The Pinhole Camera Matrix (Mathematical Concept).

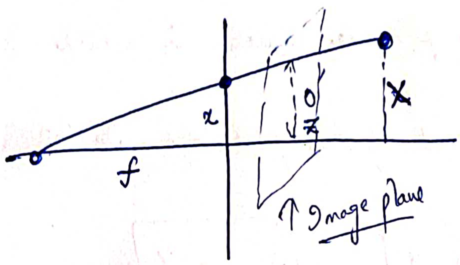
↳ Internal camera Matrix (Intrinsic)

↳ External camera Matrix (Extrinsic)

(Transforming 3D Camera co-ordinates to 2D homogeneous image co-ordinates).

Internal Camera Matrix : (Transformation matrix that converts points from Camera coordinate system to pixel co-ordinate system)

↳ Consider the pinhole camera, given below, the plane with the small hole in it and the projection plane is shown (on the left from the pinhole). The distance between the 2 planes is f (focal length distance). (I & pinhole)

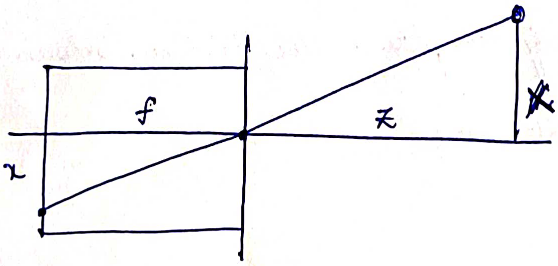


(Image)

(Virtual Model of pinhole)

↳(projection plane infront of the pinhole)

(Real)



(Physical model of pinhole camera).

- Choose the co-ordinate plane, such that O is the center of the hole. The Z-direction is along the axis perpendicular to the walls. The X and Y directions are in the plane of the wall.

- Projection wall, $\boxed{Z = -f}$
  (i.e. the plane)

- Knowing the real-world co-ordinates of X, Y, Z of point P can help us calculate the image co-ordinates x, y of the point P' projected onto the image plane Π :

$$\boxed{x = -f\frac{X}{Z}} \quad , \quad \boxed{y = -f\frac{Y}{Z}}$$

Minus sign indicates that the projected image on the back of the pinhole camera is upside down (inverted). To correct this mirroring ~~error~~ in our model, we use a virtual pinhole camera in which the retinal plane is put at $Z = f$, i.e infront of the pinhole. In this, we have -

$$\boxed{x = f\frac{X}{Z}} \quad , \quad \boxed{y = f\frac{Y}{Z}}$$

from geometrical point of view of pinhole camera, is thus characterised with a point and the projection plane. The distance between the point & the plane is the focal distance.

Modelling of Ideal pinhole camera

→ Using homogeneous co-ordinates, we have →

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

→ The above pinhole camera model is a geometrical model: the co-ordinates (X, Y, Z and x, y) and the focal length are measured at the following Internal camera model:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim K \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

where K is the Internal camera matrix:

$$K = \begin{bmatrix} f_x & \alpha & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

② Extrinsic Camera Matrix: (transformation from world coordinate system to camera co-ordinate system)

The Pinhole camera model is represented with the projection as:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim K \begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix}$$

$$\tilde{x} \sim K \tilde{X}_c$$

where ($K \rightarrow$ Internal camera matrix)

- Problem: (with this matrix)

↳ It assumes that we can represent points in 3D space in co-ordinates wrt the camera frame where the Z-axis is the optical axis. (that is why we have to use (X_c, Y_c, Z_c)).

↳ Practically, we cannot measure what co-ordinate axes are. (to do this you need to open up a camera).

4

in metre (or millimetres.) Practically, the camera co-ordinate (x, y) are measured in pixel distances. (the sampling distances Δx & Δy)

Converting the geometrical model from metre to pixel distances :

We use the scale factor, S_x and S_y

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} S_x f & 0 & 0 & 0 \\ 0 & S_y f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Note that Now (x, y) are measured in pixel co-ordinates.

Often we write, $f_x = S_x f$ and $f_y = S_y f$ →

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} f_x & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Pixel co-ordinates are not given with respect to a frame that is centered at the optical axis, instead the co-ordinates are in the the Quadrant

This implies a translation of the Co-ordinate frame:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} f_x & 0 & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Finally, we need to introduce a skew factor α, that accounts for a shear of the co-ordinate system (That might occur in case the optical axis is not precisely perpendicular to the Retina). This we arrive

... functions on top of each other:

(5)

Assume, we know the 3-D co-ordinates of points wrt an arbitrary frame (often called the world frame), The camera frame is a rotated and translated version of this world frame.

