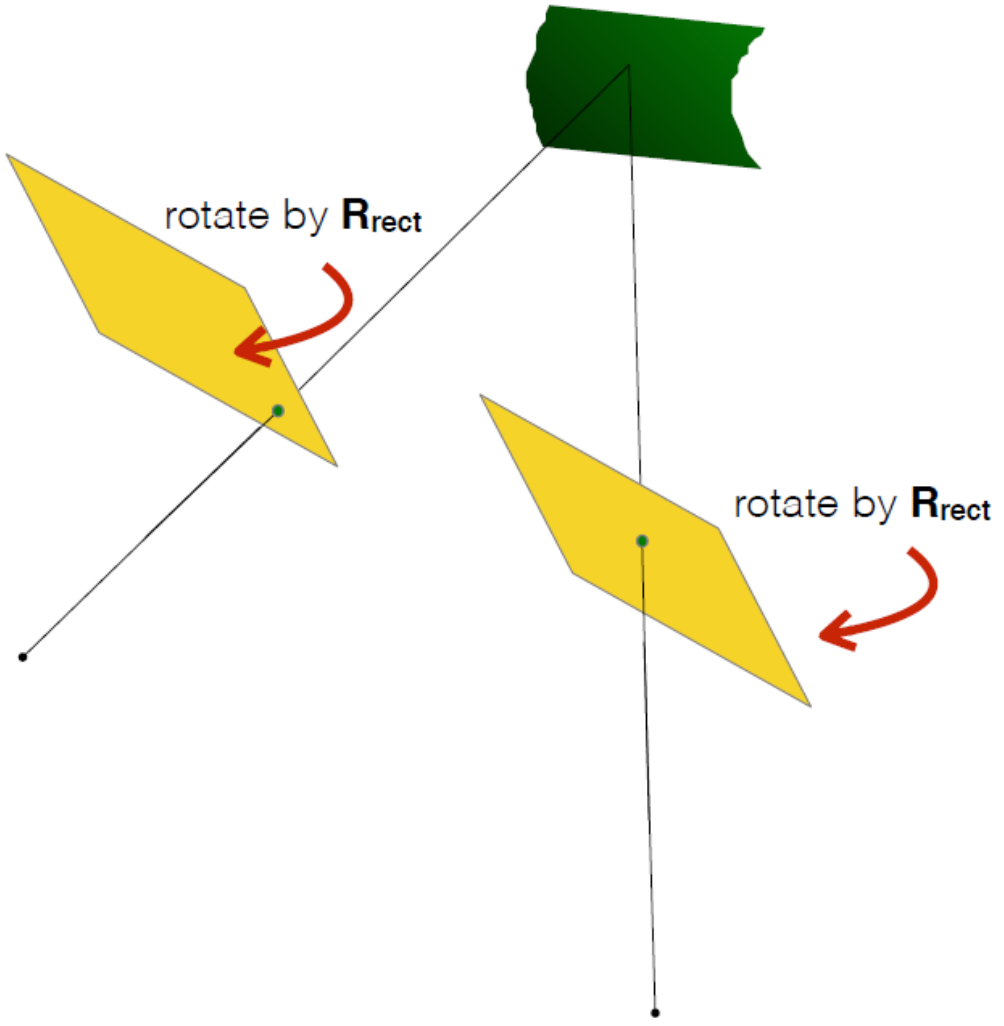
Stereo image rectification

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



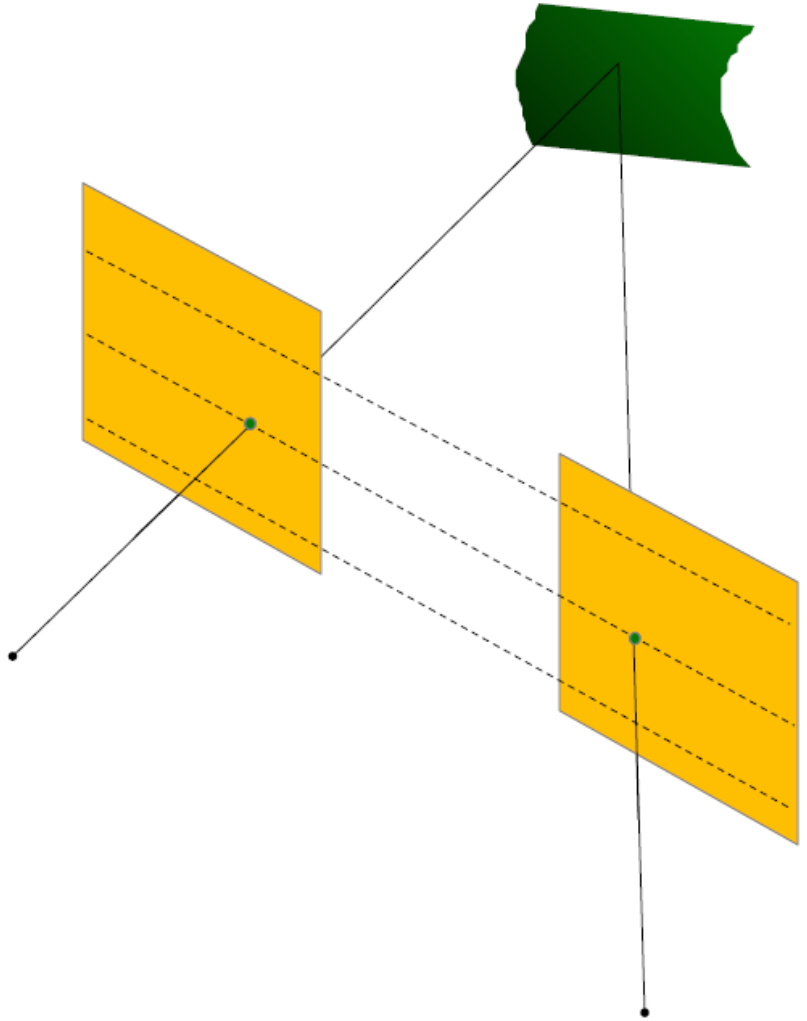
Stereo image rectification

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



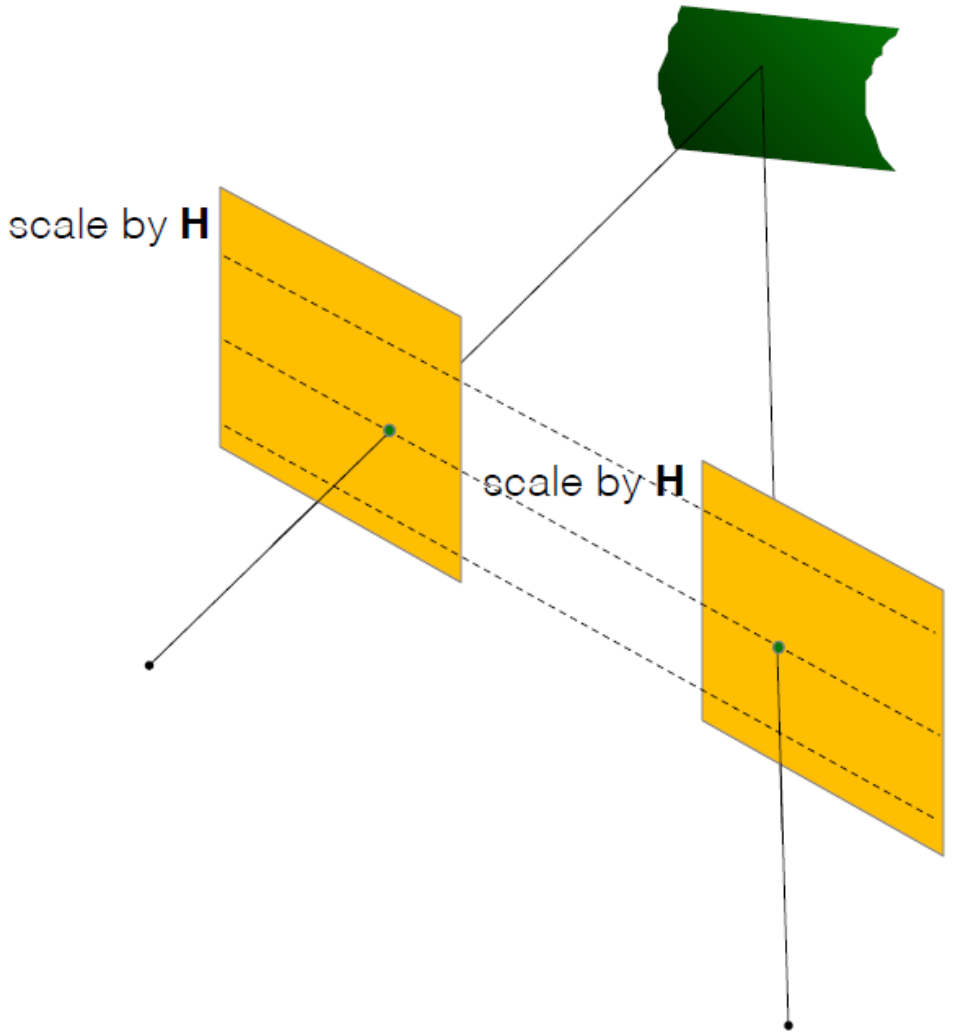
Stereo image rectification

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



