Camera models

3D Computer Vision

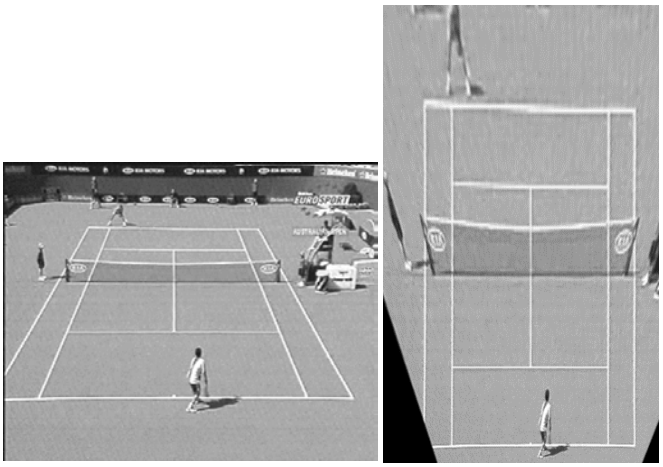
Ferdi van der Heijden ▸ University of Twente - RAM ▸ 4/24/2019

Contents

Camera models

3D Computer Vision

1 Introduction



The picture on the left is a perspective projection of a 3D scene. The right image is a geometrical transform of the left that simulates a rotation of the camera such that the camera points downwards. Such a transform is based on a camera model.

In the transform from the left image to right one, Euclidean properties are not preserved:

- Distances between corners in the first image are not equal to distance between these corners in the second image, not even after scaling.
- Angles between lines in the first are not equal to angles between the corresponding lines in the other image.
- Lines that are parallel in one image are not necessarily parallel in the other image.

Yet, some other properties are still invariant in both images. For instance, some lines in the left image have a common intersection point, the so-called *vanishing point*. The corresponding lines in the right image are parallel. These lines are said to have a common vanishing point at an infinite distance, a so-called *point at infinity*. Apparently, there is a property that says that if lines in one image share a vanishing point, then corresponding lines in the other image will also have a vanishing point. These issues are studied in *projective geometry*: a mathematical branch that studies properties that are invariant under a projection.

We will use projective geometry to geometrically model the mapping of a 3D point in a scene to a 2D point on the image plane of an optical camera. It will provide us with means to calibrate such a camera, and to establish geometrical properties if we have two cameras (stereo vision) or if we have a moving camera that is used for visual navigation.

In this chapter, we first recapitulate geometrical transforms, and apply these to describe nonlinear lens distortion. Next, we continue with homographies (projective transforms) and the usage of homogeneous coordinates. Then, we introduce the camera model. We end the chapter with discussing principles for camera calibration.

2 2D geometrical transforms

A geometrical transform is given by:

$$\begin{aligned} u &= r(x,y) \\ v &= s(x,y) \end{aligned} \tag{1}$$

Here (x,y) are the coordinates in the source domain, and (u,v) the coordinates in the destination domain. An input image $f(x,y)$ is geometrically transformed to an output image $g(u,v)$ by:

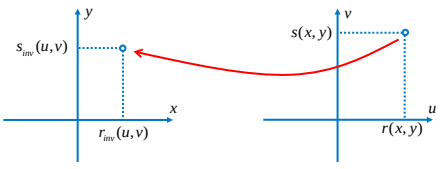
$$g(u,v) = g(r(x,y),s(x,y)) = f(x,y) \tag{2}$$

This is the forward transform. For a proper geometrical transform the reverse is also possible. The transformation $r(x,y)$ and $s(x,y)$ must be such that if eq (1) is true, then functions $r_{inv}(.,.)$ and $s_{inv}(.,.)$ exist such that:

$$\begin{aligned} x &= r_{inv}(u,v) \\ y &= s_{inv}(u,v) \end{aligned} \tag{3}$$

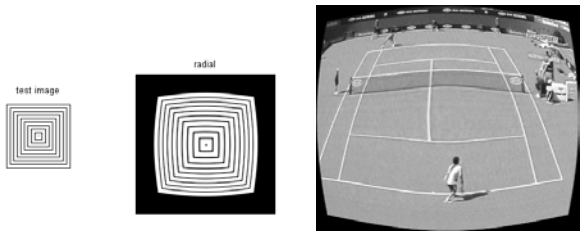
and the transform can be written as:

$$g(u,v) = f(r_{inv}(u,v), s_{inv}(u,v)) \tag{4}$$



2.1 Nonlinear lens distortion

Nonlinear lens distortion is a typical example of a geometrical transform:



The example in the figure above shows the simplest model for lens distortion. It is called *radial distortion* as the distorting terms only depend on $r = \sqrt{x^2 + y^2}$. Here, x and y are the undistorted normalized camera coordinates (to be defined later).

