Today

Early vision on multiple images

- **Estimating depth**
- **3D reconstruction**

Announcement

Quiz #6 is available in Blackboard.

Due date: 11:59pm EST, Thursday, April 3

Open book and open notes

No late submission is allowed

Recall: Fundamental Matrix

- Fundamental matrix is defined on the pixel coordinate (row-column image coordinate)
- Given fundamental matrix, the epipolar geometry can be reconstructed without camera calibration!
  - Estimate conjugate epipolar lines

$$\mathbf{l}_{el}=\mathbf{F}\boldsymbol{U}_{\boldsymbol{r}} \qquad \mathbf{l}_{er}=\mathbf{F}^{\boldsymbol{t}}\boldsymbol{U}_{\boldsymbol{l}}$$

  - Estimate epipoles - the epipole of right (left) image is the null eigenvector of **F** ($\mathbf{F}^{t}$)

$$\mathbf{F}\boldsymbol{e}_{\boldsymbol{r}} = 0 \qquad \mathbf{F}^{\boldsymbol{t}}\boldsymbol{e}_{\boldsymbol{l}} = 0$$

- **F** has a rank 2 and is determined by 7 independent parameters up to a scaling factor
- 8-point algorithm to estimate **F**: Linear solution with singularity constraint enforced

Compute E from F

$$\mathbf{F} = \mathbf{W}_l^{-\mathbf{t}} \mathbf{E} \mathbf{W}_r^{-1}$$

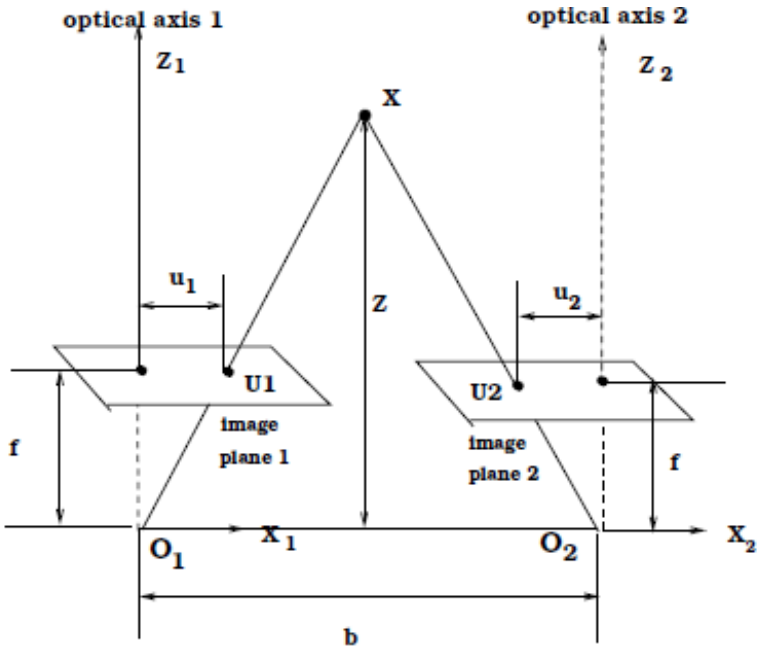
Given intrinsic parameters, E can be computed from F as

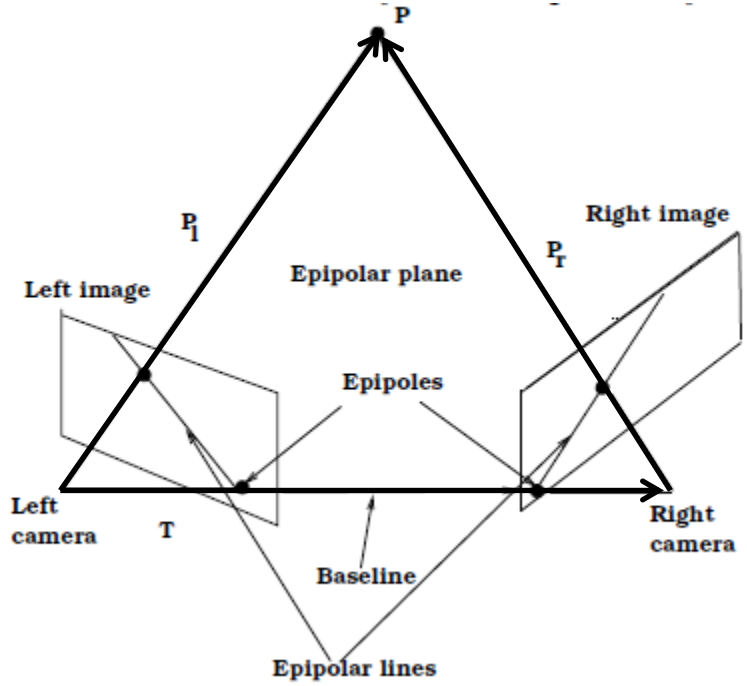
$$\mathbf{E} = \mathbf{W}_l^{\mathbf{t}} \mathbf{F} \mathbf{W}_r$$

