# Geometry of image formation

Bharath Hariharan

March 25, 2020

## 1 Basic perspective projection

Let us first revisit basic perspective projection (Figure 1). This section is a repeat of our first image formation lecture.

The word camera comes from the latin phrase *camera obscura*, literally meaning *dark chamber*. The camera obscura is a dark chamber with a small hole in one wall. On the wall opposite the hole, we see an inverted image of the world outside. Today such a camera would be called a pinhole camera.

Why does a pinhole camera work? It is because the opposite wall can only receive light through the pinhole. Because the pinhole is very small, for every point on the opposite wall, there is only a single ray of light that comes from the outside world, through the pinhole and lands at that point. If we want to figure out where a given world point is imaged on the opposite wall, we can simply draw a straight line from the world point through the pinhole and see where it hits the wall.

Let us formalize this mathematically. We place the origin of our coordinate system at the pinhole. We will let the $Z$ axis point outwards from the pinhole, and the $Y$ axis point upwards. Finally, we will choose our units so that the wall (the *image plane*) is a unit distance away from the pinhole. Thus the equation of the image plane is $Z = -1$.

We will also place a similar coordinate system on the image plane. The $X$ and $Y$ axes of the 2D image coordinate system is aligned with the $X$ and $Y$ axis of the 3D coordinate system. The origin of the image coordinate system is where the $Z$ axis intersects the image plane.

Now let us consider a point $\mathbf{P} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$ in the 3D world. [1] We want to figure out where it projects in the image. To figure this out, we consider the line $PO$ joining $\mathbf{P}$ to the origin $\mathbf{O}$ and ask where this line intersects the image plane. Points on this line can be represented as $\mathbf{Q}(\lambda) = \mathbf{O} + \lambda(\mathbf{P} - \mathbf{O}) = \begin{bmatrix} \lambda X \\ \lambda Y \\ \lambda Z \end{bmatrix}$. (To see this, observe that $\mathbf{Q}(0) = \mathbf{O}$ and $\mathbf{Q}(1) = \mathbf{P}$; other values of $\lambda$ yield points on this line.) For this point

[1] We will express points in 3D using capital letters, and vectors using bold font

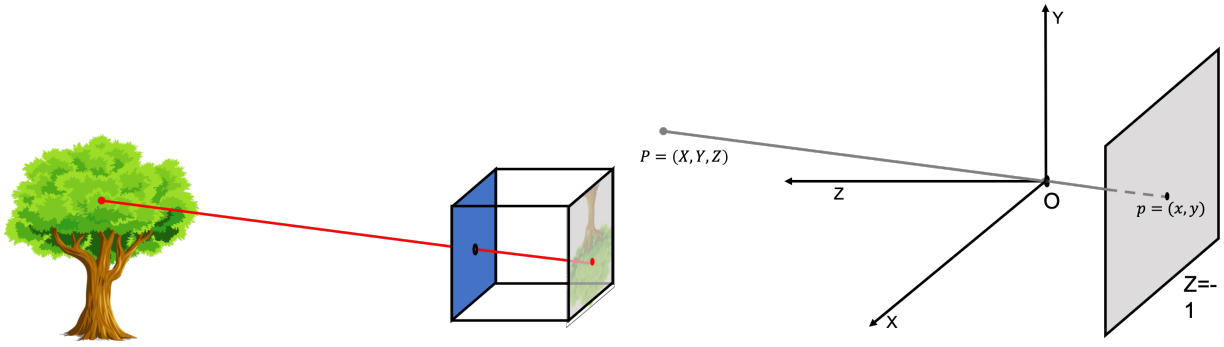


Figure 1: **Left**: Pinhole camera. **Right**: Mathematical abstraction of a pinhole camera

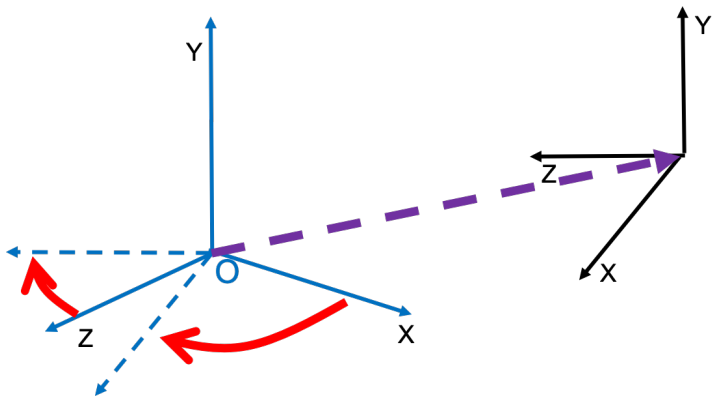


Figure 2: Transformations needed for converting from world 3D coordinates to image 3D coordinates: a rotation (red) to align axes, and a translation (purple) to move the origin

