Chapter #

Camera Models

Pinhole Model, Perspective Projection and Camera Parameters

Zhigang Zhu
Department of Computer Science, The City College of New York, New York, NY 10031

1. INTRODUCTION

In this part we will focus on the fundamentals of three-dimensional (3D, or 3-D) computer vision. The motivation of 3D computer vision is to enable computers to see the 3D world as humans do. To summarize the subject in a few words, we can define 3D Computer Vision as the study of turning 2D images back into 3D structures. Its applications include robotics, autonomous vehicles, virtual reality, surveillance and entertainment (such as image-based rendering and 3D video). This part will include the following topics:

Camera Geometric Models: the models and parameters that characterize the transformations from 2D images to 3D models.

Camera Calibration: the procedure of estimating the camera parameters and models given the 2D and 3D correspondences.

Stereo Vision: principles and techniques for computing 3D coordinates from images of two or more cameras.

Visual Motion: principles and techniques for estimating structures and motion characteristics from images recorded by a moving camera.

There are several disciplines closely related to 3D computer vision. Image processing deals with transformations from 2D images to 2D images and is often treated as a preprocessing step in computer vision, particularly in 3D computer vision. We refer to it as 2D computer vision in Part III. Computer Graphics is the study of utilizing 3D models for generating 2D

images from different views at different times and under different illuminations, and is, therefore, sometimes viewed as the forward processing of 3D transformation (whereas 3D computer vision solves the inverse problem of 3D transformation). Since they are forward and backward processes, using essentially the same sets of geometric transformations, learning 3D computer vision will be very helpful for students and professionals studying 3D graphics. Finally, Photogrammetry is the discipline of obtaining accurate measurements from images. In terms of 3D measurements, it actually shares a lot of common techniques with computer vision, even though it usually deals with satellite images. In short, we want our students to realize that learning 3D computer vision is a one-stone-many-birds deal!

The goal of this chapter is to build an understanding for the geometric relations between a 3D scene and its 2D images for both computer vision and computer graphics. The relations include two aspects: the 3D transformation that represents the viewpoints and viewing directions of the camera (real or virtual), and the perspective projection that maps 3D points to 2D points in images. Note that we can use the same set of equations for both vision and graphics. However, vision poses greater challenges than graphics: In graphics, 2D projections can easily be obtained from 3D models, given the 3D models and the (virtual) camera parameters, whereas in computer vision the projections are generated by a real camera. The reconstruction of 3D models from 2D images(the inverse problem) is much harder since (1) we need to find the intrinsic and extrinsic camera parameters by using calibration procedures that will be discussed in the next chapter; and (2) we need to recover 3D information from 2D images in which the third dimension is lost.

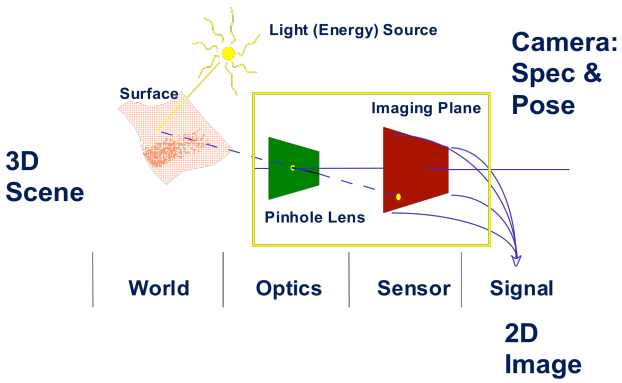


Figure 1.Camera model and camera parameters

From a geometric point of view, a camera is a device that captures 3D scenes from the real world and converts them into 2D images (Figure 1). Just as we move and orient our eyes in real life to get an optimal view, we have to change the camera's location and orientation in order to capture a good view of a scene. In order to recover the 3D structure of the scenes from their 2D images, we need to establish the relationship between the 2D shapes in the images and the 3D structures in the scenes, through the parameters of the camera. These parameters include two parts: the intrinsic parameters that map the 3D points onto 2D images, which is the camera specification; and the extrinsic parameters that represent the position and orientation (i.e., pose) of the camera.

In this chapter, we will discuss the geometric models of a typical camera that is widely used in computer vision – the pinhole camera. We will start by familiarizing ourselves with the geometry of a pinhole camera model, and then introduce the mathematical models of both perspective projection and weak-perspective projection. After this we will have a look at the camera parameters, including the so called intrinsic parameters, which define the mapping from 3D to 2D, and the extrinsic parameters, which define the viewpoint and viewing direction of the camera. Next, we will introduce the matrix representations of the transformation. We will formulate the linear version of the projection transformation equation, which will save a lot of space in writing the equations, and save a lot of time in deriving the relations for the later discussions of camera models in camera calibration, stereo vision and visual motion. We will provide four typical models using the linear algebra representations: the perspective camera model, the weak-perspective camera model, the affine camera model, and the camera model for planes. Finally, we summarize the key messages of this chapter and point out how we could use the models for 3D computer vision.

2. GEOMETRIC PROJECTION OF A CAMERA

We will start with the geometry of a pinhole camera model, and then introduce the mathematical models for both perspective projection and weak-perspective projection.

2.1 Pinhole camera model

First, let us take a look at the pinhole camera model without taking the position of the camera into account. The focal length f of the pinhole lens is the distance between the center of the lens and the image plane measured

along the optical axis (see Figure 2). The center of the lens (center of projection) is the point in the lens system through which all rays must pass.

From the diagram, we can easily see that the image projected onto the image plane is inverted. The ray from the cartoon figure's head, for example, is projected to a point near the bottom of the image plane, while the ray from a point on the figure's foot is projected to a point near the top of the image plane.

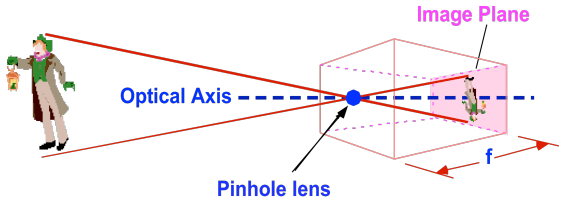


Figure 2.A diagram of the pinhole camera model. The focal length f marks the distance between the image plane and the pinhole.

Various coordinate reference frames can be associated with the pinhole camera system. In most of them, the Z axis coincides with the optical axis (also called the central projection ray). By convention, the image plane is located at Z = -f and the lens is located at Z = 0. Z is also the distance to an object as measured along the optical axis; also by convention, Z is positive coming out of the camera. The X and Y axis lie on the image plane - we will establish the exact relationships later.

Before we proceed, we would like to define the important concepts and notations for perspective projection:

(1) Optical axis: the direction of imaging, i.e. the Z axis.
(2) Image plane: a plane perpendicular to the optical axis upon which the 3D image is projected into 2D.
(3) Center of projection: the pinhole. It is also called focal point, viewpoint, or nodal point in imaging, computer vision, and photogrammetry.
(4) Focal length (f): distance from the focal point to the image plane.
(5) Field of View (FOV): viewing angles in horizontal and vertical directions give us the horizontal and vertical fields of view, respectively.

