Part I

An Automatic Video Segmentation System

Equations are just the boring part of mathematics.
I attempt to see things in terms of geometry.
(Stephen Hawking)

Chapter 2

Projective Geometry

The geometric relations between objects in the 3-D world and a 2-D image of them is of central importance when we want to estimate the motion of the camera from a sequence of images. In particular, we need a geometric model that describes the observed motion fields resulting from different kinds of camera motion. Object motion and image formation can be expressed mathematically with the concept of projective geometry. This chapter gives an introduction to projective geometry as far as it is required to understand the subsequent chapters. We start with defining the projective space and deriving basic operations on points and lines in it. We proceed by discussing geometric transformations in the projective plane and in the 3-D Euclidean space, including the projection of 3-D space onto a 2-D image plane. Finally, we construct a detailed model of the image formation process for a moving camera. The reader familiar with projective geometry and its application to computer vision can jump to the concluding section of this chapter where we summarize the notation that will be used for the following chapters.

2.1 Introduction

The theory of projective geometry establishes the basis for three-dimensional computer vision and computer graphics. Compared to Euclidean geometry, it facilitates the description of rigid[1] three-dimensional motion and the perspective projection onto planar images, because it enables to formulate both with linear algebra techniques. In particular, projective geometry provides a uniform description of situations that require special cases in Euclidean geometry, like the intersection of parallel lines. Consequently, projective geometry has become the standard technique to describe three-dimensional geometry.

Since we employ the mathematical tools of projective geometry in many of the succeeding chapters, this chapter gives an introduction to projective geometry. Thereby, we concentrate on the aspects that are required in our applications. First, we describe the basic notion of homogeneous coordinates and we show how points and lines are represented in two-dimensional images. Second, we consider geometric transformations in the plane. We show how elementary operations like translations or rotations can be described and we introduce the classes of affine and projective motion. Third, the planar transformations are generalized to three-dimensional rigid motion. This gives us the required tools to describe the complete geometric image formation process as a concatenation of elementary operations. Finally, we deduce motion models for important classes of camera motion like rotational camera motion.

While this chapter only gives a brief introduction, a thorough discussion of 3-D geometry, estimation of camera parameters, and multi-view geometry can be found in the books [85] and [69]. Book [85] concentrates on the description of camera geometry for 3-D reconstruction, while [69] covers a wider range and also includes computer-vision algorithms, e.g., for multi-view tracking of objects. A detailed introduction to the mathematical theory of projective geometry can be found in [166]. Many practical aspects of projective transformations for image warping are described in [90]. These include optimal filtering of the transformed image to prevent aliasing and fast algorithms to compute transformation parameters and inverse transforms. Finally, [99] and [127] provide condensed introductions to the field.

[1] Rigid: shape and size preserving. Motion of non-deformable solid objects.

2.2 Projective spaces

Elementary geometry is usually described in the Euclidean space, which is a direct description of space as we perceive it with our human intuition. In this formulation, points in n-dimensional Euclidean space $\mathbb{E}^n$ are simply represented as vectors of length n. However, this supposedly convenient definition of Euclidean space has several practical drawbacks. First, there is no notion about points at infinity, so that these must be considered separately as a special case. For example, this problem arises when computing the intersection of parallel lines, which is not defined in Euclidean space. A second drawback becomes apparent when we use geometric transforms to describe object motion. Even for the basic types of motion, different formalisms are required. While translation is described using a vector addition, rotation is written as a matrix multiplication, and the perspective projection of points onto a plane requires a division operation.

The concept of *projective spaces*, provides an alternative convenient formalism to describe all types of rigid motion and the perspective projection in a unified way. Moreover, points at infinity are an integral part of the projective space and require no special consideration.

The remainder of Section 2.2 introduces the construction of projective space, its relationship to Euclidean space, and it introduces basic operations on points and lines.

2.2.1 Homogeneous coordinates

In the n-dimensional Euclidean space $\mathbb{E}^n$, each point is written as a vector of length n, where each component is the position along one coordinate axis. In contrast, points in n-dimensional projective space $\mathbb{P}^n$ are represented by $n+1$ dimensional vectors. The construction is such that each point $(x_1, \ldots, x_n)^\top$ in $\mathbb{E}^n$ corresponds to a one-dimensional subspace $(wx_1, \ldots, wx_n, w)^\top$ in $\mathbb{P}^n$ with the free scaling parameter $w \neq 0$. Euclidean space $\mathbb{E}^n$ can be embedded into $\mathbb{P}^n$ in a simple way by using the canonical injection $\mathbb{E}^n \ni (x_1, \ldots, x_n)^\top \mapsto (x_1, \ldots, x_n, 1)^\top \in \mathbb{P}^n$. In the reverse direction, Euclidean coordinates can be recovered by the mapping

$$\mathbb{P}^n \ni (x_1, \ldots, x_n, w)^\top \longmapsto \left(\frac{x_1}{w}, \ldots, \frac{x_n}{w}\right)^\top \in \mathbb{E}^n. \tag{2.1}$$

This projective representation is denoted as the point's *homogeneous coordinates*. A direct consequence of the definition is the important property that homogeneous coordinates are scaling invariant. Thus, $(x_1, \ldots, x_n, w)^\top$ and $(\lambda x_1, \ldots, \lambda x_n, \lambda w)^\top$ represent the same point for all $\lambda \neq 0$. In other words, each Euclidean point $\mathbf{p}$ is represented by one equivalence class $E(\mathbf{p}) \subset \mathbb{P}^n$

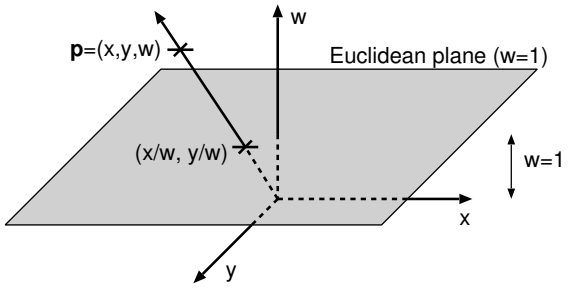


Figure 2.1: *The Euclidean plane* $\mathbb{E}^2$ *can actually be viewed as the plane* $w = 1$ *in projective space* $\mathbb{P}^2 = \{(x, y, w)^\top\}$.

in the projective space. All vectors in the equivalence class can be obtained by a uniform scaling of the coordinates.

Projective points with $w = 0$ represent *ideal* points at infinity with no correspondence in Euclidean space. For these points at infinity, the first n vector components indicate the direction in which the point is located. The non-sense null-vector $(0, \ldots, 0)^\top$ is not included in $\mathbb{P}^n$.

Visualization of $\mathbb{P}^2$

In the following discussion, the projective plane $\mathbb{P}^2$ is of central importance. For this case, the relation between points in Euclidean space $\mathbb{E}^2$ and corresponding points in projective space $\mathbb{P}^2$ can be visualized easily. We draw $\mathbb{P}^2 = \{(x, y, w)^\top\}$ as a three-dimensional space with the dimensions x, y, w (see Figure 2.1). In this space, the equivalence class induced by a point $\mathbf{p}$ in $\mathbb{P}^2$ can be visualized as the line through the origin and the point $\mathbf{p}$. The intersection of this line with the plane $w = 1$ defines the coordinate in the Euclidean plane. Since a scaling of the homogeneous coordinates of point $\mathbf{p}$ by a non-zero constant only moves the point along the line, its position on the Euclidean plane stays the same. This explains the scaling invariance of homogeneous coordinates. By choosing the special representation with the last coordinate $w = 1$, it becomes clear that the Euclidean plane itself can be embedded into the projective space as the plane at $w = 1$.

Ideal points $\mathbf{p}$ with $w = 0$ correspond to lines parallel to the Euclidean plane and thus intersect them *ideally* at infinity. Since these lines are parallel to the Euclidean plane $w = 1$, the ideal points are not part of the Euclidean space. This is consistent with the definition of Euclidean space, because it has no notion about points at infinity.

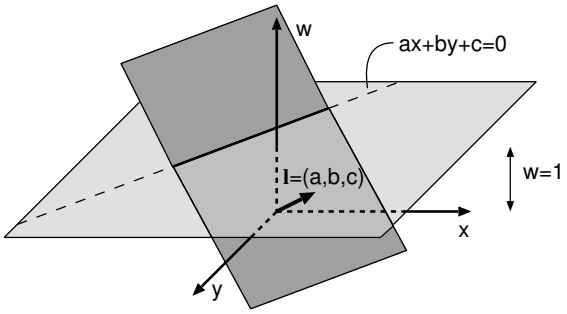


Figure 2.2: *Lines* $\mathbf{l} = (a, b, c)^\top$ *in the* $\mathbb{P}^2$ *can be represented as planes through the origin with normal vector* $\mathbf{l}$*. The intersection of this plane with the Euclidean plane forms the corresponding line in the Euclidean plane.*

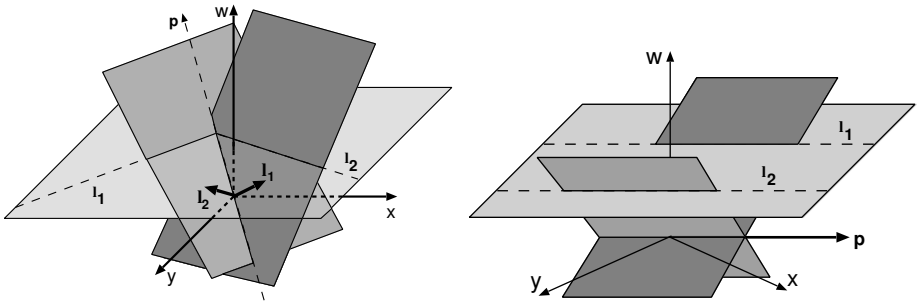
Normalization of homogeneous coordinates

It is common practice to normalize homogeneous coordinates to have unit norm $||\mathbf{p}|| = 1$ during computations. Regarding the above-mentioned visualization model, this has the effect that all points in homogeneous coordinates lie on the unit sphere around the origin of $\mathbb{P}^2$. The normalization resolves numerical problems that can occur when working with coordinates of significantly different magnitude. We do not explicitly mention the normalization of coordinates in the following, but simply note that this normalization procedure can be applied at arbitrary times during the computation, whenever it seems appropriate.

2.2.2 Lines in the projective plane

In Euclidean geometry, lines are defined as $ax + by + c = 0$, where a, b, c are the line parameters. Note that these line parameters are invariant to scaling with a constant $w \neq 0$, since $w(ax) + w(by) + wc = 0$ describes the same line. If we denote the line parameters by the vector $\mathbf{l} = (a, b, c)^\top$ and specify points using homogeneous coordinates $\mathbf{p} = (wx, wy, w)^\top$, we can rewrite the line equation conveniently as $\mathbf{l}^\top \cdot \mathbf{p} = 0$.

It is possible to visualize lines in the projective plane in the same way as we did with points in the last section. The construction represents each line in $\mathbb{E}^2$ as a plane in $\mathbb{P}^2$ which goes through the origin and which has a normal vector equal to the line parameters $\mathbf{l}$. This plane intersects the Euclidean plane $w = 1$ in the desired line (see Figure 2.2). Since the norm of the plane normal is irrelevant, the line parameters are also invariant to scaling with a non-zero value.