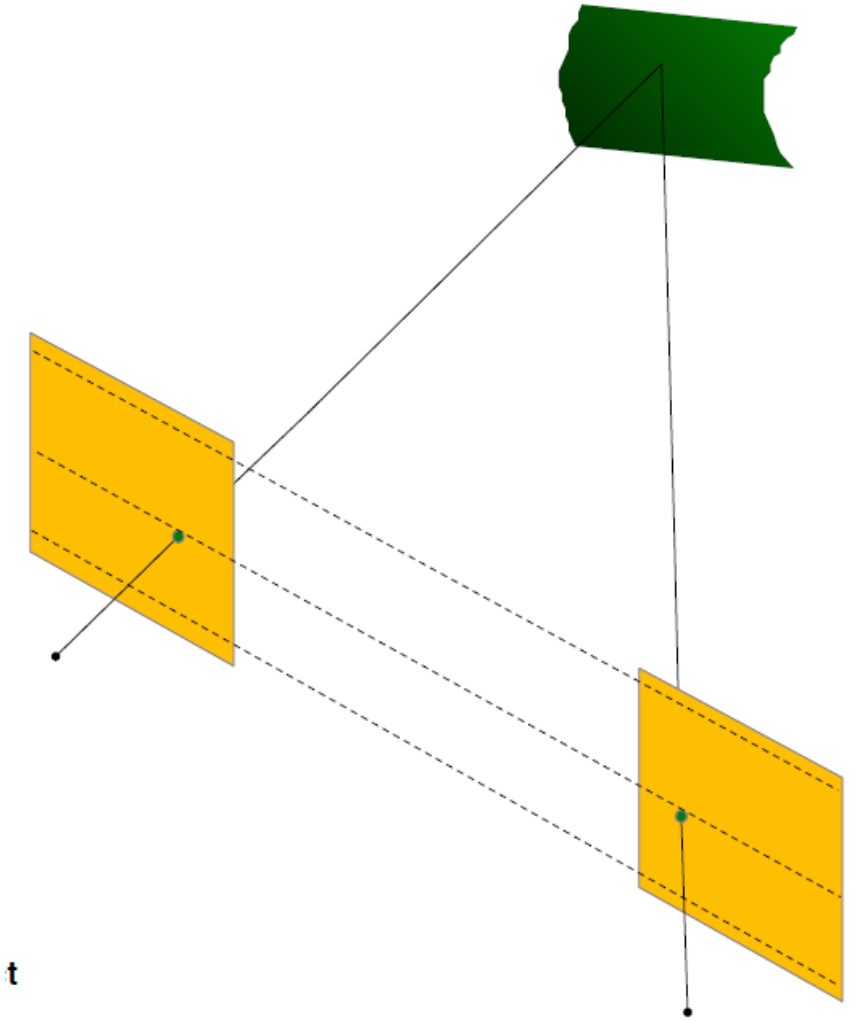
Stereo image rectification

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

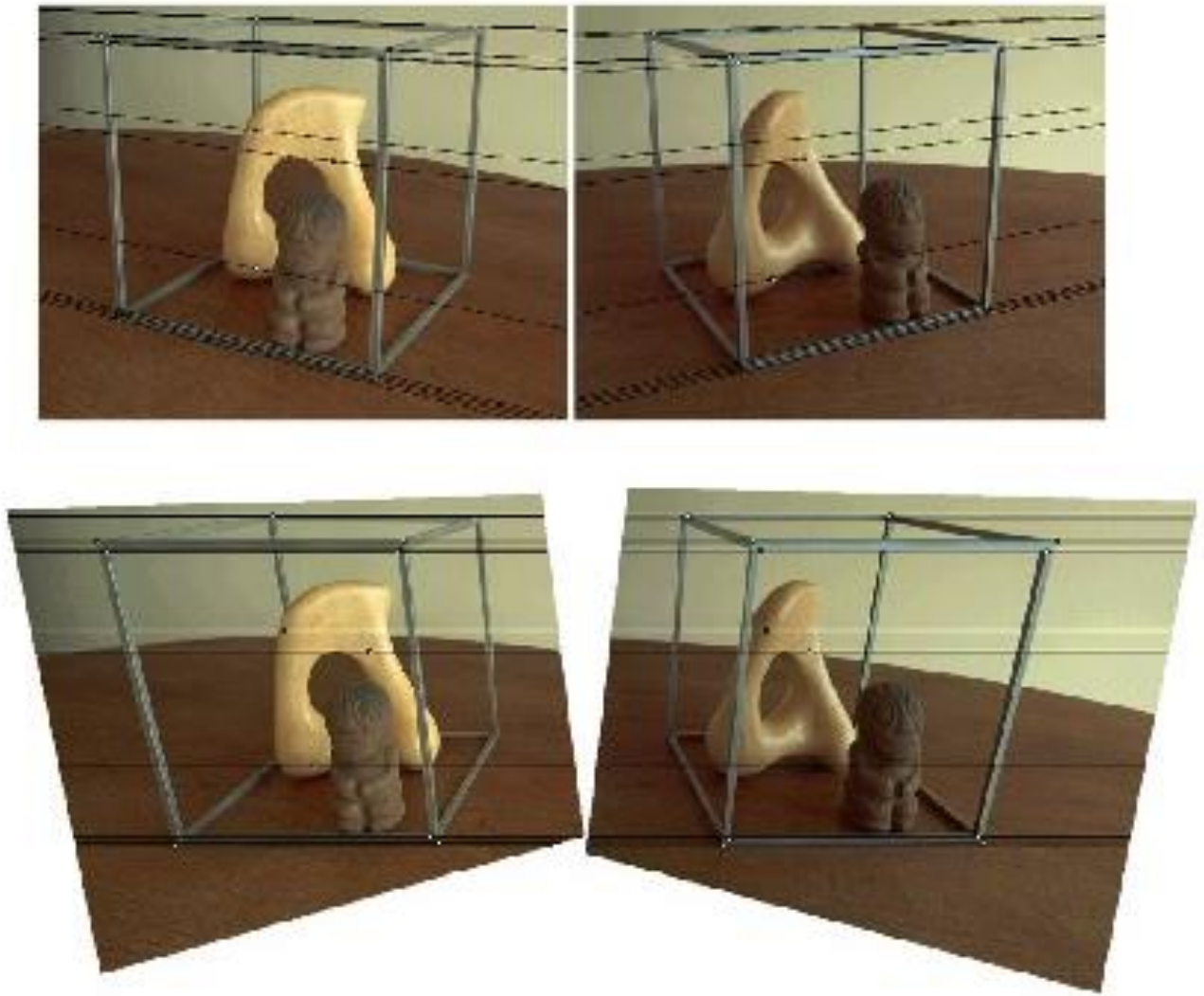


Stereo image rectification

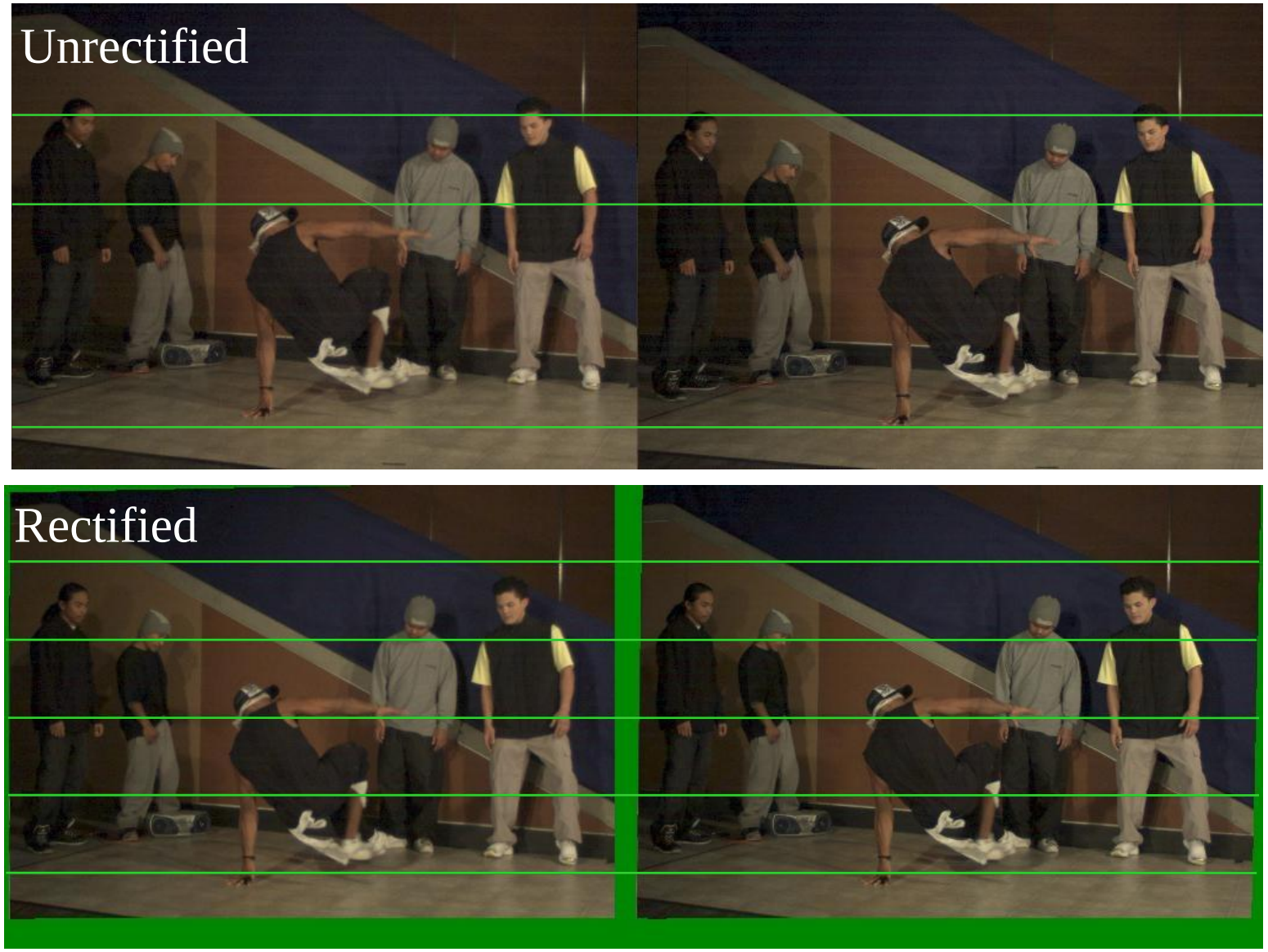
1. Compute E to get R
2. Rotate right image by R
3. Rotate both images by R_{rect}
4. Scale both images by H



