香港科技大学(广州)
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

信息枢纽
INFORMATION HUB
人工智能学域
ARTIFICIAL INTELLIGENCE

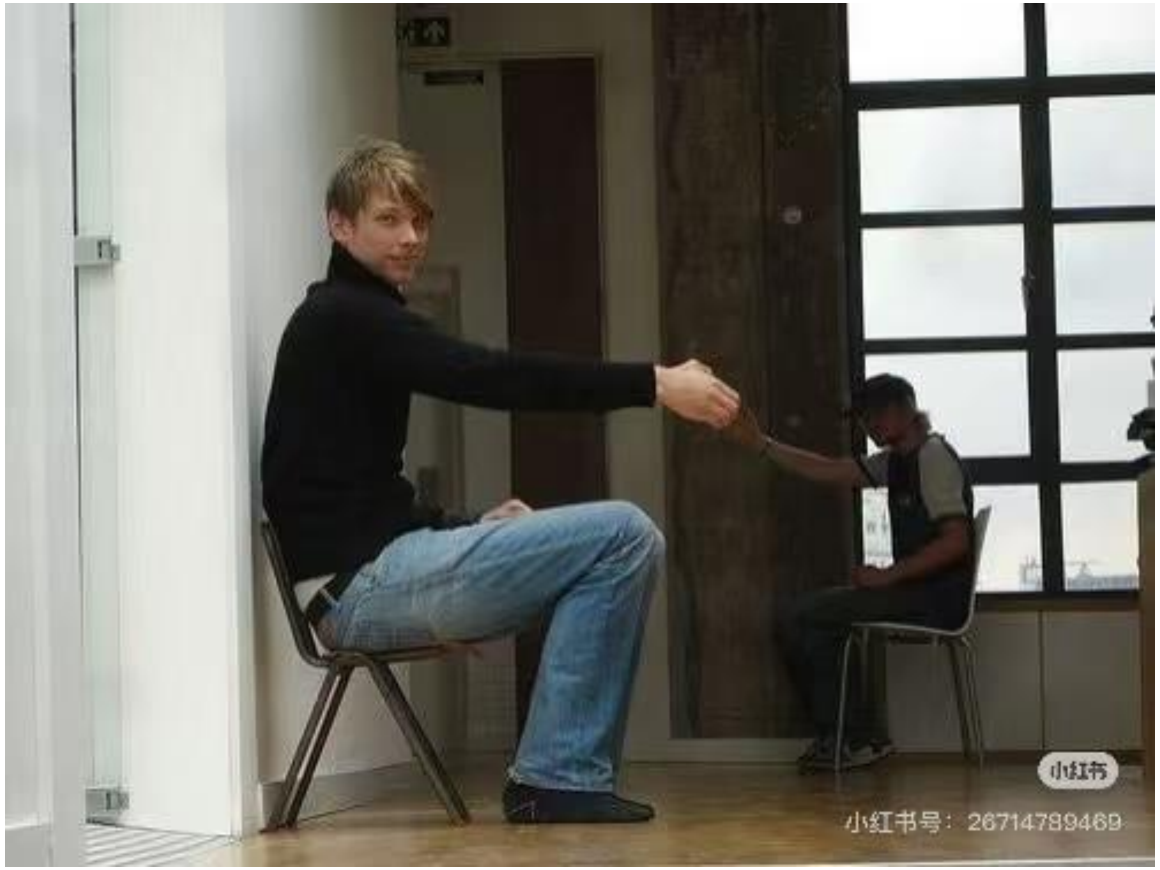
Camera Model

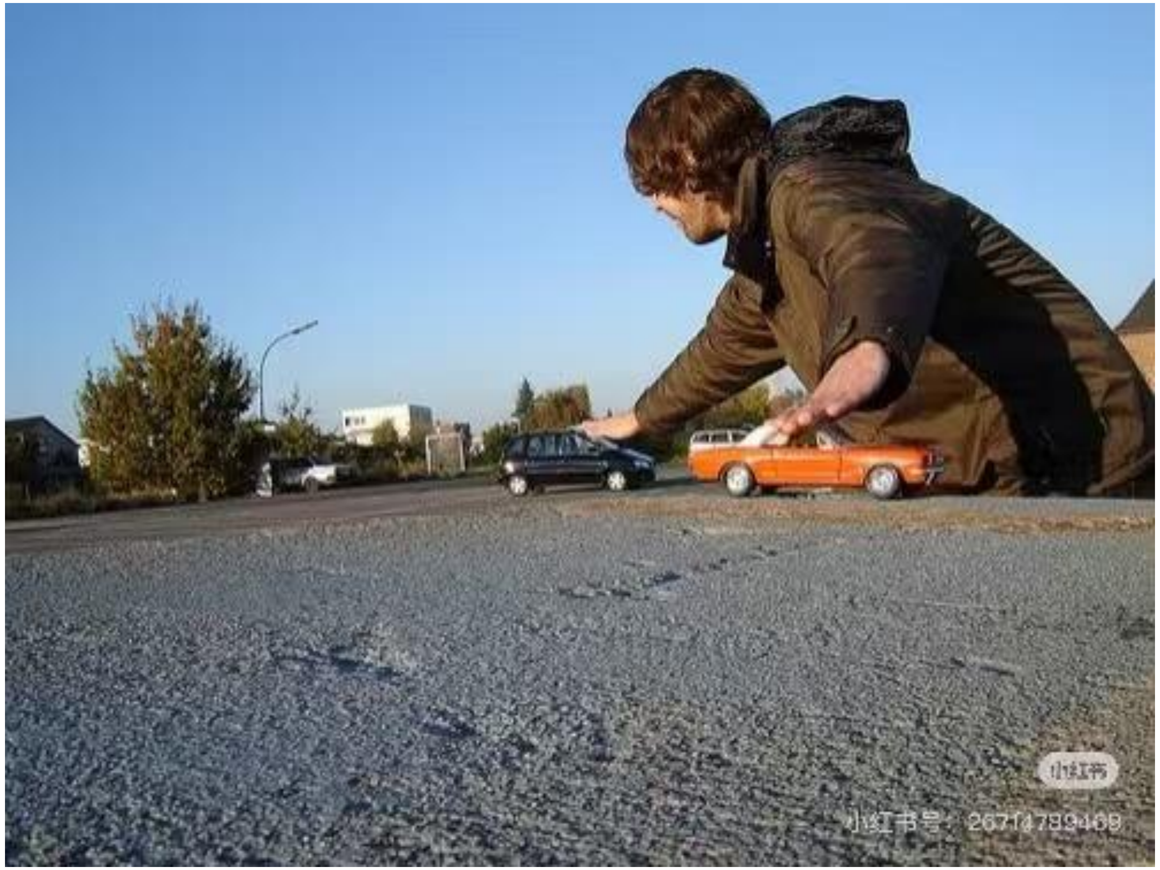
Course AIAA 4220

Week 2 - Lecture 4

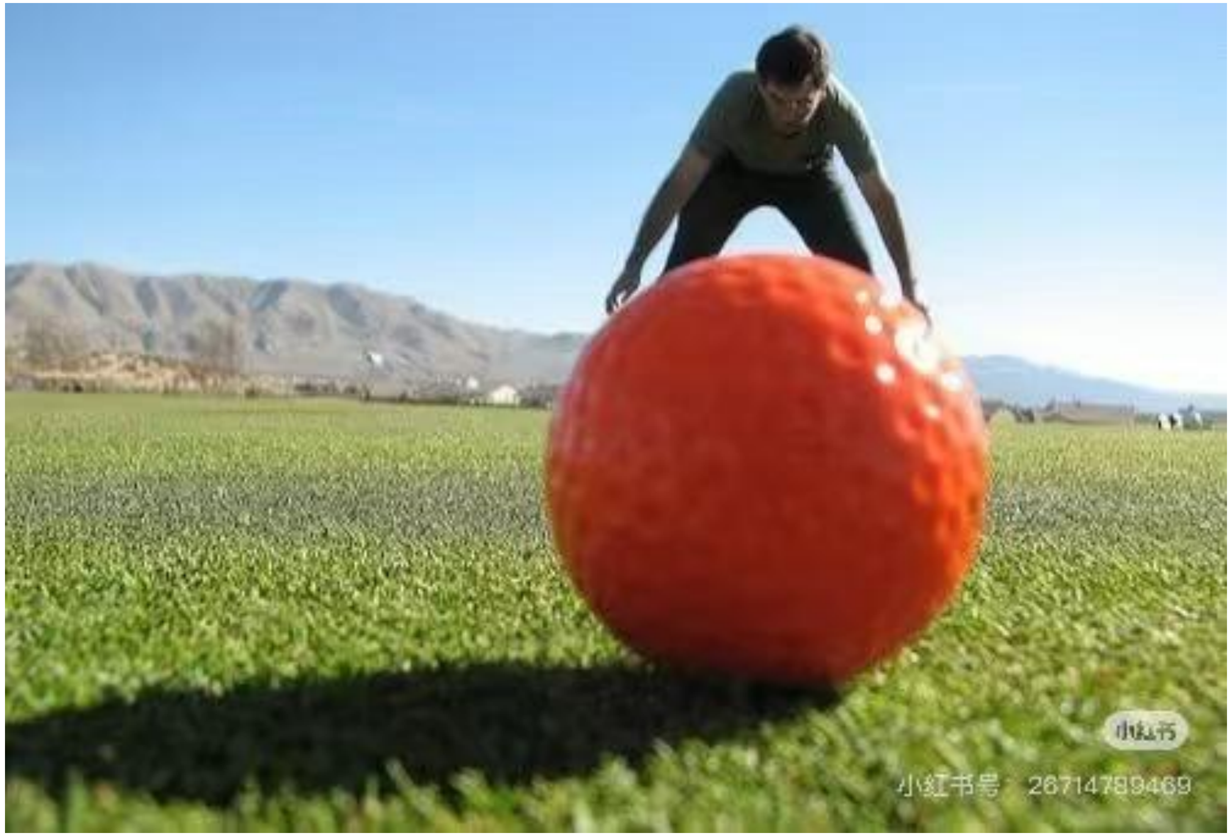
Changhao Chen
Assistant Professor
HKUST (GZ)