(a) Intersection of lines. (b) Intersection of parallel lines.

Figure 2.3: *(a) The intersection point* $\mathbf{p}$ *of two lines* $\mathbf{l_1}, \mathbf{l_2}$ *can be calculated by* $\mathbf{p} = \mathbf{l_1} \times \mathbf{l_2}$*, since the vector* $\mathbf{p}$ *must lie on both planes and hence, it must be orthogonal to* $\mathbf{l_1}$ *and* $\mathbf{l_2}$*. (b) For two parallel lines, the intersection is an ideal point at infinity.*

Similarly to the points at infinity, a special *line at infinity* is defined by $\mathbf{l}_\infty = (0, 0, 1)^\top$. Since $\mathbf{l}_\infty^\top \cdot (x, y, w)^\top = 0$ iff $w = 0$, the line $\mathbf{l}_\infty$ is the set of all points at infinity. Note that just like the points at infinity, $\mathbf{l}_\infty$ has no correspondence in Euclidean geometry. In our visualization, where we consider the vector of line parameters to be a plane normal, $\mathbf{l}_\infty$ defines a plane parallel to the Euclidean plane, intersecting it ideally at infinity (see Figure 2.4(b)).

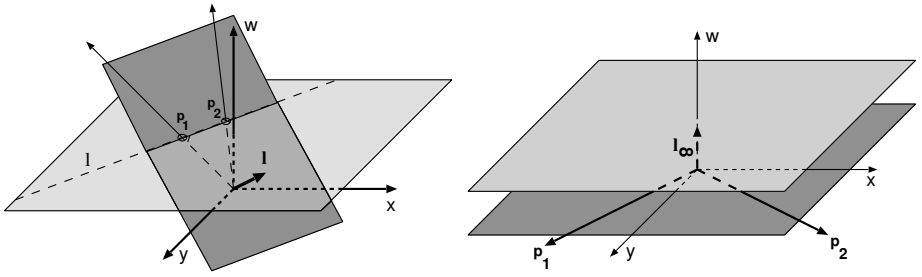
Intersection of lines

Let $\mathbf{l_1}$ and $\mathbf{l_2}$ denote two lines in $\mathbb{P}^2$. Visualized in the (x, y, w)-space, these can be viewed as two planes through the origin with normal vectors $\mathbf{l_1}$ and $\mathbf{l_2}$ as shown in Figure 2.3(a). The intersection point $\mathbf{p}$ of the two lines $\mathbf{l_1}, \mathbf{l_2}$ obviously must lie on both planes in the (x, y, w)-space. This means that $\mathbf{p}^\top \cdot \mathbf{l_1} = 0$ and $\mathbf{p}^\top \cdot \mathbf{l_2} = 0$ must hold. In other words, $\mathbf{p}$ is orthogonal to both $\mathbf{l_1}$ and $\mathbf{l_2}$. Consequently, we can compute it as $\mathbf{p} = \mathbf{l_1} \times \mathbf{l_2}$.

Note that when using homogeneous coordinates, the intersection of two parallel lines is well defined and results in an ideal point at infinity (Figure 2.3(b)). In this case, the intersection point has $w = 0$, and the other two coordinates indicate the direction towards the intersection at infinity.

Line through two points

Let $\mathbf{p_1}$ and $\mathbf{p_2}$ be two points in $\mathbb{P}^2$. Following a similar approach as above, we remember that $\mathbf{p_1}$ and $\mathbf{p_2}$ can be visualized by two rays, starting from the origin of the (x, y, w)-space (Figure 2.4). To find the projective parame-



(a) Line through two points. (b) Line through two points at infinity.

Figure 2.4: *The line* $\mathbf{l}$ *through two points* $\mathbf{p_1}, \mathbf{p_2}$ *is defined by a plane, which must include the two vectors* $\mathbf{p_1}, \mathbf{p_2}$*. Consequently, its normal vector* $\mathbf{l}$ *must be orthogonal to both vectors* $\mathbf{p_1}, \mathbf{p_2}$*.*

ters of the line through $\mathbf{p_1}$ and $\mathbf{p_2}$, we have to find a plane in (x, y, w)-space which contains $\mathbf{p_1}$, $\mathbf{p_2}$ and the origin. If we specify this plane by its plane normal $\mathbf{l}$, this means that $\mathbf{l}^\top \cdot \mathbf{p_1} = 0$ as well as $\mathbf{l}^\top \cdot \mathbf{p_2} = 0$ must hold. Therefore, we can determine $\mathbf{l}$ by using the cross product $\mathbf{l} = \mathbf{p_1} \times \mathbf{p_2}$. Note that the line through two points at infinity results in the line $\mathbf{l}_\infty$.

The similarity of the expressions for computing intersection points between two lines and the expression for computing the line through two points is not accidental. In fact, it is one example of the *duality principle* between points and lines in projective space. This fundamental property implies that every theorem in projective space $\mathbb{P}^2$ stays true if all references to lines and points are interchanged.

2.3 Geometric transformations in 2-D

In this section, we introduce the projective transformation, which is used frequently to describe object motion as a geometric transformation in the image plane. We also derive some basic properties of projective transforms and introduce affine motion as an important sub-class of projective transforms. The class of affine transformations allows us to develop an intuitive understanding of the physical meaning of the transform parameters. Finally, we see how the projective motion model, which is commonly used for motion analysis in its inhomogeneous formulation, can be derived directly from the homogeneous definition.

2.3.1 Projective transformation

A *projective transformation* is defined as a linear transformation between homogeneous coordinates. We only consider non-degenerate cases where the transform is invertible. Since the transform is linear and invertible, it can be written as a multiplication with a non-singular matrix $\mathbf{H} = \{h_{ik}\}$. For a projective space $\mathbb{P}^n$, the matrix is of size $(n+1) \times (n+1)$. In the planar case $\mathbb{P}^2$, we get specifically

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{pmatrix} x \\ y \\ w \end{pmatrix}. \tag{2.2}$$

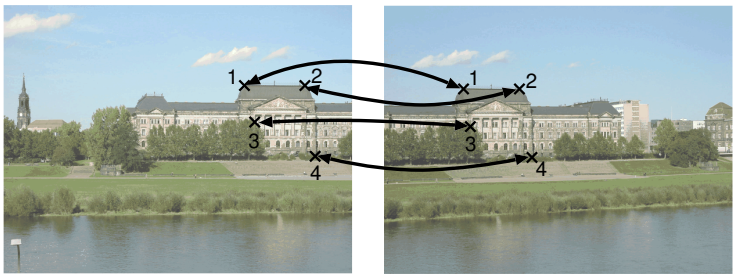
The matrix entries $\{h_{ik}\}$ are the transform parameters. Note that because of the use of homogeneous coordinates, the transformation matrix is only defined up to a scaling factor. Hence, in $\mathbb{P}^2$, the transform has only 8 degrees of freedom even though the transformation matrix has nine elements.

Outlook: Estimation of projective transforms.

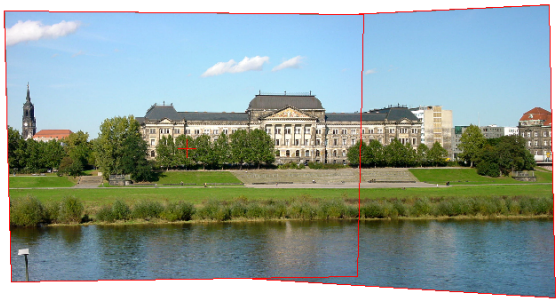
To estimate the transformation between two frames, we will investigate feature-based motion estimators in Chapters 3 and 4. The idea is that we can identify a set of points $(x_i, y_i, 1)$ in one image and a corresponding set of points $(x'_i, y'_i, 1)$ in the second image. When inserting pairs of points into Eq. (2.2), we obtain two constraints for each pair of points. With at least four pairs of points and one additional constraint (e.g., $h_{22} = 1$) to remove the scaling invariance, we can determine the transformation between the two images. These transformation parameters can then be used, e.g., to stitch both images together (Fig. 2.5).

Equivalence to collineations and the mapping of lines

A *collineation* in the plane is defined as a transform that maps lines onto lines. It can be shown that every collineation can be written as a projective transformation, and vice versa. In the following, we only give the proof for one direction. Given a non-singular transformation matrix $\mathbf{H} = \{h_{kl}\}$, we can easily show that lines are always mapped onto lines. To prove this, let $\{\mathbf{p_i}\}$ be a set of points that all lie on a line $\mathbf{l}$, so $\mathbf{l}^\top \cdot \mathbf{p_i} = 0$. Since we assumed that $\mathbf{H}$ is invertible, this equals $\mathbf{l}^\top(\mathbf{H}^{-1}\mathbf{H})\mathbf{p_i} = 0$. By changing parenthesis, we can read this equation as $\left(\mathbf{l}^\top\mathbf{H}^{-1}\right)\left(\mathbf{H}\mathbf{p_i}\right) = 0$. But this means that the transformed points $\mathbf{H}\mathbf{p_i}$ all lie on the line $\mathbf{l}^\top\mathbf{H}^{-1}$. Thus, the projective transformation preserved lines. The reverse direction, which



(a) Two images with four point-correspondences.



(b) Both images can be aligned using the computed projective transform.

Figure 2.5: *Images captured with a rotating camera can be aligned by identifying four point-correspondences and computing the projective transformation matrix from these correspondences.*

states that every collineation can be written as a projective transform, is far more complicated to show, and the proof is omitted here.

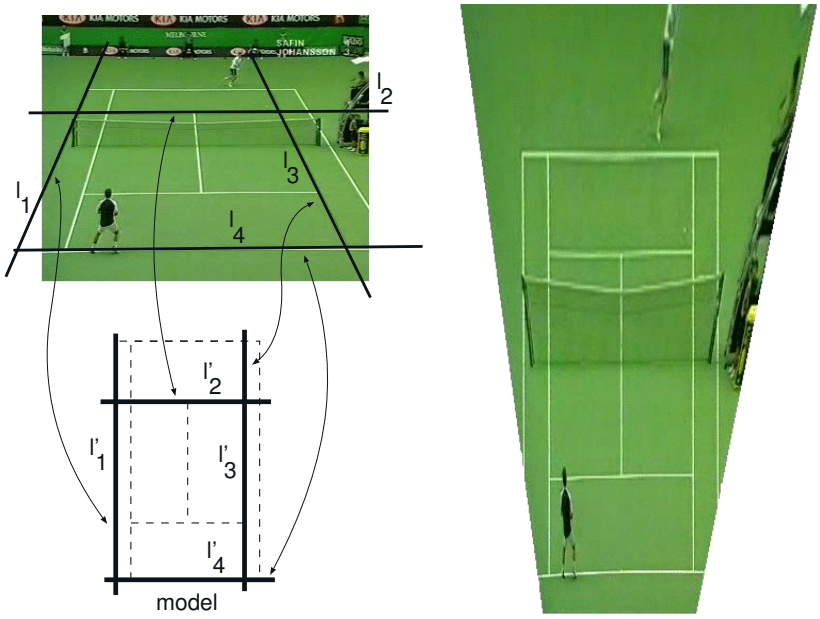
The above proof also gives us a useful side-result. For a given transformation $\mathbf{p}' = \mathbf{H}\mathbf{p}$ between points, we can find a corresponding transformation that maps a line parameter vector $\mathbf{l}$ to the line parameters $\mathbf{l}'$ of the transformed line. From the above proof, it follows immediately that

$$\mathbf{l}'^{\top} = \mathbf{l}^{\top}\mathbf{H}^{-1} \quad \text{or, equivalently,} \quad \mathbf{l}' = \mathbf{H}^{-\top}\mathbf{l}. \tag{2.3}$$

Consequently, we can say that for a point-transform $\mathbf{H}$, the corresponding line-transform is $\mathbf{H}^{-\top}$.