Rectification example

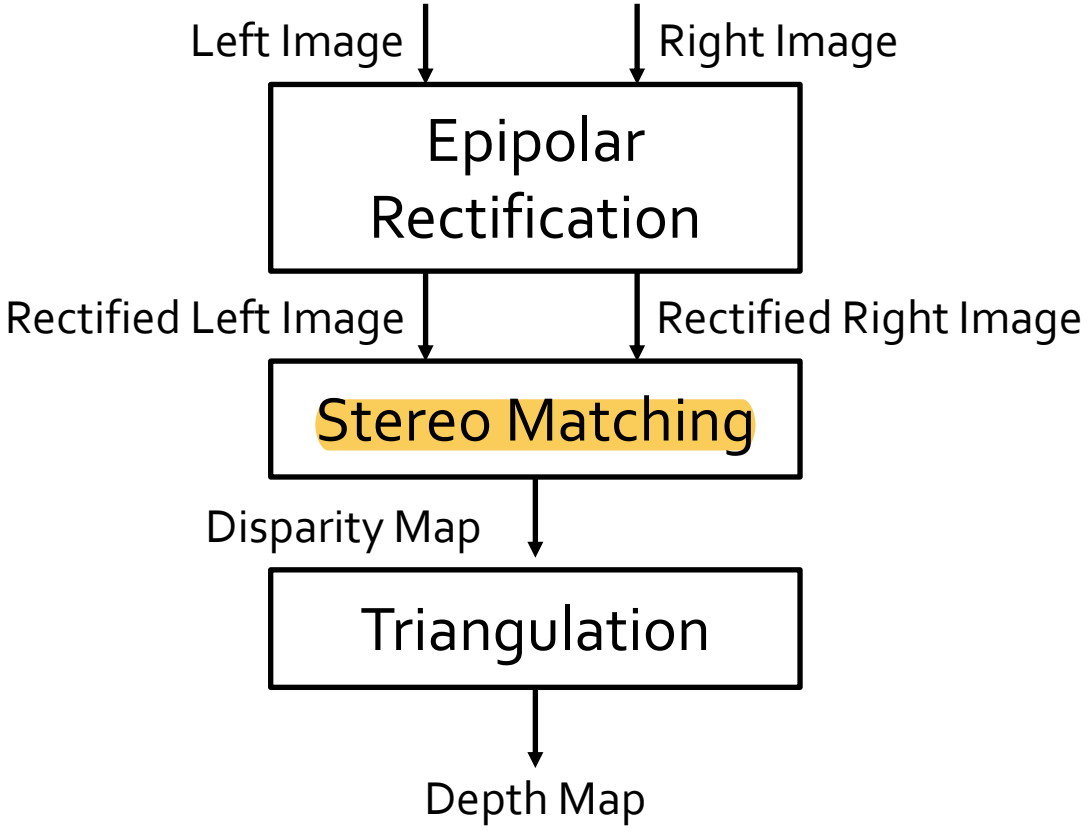


Another rectification example

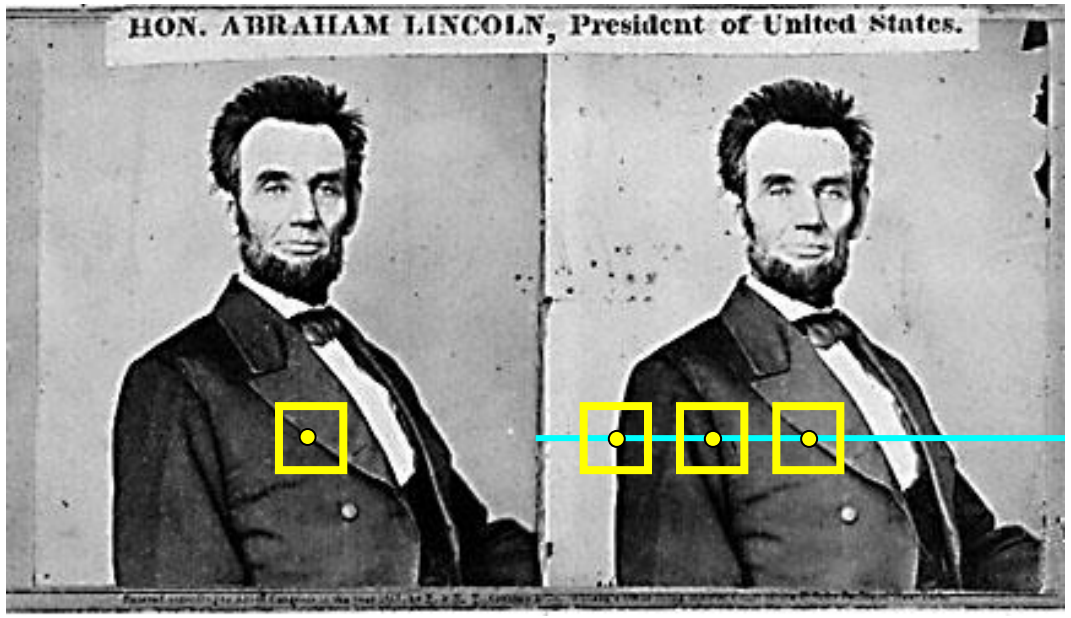


Stereo Matching

- **Finding the corresponding points in epipolar line**

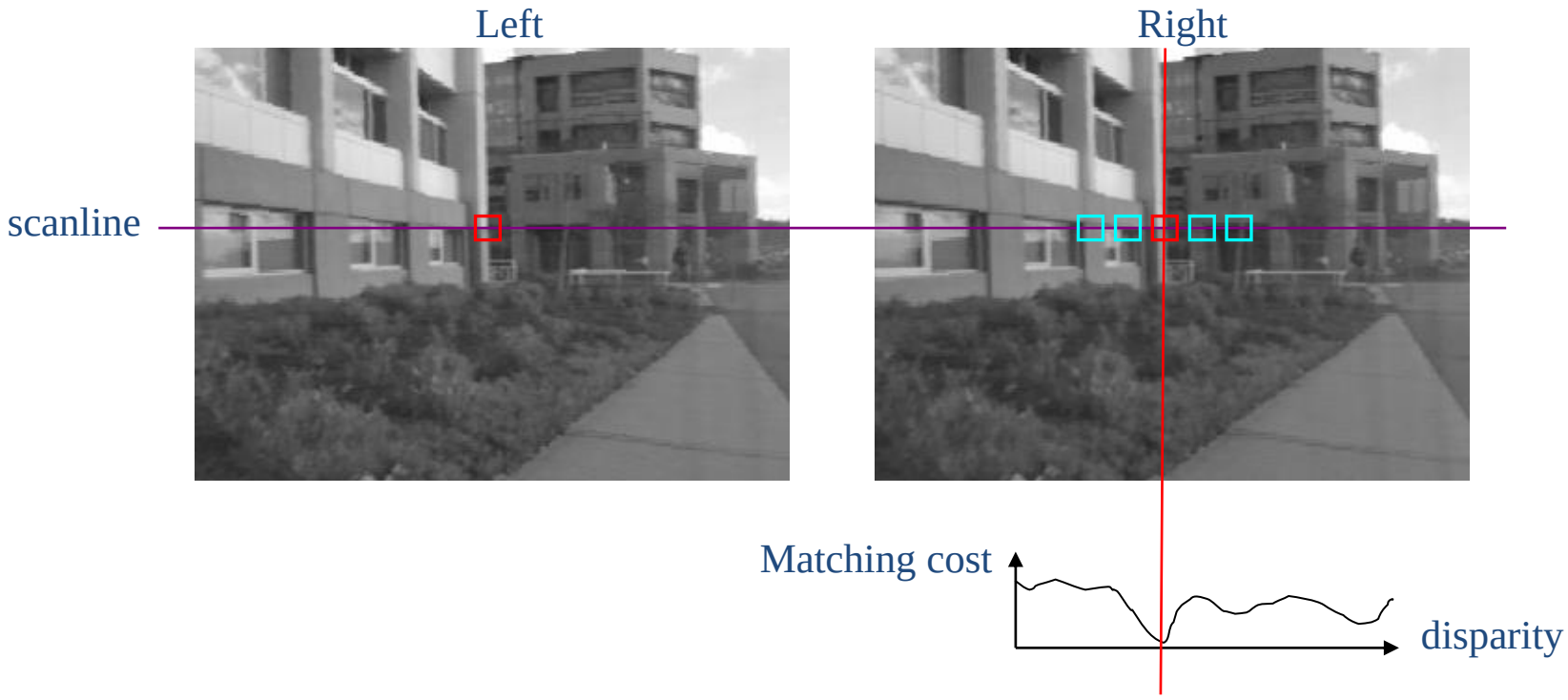


Basic stereo matching algorithm



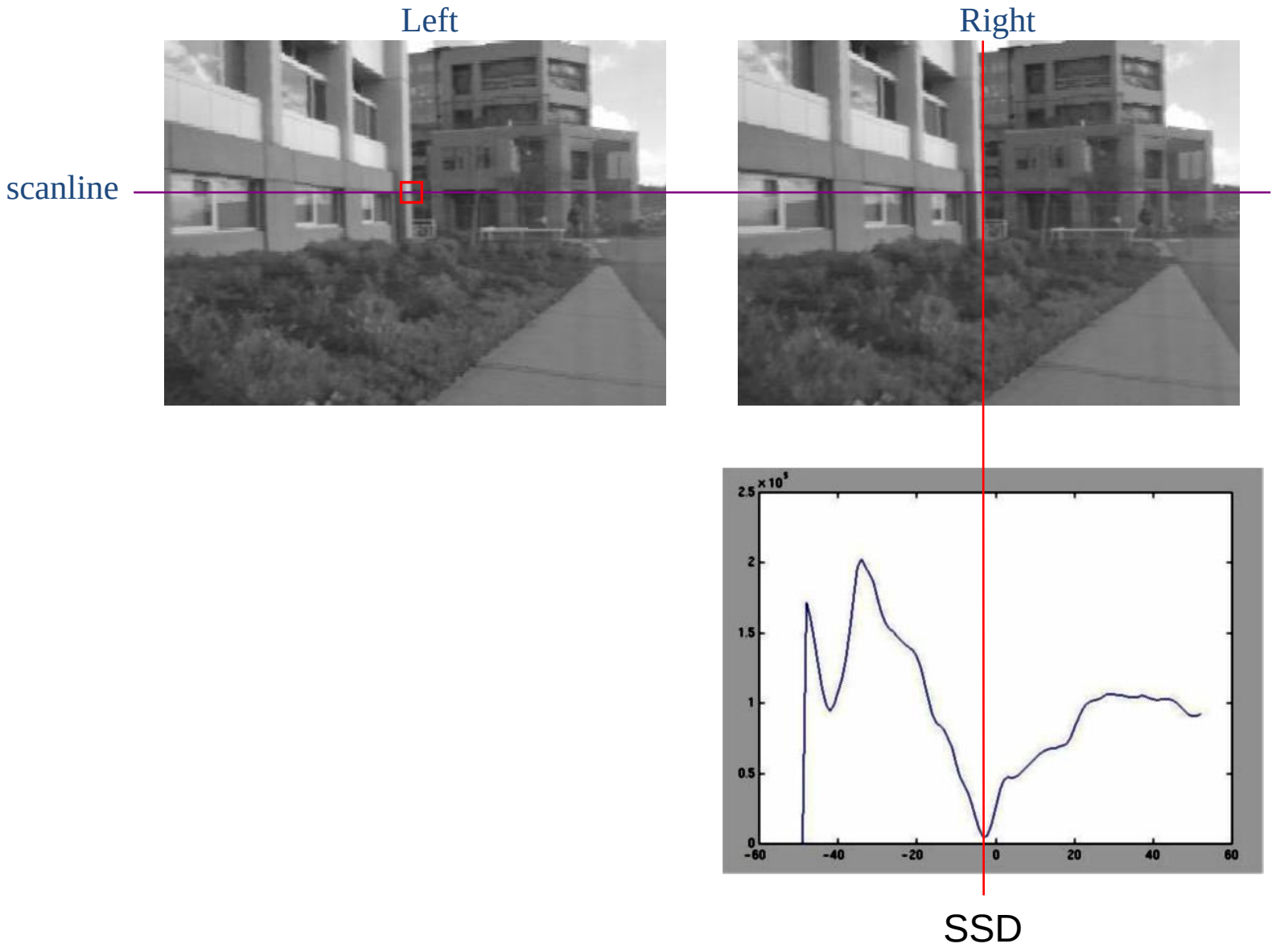
- **If necessary, rectify the two stereo images to transfo rm epipolar lines into scanlines**
- **For each pixel in the first image**
  - Find corresponding epipolar line in the right image
  - Examine all pixels on the epipolar line and pick the best match

Correspondence search

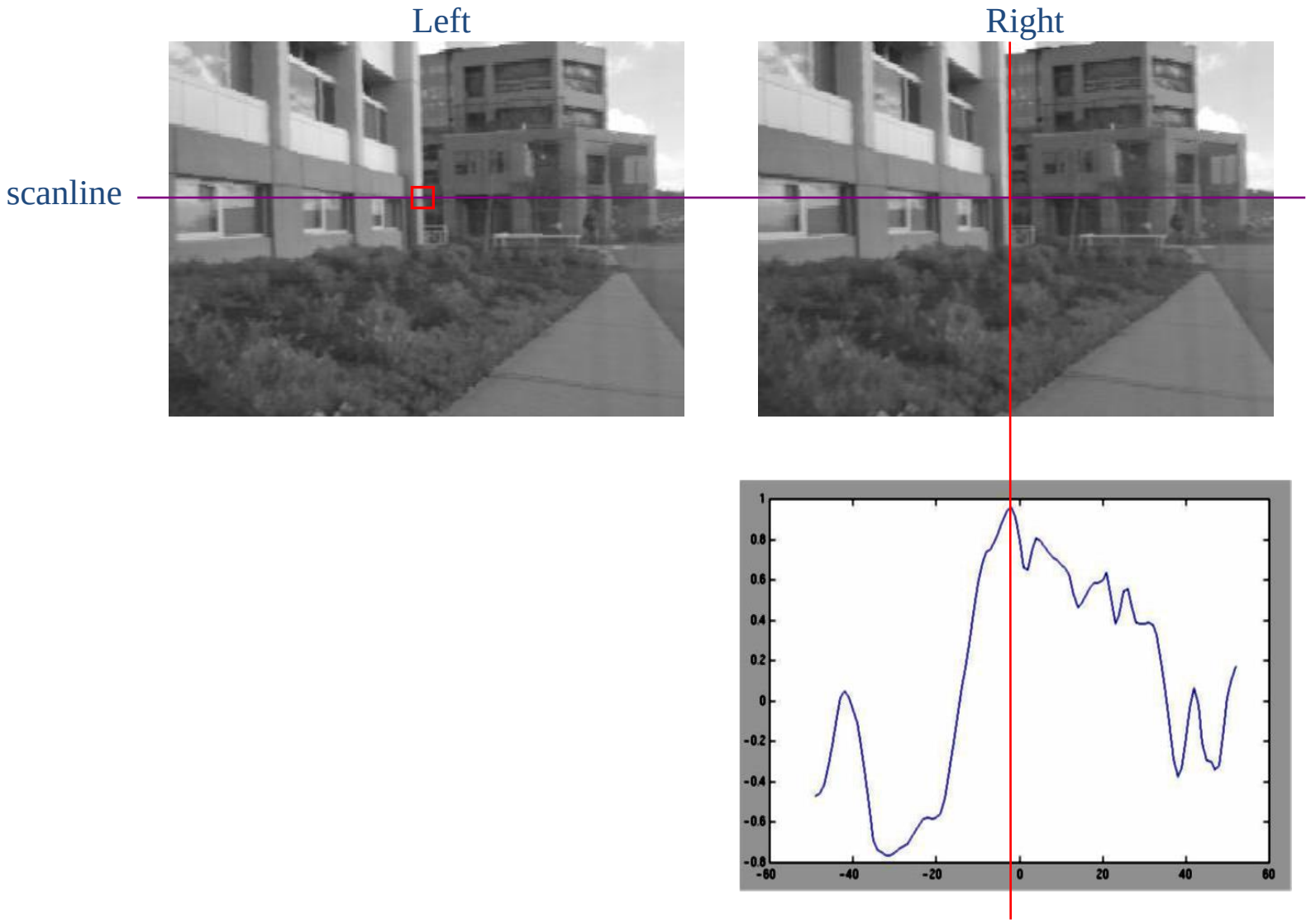


- Slide a window along the right scanline and com pare contents of that window with the reference window in the left image
- Matching cost: SSD or normalized correlation

