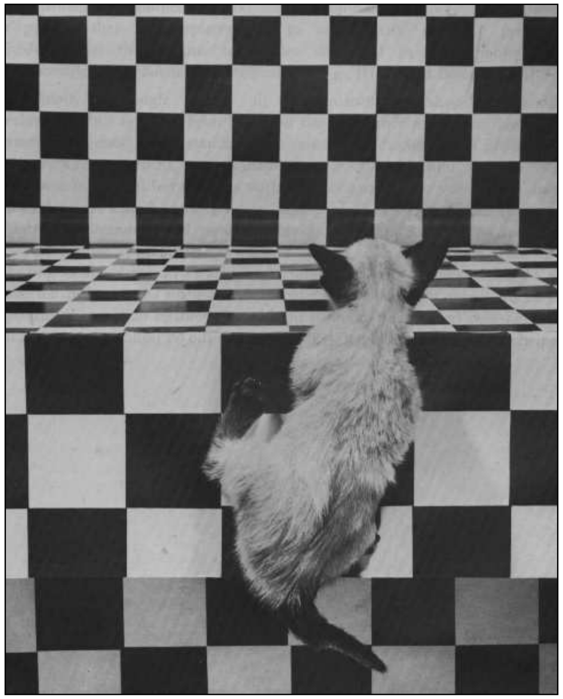
University of Cambridge
Engineering Part IIB & EIST Part II

Module I12: Computer Vision and Robotics

Handout 3: Projection

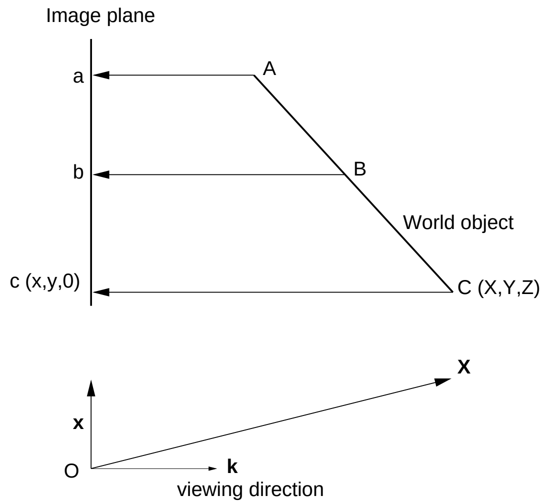


Roberto Cipolla and Andrew Gee
October 1999

Orthographic projection

Recall that computer vision is about discovering from images what is present in the scene and where it is. If we are going to successfully invert the imaging process, we need to understand the imaging process itself.

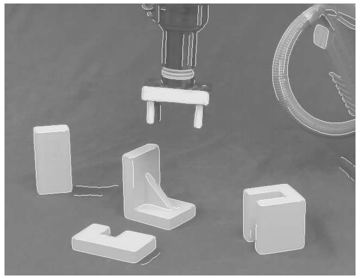
In mechanical drawing, we have already seen how to construct images of 3D scenes using **orthographic projection**: we project the scene onto an **image plane** using *parallel* rays.



$$\mathbf{x} = \mathbf{X} - (\mathbf{X}.\mathbf{k})\mathbf{k} = (\mathbf{k} \times \mathbf{X}) \times \mathbf{k}$$

Orthographic projection

Some of the images which we take with CCD cameras do, indeed, look as if they have been formed by orthographic projection. The image on the left resembles an orthographic projection. Parallel lines in the scene appear as parallel lines in the image, and length ratios along parallel lines are preserved.



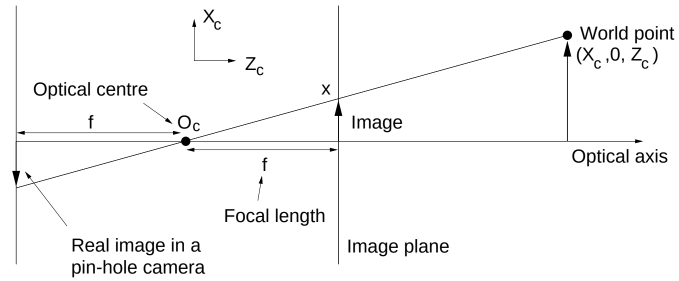
Orthographic?

Certainly not orthographic

However, some CCD images are not explained by orthographic projection. In the image on the right, parallel lines in the scene appear to converge in the image. We clearly need a more general model of projection to explain what is happening in CCD cameras.

Perspective projection

The projection model we adopt is inspired by the **pin-hole camera**. The figure below illustrates the operation of the pin-hole camera in two dimensions ($Y_c = 0$).



This type of projection is called **planar perspective projection**. By analysing the similar triangles, we find that

$$\frac{x}{f} = \frac{X_c}{Z_c} \Leftrightarrow x = \frac{fX_c}{Z_c}$$

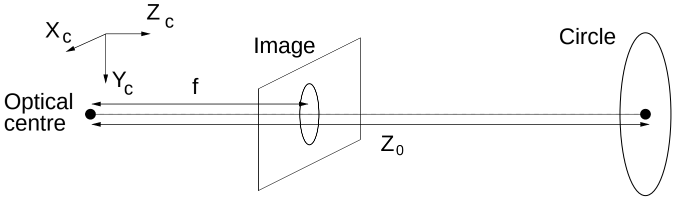
For the three-dimensional case, we also have

$$y = \frac{fY_c}{Z_c}$$

The notation we adopt is $\mathbf{X}_c = (X_c, Y_c, Z_c)$ for world points, and $\mathbf{x} = (x, y)$ for image plane points, both measured in the camera-centered coordinate system (Z_c along the optical axis).

Projection examples

(a) Circle in space, radius a, orthogonal to the optical axis and centered on the optical axis.



$$\mathbf{X}_c = (a\cos\theta, a\sin\theta, Z_0)$$
$$\mathbf{x} = \left(\frac{fa\cos\theta}{Z_0}, \frac{fa\sin\theta}{Z_0}\right)$$

So the image is a circle of radius fa/Z_0. The scaling is inversely proportional to the distance of the circle from the optical centre.

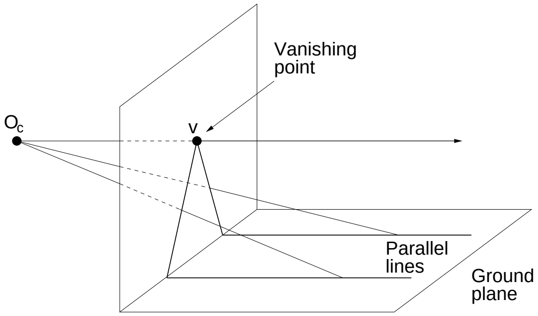
(b) Move the circle in the X_c direction.

$$\mathbf{X}_c = (a\cos\theta + X_0, a\sin\theta, Z_0)$$
$$\mathbf{x} = \left(\frac{fa\cos\theta + fX_0}{Z_0}, \frac{fa\sin\theta}{Z_0}\right)$$

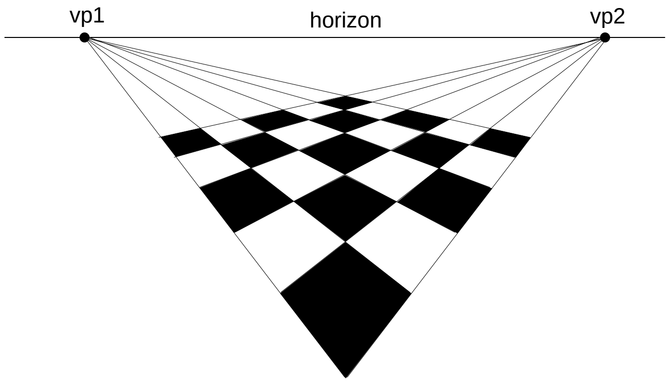
So the image is still a circle of radius fa/Z_0, though the centre of the circle has moved in the image plane.

Vanishing points

As we shall shortly see, a circle does not *always* project to a circle. An important property of perspective projection is the existence of **vanishing points**. These are points in the image where parallel lines appear to meet.

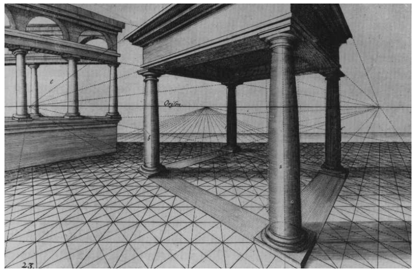


Each set of parallel lines in the world will have a *different* vanishing point in the image.

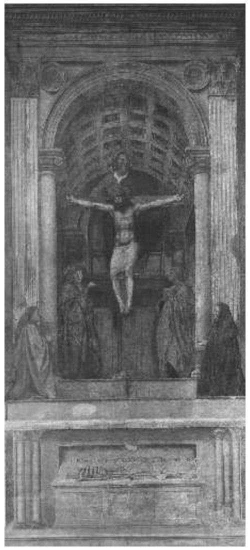


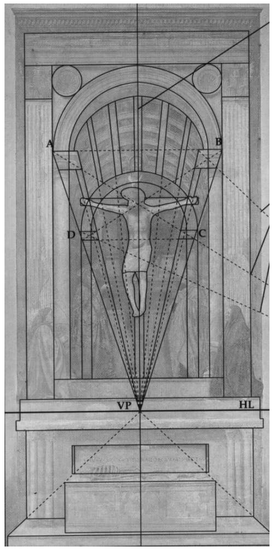
Vanishing points

Similarly, parallel planes in the world meet in a line in the image, often called a **horizon line**. Any set of parallel lines lying on these planes will have a vanishing point on the horizon line.



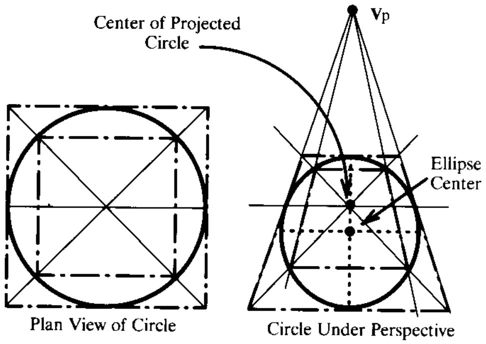
Renaissance painters used perspective constructions to introduce a new realism into art.



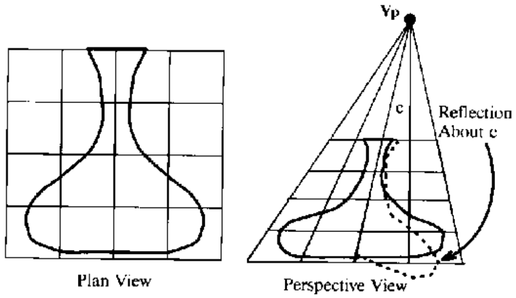


Properties of perspective projection

Armed with the concept of vanishing points, we can now construct the projection of a circle which is not parallel to the image plane.



The circle example reveals that ratios of lengths and areas are *not* preserved under perspective projection. Neither is symmetry.