Figure 3 depicts some of the properties of perspective projection. From Figure 3(a) and (b), it can easily be seen that increasing the focal length f will enlarge the figure of the object, but decrease the FOV of the image, given the fixed size of the image plane area.

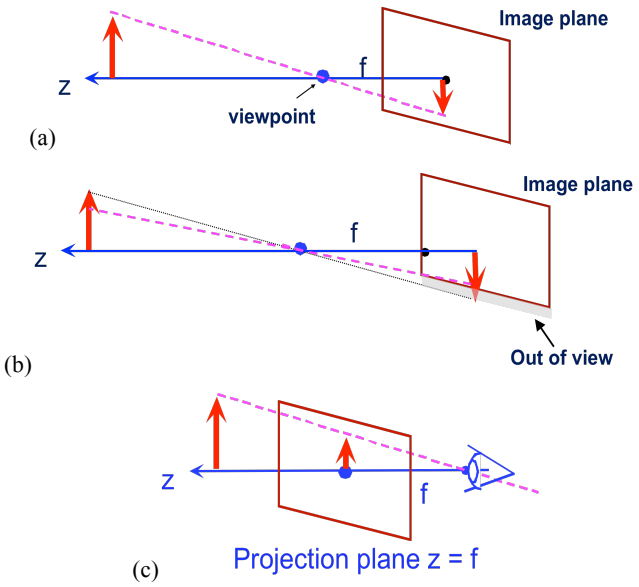


Figure 3. Pinhole camera model: coordinate system

Mathematically, it is possible to place the image plane in front of the camera lens so that the projected image is upright (Figure 3c). In the upright case, the center of projection is still located at the origin 0 along the optical axis. However, the image plane is placed at z=f. The equivalence of both geometries can be established by recognizing that the two triangles in the upper diagram are similar to the two triangles in the lower diagram. In fact, it is exactly this relationship which will allow us to derive the perspective projection equation.

2.2 Perspective projection

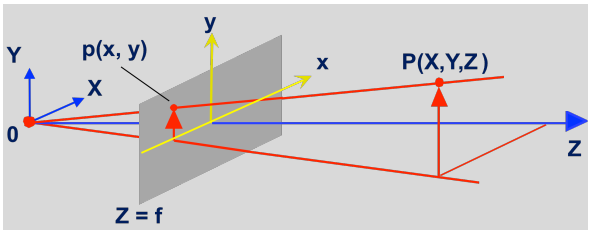


Figure 4. Perspective projection equation

Our goal is to compute the image coordinates (x, y) of the point P located at (X, Y, Z) in the world coordinate system under perspective projection. Equivalently, we want to know where the ray going from the point P to the center of the camera lens located at the center of projection (O) intersects with the image plane. We can do this rather simply by examining the projections of the point P on the XZ and YZ planes, and using similar triangles, we have:

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

Here we are assuming the camera coordinate system coincides with the world coordinate system.

We can see that, given the coordinates of a point (X,Y,Z) and the focal length of the lens f, the image coordinates (x,y) of the projection of the point on the image plane can be uniquely computed. Now let us ask the reverse question: Given the coordinates (x,y) of a point in the image, can we uniquely compute the world coordinates (X,Y,Z) of the point? In particular, can we recover the depth of the point (Z)?

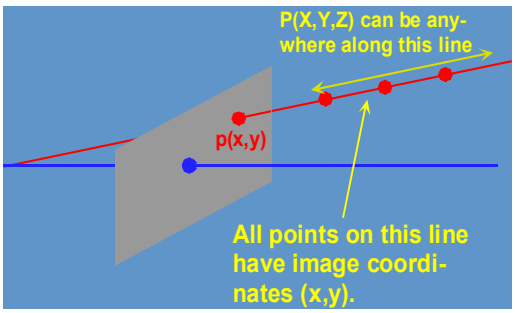


Figure 5. The reverse question: depth recovery from an image

To see that this is impossible, consider the infinitely extended ray through the center of projection and the point (x,y) on the image plane. Any point P in the world coordinate system which lies along this ray will project to the same image coordinates. Thus, in the absence of any additional information about P, we cannot determine its world coordinates solely from the knowledge of the coordinates of its image projection.

However, it is easy to show that if we know the image coordinates of the point P in two images taken from different locations, and if we know the relationship between the two sensor locations, then 3D information about the point can be recovered. This concept is known as stereo vision; in the human

visual system, our two eyes obtain slightly different images of the scene and which allows for depth perception.

In addition to using stereo vision, we can actually infer 3D structure using many other visual cues. In fact, we can get much more than 3D structure from an image. Figure 6 shows a picture of Amsterdam. Ask yourself: What do you see in this picture?

First of all, we may see "objects", such as a river, river bank, cars parked on the bank, buildings, trees in front of the buildings, a bridge, the sky, etc. Unfortunately, a machine vision system has a hard time getting to this level of understanding. Employing the immense computing power of the human brain, we can figure out relative distances, and we do see the 3D structure using just one image. This turns out to be a nontrivial task for a machine vision system.

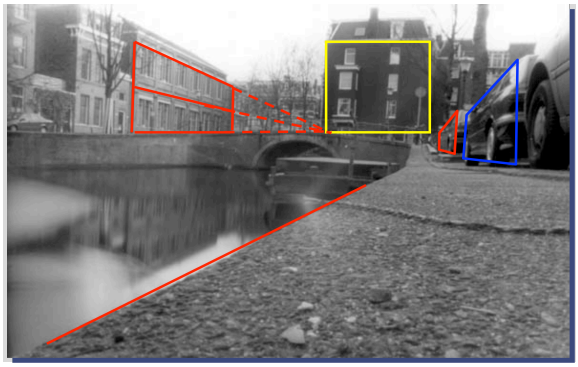


Figure 6. Amsterdam: What do you see in this picture? Photo by Robert Kosara, robert@kosara.net

Let's intuitively study some fundamentals of a perspective image. We will see what geometric properties are preserved in the 2D perspective image. This may provide some cues for us to infer 3D structure from images using machine vision algorithms.

(1) Straightness of lines is preserved (e.g., the straight line along the river bank). You can prove this using the perspective projection equation - Eq. (1): a 3D line in space which is not perpendicular to the image place projects to a 2D line in the image.

(2). Size is inversely proportional to the depth (distance) Z. This conclusion can be easily obtained from Eq. (1). Examples are the cars parked along the river.

(3). In general, angles and parallelism are not preserved unless they are of the front planes, i.e. 3D planes parallel to the image plane. For example, the right angles of the rectangular building change to acute angles (less than 90 degrees) in the nearer end and to obtuse angles (greater than 90 degrees) in the farther end.

(4). The projections of parallel lines converge to a point, known as the vanishing point of the set of parallel lines. A vanishing point is also shown in the figure.

(5). The shapes of objects usually change. However, planes that are parallel to the image plane keep their shape. This can be easily proven using Eq. (1). We may not be able to exactly find such a plane in the picture of Figure 6, but the façade of the building in the far end is approximately parallel to the image plane, so the shape is almost preserved.

