Lecture 07 – Single View Geometry

Oct 30nd, 2018

Danping Zou,

Associate Professor

Institute for Sensing and Navigation

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

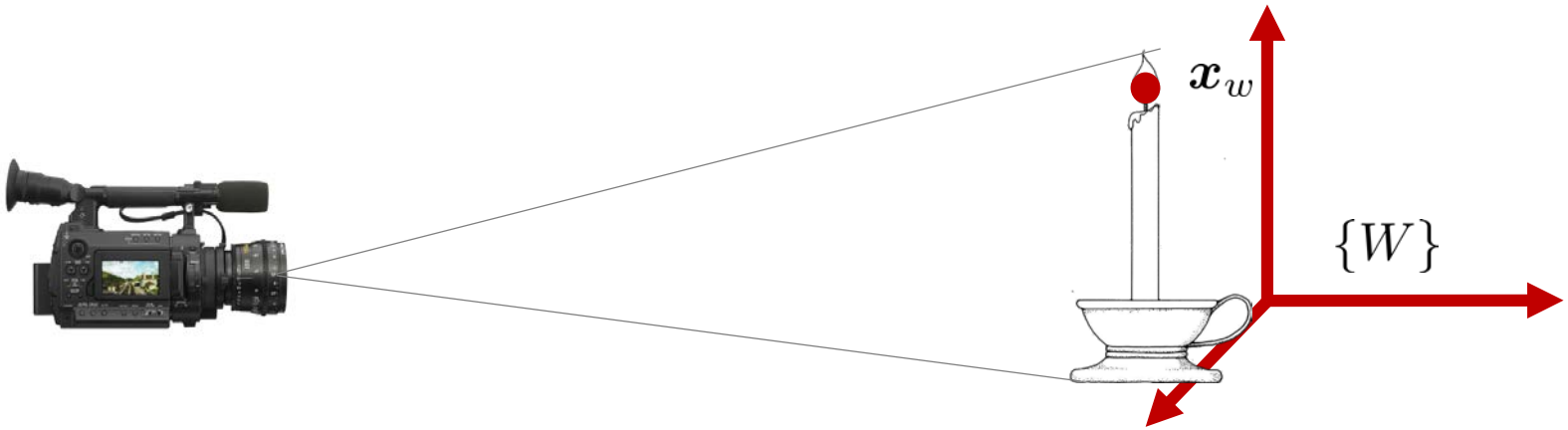
Outline

- **Single view geometry**
  - Pinhole camera model
  - Pixel density
  - Camera intrinsic parameters
- **Vanishing point/lines**
- **Camera calibration**
  - Calibrated by 2D-3D corresponding points
  - Checker-board pattern (Zhengyou Zhang)
- **Image distortion**
  - Radial-Tangential model
  - FOV model
  - Equidistant model
  - Distortion removal
  - Camera calibration with distortion

Pinhole camera model

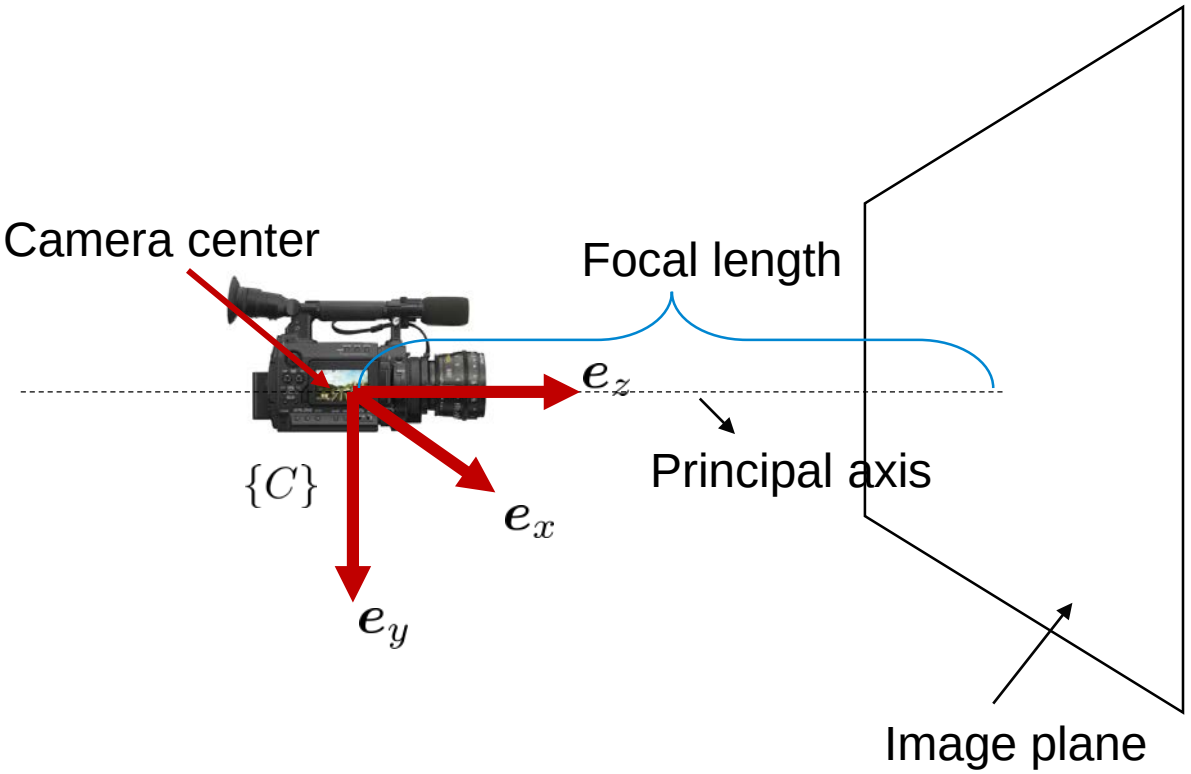
Pinhole camera model

- Given a point $\boldsymbol{x}_w$ in the world coordinate system $\{W\}$, we want to get its image in the camera m .
- First coordinate system – The world coordinate system