to lie on the image plane, we require that the $Z$ coordinate of this point should be $-1$. This implies that $\lambda Z = -1 \Rightarrow \lambda = -\frac{1}{Z}$. This implies that the X and Y coordinates of the image point must be $\frac{-X}{Z}, \frac{-Y}{Z}$. Thus if the image point is $\mathbf{p} = \begin{bmatrix} x \\ y \end{bmatrix}$, then:

$$x = \frac{-X}{Z} \tag{1}$$

$$y = \frac{-Y}{Z} \tag{2}$$

This gives us an inverted image. We can get an upright image by inverting this image back. Mathematically, we can consider a *virtual* image plane that is in front of the pinhole, with an equation given by $Z = 1$. With this virtual image plane, the equations for perspective projection become:

$$x = \frac{X}{Z} \tag{3}$$

$$y = \frac{Y}{Z} \tag{4}$$

# 2 Incorporating coordinate system changes

## 2.1 World coordinate system

Above, we assumed that 3D points were expressed in a coordinate system centered on the camera pinhole, and aligned with the camera viewing direction. However, in general we may have a different world coordinate system for expressing 3D points. This is especially useful if the camera moves: we want the world points expressed in a fixed coordinate frame.

In this case, before we apply equations (3) and (4) we want to perform a change of coordinates from the world coordinate system to the camera coordinate system (centered on the pinhole with the Z axis pointing along the viewing direction. In general, this would require two steps. First, we would need to *rotate* to align the axes, and second, we will need to *translate* to move the origin to the pinhole (Figure 2). Thus, if $\mathbf{P} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$ is the representation of a point in the world coordinate system, and $\mathbf{P'} = \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix}$ is its representation in the camera coordinate system,

$$\mathbf{P'} = R\mathbf{P} + \mathbf{t} \tag{5}$$

Here $R$ is a rotation matrix and $\mathbf{t}$ is a translation vector.

Figure 3: The image coordinate system must be changed from the one shown in red, to the one shown in cyan.

## 2.2 A detour into rotations

Above the vector $\mathbf{t}$ can be any arbitrary vector. However, $R$ represents a rotation matrix and must therefore satisfy some special properties.

In particular, $R$ is an *orthonormal matrix*. This means that $R^T R = I$, where $I$ is the identity matrix. If the columns of $R$ are $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3$, then this orthonormality criterion implies that $\mathbf{r}_i^T \mathbf{r}_i = 1$, and $\mathbf{r}_i^T \mathbf{r}_j = 0$ for $i \neq j$. In other words, the columns of a rotation matrix all have unit length and are orthogonal to each other.

However, not all orthonormal matrices are rotation matrices. For orthonormal matrices, the determinant can be either 1 or $-1$. Rotation matrices have determinant 1. Matrices with determinant $-1$ include reflections, and transformations from left-handed to right-handed coordinate systems or vice-versa.

Finally, rotation matrices in 3D always involve rotation about *an axis by an angle* $\theta$. Points $\mathbf{v}$ on the axis are those that are unchanged by the rotation. Thus $R\mathbf{v} = \mathbf{v}$. Given the axis and the angle, it is possible to construct the rotation matrix using *Rodriguez' formula*, but we will not explore the construction here.

## 2.3 Image coordinate systems

Just as the world coordinate system might be different, the image coordinate system may also be different. In particular, the origin of the image coordinate system might be at the bottom-left corner of the image, for instance. The *units* are also different: image coordinates are typically in terms of *pixels*.

Thus, after the perspective projection in equations 3 and 4, we also need to perform a separate coordinate transformation on the image side. This will involve a *translation* of the origin, and a *scaling* to change the units:

$$x' = fx + u \tag{6}$$
$$y' = fy + v \tag{7}$$
$$\tag{8}$$

Putting everything together, we have the following steps necessary to project a 3D point $\mathbf{P}$ in the world to the image:

$$\mathbf{P}' = R\mathbf{P} + \mathbf{t} \qquad \text{Change 3D coordinates} \qquad (9)$$

$$x = \frac{X}{Z} \qquad \text{Perspective projection} \qquad (10)$$
$$y = \frac{Y}{Z}$$

$$x' = fx + u \qquad \text{Change image coordinates} \qquad (11)$$
$$y' = fy + v$$

# 3 Homogenous coordinates

Observe that the transformation from 3D points to 2D image locations is *not* a linear transformation, and so cannot be represented as a matrix multiplication. This is a problem, because we cannot use the considerable mathematical machinery we have for matrices and linear transformations. However, we can change our representation of points to make this transformation linear.

To motivate this new representation, we hark back to the original discussion of perspective projection in Section 1. Observe from that discussion that *associated with every 2D image location, there is a single 3D ray*: this is the ray that connects the image point with the pinhole and extends out into the world. Thus, we can represent image points using the corresponding 3D rays through the origin. But how do we represent rays?

