Lecture 13 –Stereo vision

Dec 11th, 2018

Danping Zou,

Associate Professor

Institute for Sensing and Navigation

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Outline

- Stereo vision
  - Depth from stereo
  - Ideal stereo system
  - Disparity
- Stereo calibration & rectification
- Stereo matching
  - Sparse matching
  - Dense matching
    - Local approach - Winner-take-all
    - Matching costs: SAD, SSD, ZNCC
    - Typical failures
    - Global approach - Graph cut, Belief Propagation

Stereo vision

Two view geometry

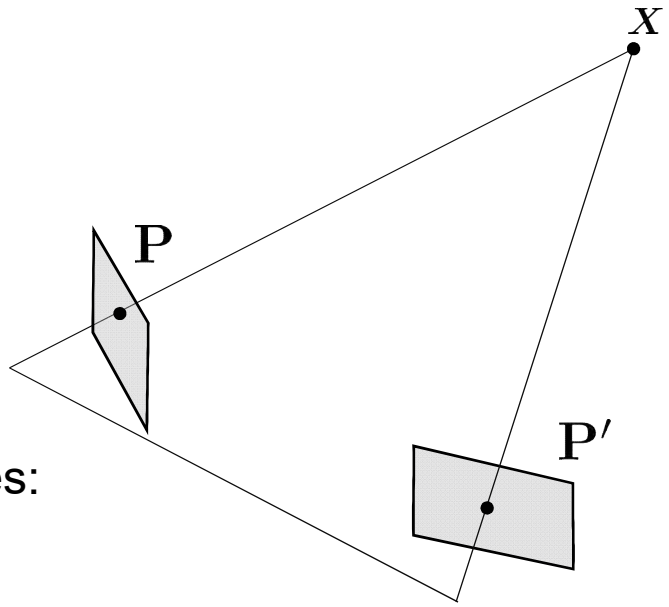
Stereo matching

3D world

Stereo vision

- Depth from stereo
  - For each pair of corresponding points, we can apply triangulation to get the coordinates of the 3D point

$$x \leftrightarrow x' \Rightarrow X$$



Method 1 - Using **homogeneous** coordinates:

$$\begin{aligned} x \times PX &= 0 \\ x' \times P'X &= 0 \end{aligned} \Rightarrow A_{6\times 4}X = 0_{6\times 1}$$

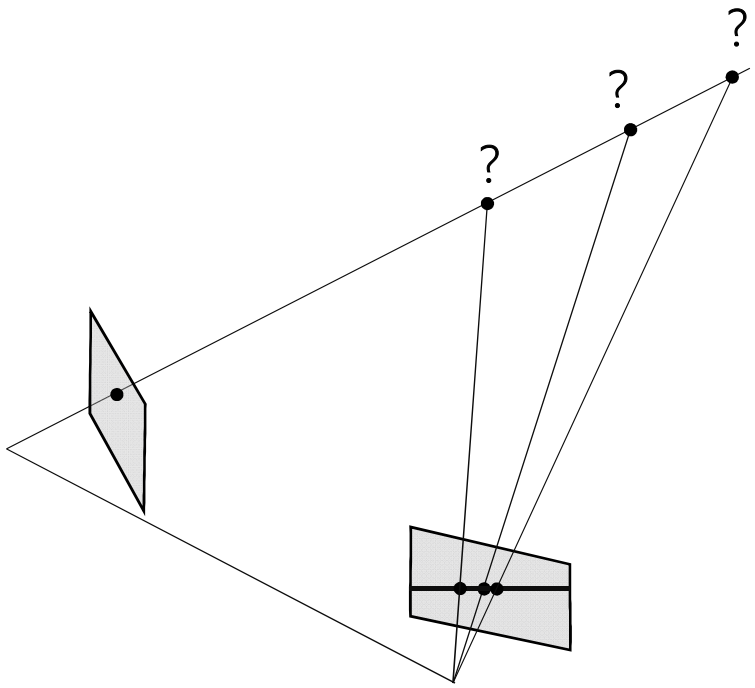
Method 2 - Using **inhomogeneous** coordinates:

$$u = \frac{P_1^T X}{P_3^T X} \quad v = \frac{P_2^T X}{P_3^T X}$$

$$u' = \frac{P_1'^T X}{P_3'^T X} \quad v' = \frac{P_2'^T X}{P_3'^T X} \Rightarrow A_{4\times 3}x = b_{4\times 1}$$

Stereo vision

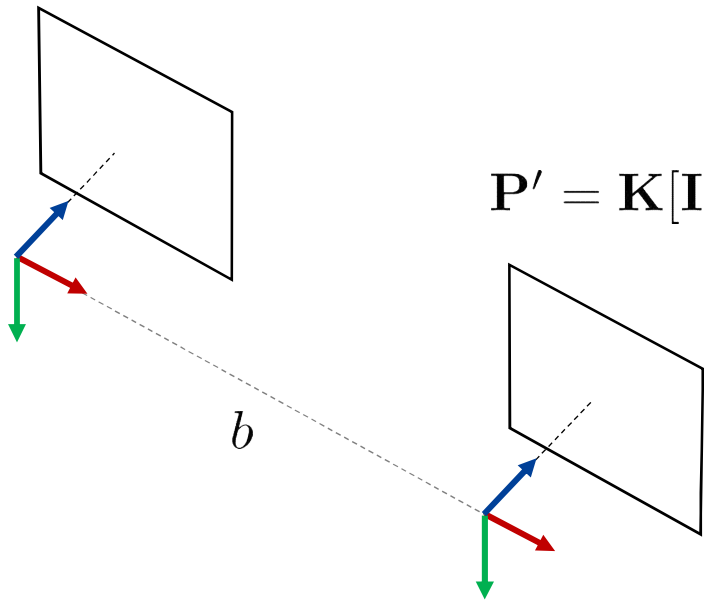
- The key for stereo vision is seeking the point correspondence along the epipolar line – stereo matching.
- For each point, we need to
  - 1) compute its epiplor line, and
  - 2) search its correspondence along the epipolar line



Stereo vision

- There is one good setting for stereo matching - two cameras with only a side translation (in x direction) and the same intrinsic parameters.

$$\mathbf{P} = \mathbf{K}[\mathbf{I}\ \mathbf{0}]$$

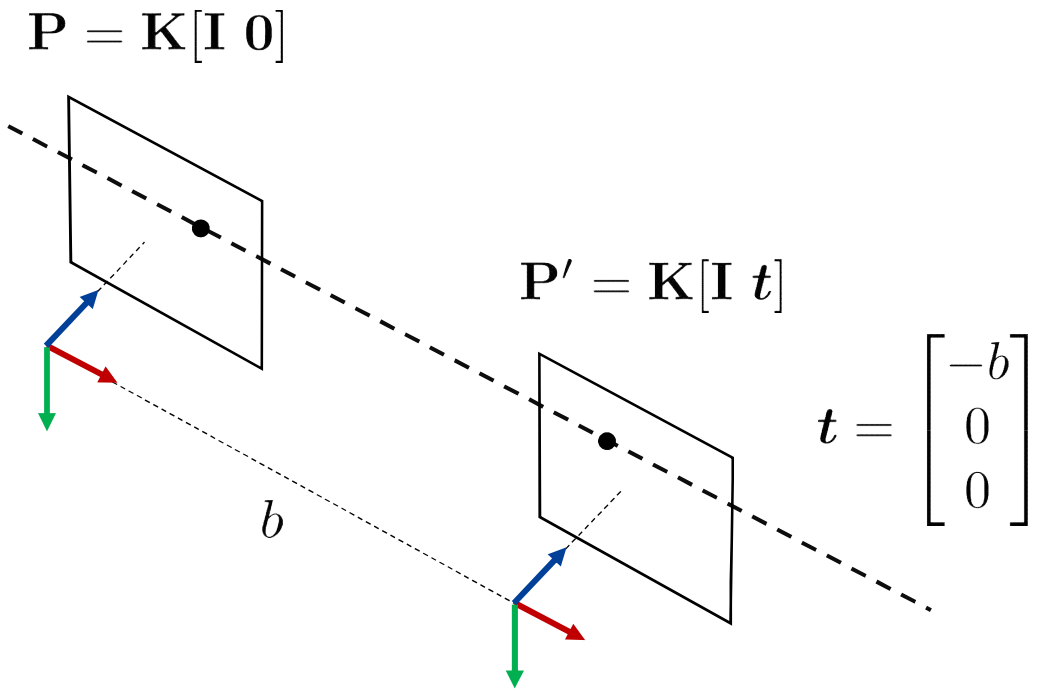


$$\mathbf{P}' = \mathbf{K}[\mathbf{I}\ \boldsymbol{t}]$$

$$\boldsymbol{t} = \begin{bmatrix} -b \\ 0 \\ 0 \end{bmatrix}$$

Stereo vision

- In such cases, we search the corresponding points simply at the corresponding rows.



Stereo vision

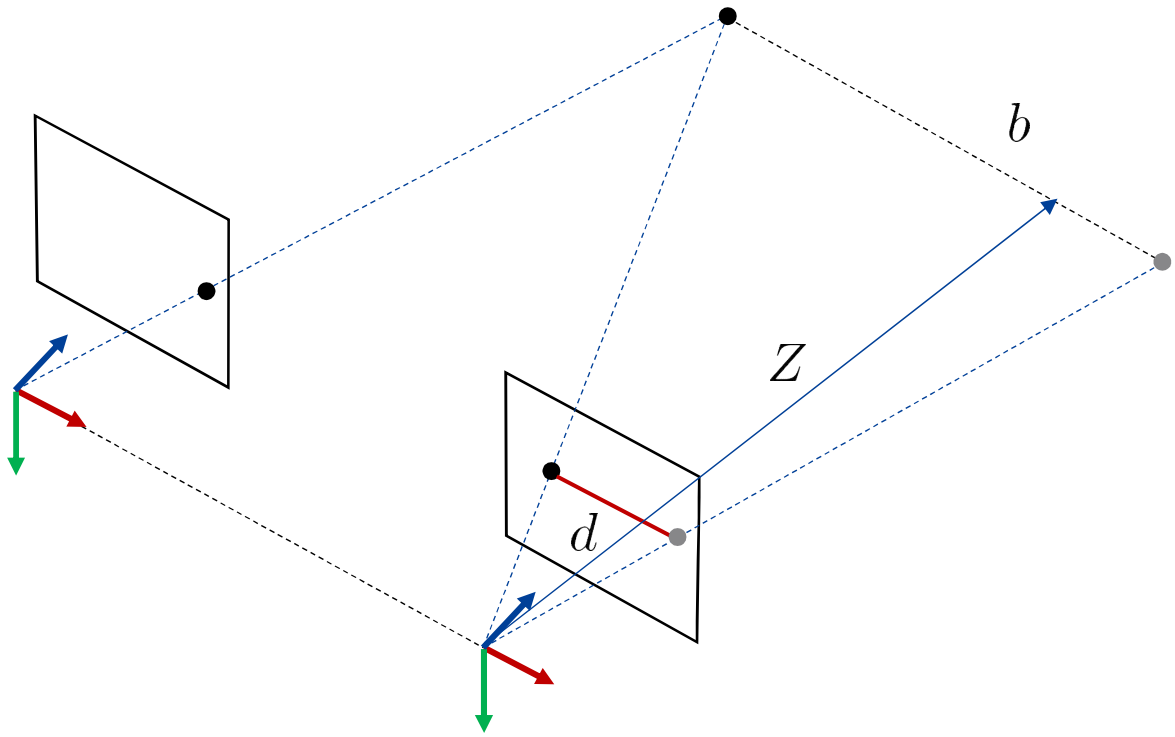
- **Disparity vs Depth** - The pixel distance between the corresponding points is inversely proportional to the depth.