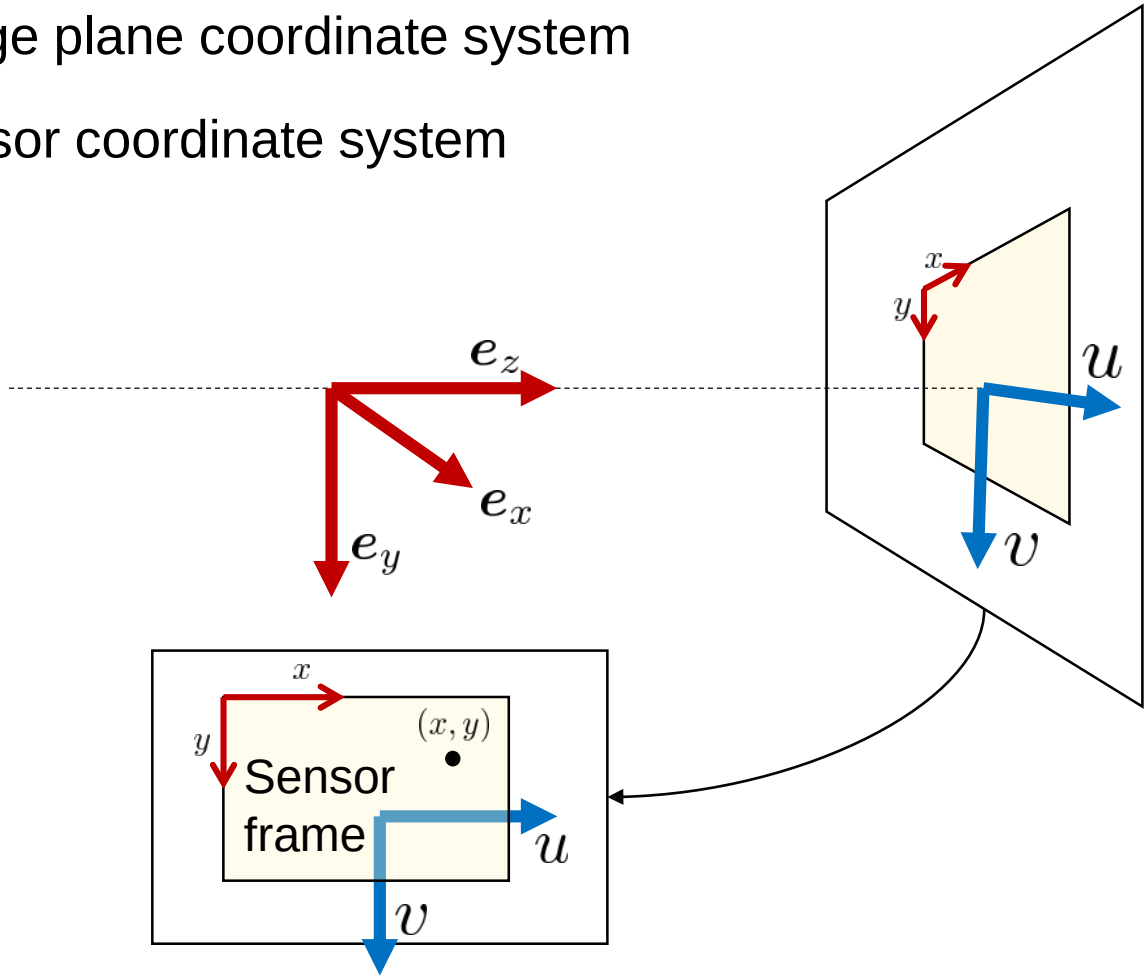
Pinhole camera model

- Camera coordinate system $\{C\}$ - The coordinate system inside of the camera

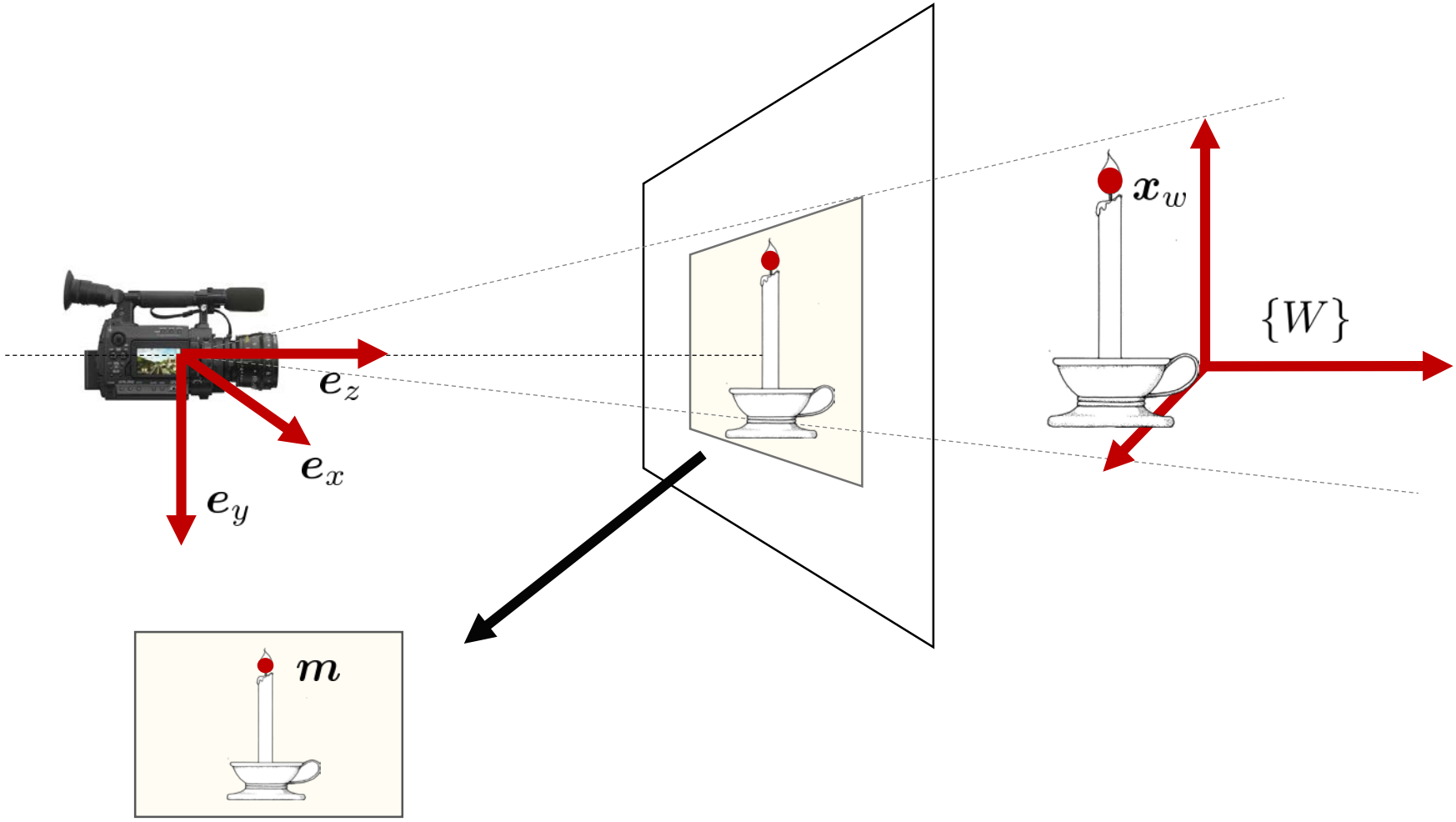


Pinhole camera model

- Image plane coordinate system
- Sensor coordinate system



Pinhole camera model



Pinhole camera model

- The whole process consists of three transformations:
  - Step1. World frame -> Camera frame (3D to 3D)
  - Step2. Camera frame -> Image frame (3D to 2D)
  - Step3. Image frame -> Sensor frame (2D to 2D)

Pinhole camera model

- Step 1. World to camera transformation (Euclidean coordinates)

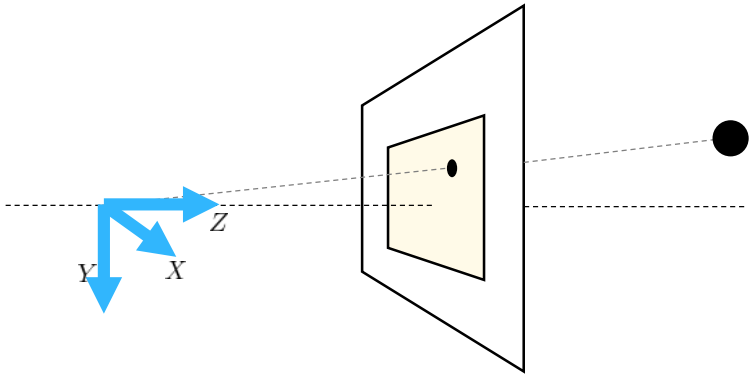
$$\boldsymbol{x}_c = \mathbf{R}\boldsymbol{x}_w + \boldsymbol{t}$$

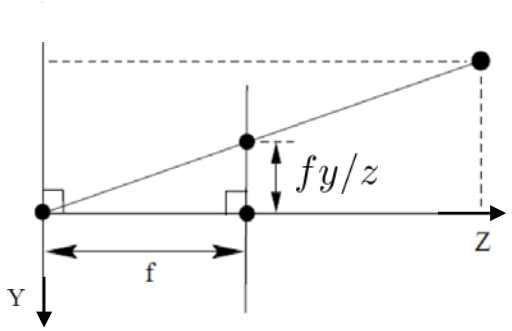
- If we use homogenous coordinates, we have

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

Pinhole camera model

- Step 2. Camera frame to image frame (Perspective projection)





$$\boldsymbol{x}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \Rightarrow \quad \boldsymbol{m}_f = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} fx_c/z_c \\ fy_c/z_c \end{bmatrix} \quad \boldsymbol{m}_f \sim \begin{bmatrix} fx_c \\ fy_c \\ z_c \end{bmatrix}$$

