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

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

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

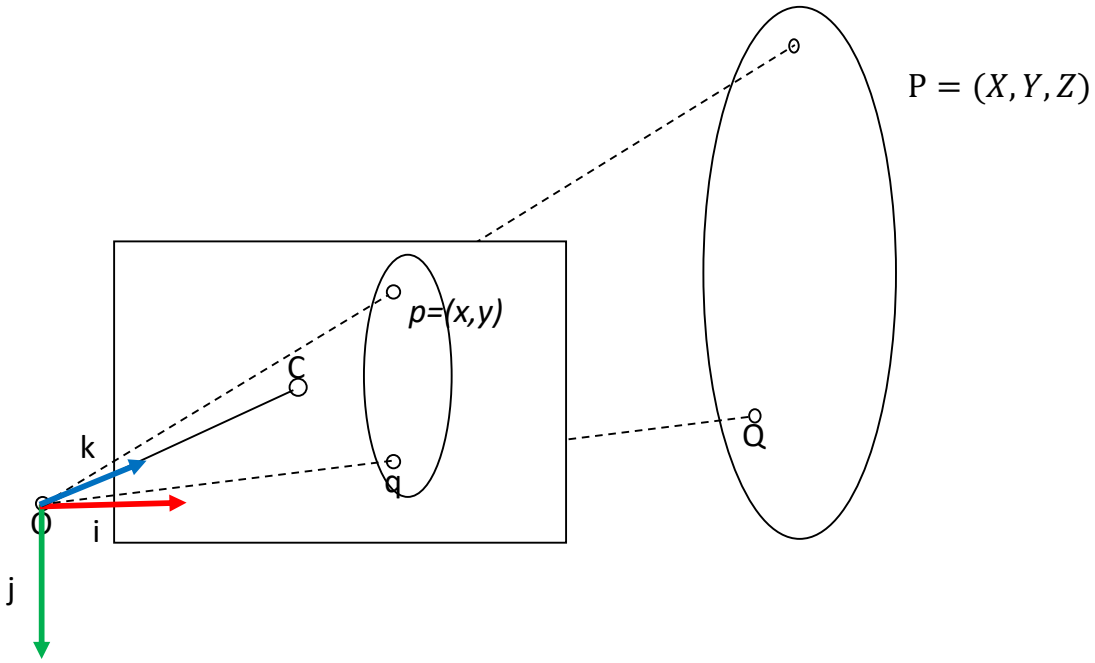

Figure 1: Pinhole camera model for projecting 3D points P in the scene to 2D points p in the image.

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:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} X/Z \\ Y/Z \end{bmatrix} \tag{1}$$

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.

The pinhole camera models that fact that 3D space is projected into 2D, and in particular we lose depth information. In camera coordinates, the Z coordinate of a 3D point is called the **depth** of the point. But Equation 1 shows that in the 2D projection, only the proportion of X resp. Y with respect to Z is retained: we cannot actually recover the actual 3D location of the point after projection. Indeed, if we have a point $P = (X, Y, Z)$, and multiply all coordinates by 2, the projection is the same:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2X/2Z \\ 2Y/2Z \end{bmatrix} = \begin{bmatrix} X/Z \\ Y/Z \end{bmatrix} \tag{2}$$

In general, all points on the **viewing ray** $\lambda P = (\lambda X, \lambda Y, \lambda Z)$, with λ an arbitrary scalar, will yield the same projection in the image.