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$Z = f_x \frac{b}{d}$$

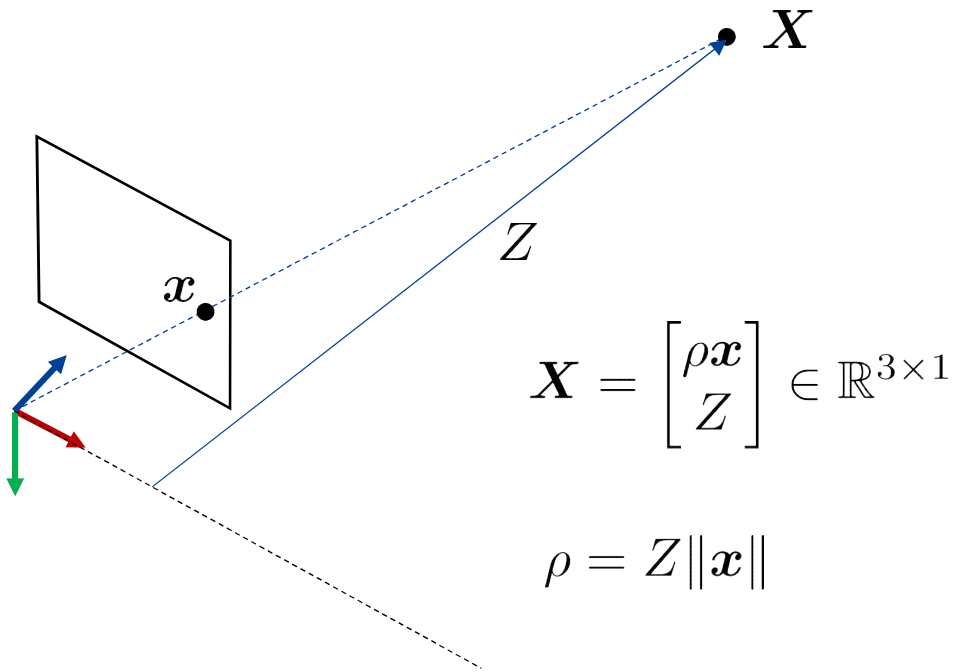
d : Disparity

Z : Depth



Stereo vision

- 3D from depth – Depth is the Z value in the camera frame

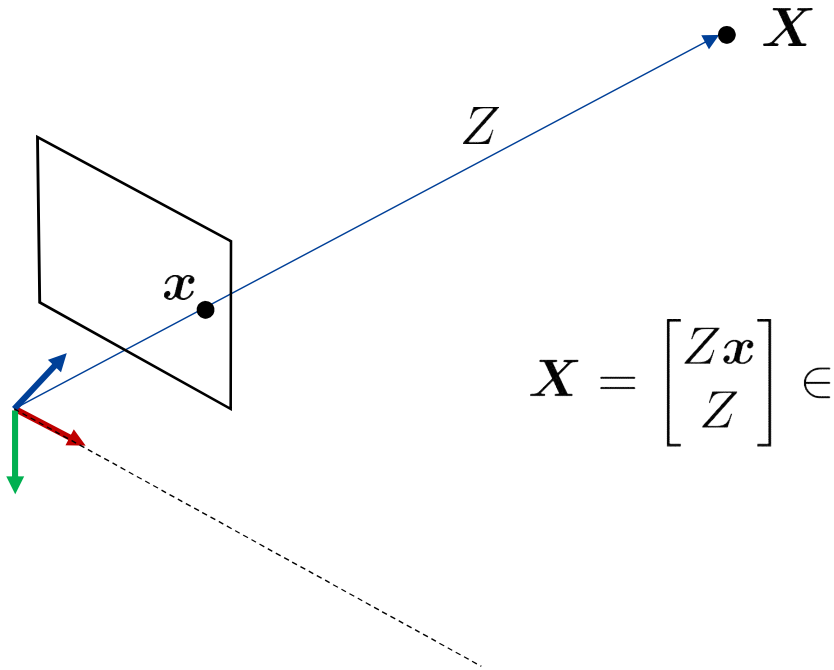


$$\boldsymbol{X} = \begin{bmatrix} \rho \boldsymbol{x} \\ Z \end{bmatrix} \in \mathbb{R}^{3\times 1}$$

$$\rho = Z\|\boldsymbol{x}\|$$

Stereo vision

- 3D from depth – Depth is the distance between the 3D point and the optical center.



$$\boldsymbol{X} = \begin{bmatrix} Z\boldsymbol{x} \\ Z \end{bmatrix} \in \mathbb{R}^{3\times 1}$$

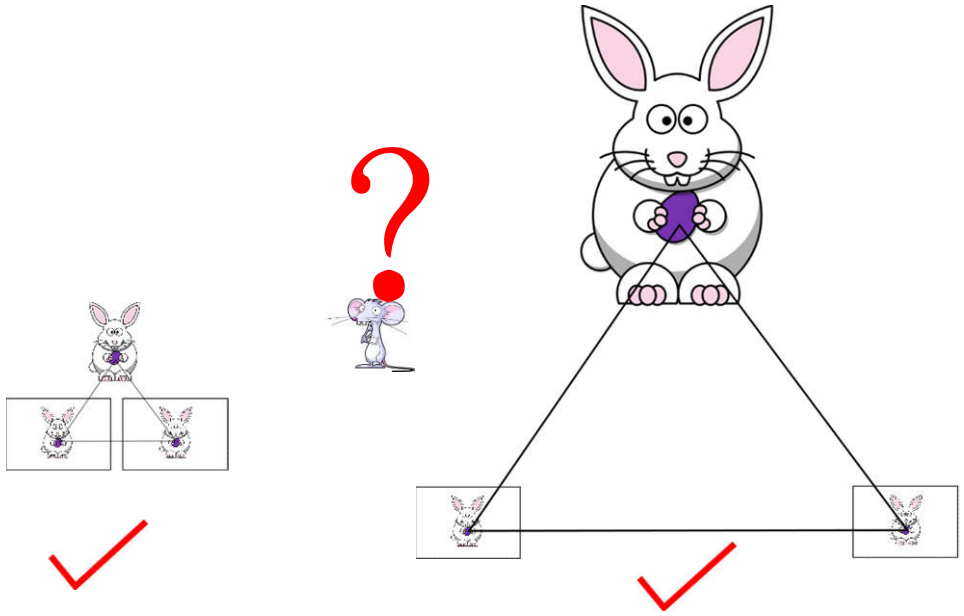
Stereo calibration

- Stereo calibration is to obtain the relative pose between two views (or the stereo cameras)
- From two-view geometry, we can solve the relative pose through the following steps:
  - 1. calibrate the camera intrinsic parameters
  - 2. take two pictures by your phone
  - 3. Match feature points (SIFT, SURF)
  - 4. Use Eight/Five point RANSAC algorithm to estimate the fundamental matrix and reject the outliers
  - 5. extract the $\mathbf{R}$ and $\boldsymbol{t}$ from the essential matrix $\mathbf{E}$

Scale ambiguity

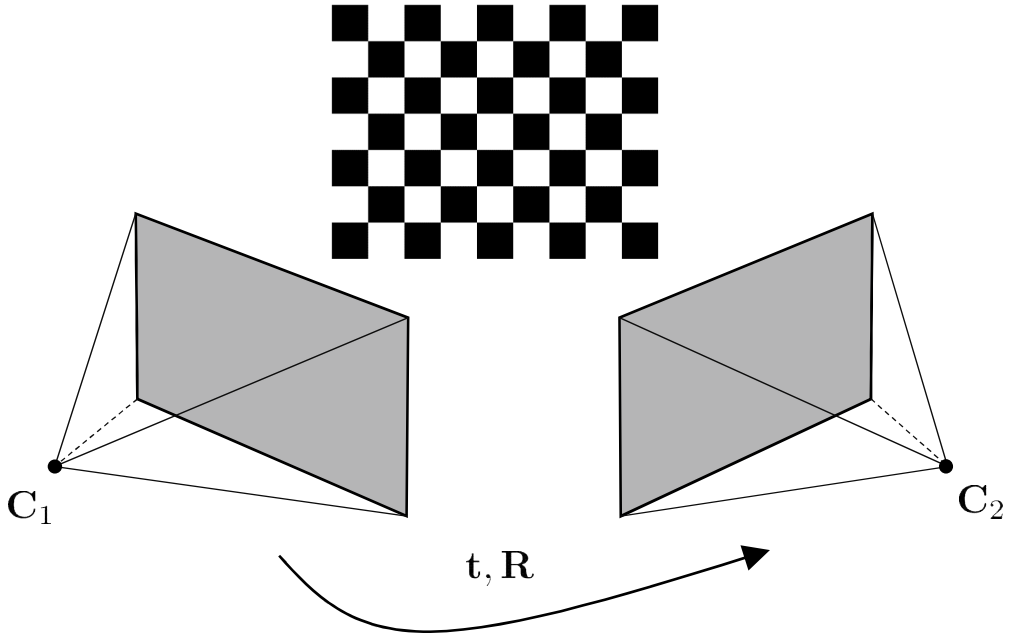
- But the translation vector extracted from the essential matrix has arbitrary length

$$\mathbf{E} = [t]_\times \mathbf{R} \sim s\mathbf{E} = [st]_\times \mathbf{R}$$



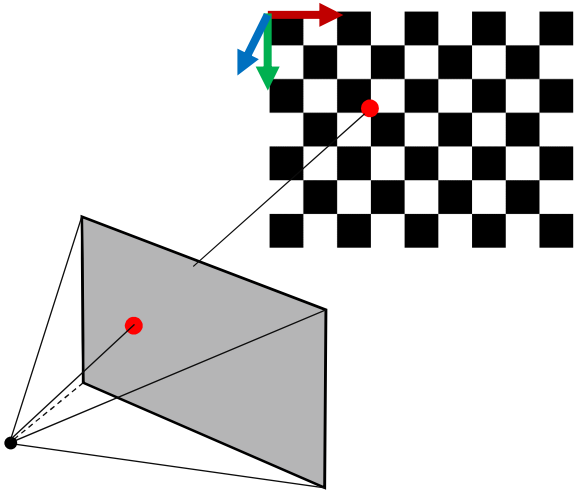
Stereo calibration

- Get the extract **relative pose** between two views using the known object (usually a checkerboard)



Stereo calibration

- Suppose the camera intrinsic parameters have been calibrated for each camera.
- Given a checkerboard pattern, we can compute the camera pose with respect to the world frame built on the checkerboard.



$$\boldsymbol{x} \sim \mathbf{H}\tilde{\boldsymbol{X}}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \mathbf{K}[\boldsymbol{r}_1\, \boldsymbol{r}_2\, \boldsymbol{t}] \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Stereo calibration

- After the homography matrices being estimated, we decompose the estimated homography into camera pose

$$\mathbf{H} = \mathbf{K}[\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{t}]$$

$$\Rightarrow [\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{t}] = \mathbf{K}^{-1}\mathbf{H}$$

$$\Rightarrow \boldsymbol{r}_3 = \boldsymbol{r}_1 \times \boldsymbol{r}_2 \Rightarrow \mathbf{A} = [\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3] \approx \mathbf{R}$$

- Use SVD to the rotation matrix

$$\mathbf{A} = \mathbf{U}\begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix}\mathbf{V}^{\mathrm{T}} \Rightarrow \mathbf{R} = \mathbf{U}\mathbf{V}^{\mathrm{T}}$$

Stereo calibration

- After that, we get the camera poses with respect to the calibration board for each view:

$$\begin{cases} \boldsymbol{x}_1 &= \mathbf{K}[\mathbf{R}_1\, \boldsymbol{t}_1]\boldsymbol{X} \\ \boldsymbol{x}_2 &= \mathbf{K}[\mathbf{R}_2\, \boldsymbol{t}_2]\boldsymbol{X} \end{cases}$$

- Transform the 3D point $\boldsymbol{X}$ from the world frame into the camera frame in the left camera.

$$\boldsymbol{X}_1 \sim \begin{bmatrix} \mathbf{R}_1 & \boldsymbol{t}_1 \\ 0 & 1 \end{bmatrix} \boldsymbol{X} \Rightarrow \boldsymbol{X} \sim \begin{bmatrix} \mathbf{R}_1^{\mathrm{T}} & -\mathbf{R}_1^{\mathrm{T}}\boldsymbol{t}_1 \\ 0 & 1 \end{bmatrix} \boldsymbol{X}_1$$

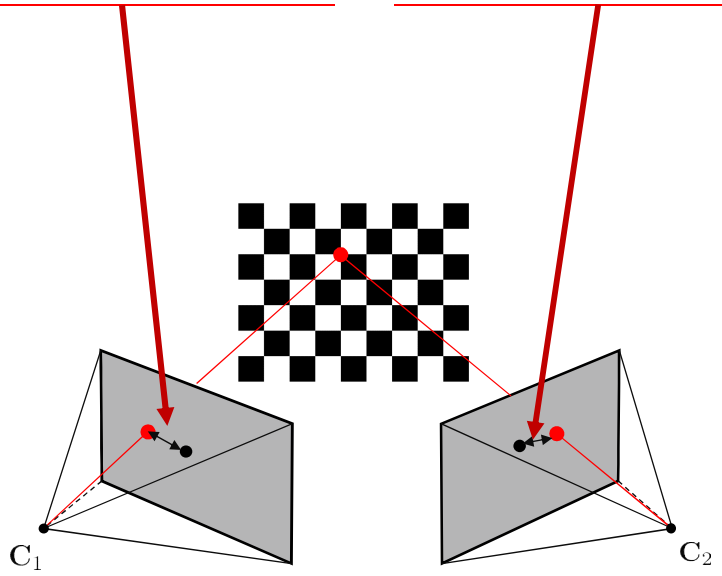
- Then we have

$$\begin{cases} \boldsymbol{x}_1 &= \mathbf{K}[\mathbf{I}\,\mathbf{0}]\boldsymbol{X}_1 \\ \boldsymbol{x}_2 &= \mathbf{K}[\mathbf{R}\,\boldsymbol{t}]\boldsymbol{X}_1 \end{cases} \quad (\mathbf{R} = \mathbf{R}_2\mathbf{R}_1^{\mathrm{T}}, \boldsymbol{t} = \boldsymbol{t}_2 - \mathbf{R}_2\mathbf{R}_1^{\mathrm{T}})$$