Outlook: Estimation of transform parameters based on corresponding lines.

Equation (2.3) does not only provide the parameters of lines after a transformation. The equation can also be used to determine the transformation parameters if a set of corresponding lines are known in two images. The approach is similar to estimating the transformation parameters from a set of points, with the only difference that an additional matrix inversion has to be computed. In some applications, the estimation of lines is more reliable than the estimation of points, and the approach to estimate the transformation directly from the lines is more convenient (Fig. 2.6). We use this technique in Chapter 13 to obtain a camera calibration for sport sequences. In this case, we detect the lines of the sports court and match them with lines in a model of the court.



(a) An image of a tennis court and a tennis court model. Four corresponding lines are indicated.

(b) Using a projective transform obtained from the line correspondences, the tennis court image can be rectified to the real-world geometry.

Figure 2.6: *Rectifying a tennis court image based on the court lines. It is more robust to determine the transformation from line correspondences, because the court-lines are not occluded as often as specific feature-points.*

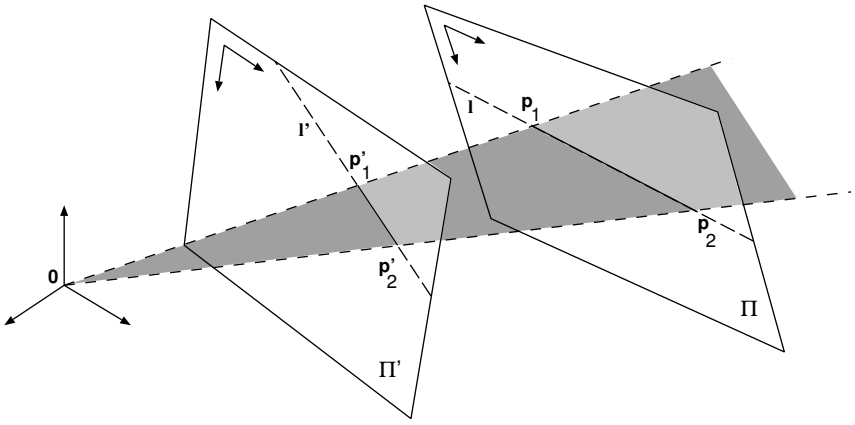


Figure 2.7: *In a projective plane-to-plane mapping, points on plane* Π *are projected along rays starting from the origin* $\mathbf{0}$ *onto the plane* Π'*. This plane-to-plane transform maps lines on* Π *onto lines on* Π'*.*

Equivalence to perspective plane-to-plane mapping

One transform of special importance for computer graphics is the perspective projection of one plane onto another. As depicted in Figure 2.7, points $\mathbf{p_i}$ on plane Π are projected onto points $\mathbf{p'_i}$ on plane Π' along rays emanating from the origin $\mathbf{0}$. Consider the line $\mathbf{l}$ through the points $\mathbf{p_1}$ and $\mathbf{p_2}$. The plane through the points $\mathbf{0}, \mathbf{p_1}, \mathbf{p_2}$ intersects Π at $\mathbf{l}$. The same plane will intersect Π' in a line $\mathbf{l'}$. Consequently, each perspective plane-to-plane mapping is also a collineation and therefore a projective transform.

2.3.2 Affine motion

Before we discuss properties of the general perspective transform, let us consider the important sub-class of affine motion. From elementary geometry, we know that planar affine motion in $\mathbb{E}^2$ is written as

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}, \tag{2.4}$$

comprising a 2×2 transformation matrix $\mathbf{A} = \{a_{kl}\}$ and a translation vector $\mathbf{t} = (t_x, t_y)^\top$. The matrix $\mathbf{A}$ can always be factorized into a sequence of the elementary operations *rotation*, *non-isotropic scaling*, and *skewing*. The

transformation matrices for each of these elementary operations is given in Table 2.1. It is easy to see that all of these elementary operations map parallel lines onto parallel lines. Consequently, this property also holds for the general affine transform, since it can be decomposed into a sequence of elementary operations.

We can also write the affine transform using homogeneous coordinates, in which case we can integrate the translation vector into the matrix, leading to the unified formulation

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{bmatrix} a_{00} & a_{01} & t_x \\ a_{10} & a_{11} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ w \end{pmatrix} . \tag{2.5}$$

The affine transformation has six degrees of freedom. However, in this general form, the affine transformation includes types of motion that are no valid motion of rigid objects. Therefore, we can further restrict the transformation by disallowing skewing (force $k = 0$) and allowing only isotropic scaling ($s := s_1 = s_2$). As a consequence, only four parameters remain, namely one for rotation, one for isotropic scaling, and two for the translation vector. In this case, we call the transform a *similarity transform*, which can be expressed as

$$\begin{bmatrix} s\cos\alpha & s\sin\alpha & t_x \\ -s\sin\alpha & s\cos\alpha & t_y \\ 0 & 0 & 1 \end{bmatrix} , \tag{2.6}$$

with α being the rotation angle and s representing the isotropic scaling factor.

In Section 2.4.3, we will see that the affine motion model can describe the motion of planar objects in 3-D space if the camera is located at an infinite distance. Even though this is never true in practice, the affine camera model is still often used as an approximation if the distance between camera and scene is large and the motion is small.

2.3.3 Projective motion

After we have covered the case of affine motion and described it as as a sub-class of projective motion, we now make the step to the general projective transformation. If we recall Equation (2.5) for affine motion in homogeneous notation, we see that the last row of the matrix is always $(0, 0, 1)$. Therefore, $w' = w$ holds for every possible transform, i.e., the transform does not change the w-coordinate. Considering our visualization model from Figure 2.1, this means that motion of a point is only performed

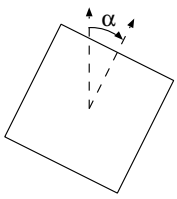
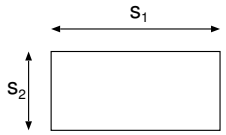
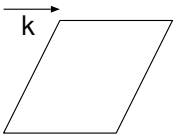
rotation	non-isotropic scaling	skewing
$\begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}$	$\begin{bmatrix} s_1 & 0 \\ 0 & s_2 \end{bmatrix}$	$\begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix}$
		

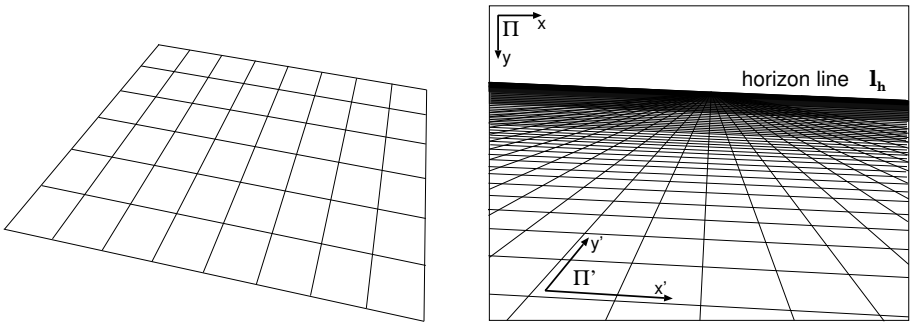
Table 2.1: *Transformation matrices* $\mathbf{A}$ *for the elementary affine transforms, written in inhomogeneous form* $\mathbf{p}' = \mathbf{A}\mathbf{p} + \mathbf{t}$. *Isotropic scaling is obtained for* $s_1 = s_2$.

within a plane parallel to the Euclidean projection plane. Since no change of *depth* occurs, we do not observe a perspective projection effect.

If we generalize the model such that the last line can hold arbitrary values, we have the general projective transformation. As we have seen previously, this models a general plane-to-plane transform, which allows to describe the 3-D motion of a plane, viewed by a pinhole camera. The difference to the affine transform is that, in general, parallel lines do not remain parallel after the transform (Figure 2.8(a)). Instead, the projective transform has the property that parallel lines are mapped to lines that intersect in a common vanishing point. Note that this includes the special case that parallel lines remain parallel, since parallel lines are intersecting at an ideal point at infinity. Because parallel lines intersect at the same vanishing point, it is clear that the location of the vanishing point can only depend on the direction of the lines in the source image.

The horizon and the line at infinity

The set of all vanishing points is called the vanishing line. In the real world, we know this vanishing line of the ground plane as the *horizon* (Figure 2.8(b)). Since the horizon line and its relationship to the line at infinity plays a special role, we explore in this section how the parameters of the horizon line can be obtained from a transformation matrix. Consider the case depicted in Figure 2.8(b), where an object plane Π' is projected by a general projective transformation onto an image plane Π. Let us denote the transform of image coordinates back onto the object plane by $\mathbf{H}$. In this formulation, we can define the horizon line as those points $\mathbf{p}$ that are mapped to points at infinity by the transform $\mathbf{H}$. Recalling that points



(a) Parallel lines do not stay parallel, but they intersect in their vanishing point.

(b) The horizon is the line of all vanishing points.

Figure 2.8: *Planes under perspective projection.*

(x, y, w) lie at infinity iff $w = 0$, it must hold that

$$\begin{pmatrix} x' \\ y' \\ 0 \end{pmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{pmatrix} x \\ y \\ w \end{pmatrix}. \tag{2.7}$$

From this, we derive the constraint that for points on the horizon,

$$(h_{20}, h_{21}, h_{22}) \cdot \mathbf{p} = 0 \tag{2.8}$$

must hold. Since this equation has the same form as a general line equation, we can regard the last row of the matrix $\mathbf{H}$ as the line parameters for the horizon line $\mathbf{l_h}$. Hence, we have the simple result that

$$\mathbf{l_h} = (h_{20}, h_{21}, h_{22})^\top. \tag{2.9}$$

Let us finally examine the special case of an affine transform. We know already that the affine transform preserves parallelism. Consequently, the vanishing points will always lie on the line at infinity. This observation is supported by the fact that the last line of the affine transformation matrix is $(0, 0, 1)$, which equals $\mathbf{l}_\infty^\top$. Hence, the location of the line at infinity is invariant to any affine motion.

Inhomogeneous formulation of the projective transformation

Writing the projective transformation with the inhomogeneous formulation, we have

$$x' = \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + h_{22}}, \quad y' = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + h_{22}}. \tag{2.10}$$

Since the parameters $\{h_{kl}\}$ are invariant to an overall scaling, it is common practice to apply the normalization $h_{22} = 1$. Especially in the literature about motion estimation for video coding, the inhomogeneous form is usually chosen. However, it should be noted that this normalization is only possible if $h_{22} \neq 0$. It is important to know in which cases h_{22} equals zero to decide if the inhomogeneous form is applicable.