We will represent 3D rays using *any* point on this ray. This is of course a non-unique representation. In particular $\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$ and $\begin{bmatrix} \lambda X \\ \lambda Y \\ \lambda Z \end{bmatrix}$, $\lambda \neq 0$ are both points on the same ray (see Section 1) and so both represent the same 3D ray. In other words, we represent 3D rays using 3 coordinates, but *multiplying all coordinates with the same non-zero scalar* results in the same ray.

Let us now get back to image points. Consider an image point $\mathbf{p} = \begin{bmatrix} x \\ y \end{bmatrix}$. The image plane has equation $Z = 1$. Thus a point on the corresponding 3D ray is $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$. Multiplying all coordinates by a non-zero scalar $\lambda$ gives us another point on the ray $\begin{bmatrix} \lambda x \\ \lambda y \\ \lambda \end{bmatrix}$. We will *represent* the image point $\mathbf{p}$ using this 3-coordinate representation, $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$, but with the understanding that this representation is *equivalent* to $\begin{bmatrix} \lambda x \\ \lambda y \\ \lambda \end{bmatrix}$. We call this non-unique 3-coordinate representation *homogenous coordinates*, and we will call the traditional 2-coordinate representation *Euclidean coordinates*. Thus a point $\begin{bmatrix} x \\ y \end{bmatrix}$ in Euclidean coordinates is represented in homogenous coordinates as $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$, or $\begin{bmatrix} 2x \\ 2y \\ 2 \end{bmatrix}$, or in general $\begin{bmatrix} \lambda x \\ \lambda y \\ \lambda \end{bmatrix}$, $\lambda \neq 0$.

The above suggests that to convert from Euclidean coordinates to homogenous coordinates, we can simply append a 1. Let us see how we can do the reverse. Suppose we are given the homogenous coordinate representation of a point as $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$. We know that we can scale all coordinates by the same scale factor

without changing the point. So the homogenous coordinates $\begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix}$ represent the same point. The previous discussion suggests that the point represented by $\begin{bmatrix} x/z \\ y/z \end{bmatrix}$ in Euclidean coordinates will have this representation in homogenous coordinates. Thus, to convert from homogenous coordinates to Euclidean coordinates, we divide by the last coordinate (here $z$) and then drop the last coordinate.

Above we have denoted points using boldface $\mathbf{p}$. We will restrict this notation for the Euclidean coordinate representation of points. We will denote the corresponding homogenous coordinate representation as $\vec{\mathbf{p}}$.

Thus:

1. Homogenous coordinates are a different representation for points.
2. To convert the usual Euclidean coordinates $\mathbf{p}$ into homogenous coordinates, we append a 1: $\vec{\mathbf{p}} \equiv \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix}$.
3. The homogenous coordinate representation is non-unique. In particular $\vec{\mathbf{p}} \equiv \lambda\vec{\mathbf{p}}$, where $\lambda \neq 0$. Therefore in homogenous coordinates we will always talk of *equivalence* ($\equiv$) and not equality ($=$).
4. To convert homogenous coordinates back to Euclidean coordinates, we simply divide by the last coordinate and then drop the last coordinate.

Note that we can use homogenous coordinates for 3D points too. 3D points will have 4 homogenous coordinates. All the above bullets apply.

# 4 Transformations in homogenous coordinates

Let us now look at the different transformations and see how they are represented in homogenous coordinates.

**Rotation and other linear transformations** : In Euclidean coordinates, rotation and other linear transformations are represented using a matrix multiplication.

$$\mathbf{Q} = R\mathbf{P} \tag{12}$$

What about in homogenous coordinates? We claim that multiplication by the matrix $\tilde{R} = \begin{bmatrix} R & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}$ will perform the same operation in homogenous coordinates. To see this, observe that:

$$\tilde{R}\vec{\mathbf{P}} \equiv \begin{bmatrix} R & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{P} \\ 1 \end{bmatrix} \tag{13}$$

$$\equiv \begin{bmatrix} R\mathbf{P} + \mathbf{0}1 \\ \mathbf{0}^T\mathbf{P} + 1 \cdot 1 \end{bmatrix} \tag{14}$$

$$\equiv \begin{bmatrix} R\mathbf{P} \\ 1 \end{bmatrix} \tag{15}$$

To convert this result to Euclidean coordinates, we divide by the last coordinate and drop the last coordinate. This gives us $R\mathbf{P}$, which exactly matches the result of the transformation in Euclidean coordinates.