Recover 3D Information – Estimate Depth

Depth can be recovered given the disparity in the rectified geometry:

$$Z = \frac{fb}{u_1 - u_2}$$





Recover 3D Information – Estimate Depth

Depth can be recovered given the disparity in the rectified geometry:

$$Z = \frac{fb}{u_1 - u_2}$$

How to rectify images

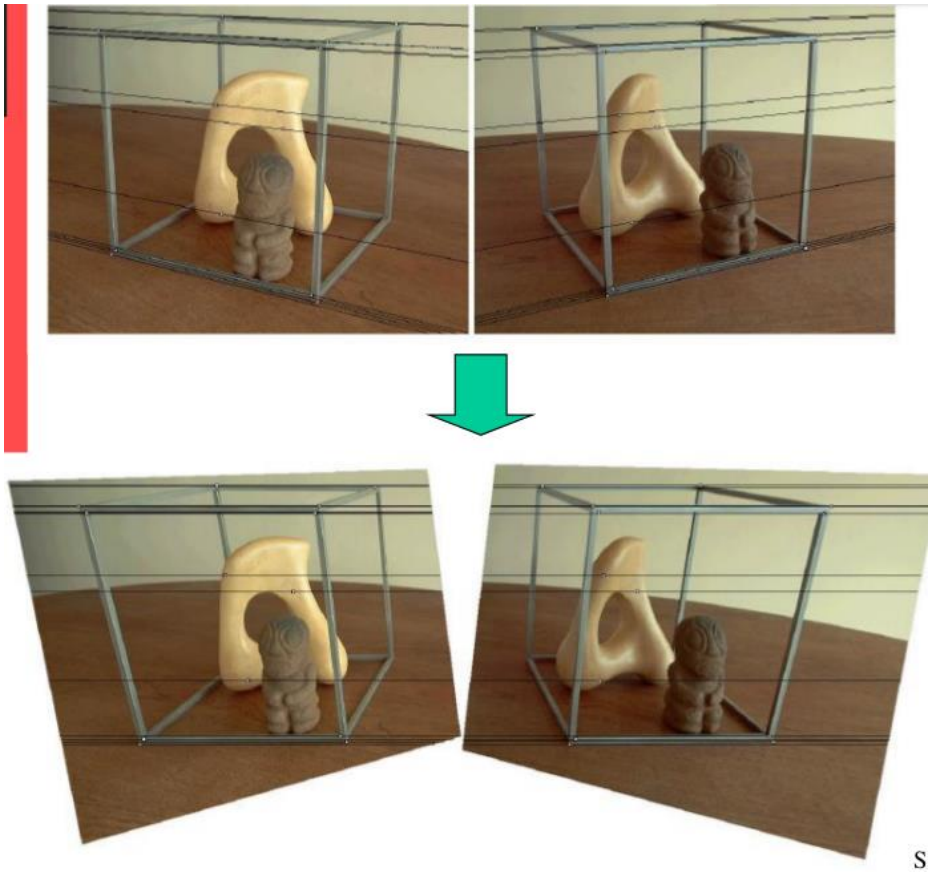


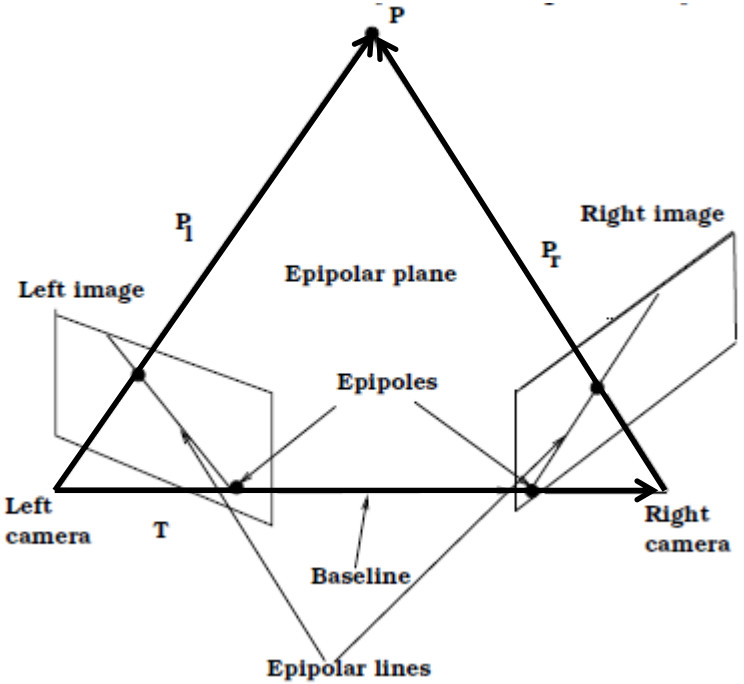
Figure from Stack Overflow

Recover 3D Information – Estimate Depth

How to rectify images?

$$P_l = \mathbf{R}P_r + T$$

How to get **R** & **T**?