Vanishing points

Example. Derive the image location $\mathbf{x}_{\mathrm{vp}}$ of the vanishing point for a line in the world.

$$\begin{aligned} \mathbf{X}_c &= \mathbf{a} + \lambda \mathbf{b} \\ \Rightarrow \mathbf{x} &= f\left(\frac{a_x + \lambda b_x}{a_z + \lambda b_z}, \frac{a_y + \lambda b_y}{a_z + \lambda b_z}\right) \end{aligned}$$

As $\lambda \to \infty$, we move further down the line, and $\mathbf{x}$ converges to the vanishing point:

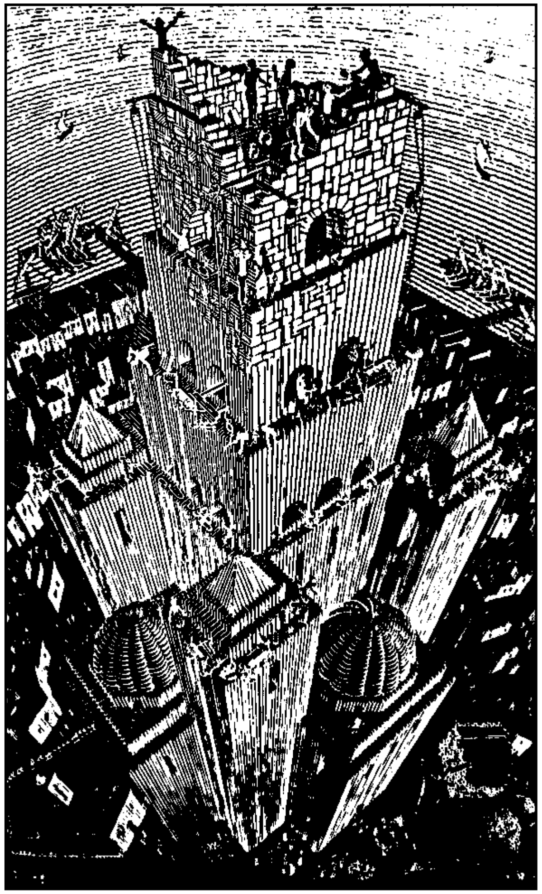
$$\mathbf{x}_{\mathrm{vp}} = f\left(\frac{b_x}{b_z}, \frac{b_y}{b_z}\right)$$

As expected, the vanishing point depends only on the line's orientation and not its position. When $b_z = 0$, the line is parallel to the image plane and the vanishing point is at infinity.

Note that the axes we have defined are relative to the camera, so when $b_z = 0$, the line has no component along the *camera's z*-axis (the optical axis). With a horizontal camera, the image plane is vertical and so vertical lines in the world have a vanishing point at infinity. If the camera is not horizontal, vertical lines in the world *will* have a vanishing point in the image.

Vanishing points

Here's an example of an image with converging vertical lines.

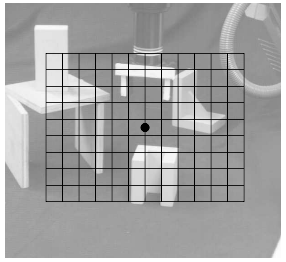


The Tower of Babel, by Maurits Escher

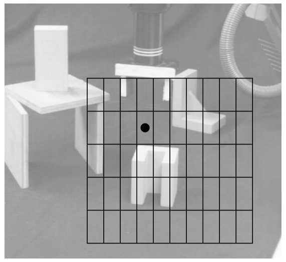
Full camera model

A full camera model describes the mapping from world to pixel coordinates. It must account for the following transformations:

- The **rigid body motion (an isometry)** between the camera and the scene;
- **Perspective projection** onto the image plane;
- **CCD imaging** — the geometry of the CCD array (the size and shape of the pixels) and its position with respect to the optical axis.



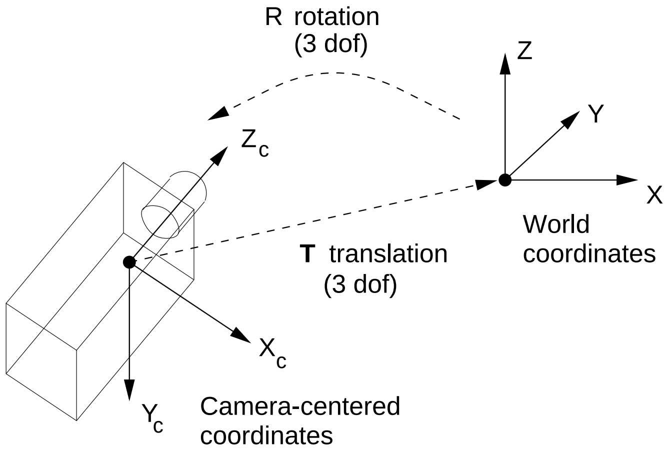
Array centered on optical axis
Square pixels



Array not centered on optical axis
Rectangular pixels

Full camera model

To model the rigid body motion, we attach a coordinate system $\mathbf{X} = (X, Y, Z)$ to the world, and another coordinate system $\mathbf{X}_\mathrm{c} = (X_c, Y_c, Z_c)$ to the camera.



The rigid body motion can be described by a rotation matrix R and a translation vector $\mathbf{T}$:

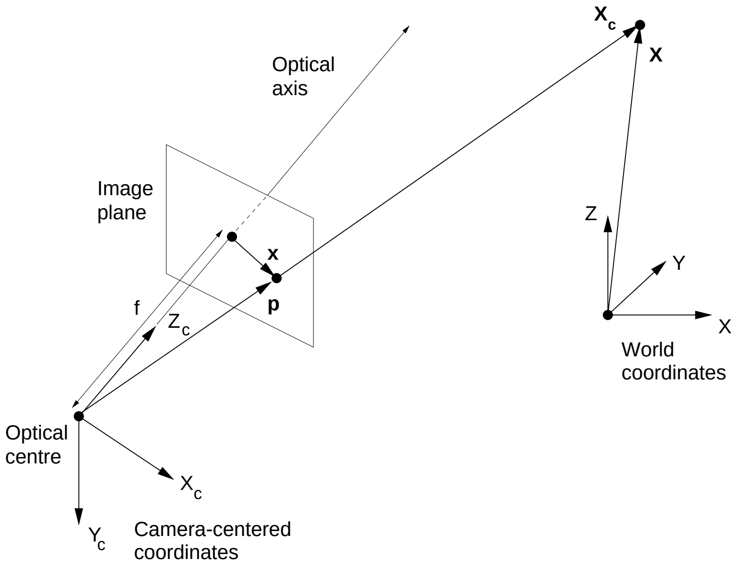
$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}$$

$$\mathbf{X}_\mathrm{c} = \mathrm{R}\mathbf{X} + \mathbf{T}$$

Full camera model

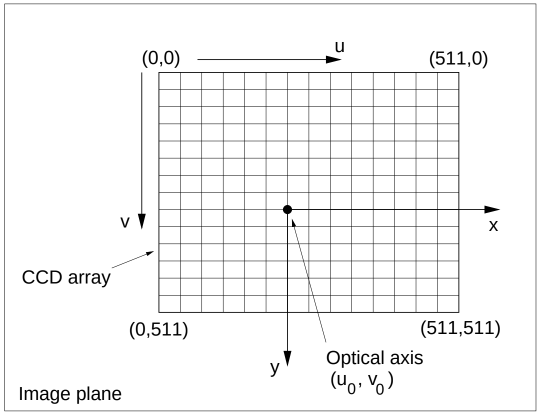
As introduced before, planar perspective projection onto the imaging surface is modelled by:

$$x = \frac{fX_c}{Z_c}$$
$$y = \frac{fY_c}{Z_c}$$



Full camera model

To model CCD imaging, we define pixel coordinates $\mathbf{w} = (u, v)$ in addition to the image plane coordinates $\mathbf{x} = (x, y)$.



$\mathbf{w}$ and $\mathbf{x}$ are related as follows:

$$u = u_0 + k_u x \ , \quad v = v_0 + k_v y$$

The overall mapping from world coordinates $\mathbf{X}$ to pixel coordinates $\mathbf{w} = (u, v)$ is

$$u = u_0 + \frac{k_u f X_c}{Z_c} = u_0 + \frac{k_u f(r_{11}X + r_{12}Y + r_{13}Z + T_x)}{r_{31}X + r_{32}Y + r_{33}Z + T_z}$$
$$v = v_0 + \frac{k_v f Y_c}{Z_c} = v_0 + \frac{k_v f(r_{21}X + r_{22}Y + r_{23}Z + T_y)}{r_{31}X + r_{32}Y + r_{33}Z + T_z}$$

Homogeneous coordinates

The expressions at the foot of page 13 are messy! **Homogeneous coordinates** offer a more natural framework for the study of projective geometry. The imaging process can be expressed as a *linear* matrix operation in homogeneous coordinates. Furthermore, a series of projections can be expressed as a single matrix operation.

We usually express the location of a point in Cartesian coordinates. In 2D space, for example, we would use coordinates $\mathbf{x} = (x, y)$. Cartesian coordinates become cumbersome when dealing with points at infinity, a crucial ingredient in the projection process. The Cartesian coordinates of a point at infinity are in general both infinite but have a definite ratio x/y, depending on the direction of the point from the origin. Calculation with infinite quantities of this kind is confusing, and it is convenient to represent each point not by two numbers $\mathbf{x} = (x, y)$ but by three numbers $\tilde{\mathbf{x}} = (x_1, x_2, x_3)$ such that

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_1/x_3 \\ x_2/x_3 \end{bmatrix}$$

Homogeneous coordinates

If λ is any non-zero number, then $(\lambda x_1, \lambda x_2, \lambda x_3)$ denotes the same point as (x_1, x_2, x_3): it is only the *ratios* of the elements of $\tilde{\mathbf{x}}$ that matter. If now $x_3 = 0$, then $x = x_1/x_3$ and $y = x_2/x_3$ are infinite but have the definite ratio $x/y = x_1/x_2$; the numbers $(x_1, x_2, 0)$ denote points at infinity, obviating calculation with infinite coordinates.

Such a method of representing a point is called a *homogeneous* coordinate system, because any equation in (x, y) is equivalent to a *homogeneous* equation (ie. one in which all the terms are of the same degree) in (x_1, x_2, x_3). For instance, any line has an equation of the form

$$a_1 x + a_2 y + a_3 = 0$$

On substituting x_1/x_3 and x_2/x_3 for x and y, this becomes

$$\begin{aligned} a_1 \frac{x_1}{x_3} + a_2 \frac{x_2}{x_3} + a_3 &= 0 \\ \Leftrightarrow a_1 x_1 + a_2 x_2 + a_3 x_3 &= 0 \end{aligned}$$

The line at infinity, incidentally, also has an equation of this form, namely $x_3 = 0$.