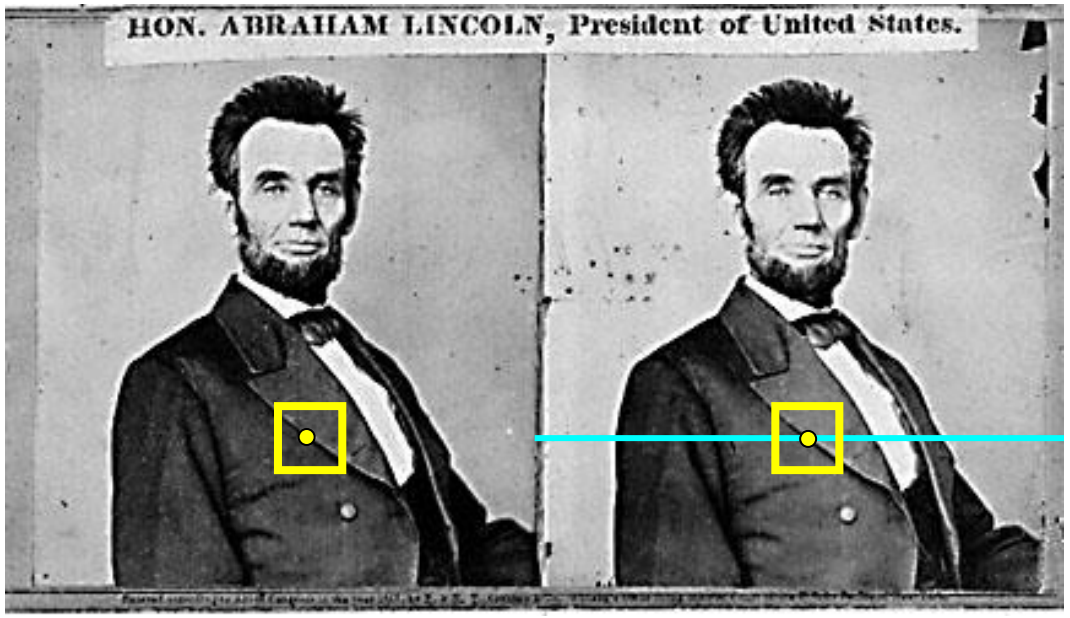
Correspondence search



Correspondence search



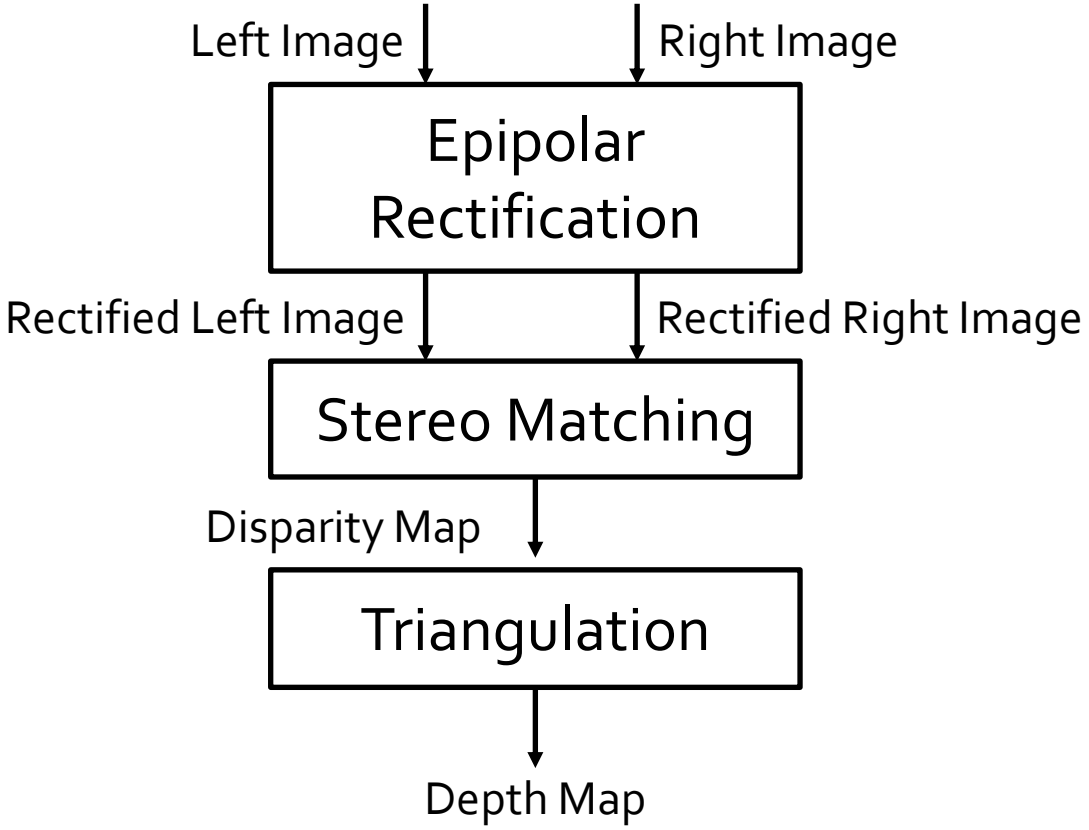
Basic stereo matching algorithm



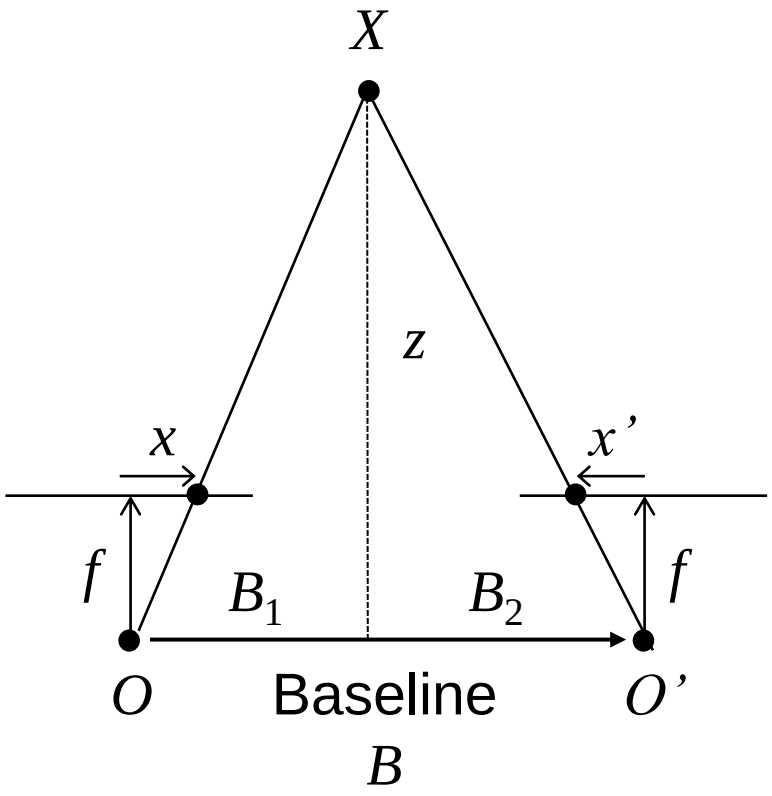
- **If necessary, rectify the two stereo images to transform epipo lar lines into scanlines**
- **For each pixel *x* in the first image**
  - Find corresponding epipolar scanline in the right image
  - Examine all pixels on the scanline and pick the best match *x′*
  - Triangulate the matches to get depth information

Depth from disparity

- **Performing triangulation**



Depth from disparity



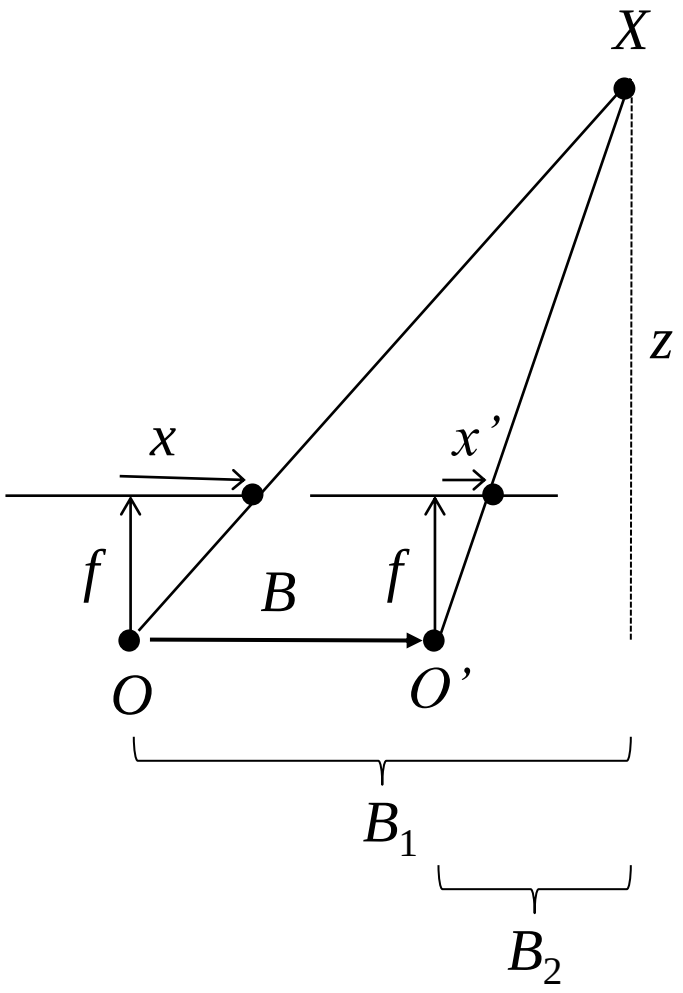
$$\frac{x}{f} = \frac{B_1}{z} \qquad \frac{-x'}{f} = \frac{B_2}{z}$$

$$\frac{x - x'}{f} = \frac{B_1 + B_2}{z}$$

$$disparity = x - x' = \frac{B \cdot f}{z}$$

Disparity is inversely proportional to depth!

Depth from disparity

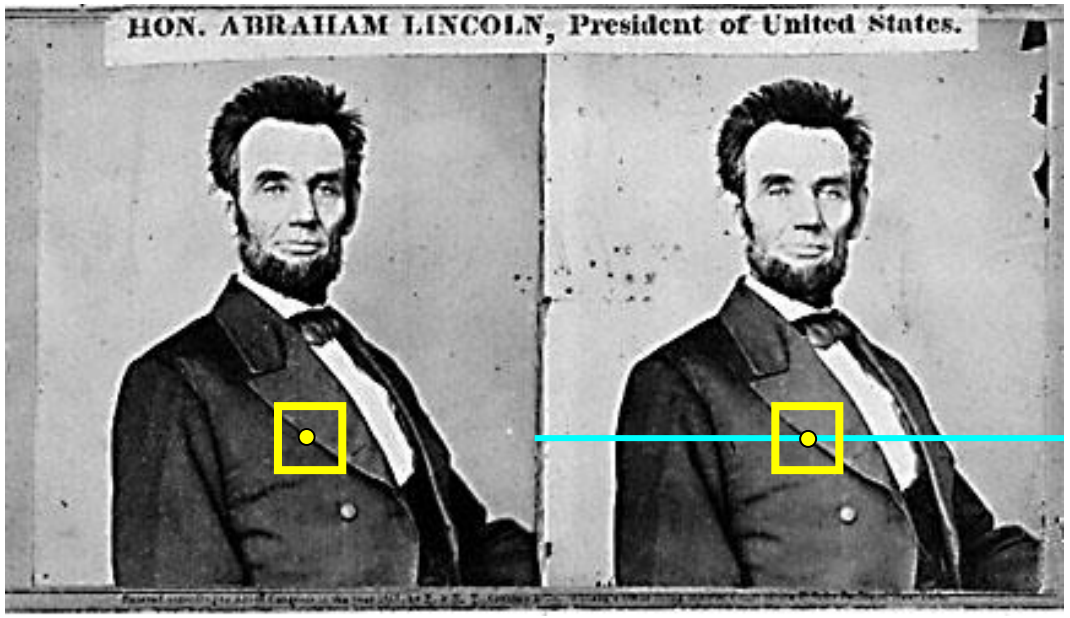


$$\frac{x}{f} = \frac{B_1}{z} \qquad \frac{x¢}{f} = \frac{B_2}{z}$$

$$\frac{x - x¢}{f} = \frac{B_1 - B_2}{z}$$

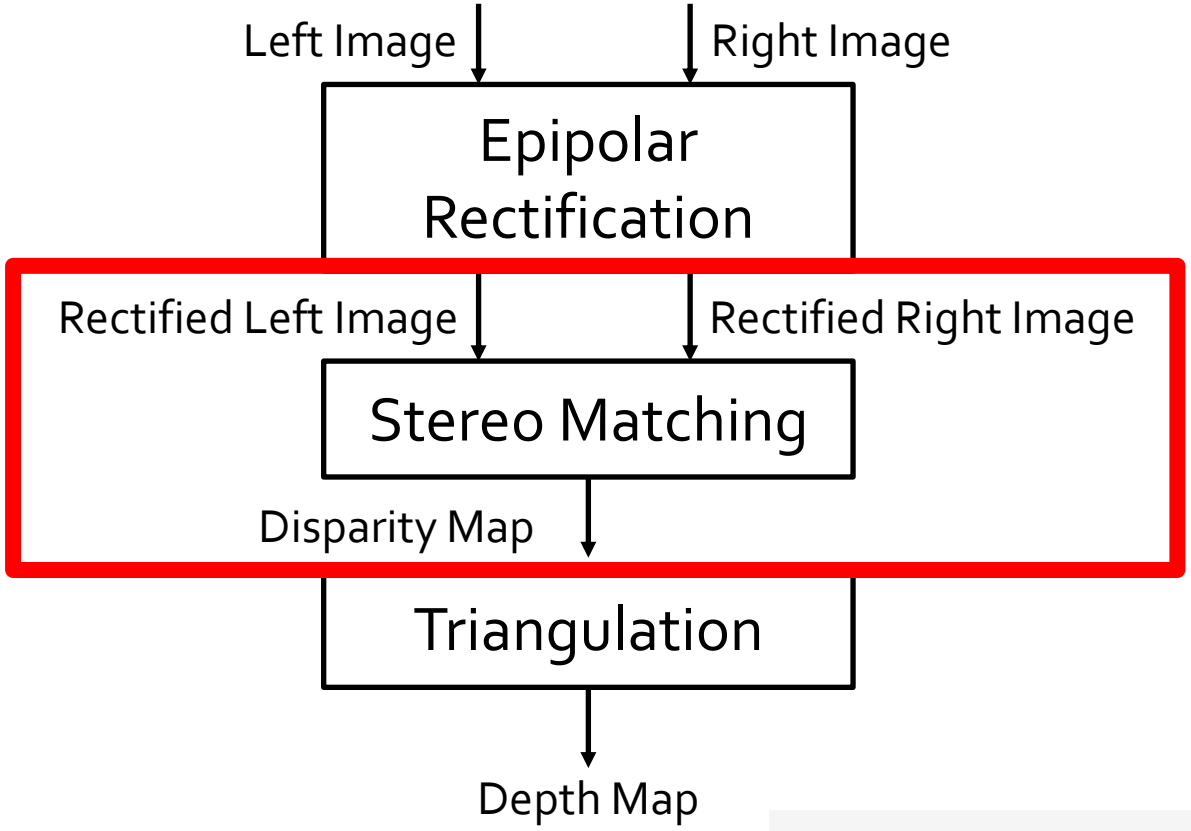
$$disparity = x - x' = \frac{B \cdot f}{z}$$