Using these perspective cues, we can infer certain 3D structures from a single image. If you think about developing a computer algorithm to do the job instead of using your eyes and brain, here are a couple of important reminders for you to consider:

(1) Our eyes do not see individual pixels, but rather spatial shapes. The information of shapes could come from color patterns, texture patterns and boundaries.
(2) We typically include knowledge of object characteristics in our visual analysis (e.g., rectangular buildings, similar sizes of cars, rivers and banks in a plane, etc.).

This type of processing can be called a knowledge-based vision approach, or top-down approach, or the perceptual organization approach [Lowe'89]. Employing the projections of parallel lines of the façade of a building, we can e. g. estimate the orientation of the building and therefore its 3D structure. The order of the objects and their occlusion relations also offer clues of 3D information. For instance, the relative distances between the three cars along the bank of the river and the image plane can be easily inferred from the occlusions of the cars. Relative sizes of objects provide similar clues for relative distances: the cars which look smaller on the image are perceived as being farther away from the viewer. This conclusion is made by using the assumption that the cars have similar physical sizes, or by identifying the types of these vehicles and their physical sizes. We should, however, consider that when we say we can "easily infer", we are referring to human visual system. Often when we talk about sizes, occlusions, rectangular shapes, etc., we are actually utilizing a high level understanding

of the scene or high level knowledge of the displayed objects. It is extremely hard for a computer to use high-level knowledge and there has been only limited success in this direction [Lowe, Cambridge, Kanade] since the high-level knowledge is difficult to be described in computer algorithms. Therefore, top-down paradigms seem to be mainly the privilege of human visual perception, even though computer vision researchers keep coming back to this interesting approach once a while.

Instead, state-of-the-art computer vision techniques mainly employ bottom-up approaches for 3D reconstruction, which start from image pixels, and group them into features and objects, in both 2D and 3D. The major work in the past 50 years was based on the "structure from X" approaches, where X mainly represents stereo vision or visual motion. The idea is to recover 3D information of small local pieces (points, lines, patches), and then group and/or segment them into meaningful objects so that the computer may recognize the objects in images. Fully automated and complete 3D reconstruction is still in the process of being developed, but a large number of milestones have been reached by a number of interesting applications, such as image and video representations, coding and compression, interactive 3D video, object and event detection, image mosaics for entertainment, interface, virtual reality, surveillance and monitoring.

2.3 Weak-perspective projection

Even though angles and shapes are generally not preserved when 3D objects are projected on a 2D image, there have been efforts to generate simplified models which can be used under certain circumstances. A so-called weak perspective projection model can be derived, which actually is like an orthogonal projection with parallel rays followed by a scaling operation. This is plausible when the relative depth range of an object is much smaller than the distance from the camera to the object. In other words, if the average depth $\overline{Z}$ is much larger than the relative distance between any two points in the scene measured along the optical axis, the perspective projection equation can be simplified as

$$x = f\frac{X}{\overline{Z}}, y = f\frac{Y}{\overline{Z}} \tag{2}$$

where the depth Z of each point is replaced by an average depth $\overline{Z}$. Since all points of an object have the same depth, its image is simply a scaled version of the real object. As we have mentioned, equivalently, the weak

perspective projection can be viewed as an orthogonal projection with parallel rays from the object to the image plane, followed by an isotropic scaling. Weak perspective projection preserves both angles and shapes. We will come back to this model when we discuss camera parameters in section XY.

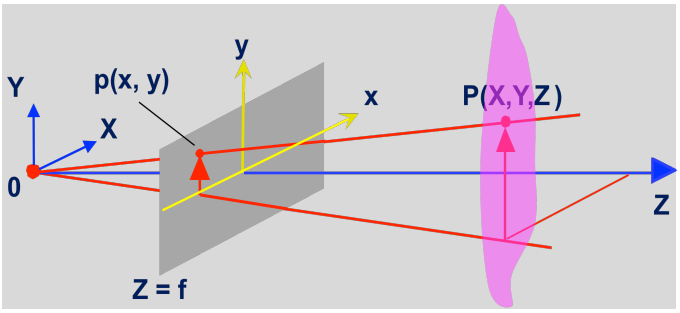


Figure 7. Weak perspective projection

3. CAMERA PARAMETERS

In this section, we will have a look at the camera parameters, including the intrinsic parameters, which define the mapping from 3D to 2D, and the extrinsic parameters, which define the viewpoint and viewing direction of the camera.

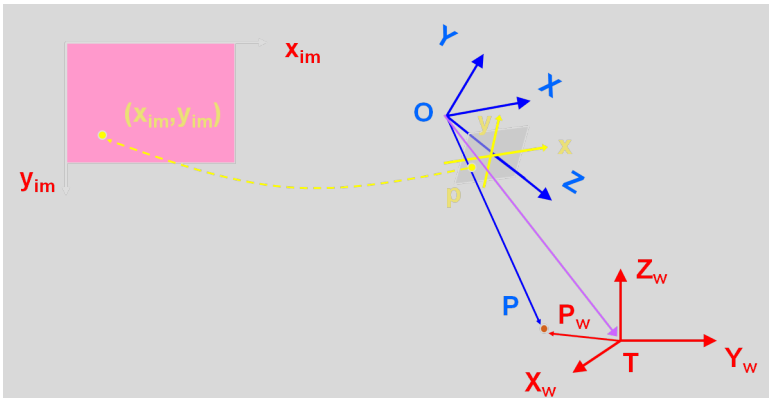


Figure 8. Intrinsic and extrinsic camera parameters

We will use the transformations of a 3D point in the scene to a 2D point in the image to describe the camera model. This transformation can be divided into three consecutive steps: the translation of 3D representations from a world coordinate system to a camera coordinate system, from its 3D representation in the camera coordinate system to its projection in a 2D

image of the camera, and finally from the 2D image inside the camera to a digital image coordinate system that we call "digital frame coordinate system". Overall, there are four points represented in four related coordinate systems (Figure 8):

(1) Point $p_{im}(x_{im}, y_{im})$ in the digital frame coordinate system, measured in pixels;
(2) Point p(x,y) in the physical image coordinate system, measured in mm;
(3) Point P(X,Y,Z) in the camera coordinate system, measured in mm;
(4) Point $P_w(X_w,Y_w,Z_w)$in the world coordinate system, measured in mm.

Note that the points P and P_w are actually two representations of the same 3D point, in two different coordinate systems. The camera parameters can be divided into two groups:

(1) Intrinsic parameters, which link the digital frame coordinates of a physical image point to its corresponding camera coordinates.
(2) Extrinsic parameters, which define the location and orientation of the camera coordinate system with respect to the world coordinate system.

Some of the intrinsic parameters can be found in the *specification* of the camera and the digital image capture system (or frame grabber when the camera output is analog video signals), although they might be incomplete or inaccurate. Extrinsic parameters describe the *pose* (position and orientation) of the camera relative to the scene. These parameters need to be estimated with respect to a world coordinate system, especially when the camera is in motion. If we are only concerned with the 3D measurements obtained from individual camera poses, then the two coordinate systems (the camera and the world coordinate systems) can be the same. In this case the 3D measurements are only camera-centered, or viewer-centered. However, if you want to associate 3D measurements from two or more camera poses, you need to define a common "world coordinate system", which may be defined in relation to an object in the scene, or simply a *reference* camera pose. In this way, the 3D reconstruction is sometimes referred to as "object-centered".