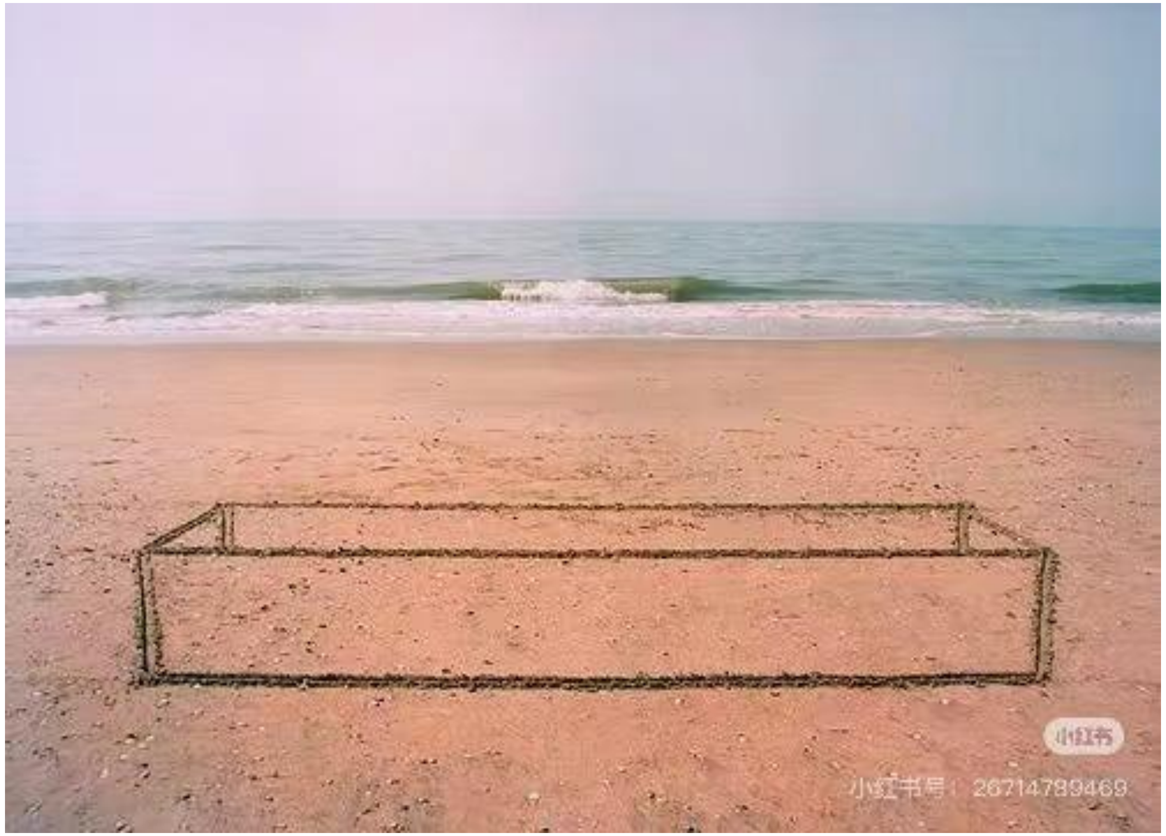
Some Photos





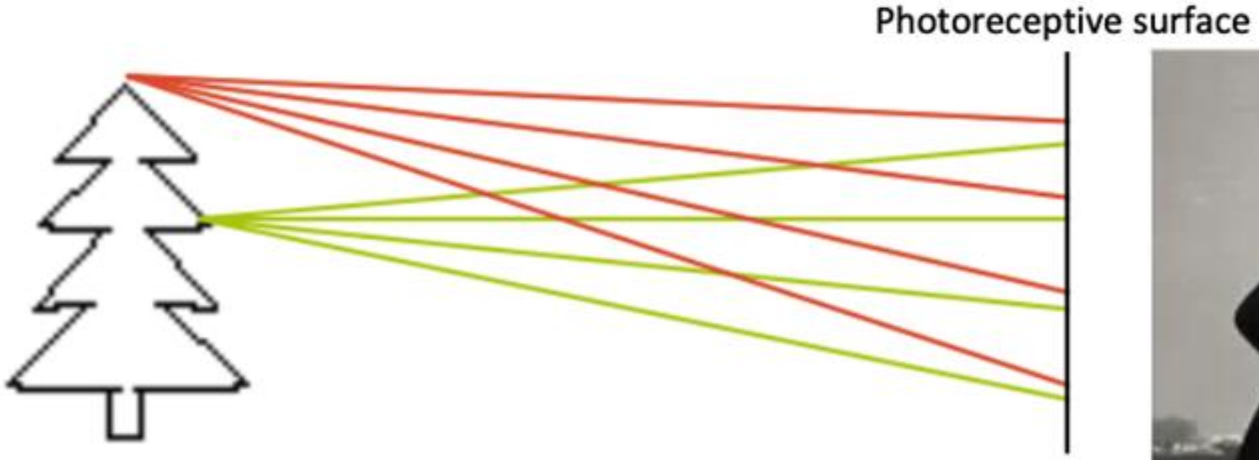
Some Photos

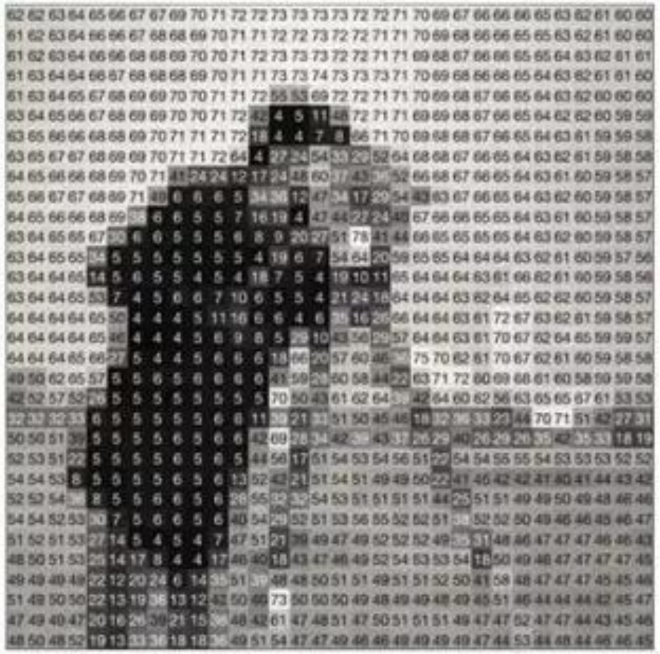




Cameras

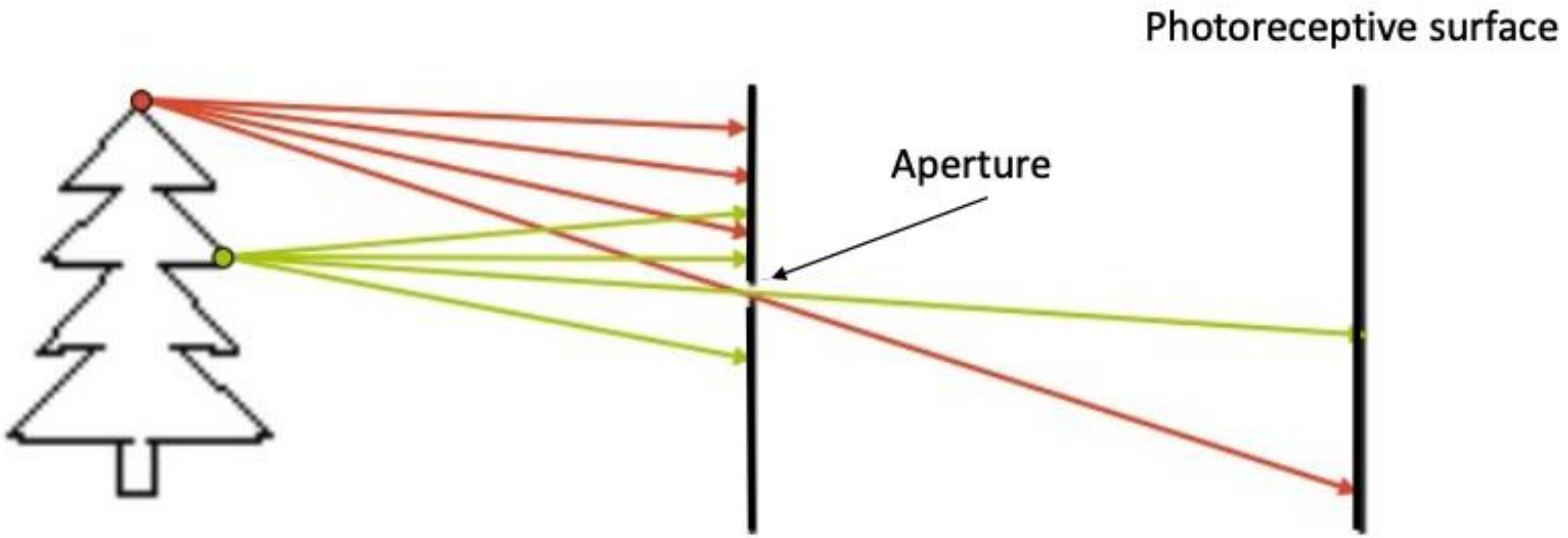
- How to capture an image of the world
  - (Digital) Cameras -> capture light -> converted to digital image
  - Light is reflected by the object and scattered in all directions
  - if we simply add a photoreceptive surface, the captured image will be extremely blurred





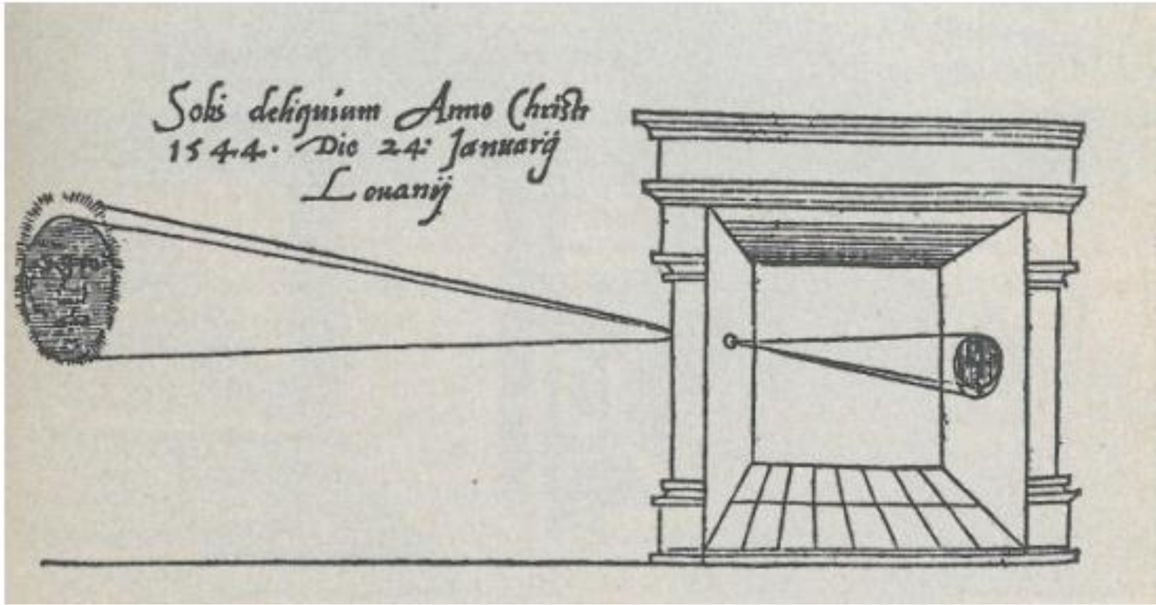
Pinhole Camera

- Place an opaque barrier with a tiny aperture (a pinhole) between the scene and the sensor.
- This barrier blocks most light rays, allowing only a narrow ray bundle from each point in the scene to pass through the pinhole.
- Pinhole camera: a camera without a lens but with a tiny aperture, a pinhole
- The light is spread on the surface