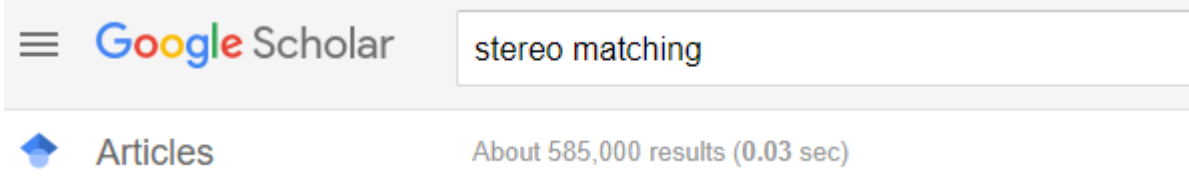
Basic stereo matching algorithm



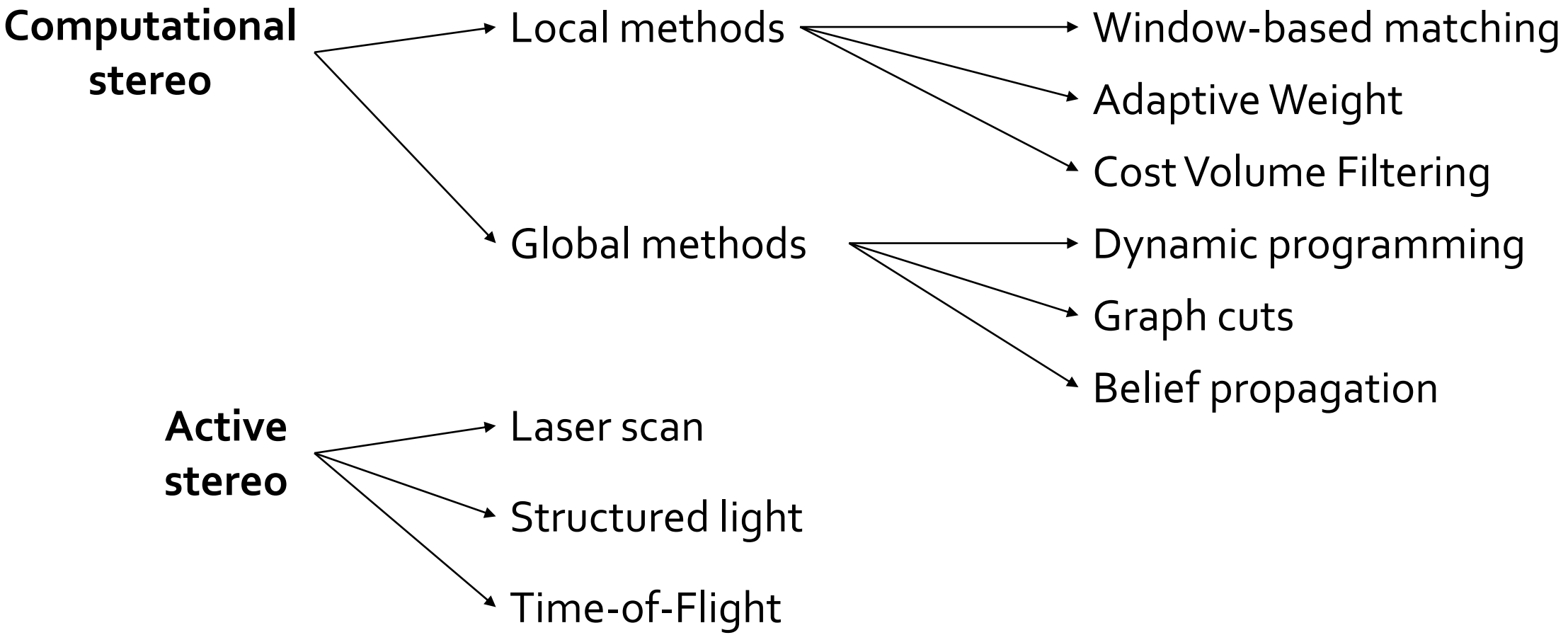
- **If necessary, rectify the two stereo images to transform epipo lar lines into scanlines**
- **For each pixel *x* in the first image**
  - Find corresponding epipolar scanline in the right image
  - Examine all pixels on the scanline and pick the best match *x′*
  - Compute disparity $x–x'$ and set $\text{depth}(x) = B*f/(x–x')$

So the key point is..





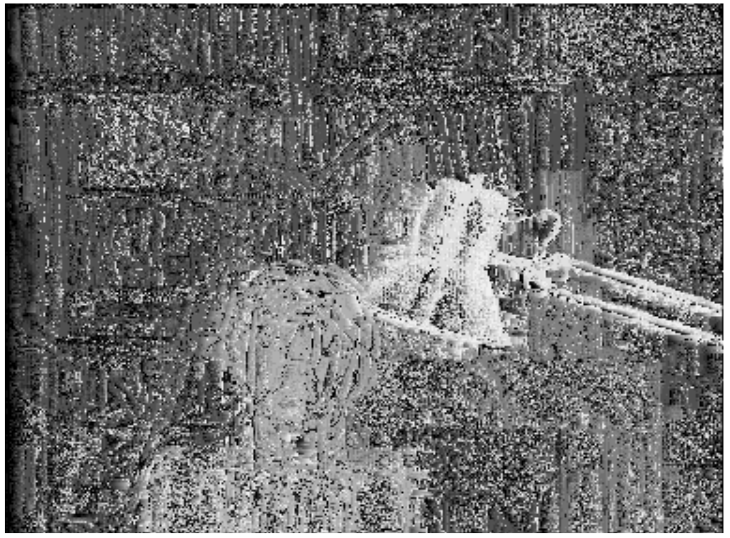
Taxonomy of stereo matching



+ Deep learning based approaches

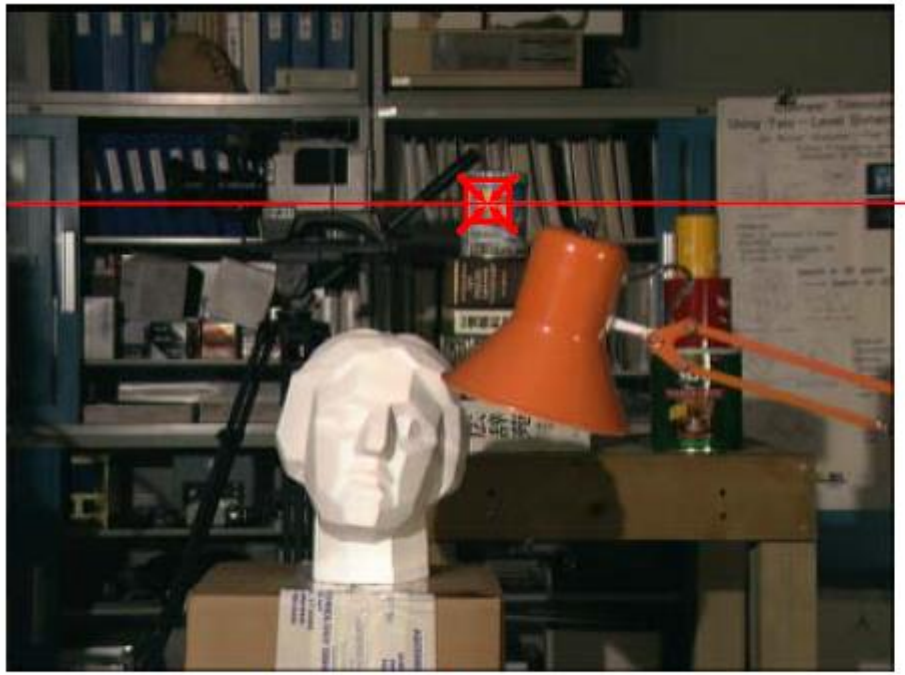
A naïve stereo algorithm

- **For each pixel *x* of the left image:**
  - Compare color of *x* against the color of each pixel on the same horizontal scanline in the right image.
  - Select the pixel of most similar color as matching point



Window-based matching

- **Instead of matching single pixel,**
  - center a small window on a pixel and match the whole window in the right image.



(a) Left image

(b) Right image

Window-based matching

- In a formal way,
  - the disparity $d_{\boldsymbol{x}}$ of a pixel $\boldsymbol{x}$ in the left image is computed as

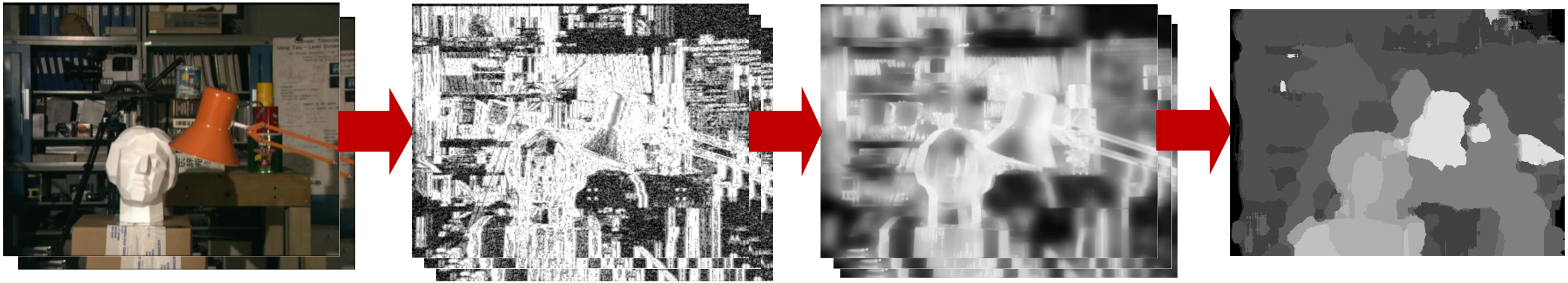
$$d_{\boldsymbol{x}} = \underset{0 \le d \le d_{max}}{\operatorname{argmin}} \sum_{\boldsymbol{q} \in W_{\boldsymbol{x}}} c(\boldsymbol{q}, \boldsymbol{q} - \boldsymbol{d})$$

  - Where

Let's split this equation into three steps!

    - argmin returns the value at which the function takes a minimum
    - d_{max} is a parameter defining the maximum disparity (search range).
    - $W_{\boldsymbol{x}}$ is the set of all pixels inside the window centred on $\boldsymbol{x}$
    - $c(\boldsymbol{q}, \boldsymbol{q} - \boldsymbol{d})$ is a function that computes the colour difference between a pixel $\boldsymbol{q}$ of the left image and a pixel $\boldsymbol{q} - \boldsymbol{d}$ of the right image (e.g. Sum of absolute difference in RGB values)

A common pipeline for local methods



Matching cost computation

Cost aggregation

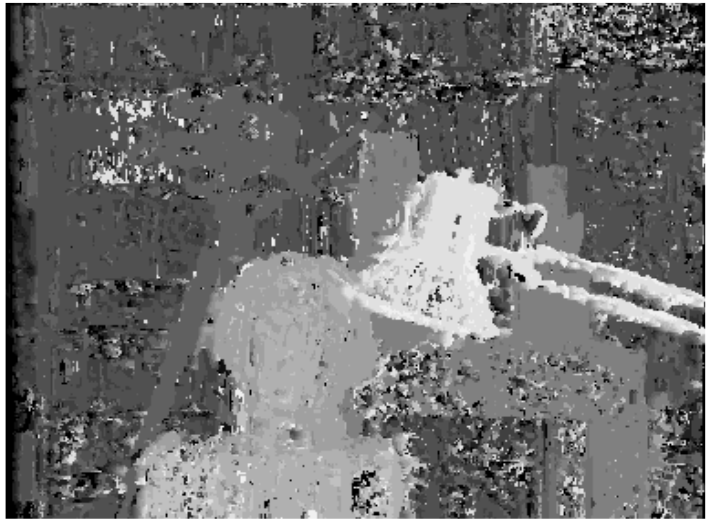
Disparity computation

$$c(\boldsymbol{q}, \boldsymbol{q}-\boldsymbol{d})$$

$$\sum_{\boldsymbol{q}\in W_{\boldsymbol{x}}} c(\boldsymbol{q}, \boldsymbol{q}-\boldsymbol{d})$$

$$d_{\boldsymbol{x}} = \underset{0\le d\le d_{max}}{\operatorname{argmin}} \sum_{\boldsymbol{q}\in W_{\boldsymbol{x}}} c(\boldsymbol{q}, \boldsymbol{q}-\boldsymbol{d})$$

Effect of window size



W = 3

- Smaller window
  - + More detail
  - • More noise

W = 21

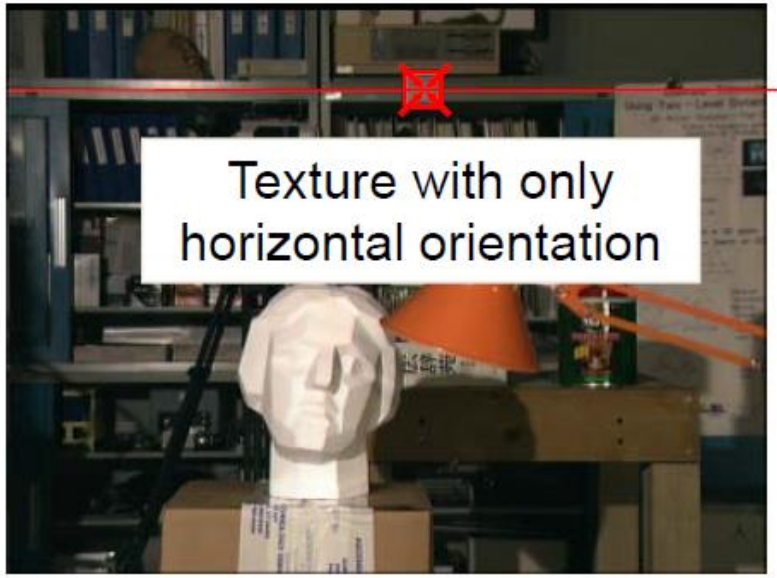
- Larger window
  - + Smoother disparity maps
  - • Less detail

