World vs. camera coordinates

Up to now, we have defined 3D points in the camera coordinate system. Often we would also like to express the position of points (X, Y, Z) in a world coordinate system which is independent of the camera. This should make sense intuitively: we would like to be able to speak about the positions of objects independently of where the camera is.

If we are going to express point positions both in camera and in world coordinates, then we will need a way of transforming between these coordinate systems. We need to define a transformation from point (X_w, Y_w, Z_w) in world coordinates to its location (X_c, Y_c, Z_c) in the camera coordinates. We will use 4×4 translation and rotation matrix to do so.

Camera extrinsic (or external) parameters

Suppose the position of the camera's center in world coordinates is a 3D point $\mathbf{C}_w$. If we wish to transform any other point $\mathbf{X}_w$ into the camera's coordinate system, we first subtract off $\mathbf{C}_w$ and then we perform a rotation:

$$\mathbf{X}_c = \mathbf{R}(\mathbf{X}_w - \mathbf{C}_w) \ .$$

The matrix $\mathbf{R}$ relates the coordinate axes of the world coordinates and camera coordinates, namely its rows are the camera coordinate axes when written in the world coordinate system. You can think of this rotation matrix as as $\mathbf{R}_{c \leftarrow w}$. The rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{C}_w$ define the *camera's extrinsic coordinates* (or *external parameters*), namely its orientation and position, respectively, in world coordinates.

Using 3D vectors and 3×3 matrices, we can write the transformation as follows,

$$\mathbf{X}_c = \mathbf{R}\mathbf{X}_w - \mathbf{R}\mathbf{C}_w.$$

In homogeneous coordinates, we would write

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & -\mathbf{R}\mathbf{C}_w \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} .$$

Alternatively, the transformation from world to camera coordinates is the product of a 4×4 rotation matrix and a 4×4 translation matrix:

$$\begin{bmatrix} \mathbf{R} & -\mathbf{R}\mathbf{C}_w \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I} & -\mathbf{C}_w \\ 0 & 1 \end{bmatrix}$$

where $\mathbf{I}$ is the 3×3 identity matrix.

Camera intrinsics (internal parameters)

Assume we now have any scene point in camera coordinates. We next project from camera coordinates into the image plane. Recall from lecture 12 that we defined the projection from 3D (X, Y, Z) into the 2D coordinates (x, y) via

$$(x, y) = (f\frac{X}{Z}, f\frac{Y}{Z}).$$

Let's rewrite this 2D point in homogeneous coordinates:

$$(x, y, 1) = (f\frac{X}{Z}, f\frac{Y}{Z}, 1)$$

If $Z \neq 0$ then

$$(f\frac{X}{Z}, f\frac{Y}{Z}, 1) \equiv (fX, fY, Z).$$

This equivalence allows us to write the projection from a 3D point (X, Y, Z) – which we are assuming is now in camera coordinates – to its 2D image point (x, y), using a 3×4 matrix:

$$\begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Note that the fourth column doesn't contribute anything. For this reason it is common to use a 3×3 matrix rather than a 3×4 and to write the projection transformation as follows:

$$\begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Transforming to pixel coordinates

When we are working with real cameras, typically we do not want to index points on the projection plane by their actual position (measured in say millimeters). Rather we index points by pixel coordinates. Transforming from real physical positions to pixel positions involves a scaling and a translation, as we'll see below. Let's first deal with the scaling.

Let's write out the transformation from the assumed mm units on the projection plane to pixel units. We multiply the x, y values by the *number of pixels per mm* in the x and y direction, respectively. These scale factors are denoted m_x, m_y. Surprisingly, m_x and m_y are not always exactly the same value, though they are usually very close. The projection followed by scaling transformation is now:

$$\begin{bmatrix} m_x & 0 & 0 \\ 0 & m_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

which can be written

$$\begin{bmatrix} fm_x & 0 & 0 & 0 \\ 0 & fm_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Again, note that the 4^{th} column can be dropped if we use (X, Y, Z) for the scene point (in camera coordinates) rather than the homogeneous coordinate representation $(X, Y, Z, 1)$.

We mentioned above that it is also necessary to do a translation of the pixel positions. To understand why, note that the position of the optical center, $(x, y) = 0$, on the sensor typically doesn't correspond exactly to physical center of the image sensor. (The position of the optical

center is called the *principal point*.) In particular, the principal point doesn't necessarily correspond exactly to the center of the pixel grid. For example, if the grid had 1536 rows and 2048 columns, then the center of the pixel grid would be halfway between row 767 and 768 and column 1023 and 1024 (assuming we start our count from 1). Moreover, because of variability in the manufacturing process, the principal point might be shifted slightly.

Let (p_x, p_y) be the position of the principal point *in pixel coordinates*. Then, in order for the principal point $(x, y) = (0, 0)$ to map to (p_x, p_y), we need to translate:

$$\begin{bmatrix} 1 & 0 & p_x \\ 0 & 1 & p_y \\ 0 & 0 & 1 \end{bmatrix}$$

This gives us a sequence (projection, scaling, translation):

$$\begin{bmatrix} fm_x & 0 & p_x & 0 \\ 0 & fm_y & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & p_x \\ 0 & 1 & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} m_x & 0 & 0 \\ 0 & m_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

A few more details...

- The projection model assumes that we are projecting onto a plane $Z = f$ in front of the center of projection. In fact, however, if we are approximating the geometry as a *pinhole* camera (with the origin being the pinhole), then the sensor plane is behind the camera so it should have a negative value. The way around this is to think of the sensor plane behind the pinhole at $Z = -f$, but we are flipping the image x and y axes.

- While the above model typically describes all the parameters needed to transform from a scene point in camera coordinates into the projected image point in pixel coordinates, one sometimes adds another parameter called the *skew parameter* s. So a more general transformation is:

$$\mathbf{K} \equiv \begin{bmatrix} fm_x & s & p_x \\ 0 & fm_y & p_y \\ 0 & 0 & 1 \end{bmatrix}$$

  This matrix $\mathbf{K}$ is commonly called the *camera calibration matrix*. In a few lectures from now, we will show how to estimate this matrix.

  Notice that the parameters f, m_x, m_y only appear as two products in this matrix, so when we try to estimate these parameters there is no way to disentangle them. Therefore, sometimes one writes $\alpha_x = fm_x$ and $\alpha_y = fm_y$, so

$$\mathbf{K} = \begin{bmatrix} \alpha_x & s & p_x \\ 0 & \alpha_y & p_y \\ 0 & 0 & 1 \end{bmatrix}.$$

- We now have a transformation takes a point $(X, Y, Z, 1)$ in world coordinates to a pixel position (wx, wy, w). One often writes this transformation as:

$$\mathbf{P} \;=\; \mathbf{K}\,\mathbf{R}\,[\,\mathbf{I}\,|\,-\mathbf{C}\,]$$

where $\mathbf{K}$, $\mathbf{R}$, and $\mathbf{I}$ are 3×3, and where the matrix on the right is 3×4, and the symbol $|$ is used to separate the 3×3 identify matrix $\mathbf{I}$ from the vector $-\mathbf{C}$, as opposed to a difference **I - C** which would make no sense. This transformation is sometimes called a *finite projective camera*. It describes the mapping from points in world coordinates to points in pixel coordinates.