Homogeneous coordinates

Here are some further examples of homogeneous representations, this time using points in 3D space. To convert from homogeneous to Cartesian coordinates, we take ratios:

$$\underset{\tilde{\mathbf{X}}}{(x_1, x_2, x_3, x_4)} \;\rightarrow\; \underset{\mathbf{X}}{\left(\frac{x_1}{x_4}, \frac{x_2}{x_4}, \frac{x_3}{x_4}\right)}$$

If x_4 is zero, then $\tilde{\mathbf{X}}$ represents a point at infinity. $\tilde{\mathbf{X}} = \mathbf{0}$ has no meaning and is undefined.

To convert from Cartesian to homogeneous coordinates, we add an extra dimension and introduce an arbitrary scaling:

$$\underset{\mathbf{X}}{(X, Y, Z)} \;\rightarrow\; \underset{\tilde{\mathbf{X}}}{(\lambda X, \lambda Y, \lambda Z, \lambda)}$$

By convention, λ is set to 1 (where possible).

To appreciate the power of homogeneous coordinates, we need to study some examples, starting with the perspective projection of the point $\mathbf{X}_c = (X_c, Y_c, Z_c)$ onto the image plane $\mathbf{x} = (x, y)$. In homogeneous coordinates we have $\tilde{\mathbf{X}}_c = (\lambda X_c, \lambda Y_c, \lambda Z_c, \lambda)$ and $\tilde{\mathbf{x}} = (sx, sy, s)$.

Perspective projection revisited

Perspective projection can be expressed as

$$\begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \lambda X_c \\ \lambda Y_c \\ \lambda Z_c \\ \lambda \end{bmatrix}$$

or, equivalently,

$$\tilde{\mathbf{x}} = \mathrm{P}_p \tilde{\mathbf{X}}_c \;, \text{ where } \mathrm{P}_p = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$\tilde{\mathbf{x}}$ is the homogeneous representation of the image point $\mathbf{x}$. Notice how perspective projection is a simple matrix multiplication by P_p in homogeneous coordinates.

To check that the homogeneous representation of perspective projection works, we can convert $\tilde{\mathbf{x}}$ into its Cartesian equivalent $\mathbf{x}$:

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} sx/s \\ sy/s \end{bmatrix} = \begin{bmatrix} fX_c/Z_c \\ fY_c/Z_c \end{bmatrix}$$

Notice how the value of λ has no effect on the projection (we would conventionally set λ to 1). Equivalently, the same projection is achieved by multiplying by $\mu \mathrm{P}_p$ ($\mu \neq 0$).

Exercise — horizon lines

As an exercise in the use of homogeneous coordinates, let's consider the two parallel planes

$$\begin{aligned} n_x X_c + n_y Y_c + n_z Z_c &= d_1 \\ n_x X_c + n_y Y_c + n_z Z_c &= d_2\,, \quad d_2 \neq d_1 \end{aligned}$$

and find the equation of their horizon line in the image. Converting to homogeneous coordinates, points $\mathbf{X}_c = (X_c, Y_c, Z_c)$ become $\tilde{\mathbf{X}}_c = (X_1, X_2, X_3, X_4)$, where

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} X_1/X_4 \\ X_2/X_4 \\ X_3/X_4 \end{bmatrix}$$

The homogeneous equations of the planes are

$$\begin{aligned} n_x \frac{X_1}{X_4} + n_y \frac{X_2}{X_4} + n_z \frac{X_3}{X_4} &= d_1 \\ \Leftrightarrow n_x X_1 + n_y X_2 + n_z X_3 &= d_1 X_4 \\ \text{and } \; n_x X_1 + n_y X_2 + n_z X_3 &= d_2 X_4 \end{aligned}$$

Notice that the planes intersect along a line at infinity, which has a well-defined equation in homogeneous coordinates:

$$n_x X_1 + n_y X_2 + n_z X_3 = X_4 = 0 \qquad (1)$$

Exercise — horizon lines

The image of a point $\tilde{\mathbf{X}}_c$ is given by

$$\tilde{\mathbf{x}} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \mathrm{P}_p \tilde{\mathbf{X}}_c = \begin{bmatrix} fX_1 \\ fX_2 \\ X_3 \end{bmatrix} \qquad (2)$$

Combining (1) and (2) we obtain

$$\begin{aligned} \frac{n_x x_1}{f} + \frac{n_y x_2}{f} + n_z x_3 &= 0 \\ \Leftrightarrow n_x x_1 + n_y x_2 + f n_z x_3 &= 0 \qquad (3) \end{aligned}$$

This is the homogeneous equation of the horizon line in the image.

To convert back to Cartesian image coordinates, we take ratios:

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_1/x_3 \\ x_2/x_3 \end{bmatrix} \qquad (4)$$

Combining (3) and (4) we obtain

$$\begin{aligned} n_x\, x\, x_3 + n_y\, y\, x_3 + f\, n_z\, x_3 &= 0 \\ \Leftrightarrow n_x x + n_y y + f n_z &= 0 \end{aligned}$$

This is the Cartesian equation of the horizon line in the image. The horizon of the ground plane can be found by setting $n_x = 0$, $n_y = 1$, $n_z = 0$, which gives $y = 0$, as expected.

Camera projection matrix

Let's look again at the full camera model, this time in homogeneous coordinates. We can construct a camera projection matrix in three stages.

1. Rigid body transformation

There is a rigid body transformation between the world coordinates $\tilde{\mathbf{X}}$ and the camera-centered coordinates $\tilde{\mathbf{X}}_c$. This accounts for rigid body motion between the camera and the scene:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

or, equivalently,

$$\tilde{\mathbf{X}}_c = \mathrm{P}_r \tilde{\mathbf{X}} \ , \ \text{where } \ \mathrm{P}_r = \left[\begin{array}{c|c} \mathrm{R} & \mathbf{T} \\ \hline 0 \ 0 \ 0 & 1 \end{array} \right]$$

$\tilde{\mathbf{X}}$ is the homogeneous representation of the world point $\mathbf{X}$, and likewise for $\tilde{\mathbf{X}}_c$. P_r is the rigid body transformation matrix (rotation and translation).

Camera projection matrix

2. Perspective projection

The next stage is perspective projection of $\tilde{\mathbf{X}}_c$ onto $\tilde{\mathbf{x}}$ in the image plane. We have already seen this on page 17:

$$\begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}$$

or, equivalently,

$$\tilde{\mathbf{x}} = \mathrm{P}_p \tilde{\mathbf{X}}_c \ , \ \text{where } \ \mathrm{P}_p = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$\tilde{\mathbf{x}} = (sx, sy, s)$ is the homogeneous representation of the image point $\mathbf{x} = (x, y)$. P_p is the perspective projection matrix.

3. CCD imaging

Finally, we have to convert to pixel coordinates $\mathbf{w} = (u, v)$:

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} sx \\ sy \\ s \end{bmatrix}$$

Camera projection matrix

Equivalently,

$$\tilde{\mathbf{w}} = \mathrm{P}_c\,\tilde{\mathbf{x}}\ , \ \text{where}\ \ \mathrm{P}_c = \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

$\tilde{\mathbf{w}} = (su, sv, s)$ is the homogeneous representation of the pixel coordinates $\mathbf{w} = (u, v)$. P_c is the CCD calibration matrix.

We can now express the overall imaging process, from $\tilde{\mathbf{X}}$ to $\tilde{\mathbf{w}}$, as a single matrix multiplication in homogeneous coordinates:

$$\begin{aligned} \tilde{\mathbf{w}} &= \mathrm{P}_{ps}\tilde{\mathbf{X}} \\ \text{where } \mathrm{P}_{ps} &= \mathrm{P}_c\mathrm{P}_p\mathrm{P}_r \\ &= \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \left[\begin{array}{c|c} \mathrm{R} & \mathbf{T} \\ \hline 0\ 0\ 0 & 1 \end{array}\right] \end{aligned}$$

P_{ps} is the camera projection matrix for a perspective camera. It is a 3×4 matrix with 10 degrees of freedom[1]. The product $\mathrm{P}_c\mathrm{P}_p$ accounts for all the *intrinsic* (or internal) camera parameters. P_r accounts for the *extrinsic* parameters.

[1] At first sight, it appears to have 11 degrees of freedom: 3 for R, 3 for $\mathbf{T}$, and one each for f, k_u, k_v, u_0 and v_0. However, these parameters

The projection matrix

The projection matrix, P_{ps} is *not* a general 3×4 matrix, but has a special structure composed of P_r, P_p and P_c. It can be conveniently decomposed into the following two matrices – a 3×3 upper triangular matrix called the **camera calibration matrix K** and a matrix representing the rigid-body motion:

$$\begin{aligned} \mathrm{P}_{ps} &= \mathbf{K}[\mathrm{R}|\mathbf{T}] \\ &= \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \end{bmatrix} \end{aligned}$$

where the image scaling factors are $\alpha_u = fk_u$ and $\alpha_v = fk_v$. The ratio α_v/α_u is known as the aspect ratio.

are not all independent in their effect on the projection. If you refer back to the equations at the foot of page 13, you should be able to see that f, k_u, k_v provide only 2 degrees of freedom between them.

The projective camera

We could also consider another camera model, the **projective camera**, which is described by the *general* 3×4 matrix P:

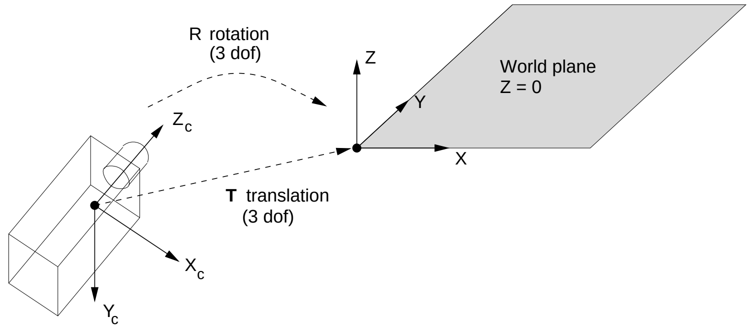
$$\tilde{\mathbf{w}} = \mathrm{P}\tilde{\mathbf{X}}, \text{ where } \mathrm{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}$$

The projective camera has 11 degrees of freedom (since the overall scale of P does not matter). It is often far more convenient to deal with a projective camera than a perspective one, since we do not have to worry about any nonlinear constraints on the elements of P.

Since the perspective camera is a special case of the projective camera, any results we derive for the projective camera will also hold for the perspective camera.

Viewing a plane

Camera models can be simplified under restrictive viewing conditions. Suppose, for example, we are viewing a planar scene (a tabletop, for instance). The geometry of the scenario is illustrated below.



Without loss of generality, assume that the plane we are viewing has equation $Z = 0$. The rigid body displacement between the camera and the plane can be expressed in homogeneous coordinates as

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Viewing a plane

However, we know that $Z = 0$, so we can reduce this to

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & T_x \\ r_{21} & r_{22} & T_y \\ r_{31} & r_{32} & T_z \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

or, equivalently,

$$\tilde{\mathbf{X}}_c = \mathrm{P}_r^p \tilde{\mathbf{X}}^p \ , \text{ where } \ \mathrm{P}_r^p = \begin{bmatrix} r_{11} & r_{12} & T_x \\ r_{21} & r_{22} & T_y \\ r_{31} & r_{32} & T_z \\ 0 & 0 & 1 \end{bmatrix}$$

$\tilde{\mathbf{X}}^p$ is the homogeneous representation of a point $\mathbf{X}^p = (X, Y)$ on the world plane. P_r^p is the planar rigid body transformation matrix (rotation and translation).