Stereo calibration

- Refine the relative pose by minimizing the squared sum of the re-projection errors

$$\sum_i \|\boldsymbol{m}_1^i - f(\mathbf{R}_1, \boldsymbol{t}_1, \boldsymbol{X}^i)\|^2 + \|\boldsymbol{m}_2^i - f(\mathbf{R}_2, \boldsymbol{t}_2, \boldsymbol{X}^i)\|^2$$



$$\Theta = (\mathbf{R}_1, \boldsymbol{t}_1, \mathbf{R}_2, \boldsymbol{t}_2)$$

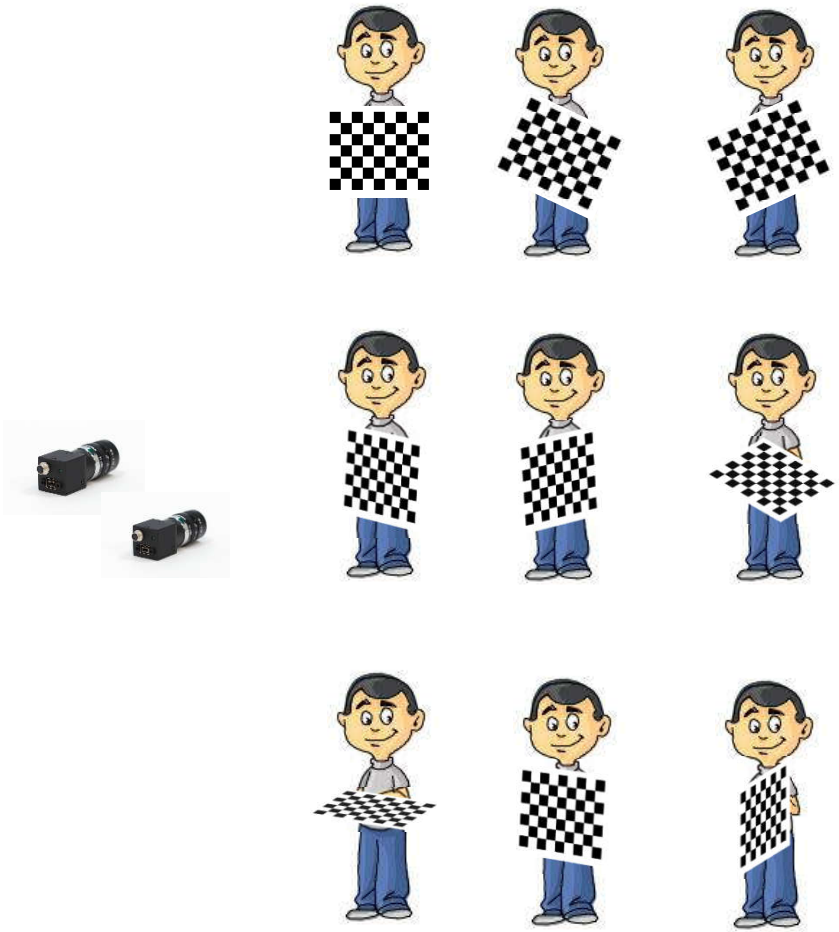
$$\Delta\Theta = (\Delta\theta_1, \Delta\boldsymbol{t}_1, \Delta\theta_2, \Delta\boldsymbol{t}_2)$$

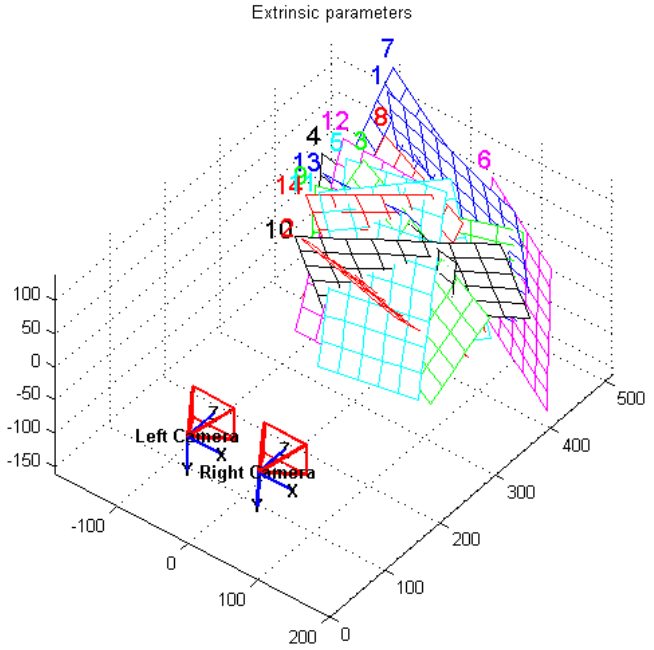
$$\Theta \leftarrow \Theta \boxplus \Delta\Theta$$

$$\begin{pmatrix} \mathbf{R}_1 \\ \boldsymbol{t}_1 \\ \mathbf{R}_2 \\ \boldsymbol{t}_2 \end{pmatrix} \leftarrow \begin{pmatrix} \mathbf{R}_1 \exp(\Delta\theta_1^\wedge) \\ \boldsymbol{t}_1 + \Delta\boldsymbol{t}_1 \\ \mathbf{R}_1 \exp(\Delta\theta_1^\wedge) \\ \boldsymbol{t}_1 + \Delta\boldsymbol{t}_1 \end{pmatrix}$$

Stereo calibration

- The process of calibration:





Stereo calibration

- OpenCV

```
double cv::stereoCalibrate(InputArrayOfArrays  objectPoints,
                           InputArrayOfArrays  imagePoints1,
                           InputArrayOfArrays  imagePoints2,
                           InputOutputArray    cameraMatrix1,
                           InputOutputArray    distCoeffs1,
                           InputOutputArray    cameraMatrix2,
                           InputOutputArray    distCoeffs2,
                           Size imageSize,
                           OutputArray  R,
                           OutputArray  T,
                           OutputArray  E,
                           OutputArray F,
                           int flags = CALIB_FIX_INTRINSIC,
                           TermCriteria criteria =
TermCriteria(TermCriteria::COUNT+TermCriteria::EPS, 30, 1e-6))
```

Stereo rectification

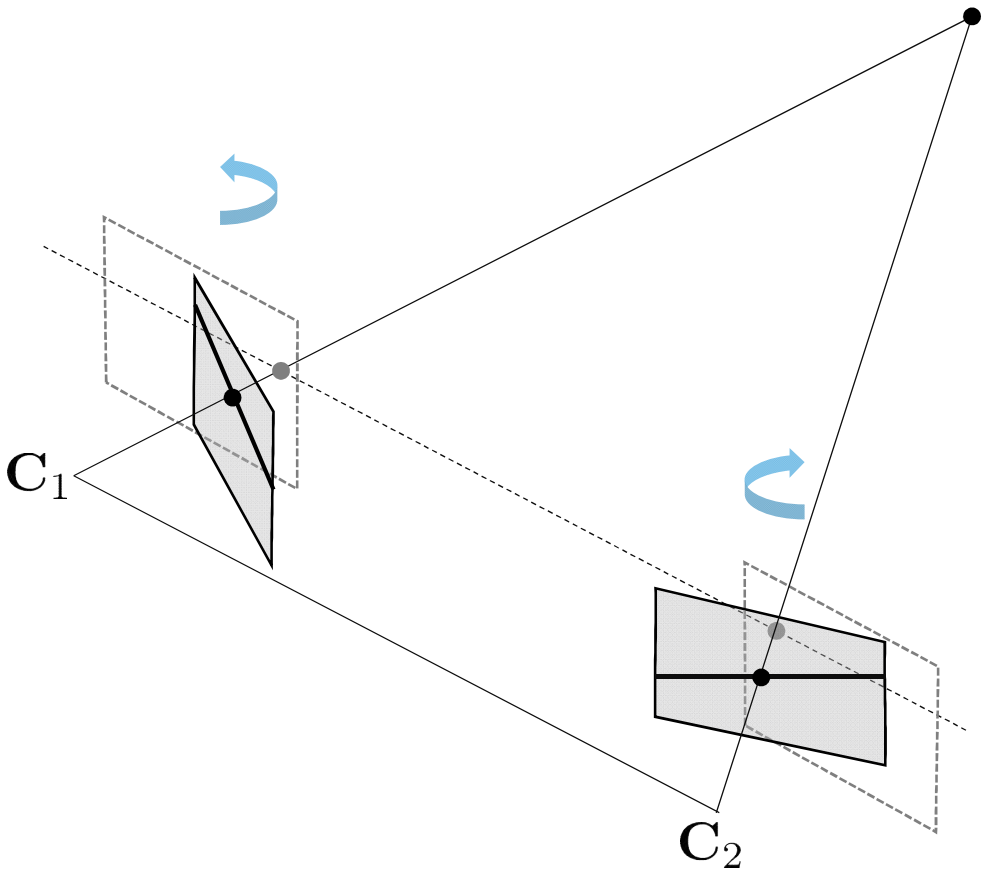
- Rotate the cameras to become an ideal setting:

$$\mathbf{T}_1 = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix} \quad \mathbf{T}_2 = \begin{bmatrix} \mathbf{R} & \boldsymbol{t} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix}$$

$$\mathbf{T}_1' = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix} \quad \mathbf{T}_2' = \begin{bmatrix} \mathbf{I}' & \boldsymbol{t}' \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix}$$

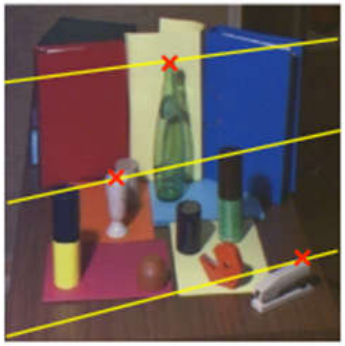
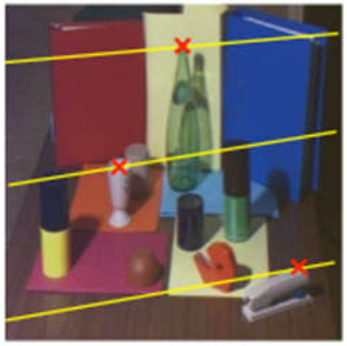
$$(\boldsymbol{t}' = [-b, 0, 0]^{\mathrm{T}})$$

$$\mathbf{K}_1, \mathbf{K}_2 \Rightarrow \mathbf{K}$$

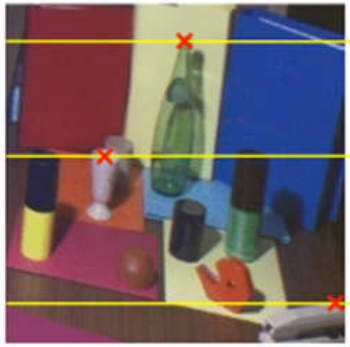
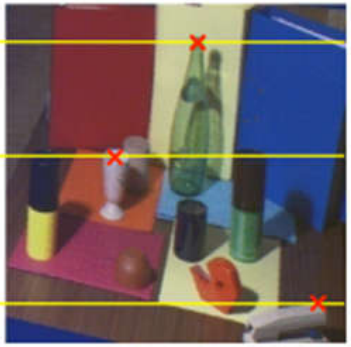


Stereo rectification

- Benefits of stereo rectification
  - To change the 2D searching problem into a 1D searching problem (more reliable and faster)
  - Depth directly related to horizontal disparities

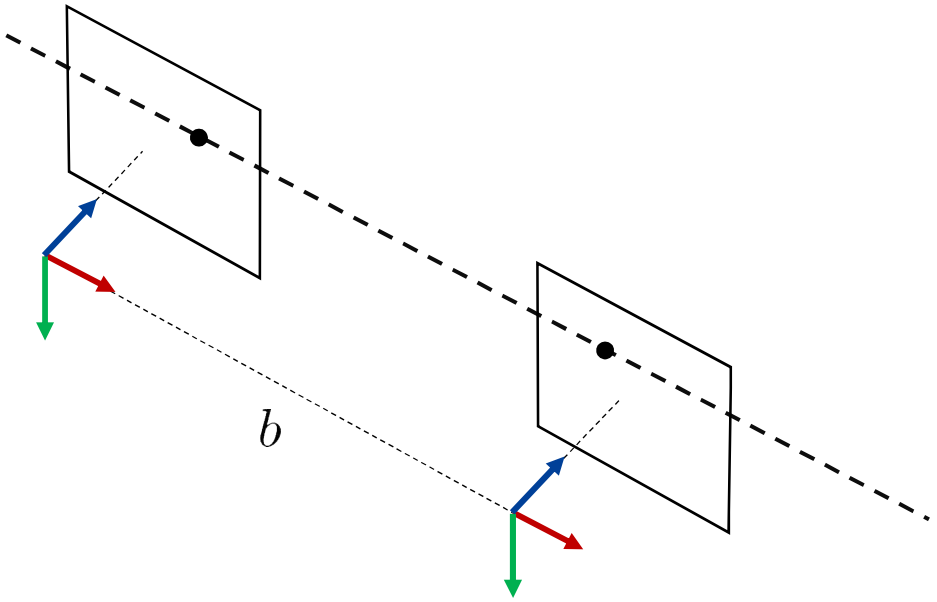
Before rectification

After rectification

Stereo rectification

- Ideal setting :
  - For both camera frames, the *X* axis is aligned with the base line.