The examination about the line at infinity gives us more insight into the case. We know that the horizon line in the destination image is given by (h_{20}, h_{21}, h_{22}). The specific case $h_{22} = 0$ induces that the horizon line has the form $h_{20}x + h_{21}y = 0$, which is just the pencil of lines that goes through the origin. Consequently, the normalization $h_{22} = 1$ is invalid iff the horizon line includes the origin. If this situation can occur in a specific application, the inhomogeneous form should not be used.

In our application of describing object motion by a perspective transform between successive frames, motion is so small that the normalization is usually not a problem. On the other hand, if motion is estimated over a large time distance (as it is the case in the background sprite construction), this can be a problem. However, especially for background sprite generation, other factors make the long-distance motion description impractial, even before we approach the problem of the inhomogeneous formulation.

Ignoring the cases where the normalization $h_{22} = 1$ is invalid, we get the most widely-used formulation of projective motion as

$$x' = \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + 1}, \quad y' = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + 1}, \tag{2.11}$$

or, when using different symbols for the transformation parameters, to better reflect better their relation to the affine transform, we write it as

$$x' = \frac{a_{00}x + a_{01}y + t_x}{p_x x + p_y y + 1}, \quad y' = \frac{a_{10}x + a_{11}y + t_y}{p_x x + p_y y + 1}. \tag{2.12}$$

We see that the affine transformation results as a special case of the perspective transformation for $p_x = p_y = 0$. Because the denominator only disappears in the affine case, the perspective transformation is a non-linear transform. We will see in Chapter 4 that this complicates the parameter estimation process.

2.4 Geometric transformations in 3-D

To describe object or camera motion in 3-D, we make use of the same formulation as for 2-D. We also use homogeneous coordinates in $\mathbb{P}^3$ even

though we only consider affine motion. Affine motion is sufficient to describe rigid object motion, but the use of homogeneous coordinates enables to use the unified formulation in which affine motion can be described as a matrix multiplication.

2.4.1 Affine motion in 3-D

The generalization of the affine motion in 2-D to affine motion in 3-D is straightforward. In the 2-D case, we described affine motion in inhomogeneous coordinates by $\mathbf{p}' = \mathbf{A}\mathbf{p}+\mathbf{t}$, where $\mathbf{p}', \mathbf{p}, \mathbf{t}$ are two-component vectors and $\mathbf{A}$ is a 2×2 matrix. When we go to 3-D, we can take the same equation and simply substitute the vectors by three-component vectors and the matrix by a 3×3 matrix.

In the 2-D case, we used homogeneous coordinates to unify the formulation. Clearly, we can apply the same approach also in the 3-D generalization of affine motion by augmenting vectors with a homogenization element and matrices with an additional row and column. As a consequence, translation can now also be described as a simple matrix multiplication. In its general form, affine motion in 3-D is consequently described by a multiplication with a matrix, where the last row is $(0, 0, 0, 1)$:

$$\begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix} = \begin{bmatrix} a_{00} & a_{01} & a_{02} & t_x \\ a_{10} & a_{11} & a_{12} & t_y \\ a_{20} & a_{21} & a_{22} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}. \tag{2.13}$$

A collection of the most important elementary transformations in this homogeneous formulation is shown in Table 2.2. Besides the basic affine operations *translation*, *scaling*, and *rotation*, a special operation is included to carry out a perspective projection from 3-D space onto a 2-D plane. This is a special operation that will be described in Section 2.4.3.

translation	isotropic scaling	rotation	projection
$\begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix}$	$\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

Table 2.2: *Most important transformation matrices* $\mathbf{H}$ *expressed in homogeneous coordinates. Matrix* $\mathbf{R}$ *is a* 3×3 *rotation matrix; see Section 2.4.2 for details.*

2.4.2 Rotation in 3-D

Describing rotation in 3-D space is a surprisingly complicated problem. Several techniques for parameterization of rotations have been proposed, such as Euler angles or Quaternions [108]. We use the Euler-angle notation here since this mimics the frequently-used practice to describe rotation by elementary rotations around the coordinate system axes.

In the following, we only consider rotation around the coordinate system origin, which is sufficient for the successive discussion. Furthermore, we assume that the observer (i.e., the camera) is located at the origin and that observations are made in a local camera coordinate system.

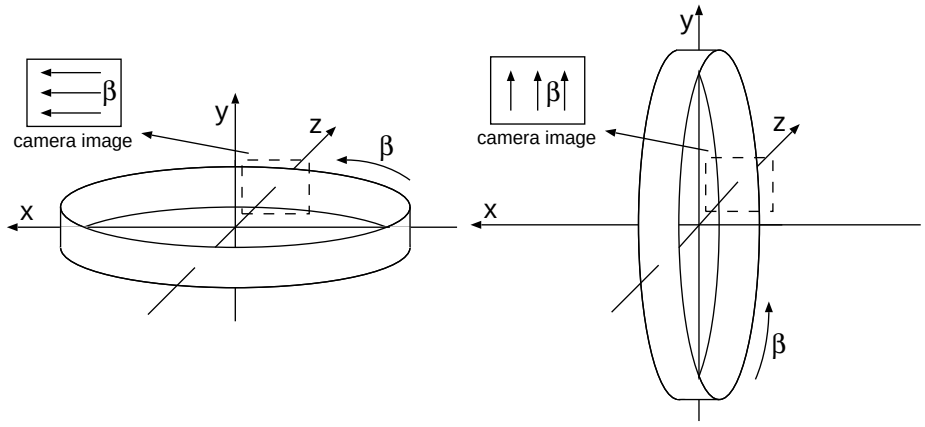
Let us begin with a simple rotation around one of the coordinate system axes, which can be described like a rotation in two dimensions where the third dimension is not affected. This can be written as an elementary rotation matrix (*Jacobi rotation matrix*). In 3-D space, we obtain the three elementary rotation matrices

$$\mathbf{R_x}(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\alpha & -s_\alpha \\ 0 & s_\alpha & c_\alpha \end{bmatrix}, \quad \mathbf{R_y}(\beta) = \begin{bmatrix} c_\beta & 0 & s_\beta \\ 0 & 1 & 0 \\ -s_\beta & 0 & c_\beta \end{bmatrix}, \quad \mathbf{R_z}(\gamma) = \begin{bmatrix} c_\gamma & -s_\gamma & 0 \\ s_\gamma & c_\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.14}$$

where we use the abbreviations $c_\alpha = \cos\alpha$, and $s_\alpha = \sin\alpha$. The signs of the $\sin(\cdot)$-terms have been chosen such that it conforms to a right-handed coordinate system (see Figure 2.11(a)). It is clear that the inverse rotation to some angle α is the rotation with the same angle in the opposite direction (i.e., a rotation with the angle $-\alpha$). Since the rotation matrices are skew symmetric and $\sin(-\alpha) = -\sin\alpha$, the inverse of each elementary rotation matrix is simply its transpose ($\mathbf{R}^{-1} = \mathbf{R}^\top$).

According to *Euler's rotation theorem*, an arbitrary rotation can be decomposed into three successive rotations around predefined axes. Consequently, we can choose a sequence of three elementary rotation matrices to describe every possible rotation. Obviously, two successive rotations must use different elementary rotations, since rotating two times around the same axis can be trivially combined into only one rotation. However, it is possible to use the same axis twice (e.g., a rotation sequence using a Z-Y-Z axes order is also capable to describe all possible rotations).

Rotations are not commutative and hence, the order in which we perform the rotations is important. Moreover, without further restrictions on allowed rotation angles, the same rotation can often be specified with different sets of parameters even if the order of rotation axes is fixed.



(a) $\gamma = 0$, rotation with pan angle β shows as horizontal motion.

(b) $\gamma = 90°$, rotation with pan angle β shows as vertical motion.

Figure 2.9: *Rotation with a fixed order of axes can result in non-intuitive behaviour. The camera is located at the origin and looks along the z-axis. In the rotation sequence* $\mathbf{R_z}(\gamma)\mathbf{R_y}(\beta)$*, we intuitively understand* β *as the horizontal pan. However, for* $\gamma \approx 90°$*, a change of* β *results in vertical motion.*

Euler angles vs. human intuition

The human understanding of rotations is strongly related to our physical environment. If there are several ways to describe the same rotation, we have a strong tendency to describe it in a way which corresponds best to the physically most probable action. For example, when we watch our image in the mirror, we perceive it with swapped left and right orientation. However, it could also be understood just as well with swapped up and down orientation. Only the fact that we turn around our vertical axis more frequently than we stand on our head makes the distinction.

Moreover, it seems that human intuition always considers rotation as an iterative action which is relative to the last state. Changing the angle of a rotation that is not the last one in the rotation sequence does not match our expectations. As an example, consider the set-up of Figure 2.9 with the rotation sequence $\mathbf{R_z}(\gamma)\mathbf{R_y}(\beta)$, which is a rotation by β around the vertical axis (camera pan), followed by a rotation by γ around the optical axis. Usually, we intuitively associate a camera pan (rotation around y-axis) with a left-right motion, which is true if we consider the rotation independently. However, if the pan is followed by another rotation, e.g., $\mathbf{R_z}(\gamma)$, the motion direction can change completely because of the second transform. If $\gamma \approx 90°$, a change of the camera pan angle β does not result

in horizontal motion, but it induces a vertical motion in the camera image.

This conflict with our intuition occurs, because we tend to think that changes of rotation angles occur relative to the last position. If we make a small change of camera pan $\Delta\beta$, our intuition suggests that the overall camera transform would be $\mathbf{R_y}(\Delta\beta)\mathbf{R_z}(\gamma)\mathbf{R_y}(\beta)$, instead of $\mathbf{R_z}(\gamma)\mathbf{R_y}(\beta+\Delta\beta)$. If we stick to the Euler-angles parameterization, we have to live with this discrepancy to our intuition. Fortunately, this problem only shows at large rotation angles. Since the freedom of camera motion is usually very limited (mostly horizontal pan, sometimes vertical pan, but usually no tilting), this problem can be reduced in practice. Using the pre-knowledge of typical camera motion, we select an appropriate parameterization that mostly behaves according to our expectations. Since rotations can be influenced by subsequent transformations, the rotation that has the largest dynamic range should be the last in the sequence. Using this order, we get

$$\mathbf{R}(\alpha,\beta,\gamma) = \mathbf{R_y}(\beta)\mathbf{R_x}(\alpha)\mathbf{R_z}(\gamma) \tag{2.15}$$

as a good choice for the rotation sequence.