The rest of the imaging process can be achieved using the same perspective projection (P_p) and CCD imaging (P_c) matrices as before.

Viewing a plane

The overall imaging process is:

$$\begin{aligned} \tilde{\mathbf{w}} &= \mathrm{P}_{ps}^p \tilde{\mathbf{X}}^p \\ \text{where } \mathrm{P}_{ps}^p &= \mathrm{P}_c \mathrm{P}_p \mathrm{P}_r^p \\ &= \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & T_x \\ r_{21} & r_{22} & T_y \\ r_{31} & r_{32} & T_z \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

P_{ps}^p is the camera projection matrix for a perspective camera viewing a plane. It is a 3×3 matrix with a special structure composed of P_r^p, P_p and P_c.

As with the 3D case, we can relax the constraints on the elements of P_{ps}^p to obtain a more tractable camera model described by the general 3×3 matrix:

$$\tilde{\mathbf{w}} = \mathrm{P}^p \tilde{\mathbf{X}}^p, \text{ where } \ \mathrm{P}^p = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}$$

The transformation between $\tilde{\mathbf{w}}$ and $\tilde{\mathbf{X}}^p$ is known as a **planar projective transformation** or a **homography** or collineation. It has 8 degrees of freedom (the scale of P^p does not matter).

Viewing a line

Finally, we can consider the special case of viewing a world line. Without loss of generality, assume we are interested in the line defined by the world X-axis. The overall imaging process is:

$$\tilde{\mathbf{w}} = \mathrm{P}^l_{ps}\tilde{\mathbf{X}}^l$$

$$\text{where } \mathrm{P}^l_{ps} = \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & T_x \\ r_{21} & T_y \\ r_{31} & T_z \\ 0 & 1 \end{bmatrix}$$

P^l_{ps} is the camera projection matrix for a perspective camera viewing a line. It is a 3×2 matrix with a special structure composed of P^l_r, P_p and P_c.

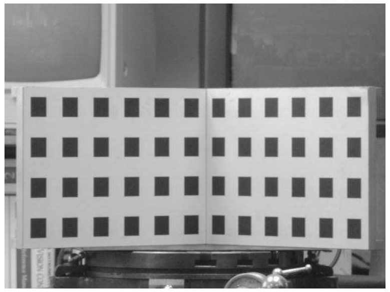
As with the 3D and 2D cases, we can relax the constraints on the elements of P^l_{ps} to obtain a more tractable camera model described by the general 3×2 matrix:

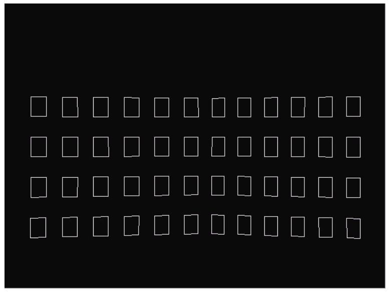
$$\tilde{\mathbf{w}} = \mathrm{P}^l\tilde{\mathbf{X}}^l, \text{ where } \mathrm{P}^l = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \\ p_{31} & p_{32} \end{bmatrix}$$

This projective camera model has 5 degrees of freedom (since the overall scale of P^l does not matter).

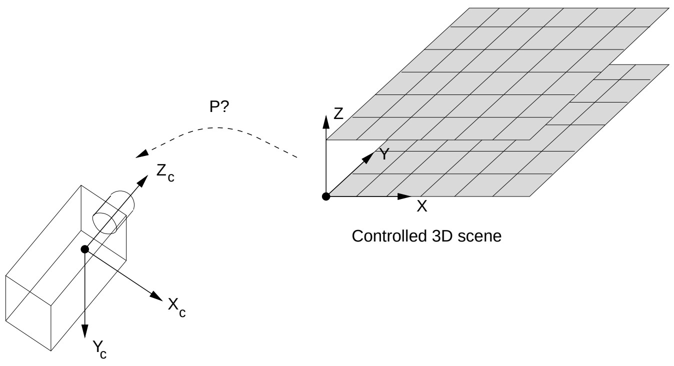
Camera calibration: 3D $\rightarrow$ 2D

Camera calibration is the name given to the process of discovering the projection matrix (and its decomposition into camera matrix and the position and orientation of the camera) from an image of a controlled scene. For example, we might set up the camera to view a calibrated grid of some sort.





Camera Calibration



For a projective camera we have:

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

There are 11 parameters to estimate (since the overall scale of P does not matter, we could, for example, set p_{34} to 1).

Camera Calibration

Each point we observe gives us a pair of equations. Setting p_{34} to 1 we obtain:

$$u_i = \frac{su_i}{s} = \frac{p_{11}X_i + p_{12}Y_i + p_{13}Z_i + p_{14}}{p_{31}X_i + p_{32}Y_i + p_{33}Z_i + 1}$$

$$v_i = \frac{sv_i}{s} = \frac{p_{21}X_i + p_{22}Y_i + p_{23}Z_i + p_{24}}{p_{31}X_i + p_{32}Y_i + p_{33}Z_i + 1}$$

Since we are observing a calibrated scene, we know X_i, Y_i, and Z_i, and we observe the pixel coordinates u_i and v_i in the image. The equations above can be rearrange to give two linear equations in the unknown projection matrix parameters.

Since there are 11 unknowns, we need to observe at least 6 points to calibrate the camera. The equations can be solved using linear least squares. Note how the use of the projective camera has linearized the problem.

Camera Calibration

The linear solution is, however, only approximate and should ideally be used as the starting point for non-linear minimisation: i.e. finding the parameters of the projection matrix that minimise the errors between measured image points, (u_i, v_i) and projected (or modelled) image positions, $((\hat{u}_i, \hat{v}_i)$:

$$\min_{\mathbf{P}} \sum_i ((u_i - \hat{u}_i)^2 + (v_i - \hat{v}_i)^2)$$

Having obtained the projection matrix it is possible to decompose it into the camera calibration matrix and the orientation and position of the camera (if necessary):

$$\mathrm{P}_{ps} = \mathbf{K}[\mathrm{R}|\mathbf{T}]$$

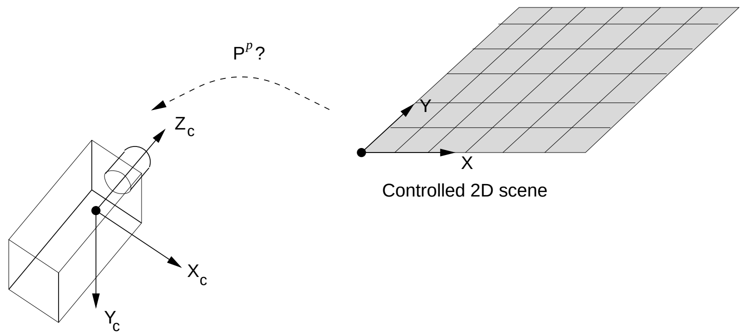
Standard matrix techniques exist for decomposing the 3×3 sub-matrix into the product of an upper triangular matrix, $\mathbf{K}$, and a rotation (orthogonal) matrix $\mathbf{R}$ (known as QR decompositon).

The translation vector or position of the camera can then be obtained by:

$$\mathbf{T} = \mathbf{K}^{-1}(p_{14}, p_{24}, p_{34})^T$$

Camera calibration: 2D → 2D

To calibrate the camera for viewing planar scenes, we could set up the camera to view some sort of calibrated planar grid.



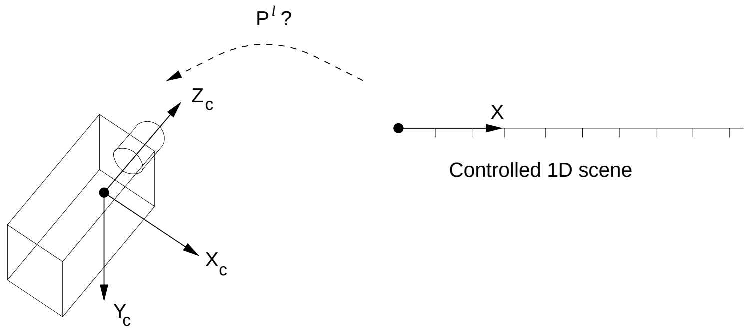
For a plane to plane projectivity, we have

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

There are 8 parameters to estimate (if we set, for example, p_{33} to 1), and each observed point gives us a pair of linear equations, so we need to observe at least 4 points. Again, we use linear least squares to solve for the elements of P^p.

Camera calibration: 1D $\rightarrow$ 1D

Finally, we consider the calibration of a camera viewing a line. This is accomplished by viewing a line with some markings at known positions.



For a projective camera we have

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \\ p_{31} & p_{32} \end{bmatrix} \begin{bmatrix} X \\ 1 \end{bmatrix}$$

There are 5 parameters to estimate (if we set, for example, p_{32} to 1), and each observed point gives us a pair of linear equations, so we need to observe at least 3 points. Again, we use linear least squares to solve for the elements of P^l.

Recovery of world position

With a calibrated camera, we can attempt to recover the world position of image features.

1D case (line to line): given u, we can uniquely determine the position of the point on the line.

$$\begin{bmatrix} su \\ s \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} \\ p_{31} & 1 \end{bmatrix} \begin{bmatrix} X \\ 1 \end{bmatrix}$$

$$\Leftrightarrow u = \frac{su}{s} = \frac{p_{11}X + p_{12}}{p_{31}X + 1}$$

$$\Leftrightarrow X = \frac{p_{12} - u}{p_{31}u - p_{11}}$$

2D case (plane to plane): given u and v, we can uniquely determine the position of the point on the world plane. For a plane to plane projectivity, we have

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & 1 \end{bmatrix} \begin{bmatrix} \lambda X \\ \lambda Y \\ \lambda \end{bmatrix}$$

$$\Leftrightarrow \begin{bmatrix} \lambda X \\ \lambda Y \\ \lambda \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & 1 \end{bmatrix}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

$$\Leftrightarrow X = \frac{p^i_{11}u + p^i_{12}v + p^i_{13}}{p^i_{31}u + p^i_{32}v + p^i_{33}}, \quad Y = \frac{p^i_{21}u + p^i_{22}v + p^i_{23}}{p^i_{31}u + p^i_{32}v + p^i_{33}}$$

Recovery of world position

3D case (3D world to image plane): given u and v, we cannot uniquely determine the position of the point in the world.

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\Rightarrow u = \frac{su}{s} = \frac{p_{11}X + p_{12}Y + p_{13}Z + p_{14}}{p_{31}X + p_{32}Y + p_{33}Z + p_{34}}$$

$$\Rightarrow v = \frac{sv}{s} = \frac{p_{21}X + p_{22}Y + p_{23}Z + p_{24}}{p_{31}X + p_{32}Y + p_{33}Z + p_{34}}$$

Each observed image point (u, v) gives us two equations in three unknowns (X, Y, Z). These equations define a line (ie. a ray) in space, on which the world point must lie.