Pinhole camera model

- Using homogenous coordinates, we have

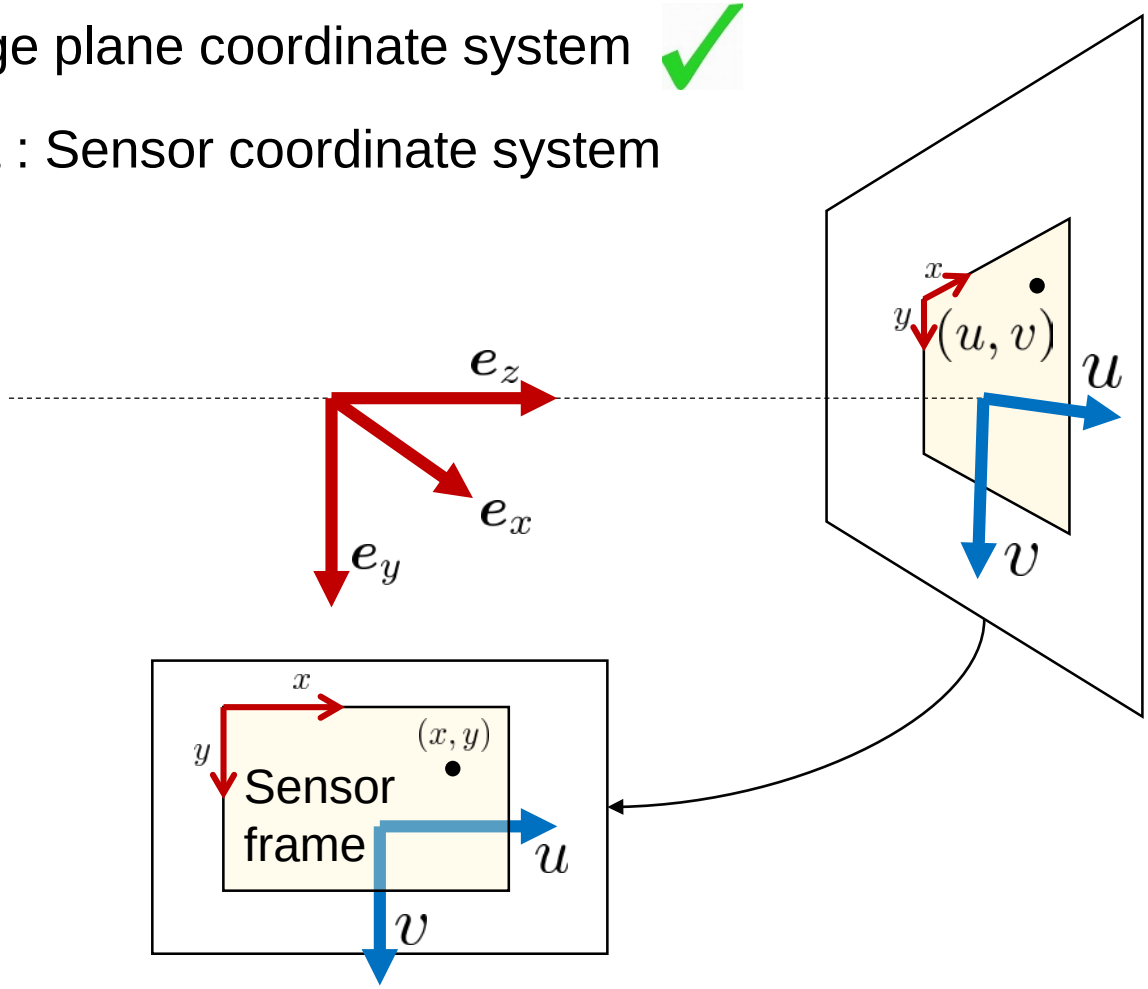
$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \sim \begin{pmatrix} fx_c \\ fy_c \\ z_c \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} x_c \\ y_c \\ z_c \\ 1 \end{pmatrix}$$

image point **3x4 projection matrix** **world point**

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \sim \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

Pinhole camera model

- Image plane coordinate system ✓
- Next : Sensor coordinate system



Pinhole camera model

- Step 3. Image plane -> Sensor frame

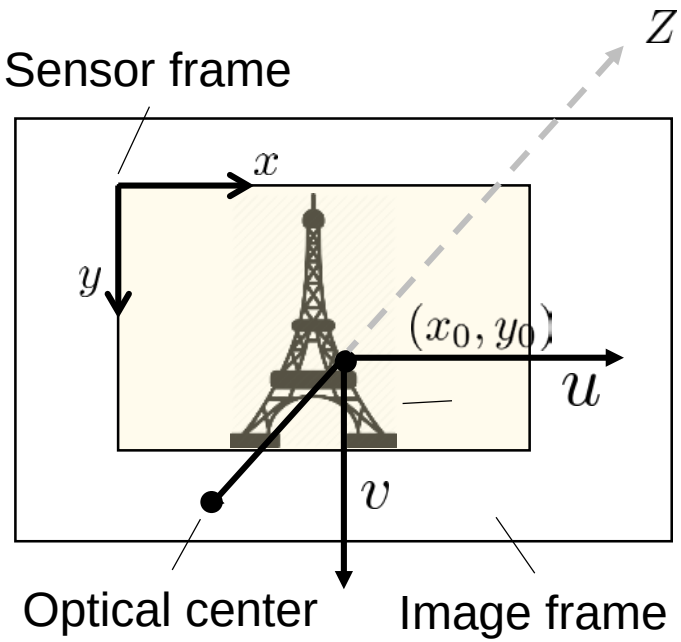
$$x = k_x u + x0$$
$$y = k_y v + y0$$

Pixel density: Pixels/length

Optical center in the picture

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} k_x & 0 & x_0 \\ 0 & k_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

Using homogenous coordinates

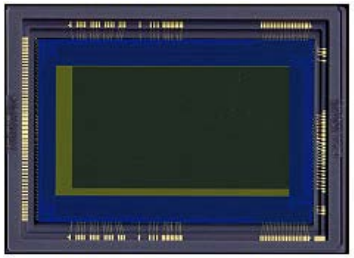


Pinhole camera model

- An example about the pixel density
  - Nikon D610
  - Maximum resolution： 6016 x 4016
  - CMOS size： 35.9mm×24mm

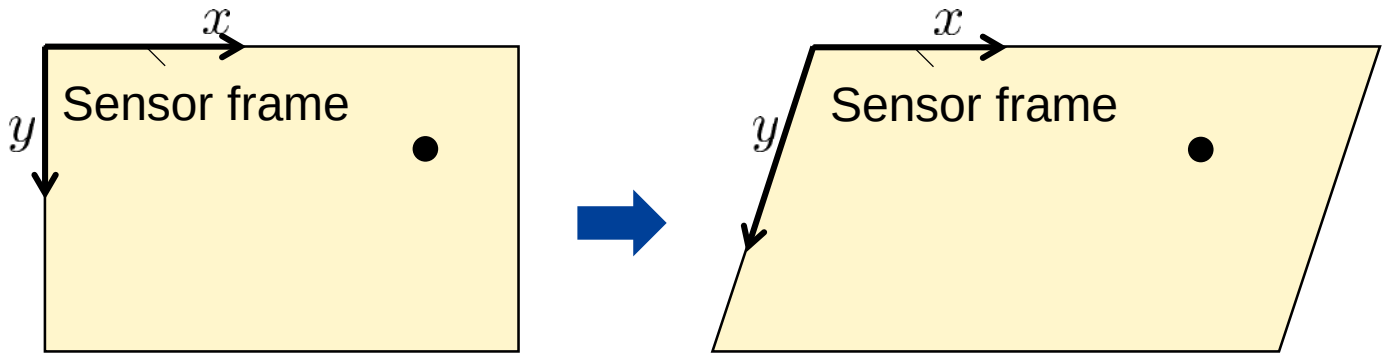
- The pixel density is computed as

$$k_x = 6016/35.9 \approx 167.55 pixel/mm$$
$$k_y = 4016/24.0 \approx 167.33 pixel/mm$$



Camera projection

- Skew effect – The angle between of two image axes is not equal to **90** degrees.



$$\begin{cases} x = k_x u + x_0 \\ y = k_y v + y_0 \end{cases}$$

$$\begin{cases} x = k_x u + sv + x_0 \\ y = k_y v + y_0 \end{cases}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} k_x & 0 & x_0 \\ 0 & k_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} k_x & s & x_0 \\ 0 & k_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

Pinhole camera model

- The whole process

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{bmatrix} k_x & 0 & x_0 \\ 0 & k_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{bmatrix} fk_x & 0 & x_0 \\ 0 & fk_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

Camera intrinsic matrix

Camera pose

Pinhole camera model

- Briefly, we have the following perspective projection model

$$\boldsymbol{m} \sim \mathbf{K}[\mathbf{R} \quad \boldsymbol{t}]\boldsymbol{x}_w$$

- Or

$$\boldsymbol{x} \sim \mathbf{P}\boldsymbol{X} \quad (\text{Let } \boldsymbol{x} = \boldsymbol{m}, \boldsymbol{X} = \boldsymbol{x}_w)$$

Here,

- $\mathbf{K}$ is the camera intrinsic matrix
- $\mathbf{P}$ is the projection matrix

Pinhole camera model

- Perspective projection in Euclidean coordinates

$$\boldsymbol{x} \sim \mathbf{P}\boldsymbol{X}$$

$$x = \frac{\mathbf{P}_1\boldsymbol{X}}{\mathbf{P}_3\boldsymbol{X}} \quad y = \frac{\mathbf{P}_2\boldsymbol{X}}{\mathbf{P}_3\boldsymbol{X}}$$

$$\boldsymbol{x} = \mathcal{P}(\boldsymbol{X})$$

Summary

- The full process of perspective projection of a pinhole camera
  - World frame -> camera frame -> image frame -> sensor frame

$$\boldsymbol{m} \sim \mathbf{K}[\mathbf{R} \quad \boldsymbol{t}]\boldsymbol{x}_w$$

- Camera intrinsic matrix

$$\mathbf{K} = \begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Camera projection matrix