1.2 Vanishing Points

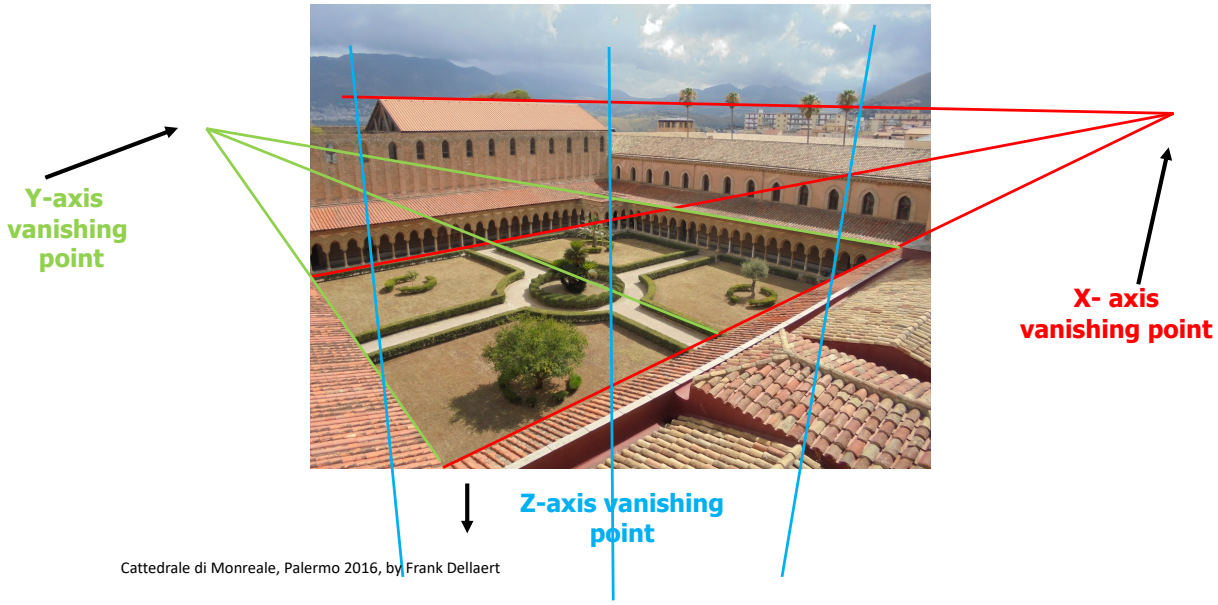

Figure 2: This image, taken in Palermo, Sicily, shows that parallel lines in the scene intersect when projected by a camera, although not necessarily within the image bounds.

In addition to losing the notion of depth, as discussed above, another property we lose projecting into an image is parallelism of lines. In other words: parallel lines in the world do not remain parallel in the image. This is illustrated vividly in Figure 2, where we have associated the three principal directions in the scene with the world X, Y, and Z axes, respectively.

Let us model this mathematically. Suppose an environment contains a collection of parallel lines such as in Figure 2. We can write the equations for these lines in parametric form as

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} + \gamma \begin{bmatrix} U_i \\ V_i \\ W_i \end{bmatrix}$$

The terms in this equation can be interpreted as follows:

- $P_i = (X_i, Y_i, Z_i)$ gives the coordinates for some point on the i^{th} line, specified with respect to the camera frame.
- $D_i = (U_i, V_i, W_i)$ is the direction of lines i, also specified with respect to the camera frame. Since the lines are parallel, they all have the same direction vector D_i.
- γ is a parameter that determines distance along the line from the point P_i.

Even though the lines are parallel in the world, due to the effects of perspective projection, the images of these lines will intersect, except when the lines are parallel to the image plane. This intersection point is known as the **vanishing point**. We can find the vanishing point by examining

the case where $\gamma \to \infty$. For a given line, if we let (u_∞, v_∞) denote the image plane coordinates for the vanishing point, we have

$$\begin{aligned} u_\infty &= \lim_{\gamma\to\infty} \lambda \frac{X}{Z} \\ &= \lim_{\gamma\to\infty} \frac{X+\gamma U}{Z+\gamma W} \\ &= \frac{U}{W} \end{aligned}$$

and by similar reasoning,

$$v_\infty = \frac{V}{W}$$

In other words, the projection of a vanishing point only needs the 3D direction (U, V, W), and all 3D lines along the same direction project to the same vanishing point $(u_\infty, v_\infty) = (U/W, V/W)$.

1.3 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

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = F \begin{bmatrix} X/Z \\ Y/Z \end{bmatrix} \tag{3}$$

In Equation 3 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 3 , 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

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} + F \begin{bmatrix} kX/Z \\ lY/Z \end{bmatrix} = \begin{bmatrix} u_0 + \alpha X/Z \\ v_0 + \beta Y/Z \end{bmatrix} \tag{4}$$

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**.

The set of parameters u_0, v_0, f are called **camera calibration parameters**, and these three (image center and focal length) are actually the most important for most cameras. Other parameters include the aspect ratio $a = \alpha/\beta$, which is 1 if the cameras has square pixels (and most of them do) and the skew s, which we will ignore here. What can be important in wide-angle lenses are additional distortion parameters κ that model the radial distortion that gets increasingly worse near the edges of the image.

1.4 Homogeneous Coordinates

We now introduce **homogeneous coordinates**, which allow us to do two cool things: (a) we will be able to talk about 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

$$P_0 = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 1 \end{bmatrix} \equiv \begin{bmatrix} 4 \\ 4 \\ 8 \\ 4 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -2 \\ -1 \end{bmatrix}$$

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

In particular, any *finite* point in 2D or 3D has a non-zero value in the last slot,

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

and 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.

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

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

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

Above, P_2 and P_3 are "at the end of the X and Z axis, respectively, whereas P_4 and P_5 are at infinity in two different, arbitrary directions.