For general 3D scene interpretation, we need to use more than one view. Later in the course we will take a detailed look at stereo vision and structure from motion.

Case study – Image mosaicing

Any two images of a general scene with the same camera centre are related by a planar projective transformation given by:

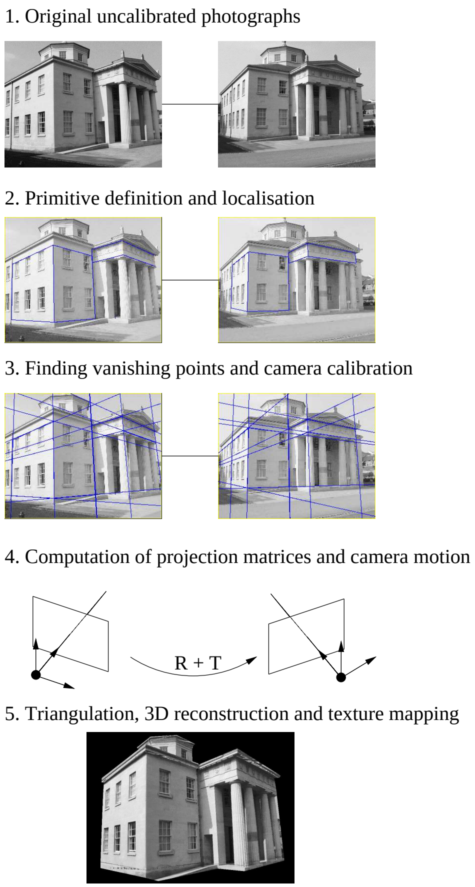
$$\tilde{\mathbf{w}}' = \mathbf{K}\mathrm{R}\mathbf{K}^{-1}\tilde{\mathbf{w}}$$

where $\mathbf{K}$ represents the camera calibration matrix and R is the rotation between the views.

This projective transformation is also known as the homography induced by the plane at infinity. A minimum of four image correspondences can be used to estimate the homography and to warp the images onto a common image plane. This is known as **mosaicing**.

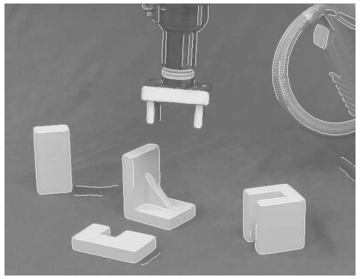
Case study – Photobuilder

Vanishing points corresponding to three orthogonal directions can be used to recover the projection matrix of the viewpoint.



Parallel projection

Recall that we have seen plenty of CCD images which appear to have been formed by orthographic projection. For example:



Orthographic?

Certainly not orthographic

It might be useful to analyse what is special about the image on the left. This will allow us to identify a simpler, more tractable camera model for use under certain viewing conditions.

It appears that parallel projection is a good approximation when the *depth* of the objects in the scene is small compared with the distance of the camera from the scene. On the left all the objects are within a narrow depth band, so ΔZ_c is small compared to Z_c. On the right there is a large depth variation ΔZ_c in the image.

Parallel projection

Recall that perspective projection is

$$\begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}$$

or, equivalently,

$$\tilde{\mathbf{x}} = \mathrm{P}_p \tilde{\mathbf{X}}_\mathrm{c} \ , \ \text{where} \ \ \mathrm{P}_p = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

The only component of $\tilde{\mathbf{x}}$ that depends on Z_c is the scaling term, s. If we consider Z_c as approximately constant for all objects in the scene, so that $Z_c = Z_c^{av}$, we can rewrite the projection as

$$\begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 0 & Z_c^{av} \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}$$

or, equivalently,

$$\tilde{\mathbf{x}} = \mathrm{P}_{pll} \tilde{\mathbf{X}}_\mathrm{c} \ , \ \text{where} \ \ \mathrm{P}_{pll} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 0 & Z_c^{av} \end{bmatrix}$$

This is called **weak perspective** projection.

Weak perspective

We can now derive the form of the overall weak perspective projection matrix from world coordinates $\mathbf{X}$ to pixel coordinates $\mathbf{w}$.

Following an identical derivation to the one we used in the perspective case on page 22, but inserting the parallel projection matrix P_{pll} in place of the perspective matrix P_p, we obtain

$$\begin{aligned} \tilde{\mathbf{w}} &= \mathrm{P}_{wp} \tilde{\mathbf{X}} \\ \text{where } \mathrm{P}_{wp} &= \mathrm{P}_c \mathrm{P}_{pll} \mathrm{P}_r \end{aligned}$$

$$= \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 0 & Z_c^{av} \end{bmatrix} \left[\begin{array}{c|c} \mathrm{R} & \mathbf{T} \\ \hline 0\ 0\ 0 & 1 \end{array} \right]$$

$$= \begin{bmatrix} fk_u r_{11} & fk_u r_{12} & fk_u r_{13} & fk_u T_x + u_0 Z_c^{av} \\ fk_v r_{21} & fk_v r_{22} & fk_v r_{23} & fk_v T_y + v_0 Z_c^{av} \\ 0 & 0 & 0 & Z_c^{av} \end{bmatrix}$$

P_{wp} is the projection matrix for a weak perspective camera. It is a 3×4 matrix with a special structure composed of P_r, P_{pll} and P_c.

The affine camera

As usual, we prefer to discard the nonlinear constraints on the elements and consider the general 3×4 matrix of this form:

$$\mathrm{P}_{aff} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ 0 & 0 & 0 & p_{34} \end{bmatrix}$$

P_{aff} is the projection matrix for the **affine camera**. It has 8 degrees of freedom (since the overall scale of P_{aff} does not matter). If we set p_{34} to 1, we can write the projection as

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

We therefore need only 4 points to calibrate an affine camera (compare with 6 points for the full projective camera). This is one of the principle attractions of using an affine camera where appropriate. Note that the affine camera is *linear*.

Planar weak perspective

We can also consider a weak perspective camera viewing a plane. This would be a good model to use when the plane in the image has little depth variation compared with the viewing distance.

Following an identical derivation to the one we used in the perspective case on page 27, but inserting the parallel projection matrix P_{pll} in place of the perspective matrix P_p, we obtain

$$\tilde{\mathbf{w}} = \mathrm{P}^p_{wp} \tilde{\mathbf{X}}^p$$

$$\text{where } \mathrm{P}^p_{wp} = \mathrm{P}_c \mathrm{P}_{pll} \mathrm{P}^p_r$$

$$= \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 0 & Z_c^{av} \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & T_x \\ r_{21} & r_{22} & T_y \\ r_{31} & r_{32} & T_z \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} fk_u r_{11} & fk_u r_{12} & fk_u T_x + u_0 Z_c^{ac} \\ fk_v r_{21} & fk_v r_{22} & fk_v T_y + v_0 Z_c^{ac} \\ 0 & 0 & Z_c^{av} \end{bmatrix}$$

P^p_{wp} is the projection matrix for a weak perspective camera viewing a plane. It is a 3×4 matrix with a special structure composed of P^p_r, P_{pll} and P_c.

Planar affine imaging

As usual, we prefer to discard the nonlinear constraints on the elements and consider the general 3×3 matrix of this form:

$$\mathrm{P}^p_{aff} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ 0 & 0 & p_{33} \end{bmatrix}$$

P^p_{aff} is the projection matrix for the affine camera viewing a plane. It has 6 degrees of freedom (since the overall scale of P^p_{aff} does not matter). If we set p_{33} to 1, we can write the projection as

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

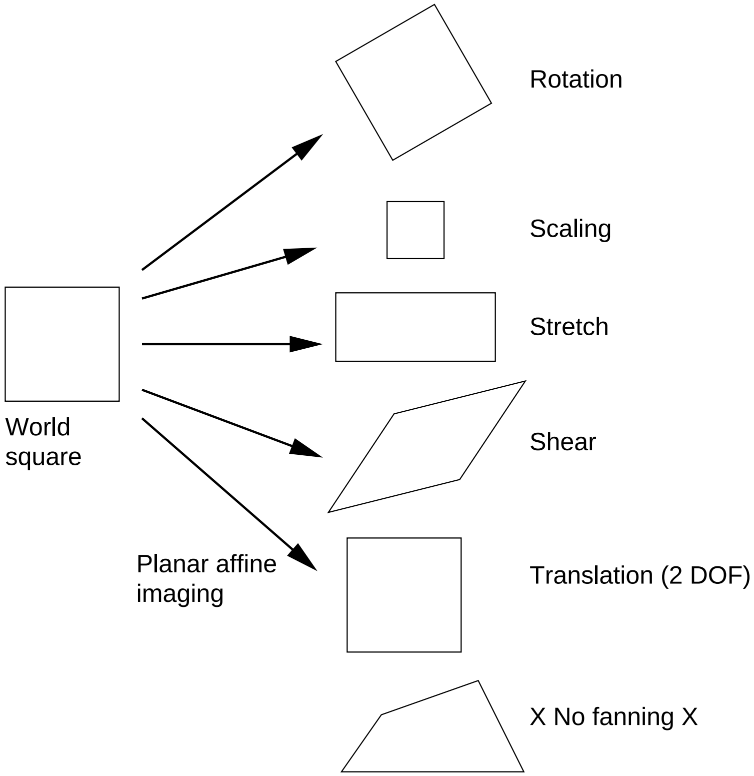
We need 3 points to calibrate this camera.

Finally, 1D affine imaging (viewing collinear features which have little depth variation compared with the viewing distance) can be described as

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \begin{bmatrix} X \\ 1 \end{bmatrix}$$

Two points are required for calibration.

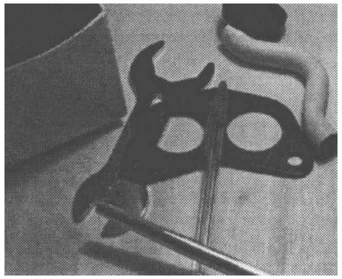
Planar affine imaging



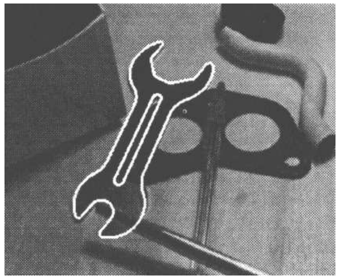
Here are the six degrees of freedom of planar affine imaging. Fanning is not possible: when perspective effects are significant in the image (converging parallel lines), an affine camera is not appropriate and a projectivity (8 degrees of freedom) should be used instead.

Planar object recognition

Planar object recognition is of significant industrial importance. Given an uncalibrated camera's view of a cluttered scene containing many, possibly occluded, planar objects, we wish to identify which objects are present in the scene and where they are. We assume that we hold geometric models of the objects in a **library**.



Cluttered scene



Spanner identified

To appreciate the difficulty of the task, let's consider a naive approach. Suppose we're trying to recognise any instances of the spanner in the left hand image. The desired output of the system is shown on the right. We know the shape of the spanner (it's in our model library), so we could project it into the image and use Canny's algorithm to look for evidence of edges around the projected outline.