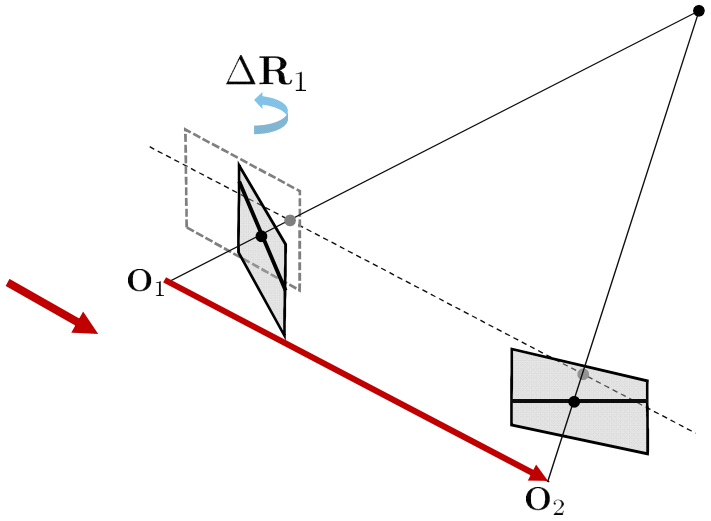


Stereo rectification

- Hence all we need to do is seek a rotation, that transforms **the unit baseline vector** into the **a unit vector in x direction** in the new coordinate system
- For example, considering the left camera,

$$\Delta \mathbf{R}_1 \boldsymbol{o}_{12} = \mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$(\boldsymbol{o}_{12} = \overrightarrow{\mathbf{O}_1\mathbf{O}_2} / \|\overrightarrow{\mathbf{O}_1\mathbf{O}_2}\|)$$



Stereo rectification

Computation of the baseline vector

- The poses of the two cameras are

$$\mathbf{T}_1 = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix} \quad \mathbf{T}_2 = \begin{bmatrix} \mathbf{R} & t \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix}$$

- The baseline vector is computed as

$$\overrightarrow{\mathbf{O}_1\mathbf{O}_2} = -\mathbf{R}^{\mathrm{T}}t \quad (\mathbf{R}\overrightarrow{\mathbf{O}_1\mathbf{O}_2} + t = \mathbf{0})$$

$$\Rightarrow \boldsymbol{o}_{12} = \overrightarrow{\mathbf{O}_1\mathbf{O}_2} / \|\overrightarrow{\mathbf{O}_1\mathbf{O}_2}\|$$

Stereo rectification

- Extra conditions are required to get the rotation
  - We want to get the other two axes of the new coordinate system.

$$\Delta \mathbf{R}_1 \boldsymbol{o}_{12} = \boldsymbol{e}_x = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\Delta \mathbf{R}_1 = \begin{bmatrix} \boldsymbol{o}_{12}^T \\ unknown^T \\ (\boldsymbol{o}_{12} \times unknown)^T \end{bmatrix}$$

- The $unknown$ is a direction in the original frame related to the y axis of the new frame.

Stereo rectification

- How do we select the $unknown$ vector in the original camera frame ?
- Note that firstly, it should be perpendicular the baseline vector

$$unknown \perp \overrightarrow{\mathbf{O}_1\mathbf{O}_2}$$

- We can choose to let

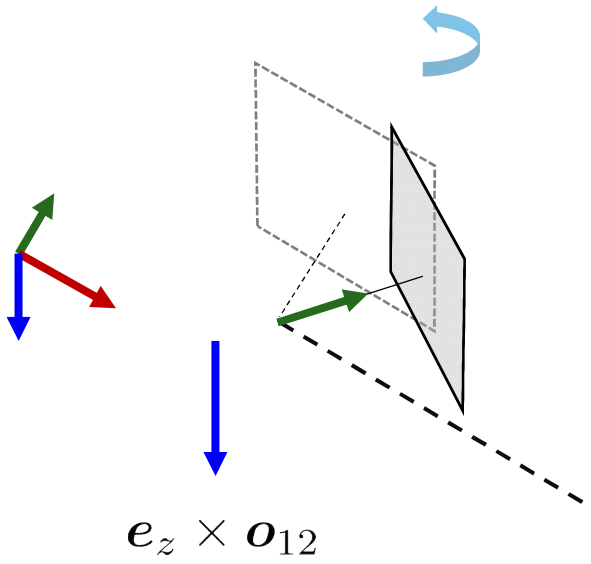
$$unknown \sim \boldsymbol{e}_z \times \overrightarrow{\mathbf{O}_1\mathbf{O}_2}$$

$$\boldsymbol{d} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \boldsymbol{o}_{12}$$

Z axis of the original camera frame – optical axis of 1st camera

Stereo rectification

- After the $unknown$ vector is determined, we get the rotation for the left camera.



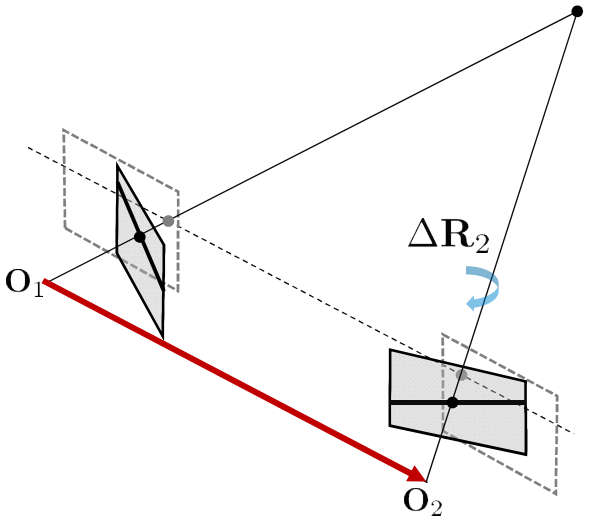
$$\Delta \mathbf{R}_1 = \begin{bmatrix} \boldsymbol{o}_{12}^T \\ unknown^T \\ (\boldsymbol{o}_{12} \times unknown)^T \end{bmatrix}$$

$$unknown = \boldsymbol{d} = \boldsymbol{e}_z \times \boldsymbol{o}_{12}$$

$$\Delta \mathbf{R}_1 = \begin{bmatrix} \boldsymbol{o}_{12}^T \\ \boldsymbol{d}^T \\ (\boldsymbol{o}_{12} \times \boldsymbol{d})^T \end{bmatrix}$$

Stereo rectification

- For the right camera, the rotation is given



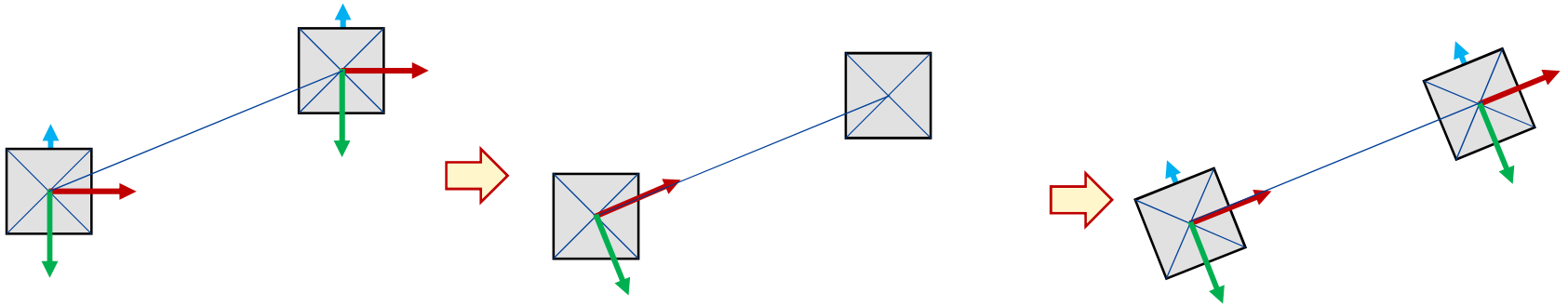
$$\Delta\mathbf{R}_2 = \begin{bmatrix} \boldsymbol{o}'_{12}{}^{\mathrm{T}} \\ \boldsymbol{d}'^{\mathrm{T}} \\ (\boldsymbol{o}'_{12} \times \boldsymbol{d}')^{\mathrm{T}} \end{bmatrix}$$

$\boldsymbol{o}'_{12}, \boldsymbol{d}'$ are the directions of $\boldsymbol{o}_{12}, \boldsymbol{d}$ expressed in the second camera frame.

$$\boldsymbol{o}'_{12} = \mathbf{R}\boldsymbol{o}_{12}$$
$$\boldsymbol{d}' = \mathbf{R}\boldsymbol{d}$$

Stereo rectification

- Illustration of finding rotations for stereo rectification

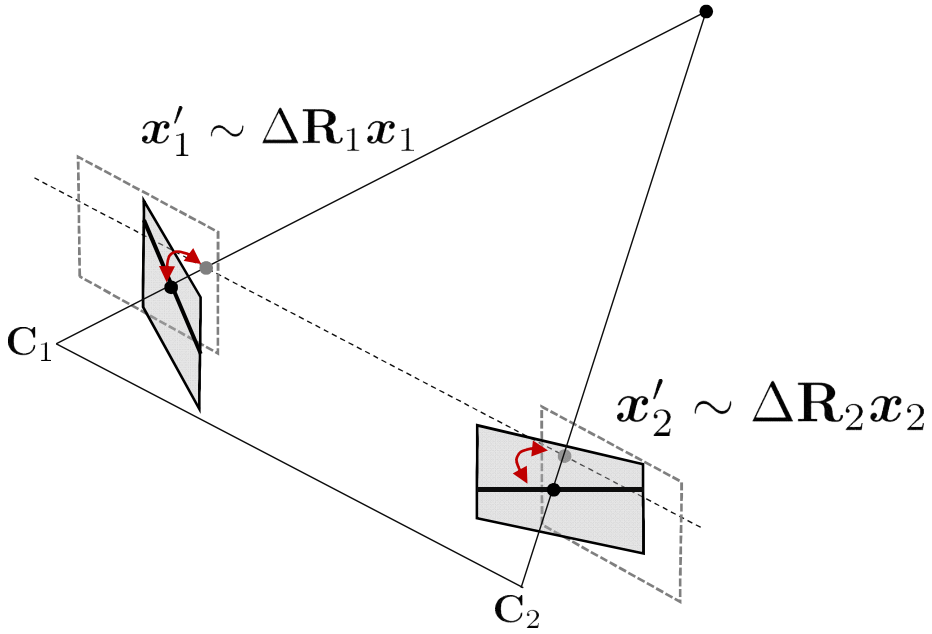


1. Baseline direction
2. Y direction

Apply rotations to both cameras

Stereo rectification

- Warping – Map the image into the rectified image plane (use Homography transformation)



Stereo rectification

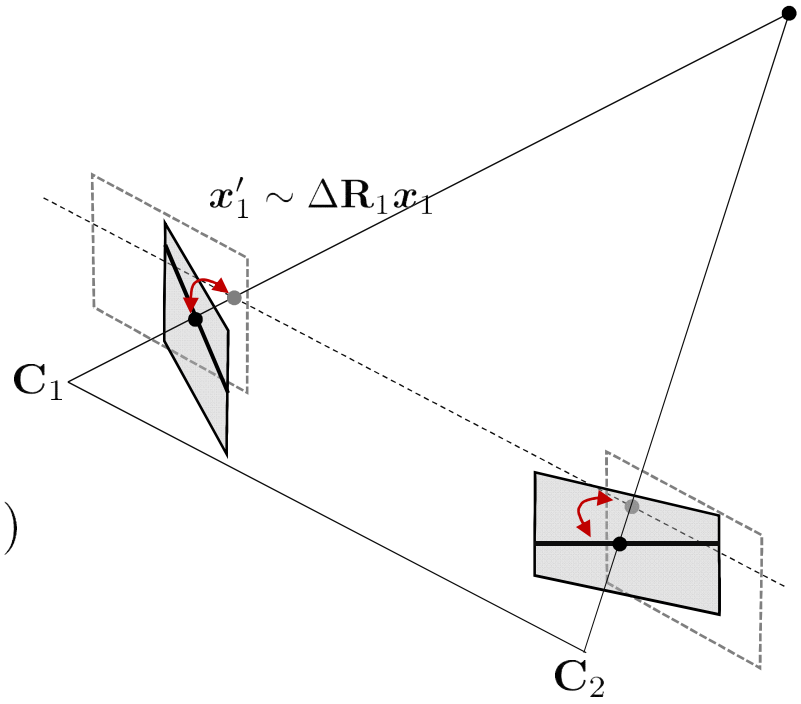
- The transformation of the new image point m_1' from the old image point m_1.

$$m_1' \sim \mathbf{K}x_1'$$

$$= \mathbf{K}\Delta\mathbf{R}_1 x_1$$

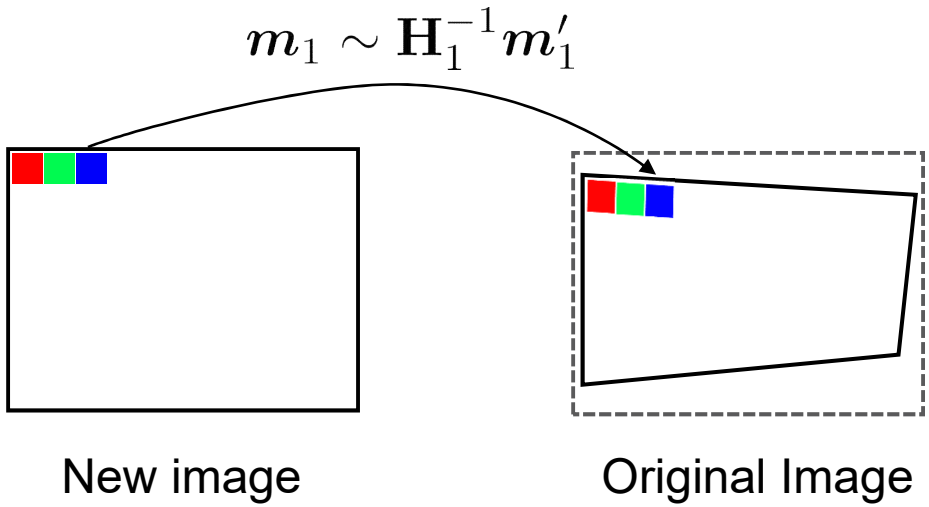
$$= \mathbf{K}\Delta\mathbf{R}_1\mathbf{K}_1^{-1}m_1$$

$$m_1' \sim \mathbf{H}_1 m_1 \quad (\mathbf{H}_1 = \mathbf{K}\Delta\mathbf{R}_1\mathbf{K}_1^{-1})$$