For accurate 3D reconstruction or 3D measurements, we first need to find all these parameters by a procedure called camera calibration (Chapter 4). In this chapter, we will focus on understanding the camera model and its parameters in detail.

3.1 Intrinsic parameters: mapping from 3D to 2D

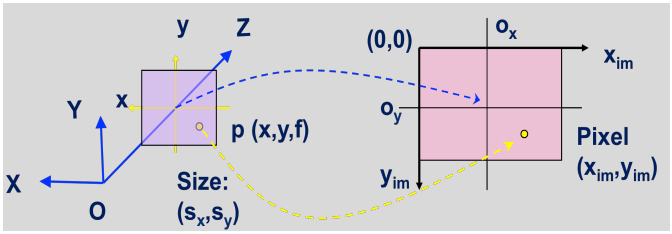


Figure 9. Intrinsic camera parameters

First, using the perspective projection equation – Eq. (1), a 3D point P (X,Y,Z) is projected into a 2D physical image point p(x,y), represented in the physical image coordinate system. For easy reading, the equation is listed here again:

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

where f is the focal length of the camera. Note that the origin of the physical image coordinate system is the point where the optical axis (Z axis) passes through the physical image plane which is parallel to the XOY plane of the camera coordinate system. We also have x in parallel to X (x//X), and y//Y. The image on the physical image plane is measured in metric units, e.g., millimeters (mm), whereas, for a digital frame (to differentiate it from the image on the physical image plane, we call it a digital frame, named after video frames), we will follow the conventional representation: The origin (0,0) is conventionally placed at the top-left corner of the image, and the x_{im} axis runs from left to right, and the y_{im} axis runs from top to bottom. A frame consists of pixels (picture elements); each pixel has a location characterized by its column and row numbers, (x_{im}, y_{im}), and each of the pixels corresponds to a rectangular area of the same size in the physical image plane, characterized by the dimensions of the pixel in x and y directions, as (s_x, s_y), measured in millimeters. To transform a point from the physical image coordinate system to the digital frame coordinate system, we need to define the image center, which is the point corresponding to the origin of the physical image coordinate system. Given these definitions, the relation between the physical image point and the digital frame point can be written as

$$x = -(x_{im} - o_x)s_x$$
$$y = -(y_{im} - o_y)s_y \quad (3)$$

In summary, the intrinsic parameters of a perspective camera mainly include the following:

- (o_x, o_y) : the image center (in pixels);
- (s_x, s_y) : the effective size of the pixel (in mm);
- f: the focal length of the camera (in mm).

The mapping from the 3D point P (X,Y,Z) to the frame point (x_{im},y_{im}) are represented by Eq. (1) and Eq. (3).

In some real applications, such as accurate 3D measurements, and particularly when using distorted lenses (for example, wide angle lenses), distortions may need to be removed before subsequent processing. Usually, the lens distortion can be modeled as a simple radial distortion (Figure 10):

$$x = x_d(1 + k_1 r^2 + k_2 r^4)$$
$$y = y_d(1 + k_1 r^2 + k_2 r^4) \quad (4)$$

where (x_d, y_d) is the distorted point in the real image plane, (x,y) is the corrected image point, $r^2 = x_d^2 + y_d^2$, and k_1, k_2 are the distortion coefficients. Typically, a model with $k_2 = 0$ is still accurate for a CCD camera of size 500x500, with about 5 pixels distortion at the outer boundary. Adding the lens distortions to the transformations will significantly complicate the equations. Usually distortion can be removed before we use images for 3D measurements. For the purpose of focusing on the principle of 3D geometry and calibration, let's ignore the radial distortion for the moment.

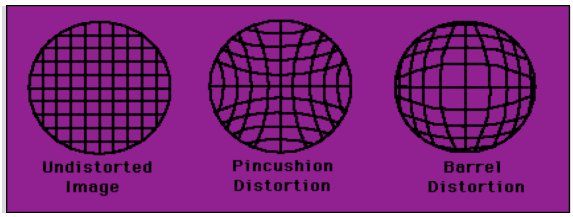


Figure 10. Lens distortions

3.2 Extrinsic parameters: viewpoint and viewing direction

The extrinsic parameters of the camera define the "pose" (position and direction) of the camera, which include:

(1) A 3D translation vector T, describing the relative positions of the origins of the world and camera coordinate systems to each other.
(2) A 3x3 rotation matrix R, an orthogonal matrix that brings the corresponding axes of the two systems into alignment.

The transformation of the 3D point from its representation in the world coordinate system to that in the camera coordinate system can be written as:

$$\mathbf{P} = \mathbf{R}\,\mathbf{P}_w + \mathbf{T} \tag{5}$$

Understanding this equation, including its rotation matrix and translation vector, is key to understanding the 3D transformations. We will discuss this further in the next few paragraphs. Since we are using linear algebra for the ease of representation and manipulation, we will first review the basic concepts and operations in linear algebra, namely vectors and matrices, and then explain the physical meaning of the 3D rotation matrix and the translational vector.

3.2.1 Vectors and operations

A point is represented as a 2D/ 3D (column) vector. For example:

- Image point: 2D vector $\mathbf{p} = \begin{pmatrix} x \\ y \end{pmatrix} = (x, y)^T$
- Scene point: 3D vector $\mathbf{P} = (X, Y, Z)^T$
- Translation: 3D vector $\mathbf{T} = (T_x, T_y, T_z)^T$

Note that in order to save space in writing these vectors, we sometimes use the transpose of the corresponding row vector for a column vector. We only show an example of the image point above. The transpose of the row vector, $(x,y)^T$, equals to the original column vector.

The basic vector operations include addition, dot product and cross product.

Addition and subtraction. An example is the translation of a 3D point. If there is no rotation between the world and the camera coordinate systems, the transformation of a 3D point from the world to the camera is a *pure translation* of the vector $\mathbf{P}_w$ by the amount of $\mathbf{T}$:

$$\mathbf{P} = \mathbf{P_w} + \mathbf{T} = (X_w + T_x, Y_w + T_y, Z_w + T_z)^T \tag{6}$$

This is possible since the point $\mathbf{P_w}$ as a vector is the same in both coordinate systems since the three pairs of coordinate axes are all aligned. Alternatively, we could say that the camera coordinate system is generated by moving (translating) the world coordinate system backward by the amount of **T**. The three components of the new vector **P** are simply the additions of the corresponding components of the vectors $\mathbf{P_w}$ and **T**. On the other hand, the transformation from the camera coordinates to the world coordinates can be written as the subtraction of vectors:

$$\mathbf{P_w} = \mathbf{P} - \mathbf{T} \tag{6a}$$

From Eq. (6), it can be seen that *the translational vector T is the representation of the origin of the world coordinate system in the camera coordinate system*. This is also true when there is a rotation between the two systems. To illustrate this, we can simply plug in the values of the origin $\mathbf{P_w} = (0,0,0)^T$ in Eq. (5), and we will obtain $\mathbf{T} = \mathbf{P}$, which means that the translation vector **T** is the vector representation of the world origin in the camera coordinate system.

