

# 1 Geometry for Computer Vision

## Motivation

We need to model the image formation process. The camera can act as an (angular) measurement device. However, we need a mathematical model for a simple camera. Two cameras are better than one: they can provide metric measurements.

## Notation

A word about notation: we will often use lowercase letters for image quantities such as 2D points  $p$  and  $q$ , and uppercase letters for 3D quantities, such as 3D points  $P$  and  $Q$ . We also follow this rule when talking about the coordinates of a point, which we will write in two different ways, depending on whether we mention in the text, such as  $p = (x, y)$ , or in a display formula, such as

$$p = \begin{bmatrix} x \\ y \end{bmatrix} \quad \text{or} \quad P = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Mathematically, we think of 2D and 3D points as column vectors, but the  $(.)$  notation helps us avoid always writing a transpose, i.e., we have  $p = \begin{bmatrix} x & y \end{bmatrix}^T = (x, y)$ .

### 1.1 Pinhole Camera Model

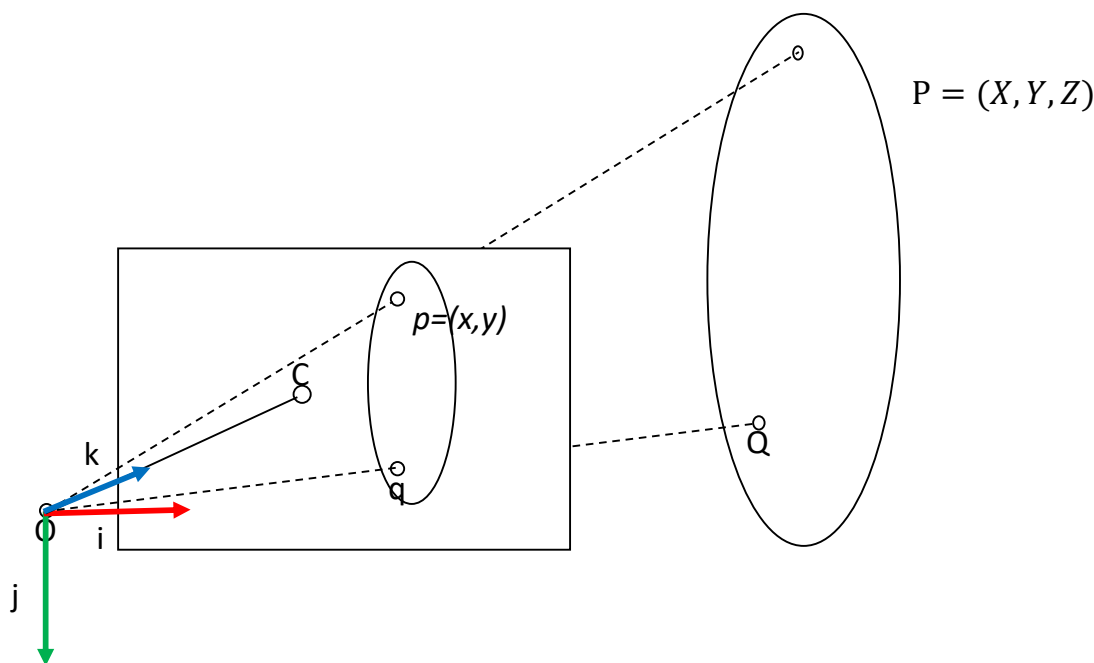


Figure 1: Pinhole camera model for projecting 3D points  $P$  in the scene to 2D points  $p$  in the image.

Geometrically, the simplest and most-used camera model is the **pinhole camera model**. In short, to project a 3D point  $P = (X, Y, Z)$  to a 2D image, we use the following equation:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} X/Z \\ Y/Z \end{bmatrix} \quad (1)$$

This assumes that the 3D point  $P$  is expressed in camera coordinates, with the  $Z$  axis pointing into the scene, the  $X$  axis pointing to the right, and the  $Y$  axis pointing *down*. This then yields 2D coordinates  $(x, y)$  where  $x$  increases to the right and  $y$  increases downward. The pinhole camera model is illustrated in Figure 1. The camera coordinate frame in which the 3D points are expressed is shown on the left. The point  $O$  is the **optical center** and the point  $c$  is the **image center**, i.e., where the **optical axis** (the 3D  $Z$ -axis, by our convention) pierces the image plane.

## 1.2 Camera Calibration Parameters

We now discuss how the dimensionless 2D coordinates  $x$  and  $y$  give rise to **image coordinates** expressed in units of pixels, which is how we typically access images.

