Computation & Robot Vision: Lecture 2, Camera parameters

Sam Barrett

March 4, 2021

Digital images are fundamentally **spatially discrete**, meaning they are divided up into a countable number of subsections. Usually, these subsections take the form of rectangular picture elements or *pixels*.

Note: the perspective equation derived in the previous lecture is only valid when all distances are measured from the camera's reference frame and when image coordinates have their origin at the image centre, where the axis of symmetry of the camera pierces the retina or sensor.

In practice, the world and camera coordinate systems are related by a set of physical parameters including:

- The focal length f of the the lens
- the size of the individual pixels on the sensor.
- the position of the image centre
- the position and orientation of the camera.

There are a number of issues with this system however, including that one unit in the camera's coordinate system may not be the same as one unit in the world coordinate system (resulting in perspective shift or objects appearing relatively larger than they actually are). This is related to what is known as the **intrinsic parameter problem** where intrinsic parameters include focal length, principal point, aspect ratio and the angle between the axis.

Another issue is that a cameras coordinate system will have a different position and rotation in space to that of the world. This is related to the **extrinsic parameter problem**

$$\begin{pmatrix} U \\ V \\ W \end{pmatrix} = \overbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}^{\text{transformation representing intrinsic parameters}} \cdot \underbrace{\begin{pmatrix} X \\ Y \\ Z \\ T \end{pmatrix}}_{\text{transformation representing extrinsic parameters}}$$

1 Single-view geometry

Here we work with a single camera. Imagine there is one point in a world 3D coordinate system (c.s.), P. Our 3D c.s. has some origin O, so we can now represent P in the world with a set of coordinates relative to O. We want now to project P onto the image plane, the image plane is located in the camera 3D c.s., which is separate from the world c.s.. We can have two different kinds of projection:

1. "Extrinsic" projection: the 3D world coordinate system should be projected onto the 3D camera coordinate system
2. "Intrinsic" projection: the 3D camera coordinate system should be projected onto the 2D image plane.

1.1 Intrinsic projection

The steps of an intrinsic projection are as follows:

- Given a point P in the camera 3D coordinate system which is measured in **metres**
- We project P onto the camera image plane, also measured in **metres**
- We then project from our 2D image plane to a discretisesd image which is measured in **pixels**

1.1.1 Homogeneous co ordinates

Euclidean geometry uses the **Cartesian coordinates system**, however, for a projective geometry, **homogeneous coordinates** are more appropriate.

Conversion is simple: add an extra element at the *'end'*:

$$\overbrace{\begin{bmatrix} x \\ y \end{bmatrix}}^{\text{Cartesian form}} \rightarrow \overbrace{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}^{\text{Homogeneous form}}$$

This system has the benefit that if a point is multiplied by a non-zero scalar value w, our point does not change:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \equiv \begin{bmatrix} wx \\ wy \\ w \end{bmatrix}$$

In order to convert from a homogeneous to a Euclidean system: divide by the last coordinate to make it equal to 1 and ignore it.

1.1.2 Concepts

- Principal axis: this is a line from the camera centre perpendicular to the image plane.
- Principal point: this is a point where the principal axis punctures the image plane
- Normalised camera coordinate system: this is a system with its origin at the principal point

1.1.3 Pinhole camera: revisited

With our new coordinate system, we can now revisit the pinhole camera from the previous lecture.

A 3D point in the world coordinate system can be mapped to a 2D projection in the image plane as follows:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \mapsto \begin{bmatrix} f\frac{X}{Z} \\ f\frac{Y}{Z} \\ 1 \end{bmatrix}$$

This can be represented as a vector-matrix multiplication:

$$\overbrace{\begin{pmatrix} fX \\ fY \\ Z \end{pmatrix}}^{\mathbf{x}} = \underbrace{\begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix}}_{\mathbf{P_0}} \overbrace{\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}}^{\mathbf{X}}$$

which can also be written

$$\mathbf{x} = \mathbf{P_0}\mathbf{X}$$

We can re-write the projection matrix $\mathbf{P_0}$ to separate the focal lengths:

$$\mathbf{P_0} = \text{diag}([f, f, 1])[\text{I}|0] = \begin{bmatrix} f & & \\ & f & \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & 0 \\ & 1 & & 0 \\ & & 1 & 0 \end{bmatrix}$$

1.1.4 Image plane to image pixels

- Our normalised camera coordinate system has it's origin at the principal point $p = [p_x, p_y]^{\text{T}}$.
- Our image coordinate system has it's origin in the corner of the image sensor.