Gimbal lock

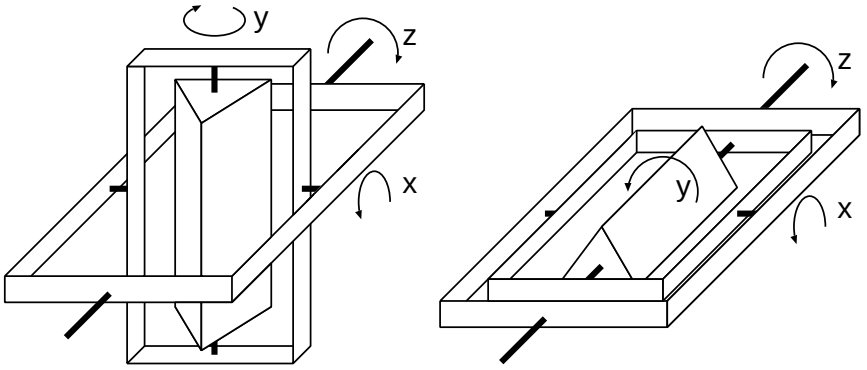
An especially annoying effect of the Euler-angle parameterization is the *gimbal lock* phenomenon. Gimbal lock is the situation that the Euler angles are chosen such that two of the rotation axes coincide. In our parameterization of Eq. (2.15), this is the case when $\alpha = \pm\pi/2$. In this situation, the y and the z axes coincide and the rotation angles β and γ both induce similar motion (Fig. 2.10). Thus, the degrees of freedom is reduced from three to only two in the gimbal lock position. To prove this, we insert $\alpha = \pi/2$ into Eq. (2.15) and get

$$\begin{aligned}\mathbf{R_y}(\beta)\mathbf{R_x}(\pi/2)\mathbf{R_z}(\gamma) &= \begin{bmatrix} c_\beta & 0 & s_\beta \\ 0 & 1 & 0 \\ -s_\beta & 0 & c_\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} c_\gamma & -s_\gamma & 0 \\ s_\gamma & c_\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} c_\beta c_\gamma + s_\beta s_\gamma & s_\beta c_\gamma - c_\beta s_\gamma & 0 \\ 0 & 0 & -1 \\ c_\beta s_\gamma - s_\beta c_\gamma & s_\beta s_\gamma + c_\beta c_\gamma & 0 \end{bmatrix} = \begin{bmatrix} c_{(\beta-\gamma)} & s_{(\beta-\gamma)} & 0 \\ 0 & 0 & -1 \\ -s_{(\beta-\gamma)} & c_{(\beta-\gamma)} & 0 \end{bmatrix}.\end{aligned} \tag{2.16}$$

Hence, β and γ induce the same rotation (in opposite directions).

Rotation matrices

Often, we do not require the parameterization into Euler angles but we can work with only the out-multiplied rotation matrix. In these cases, we can



(a) Physical illustration of an Euler rotation sequence.

(b) Gimbal lock position.

Figure 2.10: *The object is mounted into a gimbal that corresponds to our Euler rotation sequence. In the position (b), the inner frame is positioned with $\alpha = \pm\pi/2$, so that the z and the y axes coincide.*

just use general 3×3 rotation matrices

$$\mathbf{R} = \begin{bmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{bmatrix}, \tag{2.17}$$

without knowing its factorization into angle parameters. However, since we know that $\mathbf{R}$ is a rotation matrix, we can make use of some properties of rotation matrices. We have seen previously that for the inverse of a elementary rotation matrix, it holds that $\mathbf{R_i}^{-1} = \mathbf{R_i}^{\top}$. Now, let $\mathbf{R}$ be composed of a sequence of rotations $\mathbf{R} = \mathbf{R}_1 \cdots \mathbf{R}_n$. It follows that

$$\mathbf{R}^{\top} = (\mathbf{R}_1 \cdots \mathbf{R}_n)^{\top} = \mathbf{R}_n^{\top} \cdots \mathbf{R}_1^{\top} = \mathbf{R}_n^{-1} \cdots \mathbf{R}_1^{-1} \tag{2.18}$$

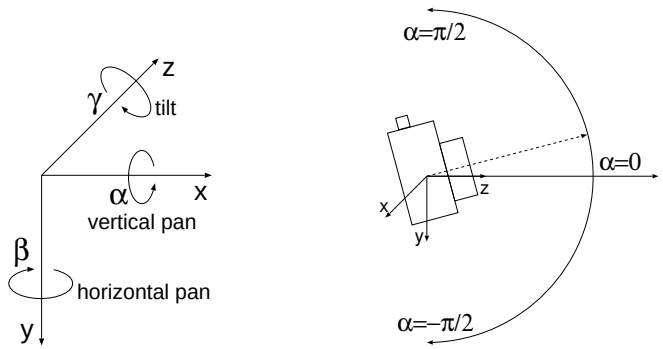
and

$$\mathbf{R} \cdot \mathbf{R}^{\top} = \mathbf{R}_1 \cdots \mathbf{R}_n \cdot \mathbf{R}_n^{-1} \cdots \mathbf{R}_1^{-1} = \mathbf{I}. \tag{2.19}$$

Consequently, the transpose of the composed rotation matrix must be equal to its inverse $\mathbf{R}^{\top} = \mathbf{R}^{-1}$. In other words, $\mathbf{R}\mathbf{R}^{\top} = \mathbf{I}$, which shows that $\mathbf{R}$ must be an orthogonal matrix.

Obtaining Euler angles from a rotation matrix

Sometimes, we have an arbitrary rotation matrix and we want to know the Euler angles for a factorization into a given sequence of elementary



(a) Rotation angles around coordinate system axes.

(b) Restricting α such that top of image always points upwards.

Figure 2.11: *Naming conventions used on the right-handed camera coordinate system.*

rotations. Assume that we fix the rotation sequence to that of Eq. (2.15), we get the out-multiplied matrix

$$\mathbf{R} = \begin{bmatrix} c_\beta c_\gamma + s_\alpha s_\beta s_\gamma & -c_\beta s_\gamma + s_\alpha s_\beta c_\gamma & c_\alpha s_\beta \\ c_\alpha s_\gamma & c_\alpha c_\gamma & -s_\alpha \\ s_\alpha c_\beta s_\gamma - s_\beta c_\gamma & s_\alpha c_\beta c_\gamma + s_\beta s_\gamma & c_\alpha c_\beta \end{bmatrix}. \tag{2.20}$$

Because the trigonometric functions are cyclic, we require additional restrictions to find a unique solution. A possible restriction is to assume that $-\pi/2 < \alpha < \pi/2$. This is a sensible assumption for a normal camera, since it limits the up-down rotation to $\pm 90°$. Now, α can be obtained directly from $r_{12} = -\sin\alpha$. Further, we see from the last column of $\mathbf{R}$ that

$$\frac{r_{02}}{r_{22}} = \frac{c_\alpha \cdot s_\beta}{c_\alpha \cdot c_\beta}\text{, and consequently,} \qquad \tan\beta = \frac{s_\beta}{c_\beta} = \frac{r_{02}}{r_{22}}. \tag{2.21}$$

The correct quadrant of β can be determined from the signs of nominator and denominator. Thus, it is important that we keep the signs at their respective terms. Since we assumed that $|\alpha| < \pi/2$, we know that $c_\alpha > 0$. This makes it easier to obtain the correct quadrant for β in Eq. (2.21). The angle γ can be obtained similarly from r_{10} and r_{11}.

Quaternion rotation

Because of the gimbal lock problems of Euler rotation sequences, a different representation of rotations has become popular in the computer graphics

community. This representation uses quaternions, which are a generalization of complex numbers to four elements. A quaternion is a number $q = a+bi+cj+dk$ together with the rules $i^2 = j^2 = k^2 = -1$, $ij = -ji = k$, $jk = -kj = i$, and $ki = -ik = j$. Furthermore, the quaternion conjugate is defined as $\bar{q} = a - bi - cj - kj$.

Alternatively to using the imaginary units, a quaternion can also be written as a vector of its four components $\mathbf{q} = (a, b, c, d)$. It can be shown that a quaternion with $||\mathbf{q}|| = 1$ represents a rotation in 3-D space using the transformation $\mathbf{p}' = \mathbf{q}\mathbf{p}\bar{\mathbf{q}}$, where $\mathbf{p} = (0, x, y, z)$ is the point coordinate.

It is especially easy to construct a quaternion that represents a rotation of θ around the axis vector $\mathbf{n}$. This rotation can be written as the quaternion $\mathbf{q} = (\cos(\theta/2), \sin(\theta/2) \cdot \mathbf{n}^\top)$. On the other hand, a quaternion rotation can also be written as the 3×3 rotation matrix

$$\mathbf{R} = \begin{bmatrix} 1 - 2q_y^2 - 2q_z^2 & 2q_xq_y - 2q_wq_z & 2q_xq_z + 2q_wq_y \\ 2q_xq_y + 2q_wq_z & 1 - 2q_x^2 - 2q_z^2 & 2q_yq_z - 2q_wq_x \\ 2q_xq_z - 2q_wq_y & 2q_yq_z + 2q_wq_x & 1 - 2q_x^2 - 2q_y^2 \end{bmatrix}. \tag{2.22}$$

The advantage of the quaternion representation compared to the Euler rotation sequences is that there is no singularity as with the gimbal lock position. Furthermore, they are faster to compute since no transcendental functions are involved. We will use the quaternion representation in Chapter 12 to describe camera rotation.

2.4.3 Perspective projection

In this section, we introduce the projection of points from 3-D space onto a 2-D image plane. This projection operation is based on the idealized model of a *pinhole camera*, which is a very good approximation to most real cameras. We derive the projection equations and formulate them, using the previously explained homogeneous coordinates framework. Finally, we introduce the *affine camera* model, which is an often used approximation that leads to the special case of affine motion in the image plane.

Perspective pinhole camera

The setup for the perspective camera model is illustrated in Figure 2.12. Object points are projected along the ray from the camera center to the object point. The intersection of the ray with the image plane defines the point's position in the image. In a real camera, the image plane is actually behind the camera center, and the image is projected onto it up-side down, but for simplicity, we assume that the image plane is in front of the camera. This is equivalent, but it relieves us from considering many minus signs.

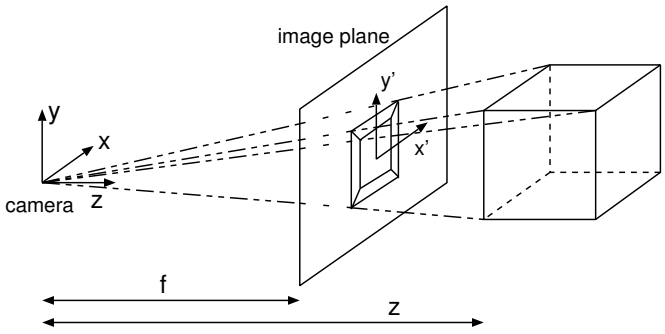


Figure 2.12: *Model of an ideal pinhole camera.*

Let us denote the distance between the camera center and the image plane by f (*focal length*) and let the object points have coordinates $(x, y, z)^\top$. Since the camera is located at the coordinate system origin, and since it is looking along the positive z-axis, the projection of the point onto the image plane is given by $(x', y')^\top = (fx/z, fy/z)^\top$. This projection can also be written using a matrix multiplication in homogeneous coordinates as

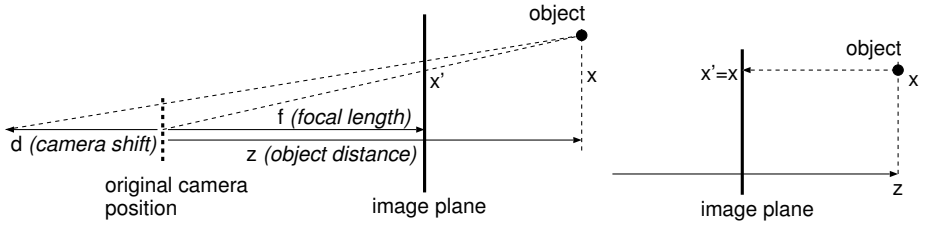
$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}. \tag{2.23}$$

We call this matrix the matrix of *intrinsic camera parameters*. In our ideal pinhole camera model, the matrix only consists of the focal length, which can be considered as an internal parameter of the camera. However, in the following sections, we extend the intrinsic matrix with additional parameters.