Let the matrix, $\begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}$ be the frame transform that transforms the world frame into the camera frame.

Then the co-ordinate transformation is the inverse of this transform. So a point (X, Y, Z) in world co-ordinates has co-ordinates:-

$$\tilde{X}_c = \begin{pmatrix} R^T & -R^T t \\ 0 & 1 \end{pmatrix} \tilde{X}$$

and thus the projection of $\tilde{X}$ on the Retina is:

$$\tilde{x} = K \begin{pmatrix} R^T & -R^T t \\ 0^T & 1 \end{pmatrix} \tilde{X}$$

After Simplification, we have,

$$\tilde{x} = K (R^T \quad -R^T t) \tilde{X}$$

$$= \begin{pmatrix} f_x & \alpha & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} (R^T \quad -R^T t) \tilde{X}$$

where we have re-defined the Internal Matrix K. The matrix $(R^T \quad -R^T t)$ is called the external matrix $P = K(R^T \quad -R^T t)$ is called the camera matrix.

Camera calibration

estimating the camera matrix :

Our camera model projects a 3D point $\tilde{X}$ on a 2D Retina resulting in point $\tilde{x}$: $\boxed{\tilde{x} \sim P\tilde{X}}$

Mathematically,

$$\tilde{x} = sP\tilde{X} \text{ for some Non-zero scalar } s$$

i.e the vector $\tilde{x}$ and $P\tilde{X}$ are parallel, therefore $\boxed{\tilde{x} \times P\tilde{X} = 0}$

→ Using $\tilde{p}_1, \tilde{p}_2, \tilde{p}_3$ to denote the 3 row vectors of P and using $\tilde{x} = (x\ y\ 1)^T$ calculating the cross product leads to -

$$\tilde{x} \times P\tilde{X} = \begin{pmatrix} x\tilde{X}^T\tilde{p}_3 - \tilde{X}^T\tilde{p}_1 \\ y\tilde{X}^T\tilde{p}_3 - \tilde{X}^T\tilde{p}_2 \\ x\tilde{X}^T\tilde{p}_2 - y\tilde{X}^T\tilde{p}_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

→ The third element in this vector is a linear combination of the first two, thus we can consider the first 2 elements only.

$$\begin{pmatrix} \tilde{X}^T & 0^T & -x\tilde{X}^T \\ 0^T & \tilde{X}^T & -y\tilde{X}^T \end{pmatrix} \begin{pmatrix} \tilde{p}_1 \\ \tilde{p}_2 \\ \tilde{p}_3 \end{pmatrix} = \begin{pmatrix} \tilde{X}^T & 0^T & -x\tilde{X}^T \\ 0^T & \tilde{X}^T & -y\tilde{X}^T \end{pmatrix} p = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

p stacks the 3 rows of P on top of each other forming a 12 component vector.

↳ The above equation is a homogeneous set of equations. It is based on just one pair of point correspondence $(X, Y, Z) \to (u, v)$.

(6) For n - correspondences, we may stack $2n$ equations on top of each other:

$$\begin{bmatrix} \tilde{X}_1^T & 0^T & -x_1 \tilde{X}_1^T \\ 0^T & \tilde{X}_1^T & -y_1 \tilde{X}_1^T \\ \vdots & \vdots & \vdots \\ \tilde{X}_n^T & 0^T & -x_n \tilde{X}_n^T \\ 0^T & \tilde{X}_n^T & -y_n \tilde{X}_n^T \end{bmatrix} p = 0$$

$Ap = 0$

↳ Note : We search for the vector p that minimizes $\|Ap\|$ subject to the constraint that $\|p\| = 1$. The 'SVD-trick' can be used here.

- Given the optimal vector p, we can reshape this vector into 3×4 camera matrix P. Given the calibrated camera matrix we may take each model 3D point (X_i, Y_i, Z_i) and project it on the camera plane and obtain (a_i, b_i) when we compare this 'reprojected' point (a_i, b_i) with the 'true' projected point (x_i, y_i), we expect that the re-projection distance $d_i = \sqrt{(x_i - a_i)^2 + (y_i - b_i)^2}$ to be small.

- However, finding the camera matrix P by minimizing $\|Ap\|$ subject to $\|p\|$ does not guarantee that the sum of square reprojection distances is minimal. But minimizing the sum of square reprojection errors is a difficult non-linear optimization problem that can be approximately solved.

• Extrinsic Matrix

→ describes the camera's location in the world, what direction it is pointing.

→ Two components:

  → Rotation Matrix, R

  → Translation Vector, t.