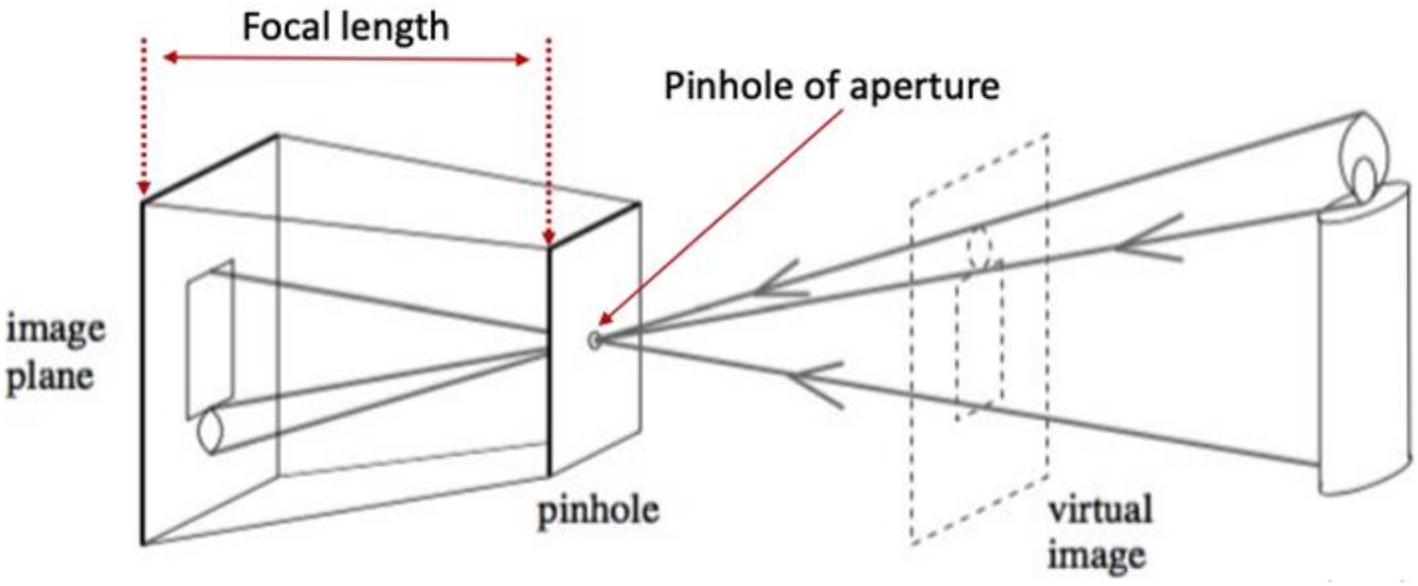


History of Pinhole Camera

- 1502: Leonardo da Vinci provides the first known clear description of the camera obscura (pinhole principle) in his notebooks.
- 1544: Gemma Frisius publishes the oldest known illustration of a camera obscura, used to observe a solar eclipse.



Pinhole Camera

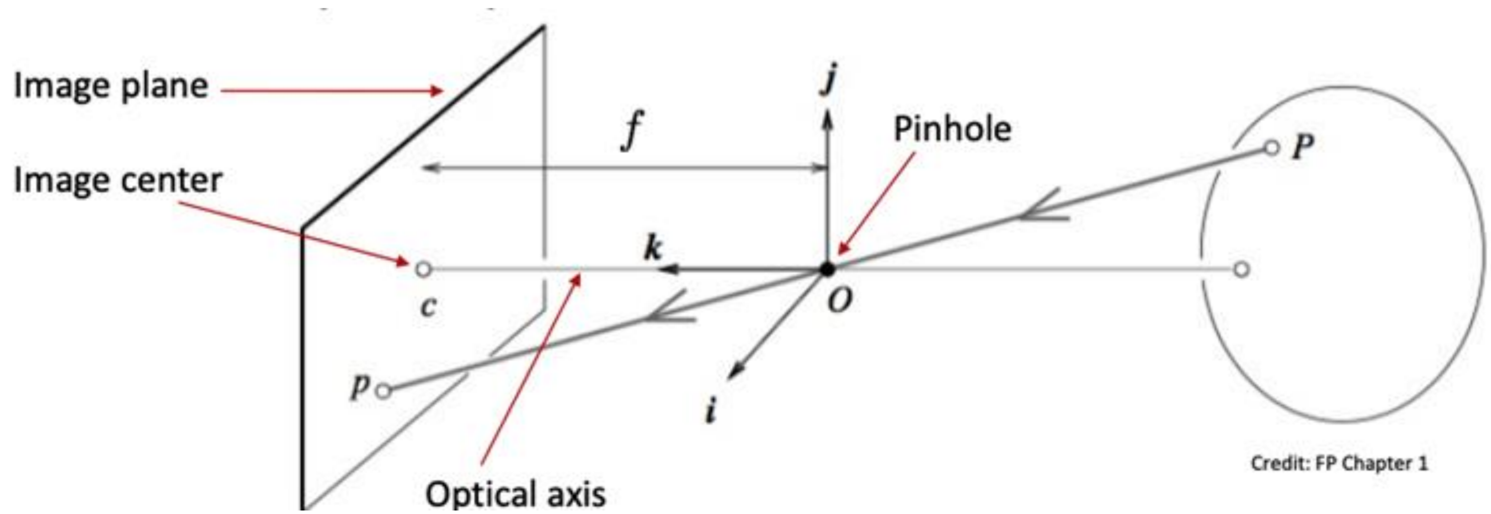


candle

- Perspective projection creates *inverted images*
- Sometimes it is convenient to consider a virtual image associated with a plane lying in front of the pinhole
- Virtual image is equivalent to the actual one except for on scale

Credit: Stanford AA 274

From 3D to 2D: The Pinhole Camera Model



Credit: FP Chapter 1

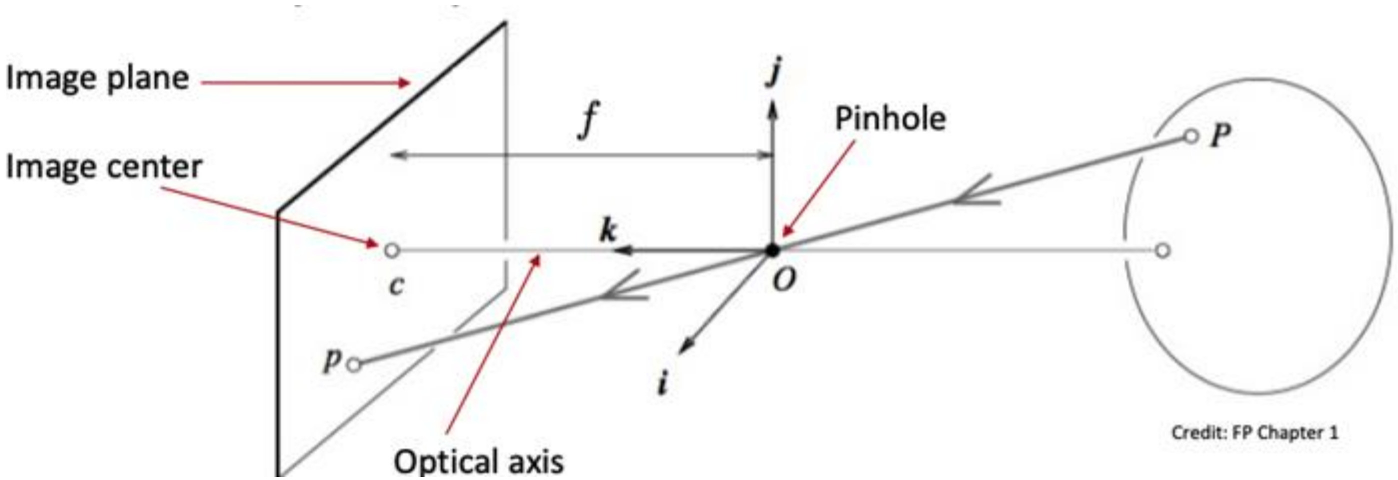
$$\begin{tikzcd} P = (X, Y, Z) \arrow[d, "\text{Perspective}"'] \\ p = (x, y, z) \end{tikzcd}$$

- How do we project a 3D point onto a 2D image?

We transform a point in the world coordinate frame, [X, Y, Z], into a pixel location on the image plane, [u, v]. This is accomplished through a series of coordinate transformations: from world, to camera, to image coordinates, and finally to pixels. The core of this process is a perspective projection, modeled by the camera's intrinsic and extrinsic parameters.

(X, Y, Z): 3D point in the Camera Coordinate System (origin at O).
(x, y) : 2D point in the Image Coordinate System (origin where the optical axis pierces the image plane).
f : Focal Length. The distance between the projection center O and the image plane.

The Geometry of Projection



Credit: FP Chapter 1

$$P = (X, Y, Z)$$

Perspective

$$p = (x, y, z)$$

The Principle of Collinearity:
The world point P, the camera center (pinhole) O, and its projected image point p are always collinear. This is the fundamental geometric constraint that enables us to model perspective projection.

$$\overline{Op} = \lambda \overline{OP}$$

$$\begin{cases} x = \lambda X \\ y = \lambda Y \\ z = \lambda Z \end{cases} \Leftrightarrow \quad \lambda = \frac{x}{X} = \frac{y}{Y} = \frac{z}{Z} \quad \Rightarrow \begin{cases} x = f\frac{X}{Z} \\ y = f\frac{Y}{Z} \end{cases}$$

Using the principle of similar triangles, we can find the coordinates of the projected point p.