Affine camera

When we use the perspective camera model together with inhomogeneous coordinates, we obtain non-linear equations for the transformation. In several situations, this non-linearity makes the computation more complex. A popular approach is to use a linear estimation to the perspective camera which assumes that the camera is placed infinitely far from the object. As is shown in the following, the simplified camera model results in affine motion in the image plane.

In Figure 2.13(a), the normal perspective camera setup is depicted. Now assume that the camera is shifted away from the image plane by an additional distance d. This shift is compensated by increasing the focal length



(a) Moving the camera to a larger distance. (b) Parallel projection.

Figure 2.13: *From perspective projection the orthographic projection.*

to $f+d$ to keep the distance between object and image plane constant. We can formally write this construction by inserting a matrix for the camera movement before the projection is made:

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{bmatrix} f+d & 0 & 0 & 0 \\ 0 & f+d & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}. \tag{2.24}$$

Multiplying out the two matrices and increasing the camera distance d up to infinity, we get

$$\begin{bmatrix} f+d & 0 & 0 & 0 \\ 0 & f+d & 0 & 0 \\ 0 & 0 & 1 & d \end{bmatrix} \sim \begin{bmatrix} \frac{f}{d}+1 & 0 & 0 & 0 \\ 0 & \frac{f}{d}+1 & 0 & 0 \\ 0 & 0 & \frac{1}{d} & 1 \end{bmatrix} \xrightarrow{d\to\infty} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.25}$$

where the first similarity holds because of the scaling invariance of homogeneous coordinates. The resulting matrix is the camera projection matrix for affine cameras. Clearly, it is independent of a focal length and the scene depth z (the column that multiplies z only contains zeroes). Written in inhomogeneous coordinates, this is simply

$$x' = x/w \quad y' = y/w, \tag{2.26}$$

which is just the conversion of homogeneous coordinates into inhomogeneous coordinates (see Eq. (2.1)). Consequently, the affine camera model describes a parallel projection of the 3-D coordinates onto the image plane (Fig. 2.13(b)). This type of projection is also called an *orthographic* projection.

Let us now examine the complete imaging process with an affine camera. We start with an arbitrary affine object motion in 3-D, followed by an

orthographic projection:

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{orthographic projection}} \underbrace{\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{affine motion in 3-D}} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}. \tag{2.27}$$

Furthermore, we assume that the observed object is planar, which means that, e.g., its z-coordinate is dependent on the others, and we can write $z = p_x x + p_y y + p_w$. Incorporating this constraint into Eq. (2.27), we get

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ p_x & p_y & p_w \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ w \end{pmatrix}. \tag{2.28}$$

Multiplying the three matrices, the total mapping onto the image plane computes as

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{bmatrix} a_{00} + p_x a_{02} & a_{01} + p_y a_{02} & p_w a_{02} + a_{03} \\ a_{10} + p_x a_{12} & a_{11} + p_y a_{12} & p_w a_{12} + a_{13} \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ w \end{pmatrix}. \tag{2.29}$$

However, since the last row is $(0, 0, 1)$, this defines an affine transformation. Consequently, planar object motion in 3-D is observed on the image plane as affine motion if an orthographic projection is applied.

2.5 Image acquisition

In this section, we discuss the image formation process in more detail. We start by looking at the pinhole camera model again and extend its description to cover more general cases. Furthermore, we include a brief discussion of lens-distortion artifacts that can become relevant in a practical application. Finally, we analyse which camera setups allow to use the projective motion model to describe global motion.

2.5.1 Intrinsic camera parameters

We start with a more detailed discussion of the ideal pinhole camera, which we will successively extend to include more general camera configurations. When we specify the camera position by saying, that the camera should be

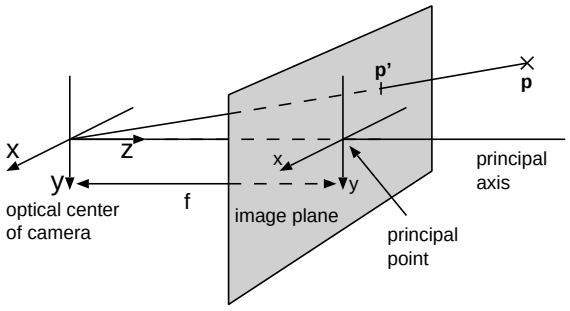


Figure 2.14: *Image formation in the case of an ideal pinhole camera.*

located at the origin of the coordinate system, this means more specifically that the pinhole of the camera is at the origin. The pinhole position, which is the center of the projection rays, is denoted as the *optical center* of the camera.

Let us assume that a local coordinate system is attached to the camera. We define this coordinate system to be right-handed, and the camera to be viewing in the direction of the positive z-axis (Figure 2.14). The x-axis increases to the right and the y-axis, consequently, to the bottom. The image plane is perpendicular to the z-axis, at a distance of the focal length f. The viewing direction, which is in our case simply the z-axis, is called the *principal axis*, and the position where the principal axis intersects the image plane is denoted as the *principal point*.

Projection of 3-D points onto the image plane

Consider the point $\mathbf{p} = (x, y, z)^\top$ in $\mathbb{E}^3$. To determine its position on the image plane using the pinhole camera model, we construct a line through the optical center and the point $\mathbf{p}$. The intersection of this line with the image plane defines its position $\mathbf{p}' = (x', y')$ on the image plane. If the origin of the local coordinate system of the image plane is also at the principal point and the coordinate axes are parallel to that of the camera, the projection can be calculated by

$$x' = \frac{x \cdot f}{z} \quad ; \quad y' = \frac{y \cdot f}{z}. \tag{2.30}$$

Using homogeneous coordinates, the projection can also be written as the matrix multiplication

$$\mathbf{p}' = \begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}. \tag{2.31}$$

Note that in the derivation of the equation, we have silently assumed a number of idealized properties: we assumed that pixels in the image plane are square, and we assumed that there is no skew in the image sensor-array.

Non-ideal image sensors

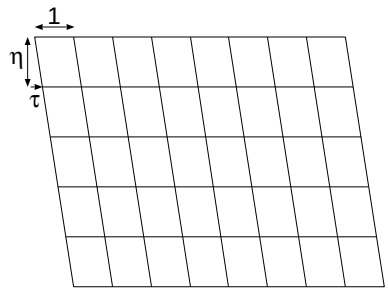
Usually, we can assume square pixels in the recording camera. However, if they are not square, the intrinsic camera-parameter matrix can be extended by a parameter η, denoting a vertical scaling factor. Moreover, the sampling grid may be skewed (see Fig. 2.15(a)). This can also be represented by an additional parameter τ. If we include these parameters, we get the more general camera projection matrix

$$\begin{bmatrix} f & \tau & 0 & 0 \\ 0 & \eta f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.32}$$

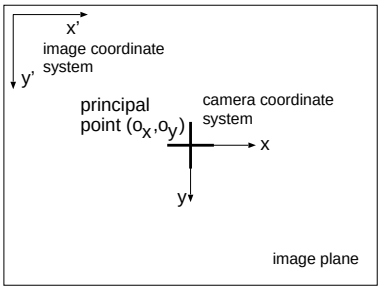
However, in most cases, it can simply be assumed that $\eta = 1$ and $\tau = 0$. In fact, in Chapter 12, we use these assumptions as constraints to recover the focal length from estimated camera motion.

Changing the image coordinate system

The usual convention of storing images in memory is to place the origin of the image coordinate system at the top left of the image, with the x-axis extending to the right, and the y-axis pointing downwards. Previously, we have assumed that the origin coincides with the principal point. Now, we relieve this constraint and assume that the principal point is located at (o_x, o_y) in image coordinates. Moreover, we might want to flip some of the coordinate system axes if their definition in the image is different. For the purpose of demonstration, we assume that we also want to flip the direction of the y-axis. By concatenating the projection matrix with the flip of the y-axis and the shift of the coordinate system, we get the generalized camera



(a) Non-ideal sampling grid in the camera sensor.



(b) Image coordinate system does not coincide with camera coordinate system.

Figure 2.15: *(a) In a non-ideal camera, the sampling-grid may have non-square pixels and the sampling-grid may be skewed. (b) Usually, the image coordinate system is assumed to have its origin at the top-left corner instead of the principal point.*

projection matrix

$$\underbrace{\begin{bmatrix}1 & 0 & o_x\\ 0 & 1 & o_y\\ 0 & 0 & 1\end{bmatrix}}_{\text{translate}} \cdot \underbrace{\begin{bmatrix}1 & 0 & 0\\ 0 & -1 & 0\\ 0 & 0 & 1\end{bmatrix}}_{\text{flip } y\text{-axis}} \cdot \begin{bmatrix}f & \tau & 0 & 0\\ 0 & \eta f & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix} = \begin{bmatrix}f & \tau & o_x & 0\\ 0 & -\eta f & o_y & 0\\ 0 & 0 & 1 & 0\end{bmatrix}. \quad (2.33)$$

The right-hand side of this equation is the most general form of the intrinsic camera-parameters matrix. The last column is often omitted so that only a 3×3 matrix is considered. Note that even the general intrinsic matrix is upper triangular, which is sometimes exploited in camera calibration techniques where we try to estimate the intrinsic matrix.

Radial lens distortion

Real cameras consist of a number of lenses instead of a simple pinhole. These lenses are usually not ideal and show some non-linear geometric distortion, which can become significant for applications requiring a high accuracy.

There is no simple general model for lens distortion, but for most practical purposes, it can be approximated by a simple radial lens distortion model. To define a distortion model, let (x', y') be the coordinates of a pixel including the lens distortion (the projected position on the sensor array), and let $(\hat{x}, \hat{y})$ be the coordinates of their ideal, undistorted coordinate (Figure 2.16). A popular model for radial lens distortion can then be described

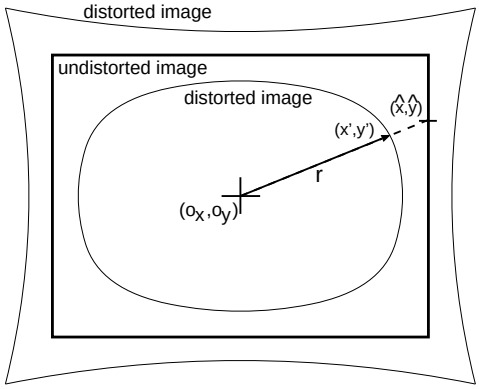


Figure 2.16: *Radial lens distortian. Opposed to the ideal pinhole model, real cameras are subject to lens distortion.*

by

$$\begin{aligned} \hat{x} &= o_x + (x' - o_x) \cdot (1 + D) \\ \hat{y} &= o_y + (y' - o_y) \cdot (1 + D) \end{aligned} \tag{2.34}$$

with D being the radial correction term

$$D = (\kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6 + \cdots), \tag{2.35}$$

where

$$r = \sqrt{(x' - o_x)^2 + (y' - o_y)^2} \tag{2.36}$$

is the distorted point's distance from the principal point. In most cases, it is sufficient to consider only the two low-order coefficients κ_1, κ_2, and assume that the higher-order coefficients are zero.