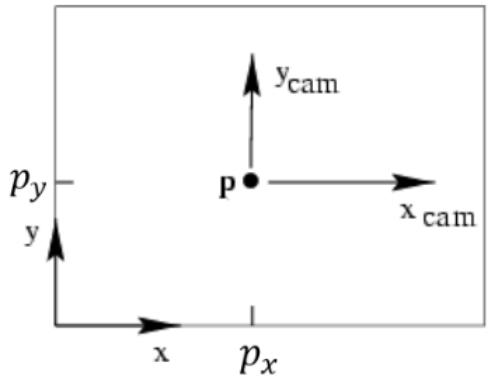

Figure 1: Figure showing camera and image coordinate systems

This can be seen in Figure 1

Moving our camera coordinate system origin to p makes our calculations much easier as we only need consider positive numbers. It allows our transformation seen above to become:

$$(X, Y, Z) \mapsto (f\frac{X}{Z} + p_x, f\frac{Y}{Z} + p_y)$$

Which, in vector-matrix multiplication becomes:

$$\begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

1.1.5 From image plane to image pixels

We now want to project onto our sensor of size Ws $\times$ Hs (in metres). We represent pixels in a rectangular $M_x \times M_y$ matrix.

Let $m_x = \frac{M_x}{W_s}$ and $m_y = \frac{M_y}{H_s}$

We now construct the following projection in vector-matrix multiplication form:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \underbrace{\begin{bmatrix} m_x & 0 & 0 \\ 0 & m_y & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{pixel /m}} \underbrace{\begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{m}} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Which can also be written:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{bmatrix} \alpha_x & 0 & x_0 & 0 \\ 0 & \alpha_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

It is often difficult to guarantee a perfectly rectangular sensor, so we also have a case for a skewed sensor, here we simply add a single value to the projection matrix $\mathbf{P_0}$ to form:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{bmatrix} \alpha_x & s & x_0 & 0 \\ 0 & \alpha_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

We can decompose $\mathbf{P_0}$ into two separate matrices to allow for easier computation and reasoning, we can construct $\mathbf{P_0}$ from the product of a square matrix K and a concatination of the 3×3 identity matrix and a 3D 0-vector:

$$\mathbf{P_0} = \mathbf{K}[\mathbf{I}|0] = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

We refer to $\mathbf{K}$ as our **projection matrix** which prescribes the projection of any 3D point in the camera coordinate system onto our pixels.

Where:

- $\alpha_x = m_x \cdot f$
- $\alpha_y = m_y \cdot f$
- $x_0 = p_x \cdot m_x$
- $y_0 = p_y \cdot m_y$
- s is our skewness factor

1.2 Extrinsic Projection

Here we are concerned with how to project the world coordinate system onto the 3D camera coordinate system.

The 3D camera coordinate system is related to the 3D world coordinate system by a rotation matrix $\mathbf{R}$ and translation $\tilde{\mathbf{t}} = \tilde{\mathbf{C}}$

In Euclidean terms we can write this process of translation followed by rotation as:

$$\tilde{X}_{\text{cam}} = \mathbf{R}(\tilde{X} - \tilde{\mathbf{C}})$$

Our camera is specified by a calibration matrix $\mathbf{K}$, the projection centre in the world coordinate system is given by $\tilde{\mathbf{C}}$ and a rotation matrix $\mathbf{R}$. A 3D point

given in (homogeneous) world coordinates $\mathbf{X}$ is projected onto pixels $\mathbf{x}$ by the following relation:

$$\mathbf{x} = \mathbf{K}[\mathrm{I}|0]\tilde{X}_{\mathrm{cam}} = \mathbf{K}[\mathbf{R}|t]\mathbf{X} = P\mathbf{X}$$

Where:

- $P = \mathbf{K}[\mathbf{R}|t]$
- $t = -\mathbf{R}\tilde{\mathbf{C}}$

Lenses add more complexity through non-linearity, previously straight lines are no longer straight, leading to image distortion.

Lens distortion assumes radially symmetric lenses. We radially expand an image to un-distort it, here only the point changes, not the angle.