Equation 1 assumes that the image plane is located at a distance of 1 of whatever the units we use for the 2D point  $p$ . If instead it is placed at a distance  $F$ , we get

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = F \begin{bmatrix} X/Z \\ Y/Z \end{bmatrix} \quad (2)$$

In Equation 2 the 2D point and the **focal length**  $F$  are always expressed in the same units. If the focal length  $F$  is expressed in mm., which is common in the industry, then  $x$  and  $y$  will be in mm. as well, etc. Note that the units of the 3D point  $P$  do not matter, which is significant: we will never know, from a single image alone, whether we are looking at a life-size scene of a dollhouse representation of it. Some might say this is why television works :-)

By Equation 2 The image center  $c$  always has the coordinates  $c = (0, 0)$ . But this is not typically how we address pixels in the image! Instead, we use image coordinates  $(u, v)$ , which have their origin in the upper left. If the origin is there, then the image center must have non-zero coordinates, and this depends on the resolution of the image sensor. By convention we call these  $u_0$  and  $v_0$ .

The resolution of the sensor also determines how the metric coordinates  $x$  and  $y$  get converted to pixels: we introduce two constants  $k$  and  $l$  which express the number of pixels per unit of  $f$  (e.g., pixels per mm. if  $f$  the focal length  $F$  is given in mm.) Hence we have, in image coordinates

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} + F \begin{bmatrix} kX/Z \\ lY/Z \end{bmatrix} = \begin{bmatrix} u_0 + \alpha X/Z \\ v_0 + \beta Y/Z \end{bmatrix} \quad (3)$$

where  $\alpha \doteq kF$  and  $\beta \doteq lF$ , both having units of pixels. In cameras with square pixels we have  $\alpha = \beta = f$ , where  $f$  is the **focal length expressed in pixels**.

## 1.3 Homogeneous Coordinates

We now introduce **homogeneous coordinates**, which allow us to do two cool things: (a) we will be able to talk points at infinity, and (b) we will be able to write the non-linear projection equation as a *linear* matrix-vector multiply. We do this both for 2D and 3D, and on the surface it just means using a third coordinate equal to 1, i.e.

$$p = (x, y) \rightarrow \tilde{p} = (x, y, 1)$$

and

$$P = (X, Y, Z) \rightarrow \tilde{P} = (X, Y, Z, 1)$$

where we use a tilde to indicate a homogeneous point representation. The cost of using homogenous coordinates is that they are not unique. Homogeneous points that only differ by a multiplicative factor are equivalent, i.e., they represent the same point. Hence, you can always write a finite 2D point as  $(x, y, 1)$ . For example, the homogeneous point  $(4, 6, 2)$  is the same as  $(2, 3, 1)$  in homogeneous coordinates, and represent the same 2D point  $(x, y) = (2, 3)$ . likewise, all of the points below represent the same 3D point

$$P_5 = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 1 \end{bmatrix} \equiv \begin{bmatrix} 4 \\ 4 \\ 8 \\ 4 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -2 \\ -1 \end{bmatrix}$$

where the symbol  $\equiv$  denotes **projective equivalence**.

Mathematically, what we are doing is representing 2D or 3D Euclidean space with 2D or 3D **projective space**, which extends the usual 3D space with a line or plane at infinity. In particular, any *finite* point in 2D or 3D has a non-zero value in the last slot:

$$p_1 = \begin{bmatrix} 2 \\ 4 \\ 2 \end{bmatrix} \equiv \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \quad P_1 = \begin{bmatrix} 8 \\ -6 \\ 10 \\ 2 \end{bmatrix} \equiv \begin{bmatrix} 4 \\ -3 \\ 5 \\ 1 \end{bmatrix}$$

are just a new way to represent the points  $(1, 2)$  and  $(4, -3, 5)$ , respectively in 2D and 3D. we should also mention two special cases: the 2D origin of the image plane is  $\tilde{c} = (0, 0, 1)$  in homogeneous coordinates, and the 3D origin of the camera coordinate frame (the optical center) is  $\tilde{O} = (0, 0, 0, 1)$ .

Points at *infinity* (mind blown!) are obtained by setting the last coordinate equal to zero, e.g.

$$p_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad p_3 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

are 2D points at infinity, in the direction of the  $u$  and  $v$  axis, respectively. Likewise, in 3D these are all points at infinity:

$$P_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad P_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad P_4 = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 0 \end{bmatrix} \quad P_5 = \begin{bmatrix} 3 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Above,  $P_1$  and  $P_2$  are “at the end of the  $X$  and  $Z$  axis, respectively, whereas  $P_3$  and  $P_4$  are at infinity in two different, arbitrary directions.