Stereo rectification

- Warping



Stereo rectification

- Scaling & Cropping - Select a good intrinsic matrix

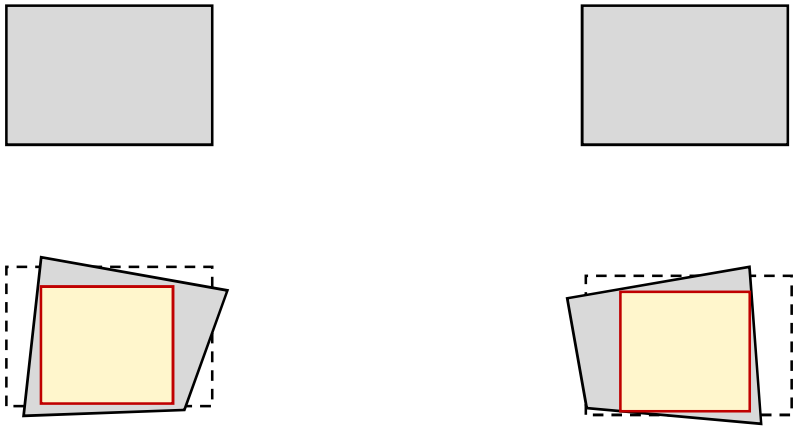
$$\mathbf{K}_1 = \begin{bmatrix} f_{x_1} & 0 & C_{x_1} \\ 0 & f_{y_1} & C_{y_1} \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{K}_1 = \begin{bmatrix} f_{x_2} & 0 & C_{x_2} \\ 0 & f_{y_2} & C_{y_2} \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$f_x = 0.5(f_{x_1} + f_{x_2})$$
$$f_y = 0.5(f_{y_1} + f_{y_2})$$

Stereo rectification

- Cropping
  - Decide the size of the image
  - Decide the optical center in the image C_x, C_y



Stereo rectification

- We can also use different intrinsic matrices with different C_x

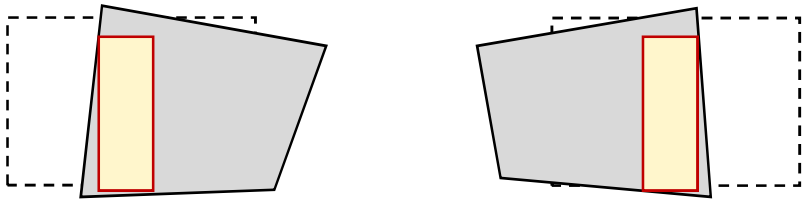
$$\mathbf{K}_1 = \begin{bmatrix} f_{x_1} & 0 & C_{x_1} \\ 0 & f_{y_1} & C_{y_1} \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{K}_1 = \begin{bmatrix} f_{x_2} & 0 & C_{x_2} \\ 0 & f_{y_2} & C_{y_2} \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{K}'_1 = \begin{bmatrix} f_x & 0 & C'_{x_1} \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{K}'_1 = \begin{bmatrix} f_x & 0 & C'_{x_2} \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{bmatrix}$$

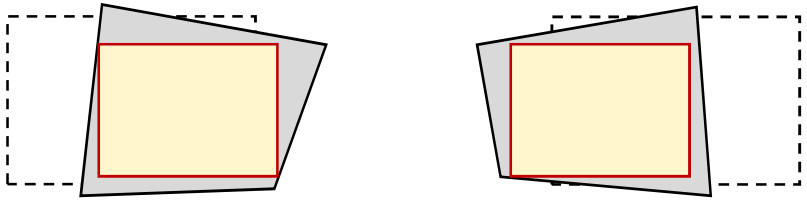
Stereo rectification

- Select different C_x

Same $\mathbf{K}$

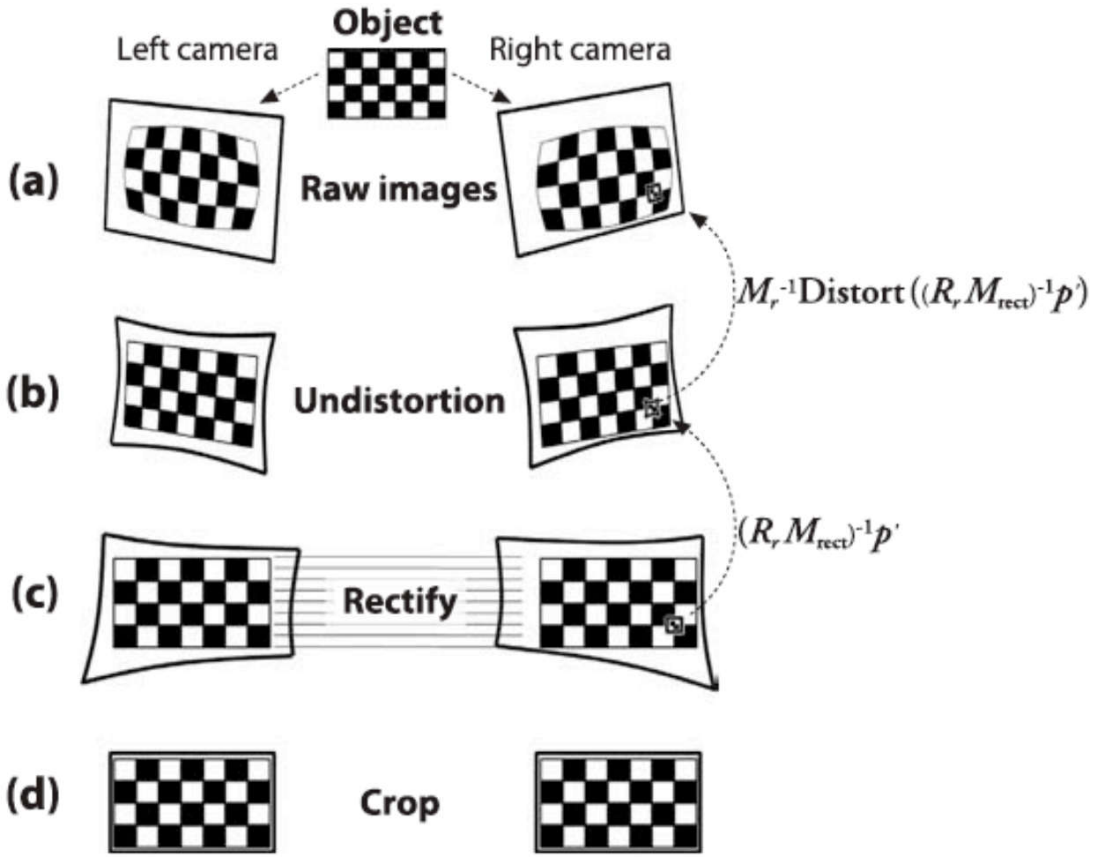


Different C_x



Stereo rectification

- The whole process of stereo rectification:



Summary

- Stereo calibration:
  - The relative pose estimated from unknown objects has scale ambiguity
  - Use the checkerboard pattern
  - Compute the pose for each view
  - Refined by minimizing the re-projection error
- Stereo rectification:
  - Virtually rotate the camera into the parallel setting
    - **Step1** – rotate the first camera
    - **Step2** – rotate the second camera
    - **Step3** – Warping & Cropping

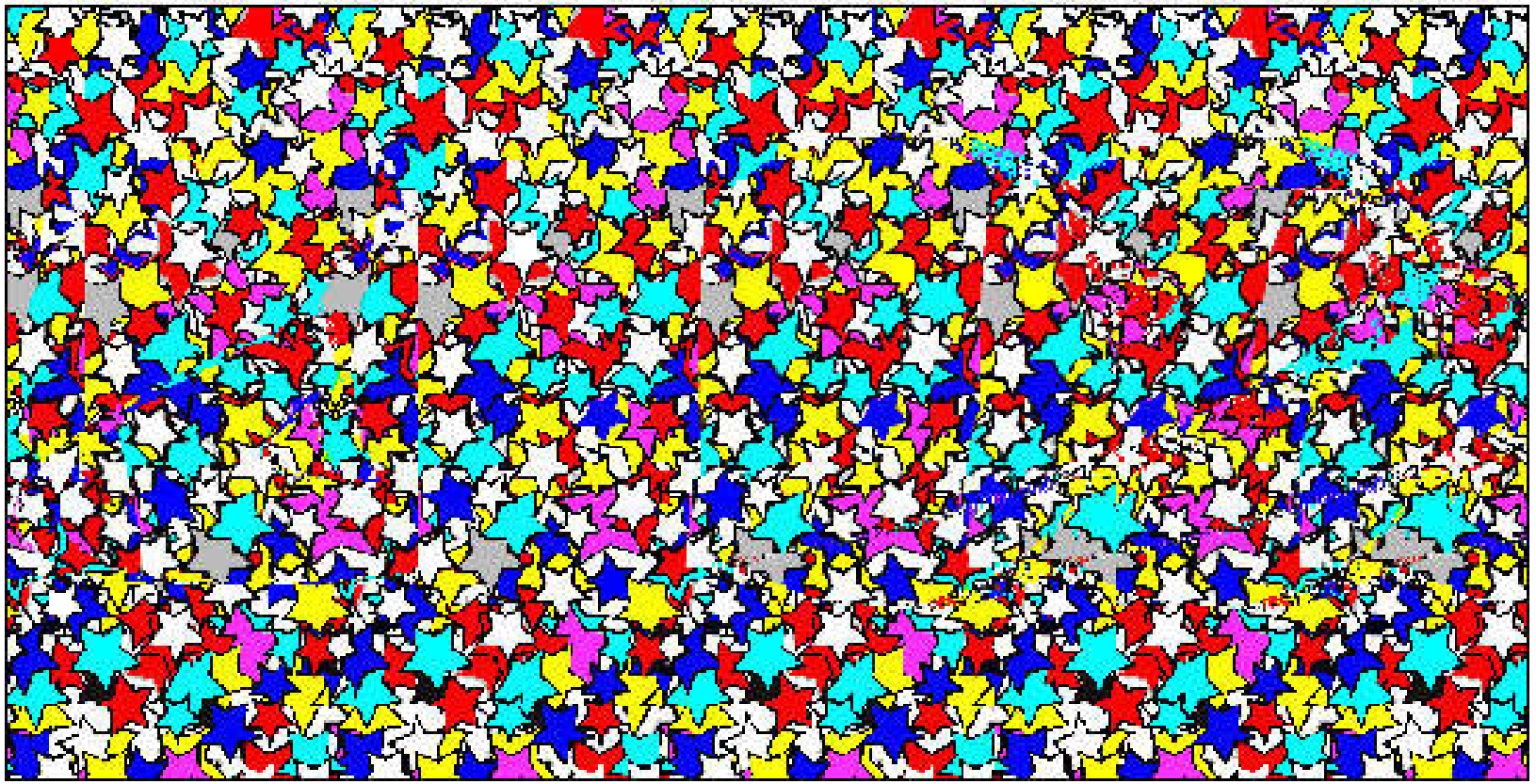
Stereo matching

- We can get a dense depth map from stereo images if corresponding points between two views are established for every pixel.