For many cameras, the effect of radial distortion can be neglected. However, for extreme wide-angle lenses, the effect can become significant and the radial distortion should be compensated. In this thesis, we assume that radial distortion is not present, or that it has been compensated previously.

2.5.2 Extrinsic camera parameters

While the intrinsic camera parameters describe properties of the camera like its focal length and the image geometry, the external camera parameters describe the camera placement and orientation in the 3-D world.

Assume that the camera is located at a position $\mathbf{t}$ and rotated according to a rotation matrix $\mathbf{R}$. To determine the image position of a 3-D point

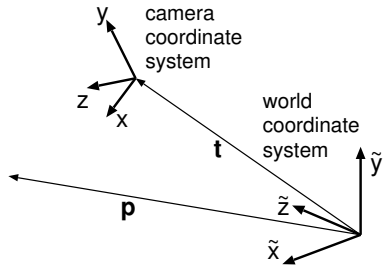


Figure 2.17: *General setting of a camera located at* $\mathbf{c}$*, observing a point* $\mathbf{p_w}$ *in 3D space.*

$\mathbf{p}$ in the observed image, we first have to bring the camera to the origin and rotate it so that its local camera coordinate system is aligned with the world coordinate system. Written as a sequence of transformations, we obtain

$$\begin{aligned}\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} &= \begin{bmatrix} f & 0 & o_x & 0 \\ 0 & f & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \underbrace{\begin{bmatrix} \mathbf{R} & \mathbf{0}_3 \\ \mathbf{0}_3^\top & 1 \end{bmatrix}}_{\text{rotation}} \underbrace{\begin{bmatrix} \mathbf{1}_{3\times 3} & -\mathbf{t} \\ \mathbf{0}_3^\top & 1 \end{bmatrix}}_{\text{translation}} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} \\ &= \underbrace{\begin{bmatrix} f & 0 & o_x & 0 \\ 0 & f & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{intrinsic parameters}} \underbrace{\begin{bmatrix} \mathbf{R} & -\mathbf{Rt} \\ \mathbf{0}_3^\top & 1 \end{bmatrix}}_{\text{extrinsic parameters}} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}. \end{aligned} \tag{2.37}$$

Since the last column of the intrinsic parameters matrix is all zero, it is common practice to remove the last column from the intrinsic parameters matrix and as a consequence thereof, also the last row of the extrinsic parameters matrix. Note that this is a slight misuse of notation, since homogeneous coordinates are mixed with an inhomogeneous notation. However, it saves us from writing matrices with constant zero rows or columns. The reduced matrices can then be defined as

$$\mathbf{K} = \begin{bmatrix} f & 0 & o_x \\ 0 & f & o_y \\ 0 & 0 & 1 \end{bmatrix} \quad ; \quad \mathbf{E} = \begin{bmatrix} \mathbf{R} \,|\, -\mathbf{Rt} \end{bmatrix}, \tag{2.38}$$

where we call $\mathbf{K}$ the 3×3 intrinsic camera parameters matrix and $\mathbf{E}$ the 3×4 extrinsic camera parameters matrix. Note that the matrix $\mathbf{E}$ transforms from $\mathbb{P}^3$ to $\mathbb{E}^3$ and $\mathbf{K}$ further from $\mathbb{E}^3$ to $\mathbb{P}^2$. The annotated arrows

in the following equation show the type of the vector after each matrix multiplication:

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \underset{\mathbb{P}^2}{\uparrow} \begin{bmatrix} f & 0 & o_x \\ 0 & f & o_y \\ 0 & 0 & 1 \end{bmatrix} \underset{\mathbb{E}^3}{\uparrow} \begin{bmatrix} \mathbf{R} \,|\, -\mathbf{R}\mathbf{t} \end{bmatrix} \underset{\mathbb{P}^3}{\uparrow} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}. \tag{2.39}$$

Inverse transformation

When a 3-D scene is projected onto the 2-D image plane, it is obvious that information about the depth is lost. In the perspective projection, every point along the projection ray is mapped onto the same image point. Therefore, it is impossible to say which point on the ray is viewed in the image. This becomes apparent if we try to invert the above transformation pipeline. Using the intrinsic parameters matrix in the inverse direction to map 2-D points $(x', y', 1) \in \mathbb{P}^2$ into 3-D space $(x, y, z) \in \mathbb{E}^3$ again, results in

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \mathbf{K}^{-1} \begin{pmatrix} \lambda \cdot x' \\ \lambda \cdot y' \\ \lambda \cdot 1 \end{pmatrix}. \tag{2.40}$$

The factor λ on the right-hand side accounts for the fact that the same point in homogeneous coordinates can also be specified by an arbitrarily scaled version of that vector. After applying the inverse $\mathbf{K}^{-1}$, each point on the image plane therefore defines a one-dimensional subspace in $\mathbb{E}^3$. Clearly, this subspace is the projection ray with the free parameter $\lambda \neq 0$.

To invert the extrinsic parameters matrix, we have to use the full 4×4 matrix, since the abbreviated 3×4 matrix is not square. Since the result of the multiplication with $\mathbf{K}^{-1}$ is in $\mathbb{E}^3$, we also have to augment that vector with the homogenizing constant 1. When we compute the inverse of the external parameters matrix, we get

$$\begin{bmatrix} \mathbf{R} & -\mathbf{R}\mathbf{t} \\ \mathbf{0}_3^\top & 1 \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{R}^{-1} & \mathbf{t} \\ \mathbf{0}_3^\top & 1 \end{bmatrix}. \tag{2.41}$$

It can be seen that the last matrix row of the inverse is still $(0, 0, 0, 1)$. This is clear since the external parameters matrix is affine, and hence, its inverse must also be affine.

The complete imaging process

If we put all the described image transforms together, we obtain a complete formulation for the entire image formation process. This image formation

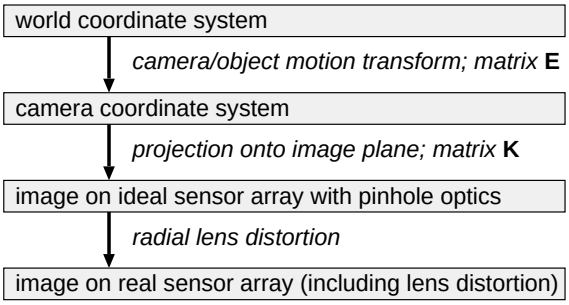


Figure 2.18: *Steps for transforming a 3-D point position to its coordinate in a camera image, including distortions.*

is divided into several steps and can be illustrated as a pipeline process, as shown in Figure 2.18. When we follow the path of a point from the 3-D real world to the image sensor, we start by transforming the point's position in the world coordinate system into the camera coordinate system using the extrinsic parameters matrix **E**. The succeeding intrinsic parameters matrix further maps the point to the position on the image sensor-array, but still neglecting lens distortion. Finally, the distortions that are introduced by non-ideal lens optics are modeled with a radial lens distortion.

2.5.3 Camera motion in a static environment

Suppose that two pictures of a static 3-D scene are taken from different positions or at different angles. We would like to explore in which cases it is possible to describe the transform between the two images as a projective transform **H**.

Since we have seen in Section 2.3.1 that a projective transformation is equivalent to a plane-to-plane mapping, it is clear that the transform can model arbitrary camera motion, as long as the observed object is planar. However, the restriction that the complete observed environment is planar is too strict in practice, and we would like to know if there are more situations which can still be described using the projective motion model.

Obviously, it is also clear that we cannot describe the general case with a simple perspective transformation. Take for example Figure 2.19 which shows a scene with four objects, viewed from two different positions. It can be observed that the objects are projected onto the camera images in a different ordering, depending on the camera position.

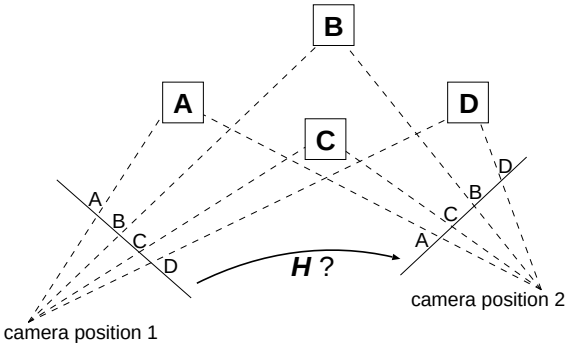


Figure 2.19: *Top view of a 3-D scene with three object A, B, C. Seen from two different angles, the objects appear on the camera images in a different order.*

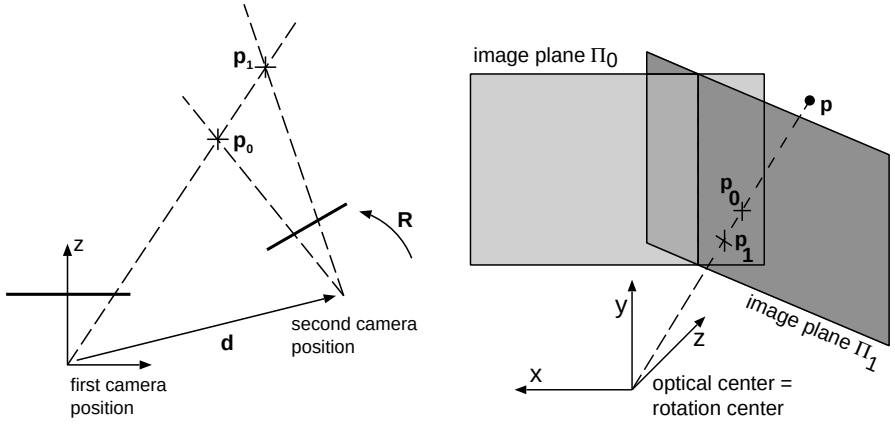
Parallax effect

From everyday experience, we know the effect that if we move a camera at a constant speed, near objects seem to move faster than objects at a larger distance. This visual effect is commonly known as the *Parallax effect*. The Parallax effect has consequences for 3-D vision, since an arrangement of objects in space can look very dissimilar from different camera locations. In general, it is not possible to describe the transform between these camera images by a projective transform.

To illustrate this, assume that we observe two points with inhomogeneous coordinates $\mathbf{p_0}$ and $\mathbf{p_1}$ at different distances from the camera, but which are projected onto the same point in the image plane (Fig. 2.20(a)). For simplicity, we assume w.l.o.g. that the camera is first located at the origin of the world coordinate system and that it is looking along the positive z-axis. Since both points are projected onto the same image position, it holds that $\mathbf{Kp_0} \sim \mathbf{Kp_1}$, and consequently $\mathbf{p_0} \sim \mathbf{p_1}$. This means that there exists a λ such that $\mathbf{p_0} = \lambda \mathbf{p_1}$.