Assume camera parameters (intrinsic and extrinsic) are known for both cameras, we can estimate the relative rotation and translation between two cameras:

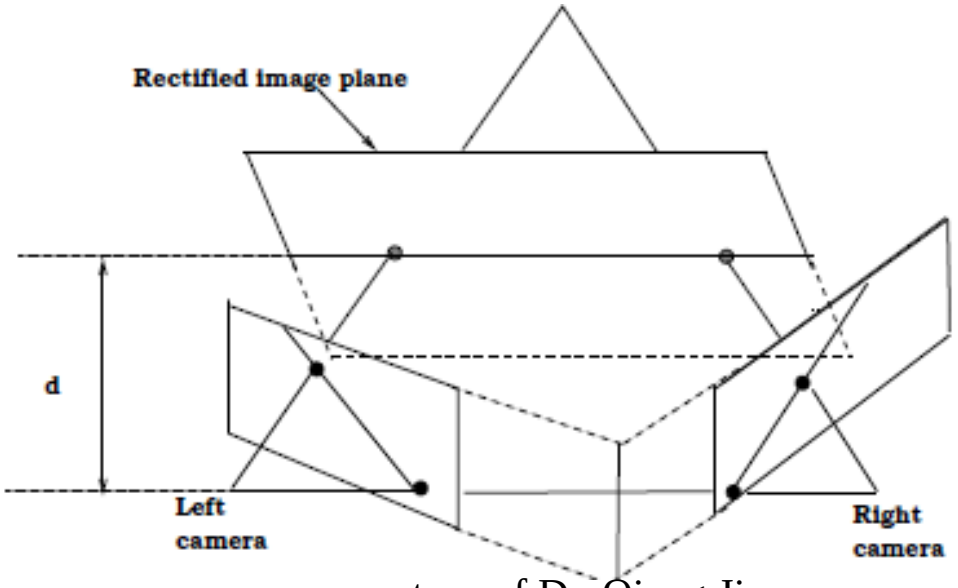
$$\mathbf{R} = \mathbf{R_l}\mathbf{R_r}^{\mathbf{t}} \qquad T = T_l - \mathbf{R}T_r$$

Rectification

Major steps (Trucco & Verri):

- Rotate the left camera such that the image plane of left camera is parallel to the baseline
- Rotate the right camera with **R**
- Rotate the right camera again using the same rotation matrix for the left camera
- Image reprojection: create

  new 2D images



courtesy of Dr. Qiang Ji

Rectification

Step1: Construct a rotation matrix R_{lrect} for left camera such that the image plane is parallel to the baseline.

Define the new axes

- the new x-axis $[1,0,0]^t$ should be on the base line – along the same direction as **T**

$$\mathbf{v}_1 = \frac{\mathbf{T}}{\|\mathbf{T}\|}$$ → In the original left camera frame

- the new y-axis must be orthogonal to the x-axis
  – cross product of **T** and optical axis $[0,0,1]^t$

$$\mathbf{v}_2 = \mathbf{v}_1 \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{t_x^2 + t_y^2}} \begin{bmatrix} t_y \\ -t_x \\ 0 \end{bmatrix}$$

- the new z-axis is determined as the cross product of the other two

$$\mathbf{v}_3 = \mathbf{v}_1 \times \mathbf{v}_2$$

Rectification

Step1: Construct a rotation matrix R_{lrect} for left camera such that the image plane is parallel to the baseline.

Let $\mathbf{r}_{lre_1}$, $\mathbf{r}_{lre_2}$, and $\mathbf{r}_{lre_3}$ be the three rows of R_{lrect}

$$R_{lrect}\mathbf{v}_1 = \begin{bmatrix}1\\0\\0\end{bmatrix} \Rightarrow \quad \mathbf{r}_{lre_1} = \mathbf{v}_1 = \frac{\mathbf{T}}{\|\mathbf{T}\|}$$

$$R_{lrect}\mathbf{v}_2 = \begin{bmatrix}0\\1\\0\end{bmatrix} \Rightarrow \quad \mathbf{r}_{lre_2} = \mathbf{v}_2 = \frac{1}{\sqrt{t_x^2 + t_y^2}}\begin{bmatrix}t_y\\-t_x\\0\end{bmatrix}$$

$$R_{lrect}\mathbf{v}_3 = \begin{bmatrix}0\\0\\1\end{bmatrix} \Rightarrow \quad \mathbf{r}_{lre_3} = \mathbf{v}_3 = \mathbf{r}_{l_1} \times \mathbf{r}_{l_2}$$

$$\mathbf{R}_{lrect} = \begin{bmatrix}\mathbf{r}_{lre_1}^t\\\mathbf{r}_{lre_2}^t\\\mathbf{r}_{lre_3}^t\end{bmatrix}$$

Rectification

Step 2: image reprojection for the left image

$U_l = (c_l, r_l, 1)^t$ ⟶ Original image point before rectification

$U_l' = (c_l', r_l', 1)^t$ ⟶ New image point after rectification

What is the relationship between U_l and U_l'?

They are created by the projection of the same 3D point on two image planes: same $\boldsymbol{W_l}$, rotated by $\boldsymbol{R_{lrect}}$

$$\lambda_l \begin{bmatrix} c_l \\ r_l \\ 1 \end{bmatrix} = \boldsymbol{W_l P_l} \qquad \lambda_l' \begin{bmatrix} c_l' \\ r_l' \\ 1 \end{bmatrix} = \boldsymbol{W_l R_{lrect} P_l}$$

$$\Rightarrow \quad \lambda_l \begin{bmatrix} c_l' \\ r_l' \\ 1 \end{bmatrix} = \boldsymbol{W_l R_{lrect} W_l^{-1}} \begin{bmatrix} c_l \\ r_l \\ 1 \end{bmatrix}$$