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$$

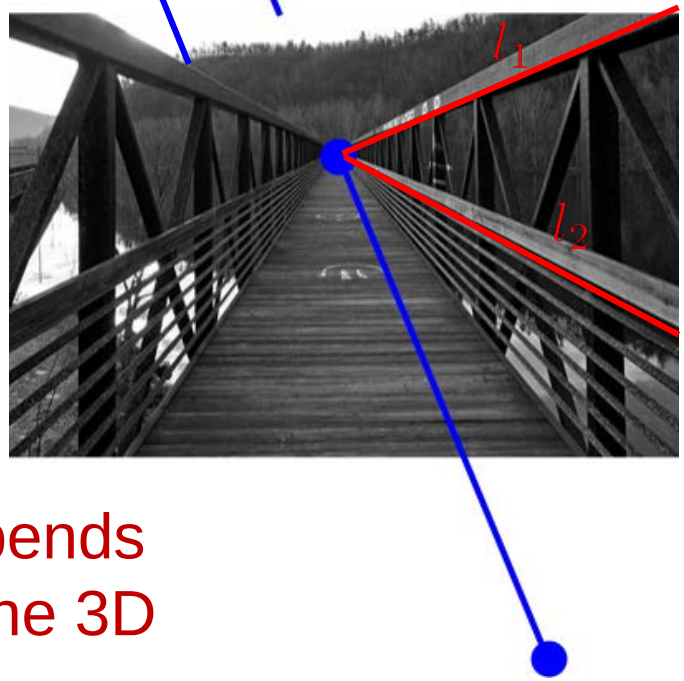
Vanishing Points

Vanishing points

- Vanishing points are the image of points at infinity.

$$\mathbf{X}_\infty = \begin{bmatrix} \mathbf{d} \\ 0 \end{bmatrix}$$

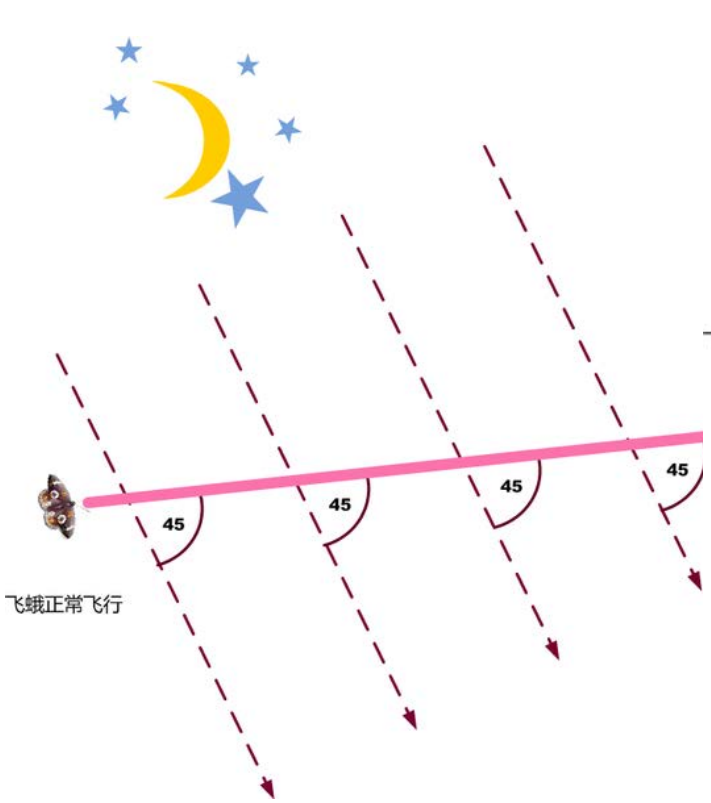
$$\begin{aligned} \mathbf{v} &\sim \mathbf{P}\mathbf{X}_\infty \\ &= \mathbf{K}[\mathbf{R} \mid \mathbf{t}] \begin{bmatrix} \mathbf{d} \\ 0 \end{bmatrix} \\ &= \mathbf{K}\mathbf{R}\mathbf{d} \end{aligned}$$

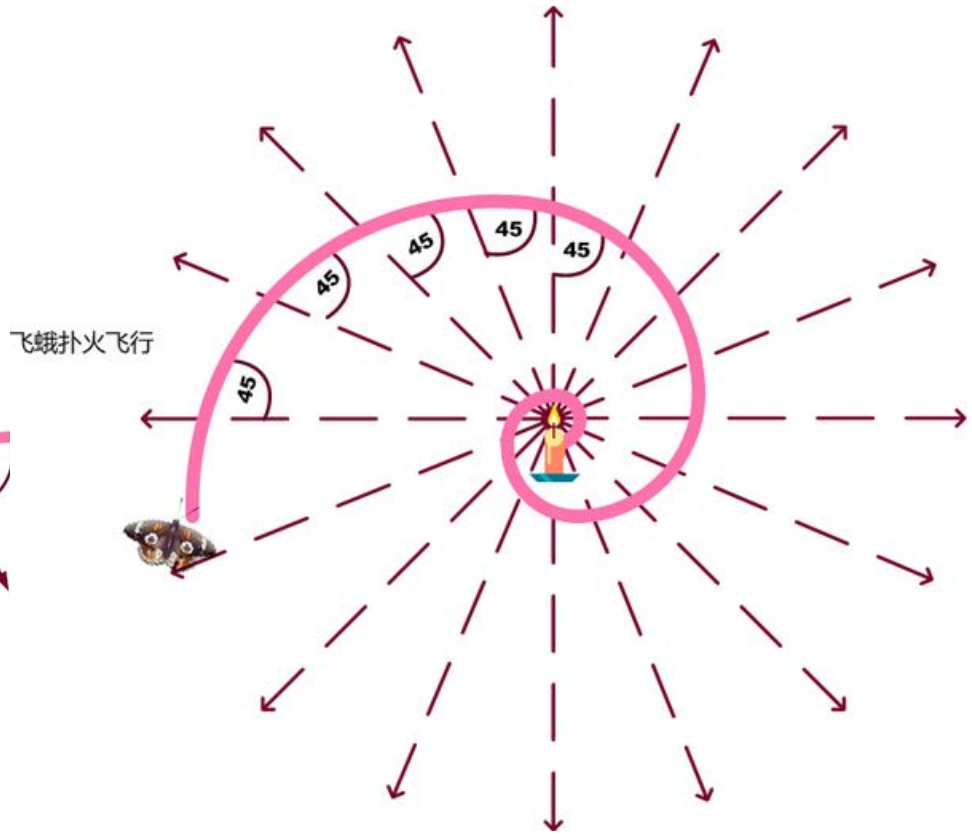


The position of a vanishing point depends only on the camera orientation and the 3D direction of the infinity point.


An example of a vanishing point – the moon

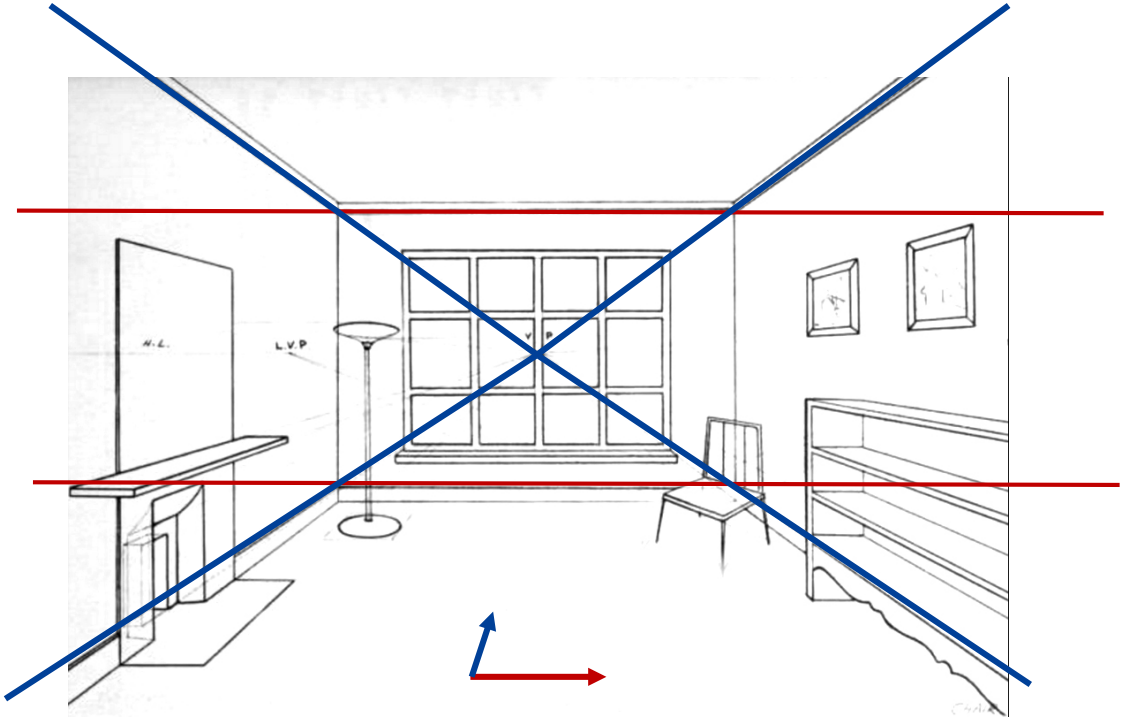
Vanishing points/lines





Vanishing points

- A set of parallel world lines have a common vanishing point.
- If we have two vanishing points, we can use them to compute the camera orientation.



