KAIST VISUAL COMPUTING Lab

HW3: Stereo Imaging

CS484 Introduction to Computer Vision

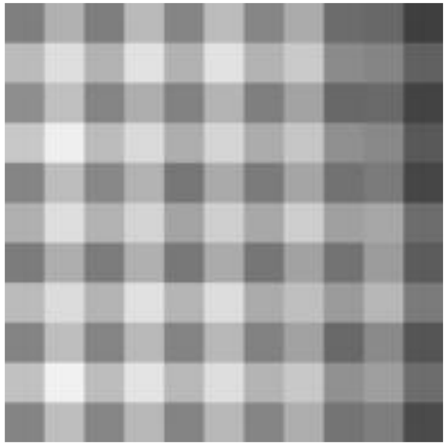
Homework 3 supplementary slides

Carefully look all materials

(webpage, code and supplementary slides)

Filter Demosaic

- Demosaic the raw image using the following three methods
  1. Down-sampling
  2. Bilinear interpolation
  3. Bicubic interpolation



Bayer pattern (RGGB)

Raw image

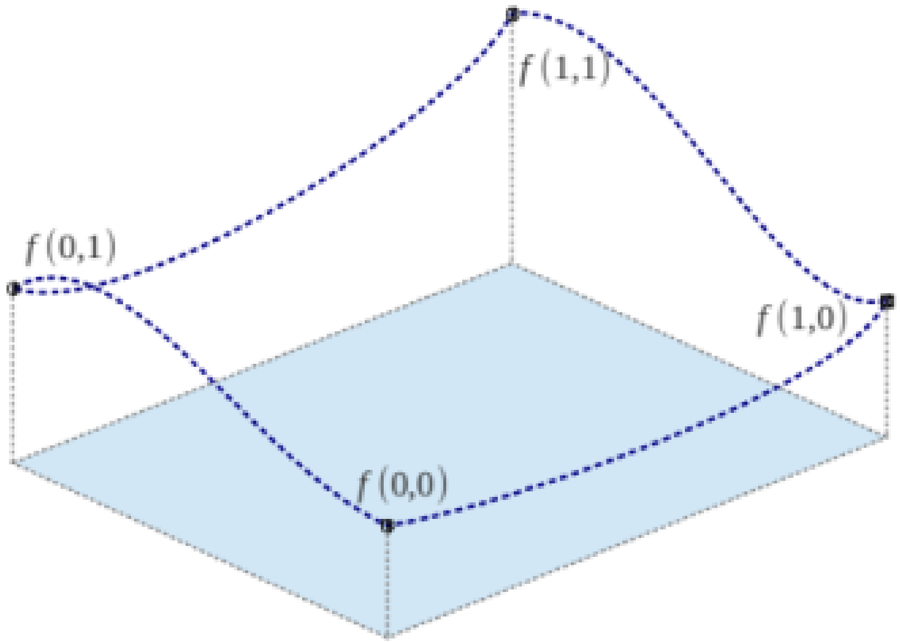
Color image (reference)

Demosaic image

(Optional) Bicubic Interpolation

- Bicubic interpolation (+10 pts)

$$f(x,y)=\sum_{i=0}^{3}\sum_{j=0}^{3}a_{ij}x^i y^j \quad x,y\in[0,1]\times[0,1]$$



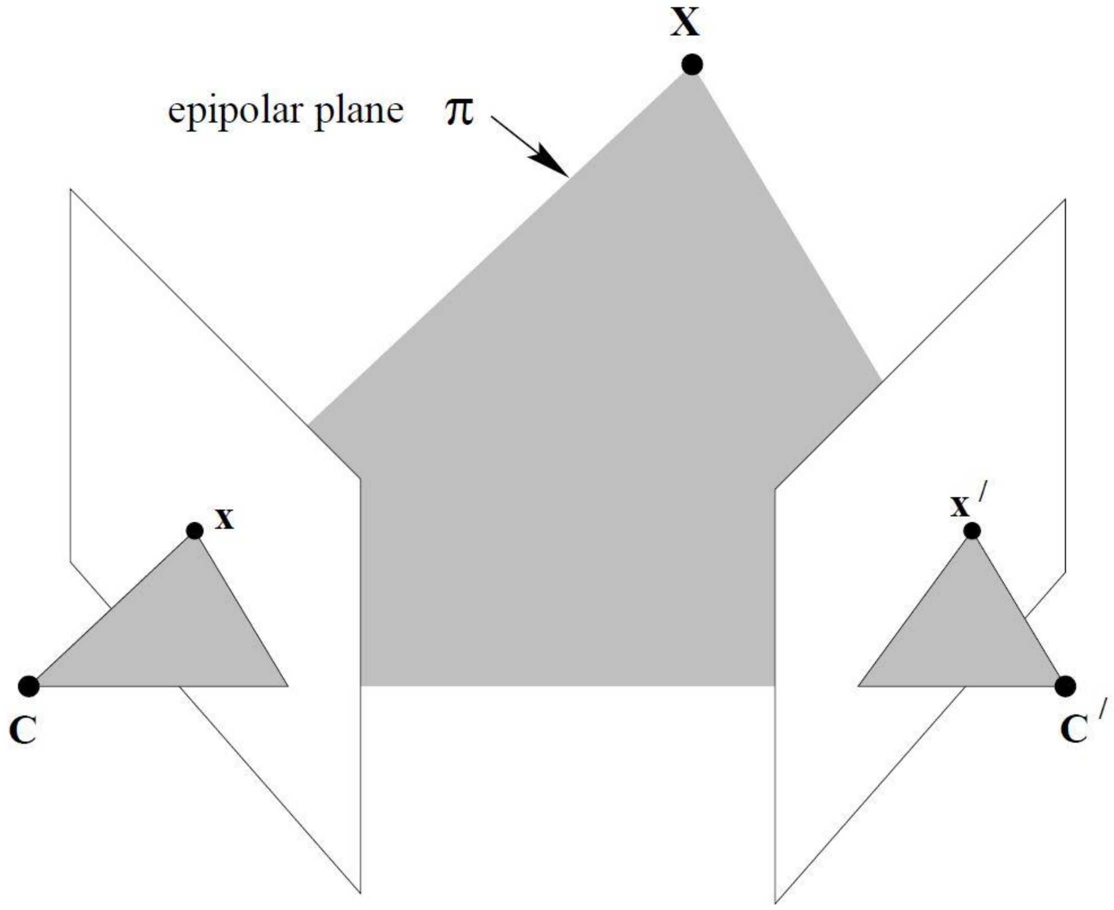
- 16 unknown coefficients a_{ij}a
- 16 known equations
  - $f(0,0), f(0,1), f(1,0), f(1,1)$
  - $f_x(0,0), f_x(0,1), f_x(1,0), f_x(1,1)$
  - $f_y(0,0), f_y(0,1), f_y(1,0), f_y(1,1)$
  - $f_{xy}(0,0), f_{xy}(0,1), f_{xy}(1,0), f_{xy}(1,1)$
- Difference approximation

$$f_x(x,y)=\left[f(x+1,y)-f(x-1,y)\right]/\,2$$
$$f_y(x,y)=\left[f(x,y+1)-f(x,y-1)\right]/\,2$$
$$f_{xy}(x,y)=\left[f(x+1,y+1)+f(x-1,y-1)-f(x+1,y-1)-f(x-1,y+1)\right]/\,4$$