Problem of Untextured Regions

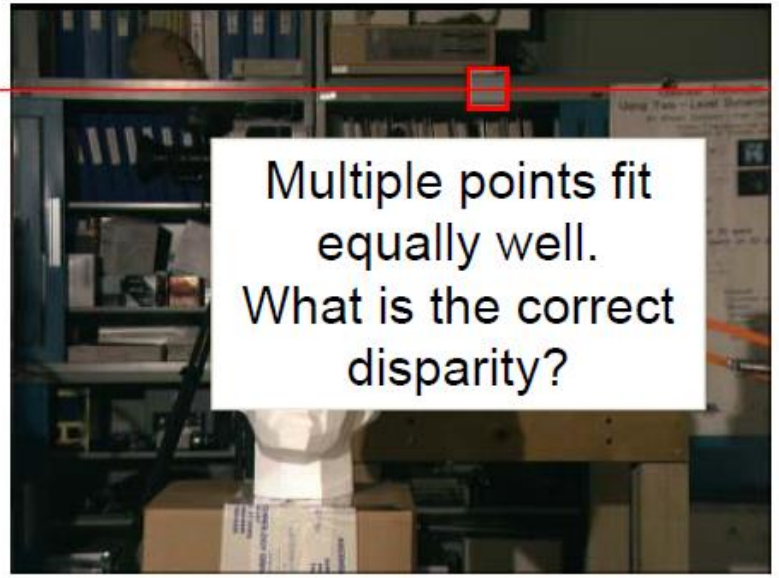
(a) Left image (b) Right image

– There has to be a certain amount of colour variation inside the window

Aperture Problem



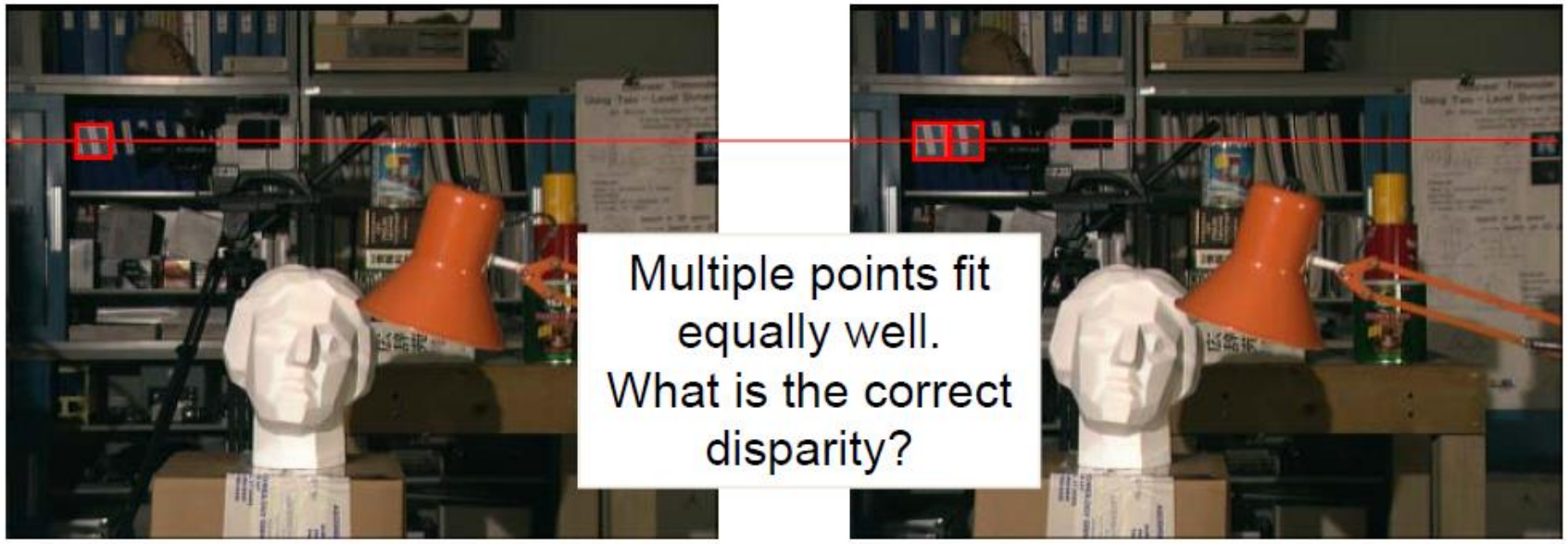
(a) Left image



(b) Right image

– There needs to be a certain amount of texture with vertical orientation

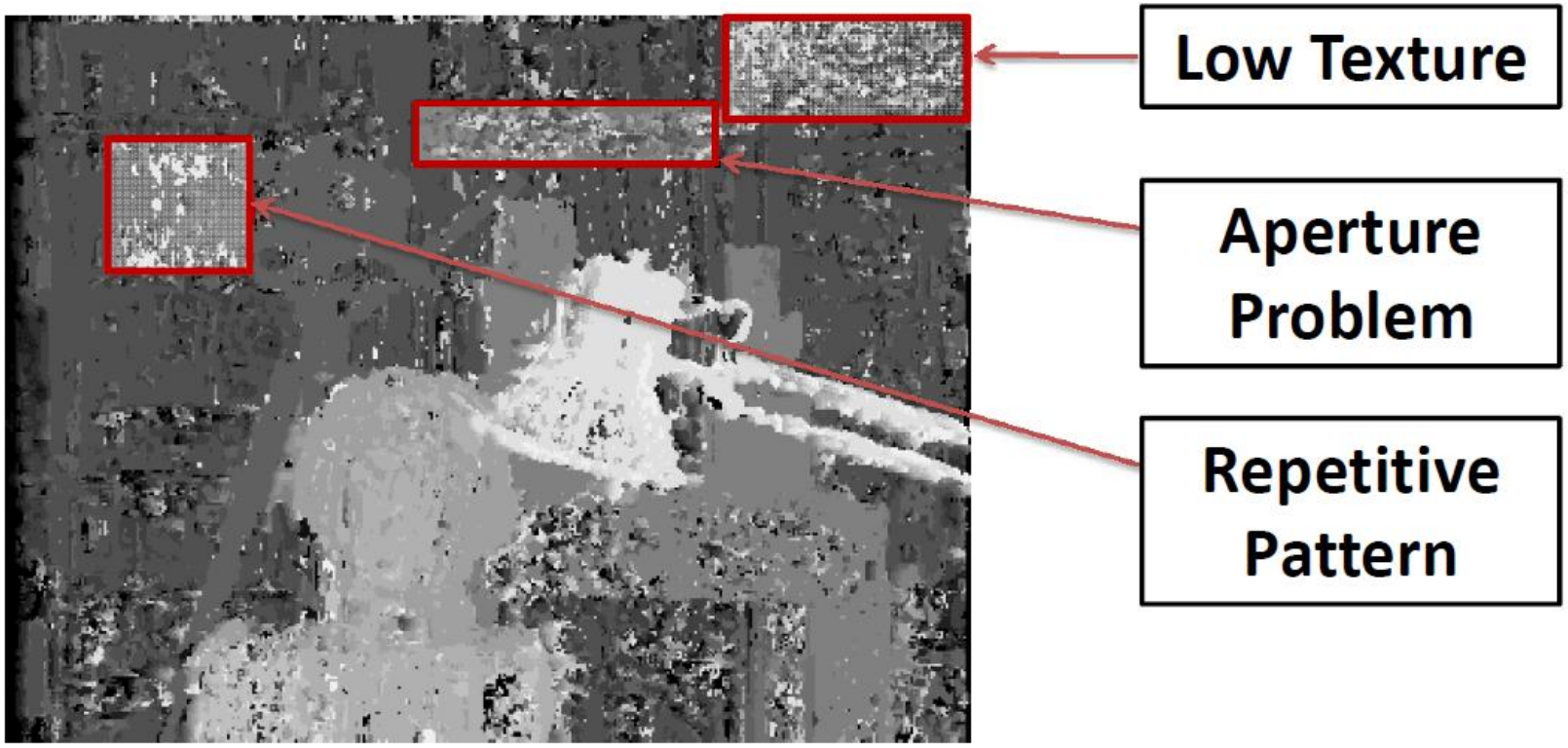
Problem of Repetitive Patterns



(a) Left image (b) Right image

– There needs to be a certain amount of repetitive texture

Effect of these problems



Window size = 3x3 pixels

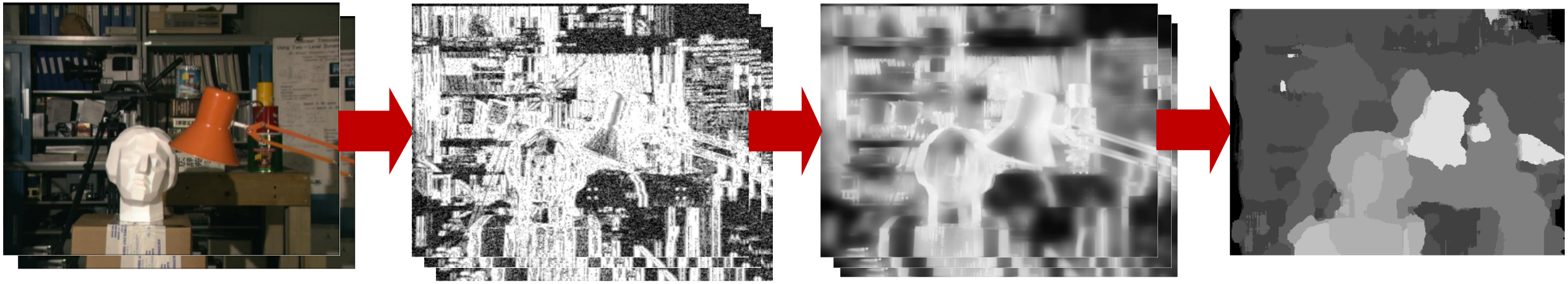
Foreground fattening problem

Ground Truth Disparities

Window size = 21x21 pixels

– Foreground objects are clearly enlarged when using large kernels (windows)

Foreground fattening problem: why?



Matching cost computation

Cost aggregation

Disparity computation

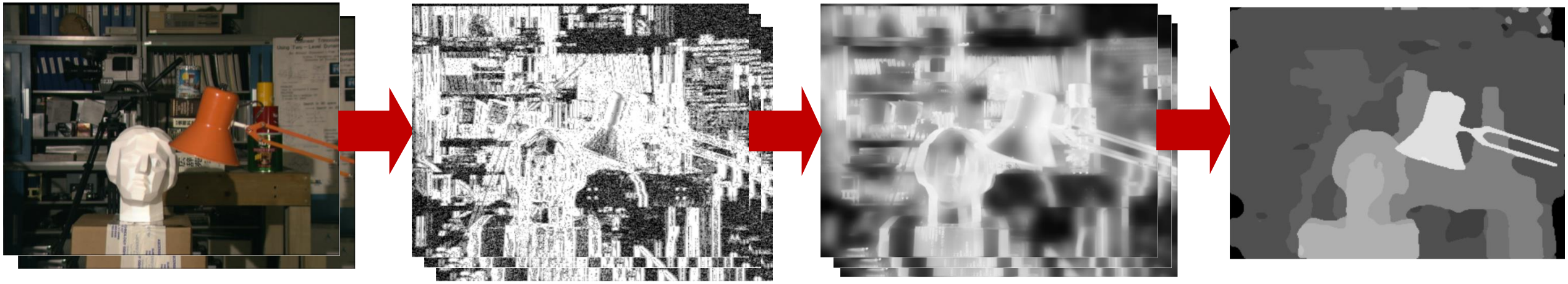
$$c(\boldsymbol{q}, \boldsymbol{q}-\boldsymbol{d})$$

$$\sum_{\boldsymbol{q}\in W_{\boldsymbol{x}}} c(\boldsymbol{q}, \boldsymbol{q}-\boldsymbol{d})$$

$$d_{\boldsymbol{x}} = \underset{0\le d\le d_{max}}{\operatorname{argmin}} \sum_{\boldsymbol{q}\in W_{\boldsymbol{x}}} c(\boldsymbol{q}, \boldsymbol{q}-\boldsymbol{d})$$

All pixels in $W_{\boldsymbol{x}}$ have the same influence on the aggregated costs