Dot product. The dot production of two vectors $\mathbf{a} = (a_1, a_2, a_3)$ and $\mathbf{b} = (b_1, b_2, b_3)$ is a scalar value c:

$$c = \mathbf{a} \bullet \mathbf{b} = a_1 b_1 + a_2 b_2 + a_3 b_3 = |\mathbf{a}||\mathbf{b}| \cos\theta \tag{7}$$

where θ is the angle between the two vectors, and |**a**| and |**b**| are the magnitudes of the two vectors. If **a** is a unit vector, i.e., |**a**| = 1, then c is the magnitude of the projection of the vector **b** on to the vector **a**. This automatically implies that the dot product of two orthogonal vectors is equal to zero.

Cross product. The cross product of two vectors **a** and **b** generates a new vector **c** that is orthogonal to both **a** and **b**:

$$\mathbf{c} = \mathbf{a} \times \mathbf{b} = (a_2 b_3 - a_3 b_2, a_3 b_1 - a_1 b_3, a_1 b_2 - a_2 b_1)^T \tag{8}$$

To illustrate this, let us assume $\mathbf{a} = (1,0,0)^T$, $\mathbf{b} = (0,1,0)^T$, then $\mathbf{c} = (0,0,1)^T$. They are actually the three unit vectors along the three axes of a 3D coordinate system. We will come back to these operations in the following sections, and chapters for their practical uses.

3.2.2 Rotation matrix and its physical meaning

We will discuss matrices and their operations mainly through the analysis of the rotation matrix and the 3D transformation in Eq. (5). The rotation matrix **R** is a 3x3 matrix that can be written as:

$$\mathbf{R} = \left(r_{ij}\right)_{3\times 3} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_1^T \\ \mathbf{R}_2^T \\ \mathbf{R}_3^T \end{bmatrix} \tag{9}$$

where $\mathbf{R}_i^T$ is the *ith* row (i=1,2,3) of the rotation matrix and can also be regarded as a row vector. In other words, $\mathbf{R_i}$ is a column vector, the *ith* column of the transpose of the rotation matrix $\mathbf{R^T}$. The following paragraphs summarize the most important properties of the rotation matrix.

The rotation matrix is an orthogonal matrix:

$$\mathbf{R}\mathbf{R}^T = \mathbf{R}^T\mathbf{R} = \mathbf{I} \tag{10}$$

where $\mathbf{R}^T$ is the transpose of **R**, i.e., $\mathbf{R}^T = (\mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3)$, and **I** is an identity matrix, a 3x3 square matrix with 1's on the main diagonal and 0's elsewhere. If you are not familiar with the orthogonality of a rotation matrix, you may work out the following example to see how it satisfies Eq. (10).

$$\mathbf{R} = \left(r_{ij}\right)_{3\times 3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos r & \sin r \\ 0 & -\sin r & \cos r \end{bmatrix}$$

Equation (10) has the following implications.

(1) The rotation matrix as an orthogonal matrix - a square matrix with real entries whose columns (and rows) are orthogonal unit vectors, i.e.

$$\mathbf{R}_i \bullet \mathbf{R}_j = \mathbf{R}_i^T\mathbf{R}_j = \begin{cases} 1, i = j \\ 0, i \neq j \end{cases} \tag{11}$$

Note that we can write the dot product of two column vectors as the multiplication of two matrices (the first one is a row vector, and the second one is a column vector), and the result is a scalar value (either 1 or 0).

(2) Any row of the rotation matrix can be calculated from the other two rows. For example,

$$\mathbf{R}_3 = \mathbf{R}_1 \times \mathbf{R}_2 \tag{12}$$

This is true since all three column vectors are unit vectors, and they are mutually orthogonal to each other. Note that here we use the cross product defined in Eq. (8).

(3) The three column vectors $\mathbf{R}_i$ (i=1, 2, 3) of the rotation matrix are actually three unit vectors in the directions of the three camera axes (X, Y and Z), represented in the world coordinate system(see Figure 11, the dashed axes). Therefore, Eq. (5) literally means that a point ($\mathbf{P_w}$) in the world coordinate system is first projected on three new axes (of the camera coordinate system) and then translated by the amount of **T** to a new origin O. The individual steps of this transformation can be retraced in the following expansion of Eq. (5):

$$\mathbf{P} = \mathbf{R}\mathbf{P_w} + \mathbf{T} = \begin{pmatrix} r_{11}X_w + r_{12}Y_w + r_{13}Z_w + T_x \\ r_{21}X_w + r_{22}Y_w + r_{23}Z_w + T_y \\ r_{31}X_w + r_{32}Y_w + r_{33}Z_w + T_z \end{pmatrix} = \begin{bmatrix} \mathbf{R}_1^T\mathbf{P_w} + T_x \\ \mathbf{R}_2^T\mathbf{P_w} + T_y \\ \mathbf{R}_3^T\mathbf{P_w} + T_z \end{bmatrix} \tag{13}$$

For example, the dot product of the two vectors, $\mathbf{R}_1 \bullet \mathbf{P_w} = \mathbf{R}_1^T\mathbf{P_w}$ represents the length of projection (or *scalar projection*) of the vector $\mathbf{P_w}$ onto the unit vector $\mathbf{R_1}$. Note that it is only valid to perform the dot product, if both vectors are represented in the same coordinate system. The result is a scalar value. In the case of *pure rotation,* that is, when all translatory components are zeros (i.e., **T** = **0**), this dot product is equal to the X coordinate of the point P_W represented in the camera coordinate system. In other words, in the case of a pure rotational transformation, the multiplication of the rotation matrix to a point in the world coordinate system obtains its representation in the camera coordinate system by performing three scalar projections. To see why this is true, let's assume the rotation matrix was an identity matrix **I**, i.e., there was no rotation between the two coordinate systems. In this case, $\mathbf{R_1} = (1, 0, 0)^T$, therefore $X = \mathbf{R}_1 \bullet \mathbf{P_w} = X_w$. With both rotation and translation, the transformation

process consists of three projections and three shifts: the X, Y, and Z components are projected onto the three camera axes and then shifted in the directions of the three axes, respectively, as shown in Eq. (6).

(4) Because the rotation matrix is orthogonal, its inverse is its transpose, i.e.

$$\mathbf{R}^{-1} = \mathbf{R}^{T} \tag{14}$$

This makes the inverse transformation of Eq. (5) much easier. Given the point **P** in the camera coordinate system, its coordinates in the world coordinate system can be calculated as

$$\mathbf{P_w} = \mathbf{R}^{-1}(\mathbf{P} - \mathbf{T}) = \mathbf{R}^{T}\mathbf{P} + (-\mathbf{R}^{T}\mathbf{T}) \tag{15}$$

Note that the rotation matrix from the camera to the world coordinate systems is simply the transpose of the original rotation matrix, and the translation vector is ($-\mathbf{R}^{\mathrm{T}}\mathbf{T}$). Similar to the physical meanings of Eq. (5), the three rows of $\mathbf{R}^{\mathrm{T}}$, i.e., the three columns (vectors) of **R** are the representations of the three coordinate axes of the world coordinate system in the camera coordinate system. This indicates that we can estimate **R** from three lines mutually orthogonal (i.e. perpendicular) to each other in space, if we use a world coordinate system that aligns with two of the three orthogonal lines. Just as the translational vector T represents the origin of the world coordinate system in the camera coordinate system, the translation vector $-\mathbf{R}^{\mathrm{T}}\mathbf{T}$ is the representation of the origin of the camera coordinate system in the world coordinate system.