$$\boldsymbol{v}_1 = \mathbf{K}\mathbf{R}\boldsymbol{d}_1$$

$$\boldsymbol{v}_2 = \mathbf{K}\mathbf{R}\boldsymbol{d}_2$$

$$\boldsymbol{v}_1' = \mathbf{R}\begin{bmatrix}1\\0\\0\end{bmatrix} \quad \boldsymbol{v}_2' = \mathbf{R}\begin{bmatrix}0\\1\\0\end{bmatrix}$$

$$\mathbf{R} = \begin{bmatrix}\boldsymbol{v}_1' & \boldsymbol{v}_2' & \boldsymbol{v}_1' \times \boldsymbol{v}_2'\end{bmatrix}$$

Vanishing line

- A vanishing line is the image of a line at infinity.
- Parallel planes in 3D space intersect with a common line at infinity.
- Vanishing line depends only on the orientation of the scene plane; it does not depend on its position.

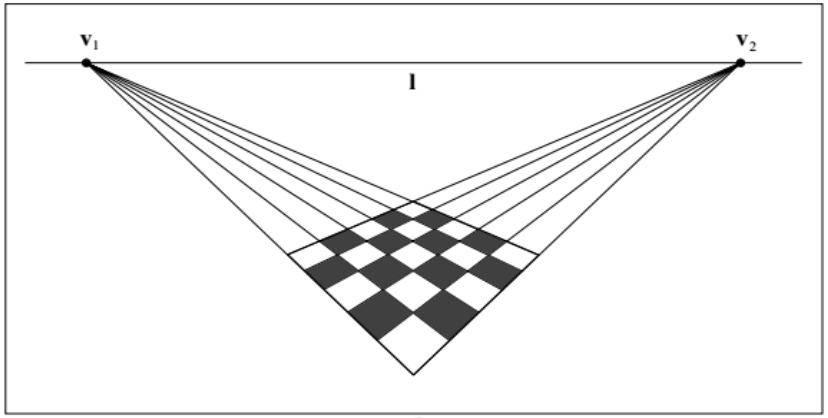
Vanishing line

- A vanishing line can be computed from two vanishing points.

$$\boldsymbol{l} = \boldsymbol{v}_1 \times \boldsymbol{v}_2$$

$$\boldsymbol{l} = \mathbf{K}\mathbf{R}(\boldsymbol{d}_1 \times \boldsymbol{d}_2)$$

$$\boldsymbol{l} = \mathbf{K}\mathbf{R}\boldsymbol{n}$$



- The horizontal line can be used to estimate the vertical direction

$$\boldsymbol{n} = (\mathbf{K}\mathbf{R})^{-1}\boldsymbol{l}$$

Summary

- A vanishing point is the projection of a infinite 3D point on the image

$$\boldsymbol{v} \sim \mathbf{K}\mathbf{R}\boldsymbol{d}$$

- We can use vanishing points to estimate the camera orientation

$$\mathbf{R} = \begin{bmatrix} \boldsymbol{v}_1' & \boldsymbol{v}_2' & \boldsymbol{v}_1' \times \boldsymbol{v}_2' \end{bmatrix}$$

- A vanishing line is the infinite 3D line projected on the image

$$\boldsymbol{l} = \mathbf{K}\mathbf{R}\boldsymbol{n}$$

- We can use horizontal line to estimate the vertical direction

Camera calibration

- How do we get the camera intrinsic parameters?

$$\mathbf{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

- The process of obtaining those intrinsic parameters is called ***camera calibration***.

Camera calibration

- $\mathbf{K}$ is a 3×3 upper triangle matrix, called camera intrinsic matrix (or camera calibration matrix)

$$\mathbf{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

- There are five parameters:
  - **Principle point** (x_0, y_0) , which is point where the optical axis intersects the image plane
  - **Skew parameter** s , model imperfect image axes (usually set to be zero)
  - **Scaling factors** α_x, α_y in image x and y directions

$$\alpha_x = f k_x \quad \Rightarrow \quad f = \alpha_x / k_x$$

Camera calibration

- Camera calibration by 2D-3D point correspondences:
  - Step 1. solve the camera projection matrix
  - Step 2. Decomposition of the camera matrix to extract intrinsic matrix

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}] \in \mathbb{R}^{3\times 4}$$

Camera pose

Intrinsic parameters

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}$$

Camera calibration

- Suppose we have n point correspondences between 2D image and 3D space,

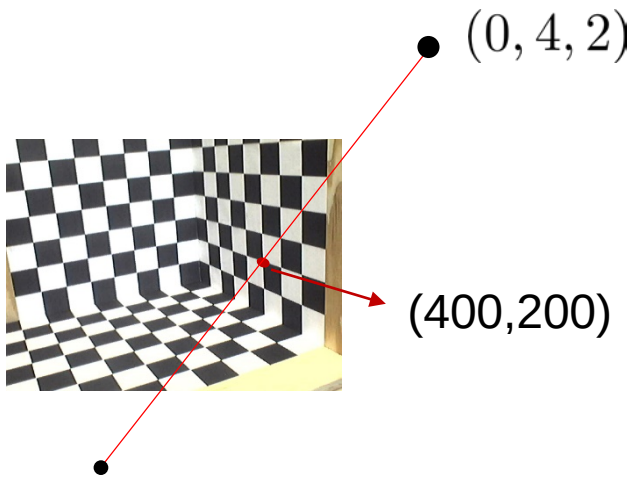
$$\boldsymbol{X}_i \leftrightarrow \boldsymbol{x}_i, i = 1, \ldots, n$$

- We can solve the camera projection matrix that satisfies

$$\boldsymbol{x}_i \sim \mathbf{P}\boldsymbol{X}_i$$

Camera calibration

- For example, a calibration cube
  - We know the 3D coordinates of each corner
  - We can extract its image coordinates in the picture



$$\boldsymbol{x}_i \sim \mathbf{P}\boldsymbol{X}_i$$

$$\begin{bmatrix} 0 \\ 4 \\ 2 \\ 1 \end{bmatrix} \qquad \begin{bmatrix} 400 \\ 200 \\ 1 \end{bmatrix}$$

Camera calibration

- Using Euclidean coordinates, we can obtain from $\mathbf{x}_i \sim \mathbf{P}\mathbf{X}_i$

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} \mathbf{P}_1^{\mathrm{T}}\boldsymbol{X}_i / \mathbf{P}_3^{\mathrm{T}}\boldsymbol{X}_i \\ \mathbf{P}_2^{\mathrm{T}}\boldsymbol{X}_i / \mathbf{P}_3^{\mathrm{T}}\boldsymbol{X}_i \end{bmatrix}$$

$$\Rightarrow \quad \begin{bmatrix} \mathbf{X}_i^{\mathrm{T}} & \mathbf{0}^{\mathrm{T}} & -x_i\boldsymbol{X}_i^{\mathrm{T}} \\ \mathbf{0}^{\mathrm{T}} & \boldsymbol{X}_i^{\mathrm{T}} & -y_i\boldsymbol{X}_i^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

- One point correspondence gives rise to two equations

Camera calibration

- Putting all the point correspondences together, we have

$$\begin{bmatrix} \boldsymbol{X}_1^{\mathrm{T}} & \mathbf{0}^{\mathrm{T}} & -x_1\boldsymbol{X}_1^{\mathrm{T}} \\ \mathbf{0}^{\mathrm{T}} & \boldsymbol{X}_1^{\mathrm{T}} & -y_1\boldsymbol{X}_1^{\mathrm{T}} \\ \cdots & \cdots & \cdots \\ \boldsymbol{X}_n^{\mathrm{T}} & \mathbf{0}^{\mathrm{T}} & -x_n\boldsymbol{X}_n^{\mathrm{T}} \\ \mathbf{0}^{\mathrm{T}} & \boldsymbol{X}_n^{\mathrm{T}} & -y_n\boldsymbol{X}_n^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix} = \mathbf{0}$$

$$\mathbf{A} \in \mathbb{R}^{2n \times 12}$$

- At least 6 point correspondences are required to solve this equation

  (Homogenous linear system again that can be solved by SVD)