Stereo images after rectification

Dense depth map

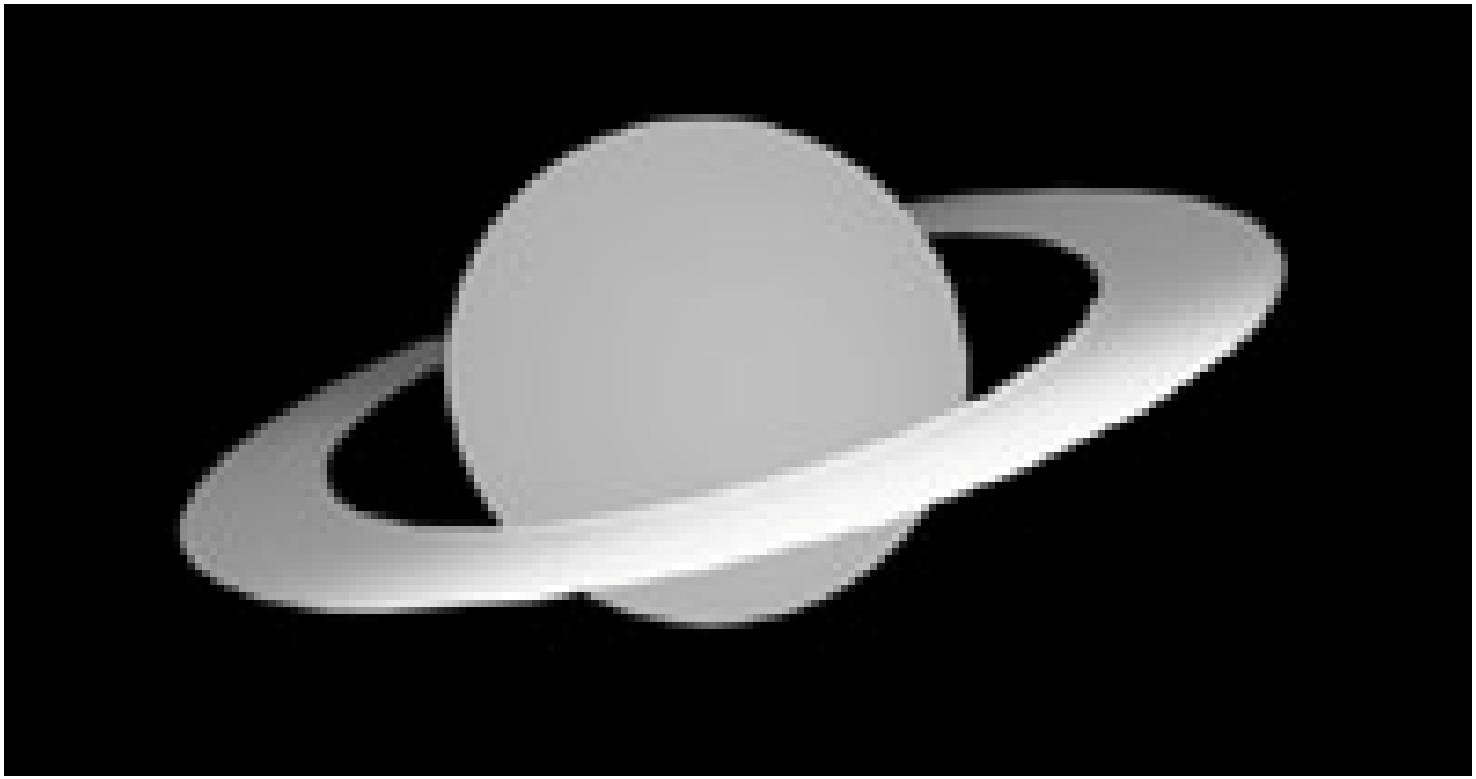
Stereo matching



Stereo matching

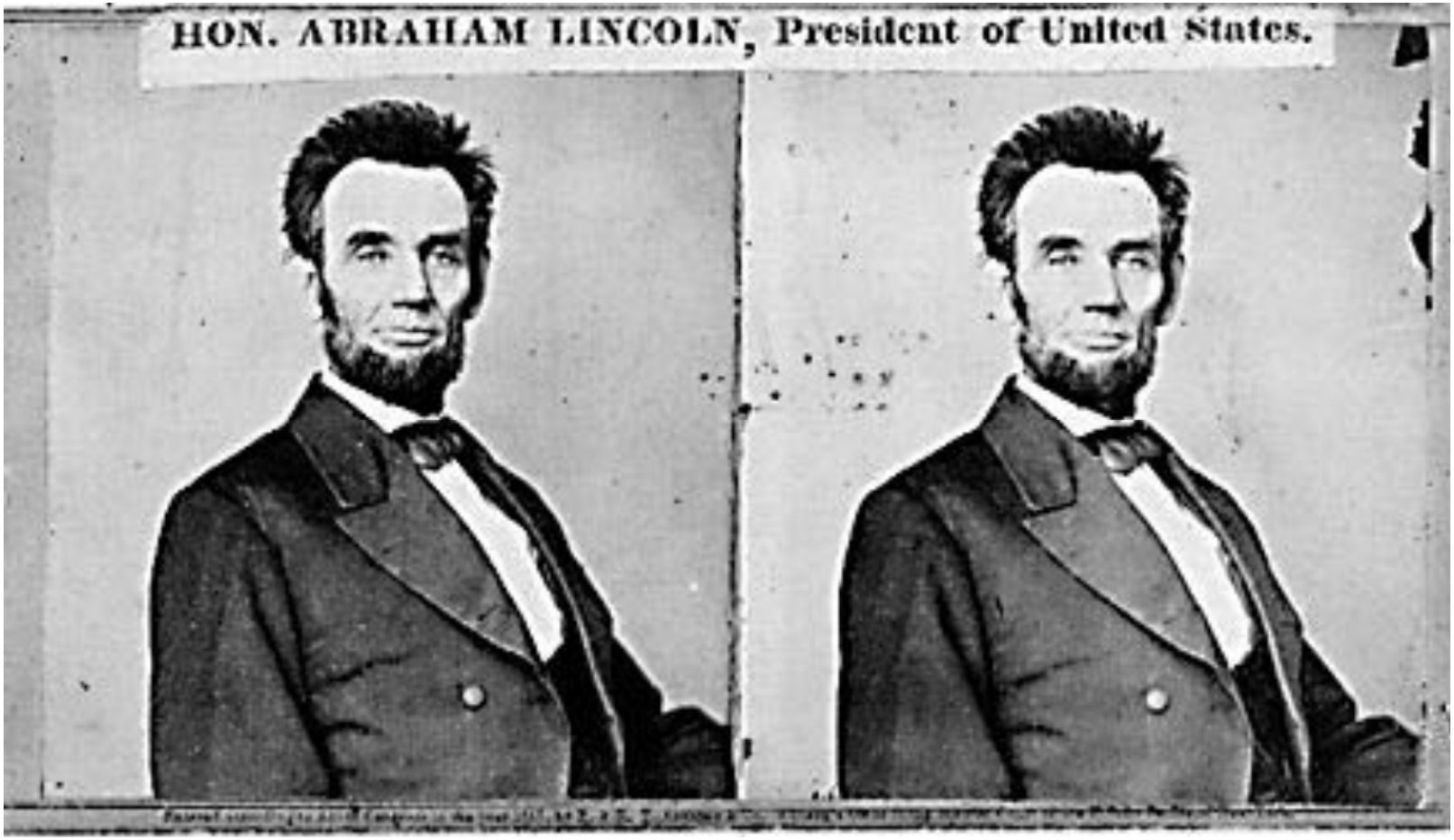
- Human can do stereo matching easily.

  The depth you can perceive from previous pictures:



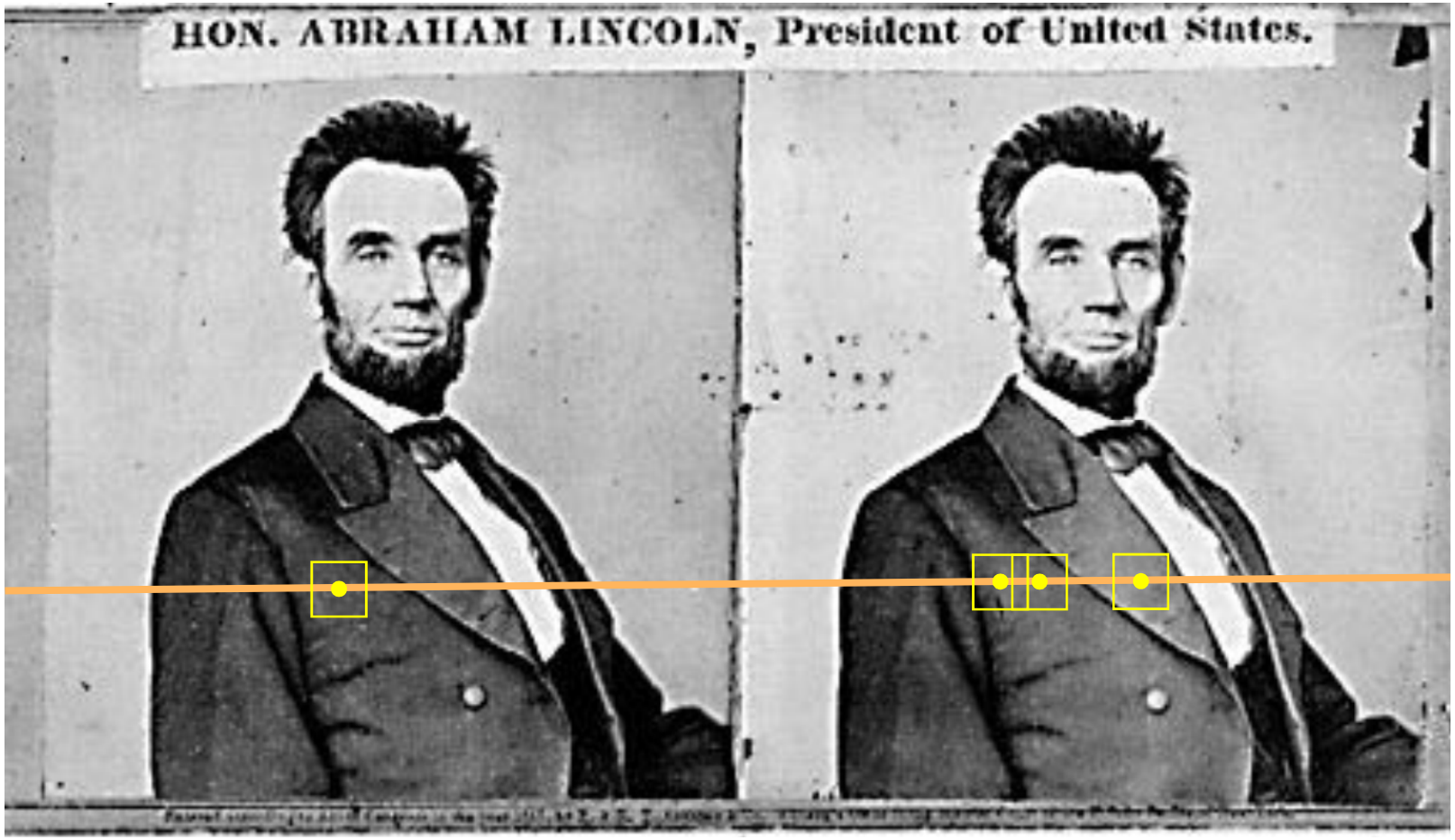
Stereo matching

- Human can do stereo matching easily.



Stereo matching

- However, it is difficult for a computer.



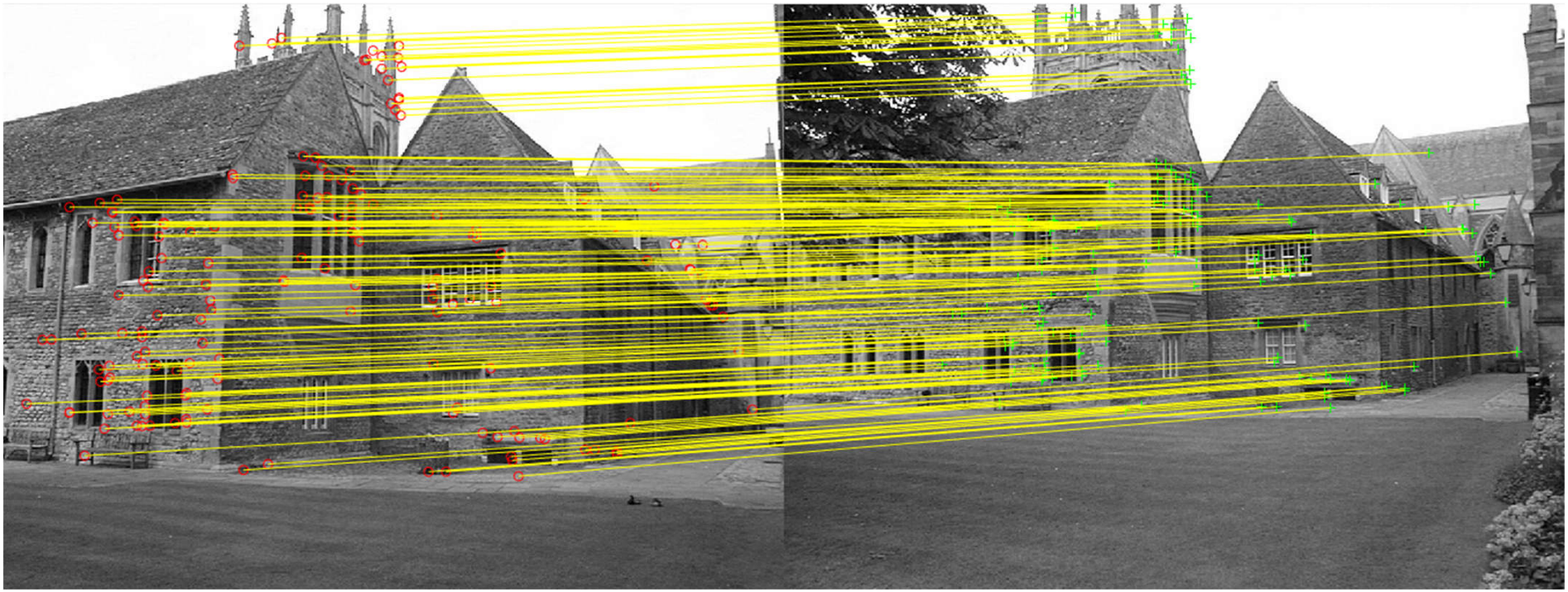
Stereo matching

Stereo matching has been a long-standing research topic in computer vision (1973 ~ Now) :

- Matching by Winner-take-all
- Matching costs :
  - Patch-based:
    - **SAD, SSD, ZNCC**
- Failure cases:
- Non-local approach
  - **Global optimization** (Graph cut, Belief Propagation)
  - **Semi Global Matching** (**SGM**)