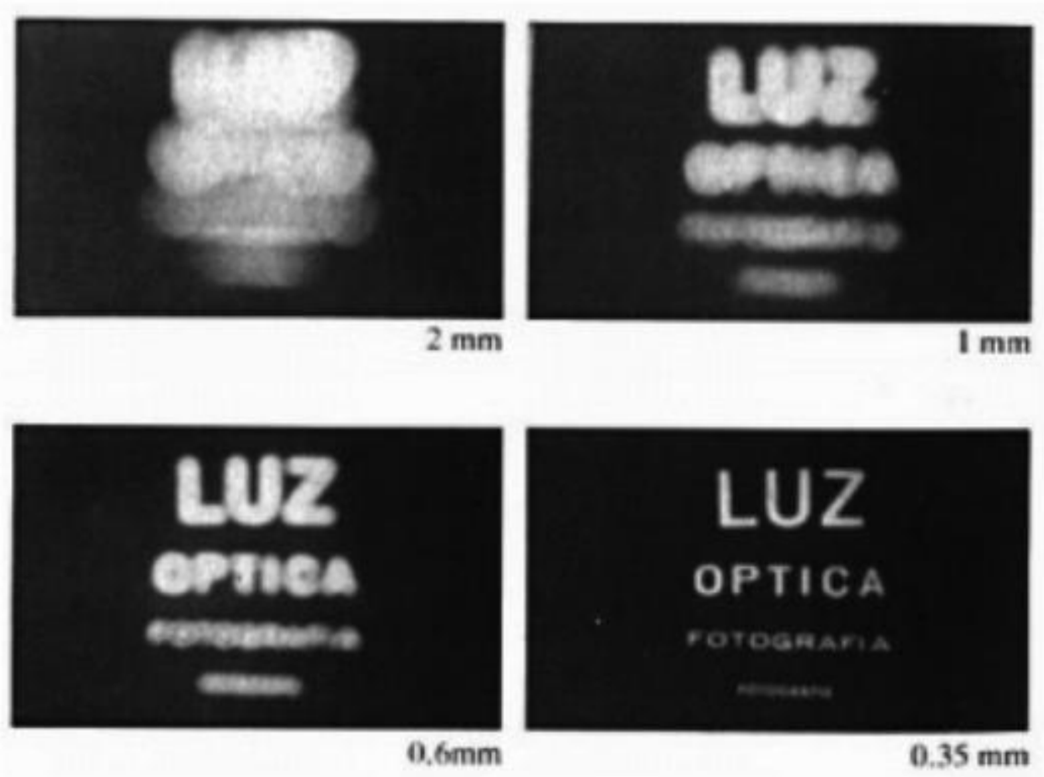
The Need for Lenses

The Pinhole Dilemma:

- A larger aperture allows more light, producing a brighter but blurrier image due to unfocused light rays.
- A smaller aperture creates a sharper image but is too dark for practical use.

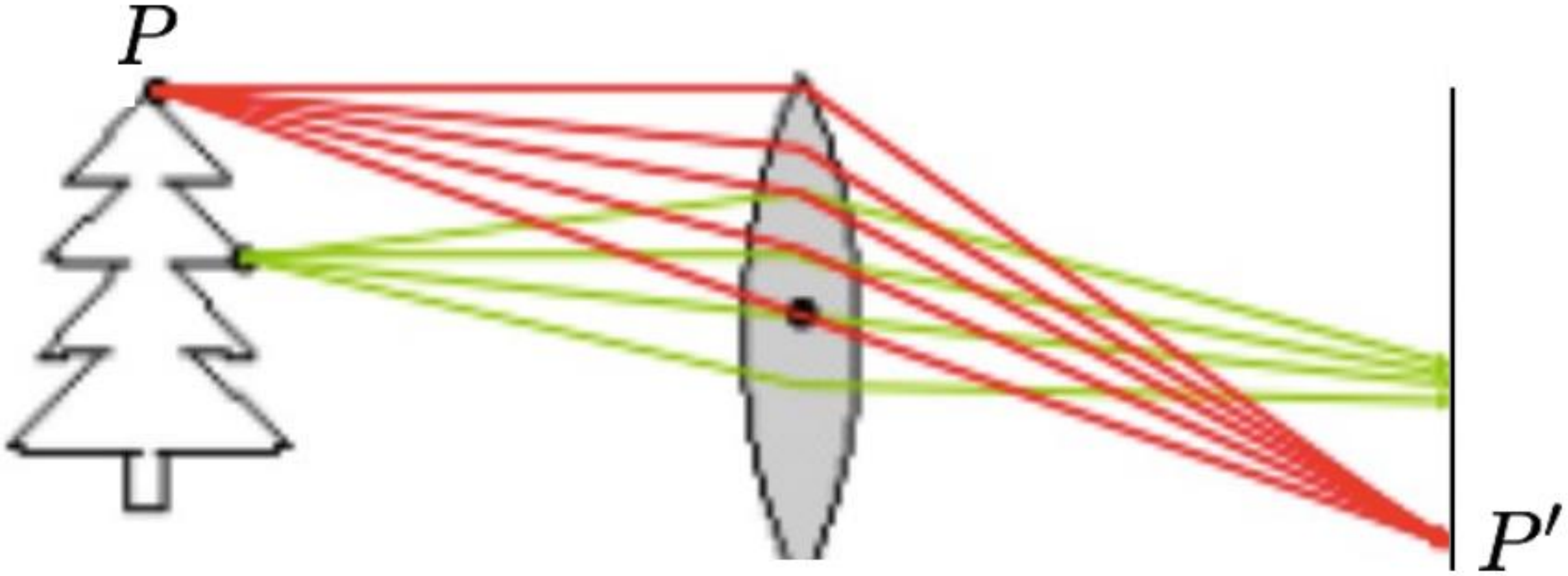
The Optical Solution:

- We replace the pinhole with a convex lens.
- The lens focuses light rays from a scene onto the image plane, providing both a bright and a sharp image.



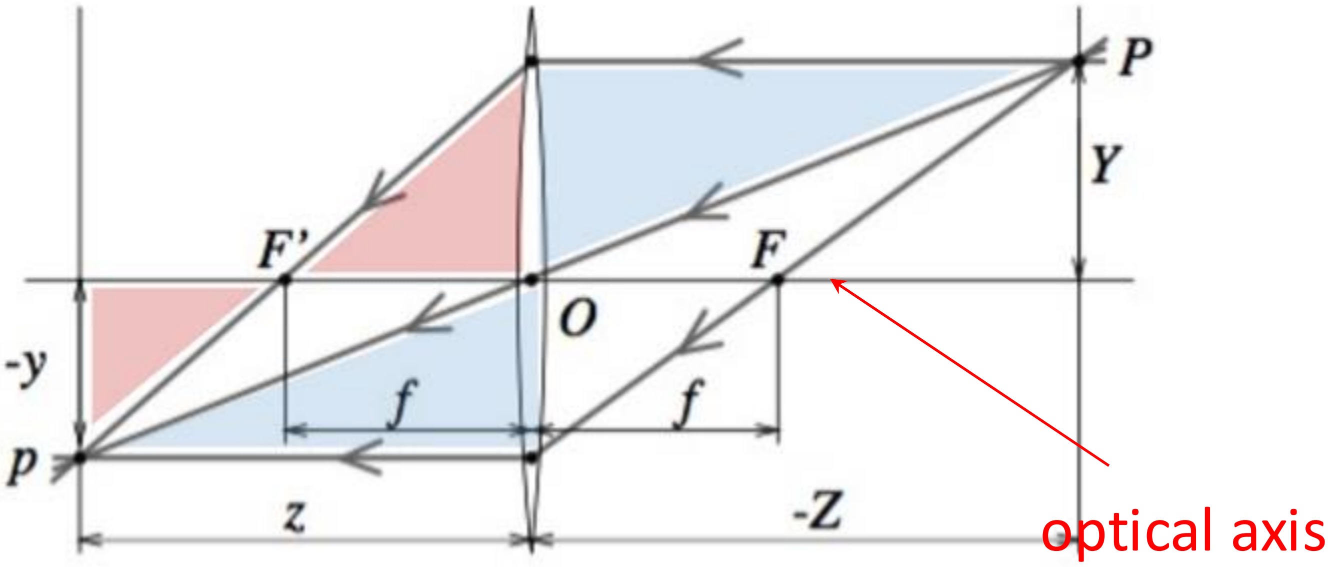
The Need for Lenses

- Lens: an optical element (glass) that focuses light by means of refraction



The Thin Lens Model

Unlike a pinhole, which allows only a single ray from P to reach p, a lens focuses a large bundle of rays from P onto the same point. This results in a brighter and clearer image.



A Practical Approximation:
The thin lens model simplifies complex lens optics into a set of predictable rules for focusing light.

Key Principles:

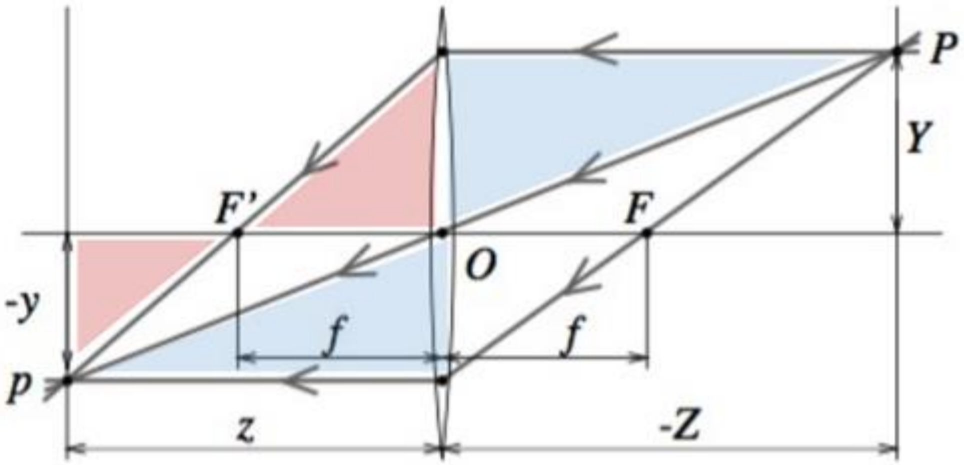
- No Deviation at the Center: Rays passing through the optical center (O) of the lens are not refracted and continue straight.
- Focusing Parallel Light: All rays parallel to the optical axis converge at a single point behind the lens: the focal point (F').
- Point-to-Point Focusing: All light rays emanating from a single scene point (P) are focused by the lens to a corresponding single point (p) on the image plane.

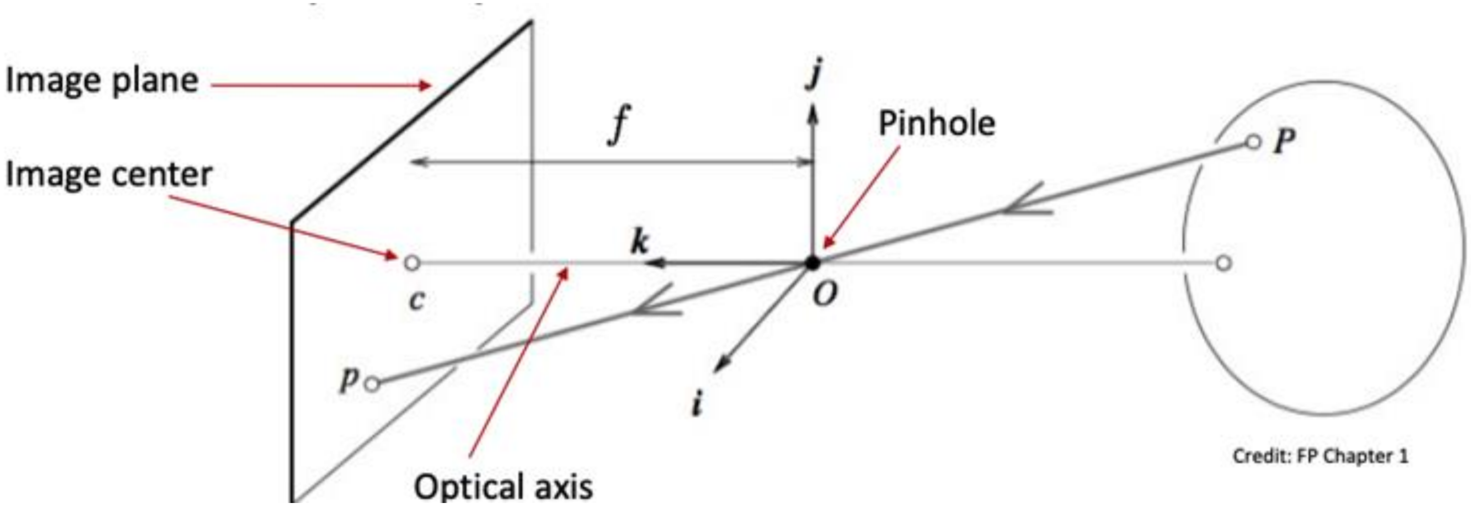
The Thin Lens Model

Assumptions:

When the image plane is placed at a distance z = f (the focal length) behind the lens, the thin lens projection geometry becomes identical to the pinhole model.