Camera calibration

- After that, we decompose the camera projection matrix

$$\mathbf{P} \rightarrow [\mathbf{Q}|\mathbf{p}_4] = [\mathbf{KR}|\mathbf{Kt}]$$

- We can use RQ decomposition to decompose

$$\mathbf{Q} = \mathbf{KR}$$

$$\mathrm{P} = \begin{bmatrix} 3.53553\,e{+}2 & 3.39645\,e{+}2 & 2.77744\,e{+}2 & -1.44946\,e{+}6 \\ -1.03528\,e{+}2 & 2.33212\,e{+}1 & 4.59607\,e{+}2 & -6.32525\,e{+}5 \\ 7.07107\,e{-}1 & -3.53553\,e{-}1 & 6.12372\,e{-}1 & -9.18559\,e{+}2 \end{bmatrix}$$

$$\begin{bmatrix} 468.2 & 91.2 & 300.0 \\ & 427.2 & 200.0 \\ & & 1.0 \end{bmatrix} \begin{bmatrix} 0.41380 & 0.90915 & 0.04708 \\ -0.57338 & 0.22011 & 0.78917 \\ 0.70711 & -0.35355 & 0.61237 \end{bmatrix}$$

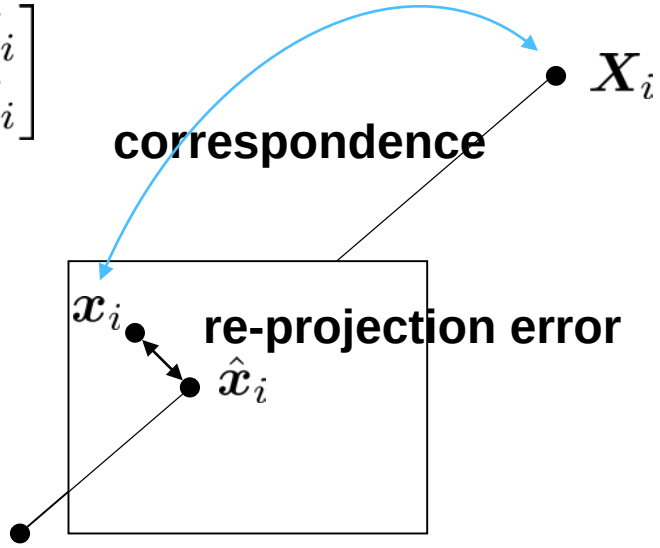
Camera calibration

- After the above steps, usually a refinement step is executed by minimizing the re-projection error :

$$f(\mathbf{R}, \boldsymbol{t}, \mathbf{K}) = \sum_i (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2$$

$$\begin{bmatrix} \hat{x}_i \\ \hat{y}_i \end{bmatrix} = \begin{bmatrix} \mathbf{P}_1^{\mathrm{T}} \boldsymbol{X}_i / \mathbf{P}_3^{\mathrm{T}} \boldsymbol{X}_i \\ \mathbf{P}_2^{\mathrm{T}} \boldsymbol{X}_i / \mathbf{P}_3^{\mathrm{T}} \boldsymbol{X}_i \end{bmatrix}$$

- This is a non-linear least square problem

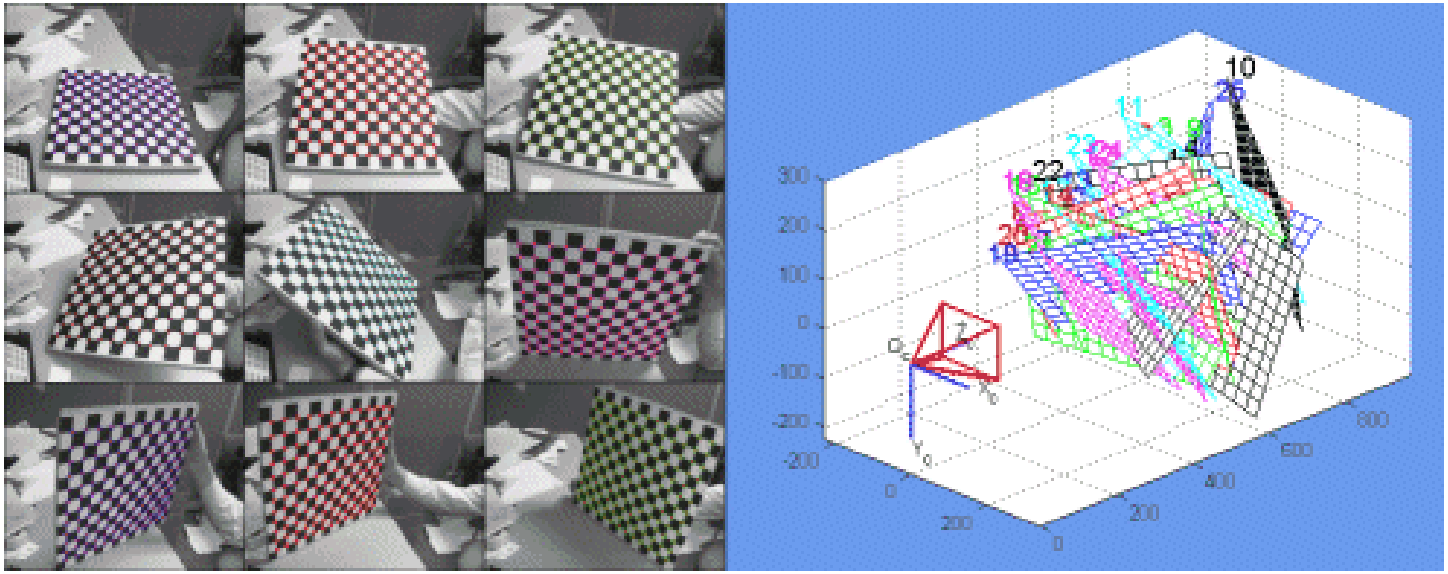


Summary

- Camera calibration is to seek the camera intrinsic parameters
- A 2D-3D calibration is introduced
  - Step 1. Compute the camera projection matrix
  - Step 2. Decompose the camera projection matrix to get the intrinsic matrix
  - Step 3. refine the intrinsic matrix and camera poses by minimizing the re-projection error

Camera calibration

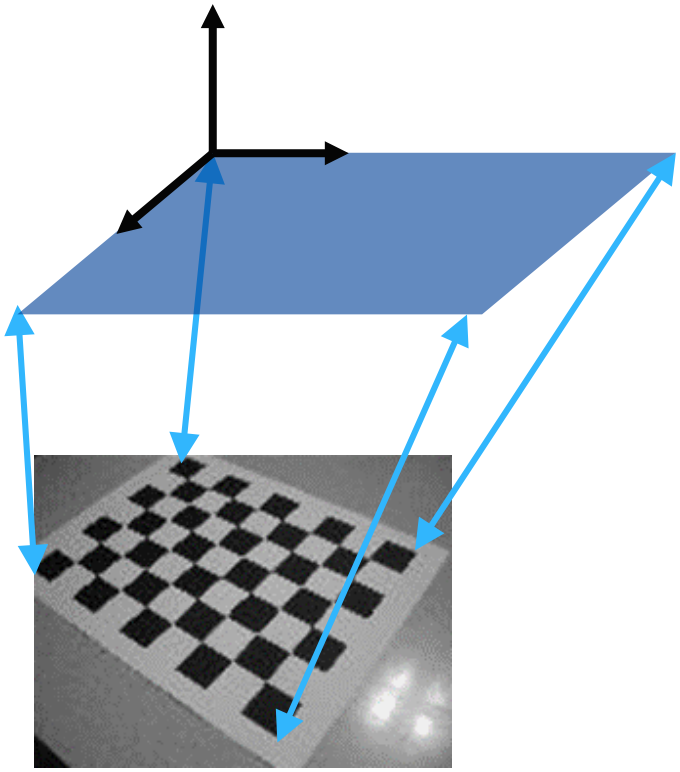
- Method II : Checkerboard approach
  - Use several snapshots of a checkerboard pattern to compute the intrinsic parameters.



Zhang, Zhengyou. "A flexible new technique for camera calibration." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22.11 (2000): 1330-1334.