Epipolar geometry

World coordinate $\mathbf{X}$ projects to image coordinate $\mathbf{x}$ and $\mathbf{x}'$

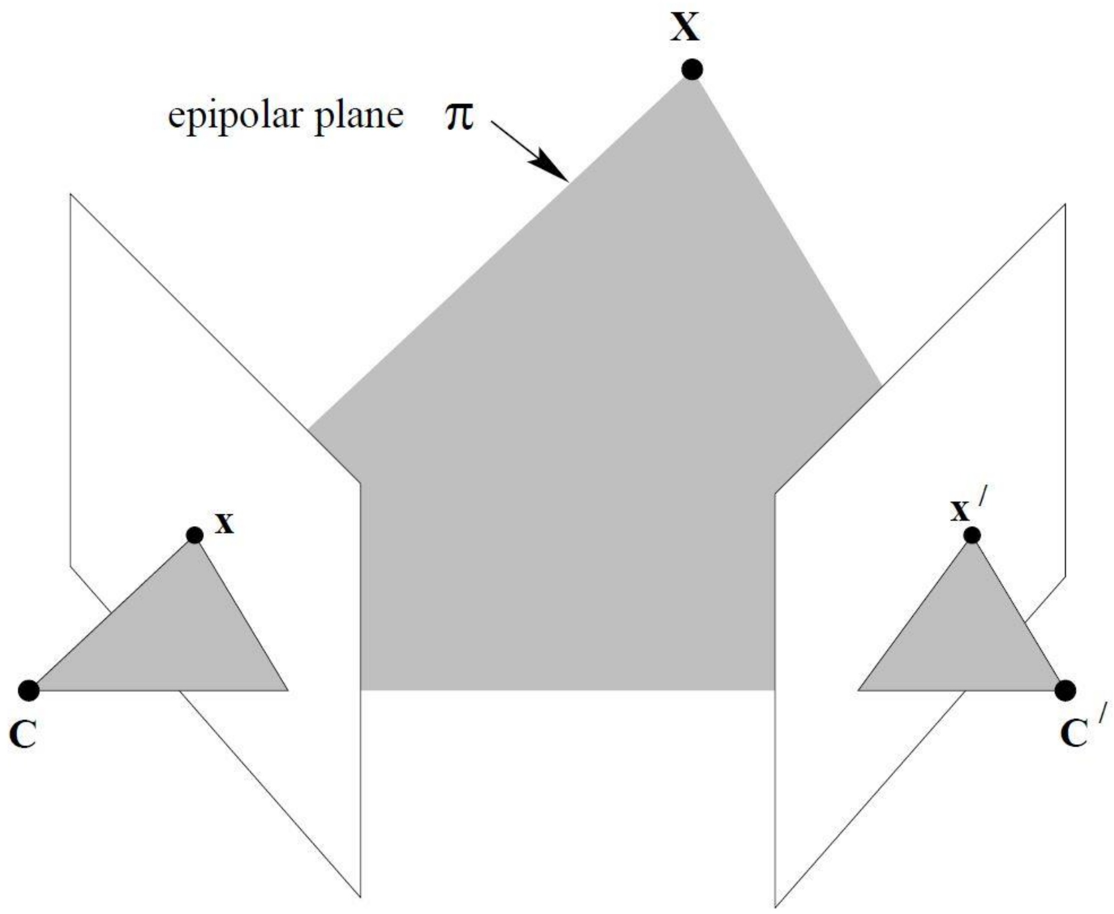
What is the relation between $\mathbf{x}$ and $\mathbf{x}'$?



Epipolar geometry

The camera centers $\mathbf{C}$ and $\mathbf{C}'$, a 3D point $\mathbf{X}$, and its image $\mathbf{x}$ and $\mathbf{x}'$ lie in a common plane π.

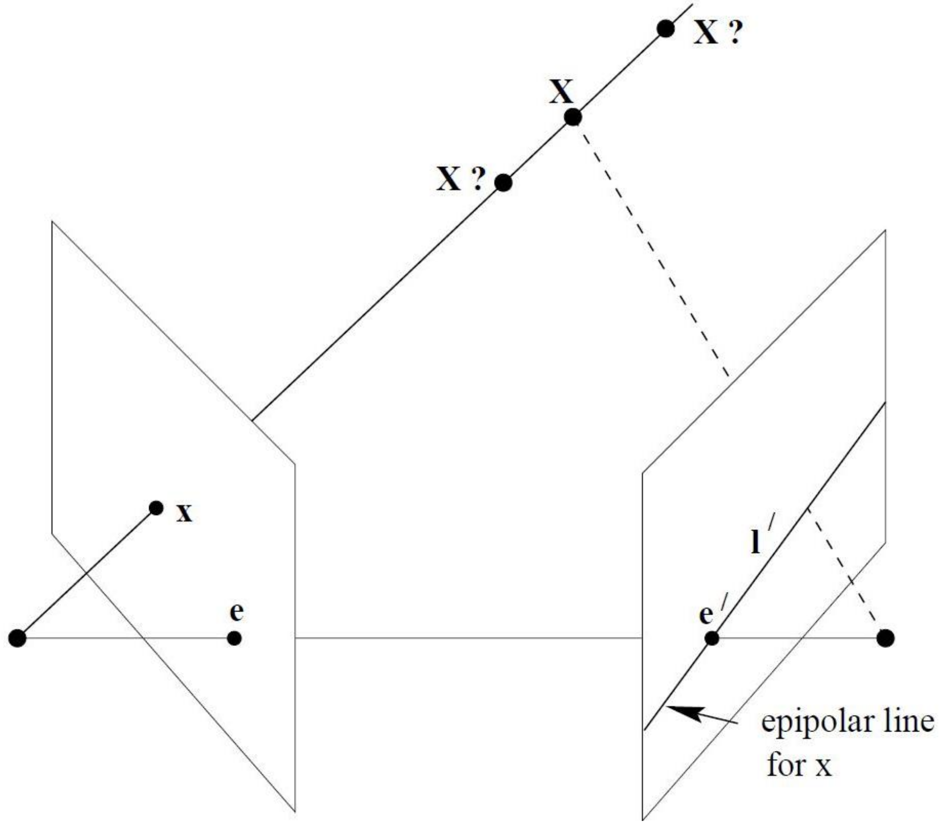
The plane π is **epipolar plane**.