For simplicity, we ignore depth of field effects, assuming the entire scene is in perfect focus. This is a valid approximation when the camera is focused at infinity.





Credit: FP Chapter 1

Perspective Projection

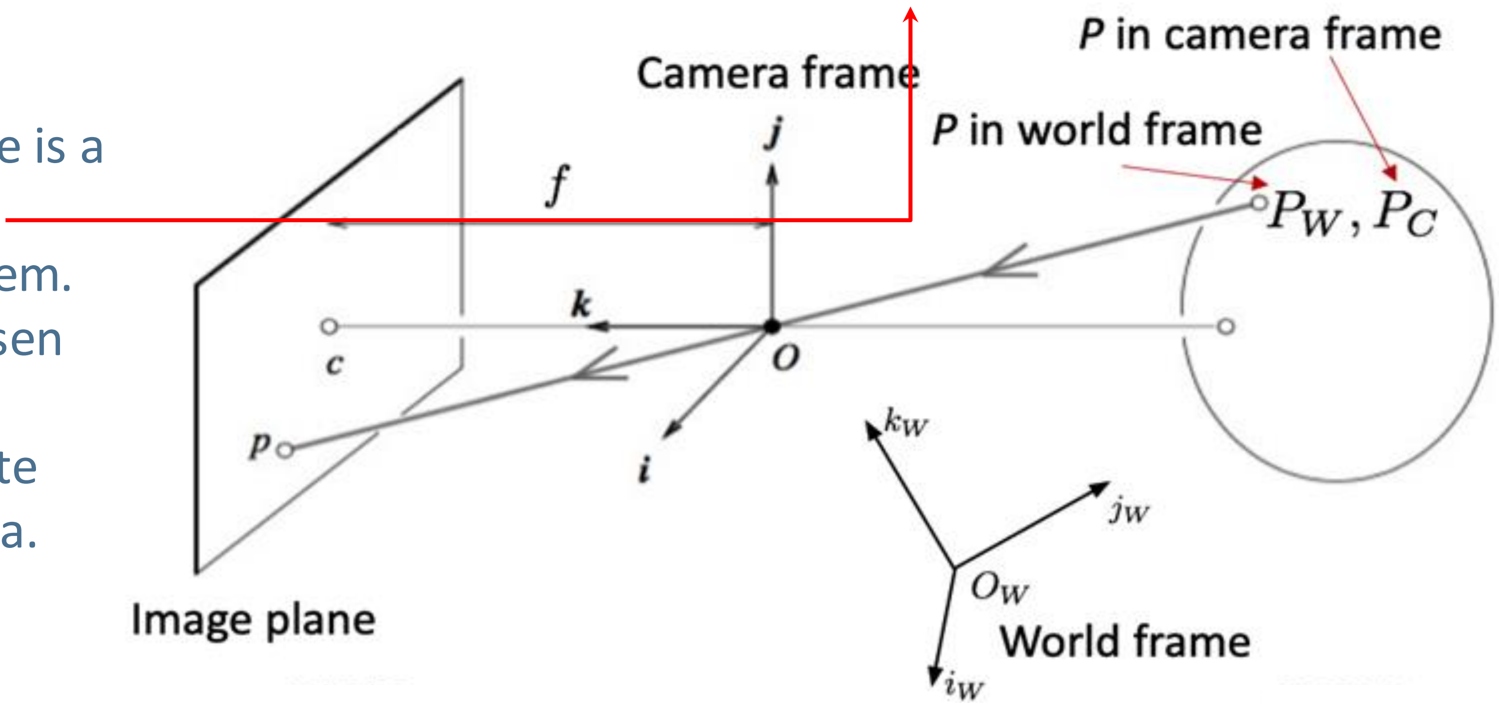
Our Goal: To project 3D points from the world onto a 2D camera image.

- We use the pinhole camera model.
- This also accurately models a thin-lens camera when focused at infinity.

Key Concept: The Camera Frame

- We define a camera coordinate frame.
- Its origin, O, is located at the lens center (the pinhole).
- The Z-axis is the optical axis, pointing out into the world.

The world frame is a fixed, global coordinate system. Its origin is chosen arbitrarily at a location separate from the camera.



Perspective Projection

1) World to Camera Transformation (P_w → P_c)

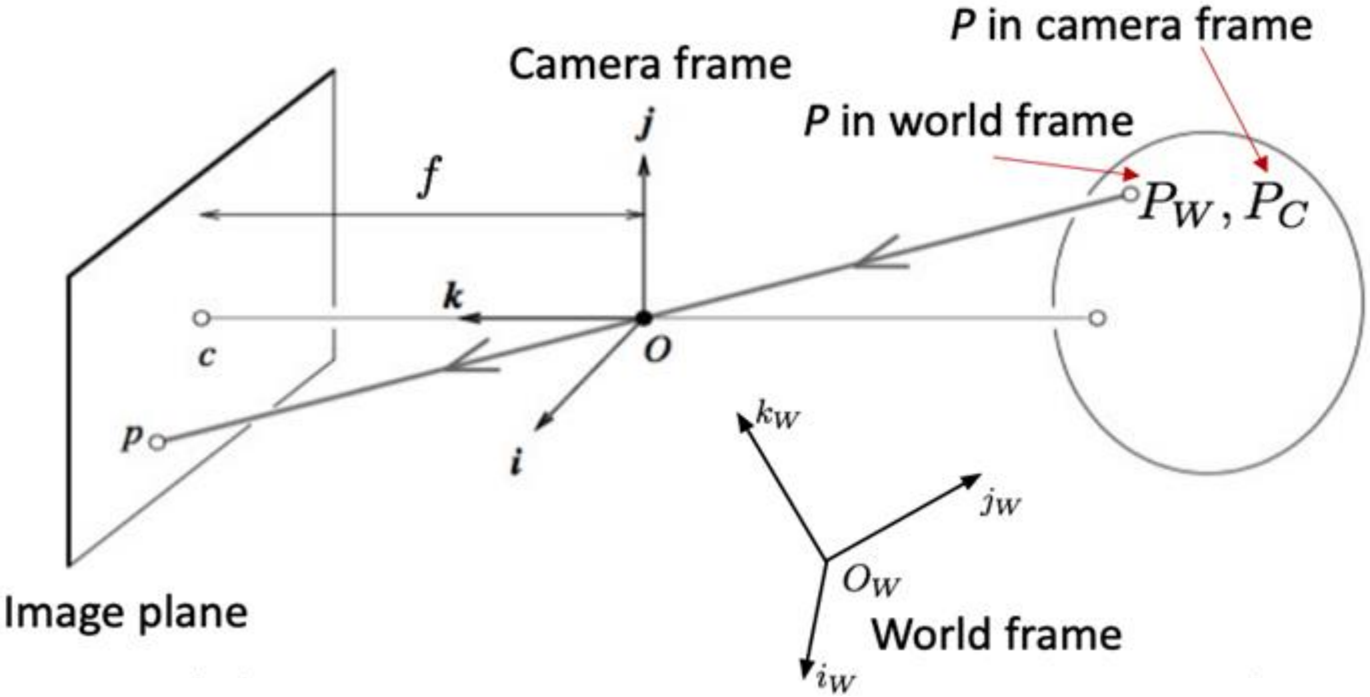
Transform the point from world coordinates to camera coordinates using the camera's pose (rotation R and translation t).

2) Perspective Projection (P_c → p)

Project the 3D point in the camera frame onto the normalized image plane using the pinhole model: (x, y) = (fX/Z, fY/Z).

3) Image to Pixel Transformation (p → (u, v))

Convert the metric image coordinates into discrete pixel coordinates using the camera's intrinsic calibration matrix K.


P in camera frame

Camera frame

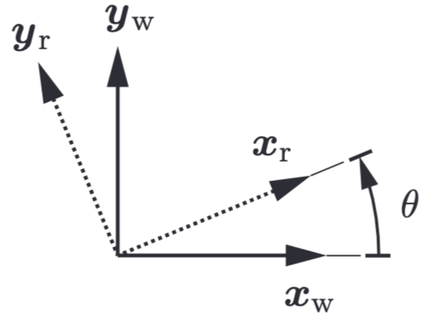
P in world frame

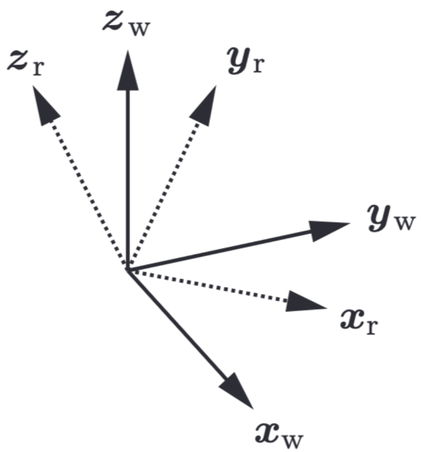
Image plane

World frame

P_w and P_c represent the same physical point in 3D space, but their coordinate values differ because they are expressed in different reference frames.