Camera calibration

- World coordinate system on the plane



$$\mathbf{x} \sim \mathbf{K}[\mathbf{R}\ \mathbf{t}]\mathbf{X} = \mathbf{K}[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{r}_3\ \mathbf{t}] \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix}$$

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

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

Camera calibration

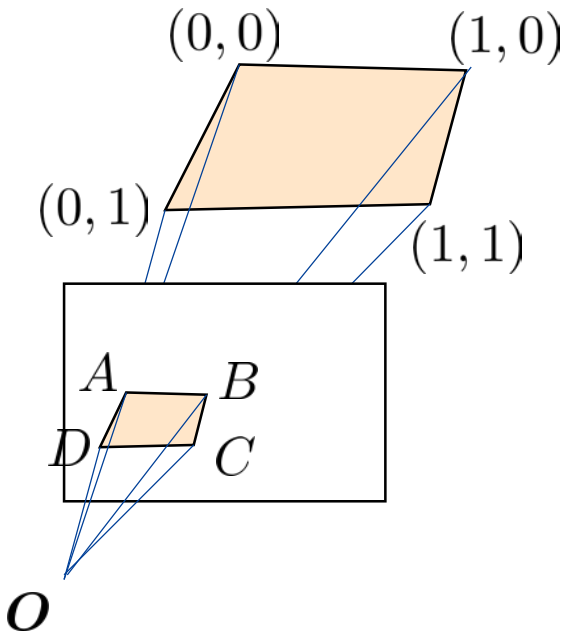
$$\mathbf{H} = \mathbf{K}[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{t}] \Rightarrow \begin{cases} \mathbf{h}_1 = \mathbf{K}\mathbf{r}_1 \\ \mathbf{h}_2 = \mathbf{K}\mathbf{r}_2 \end{cases}$$

$$\mathbf{r}_1 \perp \mathbf{r}_2 \text{ and } \|\mathbf{r}_1\| = \|\mathbf{r}_2\| = 1$$
$$\text{Let } \omega = \mathbf{K}^{-T}\mathbf{K}^{-1}$$
$$\Rightarrow \begin{cases} \mathbf{h}_1^T \omega \mathbf{h}_2 = 0 \\ \mathbf{h}_1^T \omega \mathbf{h}_1 = \mathbf{h}_2 \omega \mathbf{h}_2 \end{cases}$$

$$\omega = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_2 & a_4 & a_5 \\ a_3 & a_5 & a_6 \end{bmatrix} \Rightarrow \mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} \Rightarrow \begin{cases} \mathbf{V}_{12}^T \mathbf{a} = 0 \\ (\mathbf{V}_{11} - \mathbf{V}_{22})^T \mathbf{a} = 0 \end{cases}$$

Camera calibration

- Step 1. For each view, we compute the homography from the corresponding corners



$$(0,0) \leftrightarrow A$$
$$(1,0) \leftrightarrow B$$
$$(1,1) \leftrightarrow C$$
$$(0,1) \leftrightarrow D$$

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

- Step 2. construct two equations from a single

$$\begin{cases} \mathbf{h}_1^T \omega \mathbf{h}_2 = 0 \\ \mathbf{h}_1^T \omega \mathbf{h}_1 = \mathbf{h}_2 \omega \mathbf{h}_2 \end{cases}$$

Camera calibration

- Step 3. After at least three views, we can get the following system of linear equations:

$$\omega = \mathbf{K}^{-\mathrm{T}}\mathbf{K}^{-1} \rightarrow \boldsymbol{a}$$

$$\begin{bmatrix} \mathbf{V}_{12}^{(1)T} \\ (\mathbf{V}_{11}^{(1)} - \mathbf{V}_{22}^{(1)})^T \\ \cdots \\ \mathbf{V}_{12}^{(n)T} \\ (\mathbf{V}_{11}^{(n)} - \mathbf{V}_{22}^{(n)})^T \end{bmatrix} \boldsymbol{a} = \mathbf{0}$$

- This is a system of over-constrained homogeneous equations, which can be solved by SVD again. (The right vector corresponding to the smallest singular value)

Camera calibration

- Step 4. $\mathbf{K}$ can be obtained by Cholesky decomposition of

$$\omega = \mathbf{K}^{-T}\mathbf{K}^{-1}$$

- Cholesky decomposition: $\mathbf{A} = \mathbf{L}\mathbf{L}^{T}$
- Example:

$$\begin{pmatrix} 4 & 12 & -16 \\ 12 & 37 & -43 \\ -16 & -43 & 98 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 \\ 6 & 1 & 0 \\ -8 & 5 & 3 \end{pmatrix} \begin{pmatrix} 2 & 6 & -8 \\ 0 & 1 & 5 \\ 0 & 0 & 3 \end{pmatrix}.$$

ω

Camera calibration

- Step 5. Extraction of extrinsic parameters (camera poses of different views)

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

$$\boldsymbol{r}_1 = \mathbf{K}^{-1}\mathbf{H}(:,1)$$

$$\boldsymbol{r}_2 = \mathbf{K}^{-1}\mathbf{H}(:,2)$$

$$\Rightarrow \quad \boldsymbol{r}_3 = \boldsymbol{r}_1 \times \boldsymbol{r}_2$$

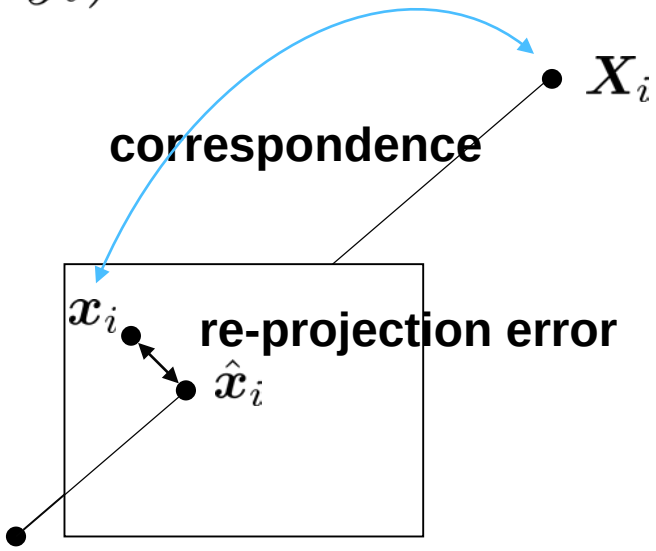
$$\boldsymbol{t} = \mathbf{K}^{-1}\mathbf{H}(:,3)$$

Camera calibration

- Step 6. Refinement by minimizing the re-projection errors

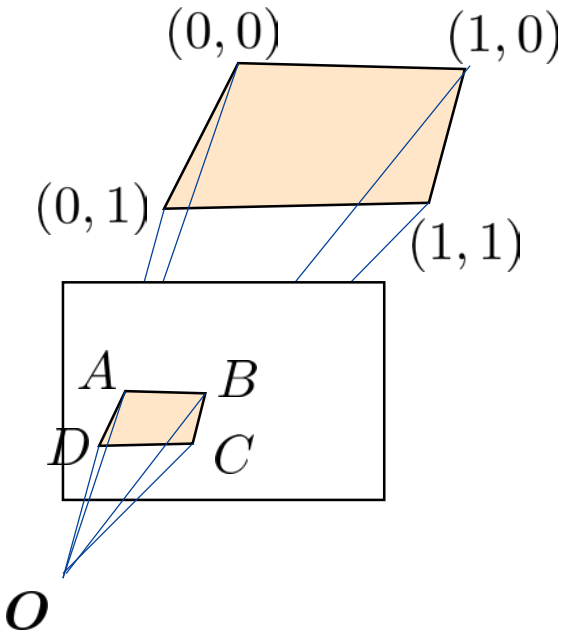
$$f(\mathbf{R}_1, \boldsymbol{t}_1, \ldots, \mathbf{R}_n, \boldsymbol{t}_n, \mathbf{K}) =$$

$$\sum_i (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2$$



Pose estimation of a planar object

- If we know the exact coordinates of the feature points on the planar object, we can estimate the pose of this planar object from the homography matrix.



$$(0,0) \leftrightarrow A$$
$$(1,0) \leftrightarrow B$$
$$(1,1) \leftrightarrow C$$
$$(0,1) \leftrightarrow D$$

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

$$\boldsymbol{r}_1 = \mathbf{K}^{-1}\mathbf{H}(:,1)$$
$$\boldsymbol{r}_2 = \mathbf{K}^{-1}\mathbf{H}(:,2)$$
$$\boldsymbol{r}_3 = \boldsymbol{r}_1 \times \boldsymbol{r}_2$$

Summary

- Two calibration approaches:
  - Camera calibration by decomposition of a camera matrix

$$\mathbf{P} \rightarrow [\mathbf{Q}|\mathbf{p}_4] = [\mathbf{KR}|\mathbf{Kt}]$$

  - Camera calibration by a planar pattern

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

Image distortion

- Due to the imperfect imaging system, images are usually distorted.