Epipolar geometry

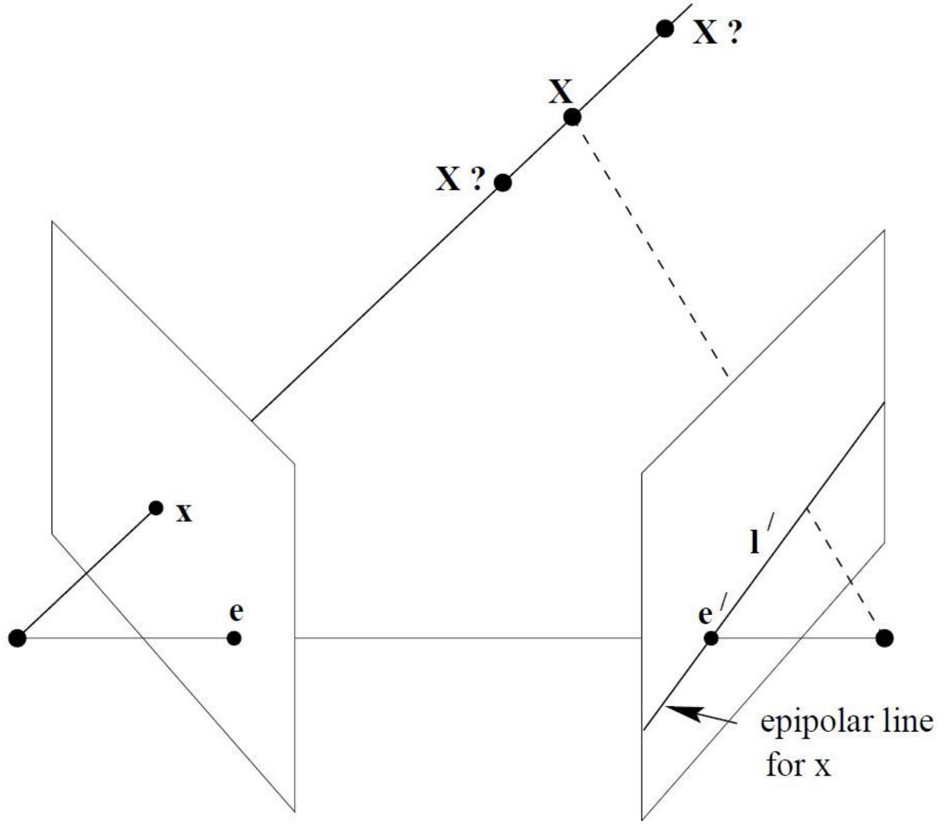
World coordinate $\mathbf{X}$ projects to image coordinate $\mathbf{x}$, but It can't distinguish with dots on the ray from $\mathbf{C}$ to $\mathbf{X}$.

The projection of the ray from $\mathbf{C}$ to $\mathbf{X}$ on the image plane 2 is the line $\mathbf{l}'$.



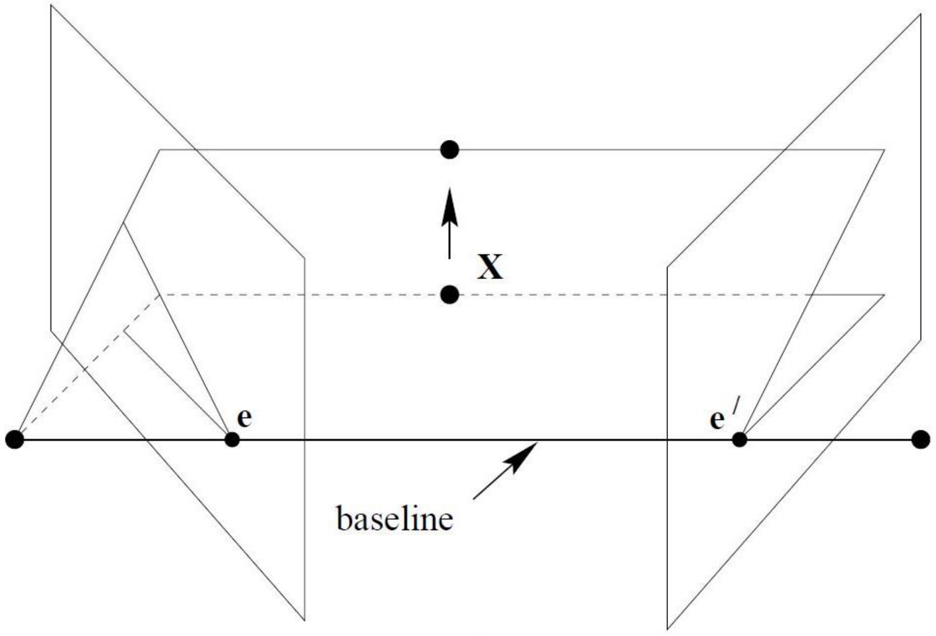
Epipolar geometry

- The line $\mathbf{l}'$ is the **epipolar line**.
- The projection of $\mathbf{X}$ should be on the line $\mathbf{l}'$.
- It is also the intersection of epipolar plane and image plane.



Epipolar geometry

- Intersection of the epipolar planes is **baseline**.
- $\mathbf{C}$ projects to $\mathbf{e}'$, that every epipolar line cross. The point $\mathbf{e}'$ is **epipole**.
- The epipole is the intersection of the baseline and the image plane.
- Every epipolar line intersect on the epipole.



Fundamental matrix

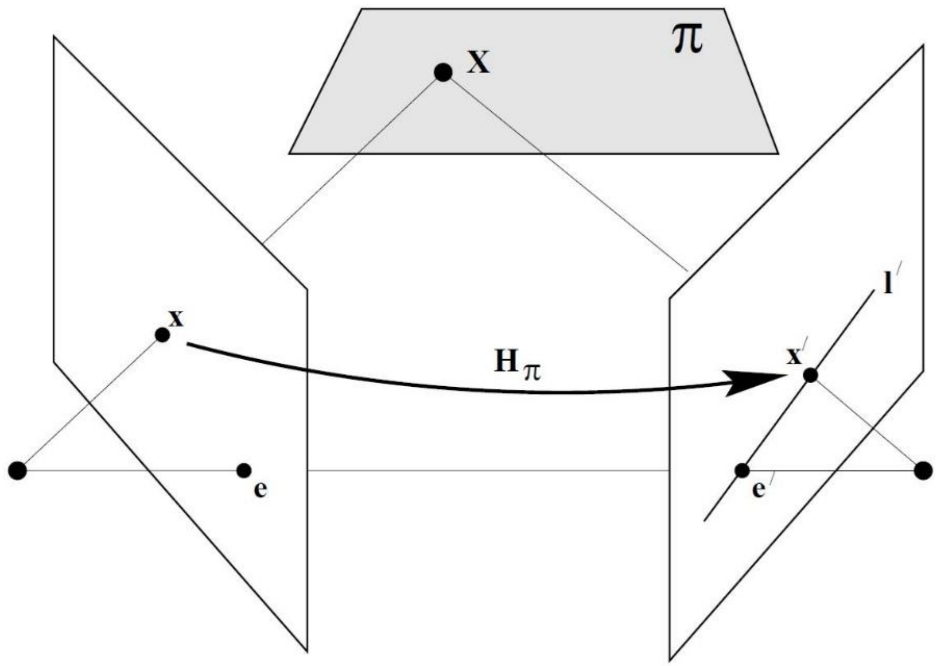
- We want to know the relation between $\mathbf{x}$ and $\mathbf{l}'$.
- The line $\mathbf{l}'$ can be represented by

$$a'x' + b'y' + c' = 0$$

- $\mathbf{l}'$ can be define as

$$\mathbf{l}' = \begin{bmatrix} a' \\ b' \\ c' \end{bmatrix} \quad \mathbf{l}'^T\mathbf{x}' = \mathbf{x}'^T\mathbf{l}' = 0$$