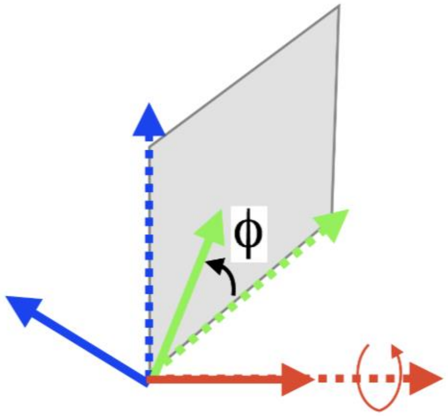
Euler Angles and Rotation Matrix





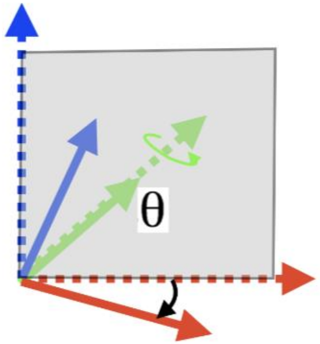
$$R_r^w = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin\theta & \cos(\theta) \end{bmatrix}$$

Rotation around the **x** axis



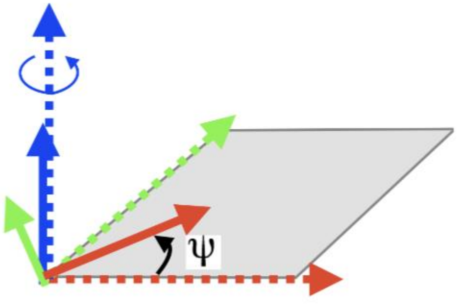
$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$

Rotation around the **y** axis



$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

Rotation around the **z** axis



$$R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Euler's Theorem implies that only three parameters (Euler angles) are needed to represent a 3D rotation.

A rotation matrix is a special matrix used to rotate vectors in a Euclidean space (like 2D or 3D) around the origin (the point (0,0)) by a given angle.

$$R_r^w = R_x(\phi_1)\, R_y(\theta_1)\, R_x(\psi_1)$$

Camera Pose

Defining Pose: Position and Orientation

The pose of a body frame {r} relative to a world frame {w} is completely described by:

Position: $\mathbf{t}_r^w$ - The translation from the origin of {w} to the origin of {r}.

Orientation: $\mathbf{R}_r^w$ - The rotation from {r} to {w}.

The pair $(\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}}, \boldsymbol{t}_{\mathrm{r}}^{\mathrm{w}})$ is the complete geometric representation of frame {r} in {w}.

A pose has 6 degrees of freedom (3 for translation, 3 for rotation). We can combine these into a single, convenient transformation matrix.

$$\boldsymbol{T}_{\mathrm{r}}^{\mathrm{w}} = \begin{bmatrix} \boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}} & \boldsymbol{t}_{\mathrm{r}}^{\mathrm{w}} \\ \mathbf{0}_d^\top & 1 \end{bmatrix}$$

Rigid-body Transformations

We now generalize from pure rotation to full pose (rotation and translation).

Given:

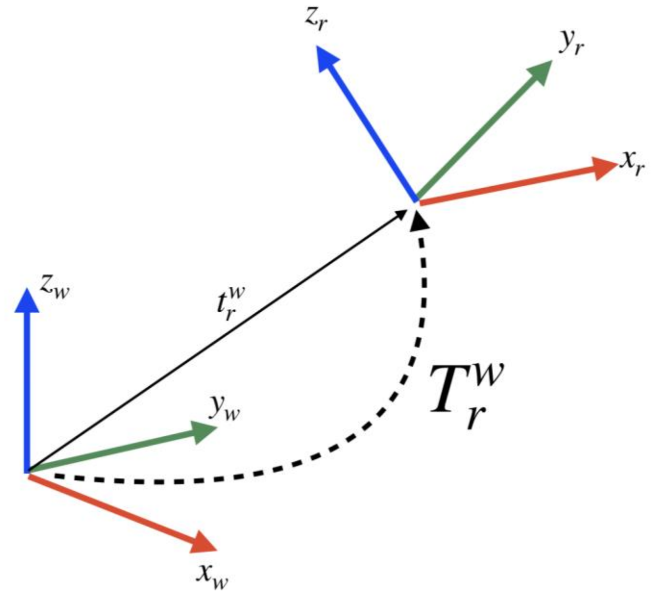
A point's coordinates in frame {r}: $\mathbf{p}^{\mathrm{r}}$

The pose of {r} relative to {w}, represented by the transformation matrix: $\mathbf{T}_{\mathrm{r}}^{\mathrm{w}}$

Compute: The point's coordinates in frame {w}: $\mathbf{p}^{\mathrm{w}}$

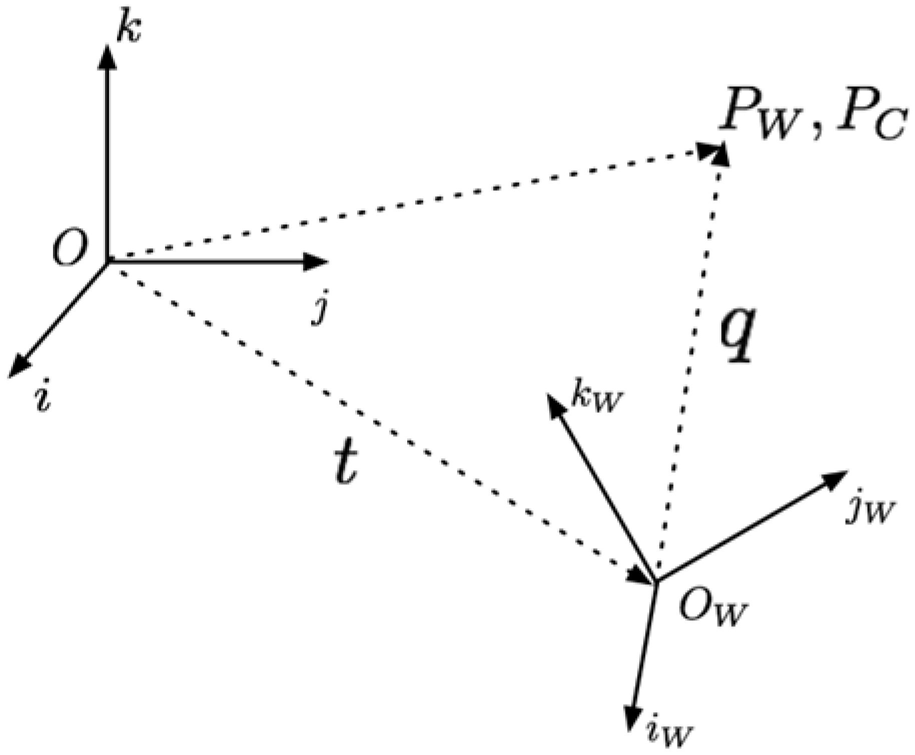
The transformation is given by:

$$\boldsymbol{p}^{\mathrm{w}} = \boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}} \boldsymbol{p}^{\mathrm{r}} + \boldsymbol{t}_{\mathrm{r}}^{\mathrm{w}}$$



World to Camera Transformation

Step 1: World to Camera Transformation (P_w → P_c)



$$P_C = t + q$$

Rotation + Translation:
Rigid transformation

$$q = R\,P_W$$

where *R* is the rotation matrix relating camera and world frames

$$R = \begin{bmatrix} i_W \cdot i & j_W \cdot i & k_W \cdot i \\ i_W \cdot j & j_W \cdot j & k_W \cdot j \\ i_W \cdot k & j_W \cdot k & k_W \cdot k \end{bmatrix}$$

$$\boxed{\Rightarrow P_C = t + R\,P_W}$$

World to Camera Transformation

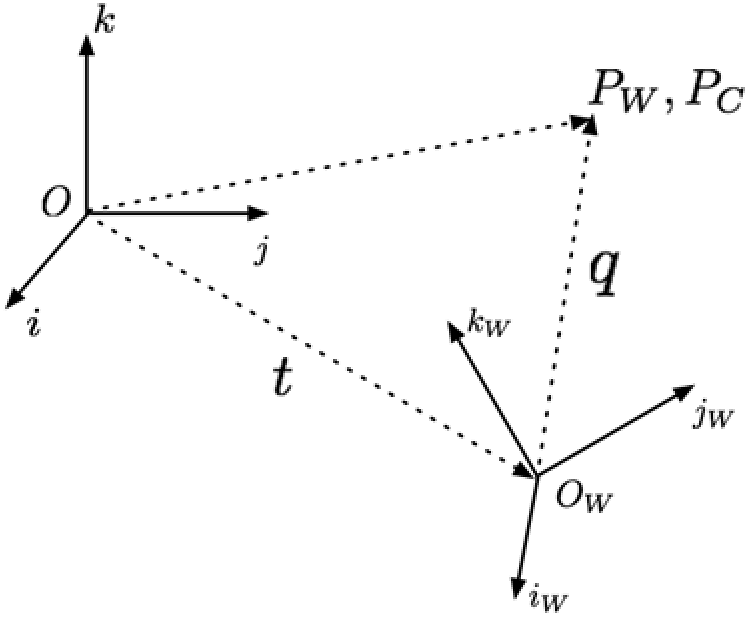
Step 1: World to Camera Transformation (P_w → P_c)
In homogeneous coordinates

$$\boxed{\Rightarrow P_C = t + R\,P_W}$$

$$\begin{pmatrix} P_C \\ 1 \end{pmatrix} = \begin{bmatrix} R & t \\ 0_{1\times 3} & 1 \end{bmatrix} \begin{pmatrix} P_W \\ 1 \end{pmatrix}$$

Point P_c in homogeneous coordinates

Point P_w in homogeneous coordinates



Perspective Projection

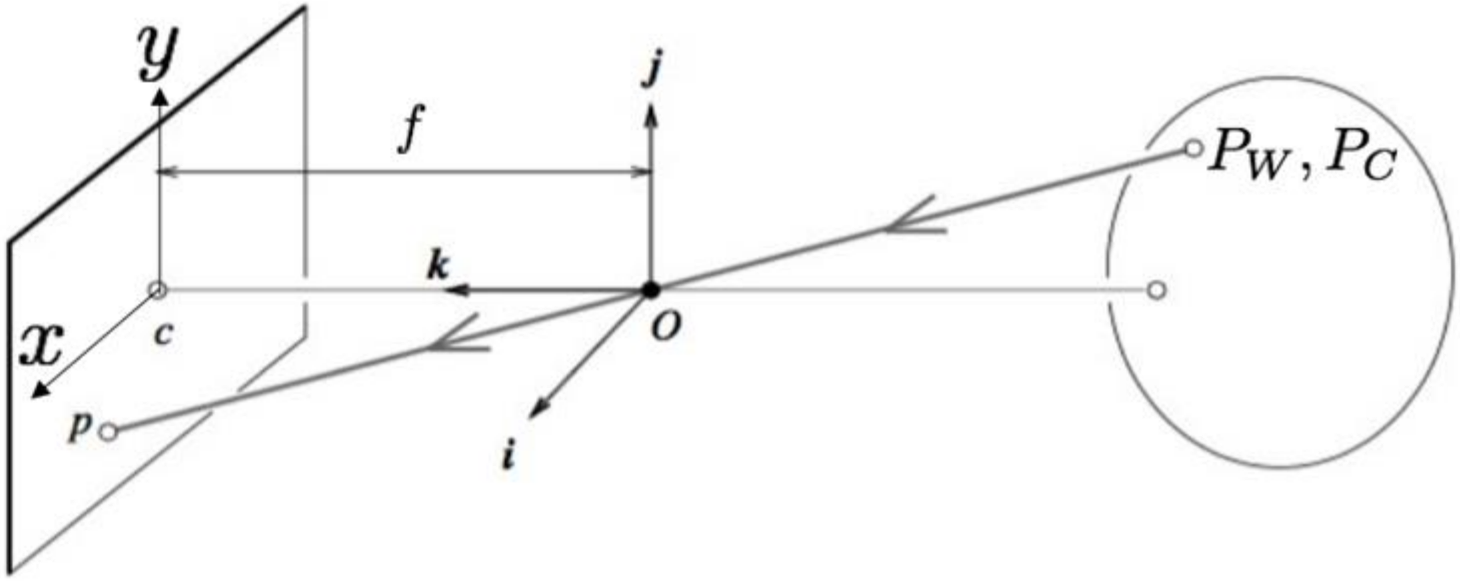
Step 2: Perspective Projection (P_c → p)