Stereo match

- Sparse matching



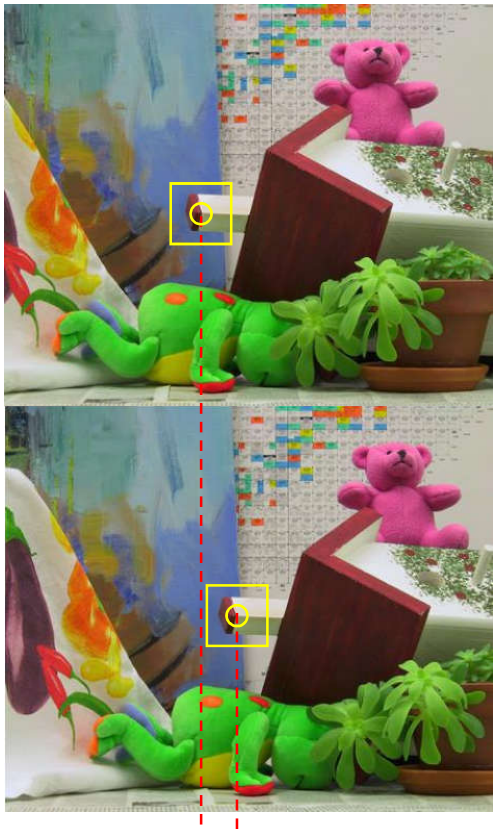
Stereo matching

- Dense matching
  - Find **dense pixel correspondence** in rectified images
- Disparity map: $D(x, y)$

$$x' = x + D(x, y), y' = y$$

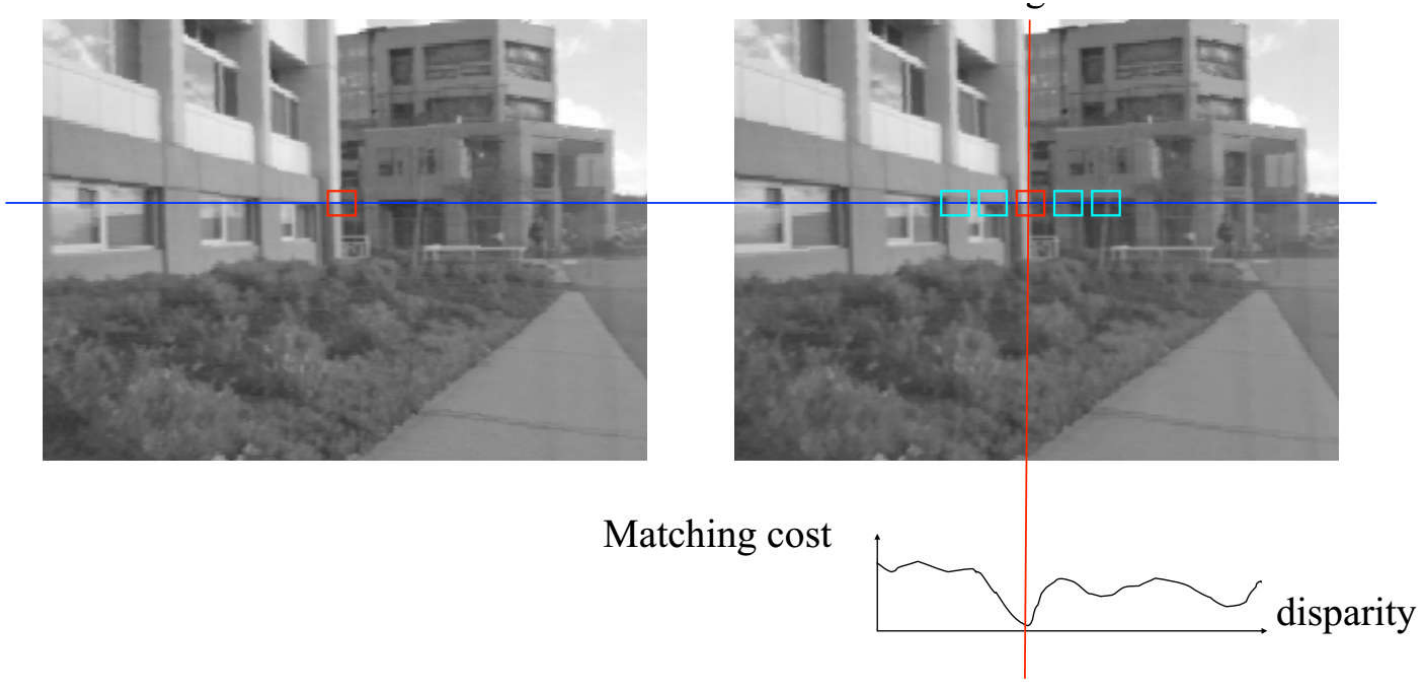
- Depth map : $Z(x, y)$

$$Z(x, y) \propto \frac{1}{D}(x, y)$$



Winner-take-all matching

- The disparity is determined independently for each pixel.
- **Winner-take-all** selects the disparity that leads to the best matching cost.



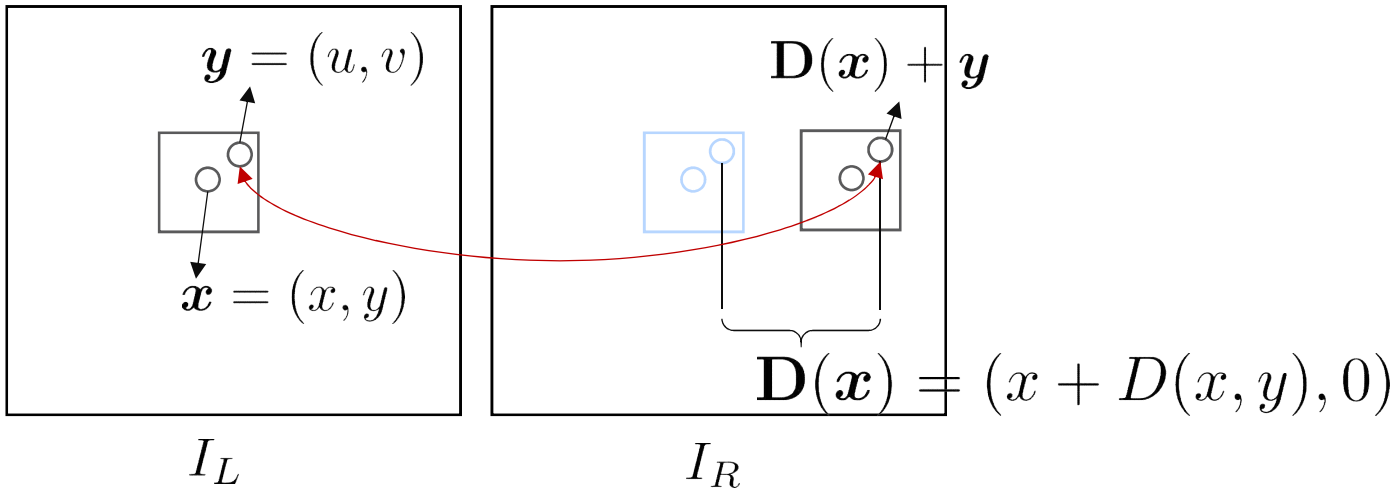
Matching costs

- Sum of Absolute Difference (**SAD**)

$$C_{SAD}(\boldsymbol{x}, D) = \sum_{\boldsymbol{y}\in[-w,w]\times[-w,w]} |I_L(\boldsymbol{x}+\boldsymbol{y}) - I_R(\mathbf{D}(\boldsymbol{x})+\boldsymbol{y})|$$

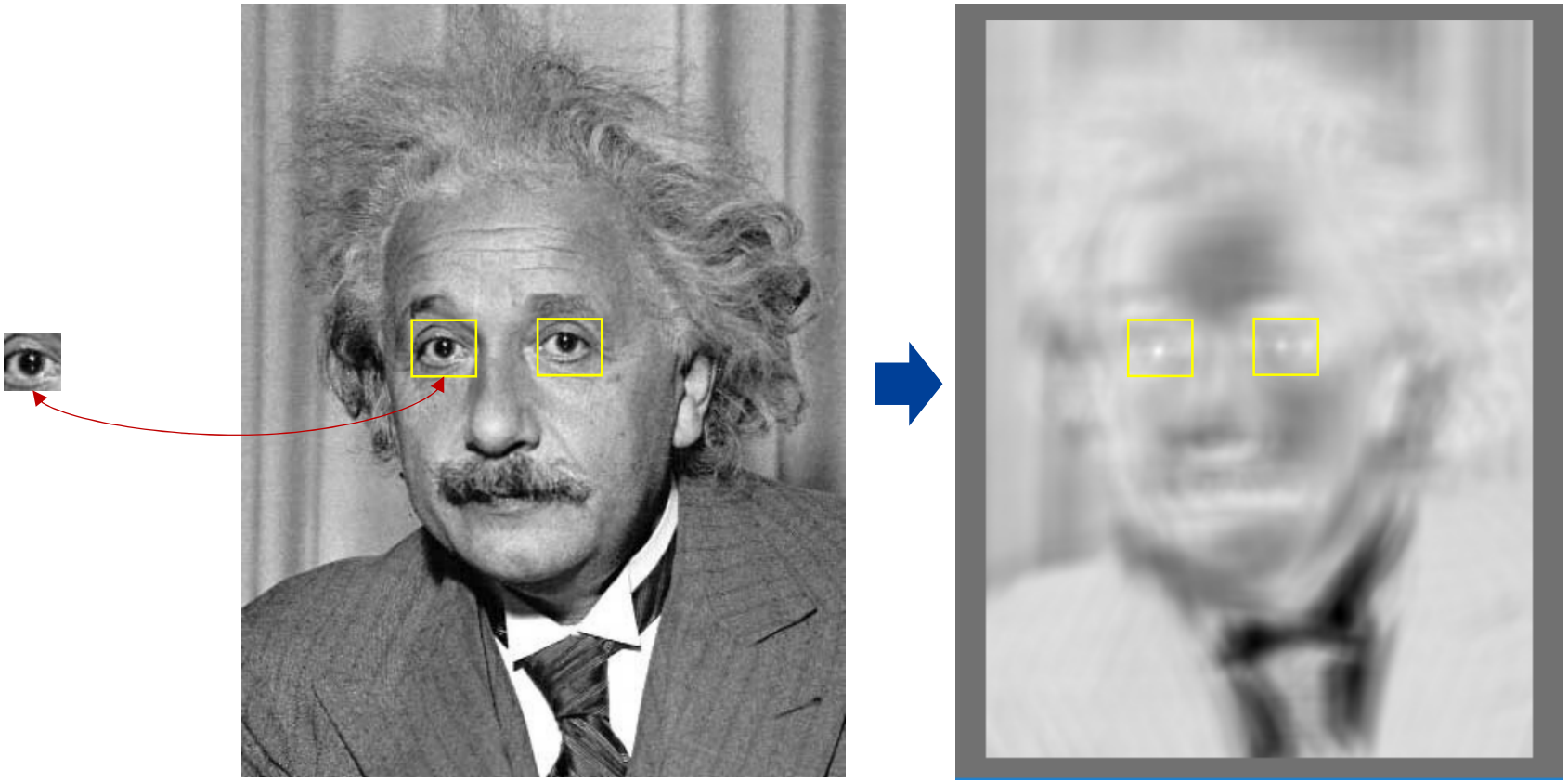
- Sum of Squared Difference (**SSD**)

$$C_{SAD}(\boldsymbol{x}, D) = \sum_{\boldsymbol{y}\in[-w,w]\times[-w,w]} |I_L(\boldsymbol{x}+\boldsymbol{y}) - I_R(\mathbf{D}(\boldsymbol{x})+\boldsymbol{y})|^2$$



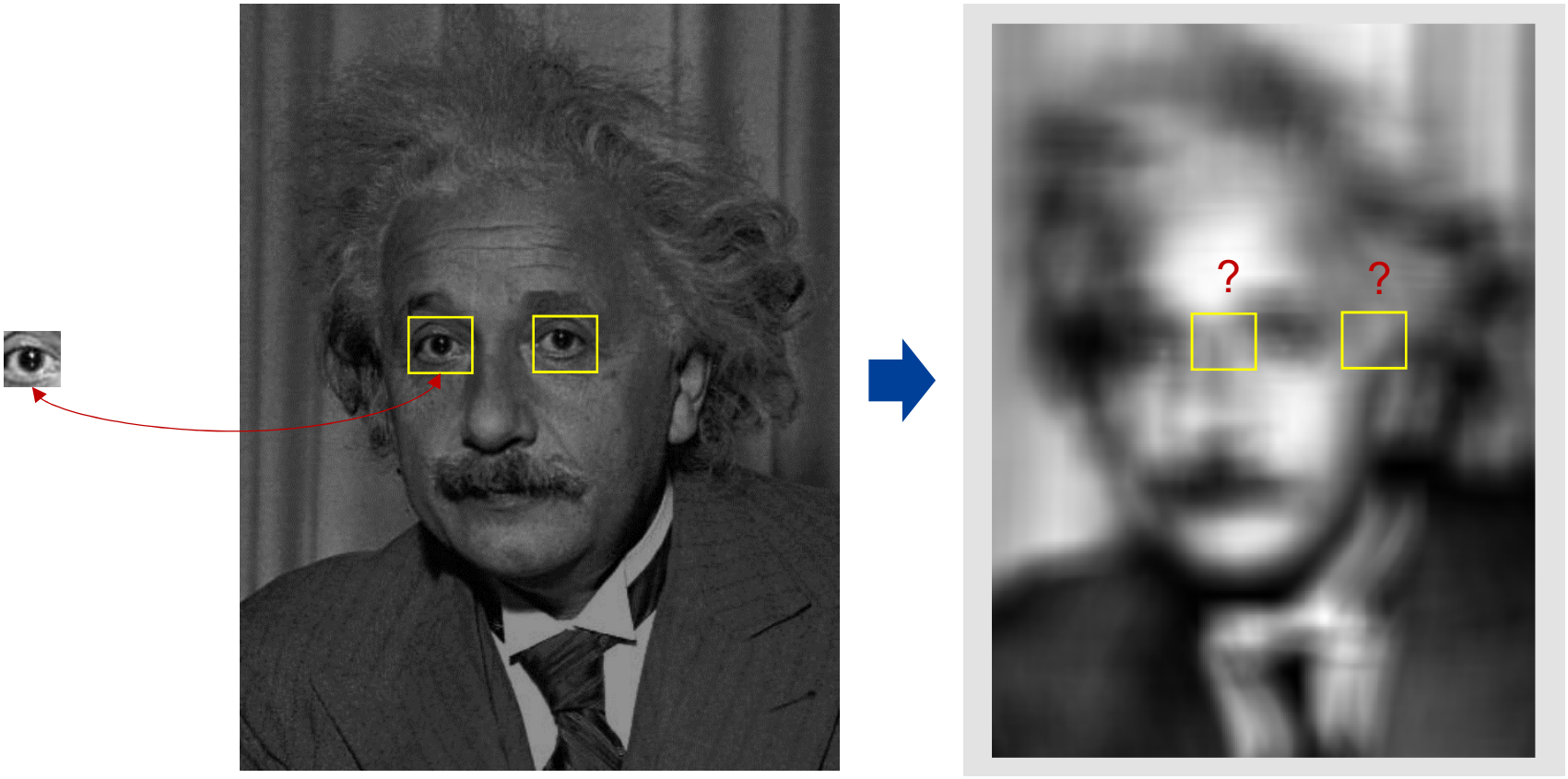
Matching costs

- An example of SAD/SSD searching : Find the eyes in the picture.