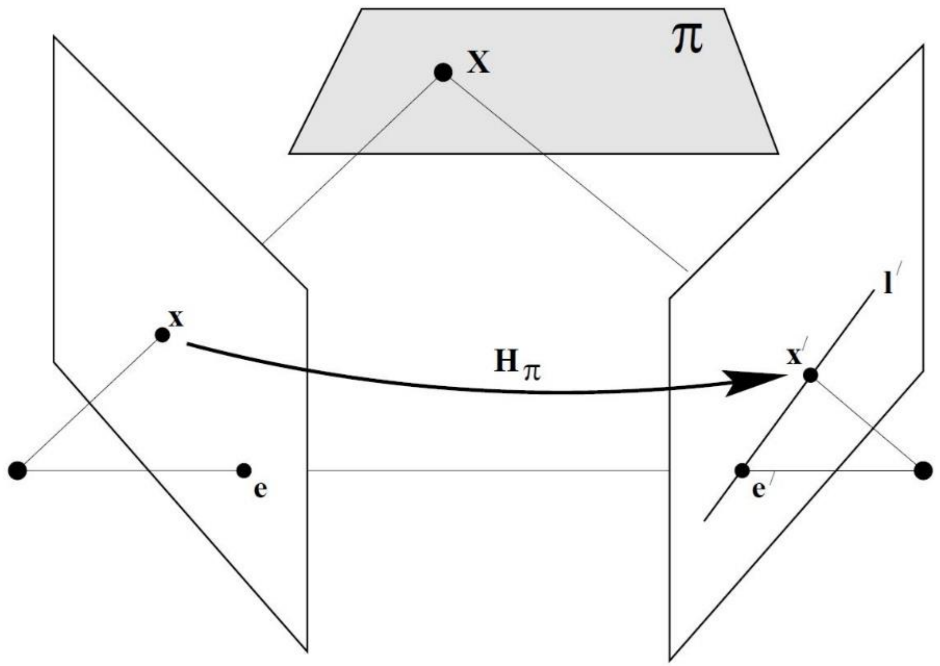
- The scale of $\mathbf{l}'$ can be changed.
(i.e., $k\mathbf{l}'$ for $k \neq 0$ indicates the identical line)



Fundamental matrix

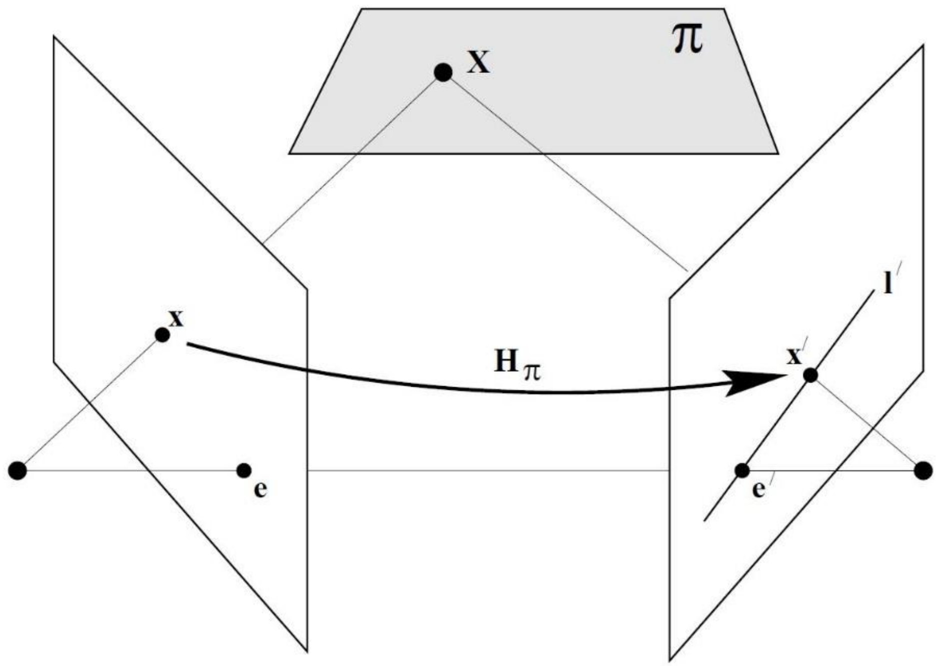
- Line $\mathbf{l}'$ passes through points $\mathbf{x}'$ and $\mathbf{e}'$.
- $\mathbf{l}'$ is perpendicular to both $\mathbf{x}'$ and $\mathbf{e}'$ (as vectors in $\mathbb{R}^3$, just numerically)
- Thus, $\mathbf{l}'$ can be written as $\mathbf{l}' = \mathbf{e}' \times \mathbf{x}'$
- Cross product can be represent by multiplication with a skew-symmetric matrix

$$[\mathbf{e}']_\times = \begin{bmatrix} 0 & -e'_3 & e'_2 \\ e'_3 & 0 & -e'_1 \\ -e'_2 & e'_1 & 0 \end{bmatrix} \quad \mathbf{e}' \times \mathbf{x}' = [\mathbf{e}']_\times \mathbf{x}'$$



Fundamental matrix

- $\mathbf{x}$ can project to **any** plane π. The projected point is $\mathbf{X}$.
- The transformation from a 2D plane to another 2D plane is homography.
- Homography can be represented by 3x3 non-singular matrix.
- Again, $\mathbf{X}$ can project to the image plane. The projected point is $\mathbf{H}_{\pi}\mathbf{x}$, where $\mathbf{H}_{\pi}$ is the homography from the image plane through plane π to another image plane.



Fundamental matrix

- $\mathbf{H}_{\pi}\mathbf{x}$ should be on the epipolar line $\mathbf{l}'$ **whether $\mathbf{H}_{\pi}\mathbf{x}$ is not same with $\mathbf{x}'$.**
- Then, $\mathbf{l}'$ can be written as

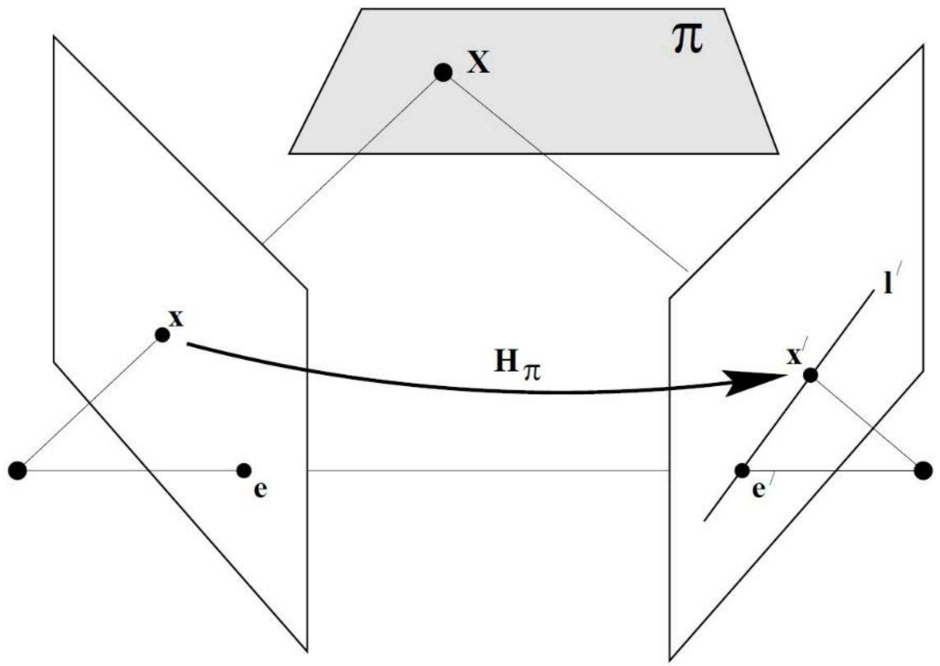
$$\mathbf{l}' = \mathbf{e}' \times \mathbf{H}_{\pi}\mathbf{x} = [\mathbf{e}']_{\times} \mathbf{H}_{\pi}\mathbf{x}$$