Rectification

Step 3: Compute the rotation matrix for the right camera

$$\mathbf{R}_{rrect} = \mathbf{R}_{lrect}\mathbf{R}$$

Step 4: image reprojection for the right image

$$\lambda_r \begin{bmatrix} {c_r}' \\ {r_r}' \\ 1 \end{bmatrix} = \boldsymbol{W_r R_{rrect} {W_r}^{-1}} \begin{bmatrix} c_r \\ r_r \\ 1 \end{bmatrix}$$

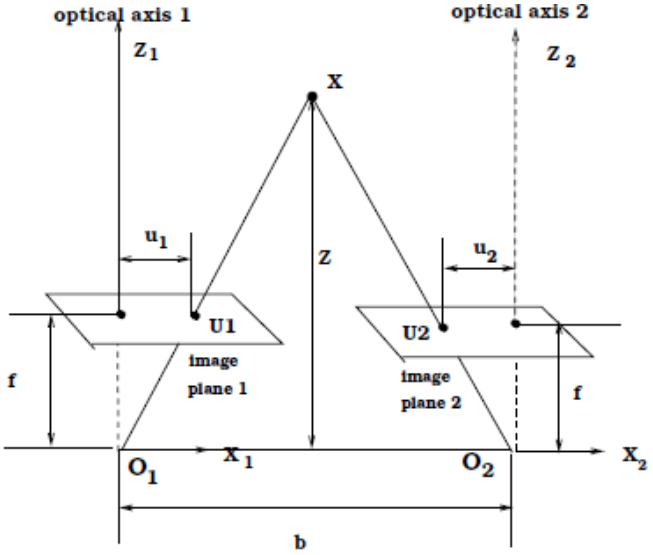
Issues on the Rectification Algorithm

Issue 1: We assume W_r=W_l. However, it is not the case in general.

Solution: $W^{rect}{}_l = W^{rect}{}_r = (W_r + W_l)/2$

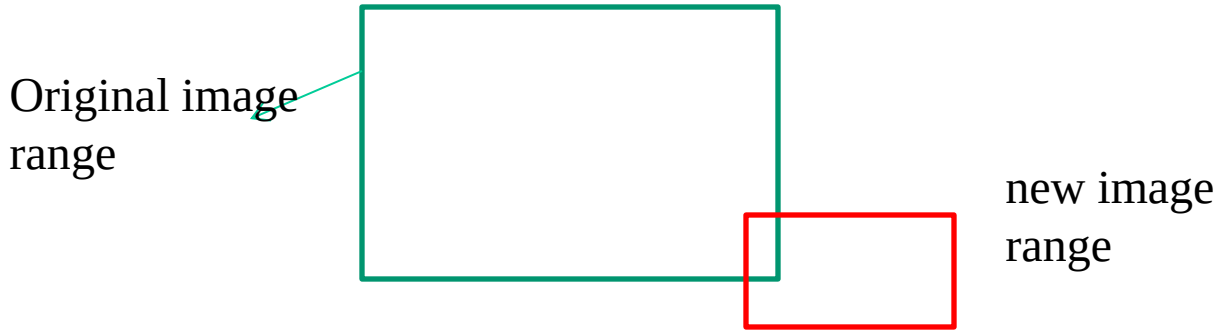
$$\lambda_l \begin{bmatrix} c_l{}' \\ r_l{}' \\ 1 \end{bmatrix} = W^{rect}{}_l R_{lrect} W_l{}^{-1} \begin{bmatrix} c_l \\ r_l \\ 1 \end{bmatrix}$$

$$\lambda_r \begin{bmatrix} c_r{}' \\ r_r{}' \\ 1 \end{bmatrix} = W^{rect}{}_r R_{rrect} W_r{}^{-1} \begin{bmatrix} c_r \\ r_r \\ 1 \end{bmatrix}$$



Issues on the Rectification Algorithm

Issue 2: the reprojected image is not in the original range



Solution: Post processing (rescaling and translating)

- Adjust $\boldsymbol{fs}_x$ and $\boldsymbol{fs}_y$ in the $\boldsymbol{W^{rect}}_{\boldsymbol{l}}$ and $\boldsymbol{W^{rect}}_{\boldsymbol{r}}$ when necessary, so that the rectified image fits to the original range
  - Same scaling factor is applied to the row and column
- Adjust $\boldsymbol{c}_0$ in the $\boldsymbol{W^{rect}}_{\boldsymbol{l}}$ and $\boldsymbol{W^{rect}}_{\boldsymbol{r}}$, respectively

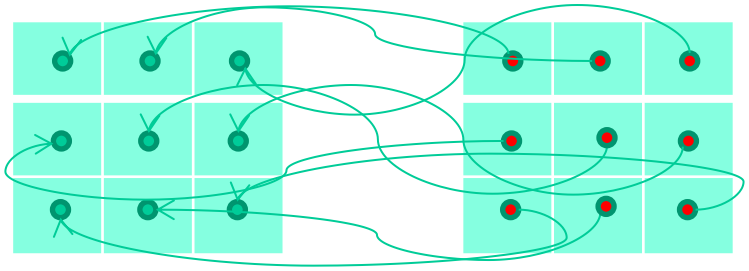
Issues on the Rectification Algorithm

Issue 3: Missing points in forward mapping

Solution: Backward mapping

For each point in the output rectified image,

$$\lambda_r \begin{bmatrix} c_r \\ r_r \\ 1 \end{bmatrix} = W_r R_{rrect}{}^t (W^{rect}{}_r)^{-1} \begin{bmatrix} c_r{}' \\ r_r{}' \\ 1 \end{bmatrix} \qquad \lambda_l \begin{bmatrix} c_l \\ r_l \\ 1 \end{bmatrix} = W_l R_{lrect}{}^t (W^{rect}{}_l)^{-1} \begin{bmatrix} c_l{}' \\ r_l{}' \\ 1 \end{bmatrix}$$