Invariants

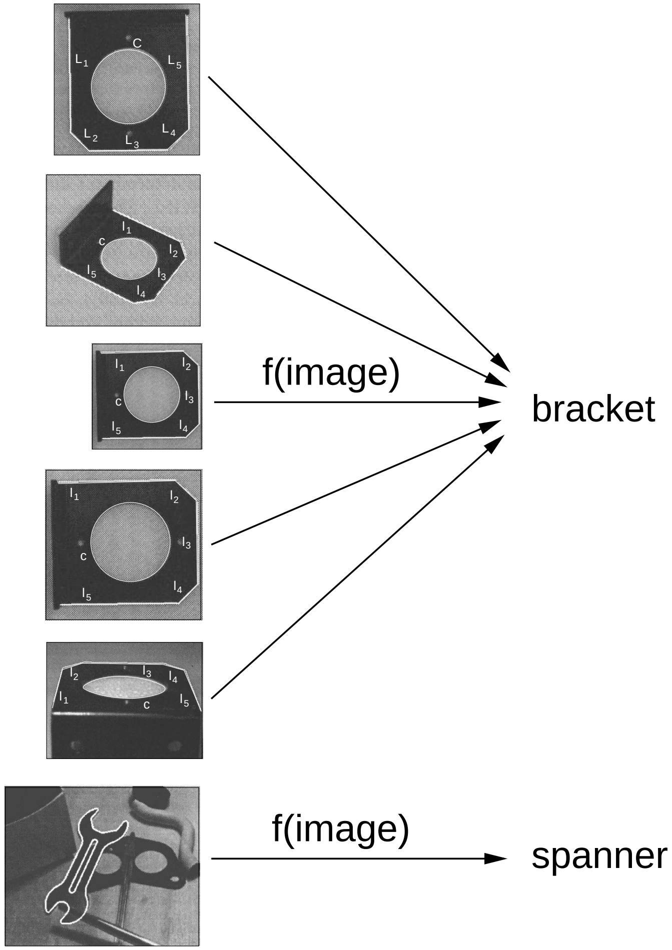
The problem with this simple approach is that we have to consider all possible **poses** (positions and orientations) of the spanner. The search space is infeasibly large! The problem is compounded when we consider that there may be several hundred objects in our model library, and we want to look for all of them.

What we require is some way of taking measurements directly in the image to identify the presence of known objects. This leads us on to the theory of **invariants**.

An invariant is a measurement we make in an image: let's call it f(image). The invariant does not change across different viewpoints of the same object. It should, however, change across different objects. We could use such measurements to directly infer the presence of known objects in the scene.

Invariants depend on the type of camera: for instance, an affine camera will have different invariants to a projective camera.

Invariants

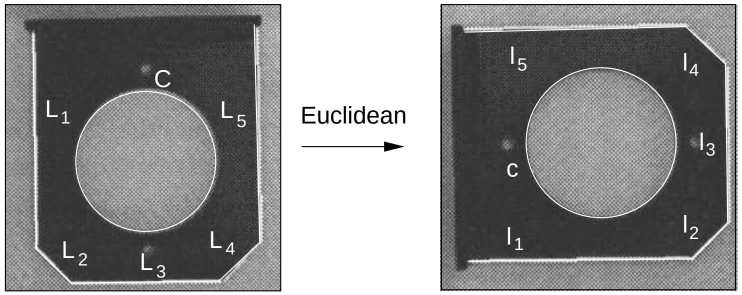


In the next few slides we'll investigate the invariants of several different cameras viewing *planar* scenes.

Euclidean invariants

Let's start by considering the simplest camera model we can imagine, a Euclidean camera. This would be an appropriate model when the image plane is parallel to, and a fixed distance from, the world plane.

Typical images (of a bracket) acquired by this camera might look like this:

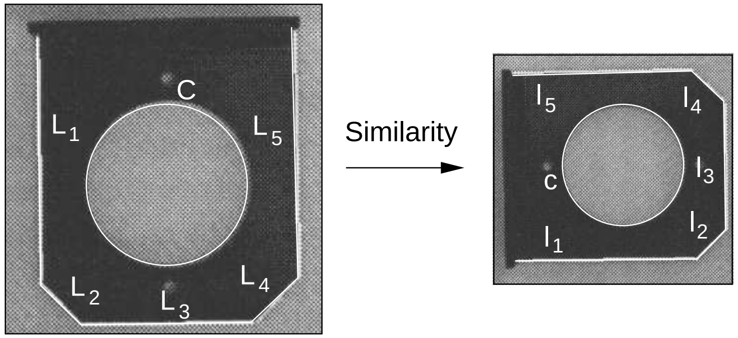


What are the invariants of this transformation? The most obvious ones are length and area. That is, $L_1 = l_1$, $L_2 = l_2$ etc. In the context of object recognition, and assuming a Euclidean camera, we could extract all circles c in an image and calculate their areas: any circle which has the same area as C provides direct evidence for the presence of the bracket in the image.

Similarity invariants

The next simplest camera we can imagine is the similarity camera. The image plane is still parallel to the world plane, but the camera is now allowed to move towards or away from the plane.

Typical images of the bracket acquired by this camera might look like this:

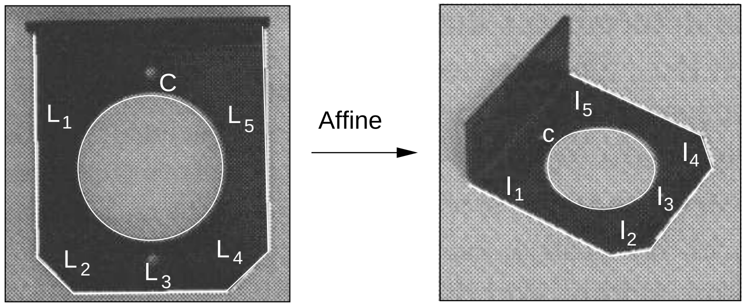


Length and area are no longer invariant. However, ratios of lengths and angles are. That is $L_1/L_2 = l_1/l_2$, etc. Assuming a similarity camera, we could extract all pairs of connected lines $\{l_i, l_j\}$ in an image and measure the ratio of their lengths: any pair for which $l_i/l_j = L_1/L_2$ provides direct evidence for the presence of the bracket in the image.

Affine invariants

The first really useful camera is the affine camera. Recall that this is an appropriate camera model when there is little depth variation in the scene compared with the viewing distance.

Typical images of the bracket acquired by this camera might look like this:

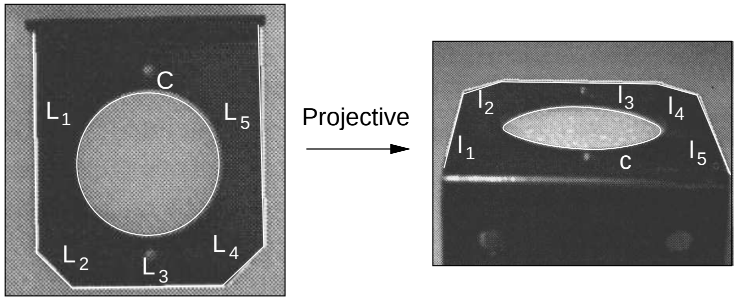


Ratios of lengths and angles are no longer invariant. However, parallelism, ratios of lengths along collinear or parallel lines (eg. midpoints) and ratios of areas are. For example, $L_1/L_5 = l_1/l_5$. Assuming an affine camera, we could extract all sets of five connected lines $\{l_1 \ldots l_5\}$ in an image: any set for which $l_1/l_5 = L_1/L_5$ provides direct evidence for the presence of the bracket in the image.

Projective invariants

If the viewing conditions are completely unrestricted, we will have to use the most general camera model, the projective camera.

Typical images of the bracket acquired by this camera might look like this:

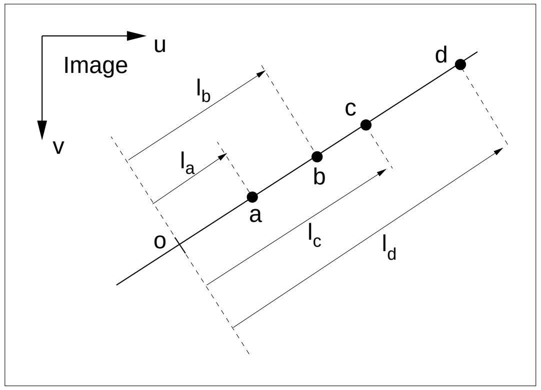


Invariants are getting hard to find! Parallelism, for instance, is no longer invariant. Some obvious invariants include concurrency, collinearity, tangent discontinuities and cusps. There is also **order of contact** between two lines or curves: intersection (1 point of contact), tangency (2 points of contact between a curve and a line) and inflections (three points of contact between a curve and a line).

The cross-ratio

To find a numerical invariant, we start with the simplest projective case, that of viewing a line. Recall that the image u-coordinate of a point X on the line is given by

$$\begin{bmatrix} su \\ s \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} \\ p_{31} & 1 \end{bmatrix} \begin{bmatrix} X \\ 1 \end{bmatrix}$$



The figure shows the image of four world points A, B, C and D, and the world origin O. Distances l measured along the image line from o are linear functions of u and can therefore be expressed as

$$\begin{bmatrix} sl \\ s \end{bmatrix} = \begin{bmatrix} p & q \\ r & 1 \end{bmatrix} \begin{bmatrix} X \\ 1 \end{bmatrix}$$

The cross-ratio

Hence we obtain

$$l_i = \frac{pX_i + q}{rX_i + 1}$$

Let's investigate whether the ratios of lengths along the line are invariant.

$$l_c - l_a = \frac{(X_c - X_a)(p - qr)}{(rX_c + 1)(rX_a + 1)}$$

$$l_c - l_b = \frac{(X_c - X_b)(p - qr)}{(rX_c + 1)(rX_b + 1)}$$

$$\Rightarrow \frac{l_c - l_a}{l_c - l_b} = \frac{(X_c - X_a)(rX_b + 1)}{(X_c - X_b)(rX_a + 1)} \qquad (5)$$

So the ratios of lengths are *not* invariant (compare with the affine case, where they are).

Similarly,

$$\frac{l_d - l_a}{l_d - l_b} = \frac{(X_d - X_a)(rX_b + 1)}{(X_d - X_b)(rX_a + 1)} \qquad (6)$$

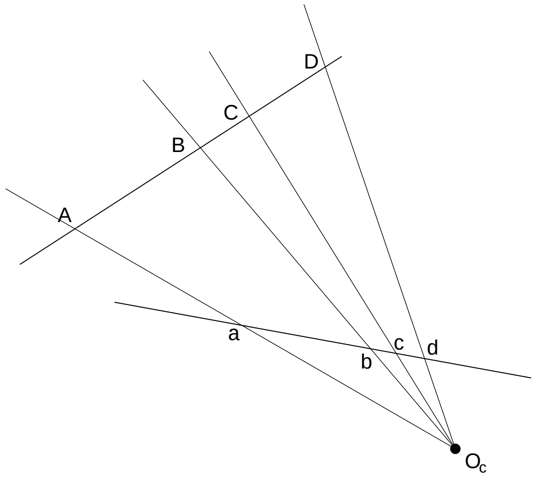
Dividing (6) by (5) we obtain

$$\frac{(l_d - l_a)(l_c - l_b)}{(l_d - l_b)(l_c - l_a)} = \frac{(X_d - X_a)(X_c - X_b)}{(X_d - X_b)(X_c - X_a)}$$

This is the **cross-ratio**, which *is* invariant.