Our extrinsic parameters include camera rotation and camera translation.

1.3 Vanishing Points

We can now give a more robust definition of vanishing points,

If we consider a point on one of two parallel lines l_1 and l_2, $\mathbf{A} = \begin{bmatrix} X_A \\ Y_A \\ Z_A \end{bmatrix}$ and a vector $\mathbf{D}$ from $\mathbf{A}$, $\mathbf{D} = \begin{bmatrix} X_D \\ Y_D \\ Z_D \end{bmatrix}$.

A point on a the line $\mathbf{X}(\lambda) = \mathbf{A} + \lambda\mathbf{D}$ is projected to a point $x(\lambda)$ in the image plane by:

$$x(\lambda) = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f\frac{X}{Z} \\ f\frac{Y}{Z} \end{bmatrix} = \begin{bmatrix} \frac{f(X_A+\lambda X_D)}{(Z_A+\lambda Z_D)} \\ \frac{f(Y_A+\lambda Y_D)}{(Z_A+\lambda Z_D)} \end{bmatrix}$$

Here we can see that as $\mathbf{X}(\lambda) \to \infty$, $x(\lambda)$ tends towards the vanishing point, $\mathbf{v}$.

$$\mathbf{v} = \lim_{\lambda\to\infty} x(\lambda) = \lim_{\lambda\to\infty} \begin{bmatrix} f\frac{X_A+\lambda X_D}{Z_A+\lambda Z_D} \\ f\frac{Y_A+\lambda Z_D}{Z_A+\lambda Z_D} \end{bmatrix}$$

Giving us a vanishing point of:

$$\mathbf{v} = \begin{bmatrix} f\frac{X_D}{Z_D} \\ f\frac{Y_D}{Z_D} \end{bmatrix}$$

Our vanishing point depends on the direction $\mathbf{D}$ and not on the point $\mathbf{A}$, meaning that a different set of parallel lines even if they share a point, have a different vanishing point.

1.4 Homography

Homography is the process of projecting points or images from image plane to image plane. It has many uses from Image stitching to augmented reality.

In order to estimate the homography we must find all corresponding points between the two image planes.

Assuming we know all corresponding points between $\mathbf{x}$ and $\mathbf{x}'$ we can represent the translation as:

$$w\mathbf{x}' = \mathbf{H}\mathbf{x}$$

Or,

$$w \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Where the elements of $\mathbf{H}$ can be estimated by applying a direct linear transform or **DLT**.

For each corresponding point i we have:

$$w\mathbf{x}'_\mathbf{i} = \mathbf{H}\mathbf{x}_\mathbf{i}$$

$$w \begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} h_1^T \\ h_2^T \\ h_3^T \end{bmatrix} X_i = \begin{bmatrix} h_1^T X_i \\ h_2^T X_i \\ h_3^T X_i \end{bmatrix}$$

As these are equal, by the definition of cross-product:

$$\mathbf{x}'_\mathbf{i} \times \mathbf{H}\mathbf{x}_\mathbf{i} = 0$$

We can rearrange this vector product into a a vector-matrix product:

$$\mathbf{x}'_\mathbf{i} \times \begin{bmatrix} h_1^T x_i \\ h_2^T x_i \\ h_3^T x_i \end{bmatrix} = \begin{bmatrix} \mathbf{x}'_{i\times} \end{bmatrix} \begin{bmatrix} x_i^T h_1 \\ x_i^T h_2 \\ x_i^T h_3 \end{bmatrix} = \begin{bmatrix} 0 & -1 & y'_i \\ 1 & 0 & x'_i \\ -y'_i & x'_i & 0 \end{bmatrix} \begin{bmatrix} x_i^T h_1 \\ x_i^T h_2 \\ x_i^T h_3 \end{bmatrix}$$

We can then multiply these matrix terms and expose the homography terms h_1, h_2, h_3 into a single vector:

$$\mathbf{x}'_\mathbf{i} \times \mathbf{H}\mathbf{x}_\mathbf{i} = \begin{bmatrix} 0^T & -\mathbf{x}_\mathbf{i}^\mathbf{T} & y'_i \mathbf{x}_i^T \\ \mathbf{x}_i^T & 0^T & -\mathbf{x}'_i \mathbf{x}_i^T \\ -y'_i \mathbf{x} & & \end{bmatrix}$$

I fucking give up, fuck this