Adaptive support weight



Matching cost computation

Cost aggregation

Disparity computation

$$c(\boldsymbol{q}, \boldsymbol{q}-\boldsymbol{d})$$

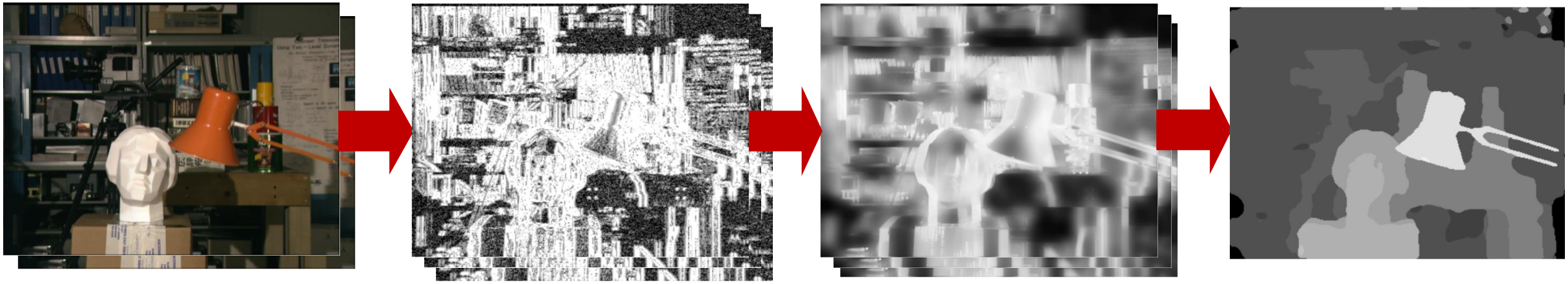
$$\sum_{\boldsymbol{q}\in W_{\boldsymbol{x}}} w(\boldsymbol{x},\boldsymbol{q})\, c(\boldsymbol{q}, \boldsymbol{q}-\boldsymbol{d})$$

$$d_{\boldsymbol{x}} = \underset{0\le d\le d_{max}}{\operatorname{argmin}} \sum_{\boldsymbol{q}\in W_{\boldsymbol{x}}} c(\boldsymbol{q}, \boldsymbol{q}-\boldsymbol{d})$$

1. Assumption: Two points are likely to lie on the same disparity if they are similar in color
2. Only pixels that lie on the same disparity contribute to the aggregated matching costs
   -> No foreground fattening

[Yoon,CVPR05]

Adaptive support weight



Matching cost computation

Cost aggregation

Disparity computation

$$c(\boldsymbol{q}, \boldsymbol{q}-\boldsymbol{d})$$

$$\sum_{\boldsymbol{q}\in W_{\boldsymbol{x}}} w(\boldsymbol{x},\boldsymbol{q})\,c(\boldsymbol{q},\boldsymbol{q}-\boldsymbol{d})$$

$$d_{\boldsymbol{x}} = \underset{0\le d\le d_{max}}{\operatorname{argmin}} \sum_{\boldsymbol{q}\in W_{\boldsymbol{x}}} c(\boldsymbol{q},\boldsymbol{q}-\boldsymbol{d})$$

$$w(\boldsymbol{x},\boldsymbol{q}) = \boldsymbol{exp}\left(-\left(\frac{\Delta c_{\boldsymbol{xq}}}{\sigma_c} + \frac{\Delta s_{\boldsymbol{xq}}}{\sigma_s}\right)\right)$$

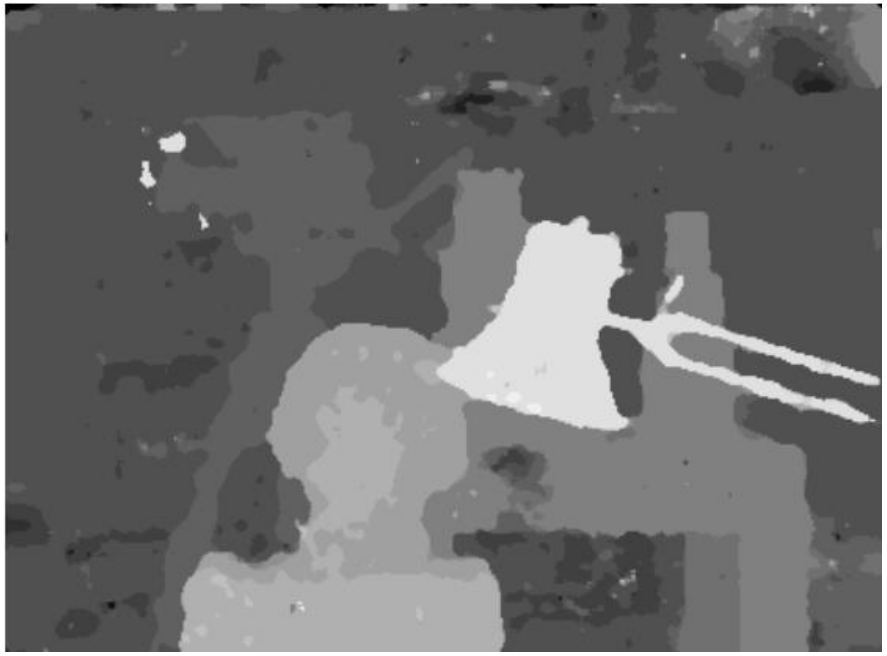
Δc_{xq}: colour distance between **x** and **q**

Δs_{xq}: spatial distance between **x** and **q**

[Yoon,CVPR05]

Results

Window-based



Adaptive support-weight

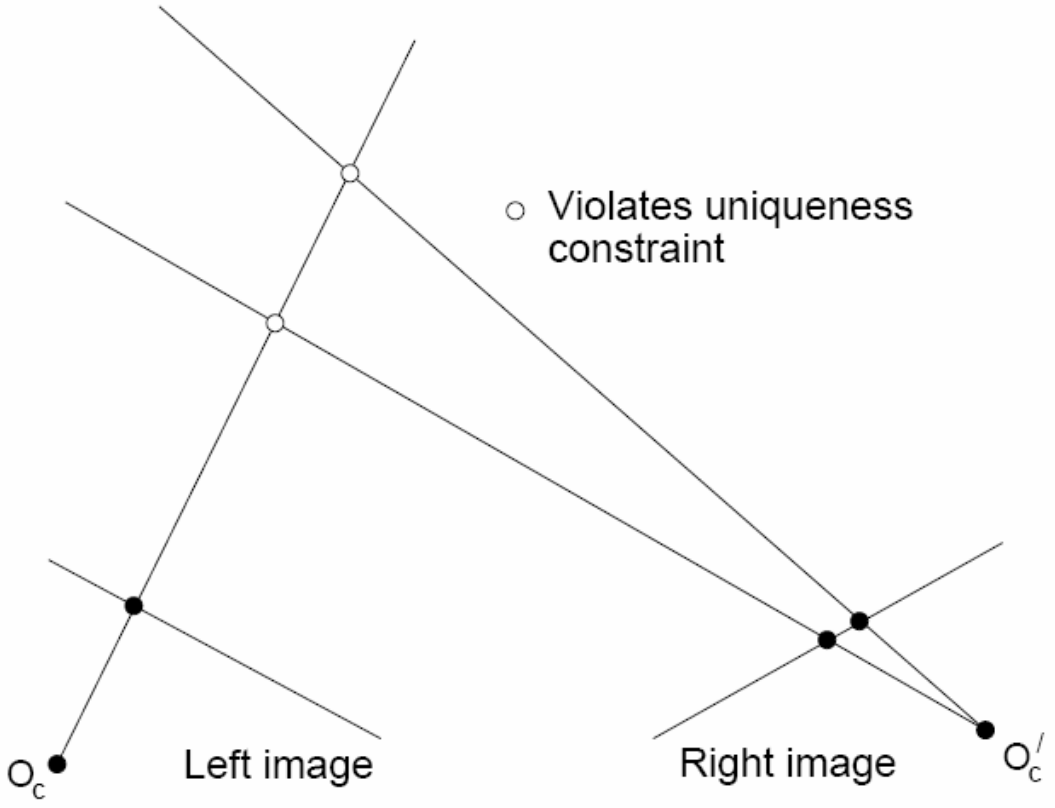
[Yoon,CVPR05]

How can we improve window-based matching?

- The similarity constraint is local
  - Each reference window is matched independently
- Need to enforce non-local correspondence constraints

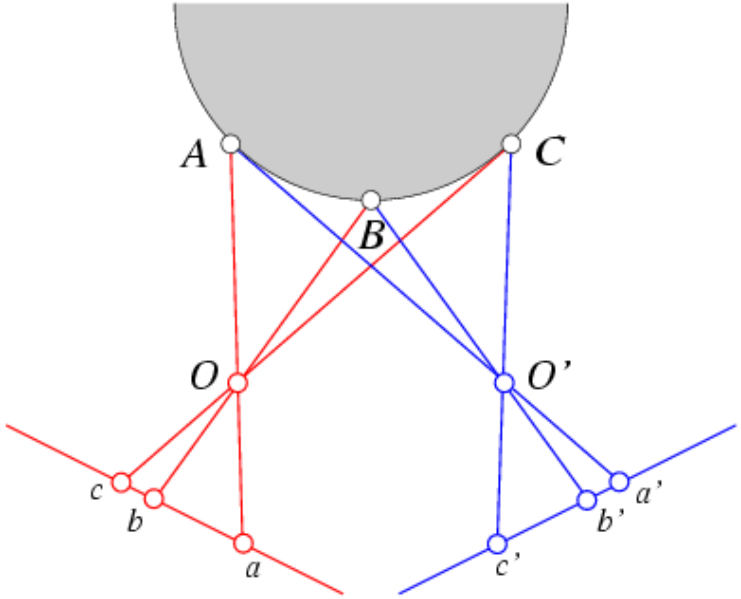
Non-local constraints

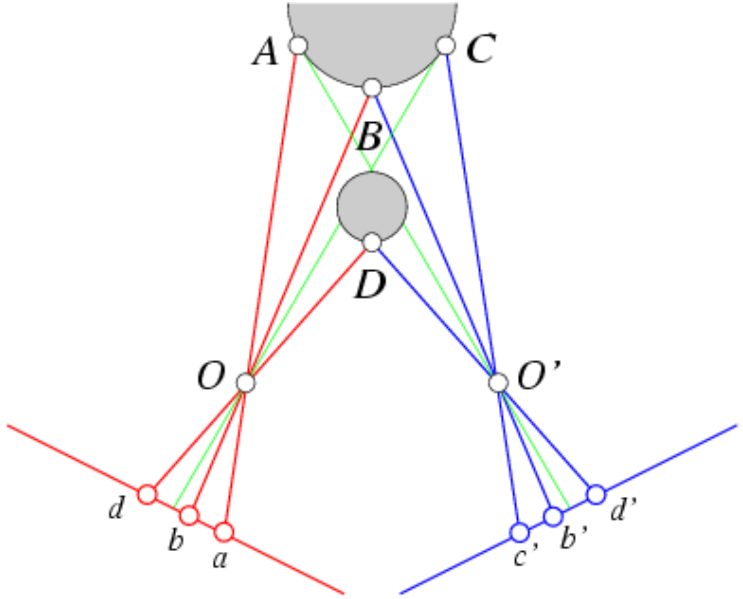
- **Uniqueness**
  - For any point in one image, there should be at most one matching point in the other i mage



Non-local constraints

- **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 constraint doesn't hold

Non-local constraints

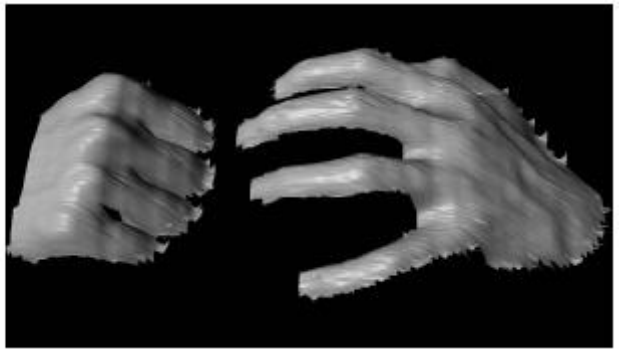
- **Uniqueness**
  - For any point in one image, there should be at most one matching point in the other i mage
- **Ordering**
  - Corresponding points should be in the same order in both views
- **Smoothness**
  - We expect disparity values to change slowly (for the most part)

Stereo datasets

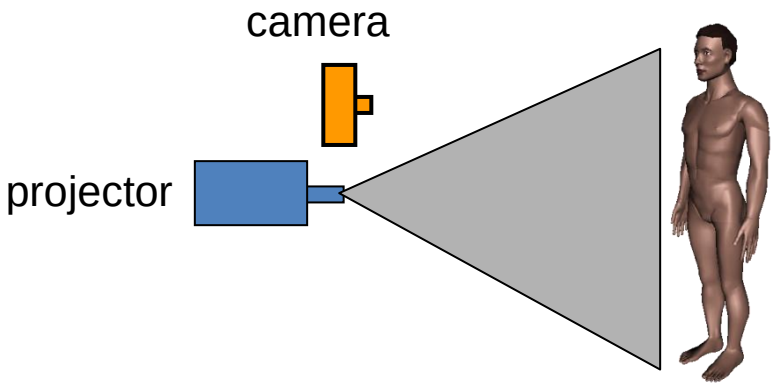
- Middlebury stereo datasets
- KITTI
- Synthetic data