- The fundamental matrix F is

$$\mathbf{F} = [\mathbf{e}']_{\times} \mathbf{H}_{\pi}$$

- The relation between $\mathbf{x}$ and $\mathbf{x}'$ is

$$\mathbf{x}'^{T}\mathbf{F}\mathbf{x} = \mathbf{x}'^{T}\mathbf{l}' = 0$$



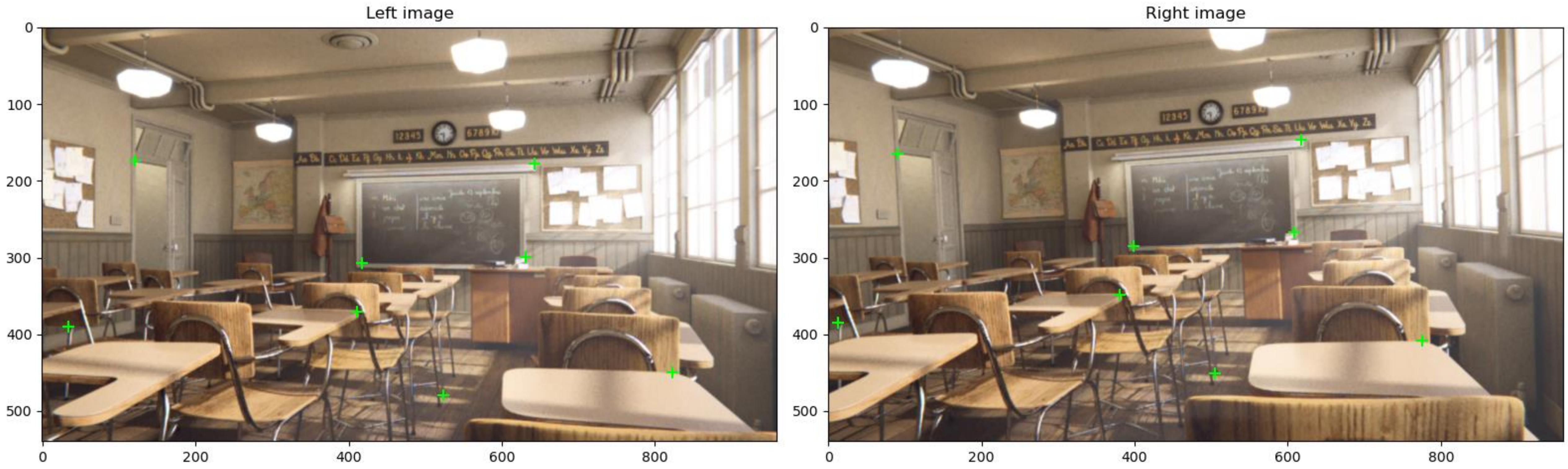
The properties of fundamental matrix

- $\mathtt{F}$ is a rank 2 homogeneous matrix with 7 degrees of freedom.
- **Point correspondence**: If $\mathbf{x}$ and $\mathbf{x}'$ are corresponding image points, then $\mathbf{x}'^{\mathsf{T}}\mathtt{F}\mathbf{x} = 0$.
- **Epipolar lines**:
  - ◇ $\mathbf{l}' = \mathtt{F}\mathbf{x}$ is the epipolar line corresponding to $\mathbf{x}$.
  - ◇ $\mathbf{l} = \mathtt{F}^{\mathsf{T}}\mathbf{x}'$ is the epipolar line corresponding to $\mathbf{x}'$.
- **Epipoles**:
  - ◇ $\mathtt{F}\mathbf{e} = \mathbf{0}$.
  - ◇ $\mathtt{F}^{\mathsf{T}}\mathbf{e}' = \mathbf{0}$.
- **Computation from camera matrices** $\mathtt{P}, \mathtt{P}'$:
  - ◇ General cameras,
    $\mathtt{F} = [\mathbf{e}']_{\times}\mathtt{P}'\mathtt{P}^{+}$, where $\mathtt{P}^{+}$ is the pseudo-inverse of $\mathtt{P}$, and $\mathbf{e}' = \mathtt{P}'\mathbf{C}$, with $\mathtt{P}\mathbf{C} = \mathbf{0}$.
  - ◇ Canonical cameras, $\mathtt{P} = [\mathtt{I} \mid \mathbf{0}]$, $\mathtt{P}' = [\mathtt{M} \mid \mathbf{m}]$,
    $\mathtt{F} = [\mathbf{e}']_{\times}\mathtt{M} = \mathtt{M}^{-\mathsf{T}}[\mathbf{e}]_{\times}$, where $\mathbf{e}' = \mathbf{m}$ and $\mathbf{e} = \mathtt{M}^{-1}\mathbf{m}$.
  - ◇ Cameras not at infinity $\mathtt{P} = \mathtt{K}[\mathtt{I} \mid \mathbf{0}]$, $\mathtt{P}' = \mathtt{K}'[\mathtt{R} \mid \mathbf{t}]$,
    $\mathtt{F} = \mathtt{K}'^{-\mathsf{T}}[\mathbf{t}]_{\times}\mathtt{R}\mathtt{K}^{-1} = [\mathtt{K}'\mathbf{t}]_{\times}\mathtt{K}'\mathtt{R}\mathtt{K}^{-1} = \mathtt{K}'^{-\mathsf{T}}\mathtt{R}\mathtt{K}^{\mathsf{T}}[\mathtt{K}\mathtt{R}^{\mathsf{T}}\mathbf{t}]_{\times}$.

Eight-point algorithm

We want to get a fundamental matrix from two images in different view.

In the images, at least 8 corresponding points are given.



Eight-point algorithm

We want to get a fundamental matrix from two images in different view.

In the images, at least 8 corresponding points are given.

If there are m correspondences, they satisfy

$$\mathbf{x}_i'^T \mathbf{F} \mathbf{x}_i = 0 \quad i = 1, \cdots, m$$

where $\mathbf{x}_i = \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$ $\mathbf{x}_i' = \begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix}$ $\mathbf{F} = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}$

Eight-point algorithm

It can be represented by 9 unknown linear system.

$$\mathbf{Af} = 0$$

where

$$\mathbf{A} = \begin{bmatrix} x'x & x'y & x' & y'x & y'y & y' & x & y & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'x & x'y & x' & y'x & y'y & y' & x & y & 1 \end{bmatrix} \qquad \mathbf{f} = \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix}$$

Eight-point algorithm

Details: $\mathbf{x}_i'^T \mathbf{F} \mathbf{x}_i = 0$ **to** $\mathbf{A}\mathbf{f} = 0$

$$\mathbf{x}_i'^T \quad \mathbf{F} \quad \mathbf{x}_i = 0$$

Eight-point algorithm

Details: $\mathbf{x}_i'^T \mathbf{F} \mathbf{x}_i = 0$ to $\mathbf{A}\mathbf{f} = 0$

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

Eight-point algorithm

Details: $\mathbf{x}_i'^T \mathbf{F} \mathbf{x}_i = 0$ to $\mathbf{A}\mathbf{f} = 0$

$$
\begin{aligned}
&x_i' f_{11} x_i + x_i' f_{12} y_i + x_i' f_{13} 1 \\
&+ y_i' f_{21} x_i + y_i' f_{22} y_i + y_i' f_{23} 1 \\
&+ 1 f_{31} x_i + 1 f_{32} y_i + 1 f_{33} 1 = 0
\end{aligned}
$$

Eight-point algorithm

Details: $\mathbf{x}_i'^T \mathbf{F} \mathbf{x}_i = 0$ **to** $\mathbf{A}\mathbf{f} = 0$

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

- Each equation for index i becomes each row of $\mathbf{A}\mathbf{f} = 0$.