The recipe to recover non-homogeneous coordinates is to simply divide by the last coordinate, i.e.

$$\tilde{p} = (u, v, w) \rightarrow p = \left( \frac{u}{w}, \frac{v}{w} \right) \quad (4)$$

and

$$\tilde{P} = (X, Y, Z, T) \rightarrow P = \left( \frac{X}{T}, \frac{Y}{T}, \frac{Z}{T} \right). \quad (5)$$

This is another way to realize that projectively equivalent points, which only differ up to a scale, represent the same point. And, you can also see that dividing by zero yields infinite points.

## 1.4 Pinhole Model in Homogeneous Coordinates

But what about the promised linear projection equation? Well, This in turn allows us to write perspective projection in a linear way. Indeed, when we write

$$\tilde{p} = \begin{bmatrix} x \\ y \\ t \end{bmatrix} = \begin{bmatrix} 1 & & 0 \\ & 1 & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ T \end{bmatrix} = \begin{bmatrix} I & 0 \end{bmatrix} \tilde{P}$$

we get the following simple expression for the 2D homogeneous coordinates of the projection of the 3D point:

$$\tilde{p} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix},$$

In other words, if a 3D point has homogeneous coordinates  $(X, Y, Z, 1)$ , the homogeneous coordinates of the 2D projection are just  $\tilde{p} = (X, Y, Z)$ . Converting back to non-homogeneous coordinates via Equation 4 we get

$$p = \begin{bmatrix} X/Z \\ Y/Z \end{bmatrix}$$

i.e., this is the pinhole camera model from Equation 1!

What if we want to express  $p$  in image coordinates? Easy, we multiply on the left by another matrix that does the scaling by  $f$  (expressed in pixels) and translation to account for the non-zero image center:

$$\tilde{p} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f & & u_0 \\ & f & v_0 \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & & 0 \\ & 1 & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ T \end{bmatrix} = K \begin{bmatrix} I & 0 \end{bmatrix} \tilde{P} \quad (6)$$

where  $K$  is the  $3 \times 3$  **camera calibration matrix**. When you work through it, you will realize that we obtain

$$\tilde{p} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} u_0 + fX \\ v_0 + fY \\ Z \end{bmatrix} \rightarrow p = \begin{bmatrix} \frac{u_0 + fX}{Z} \\ \frac{v_0 + fY}{Z} \end{bmatrix}$$

Remarkably, points at infinity with  $T = 0$  also have an image which is typically finite: note that  $T$  does not figure in the equation above. In other words: all points on the same **viewing ray** have the same images, even points at infinity.

## 1.5 Projecting Points in the World

Is not remarkable in Equation 6 how we can use *multiplying* with a  $3 \times 3$  matrix  $K$  to implement scaling and translation in 2D? That brings us to the final transformation: what if points were not expressed in the camera coordinate frame  $C$  but instead in some world frame  $W$ ? To model this, we introduce the  $3 \times 3$  **camera rotation matrix**  $R_c^w$  which has as columns the axes of the camera frame, expressed in world coordinate frames. Likewise, we also introduce the camera translation  $t_c^w$ , expressed in world frame. With this, we can express any point  $\tilde{P}^c$  expressed in homogeneous camera coordinates in the world frame by multiplying with a following  $4 \times 4$  matrix:

$$\tilde{P}^w = \begin{bmatrix} R_c^w & t_c^w \\ 0 & 1 \end{bmatrix} \tilde{P}^c.$$

Substituting this into 6 we finally obtain

$$\tilde{p} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = K \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} R_c^w & t_c^w \\ 0 & 1 \end{bmatrix}^{-1} \tilde{P}^w \quad (7)$$

Because we have (trust us on this for now) that

$$\begin{bmatrix} R_c^w & t_c^w \\ 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} R_c^{wT} & -R_c^{wT} t_c^w \\ 0 & 1 \end{bmatrix}$$

we can also write the entire linear camera projection model as

$$\tilde{p} = K \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} R_c^{wT} & -R_c^{wT} t_c^w \\ 0 & 1 \end{bmatrix} \tilde{P}^w = K R_c^{wT} \begin{bmatrix} I & -t_c^w \end{bmatrix} \tilde{P}^w \quad (8)$$

The last formula on the right can be read, from right to left, as: (a) take  $\tilde{P}^w$ , (b) subtract the camera center  $t_c^w$ , (c) use  $R_c^{wT}$  to rotate into camera coordinates, and (d) calibrate to pixels using  $K$ , to (e) obtain the homogeneous coordinates  $\tilde{p}$  of the projected point.

As one final step, we can combine all these operations using a single  $3 \times 4$  **camera matrix**  $\mathcal{P}$ :

$$\tilde{p} = K R_c^{wT} \begin{bmatrix} I & -t_c^w \end{bmatrix} \tilde{P}^w = \mathcal{P} \tilde{P}^w \quad (9)$$