Active stereo with structured light



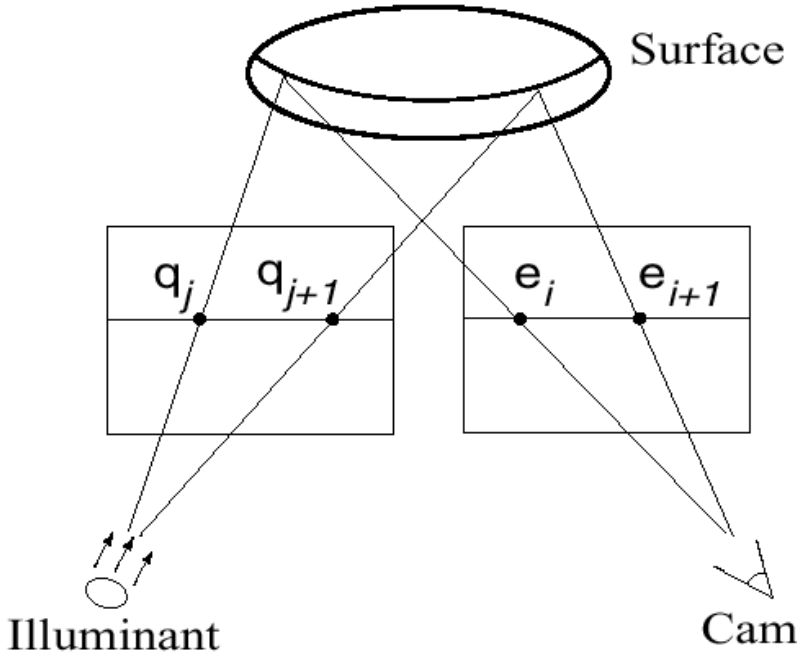


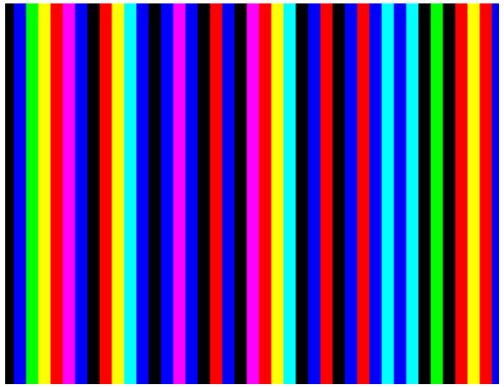
- **Project "structured" light patterns onto the object**
  - Simplifies the correspondence problem
  - Allows us to use only one camera

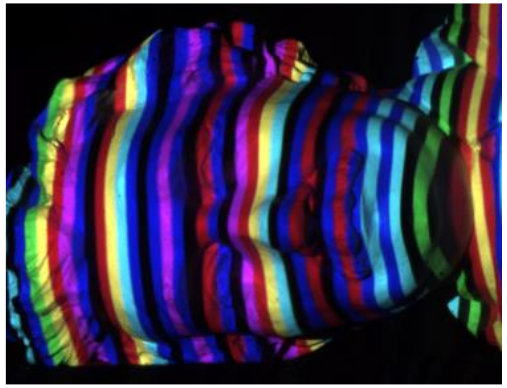


L. Zhang, B. Curless, and S. M. Seitz. Rapid Shape Acquisition Using Color Structured Light and Multi-pass Dynamic Programming. *3DPVT* 2002

Active stereo with structured light

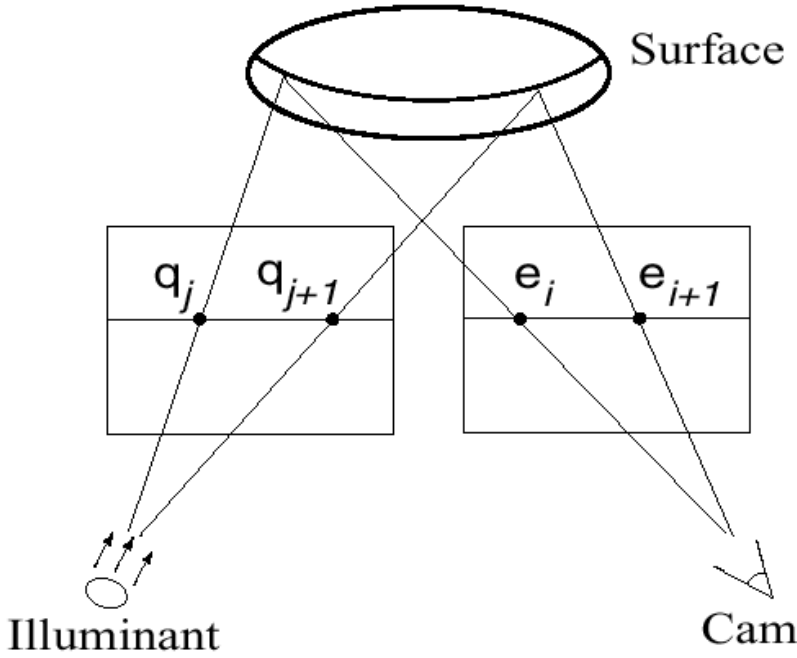


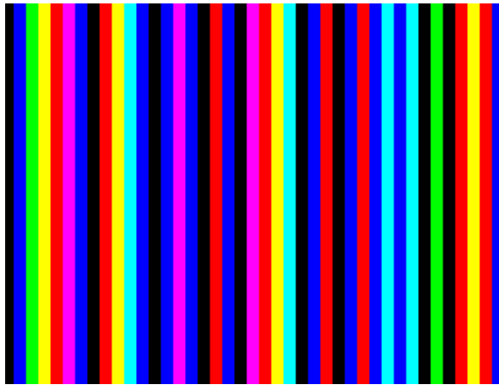


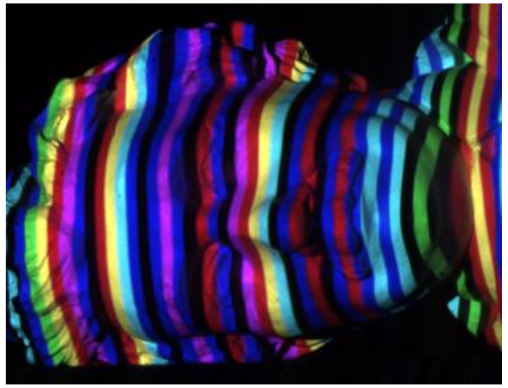


L. Zhang, B. Curless, and S. M. Seitz. Rapid Shape Acquisition Using Color Structured Light and Multi-pass Dynamic Programming. *3DPVT* 2002

Active stereo with structured light





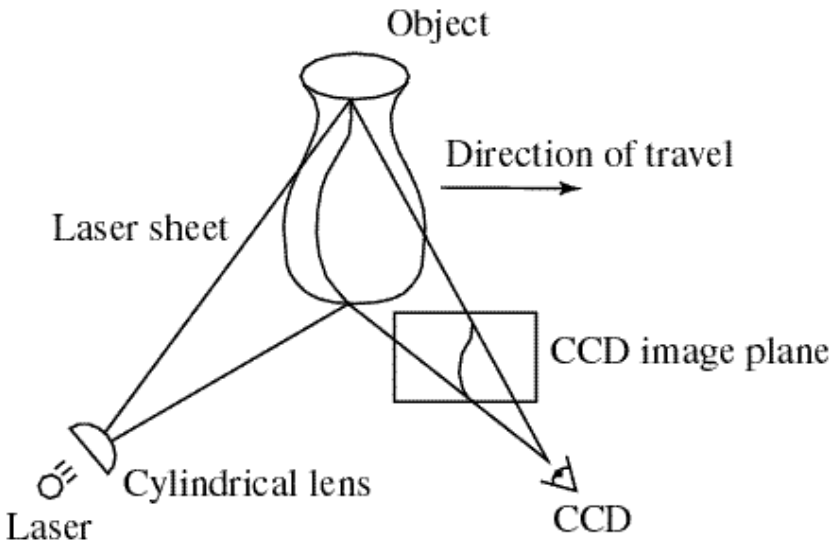


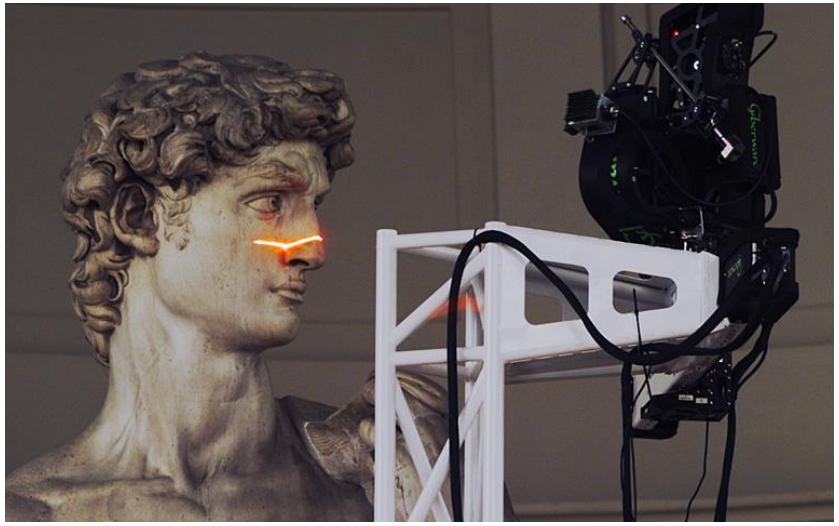
http://en.wikipedia.org/wiki/Structured-light_3D_scanner

Kinect: Structured infrared light

http://bbzippo.wordpress.com/2010/11/28/kinect-in-infrared/

Laser scanning





Digital Michelangelo Project
Levoy et al.
http://graphics.stanford.edu/projects/mich/

- **Optical triangulation**
  - Project a single stripe of laser light
  - Scan it across the surface of the object
  - This is a very precise version of structured light scanning

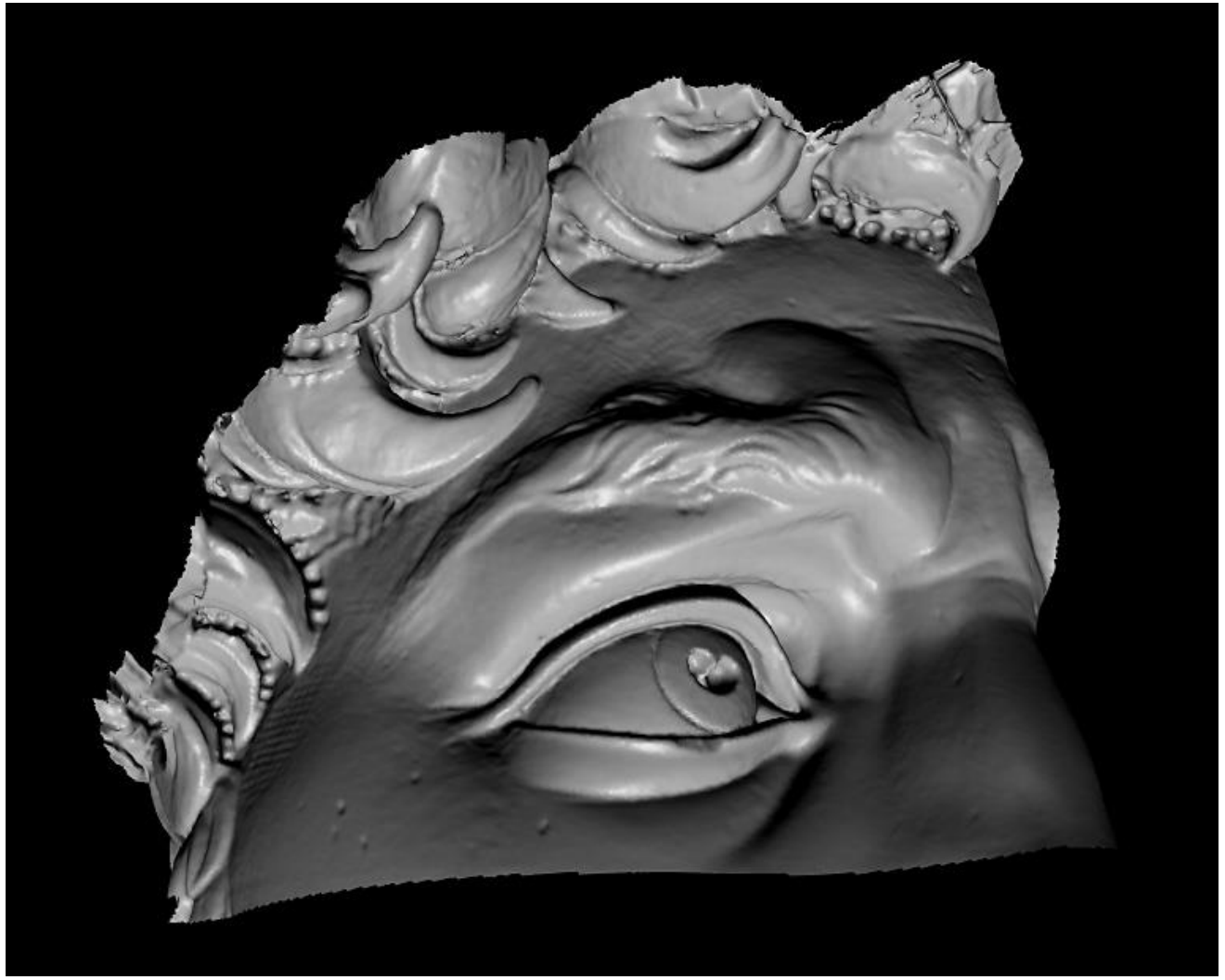
Source: S. Seitz

Laser scanned models

The Digital Michelangelo Project, Levoy et al.

Source: S. Seitz

Laser scanned models



The Digital Michelangelo Project, Levoy et al.

Source: S. Seitz