(5) The 3D rotation has only 3 degrees of freedom (DOF). You may wonder why a rotation matrix with 9 elements has only 3 DOF. This will be easy to explain given the observations above. First of all, the third row of the rotation matrix can be derived from the first two rows using Eq. (12), because we demand orthogonality between the axes. This reduces the number of independent parameters to 6 = 3+3. Since both the first and second row vectors are unit vectors, any one component can be calculated from the other two in each vector, using Eq. (11). Hence the number of independent parameters is further reduced to 4. Finally, since the first two row vectors are orthogonal, we may add an additional constraint to reduce the number of independent parameters by 1 (i.e. any of the four parameters can be represented by the rest of the three, again, using Eq. (11)). In the end we only have three (3) independent parameters in the 3x3 rotation matrix, which leads to the conclusion that the 3D rotation has only 3 DOF!

3.2.3 Generating R from three rotation angles

In the following section, we will see that the rotation matrix has only 3 DOF from a different perspective. We will try to represent the rotation matrix using three rotation angles. We first try to construct a rotation matrix from three independent rotation angles, and then we will show how to calculate the three angles from the rotation matrix. While the latter is rather simple, constructing a rotation matrix from three independent rotation angles is not trivial at all.

Let's first move the origin of the camera coordinate system to that of the world coordinate system (Fig. 11). Then the ultimate goal is to bring three axes of the world coordinate system to the corresponding axes of the camera coordinate system. The result will give us the transformation of the coordinates of a point from the world coordinate system to the camera coordinate system.

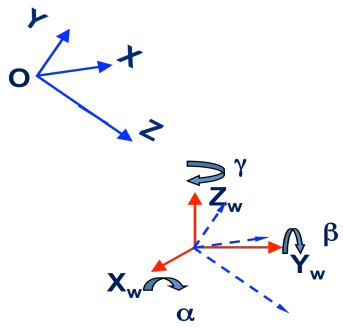


Figure 11. Constructing rotation matrix from three consecutive rotation angles

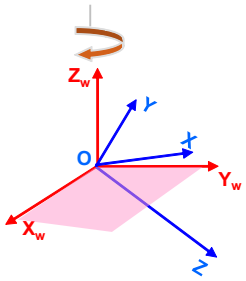
It is vital to design the rotation matrix in such a way that every column of R represents the rotation around only one of the axes of the (changing) world coordinate system. Let's assume we will rotate in the order of γ,β,α around Zw, Yw and Xw, consecutively. As a first step, we try to bring one of the axes, e.g. Xw, to its corresponding axis (in this example, X) in the camera coordinate system. This can, however, not necessarily be achieved by a single rotation around the Zw axis since X is usually not in the plane XwOYw. We, therefore, need to take two steps to achieve this goal . First, we rotate the world coordinate system around Zw axis until Yw is perpendicular to X. Due to the orthogonal coordinate systems of XwYwZw and XYZ, X (OX) needs to be in the plane XwOZw (and consequently Yw in the plane YOZ since X can only be perpendicular to a line in the plane YOZ that passes through O) . As a second step, we bring Xw to X by rotating the world coordinate system around the new Yw axis since X is in

the plane XwOZw. After this rotation, both Yw and Zw will be in the plane XOZ so that a rotation around the new Xw (i.e. X) will bring Yw to Y and Zw to Z simultaneously.

Now we are ready to write the equations for the three consecutive rotations around the three axes to achieve our goal.

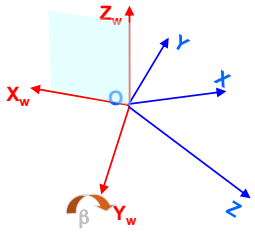
(1) Rotation γ around the Zw Axis

$$\mathbf{R}_{\gamma} = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



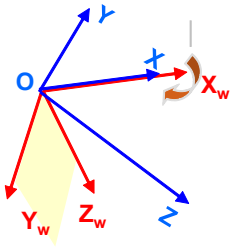
(2) Rotation β around the Yw Axis

$$\mathbf{R}_{\beta} = \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix}$$



(3) Rotation α around the Xw(X) Axis

$$\mathbf{R}_{\alpha} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$



The result of the three consecutive rotations around the three coordinate axes can be written as:

$$\mathbf{R} = \mathbf{R}_\alpha \mathbf{R}_\beta \mathbf{R}_\gamma \tag{16}$$

Here are a few additional notes:

(1) In practice we will have to be careful about the directions of the rotations (that is the sign of the sine term) to make sure that we have the correct transformation.

(2) The order of the three matrices in the multiplications matters; following the procedure just described above, we have one possible order: γ,β,α. For the same R, we could have 6 different sets of α,β,γ.

(3) R is a non-linear function of α,β,γ:

$$\mathbf{R} = \begin{bmatrix} \cos\beta\cos\gamma & -\cos\beta\sin\gamma & -\sin\beta \\ -\sin\alpha\sin\beta\cos\gamma + \cos\alpha\sin\gamma & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & -\sin\alpha\cos\beta \\ \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma & -\cos\alpha\sin\beta\sin\gamma + \sin\alpha\cos\gamma & \cos\alpha\cos\gamma \end{bmatrix} \tag{17}$$

It's easy to compute the respective potation angles from R using Eq. (17).

4. CAMERA MODELS REVISITED

As a summary of the previous discussions, we will wrap everything up in one simple linear version for the projection transformation. Then we will provide four typical models using the linear algebra representations: the perspective camera model, weak-perspective camera model, affine camera model, and camera model for planes.

4.1 Linear version of the projection transformation

The transformation from World to Camera (Linear equations)

$$\mathbf{P} = \mathbf{R}\mathbf{P}_\mathbf{w} + \mathbf{T} = \begin{pmatrix} r_{11}X_w + r_{12}Y_w + r_{13}Z_w + T_x \\ r_{21}X_w + r_{22}Y_w + r_{23}Z_w + T_y \\ r_{31}X_w + r_{32}Y_w + r_{33}Z_w + T_z \end{pmatrix} = \begin{bmatrix} \mathbf{R}_1^T\mathbf{P}_\mathbf{w} + T_x \\ \mathbf{R}_2^T\mathbf{P}_\mathbf{w} + T_y \\ \mathbf{R}_3^T\mathbf{P}_\mathbf{w} + T_z \end{bmatrix}$$

Camera: $P = (X,Y,Z)^T$
World: $Pw = (X_w,Y_w,Z_w)^T$

Transformation: R, T

The Transformation from camera to physical image (non-linear equations)