Map $P_c = (X_C, Y_C, Z_C)$ into $p = (x, y)$ (image plane)

We already have $\begin{cases} x = f\frac{X_C}{Z_C} \\ y = f\frac{Y_C}{Z_C} \end{cases}$

$$\overline{Op} = \lambda\overline{OP}$$

$$\begin{cases} x = \lambda X \\ y = \lambda Y \\ z = \lambda Z \end{cases}$$



$$z = f$$

$$\lambda = \frac{x}{X} = \frac{y}{Y} = \frac{z}{Z}$$

$$\begin{cases} x = f\frac{X_C}{Z_C} \\ y = f\frac{Y_C}{Z_C} \end{cases}$$

Image to Pixel Transformation

Step 3: Image to Pixel Transformation (p → (u, v))

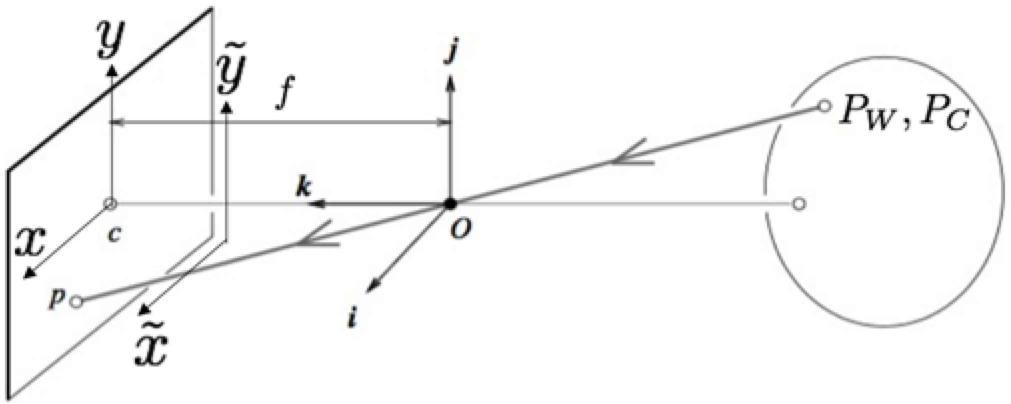
The normalized image point p exists in a metric 2D plane centered on the optical axis. We must map it to the discrete image sensor via an affine transformation:

Change of Origin: The principal point c = (c_x, c_y) defines where the optical axis pierces the image plane, in pixels. We translate the point by c.

Unit Conversion: The focal length f (in meters) is converted to pixels using the sensor's pixel density, giving us f_x and f_y.

$$\tilde{x} = f\,\frac{X_C}{Z_C} + \tilde{x}_0, \qquad \tilde{y} = f\,\frac{Y_C}{Z_C} + \tilde{y}_0,$$

$$\begin{cases} x = f\,\frac{X_C}{Z_C} \\ y = f\,\frac{Y_C}{Z_C} \end{cases}$$



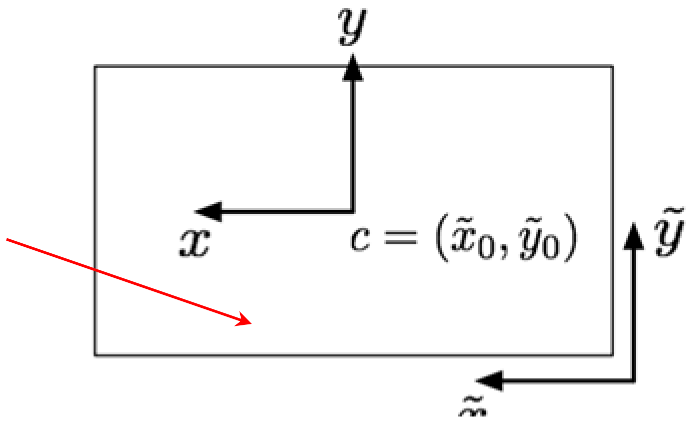


Image to Pixel Transformation

Step 3: Image to Pixel Transformation (p → (u, v))

Convert the image coordinates to pixel coordinates (image is in centimeters, etc; pixel is different)
Simply change the scale of the coordinates

We already have

$$\tilde{x} = f\,\frac{X_C}{Z_C} + \tilde{x}_0, \qquad \tilde{y} = f\,\frac{Y_C}{Z_C} + \tilde{y}_0,$$

Number of pixels per unit distance in image coordinates

$$\begin{aligned} u &= k_x\tilde{x} = \overbrace{k_x\, f}^{\alpha}\,\frac{X_C}{Z_C} + \overbrace{k_x\tilde{x}_0}^{u_0} \\ v &= k_y\tilde{y} = \underbrace{k_y\, f}_{\beta}\,\frac{Y_C}{Z_C} + \underbrace{k_y\tilde{y}_0}_{v_0} \end{aligned} \quad \Rightarrow \quad \boxed{\begin{aligned} u &= \alpha\frac{X_C}{Z_C} + u_0 \\ v &= \beta\frac{Y_C}{Z_C} + v_0 \end{aligned}}$$

Nonlinear transformation

Homogeneous coordinates

To express the multi-stage projection (rigid transformation + perspective division + intrinsics) as a single linear mapping, we must move from Euclidean coordinates to projective geometry.

Euclidean Space: A point is (X, Y, Z).

Homogeneous Space: The same point is represented as (X, Y, Z, 1). Any point (X, Y, Z, w) where w ≠ 0 corresponds to the Euclidean point (X/w, Y/w, Z/w).

$$\begin{pmatrix} x \\ y \end{pmatrix} \Rightarrow \lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \qquad \begin{pmatrix} x \\ y \\ z \end{pmatrix} \Rightarrow \lambda \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Homogeneous coordinates

Projection can be equivalently written in homogeneous coordinates

The XYZ in camera frame

The x,y in pixels

$$\left[\overbrace{\boxed{\begin{matrix}\alpha & 0 & u_0\\ 0 & \beta & v_0\\ 0 & 0 & 1\end{matrix}}}^{K}\quad\begin{matrix}0\\0\\0\end{matrix}\right]\begin{pmatrix}X_c\\ Y_c\\ Z_c\\ 1\end{pmatrix}=\begin{pmatrix}\alpha X_c+u_0Z_c\\ \beta Y_c+v_0Z_c\\ Z_c\end{pmatrix}$$

Focal length + Principal point offset + the pixel–real-world scale

Camera matrix/ Matrix of intrinsic parameters

P_c in homogeneous coordinates

Homogeneous pixel coordinates

$$p^h=[K\quad 0_{3\times 1}]P_C^h$$

Homogeneous coordinates

- Projection can be equivalently written in homogeneous coordinates

Point p in homogeneous pixel coordinates

$$p^h = [K \quad 0_{3\times 1}]P_C^h$$

Point P_c in homogeneous camera coordinates

Homogeneous coordinates

- Collecting all results

$$p^h = [K \quad 0_{3\times1}]P_C^h = K[I_{3\times3} \quad 0_{3\times1}]\begin{bmatrix} R & t \\ 0_{1\times3} & 1 \end{bmatrix} P_W^h$$

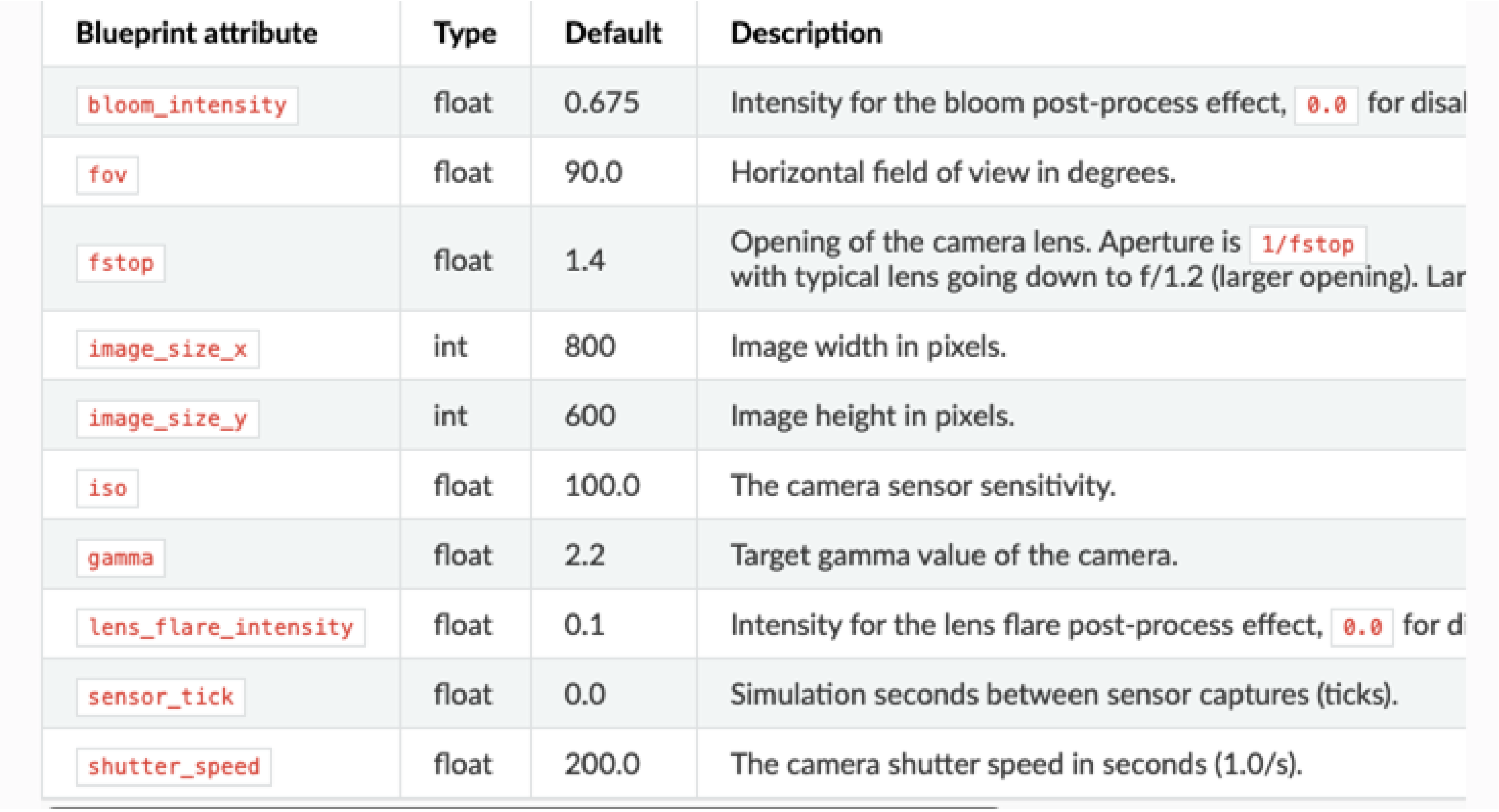
- Hence

$$\boxed{p^h = \overbrace{\underset{\text{Intrinsic parameters}}{K}\underset{\text{Extrinsic parameters}}{[R \quad t]}}^{\text{Projection matrix } M} P_W^h}$$