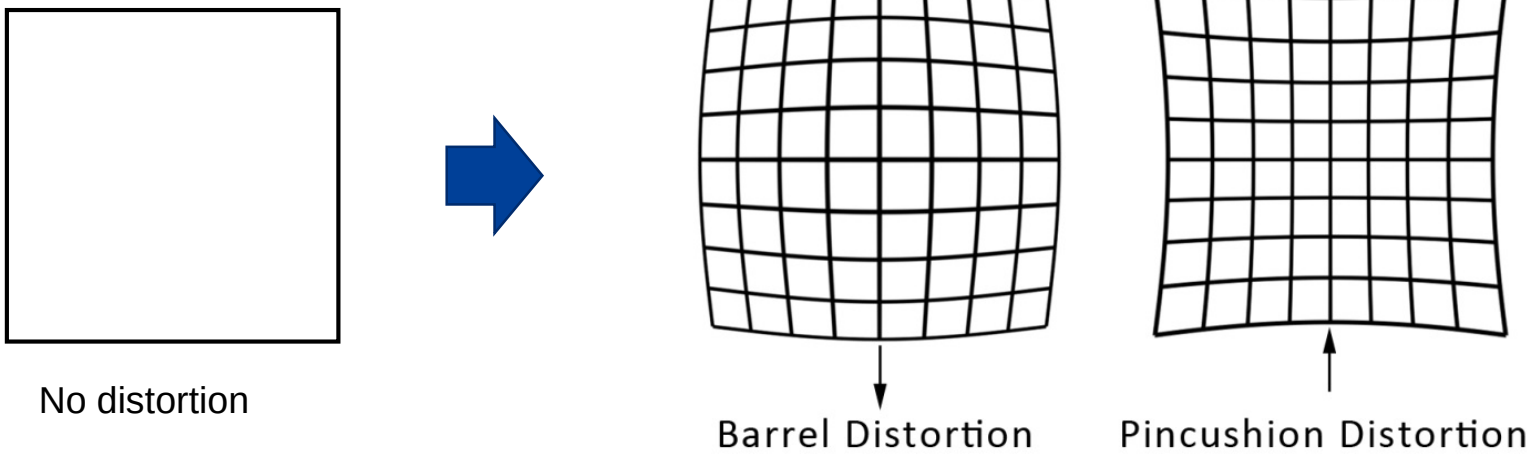


Image distortion

- Model I – (Radial-Tangential model)

$$\begin{pmatrix} u_d \\ v_d \end{pmatrix} = \underbrace{(1 + D_1 r^2 + D_2 r^4 + D_5 r^6)\begin{pmatrix} u \\ v \end{pmatrix}}_{\text{Radial distortion}} + \underbrace{\begin{pmatrix} 2D_3 uv + D_4(r^2 + 2u^2) \\ D_3(r^2 + 2v^2) + 2D_4 uv \end{pmatrix}}_{\text{Tangential distortion}}$$

Distortion coefficients: $D = [D_1, D_2, D_3, D_4, D_5]$

OpenCV, Matlab, Caltech camera calibration toolbox

Heikkila, Janne, and Olli Silven. "A four-step camera calibration procedure with implicit image correction." *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*. IEEE, 1997.

Image distortion

- Model II – PTAM distortion model (FOV model)

$$\begin{bmatrix} u_d \\ v_d \end{bmatrix} = \frac{1}{r_u \omega} \arctan\left(2 r_u \tan\left(\frac{\omega}{2}\right)\right) \begin{bmatrix} f_x \frac{x}{z} \\ f_y \frac{y}{z} \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix}$$

$$\left(r_u := \sqrt{\left(\frac{x}{z}\right)^2 + \left(\frac{y}{z}\right)^2}\right)$$

A close form of inverse (distortion removal)

$$\begin{bmatrix} \tilde{u}_d \\ \tilde{v}_d \end{bmatrix} = \begin{bmatrix} (u_d - c_x) f_x^{-1} \\ (v_d - c_y) f_y^{-1} \end{bmatrix} \Rightarrow \begin{bmatrix} \tilde{u}_u \\ \tilde{v}_u \end{bmatrix} = \frac{\tan(r_d \omega)}{2 r_d \tan\frac{\omega}{2}} \begin{bmatrix} \tilde{u}_d \\ \tilde{v}_d \end{bmatrix}$$

$$r_d := \sqrt{\tilde{u}_d^2 + \tilde{v}_d^2}$$

Image distortion

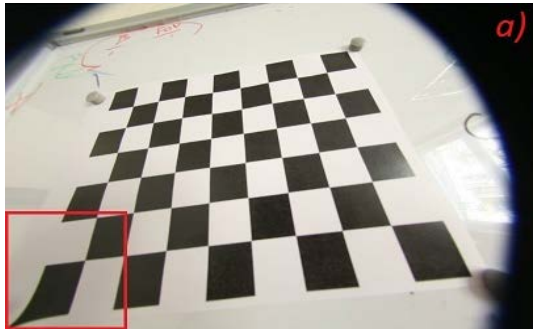
- Model III (Equidistant model)

Step 1: $\theta_d = \theta(1 + k_1\theta^2 + k_2\theta^4 + k_3\theta^6 + k_4\theta^8)$

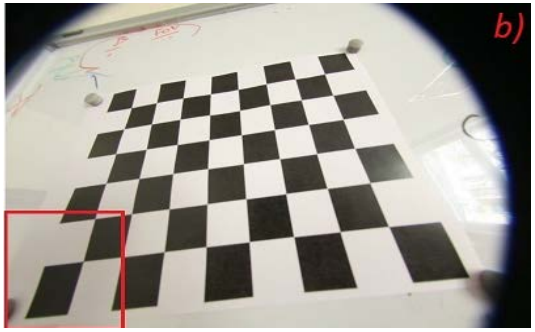
Step 2: $u' = (\theta_d / r)u$

$v' = (\theta_d / r)v$

where $r = \sqrt{u^2 + v^2}$ $\theta = atan(r)$



Radial-Tangential Model

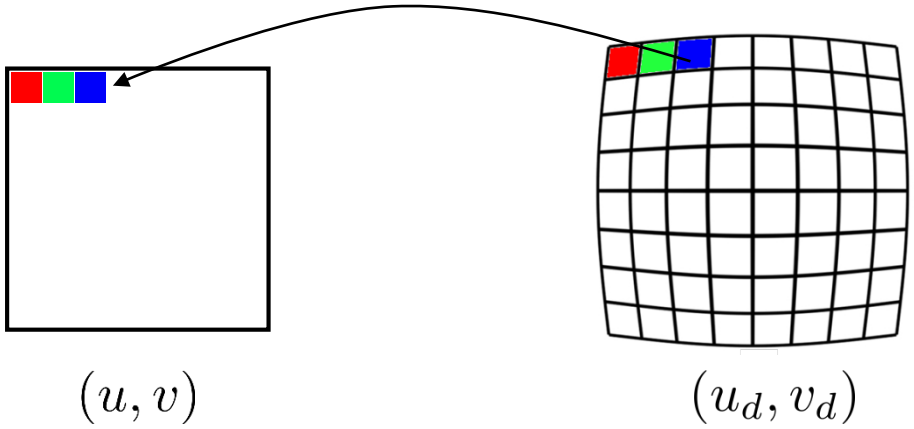


Equidistant model

Kannala, Juho, and Sami S. Brandt. "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses." *IEEE transactions on pattern analysis and machine intelligence* 28.8 (2006): 1335-1340.

Remove the distortion

- Rectify the distorted image:
  - For each pixel in the destination image (without distortion), find its corresponding pixel in the distorted image.
  - Fill the color of the corresponding pixel in the distorted image into the current pixel.
  - Repeat above steps until all pixels are filled.



Remove the distortion

- Compute the original coordinate from the distorted coordinate :

$$(u, v) \leftarrow (u_d, v_d)$$

  - Directly solve (u, v) from (u_d, v_d) is very difficult!

$$\begin{pmatrix} u_d \\ v_d \end{pmatrix} = \underbrace{(1 + D_1 r^2 + D_2 r^4 + D_5 r^6)}_{\tau} \begin{pmatrix} u \\ v \end{pmatrix} + \underbrace{\begin{pmatrix} 2D_3 uv + D_4(r^2 + 2u^2) \\ D_3(r^2 + 2v^2) + 2D_4 uv \end{pmatrix}}_{d\mathbf{x}}$$

  - We can solve it iteratively :
    - Initially we let $(u_1, v_1) \leftarrow (u_d, v_d)$
    - Repeat until convergence :
      - Compute $\tau, d\mathbf{x}$ using (u_{i-1}, v_{i-1})
      - We get (u_i, v_i) by

$$\begin{pmatrix} u_i \\ v_i \end{pmatrix} = \left(\begin{pmatrix} u_d \\ v_d \end{pmatrix} - d\mathbf{x} \right) / \tau$$

Usually, convergence can be achieved in 3~5 iterations

Summary

- A little distortion :
  - Radial-Tangential model
- Fisheye camera
  - FOV model
  - Equidistant model
- Remove the distortion
  - Undistort the image
  - Undistort the coordinates
- It depends on the cameras (fisheye or not) and the applications (measurements or visual navigation) to choose the proper model.