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

Camera: $P = (X,Y,Z)^T$
Image: $p = (x,y)^T$

The Transformation from physical image to digital frame (linear if neglecting radial distortion)

$$x = -(x_{im} - o_x)s_x$$
$$y = -(y_{im} - o_y)s_y$$

digital frame $(x_{im}, y_{im})^T$

Integrating the transformations given above, we obtain the transformation from the world coordinate system to the digital frame coordinate system, i.e., $(X_w,Y_w,Z_w)^T$ -> $(x_{im}, y_{im})^T$

$$x_{im} - o_x = -f_x \frac{r_{11}X_w + r_{12}Y_w + r_{13}Z_w + T_x}{r_{31}X_w + r_{32}Y_w + r_{33}Z_w + T_z}$$
$$y_{im} - o_y = -f_y \frac{r_{21}X_w + r_{22}Y_w + r_{23}Z_w + T_y}{r_{31}X_w + r_{32}Y_w + r_{33}Z_w + T_z} \quad (18)$$

where we define the effective focal lengths in the x and y directions as

$$f_x = f/s_x,\ f_y = f/s_y \quad (19)$$

The two equations in (18) are non-linear. In order to turn them into a linear equation system, we will employ some notations in the so-called projective space. We will add a fourth coordinate to the world point Pw, as

$$P_w = (X_w, Y_w, Z_w, 1)^T$$

Meanwhile, we define $(x_1,x_2,x_3)^T$ such that

$$\begin{pmatrix} x_{im} \\ y_{im} \end{pmatrix} = \begin{pmatrix} x_1 / x_3 \\ x_2 / x_3 \end{pmatrix} \tag{20}$$

By introducing the projective representation $(x_1,x_2,x_3)^T$ of the physical image point, we finally have a linear representation of the perspective projection transformation as

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \mathbf{M}_{\mathbf{int}}\mathbf{M}_{\mathbf{ext}} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} \tag{21}$$

where $\mathbf{M}_{\mathbf{ext}}$ is a 3x4 matrix that only includes the extrinsic parameters

$$\mathbf{M}_{ext} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \end{bmatrix} = \begin{bmatrix} \mathbf{R}_1^T & T_x \\ \mathbf{R}_2^T & T_y \\ \mathbf{R}_3^T & T_z \end{bmatrix} \tag{22}$$

and $\mathbf{M}_{\mathbf{int}}$ is a 3x3 matrix that includes only the intrinsic parameters, i.e.,

$$\mathbf{M}_{\text{int}} = \begin{bmatrix} -f_x & 0 & o_x \\ 0 & -f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \tag{23}$$

The final Projective Matrix is simply the matrix product of the intrinsic and extrinsic matrices:

$M= M_{int}M_{ext}$

M represents the linear transformation from a projective space $(X_w,Y_w,Z_w, 1)^T$ to a projective plane$(x_1, x_2, x_3)^T$. Note that M is defined up to a scale factor, and it only has 11 independent entries.

4.2 Some typical camera models

(1). Perspective Camera Model

In order to understand what the elements of the projection transformation matrix are, we will make some assumptions to simplify the representation. Let us assume that the image center is known, and its coordinates are, without loss of generality, Ox = Oy = 0. The pixels are square, i.e., Sx = Sy = 1. After these simplifications, the number of independent entries is reduced to 7 from 11. Hence we have

$$\mathbf{M} = \begin{bmatrix} -fr_{11} & -fr_{12} & -fr_{13} & -fT_x \\ -fr_{21} & -fr_{22} & -fr_{23} & -fT_y \\ r_{31} & r_{32} & r_{33} & T_z \end{bmatrix} \tag{24}$$

This equation shows how the 7 intrinsic and extrinsic parameters are packed into a single matrix.

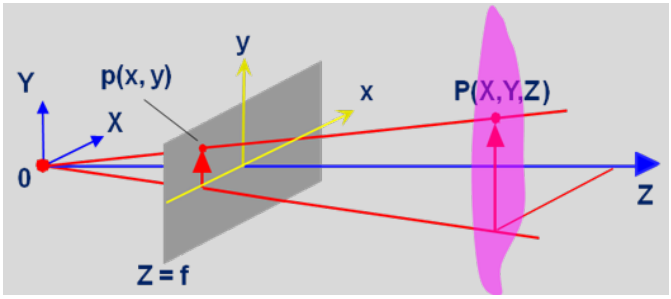


Figure 12. Weak perspective projection

(2). Weak-Perspective Camera Model

If we assume that the average distance between the viewpoint and the observed object, $\overline{\mathbf{Z}}$, is much larger than the depth range of the object in the camera coordinate system (Figure 12), we can approximate every depth Z to be the average depth. The average depth can then be calculated as the depth of the centroid vector $\overline{\mathbf{P}}_w$ of the object represented in the world coordinate system, as,

$$\mathbf{Z} = \overline{\mathbf{Z}} = \mathbf{R}_3^{\mathrm{T}}\overline{\mathbf{P}}_w + T_z$$

Therefore, the transformation matrix of the weak perspective projection can be represented as

$$\mathbf{M}_{wp} = \begin{bmatrix} -fr_{11} & -fr_{12} & -fr_{13} & -fT_x \\ -fr_{21} & -fr_{22} & -fr_{23} & -fT_y \\ 0 & 0 & 0 & \mathbf{R}_3^{\mathrm{T}}\overline{\mathbf{P}}_w + T_z \end{bmatrix} \tag{25}$$

which has only 9 non-zero entries (8 of which are independent) instead of 12. Moreover, we have made an approximation that all points of the object have the same depth 'Z' because the relative range of the object is much small than the distance from the camera to the object. This implies that, after the transformation from the world coordinate system to the camera coordinate system, the image is just a scaled version of the real object. Equivalently, the projection can be viewed as an orthogonal projection with parallel rays from the object (represented in the camera coordinate system) to the physical image plane, followed by an isotropic scaling. Note that the object in the world coordinate system is still represented with 3D coordinates. The difference lies in the fact that all points will be constrained to a plane that is perpendicular to the optical axis of the camera.

(3). Affine Camera Model

The affine camera model is a mathematical generalization of the weak-perspective transformation. Even though it does not correspond to any physical cameras, it has a simple equation and appealing geometry. While it does not preserve angles, it does preserve parallelism, and has only 8 independent entries:

$$\mathbf{M}_{af} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ 0 & 0 & 0 & b_3 \end{bmatrix} \tag{26}$$

4.3 Camera models for a plane

Planes are very common in the man-made world. By using a plane equation we add an additional constraint for all points on the plane:

$$n_x X_w + n_y Y_w + n_z Z_w = d$$

which can be written in the matrix form as

$$\mathbf{n}^T\mathbf{P}_w = d \tag{27}$$

where **n** is the parameters (normal and distance) of the plane, and $\mathbf{P}_w$ is a 3D point on the plane. Usually, Z_w can be written as a function of X_w and Y_w. If we, for example, consider the special case of a mobile robot moves on the floor, the plane equation is as simple as