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 2D, the line at infinity extends the real plane to include points at infinity in a manner analogous to adding $+\infty$ and $-\infty$ to the set of real numbers to obtain the extended real number system. In 3D, we actually have a plane at infinity: indeed, the three first coordinates of all points at infinity are 2D homogeneous coordinates!

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

$$\tilde{p} = (u, v, w) \rightarrow p = \left(\frac{u}{w}, \frac{v}{w}\right) \tag{5}$$

and

$$\tilde{P} = (X, Y, Z, T) \rightarrow P = \left(\frac{X}{T}, \frac{Y}{T}, \frac{Z}{T}\right). \tag{6}$$

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.

1.5 Pinhole Model in Homogeneous Coordinates

But what about the promised linear projection equation? Well, Equation 5 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 5 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:

$$\tilde{p} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f & & u_0 \\ & f & v_0 \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & 0 \\ & 1 & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ T \end{bmatrix} = K \begin{bmatrix} I & 0 \end{bmatrix} \tilde{P} \tag{7}$$

Above K is the 3×3 **camera calibration matrix**. When you work through it, you will realize that we obtain

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

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. This agrees with what we found in Section 1.1.

1.6 Projecting Points in the World

Is it not remarkable in Equation 7 how we can use *multiplying* with a 3×3 matrix K to implement scaling and translation in 2D? That brings us to the final transformation: what if 3D points were not expressed in the camera coordinate frame C but instead in some world frame W? To model this, we introduce the 3×3 **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 the 4×4 matrix below:

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

Substituting this into 7 we finally obtain

$$\tilde{p} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = K \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} R_c^w & t_c^w \\ 0 & 1 \end{bmatrix}^{-1} \tilde{P}^w \tag{8}$$

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

$$\tilde{p} = K \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} R_c^{wT} & -R_c^{wT} t_c^w \\ 0 & 1 \end{bmatrix} \tilde{P}^w = K R_c^{wT} \begin{bmatrix} I & -t_c^w \end{bmatrix} \tilde{P}^w \tag{9}$$

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 3×4 **camera matrix** $\mathcal{P}$:

$$\tilde{p} = K R_c^{wT} \begin{bmatrix} I & -t_c^w \end{bmatrix} \tilde{P}^w = \mathcal{P} \tilde{P}^w \tag{10}$$

Worked Example

If you take a picture with your phone while looking in the direction of the world X axis, and assuming the world Z axis is pointing up (very common convention) then the camera rotation matrix will be given by

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

In addition, suppose we defined the world coordinate frame such that we are standing at location $(X^w, Y^w) = (20, -5)$, and the camera is at eye height, say $Z^w = 1.5$ (all quantities in meter). Then we also have

$$t_c^w = \begin{bmatrix} 20 \\ -5 \\ 1.5 \end{bmatrix}$$

Finally, if we assume a typical smartphone calibration, e.g., an iPhone 5, we will have $(u_0, v_0) \approx (1600, 1200)$ and $f \approx 3000$, all in pixels. we then have

$$\mathcal{P} \doteq K R_c^{wT} \begin{bmatrix} I & -t_c^w \end{bmatrix} = \begin{bmatrix} 3000 & & 1600 \\ & 3000 & 1200 \\ & & 1 \end{bmatrix} \begin{bmatrix} & -1 & \\ & & -1 \\ 1 & & \end{bmatrix} \begin{bmatrix} 1 & & -20 \\ & 1 & 5 \\ & & 1 & -1.5 \end{bmatrix}$$

$$= \begin{bmatrix} 1600 & -3000 & & -47000 \\ 1200 & & -3000 & -19500 \\ 1 & & & -20 \end{bmatrix}$$

Projecting some points is informative. Looking towards the optical axis, we see the point at infinity in the direction of the axis projects to

$$\mathcal{P} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1600 \\ 1200 \\ 1 \end{bmatrix}$$

i.e., the image center, as expected. The two other main vanishing points, in the Y and Z directions beget

$$\mathcal{P} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -3000 \\ 0 \\ 0 \end{bmatrix} \equiv \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad \mathcal{P} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ -3000 \\ 0 \end{bmatrix} \equiv \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

This also makes sense: they are points at infinity in the image, respectively in the horizontal and vertical directions. Finally, let's project a point on the ground plane, 10 meters in the X direction:

$$\mathcal{P} \begin{bmatrix} 30 \\ -5 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1600 \\ 1650 \\ 1 \end{bmatrix}$$

which projects below the image center in the image (remember v increases downward).

Exercise

Define a large square on the ground plane, centered around the ground plane point above, and project that in the image. Sketch your solution.