Eight-point algorithm

The only nonzero solution of $\mathbf{Ax} = 0$ can exist if $rank(\mathbf{A}) = 9 - 1 = 8$

(*"only"* up to scaling $\mathbf{f}$)

Each correspondence make one equation (a row of $\mathbf{A}$)

It need eight points!

Eight-point algorithm: Implementation

1. Solve $\min_{\mathbf{f}} ||\mathbf{Af}||^2$ subject to $||\mathbf{f}||^2 = 1$
   - $\mathbf{f}$ ← the eigenvector of $\mathbf{A}^T\mathbf{A}$ corresponding the smallest eigenvalue.

2. $\mathbf{F}(3 \times 3$ fundamental matrix) ← reshape($\mathbf{f}$)

3. Enforce $\mathbf{F}$ to have rank 2
   - $\mathbf{F} = \mathbf{USV}^{\mathrm{T}}$ where $\mathbf{U}$, $\mathbf{S}$, $\mathbf{V}$ are a singular value decomposition for $\mathbf{F}$.
     Make the minimum singular value for $\mathbf{S}$ become zero.
     (the diagonal entries of $\mathbf{S}$ are the singular values for $\mathbf{S}$.)
   - $\mathbf{F}$ ← $\mathbf{USV}^T$ by using modified $\mathbf{S}$ at b.

Eight-point algorithm: Proof 1.

If there are more than 8 correspondences, we should get an approximation.

$$\min_{\mathbf{f}} \|\mathbf{Af}\|^2 \qquad \text{subject to} \qquad \|\mathbf{f}\|^2 = 1$$

$$\mathbf{g}(\mathbf{f}) = \|\mathbf{Af}\|^2 = (\mathbf{Af})^T(\mathbf{Af}) = \mathbf{f}^T\mathbf{A}^T\mathbf{Af}$$

$$\mathbf{h}(\mathbf{f}) = 1 - \|\mathbf{f}\|^2 = 1 - \mathbf{f}^T\mathbf{f}$$

Eight-point algorithm: Proof 1.

Make the Lagrangian of the optimization.

$$L(\mathbf{f},\lambda)=\mathbf{g}(\mathbf{f})-\lambda\mathbf{h}(\mathbf{f})=\mathbf{f}^T\mathbf{A}^T\mathbf{A}\mathbf{f}-\lambda\left(1-\mathbf{f}^T\mathbf{f}\right)$$

$$\min_{\mathbf{f}}\|\mathbf{A}\mathbf{f}\|^2 \quad s.t.\ \|\mathbf{f}\|^2=1 \longrightarrow \min_{\mathbf{f}} L(\mathbf{f},\lambda)$$

Eight-point algorithm: Proof 1.

Take derivatives of the Lagrangian.

$$\partial_{\mathbf{f}} L(\mathbf{f}, \lambda) = \mathbf{A}^T \mathbf{A} \mathbf{f} - \lambda \mathbf{f} = 0$$

$$\partial_{\lambda} L(\mathbf{f}, \lambda) = 1 - \mathbf{f}^T \mathbf{f} = 0$$

$\mathbf{f}$ is normalized eigenvector of $\mathbf{A}^T \mathbf{A}$

Eight-point algorithm: Proof 1.

Let $\mathbf{e}_\lambda$ is an eigenvector with eigenvalue λ.

$$\mathbf{g}(\mathbf{e}_\lambda) = \mathbf{e}_\lambda{}^T \mathbf{A}^T \mathbf{A} \mathbf{e}_\lambda = \mathbf{e}_\lambda{}^T \lambda \mathbf{e}_\lambda = \lambda$$

The eigenvector with the smallest eigenvalue is the result.

Eight-point algorithm: Proof 1.

$$\mathbf{g}(\mathbf{e}_\lambda) = \mathbf{e}_\lambda{}^T \mathbf{A}^T \mathbf{A} \mathbf{e}_\lambda = \mathbf{e}_\lambda{}^T \lambda \mathbf{e}_\lambda = \lambda$$

Details

$$\mathbf{A}^T\mathbf{A} \overset{\text{eigen decom.}}{\Longrightarrow} \mathbf{U}\mathbf{D}\mathbf{U}^{-1}$$

Numerically unstable

$$\mathbf{A} \overset{\text{SVD}}{\Longrightarrow} \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

$$\Longrightarrow \mathbf{A}^T\mathbf{A} = \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T$$

✓ Preferable

- In general, eigen decomposition may produce complex values for real input matrix while singular value decomposition (SVD) always produces real values.
- The exact values of eigen decomposition of a real symmetric ($\mathbf{M}^T = \mathbf{M}$) matrix are real, but numerical methods produce little complex-valued error.
- For a symmetric matrix formed $A^T A$, utilizing SVD for A is preferable.

Eight-point algorithm: Proof 3.

Is the result F have rank 2? ➔ It is not guaranteed.

We should reduce the dimension by singular value decomposition.

- Get SVD of F

$$\mathbf{F} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3 \end{bmatrix} \quad \mathbf{V} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \end{bmatrix} \quad \mathbf{\Sigma} = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix}$$