Now, we translate the camera by a distance $\mathbf{d}$ and rotate it according to a rotation matrix $\mathbf{R}$. The question is whether the two points $\mathbf{p_0}, \mathbf{p_1}$ are also mapped onto the same image point in the new camera view. In other words, we want to know, if

$$\underbrace{\mathbf{K}\begin{bmatrix}\mathbf{R} & -\mathbf{Rd}\\ \mathbf{0} & 1\end{bmatrix}\begin{pmatrix}\mathbf{p_0}\\ 1\end{pmatrix}}_{\text{new image position of } \mathbf{p_0}} \sim^{?} \underbrace{\mathbf{K}\begin{bmatrix}\mathbf{R} & -\mathbf{Rd}\\ \mathbf{0} & 1\end{bmatrix}\begin{pmatrix}\lambda\mathbf{p_0}\\ 1\end{pmatrix}}_{\text{new image position of } \mathbf{p_1}} \tag{2.42}$$



(a) Observing two points from different positions.

(b) Rotation-only camera.

Figure 2.20: *What kind of camera motion is allowed in arbitrary 3-D scenes? (a) Object points at different depth and moving camera. Points may fall onto the same position in one image and onto different positions in another image. Hence, there cannot be a bijective transform between both. (b) Rotational camera motion. Two points on the same projection ray stay on one ray independent of a camera rotation. Hence, a transform between images is possible even with arbitrary object depths.*

holds. After multiplying with $\mathbf{K^{-1}}$ from the left, we get

$$\mathbf{Rp_0} - \mathbf{Rd} \sim^{?} \lambda\mathbf{Rp_0} - \mathbf{Rd}. \tag{2.43}$$

It is easy to see that this is only true if $\mathbf{d} = \mathbf{0}$, i.e., there is no translatorial camera motion, or $\lambda = 1$. The former case holds if camera motion is restricted to rotation around the optical center (Fig. 2.20(b)). Since this case is of high practical importance, it will be described in more detail in the next subsection. The second case means that the two points $\mathbf{p_0}, \mathbf{p_1}$ are actually the same point. However, this means that on each projection ray, there may only be one object point. Otherwise, the two points could coincide in one image, but have a different position in another image. This second case applies if the object is planar.

Rotating camera

In the special case of a purely rotational camera, let us first assume that the optical center of the camera is located at the coordinate system origin, and the rotation is performed around the coordinate system origin. In this case,

the camera translation is $\mathbf{t} = \mathbf{0}$, and the complete viewing transformation can be written as

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \mathbf{K}\,\mathbf{R} \begin{pmatrix} x \\ y \\ z \end{pmatrix}. \tag{2.44}$$

The rotation matrix $\mathbf{R}$ assumes rotation of the world around a fixed camera, which is equivalent to rotating the camera in a fixed world, if the matrix is inversed (negative angles in opposite order).

One property of rotational camera motion is of special importance and should therefore be emphasized here. For the transformation between images taken by a rotating camera, the object depth is of no importance, since no parallax effect occurs. This fact can be justified easily with Eq. (2.44). Since the camera is assumed to be at the origin, all pixels on the projection ray through a point can be obtained by scaling the point coordinate $(x, y, z)^\top$ on the right-hand side. Moreover, the transform is linear which means that the same scaling will be effective on the left-hand side. But since homogeneous coordinates are scaling invariant, this is actually the same point in the image. This holds for any choice of $\mathbf{R}$ such that points projected onto the same image position will always coincide, independent of any camera rotation.

2.5.4 Inter-image transformation

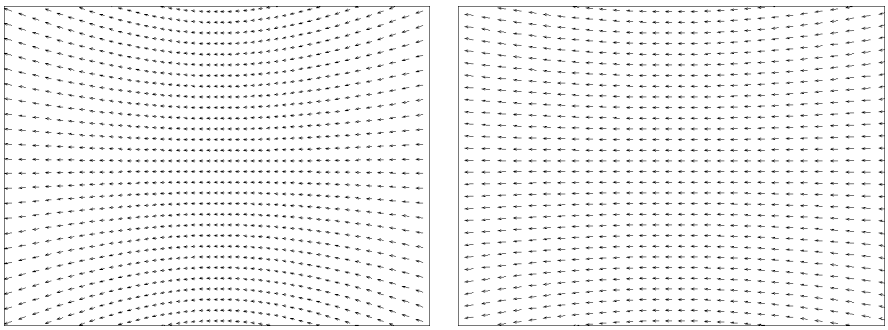
Up to now, we only considered transformations that map points from the 3-D world onto the 2-D image plane. However, in many cases, we want to describe the motion between successive images.

Inter-image transforms with a rotating camera

Let us discuss again the situation of a rotational camera. In general, the scene around the camera has varying depths, but we have seen in the last subsection that the depth of objects does not play any role in the image formation. Hence, we can assume that the environment is actually located in the camera-image plane.

Now consider that we take two images Π_0 and Π_1 at different rotation angles $\mathbf{R}_0$ and $\mathbf{R}_1$ (as depicted in Figure 2.20(b)). According to Equation (2.44), we can compute the 3-D ray on which an image point from Π_0 lies, and we can also compute the intersection of this ray with Π_1 by inverting the equation. Combining these two parts gives us the transformation between image Π_0 and Π_1 as

$$\mathbf{p_1} = \underbrace{\mathbf{K}\,\mathbf{R_1}\,\mathbf{R_0^{-1}}\,\mathbf{K^{-1}}}_{\mathbf{H}}\,\mathbf{p_0}. \tag{2.45}$$



(a) Short focal length ($f = 400$). (b) Long focal length ($f = 700$).

Figure 2.21: *Motion field of a pure horizontal camera pan for an image width of* 800.

Clearly, the four transformation matrices can be combined into a single 3×3 inter-image transformation matrix $\mathbf{H}$.

Motion field for panning camera motion

A common misunderstanding about panning camera motion is that it is often assumed that the motion field resulting from a rotating camera equals a pure translation. However, as can be seen in Figure 2.21, this is not true. The figure depicts the motion field that is observed when a camera performs a rotation around the vertical axis. The motion field depends on the focal length of the camera. For larger focal lengths, the motion trajectories get straighter, and for the limit case of infinite focal length (affine camera), the motion field becomes translatorial. For practical cases, f is in the order of the image width and, consequently, the motion cannot be approximated with a simple translation.

Inter-image transforms with a planar object

If the observed object is planar, it is also possible to find an inter-image transform. In this case, this is obvious, since the mapping from the object plane onto each of the image planes Π_0, Π_1 is an invertible plane-to-plane mapping (see Figure 2.22). Hence, we get the inter-frame transform by projecting the Π_0 plane onto the background plane, and then projecting from the background plane onto plane Π_1. Following the derivation in Section 2.5.2, let the transform from the background plane to image plane i be denoted as $\mathbf{M}_i = \mathbf{K_i}\mathbf{E_i}$. Since this product is a 3×4 matrix, it cannot be inverted. However, we are free to choose the world coordinate system

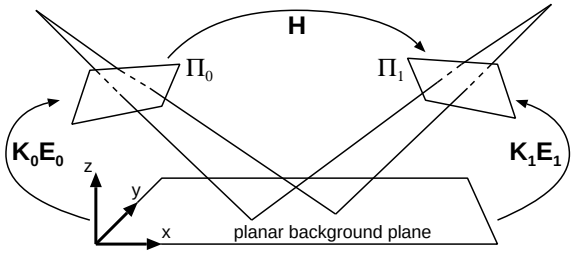


Figure 2.22: *A planar background is observed by a moving camera.*

and can thus define that the background plane coincides with the $z = 0$ plane. In this case, the projection from the background plane to image coordinates becomes

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \end{bmatrix} \begin{pmatrix} x \\ y \\ z = 0 \\ w \end{pmatrix}, \tag{2.46}$$

and the third column of the matrix $\mathbf{M} = \{m_{ik}\}$ can be omitted. Let us denote the matrix $\mathbf{M}$ without the third column as $\check{\mathbf{M}}$. Since this matrix $\check{\mathbf{M}}$ is now a 3×3 matrix, it can be inverted and we can combine the two plane-to-plane mappings to obtain the inter-frame transform as $\mathbf{H} = \check{\mathbf{M}}_\mathbf{1}\check{\mathbf{M}}_\mathbf{0}^{-1}$.

Outlook: Camera calibration from video sequences.

In this section, we have shown that the projective transform between two images can be modeled as a sequence of elementary physically meaningful operations. Usually, a camera-motion estimator only determines the final set of parameters that are not directly related to the elementary operations. In Chapter 12, we develop an algorithm to factorize the projective transformation parameters back into the elementary operations. This makes it possible to describe camera motion in physically meaningful terms, like rotation angles or the current focal length of the camera zoom lenses.

Equation	Description	Section
$\mathbf{l} = \mathbf{p_1} \times \mathbf{p_2}$	line through two points	2.2.2
$\mathbf{p} = \mathbf{l_1} \times \mathbf{l_2}$	intersection point of two lines	2.2.2
$\mathbf{p}' = \mathbf{H}\mathbf{p}$	projective transformation of points	2.3.1
$\mathbf{l}' = \mathbf{H}^{-\top}\mathbf{l}$	projective transformation of lines	2.3.1
$\mathbf{K} = \begin{bmatrix} f & \tau & o_x \\ 0 & \eta f & o_y \\ 0 & 0 & 1 \end{bmatrix}$	intrinsic camera parameters	2.5.1
$\mathbf{E} = \begin{bmatrix} \mathbf{R} \,\vert\, -\mathbf{R}\mathbf{t} \end{bmatrix}$	extrinsic camera parameters	2.5.2
$\mathbf{H} = \mathbf{K_i}\mathbf{R}\mathbf{K_j^{-1}}$	rotational camera motion	2.5.4

Table 2.3: *Summary of the most important equations.*

2.6 Summary and notational conventions

This chapter has introduced homogeneous coordinates to describe point positions. In two-dimensional space, a point is specified with a column vector $\mathbf{p} = (x, y, w)^\top$, where the Euclidean coordinates can be recovered as $(x/w, y/w)^\top$. To facilitate the formulation, we also use homogeneous coordinates as arguments to two-dimensional functions in the following chapters, assuming implicit conversion to Euclidean coordinates. This means that we use the simplified notation $I(\mathbf{p})$ instead of $I(x/w, y/w)$.

We have also seen that the projection of rigid 3-D motion of planar patches onto 2-D images can be described by a simple matrix multiplication. Considering the plane-to-plane mapping which transforms $\mathbf{p}$ onto $\mathbf{p}'$, it can be formulated as multiplication with a 3×3 matrix $\mathbf{H}$ as $\mathbf{p}' = \mathbf{H}\mathbf{p}$.

An important case is the transformation of Euclidean 3-D coordinates onto a planar image as it is seen by a pinhole camera at the origin of the coordinate system. This transformation is usually decomposed into two parts, namely the intrinsic camera parameters matrix $\mathbf{K}$, comprising internal camera parameters like the focal length and the principal point o_x, o_y, and the extrinsic camera parameters matrix $\mathbf{E}$, describing camera rotation and translation (see Eq. (2.38)). For the special case of rotational camera motion, the intrinsic and extrinsic matrices can be combined to a transformation matrix $\mathbf{H}$ denoting the motion between a pair of images.

The most important equations are summarized in Table 2.3 with references to the sections in which they were introduced.