Zw=0

For every point on the ground plane, this yields **Pw** = $(Xw, Yw, 0, 1)^T$. The 3D point (Xw, Yw, Zw) is imaged as a 2D point (Xw, Yw), and we have the following model for a plane projection:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{bmatrix} -fr_{11} & -fr_{12} & -fT_x \\ -fr_{21} & -fr_{22} & -fT_y \\ r_{31} & r_{32} & T_z \end{bmatrix} \begin{pmatrix} X_w \\ Y_w \\ 1 \end{pmatrix} \tag{28}$$

Note that the transformation matrix is only has a size of 3x3 and has only 8 independent parameters. Both characteristics are preserved in the general case. If $n_z <> 0$, without the loss of generality, we can assume $n_z = 1$, therefore the plane equation can be written as

$$Z_w = d - n_x X_w - n_y Y_w$$

We can easily derive the following 3x3 transformation matrix for the more general case by plugging the above equation into Eq. (21), with the M matrix in Eq. (24):

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{bmatrix} -f(r_{11} - n_x r_{13}) & -f(r_{12} - n_y r_{13}) & -f(dr_{13} + T_x) \\ -f(r_{21} - n_x r_{23}) & -f(r_{22} - n_y r_{23}) & -f(dr_{23} + T_y) \\ (r_{31} - n_x r_{33}) & (r_{32} - n_y r_{33}) & (dr_{33} + T_z) \end{bmatrix} \begin{pmatrix} X_w \\ Y_w \\ 1 \end{pmatrix} \tag{29}$$

The key is to distribute the column of the M matrix corresponding to Z_w to the other three columns. Here is an interesting observation: if we identify four known points on a plane (in the world coordinate system) that can also be viewed and extracted in an image, we can use Equation (28) or (29) to estimate the 3x3 transformation matrix. After this calibration, we can calculate the 3D coordinates of any point on the plane in the world

coordinate system. In that sense, we can obtain depths of points of a 3D plane from a single image.

5. CONCLUDING REMARKS

5.1 Summary

In this chapter, we have discussed the geometric projection of a camera. We started with the pinhole camera model, derived the perspective projection transformation and finally the weak-perspective projection.

We further discussed the camera parameters including the intrinsic parameters (f, o_x,o_y, s_x,s_y,k_1) and extrinsic parameters (R, T). We derived the linear equations of the camera models (without distortion) and then the 3x4 matrix for the general projection transformation equations (11 parameters). We have also introduced the following four models:

Perspective Camera Model: 11 parameters
Weak-Perspective Camera Model: 8 parameters
Affine Camera Model: generalization of weak-perspective: 8 parameters
Projective transformation of planes: 8 parameters

5.2 Applications and related issues

The camera models, in particular the full perspective transformation model, will be used in rendering, reconstruction and camera calibration. The following is a list of related issues:

(1) Graphics /Rendering
Goal: From 3D world to 2D image
Tasks:
Changing viewpoints and directions
Changing focal length
Fast rendering algorithms

(2) Vision / Reconstruction
Goal: From 2D image to 3D model
Tasks:
Inverse problem, much harder / unsolved
Robust algorithms for matching and parameter estimation

Need to estimate camera parameters first

(3) Calibration
Goal: Find intrinsic & extrinsic parameters given image-world point pairs
Tasks:
11 independent entries
<-> 10 parameters: f_x, f_y, o_x, o_y, α,β,γ, T_x,T_y,T_z
Probably a partially solved problem ?

6. QUESTIONS AND PROJECTS

6.1 Questions

[Question. 1] Please derive the equations to calculate the three rotation angles from the rotation matrix R in Eq. (17).

[Question. 2] Prove that the vector from the viewpoint of a pinhole camera to the vanishing point (in the physical image plane) of a set of 3D parallel lines is parallel to the parallel lines.

Hint: Due to perspective, all parallel lines in 3D space appear to meet at a point on the image - the *vanishing point*, which is the common intersection of all the image lines. Figure 13 shows the vanishing point VP1 in the image of a set of parallel lines on the calibration cube in space.

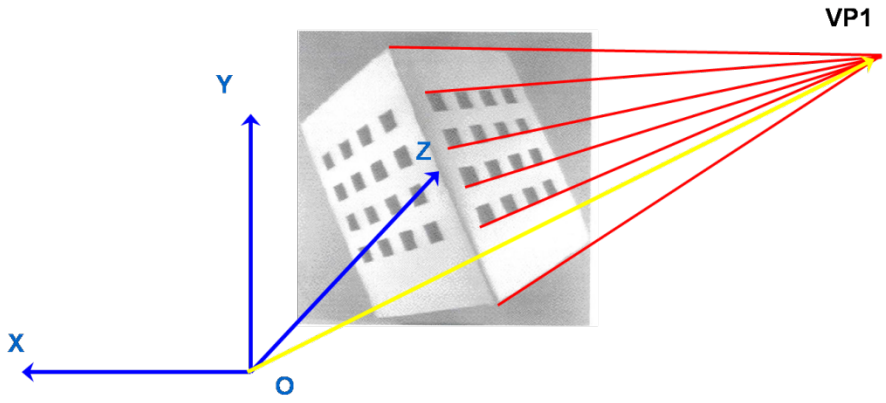


Figure 13. Vanishing point

You can either use geometric reasoning or algebraic calculation.

If you choose to use geometric reasoning, you can use the fact that the projection of a 3D line in space is the intersection of its "interpretation plane" with the physical image plane. Here the interpretation plane (IP) is a plane including both the 3D line and the center of projection (viewpoint) of the camera. Also, the interpretation planes of two parallel lines intersect in a line passing through the viewpoint, and the intersection line is parallel to the parallel lines.

If you decide to use algebraic calculation, you may use the parametric representation of a 3D line: P = P0 +tV, where $P= (X,Y,Z)^T$ is any point on the line (here ^{T} denote for transpose), $P0 = (X0,Y0,Z0)^T$ is a given fixed point on the line, vector $V = (a,b,c)^T$ represents the direction of the line, and t is the scalar parameter that controls the distance (with sign) between P and P0.

[Question. 3]. Show that relation between any image point $(x_{im}, y_{im})^T$ of a plane (in the form of $(x_1,x_2,x_3)^T$ in projective space) and its corresponding point $(X_w, Y_w, Z_w)^T$ on the plane in 3D space can be represented by a 3x3 matrix. You should start from the general form of the camera model $(x_1,x_2,x_3)^T = M_{int}M_{ext} (X_w, Y_w, Z_w, 1)^T$, where the image center (o_x, o_y), the focal length f, the scaling factors(s_x and s_y), the rotation matrix R and the translation vector T are all unknown. Note that in Eq. (29) we still assume $nz \neq 0$ which might not be true. You should use the general form of the projective matrix, and the general form of a plane $n_xX_w + n_yY_w + n_zZ_w = d$. Please derive the equation(s) that work for all the cases.

6.2 Projects

[Project 1]. Using the results of [Question 3], design a program that can map a planar surface from one image to another. You may first derive a relation between the corresponding 2D image points in the two images, and analyze how many pairs of correspondences you need to build up the relation between the two views of the plane. Then design an efficient algorithm that can map all points of one image into the other. Consider how to handle overlapping pixels and non-overlapping pixels.

REFERENCES

Digital Image Processing
Computer Graphics
Photogrammetry