- Set the smallest singular values to 0

$$\hat{\mathbf{\Sigma}} = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

- Recompute F

$$\hat{\mathbf{F}} = \mathbf{U}\hat{\mathbf{\Sigma}}\mathbf{V}^T$$

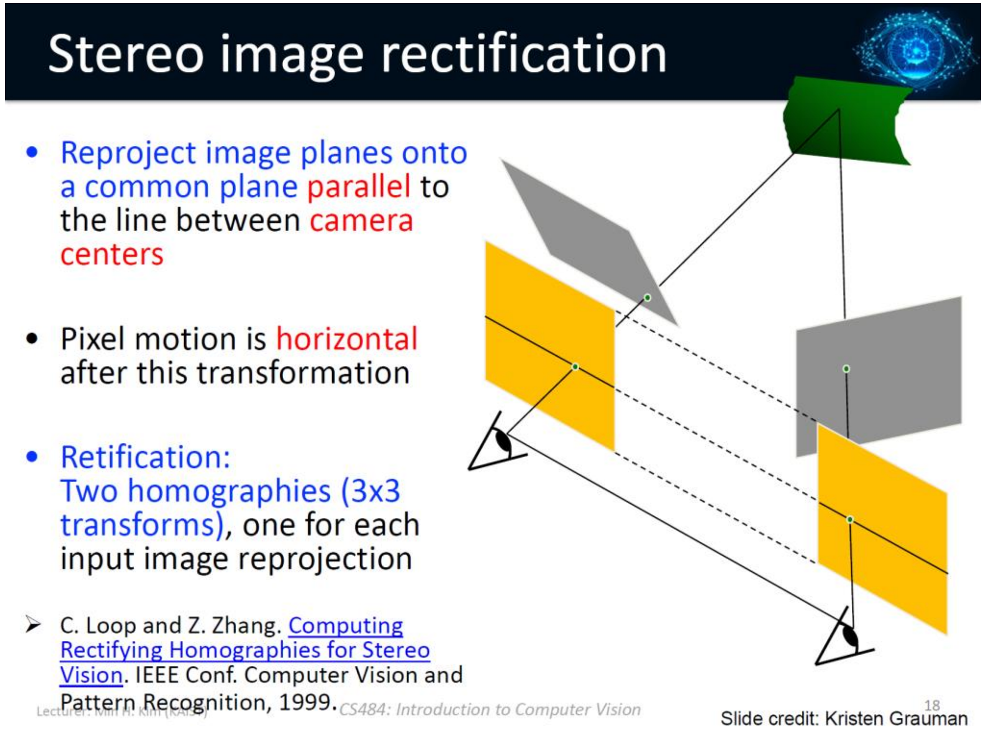
Normalized eight-point algorithm

- Center the image data at the origin, and scale it so the mean squared distance between the origin and the data points is 2 pixels
- Use the eight-point algorithm to compute **F** from the normalized points
- Enforce the rank-2 constraint (for example, take SVD of **F** and throw out the smallest singular value)
- Transform fundamental matrix back to original units: if **T** and **T'** are the normalizing transformations in the two images, then the fundamental matrix in original coordinates is $\mathbf{T'}^T \mathbf{F}\, \mathbf{T}$

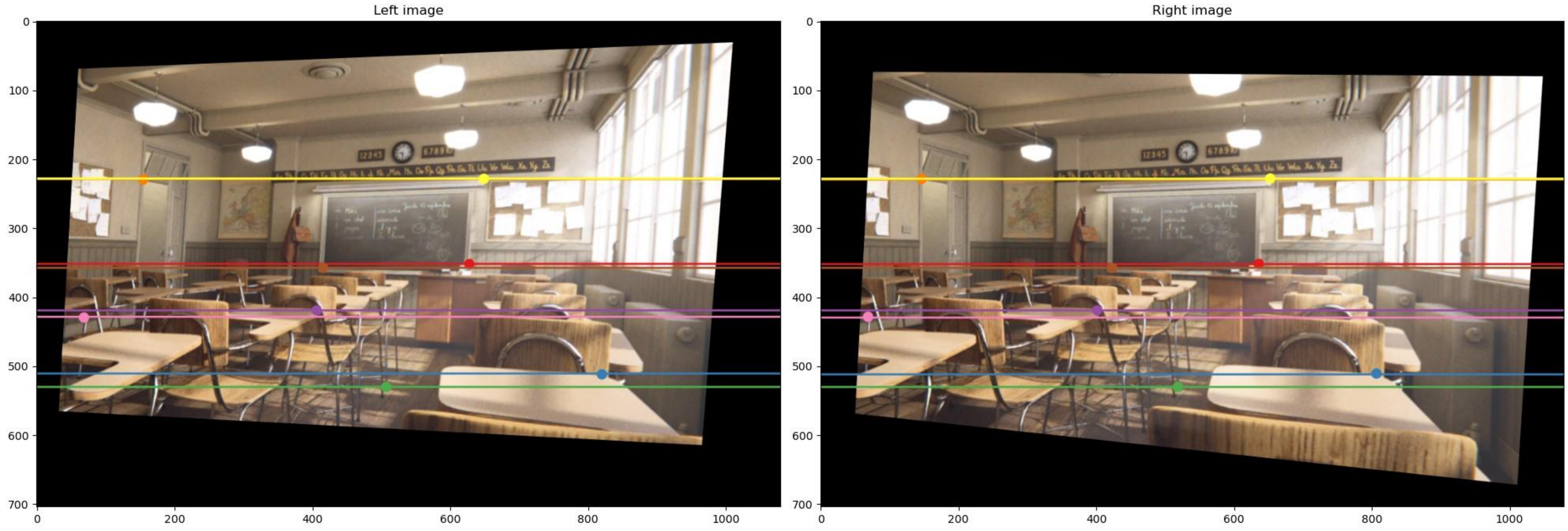
"utils.py" , function "normalize_points"

Rectification

- Left and right image should be reprojected onto the common plane parallel to the line between camera centers
  ➔ **Defined by homography matrices H and H'**
- Calculating H and H' when F and corresponding points are known is already implemented as the OpenCV native function.
- You need to slightly modify H and H' to avoid cropping and apply them to the left and right images.



Rectification



`result/rectified_imgs_epipolar_overlay.png`

(generated in `hw3_main.py` after calling `rectify_stereo_images`)

Rectification

Before rectification

After rectification

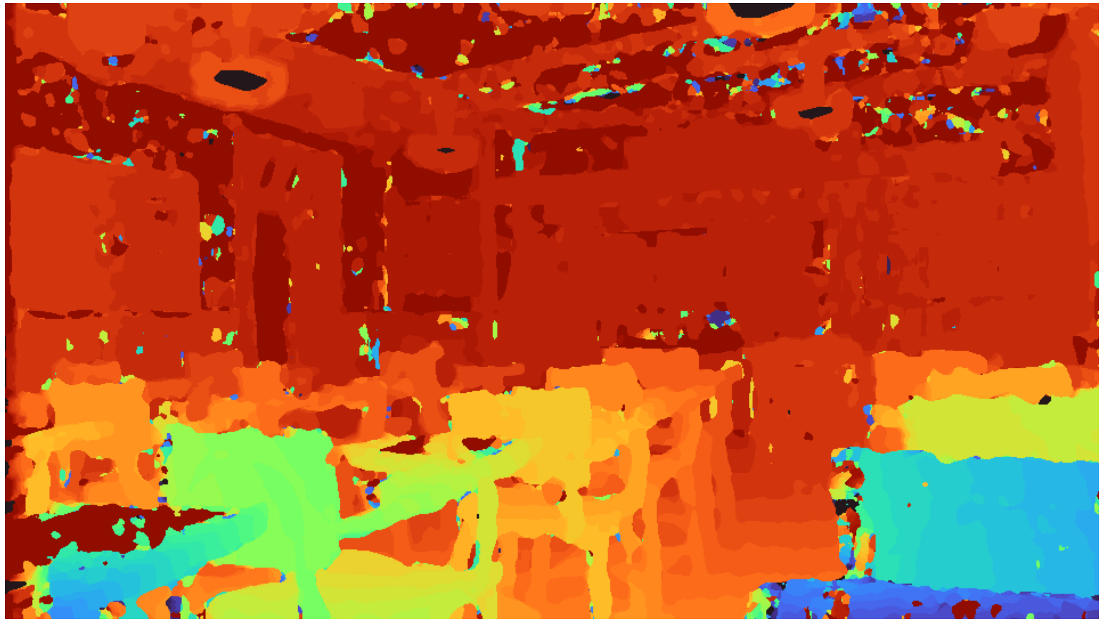
Rectification

- When you implement the part applying H and H' to the left and right images, following openCV functions might be helpful:
  - cv2.warpPerspective() https://docs.opencv.org/4.5.0/da/d54/group__imgproc__transform.html#gaf73673a7e8e18ec6963e3774e6a94b87
  - cv2.perspectiveTransform() https://docs.opencv.org/4.5.0/d2/de8/group__core__array.html#gad327659ac03e5fd6894b90025e6900a7
- Your rectified images should be aligned well as shown in the previous page.