The cross-ratio: example

Let's check the cross-ratio by constructing a line to line perspective projection and measuring lengths.



Lengths measured with a ruler are:

AD = 77.5mm, BC = 15.0mm, BD = 38.5mm, AC = 54.0mm

ad = 48.5mm, bc = 7.0mm, bd = 14.5mm, ac = 41.0mm

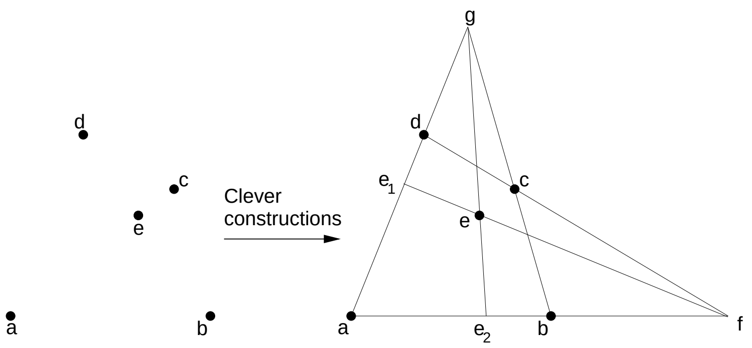
Forming the cross-ratios gives:

$$\frac{AD \times BC}{BD \times AC} = 0.56 \ , \quad \frac{ad \times bc}{bd \times ac} = 0.57$$

So the cross-ratios agree to within experimental accuracy.

Five point invariants on the plane

Even though we have developed the cross-ratio for four points *on a line*, we can also use it in planar imaging situations. We need 5 distinguished points to form invariants on the plane.



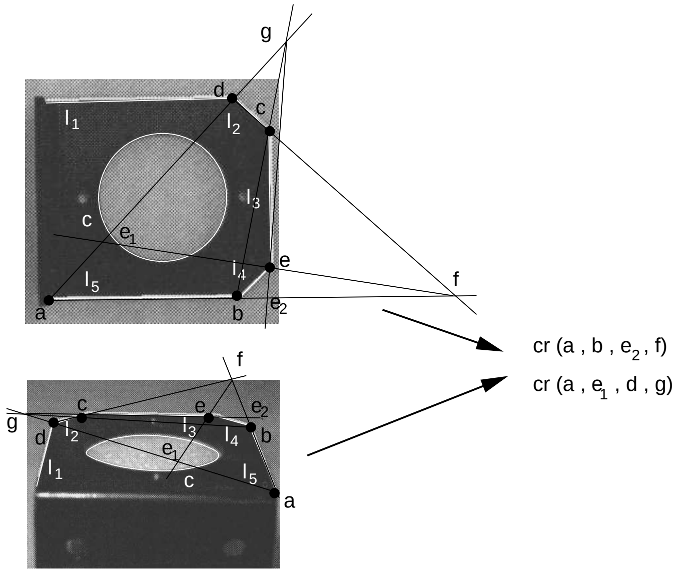
Given the image of the 5 points a . . . e, we can use the invariant property of intersection to find 4 more distinguished points: f, the intersection of the extrapolated lines a-b and d-c; g, similarly; e_1, the intersection of the line joining f and e with the side a-d, and e_2 similarly. We can now form two cross-ratios:

$$\begin{aligned} \delta_1 &= \text{cross-ratio of } \{a, e_2, b, f\} \\ \delta_2 &= \text{cross-ratio of } \{a, e_1, d, g\} \end{aligned}$$

These will be the same measured in any view of the 5 points.

Five point invariants on the plane

Here's an example of how we could use the five point invariants for object recognition.



We identify five distinguished points a . . . e at the corners of the bracket and construct intersections to find four more distinguished points f, g, e_1 and e_2. We now have two sets of four collinear points, $\{a, b, e_2, f\}$ and $\{a, e_1, d, g\}$, for which we can calculate cross-ratios. These will be the same in any view, and can be used to identify the bracket. Other configurations of five planar points will yield different cross-ratios.

Canonical views

Another way to form projective invariants for 5 coplanar points uses a calibration-like procedure. For a plane to plane projectivity, we have

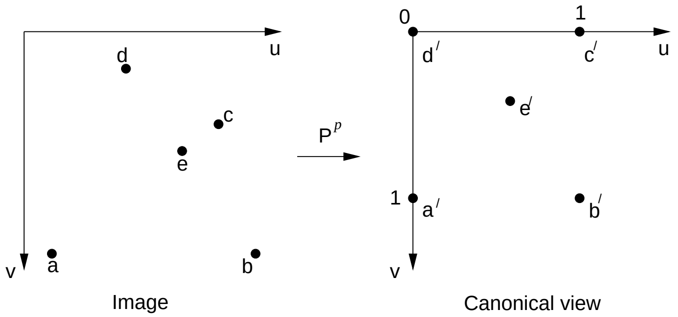
$$\tilde{\mathbf{w}} = \mathrm{P}^p \tilde{\mathbf{X}}^p$$

where P^p is a 3×3 matrix. It follows that *any* two views of the plane are related by a projectivity. If $\tilde{\mathbf{w}}'$ is another view, then

$$\tilde{\mathbf{w}}' = \mathrm{P}^{p\prime} \tilde{\mathbf{X}}^p = \mathrm{P}^{p\prime}[\mathrm{P}^p]^{-1}\tilde{\mathbf{w}} = \mathrm{P}^{p\prime\prime}\tilde{\mathbf{w}}$$

So the two views $\tilde{\mathbf{w}}$ and $\tilde{\mathbf{w}}'$ are related by a projectivity $\mathrm{P}^{p\prime\prime}$.

We can exploit this to construct a **canonical view** of the plane from any image. In the canonical view, four of the points lie at fixed, pre-determined locations, usually the corners of the unit square.

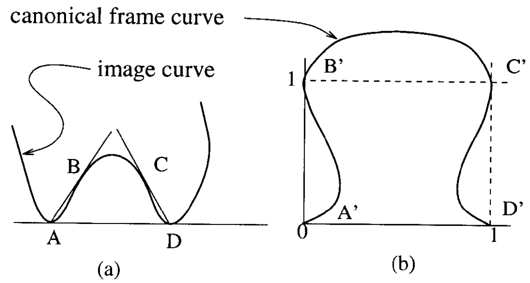


Canonical views

a . . . d are the four distinguished points, and we use their coordinates in the image to find the 3×3 projectivity P^p which maps them onto the corners of the unit square in the canonical view. This is simply a calibration process. If we then apply the projectivity P^p to the 5th point e, its coordinates u and v in the canonical view provide us with two projective invariants.

Canonical views have proved very successful for recognition of planar objects. Four distinguished points are used to map the structure in the image into a canonical view. In the canonical view, the structure is compared with a model library to spot any match.

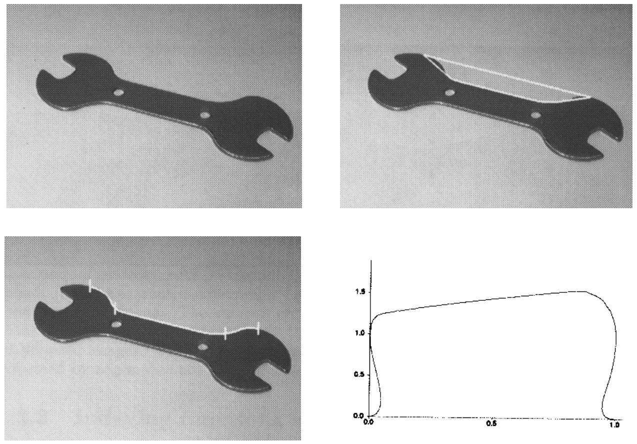
But how do we identify four distinguished points on curved outlines?



Canonical views

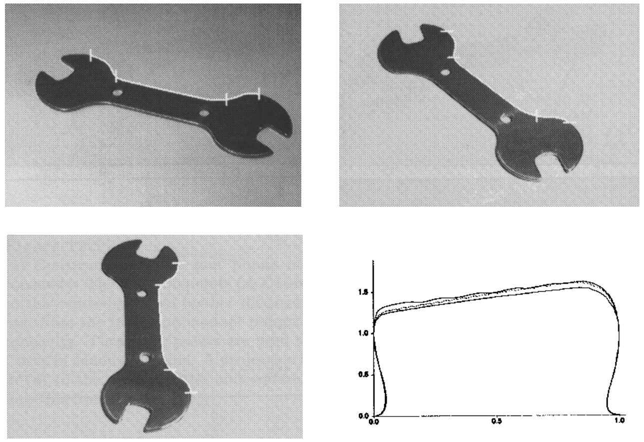
There is a nice construction which works for curve segments with concavities (many industrial parts). The bitangent across the concavity gives us two distinguished points A and D, then the tangents cast from A and D into the concavity give us another two, B and C.

Next we find the projectivity P^p which maps A, B, C and D onto the corners of the unit square in the canonical view, and the rest of the curve is mapped into the canonical view using P^p. What we end up with is an **invariant signature** of the curve in the canonical view.

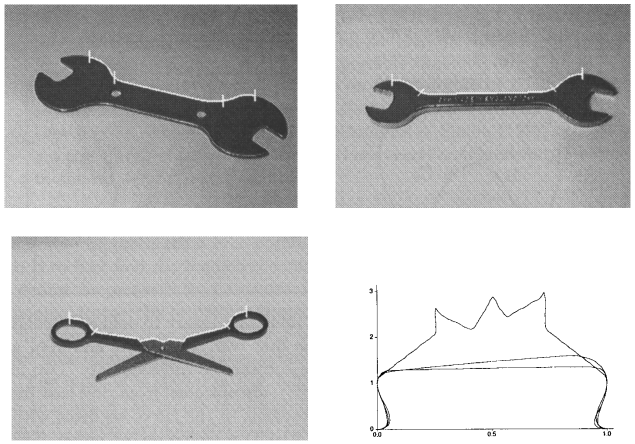


Canonical views

As expected, the construction produces the same signature for 3 different views of the spanner ...



... and different signatures for different objects.



Planar object recognition

Invariants, then, are extremely useful for planar object recognition:

- We identify structures in the image (groups of points, curves with concavities, etc.) and calculate projective invariants for them.
- An invariant which matches an entry in our model library suggests the presence of the object in the image.
- The hypothesis can be verified by "back-projecting" the library model from the canonical view into the image (using the projectivity $[\mathrm{P}^p]^{-1}$) and checking for edges around the projected outline.

