Homogenous coordinates in 2D

One can use homogenous coordinates to represent points (and points at infinity in any n-D space. For example, we will often use homogeneous coordinates for 2D spaces. Suppose we wish to translate all points (x, y) by adding some constant vector (t_x, t_y). This transformation cannot be achieved by a 2×2 matrix, so we tack on a third coordinate with value 1 $(x, y, 1)$, and translate by performing a matrix multiplication:

$$\begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}.$$

We can perform rotations using a 3×3 matrix as well, namely a 2D rotations matrix goes into the upper-left 2×2 corner:

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and we can also scale the two coordinates:

$$\begin{bmatrix} \sigma_x x \\ \sigma_y y \\ 1 \end{bmatrix} = \begin{bmatrix} \sigma_X & 0 & 0 \\ 0 & \sigma_Y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}.$$

And we can also perform a shear

$$\begin{bmatrix} x + sy \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & s & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}.$$

The arguments at the end of the lecture 3 notes about points at infinity are exactly the same for 2D points (x, y). Consider a point (x, y, ϵ) in homogeneous coordinate representation, and ask what happens as $\epsilon \to 0$. We get $(\frac{x}{\epsilon}, \frac{y}{\epsilon}, 1)$ which is in the direction of (x, y) but which goes to an infinite distance from the origin as $\epsilon \to 0$. Thus, $(x, y, 0)$ represents a 2D point at infinity in direction of vector (x, y).

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:

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

Use 3D vectors and 3×3 matrices, we can write this as follows,

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

In homogeneous coordinates, we would write

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

so the transformation from world to camera coordinates is the product of a 4×4 translation matrix and a 4×4 rotation 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.

The rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{C}_w$ define the camera's extrinsic coordinates, namely its orientation and position, respectively, in world coordinates. The matrix $\mathbf{R}$ transforms from world to camera coordinates, and so you can think of it as $\mathbf{R}_{c \leftarrow w}$.

Projection matrix (3D to 2D)

We next project from camera coordinates into the image plane. Recall from lecture 1 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 now rewrite this 2D point in homogeneous coordinates:

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

and note that 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 is in camera coordinates) to its 2D image point (x, y), using a 3×4 matrix:

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

The projection matrix is not invertible. It is of rank 3. In particular, because the matrix above maps a 4D space to a 3D space, there must be some non-zero vector that maps to $(0, 0, 0)$. This vector defines the *null space* of the projection. What is this null vector? Answer: by inspection, it is the center of projection (0,0,0,1).

Note that the point $(0, 0, 0)$ is undefined, when it is interpreted as a 2D point in homogeneous coordinates. It is neither a finite point $(x, y, 1)$, nor a point at infinity $(x, y, 0)$ since at least one of x or y would need to be non-zero to have a well defined point at infinity.

Camera intrinsic (or internal) parameters

When we are working with real cameras, typically we do not want to index points on the projection plane by their actual position on the projection plane (measured in say millimeters). Rather we want to index points by pixel numbers. Transforming from real physical positions to pixel positions involves a scaling (since the width of a pixel is typically not a standard unit like 1 mm) and a translation (since the principal point[1] typically doesn't correspond exactly to center of the image sensor).

The sensor is made up of a square grid of sensor elements (pixels). We can talk about the position which is the center of this grid. For example, if the grid had 1536 rows and 2048 columns, then we would be talking about the position on the sensor that is halfway between row 767 and 768 and column 1023 and 1024 (assuming we start our count from 0).

We define a coordinate system on the grid in whatever units we want, say millimeters. We define the coordinate system such that the origin is at center of the grid and the (u, v) axes are parallel to the camera's x, y axes, respectively.

We have expressed the position on the sensor plane in physical coordinates, in units of millimeters. What we want instead is to express it in terms of pixel units. We can do this by scaling. We multiply the x, y coordinates by the *number of pixels per mm* in the x and y direction. These scale factors are denoted m_x, m_y. Surprisingly, these are not always exactly the same value, though they are usually very close. The projection and scaling transformation is now:

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

The transformation gives positions in terms of pixel units. However, it does not correspond to the pixel indices used in an image. Typically the index (0,0) is in the corner of the image. Thus we need another transformation, namely a translation, that gives the correct pixel coordinates of the principal point. Note: the principal point might not correspond exactly to the center of the sensor. You would think it should, but keep in mind that pixels are very small and the placement of the sensor relative to the lens (which defines the optical axis and hence principal point) might not be as accurate as you hope.

Let (p_x, p_y) be the position of the principal point *in pixel coordinates*. It may indeed be at the center of the image, i.e. at position (*,*) as mentioned above. But it may be at a slightly different position too. Then, to get the position of a projected point (x, y) in pixel coordinates, we need to

[1]The *principal point* is the intersection of the optical axis with the image plane

add a translation component which is the pixel position of the principal point:

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

A few more details...

- While the above model typically describes all the parameters needs 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 projection transformation is:

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

- It is common to give the 3×3 matrix on the left side a special name

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

  is called the *camera calibration matrix.* In part 3 of this course, 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. Sometimes one writes

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

- 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:

$$\mathbf{P} = \mathbf{KR}[\,\mathbf{I} \mid -\mathbf{C}]$$

  where $\mathbf{K}$, $\mathbf{R}$, and $\mathbf{I}$ are 3×3, and where the matrix on the right is 3×4. This transformation is sometimes called a *finite projective camera.*