Disparity map

- We provide perfect rectified image (img3.png (left), img4.png (right))
- And Ground truth disparity map of img3 (img3_disp.exr)

Disparity map



- This is one example of disparity map result.
- It may poorly works in some region due to the inherent limitation of uncalibrated stereo problem.
- Also it depends on your hyperparameter.

- Disparity map can be improved by using
  - Cost aggregation with box filter (+5 pts)
  - More sophisticated cost aggregation (+5 pts)
  - Sub-pixel disparity (+10 pts)
  - Calibrated cameras (not for this homework)

Evaluation of disparity map

Evaluation functions are already in “utils.py” and used in “hw3_main.py”

EPE (End-point-error)

$\text{epe} < 3.0$	15pts
$3.0 \leq \text{epe} < 4.0$	10pts
$4.0 \leq \text{epe} < 5.0$	5pts
$5.0 \leq \text{epe}$	0pts

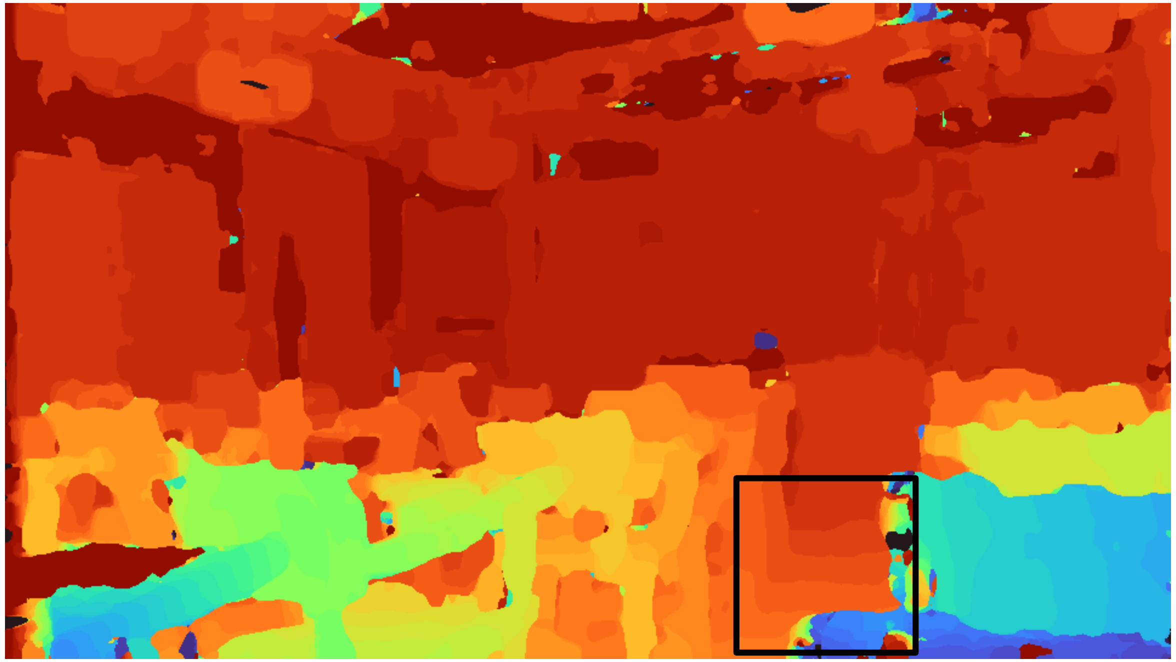
Bad pixel ratio %

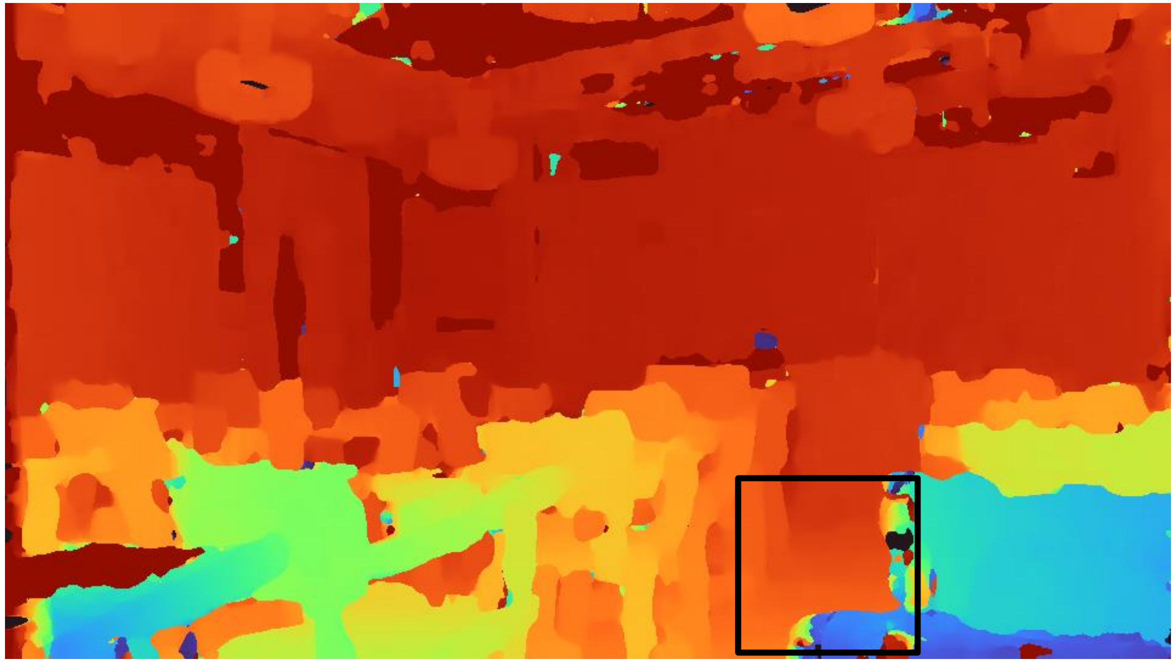
Ratio of pixels epe larger than 3.0

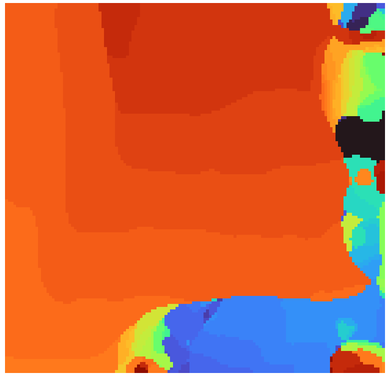
$\text{Bad pix} < 25\%$	15pts
$25\% \leq \text{Bad pix} < 30\%$	10pts
$30\% \leq \text{Bad pix} < 40\%$	5pts
$40\% \leq \text{Bad pix}$	0pts

(Optional) Sub-pixel disparity

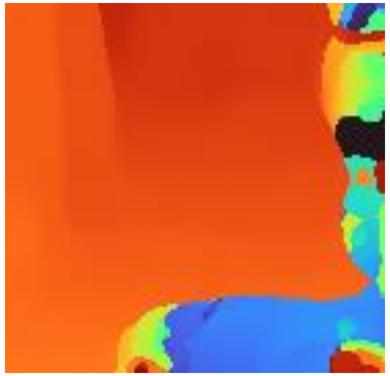
Try this **after** you successfully implement disparity map!







- EPE: 2.4632
- Bad pixel ratio: 21.41%



- EPE: 2.3567
- Bad pixel ratio: 21.31%