R: rotation
t: translation
Rt: extrinsic matrix

Degree of freedom:
K: 4
Rt: 6

Camera Parameters



Blueprint attribute	Type	Default	Description
`bloom_intensity`	float	0.675	Intensity for the bloom post-process effect, `0.0` for disal
`fov`	float	90.0	Horizontal field of view in degrees.
`fstop`	float	1.4	Opening of the camera lens. Aperture is `1/fstop` with typical lens going down to f/1.2 (larger opening). Lar
`image_size_x`	int	800	Image width in pixels.
`image_size_y`	int	600	Image height in pixels.
`iso`	float	100.0	The camera sensor sensitivity.
`gamma`	float	2.2	Target gamma value of the camera.
`lens_flare_intensity`	float	0.1	Intensity for the lens flare post-process effect, `0.0` for di
`sensor_tick`	float	0.0	Simulation seconds between sensor captures (ticks).
`shutter_speed`	float	200.0	The camera shutter speed in seconds (1.0/s).

$$\boxed{p^h = \overbrace{\underset{\text{Intrinsic parameters}}{K}[\underset{\text{Extrinsic parameters}}{R \quad t}]}^{\text{Projection matrix } M} P_W^h}$$

R: rotation
t: translation
Rt: extrinsic matrix

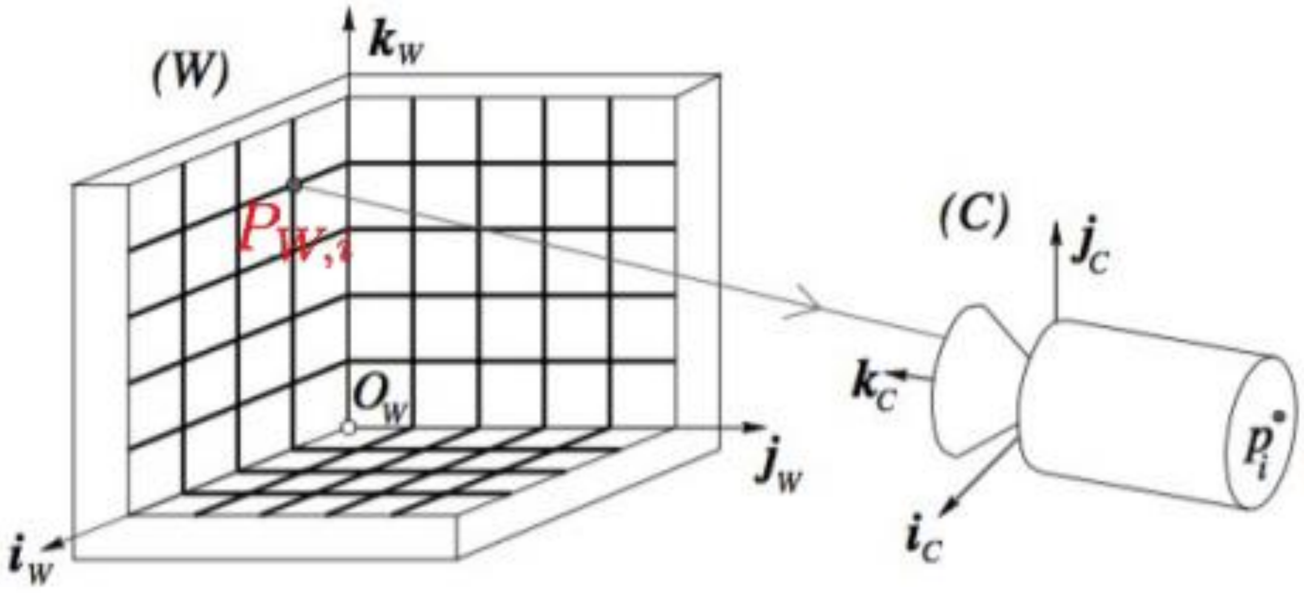
Camera Calibration

- Solve for the projection matrix P = K [R | t].
- The model has 10 essential parameters (4 from K, 6 from [R|t]). An 11th (skew) is often negligible.
- Use a set of n known 3D points (X, Y, Z) and their corresponding 2D image points (u, v).
- Each point correspondence provides 2 equations. Therefore, a minimum of 6 points is required to solve the system (since 6 points * 2 = 12 equations > 11 unknowns).

$$p_i \leftrightarrow P_{W,i}$$

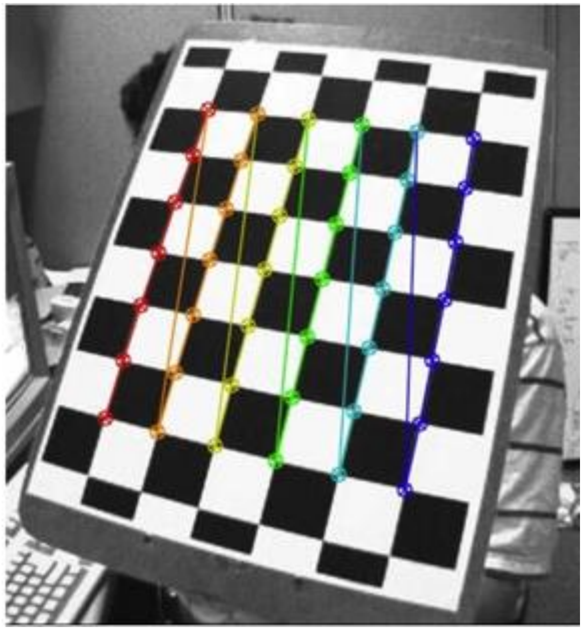
$P_{W,1}, P_{W,2}, \dots, P_{W,n}$ with known positions in world frame

$p_1, p_2, \dots, p_n$ with known positions in image frame



Camera Calibration

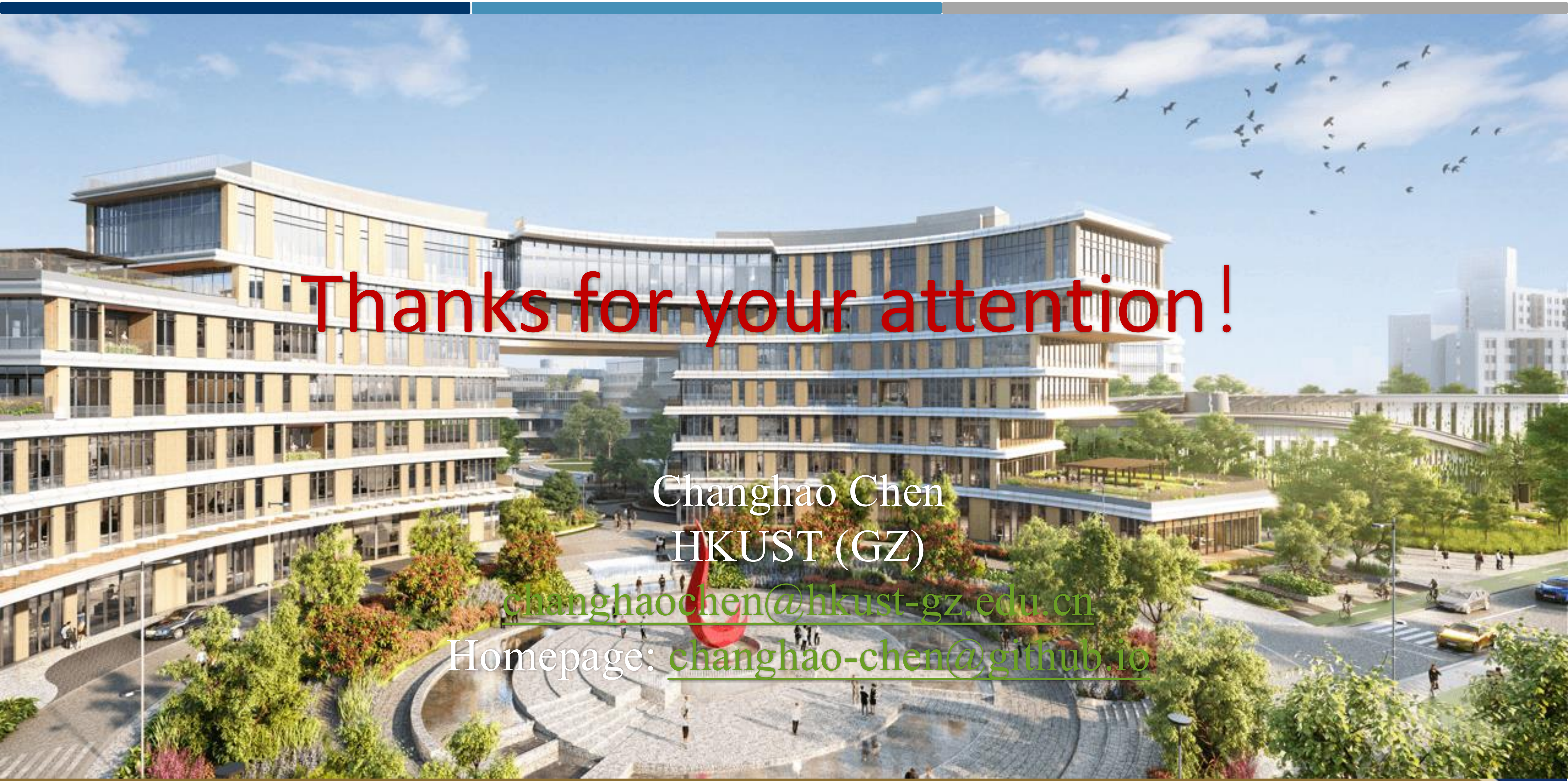
Chessboard + OpenCV



A chessboard provides a known, regular 3D structure.

Its high-contrast corners create features that are easy to detect and match across images, generating thousands of accurate correspondences automatically.

This data is used in a robust optimization to compute the camera parameters.
Sources: https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html



Thanks for your attention！

Changhao Chen
HKUST (GZ)
changhaochen@hkust-gz.edu.cn
Homepage: changhao-chen@github.io