Input Output

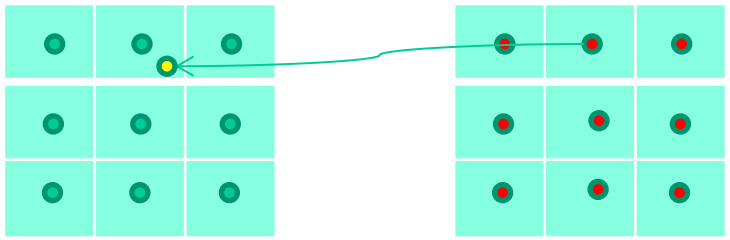
Issues on the Rectification Algorithm

Issue 4: the resulting rectified coordinates are not integer.

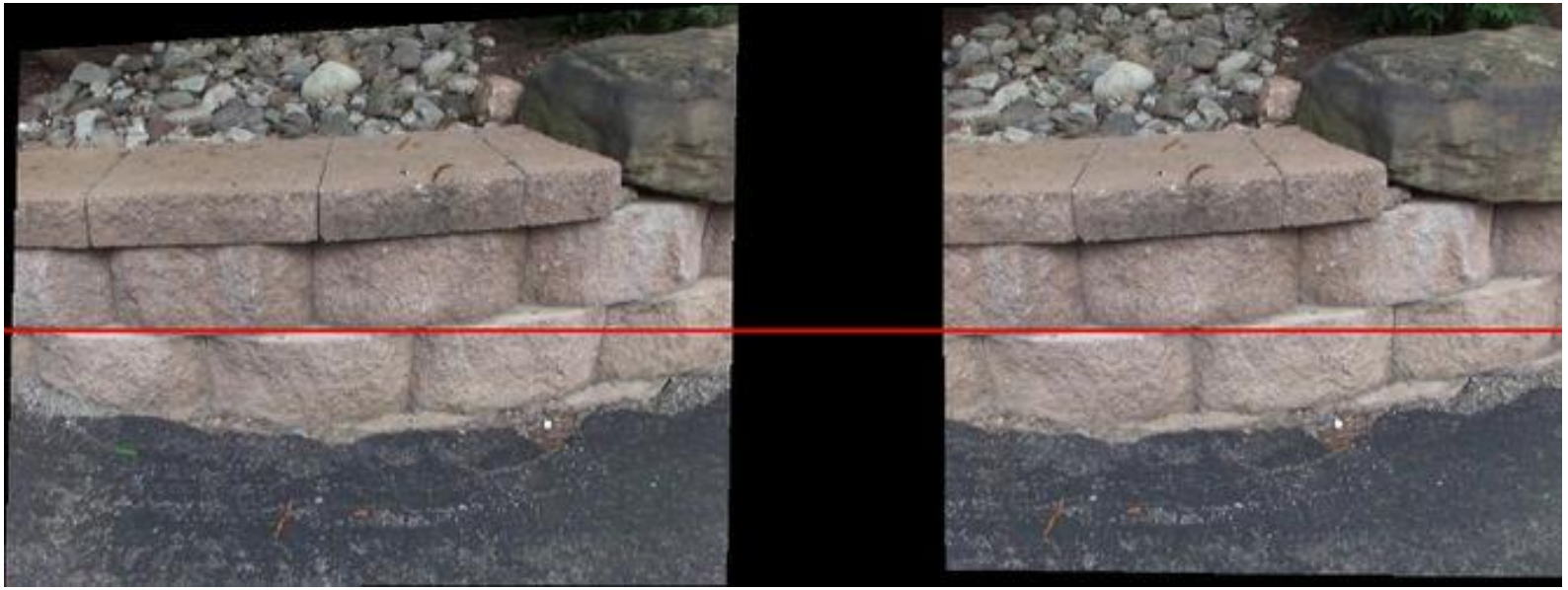
Solution:

An interpolation process is needed:

For example, bilinear interpolation
http://en.wikipedia.org/wiki/Bilinear_interpolation



Rectification Examples



http://boofcv.org/index.php?title=Example_Rectification_Uncalibrated

Reference

OpenCV implementation

http://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html#stereorectify

Matlab implementation in Computer Vision toolbox

http://www.mathworks.com/help/vision/ref/rectifystereoimages.html?requestedDomain=www.mathworks.com

Establish Correspondence

Similar ideas as obtaining measurements in tracking

- Correlation-based
  - Dense correspondences
    - Minimize SSD (Sum Squared Difference) $\sum(I_{i,j} - T_{i,j})^2$
    - Maximize Cross-correlation $\sum I_{i,j} T_{i,j}$
  - What are potential issues with SSD and cross-correlation
    - Illumination variations
    - Solution: Normalized data
- Feature-based
  - Sparse correspondences