Matching costs

- But if the illumination changes, the eye cannot be correctly found.



Matching costs

- Zero Mean Normalized Cross Correlation (ZNCC)

$$C_{ZNCC}(\boldsymbol{x}, D) = \frac{\tilde{P}_L(\boldsymbol{x}) \cdot \tilde{P}_R(\boldsymbol{x}+\mathbf{D})}{|\tilde{P}_L(\boldsymbol{x})| \cdot |\tilde{P}_R(\boldsymbol{x}+\mathbf{D})|}$$

$$\tilde{P}(\boldsymbol{x}) = P(\boldsymbol{x}) - \bar{P}(\boldsymbol{x})$$

$P(\boldsymbol{x})$ $\bar{P}(\mathbf{x})$ $\tilde{P}(\mathbf{x})$

Matching costs

- The dot product of two normalized image patches is computed as

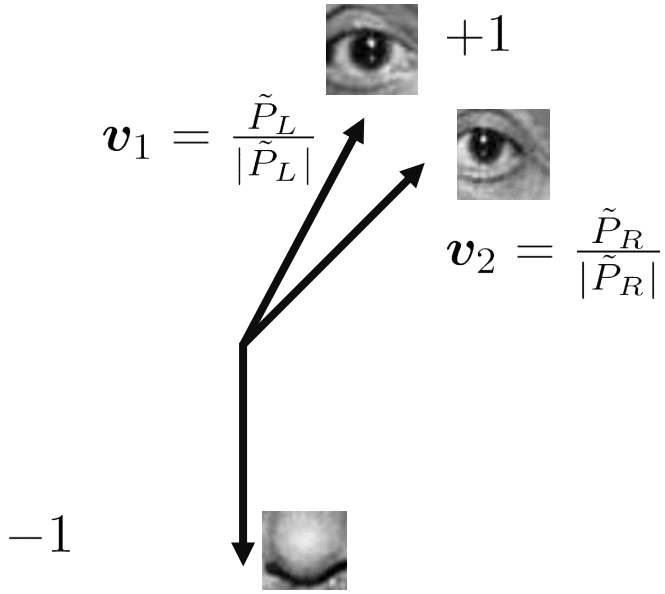
$$\tilde{P}_L(\boldsymbol{x}) \cdot \tilde{P}_R(\boldsymbol{x}+\mathbf{D})$$

$$= \sum_{\boldsymbol{y}\in[-w,w]\times[-w,w]} (\tilde{I}_L(\boldsymbol{x}+\boldsymbol{y})\tilde{I}_R(\mathbf{D}(\boldsymbol{x})+\boldsymbol{y}))$$

$$|P(\boldsymbol{x})| = \sqrt{\sum_{\boldsymbol{y}\in[-w,w]\times[-w,w]} \tilde{I}(\boldsymbol{x}+\boldsymbol{y})^2}$$

Matching costs

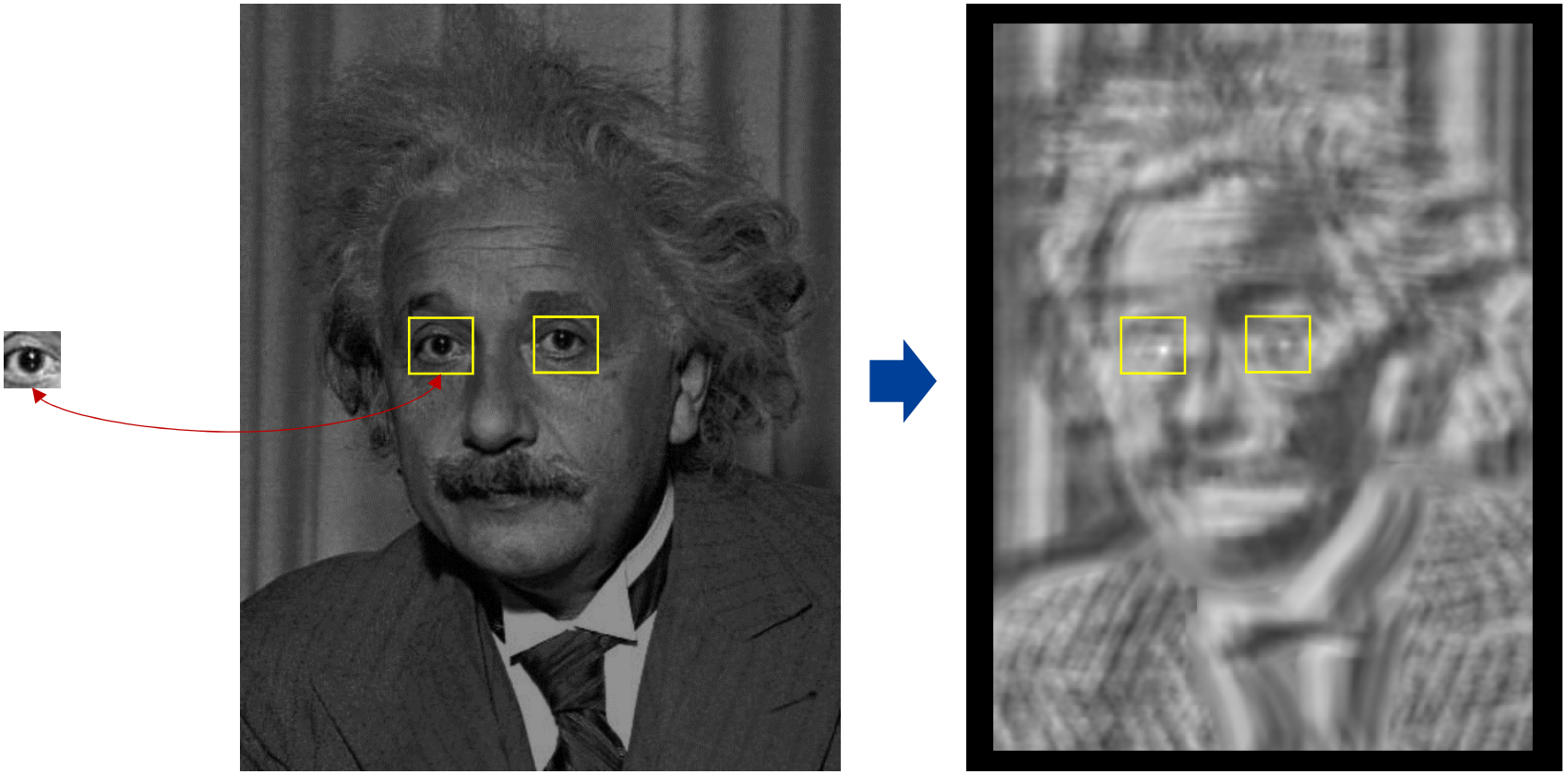
- Zero Mean Normalized Cross Correlation (ZNCC)
  - ZNCC describes the **inner product** between two unit vectors (two patches with normalized intensities)



$$C_{ZNCC} \in [-1, +1]$$

Matching costs

- ZNCC works properly when the illumination changes.



Matching - local approach

- Failure cases – 1. without enough textures

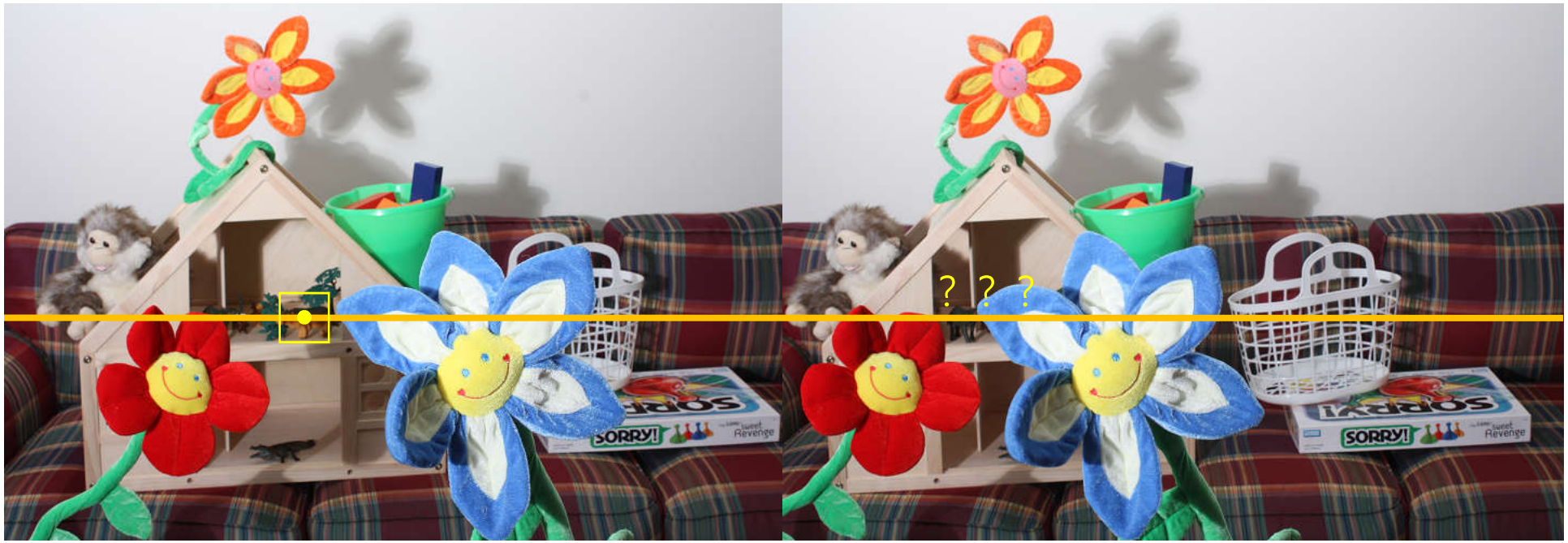
Matching - local approach

- Failure cases – 2. repetitive/similar textures

Matching - local approach

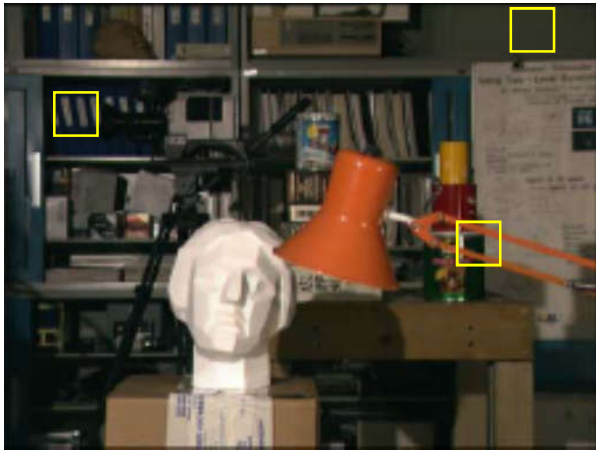
Matching - local approach

- Failure cases – 4. occlusion



Matching - local approach

- Using **SAD+Winner-Take-All** method the result looks like this :



Left image

Ground truth

Actual result (**Many errors**)

Global methods

- The disparity map is solved as a whole by minimizing an energy function :

$$E(D) = E_D(D) + \lambda E_s(D)$$

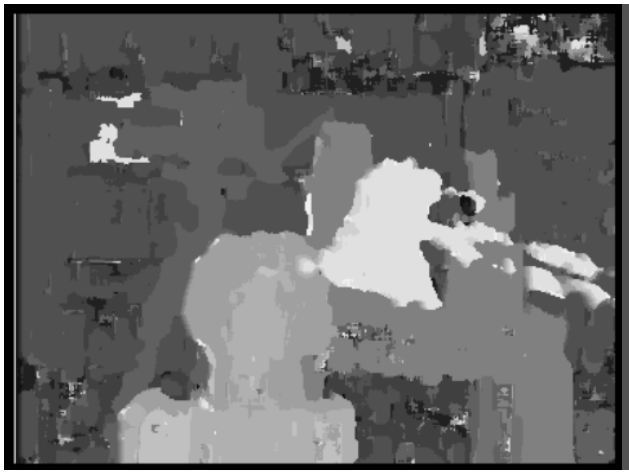
$E_D(D)$ **Data term** – represents the matching costs

$E_s(D)$ **Smoothness term** – represents the local continuity and smoothness of the disparity map

λ **Weight** – control the strength of the smoothness term

Global methods

- There are two major methods to solve (approximately) the energy minimization problem (NP hard).
  - **Belief Propagation**
  - **Graph Cuts**



Winner-take-all

Belief Propagation
Felzenszwalb & Huttenlocher, 2004

Graph Cuts
Kolmogorov & Zabih, 2001

Summary

Stereo matching has been a long-standing research topic in computer vision (1973 ~ Now) :

- Matching costs :
  - Patch-based:
    - **SAD, SSD, ZNCC**
- Local approach
  - **Winner-take-all**
- Non-local approach
  - **Global optimization** (Graph cut, Belief Propagation)
  - **Semi Global Matching** (**SGM**)