**Translation** : In Euclidean coordinates, translation is addition:

$$\mathbf{Q} = \mathbf{P} + \mathbf{t} \tag{16}$$

We claim that in homogenous coordinates, multiplication by the matrix $\tilde{T} = \begin{bmatrix} I & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$ will perform the same operation. To see this, observe that:

$$\tilde{T}\vec{\mathbf{P}} \equiv \begin{bmatrix} I & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{P} \\ 1 \end{bmatrix} \tag{17}$$

$$\equiv \begin{bmatrix} I\mathbf{P} + \mathbf{t}1 \\ \mathbf{0}^T\mathbf{P} + 1 \cdot 1 \end{bmatrix} \tag{18}$$

$$\equiv \begin{bmatrix} \mathbf{P} + \mathbf{t} \\ 1 \end{bmatrix} \tag{19}$$

Again, converting this to Euclidean coordinates requires dividing by the last coordinate and dropping the last coordinate, yielding $\mathbf{P} + \mathbf{t}$.

**Perspective projection** : Recall that in Euclidean coordinates, perspective projection involves the following operation. For a point $\mathbf{P} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$, perspective projection involves the following transformation:

$$x = \frac{X}{Z} \tag{20}$$

$$y = \frac{Y}{Z} \tag{21}$$

We posit that in homogenous coordinates, this can be achieved by multiplying with the matrix $\tilde{P} = \begin{bmatrix} I & \mathbf{0} \end{bmatrix}$. Note that in this case the matrix is a $3 \times 4$ matrix because we are transforming 3D points to 2D points. Observe that:

$$\tilde{P}\vec{\mathbf{P}} \equiv \begin{bmatrix} I & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{P} \\ 1 \end{bmatrix} \tag{22}$$

$$\equiv I\mathbf{P} + \mathbf{0}1 \tag{23}$$

$$\equiv \mathbf{P} \tag{24}$$

$$\equiv \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{25}$$

To convert this to Euclidean coordinates, we divide by the last coordinate and drop the last coordinate, yielding $\begin{bmatrix} \frac{X}{Z} \\ \frac{Y}{Z} \end{bmatrix}$.

**Scaling and translation** : It should be fairly clear that scaling and translation (as in the final step of camera projection) will also be expressible as a linear matrix transformation in homogenous coordinates. In particular, the operation described in equation 11 can be performed in homorgenous coordinates by multiplying with the matrix $K = \begin{bmatrix} f & 0 & u \\ 0 & f & v \\ 0 & 0 & 1 \end{bmatrix}$:

$$K\vec{\mathbf{p}} \equiv \begin{bmatrix} f & 0 & u \\ 0 & f & v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \equiv \begin{bmatrix} fx + u \\ fy + v \\ 1 \end{bmatrix} \tag{26}$$

# 5 Final camera projection equation

We can put all this together to represent the final camera projection in homogenous coordinates. We will do the following in order:

1. Rotate by $R$ to align world coordinate axes with the camera coordinate axes
2. Translate by $\mathbf{t}$ to bring the world coordinate system origin to the camera pinhole
3. Perform perspective projection
4. Scale and translate the image coordinates.

Using the above results, we can do this by putting together a sequence of linear transformations:

$$\vec{\mathbf{p}} \equiv K \underbrace{\begin{bmatrix} I & \mathbf{0} \end{bmatrix}}_{\text{Perspective}} \underbrace{\begin{bmatrix} I & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}}_{\text{Translation}} \underbrace{\begin{bmatrix} R & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}}_{\text{Rotation}} \vec{\mathbf{P}} \tag{27}$$

$$\equiv K \begin{bmatrix} I & \mathbf{0} \end{bmatrix} \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \vec{\mathbf{P}} \tag{28}$$

$$\equiv K \begin{bmatrix} R & \mathbf{t} \end{bmatrix} \vec{\mathbf{P}} \tag{29}$$

Here $\vec{\mathbf{P}}$ is the world point in homogenous coordinates, $\vec{\mathbf{p}}$ is the corresponding image location, and $K$ is the matrix in equation (26).

Thus, the action of a camera is given by a $3 \times 4$ *camera projection matrix* $K \begin{bmatrix} R & \mathbf{t} \end{bmatrix}$:

$$\boxed{\vec{\mathbf{p}} \equiv K \begin{bmatrix} R & \mathbf{t} \end{bmatrix} \vec{\mathbf{P}}} \tag{30}$$

Note that often $K$ is taken to be an arbitrary invertible $3 \times 3$ matrix (and not necessarily a scaling and a translation).

Equation 30 suggests that, to know where a world point projects onto the image, we need to know $K, R$ and $\mathbf{t}$, which are called *camera parameters*. $R$ and $\mathbf{t}$ are dependent on the camera's location and orientation in the 3D world. In contrast, $K$ captures manipulations of the image coordinates and is therefore independent of the camera position and orientation. As such, $R$ and $\mathbf{t}$ are called *extrinsic* camera parameters, while $K$ is called an intrinsic camera parameter.