How to specify search area

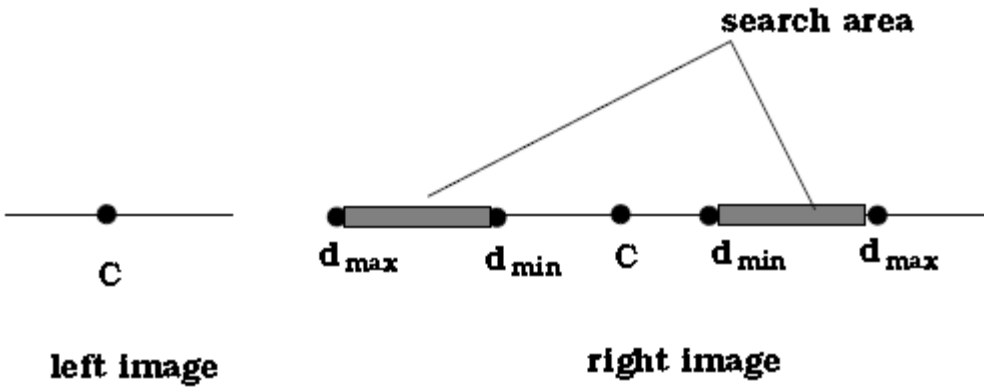
Along the same row within a range defined by disparity range $[c \pm d_{min}, c \pm d_{max}]$

Establish Correspondence

$$Z = \frac{fb}{u_1 - u_2} \quad \Rightarrow \quad d = u_1 - u_2 = \frac{fb}{z}$$

$$d_{min} = \frac{fb}{z_{max}} \qquad \text{and} \qquad d_{max} = \frac{fb}{z_{min}}$$

$$[c \pm d_{min}, c \pm d_{max}]$$



courtesy of Dr. Qiang Ji

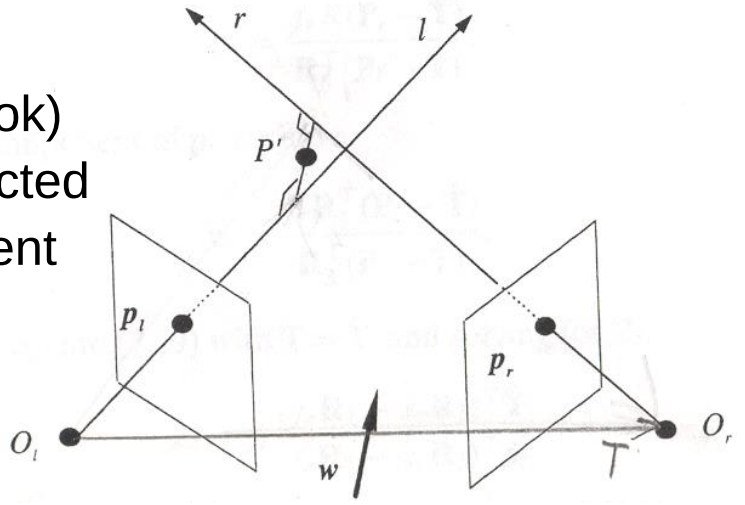
d_{min} is often set to zero

$$\Rightarrow \quad [c - d_{max}, c + d_{max}]$$

3D Reconstruction

Known camera parameters – full reconstruction

- Geometric solution:
  - Intersection of two rays (See Trucco's book)
  - Problem: the two rays may be not intersected
  - Solution: find the midpoint of a line segment perpendicular to both rays
- Algebraic solution
  - Linear solution
  - Non-linear solution



Unknown camera parameters

- Unknown extrinsic parameters – metric or Euclidean reconstruction
  - Reconstruction with respect to left/right camera coordinate system
  - Reconstruction up to a scale factor
- Unknown camera parameters – projective reconstruction
  - Reconstruction up to an unknown projective transformation matrix

3D Reconstruction -- Known Camera Parameters, Linear Solution

Known:

- A pair of 2D image points on the left and right images
- Intrinsic parameters of the two cameras $\boldsymbol{W_l}$ and $\boldsymbol{W_r}$
- The relative rotation and translation between two cameras **R** and **T**

$$\boldsymbol{\lambda_l}\begin{bmatrix} c_l \\ r_l \\ 1 \end{bmatrix} = \boldsymbol{W_l}\begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad \boldsymbol{\lambda_r}\begin{bmatrix} c_r \\ r_r \\ 1 \end{bmatrix} = \boldsymbol{W_r}[\mathbf{R} \quad \mathbf{T}]\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- 6 equations for a pair of image points

Unknown:

- 5 unknowns: λ_l, λ_r, x, y, and z ($[x, y, z]^t$ is **the reconstructed 3D point in the left camera system**)

⇒ Solving the unknown using a linear equation system

3D Reconstruction -- Unknown Extrinsic Parameters

Known:

- A pair of 2D image points on the left and right images
- Intrinsic parameters of the two cameras $\boldsymbol{W_l}$ and $\boldsymbol{W_r}$

Unknown:

- The relative rotation and translation between two cameras **R** and **T**
- 5 unknowns: λ_l, λ_r, x, y, and z ($[x, y, z]^t$ is the reconstructed 3D point in the left camera system)

Solution:

- Solve **F** matrix using 8-point algorithm
- Recover **E** from **F**
- Recover **R** and **T** from **E**
  - Horn's method

  B.K.P.Horn.Relative Orientation. International Journal Of Computer Vision,4(1):59–78, 1990.
- Follow the solution with known intrinsic & extrinsic parameters

Reading Assignments

Chapter 7 (Stereopsis) in Trucco and Verri

Hierarchy of Computer Vision Problems

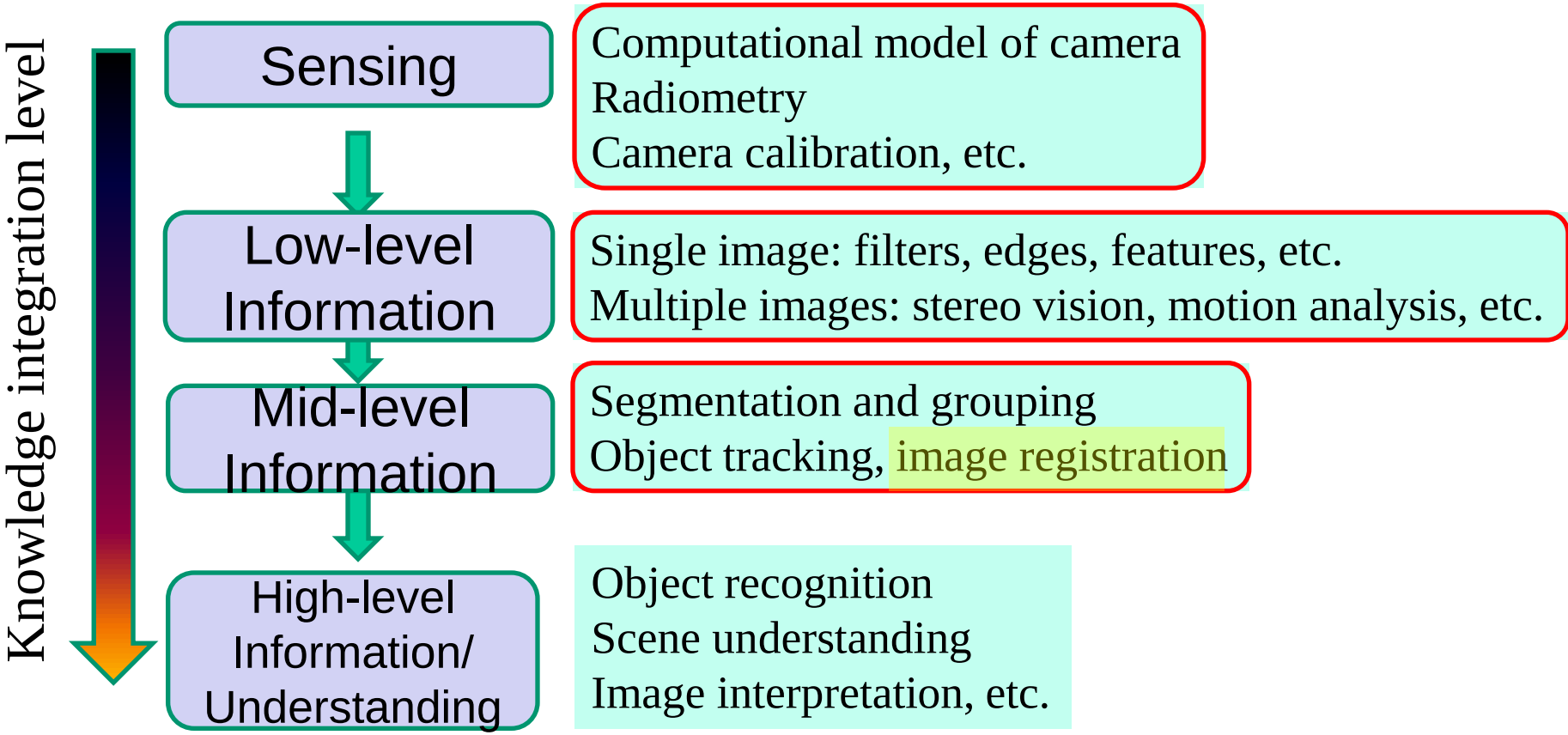


Image Mosaic -- Application of Image Registration

Panorama

Original Frames

Mosaic

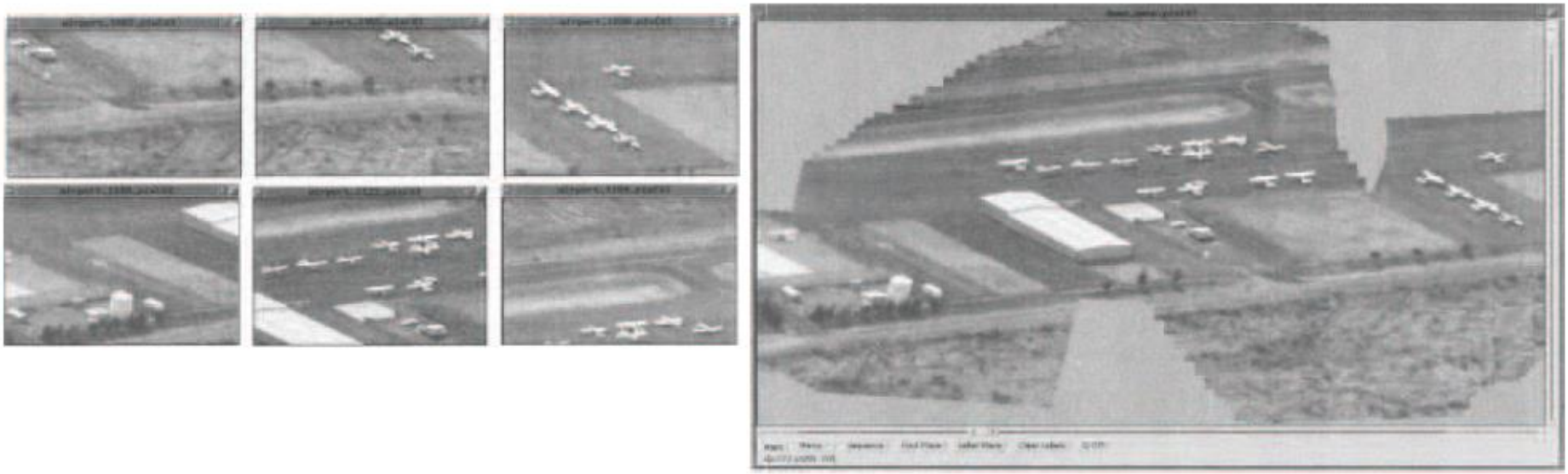


FIGURE 12.1: On the **left**, frames from a video taken by an aircraft overflying an airport. These frames are rectified to one another to form a mosaic on the **right**, which reveals (a) the overall structure of what was seen and (b) the flight path of the aircraft. *This figure was originally published as Figure 1 of "Video Indexing Based on Mosaic Representations," by M. Irani and P. Anandan, Proc. IEEE, v86 n5, 1998,* © *IEEE, 1998.*

FIGURE 12.4: The two mountain images of Figure 12.3, now rectified with a homography. Notice how well all features line up; this transformation involves more than just rotation and translation, as you can see from the fact that the corner of the second image (which can be seen in the middle, near the top), is no longer a right angle. Notice also that intensity effects in the camera far field mean that the boundary where the two images overlap is unpleasantly obvious. *This figure was originally published as Figure 1 M. Brown and D. Lowe, "Recognizing Panoramas," Proc. ICCV 2003, © IEEE, 2003.*