We are still left with a large search problem: a typical image (like the one on page 46) might contain many curves and many distinguished points. Suppose we are using 5-point invariants in our recognition scheme. To make sure we don't miss an object, we need to check *every* set of 5 distinguished points. This leads to a rapid combinatorial explosion in the size of the search space.

Perceptual grouping

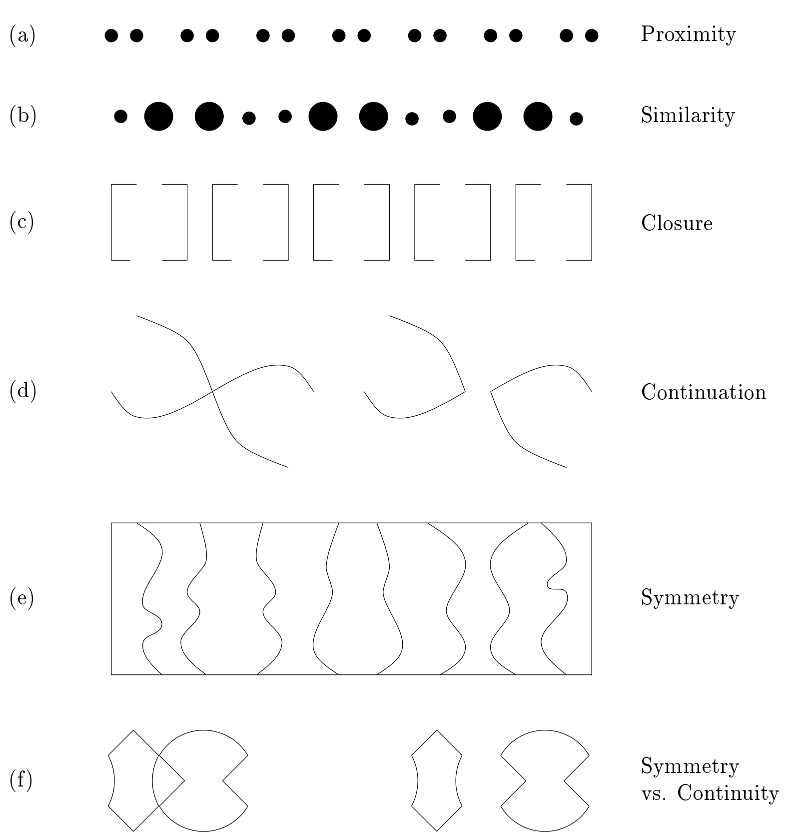
The search space can be reduced using **bottom-up segmentation**; we try to segment the image into likely objects before we've recognised them. Then, given this segmentation, we need only calculate invariants for features within the same putative object, greatly reducing the search space.

Segmentation can be attempted using **perceptual grouping**: the process of inferring structure in an image *without* exploiting any prior high-level knowledge of its content.

In perceptual grouping, features (edges, corners, curves, etc.) are grouped together on the basis of the **Gestalt principles** of proximity, similarity, closure, continuation and symmetry.

So, for example, if two distinguished points are close to each other, they are more likely to belong to the same object than two points which are distant. Using such principles, perceptual grouping can perform a coarse bottom-up segmentation of the image before invariants are calculated within groups.

Gestalt principles



The Gestalt principles were expounded by a group of German psychologists in the 1920's and 30's (the Gestalt psychologists). They were investigating the way humans subjectively group simple line and dot patterns.

Perceptual grouping

As an example, suppose we are trying to recognise rotationally symmetric objects. A Canny edge detector extracts many edges. Assuming we are using canonical view invariants, we have to search all these edges for concave curves and calculate a lot of invariants! Object recognition would be very slow.

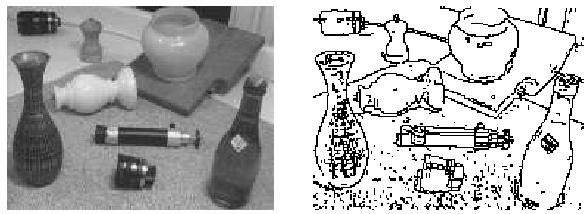
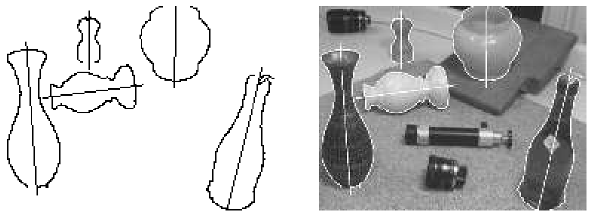


Image Detected edges



Grouped edges ... ...superimposed on image

Using the Gestalt principle of symmetry, however, we can quickly extract those curves which have matching, symmetric partners. We now only need to calculate invariants for a few curves, resulting in fast and efficient recognition.

Summary

3D $\rightarrow$ 2D camera models

Perspective

$\tilde{\mathbf{w}} = \mathrm{P}_c\mathrm{P}_p\mathrm{P}_r\tilde{\mathbf{X}} = \mathrm{P}_{ps}\tilde{\mathbf{X}}$. Tricky to calibrate.

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Projective

$\tilde{\mathbf{w}} = \mathrm{P}\tilde{\mathbf{X}}$. 11 degrees of freedom ($p_{34} = 1$). 6 points to calibrate.

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Weak perspective

$\tilde{\mathbf{w}} = \mathrm{P}_c\mathrm{P}_{pll}\mathrm{P}_r\tilde{\mathbf{X}} = \mathrm{P}_{wp}\tilde{\mathbf{X}}$. Tricky to calibrate.

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 0 & Z_c^{av} \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Affine

$\tilde{\mathbf{w}} = \mathrm{P}_{aff}\tilde{\mathbf{X}}$. 8 degrees of freedom ($p_{34} = 1$). 4 points to calibrate.

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ 0 & 0 & 0 & p_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Summary

2D $\rightarrow$ 2D camera models

Perspective

$\tilde{\mathbf{w}} = \mathrm{P}_c\mathrm{P}_p\mathrm{P}_r^p\tilde{\mathbf{X}}^p = \mathrm{P}_{ps}^p\tilde{\mathbf{X}}^p$. Tricky to calibrate.

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & T_x \\ r_{21} & r_{22} & T_y \\ r_{31} & r_{32} & T_z \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Projective

$\tilde{\mathbf{w}} = \mathrm{P}^p\tilde{\mathbf{X}}^p$. 8 degrees of freedom ($p_{33} = 1$). 4 points to calibrate.

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Weak perspective

$\tilde{\mathbf{w}} = \mathrm{P}_c\mathrm{P}_{pll}\mathrm{P}_r^p\tilde{\mathbf{X}}_\mathrm{p} = \mathrm{P}_{wp}^p\tilde{\mathbf{X}}^p$. Tricky to calibrate.

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 0 & Z_c^{av} \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & T_x \\ r_{21} & r_{22} & T_y \\ r_{31} & r_{32} & T_z \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Affine

$\tilde{\mathbf{w}} = \mathrm{P}_{aff}^p\tilde{\mathbf{X}}^p$. 6 degrees of freedom ($p_{33} = 1$). 3 points to calibrate.

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ 0 & 0 & p_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Summary

1D $\rightarrow$ 1D camera models

Perspective

$\tilde{\mathbf{w}} = \mathrm{P}_c \mathrm{P}_p \mathrm{P}_r^l \tilde{\mathbf{X}}^l = \mathrm{P}_{ps}^l \tilde{\mathbf{X}}^l$. Tricky to calibrate.

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & T_x \\ r_{21} & T_y \\ r_{31} & T_z \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ 1 \end{bmatrix}$$

Projective

$\tilde{\mathbf{w}} = \mathrm{P}^l \tilde{\mathbf{X}}^l$. 5 degrees of freedom ($p_{32} = 1$). 3 points to calibrate.

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \\ p_{31} & p_{32} \end{bmatrix} \begin{bmatrix} X \\ 1 \end{bmatrix}$$

Weak perspective

$\tilde{\mathbf{w}} = \mathrm{P}_c \mathrm{P}_{pll} \mathrm{P}_r^l \tilde{\mathbf{X}}^l = \mathrm{P}_{wp}^l \tilde{\mathbf{X}}^l$. Tricky to calibrate.

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 0 & Z_c^{av} \end{bmatrix} \begin{bmatrix} r_{11} & T_x \\ r_{21} & T_y \\ r_{31} & T_z \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ 1 \end{bmatrix}$$

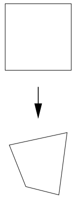
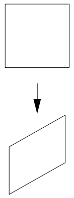
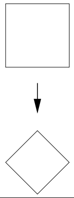
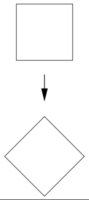
Affine

$\tilde{\mathbf{w}} = \mathrm{P}_{aff}^l \tilde{\mathbf{X}}^l$. 4 degrees of freedom ($p_{32} = 1$). 2 points to calibrate.

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \\ 0 & p_{32} \end{bmatrix} \begin{bmatrix} X \\ 1 \end{bmatrix}$$

Summary

The geometric strata: planar geometry

Group	Matrix	Distortion	Invariants
projective 8 DOF	$\begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}$		concurrency and collinearity, order of contact, tangent discontinuities and cusps, cross-ratio of four collinear points, measurements in canonical view
affine 6 DOF	$\begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ 0 & 0 & p_{33} \end{bmatrix}$		all the above, plus parallelism, ratio of areas, ratio of lengths on collinear or parallel lines (eg. midpoints)
similarity 4 DOF	$\begin{bmatrix} r_{11} & r_{12} & T_x \\ r_{21} & r_{22} & T_y \\ 0 & 0 & s \end{bmatrix}$		all the above, plus ratio of lengths, angle
Euclidean 3 DOF	$\begin{bmatrix} r_{11} & r_{12} & T_x \\ r_{21} & r_{22} & T_y \\ 0 & 0 & 1 \end{bmatrix}$		all the above, plus length, area

Note how invariants of the more general transformations are inherited by the more restricted transformations.

Bibliography

Some of the figures were taken from the following, which make good further reading.

History of perspective

M. Kemp. *The Science of Art*. Yale University Press, 1990.

Projective geometry

J. L. Mundy and A. Zisserman. Projective geometry for machine vision. In J. L. Mundy and A. Zisserman, eds. *Geometrical Invariance in Computer Vision*. MIT Press, 1992. (Figures on page 7.)

Invariants

J. L. Mundy and A. Zisserman, eds. *Geometrical Invariance in Computer Vision*. MIT Press, 1992. (Figures on pages 48, 49, 50, 51, 52, 57, 59, 60, 61.)

Planar object recognition

C. A. Rothwell. *Object Recognition through Invariant Indexing*. Oxford University Press, 1995. (Figures on page 46.)

Perceptual grouping

D. G. Lowe. *Perceptual Organization and Visual Recognition*. Kluwer Academic Publishers, 1985.

A. Zisserman et al. Class-based grouping in perspective images. *Proceedings of the International Conference on Computer Vision*, 183–188, Boston, 1995. (Figures on page 65.)