FIGURE 12.5: **Top**, 80 images registered automatically to one another to create a panoramic mosaic (which is what one would see if the camera had cylindrical film, and a 360^o field of view). **Bottom**, the images feathered into one another to suppress the effects of intensity variation between different views of the same pixel. *This figure was originally published as Figure 3 M. Brown and D. Lowe, "Recognizing Panoramas," Proc. ICCV 2003, © IEEE, 2003.*

Image Registration

Problem: given two or multiple images (or 3D volume), compute the transformation between images (or 3D volume)

$$\mathbf{I_1} = \mathbf{H}\mathbf{I_2}$$

Major steps:

- Find interest points
- Compute features such as SIFT features
- Match the interest points and establish correspondences
- Compute transformations

Image Registration

$$\mathbf{I_1 = HI_2}$$

Different forms of transformation

- Rigid object (building, car, etc)
  - Homography for planar object
    - Rigid transformation (e.g., aerial images)
      - Translation, rotation, and scaling
    - Affine homography

- Non-rigid object (people, medical image, animals, etc)
  - Free-form deformation
  - Model-based (e.g., Active Appearance Model)

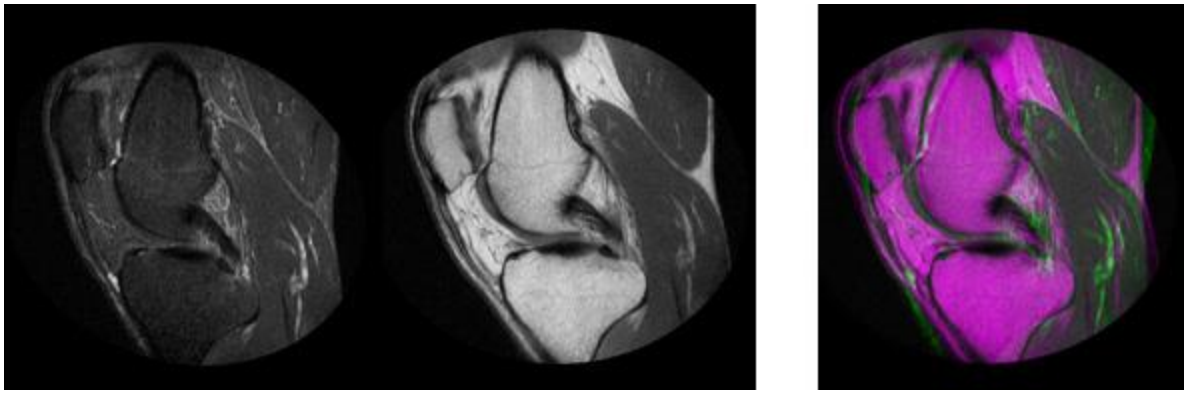
Examples of Image Registration

Moving image Reference image Registered image

Rigid object

Moving image Reference image Registered image

Non-Rigid object



http://www.mathworks.com