This distortion model has only one parameter. Other models have more parameters:

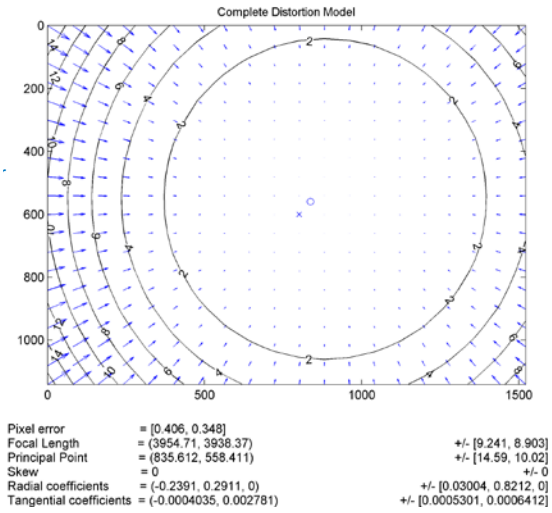
$$\begin{aligned} u &= x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ v &= y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \end{aligned} \tag{5}$$

Radial distortion occurs due to lens imperfections. Another type of distortion is tangential distortion:

$$\begin{aligned} u &= x + \left(2 p_1 xy + p_2 \left(r^2 + 2x^2\right)\right) \\ v &= y + \left(2 p_2 xy + p_1 \left(r^2 + 2y^2\right)\right) \end{aligned} \tag{6}$$

Tangential distortion occurs if the lens is not mounted fully parallel to the image plane.

During camera calibration the lens distortion parameters are estimated. An example of this is shown below. With the deformation parameters, the nonlinear distortion can be compensated, so that in further processing its influence is neutralized.



3 Homogeneous coordinates and the homography

Two coordinates, (x, y), suffice to represent a point in a 2D Euclidian space. However, in a *projective space*, a 2D point is represented by three coordinates, the so-called *homogeneous* representation. This representation is $(\alpha x, \alpha y, \alpha)$. It is obtained by introducing the auxiliary variable α. This variable can be chosen freely as long as $\alpha \neq 0$. Often, the choice is $\alpha = 1$.

The 2D point can also be represented by vectors, i.e.:

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} \quad \text{and} \quad \underline{\mathbf{x}} = \begin{bmatrix} \alpha x \\ \alpha y \\ \alpha \end{bmatrix} \tag{7}$$

The notation $\underline{\mathbf{x}}$ indicates here that the vector $\mathbf{x}$ is in the homogeneous form[1].

It is easy to recognize the following two properties:

1) The set of allowable vectors $\underline{\mathbf{x}}$ is given by: $\mathbb{R}^3 - \mathbf{0}$ (the 3D real space with the origin excluded)
2) Two points that differ only by a scaling constant α are equivalent: (8)

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} \equiv \alpha \begin{bmatrix} a \\ b \\ c \end{bmatrix} \quad \text{for any } \alpha \neq 0$$

The set of all vectors that satisfy these two properties is called the *2D projective space*.

Generalization to 3D

The *3D projective space* consists of all vectors in $\mathbb{R}^4$ with the property that all points that differ only by a scaling parameter are equivalent. As before, the null vector $\mathbf{0}$ is excluded, The vector $\underline{\mathbf{x}} = \begin{bmatrix} a & b & c & w \end{bmatrix}^T$ represents a point

$$\mathbf{x} = \frac{1}{w} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{9}$$

in $\mathbb{R}^3$.

[1] Formally, $\underline{\mathbf{x}}$ cannot be called a vector in the sense that it is an element from a linear vector space. By definition, linear vector spaces always have a null vector, but in a projective space this null vector is excluded.

3.1 Homographies

The advantage of homogeneous representations is that a so-called *projective transform* can be written as a matrix-vector product:

$$\underline{\mathbf{u}} = \mathbf{H}\underline{\mathbf{x}} \tag{10}$$

$\mathbf{H}$ is a 3×3 matrix containing the parameters of the projective transform:

$$\mathbf{H} = \begin{bmatrix} A & B & C \\ D & E & F \\ G & H & 1 \end{bmatrix} \tag{11}$$

To see which geometrical transform corresponds to this operation, expand (7) and (10):

$$\underline{\mathbf{u}} = \begin{bmatrix} A & B & C \\ D & E & F \\ G & H & 1 \end{bmatrix} \begin{bmatrix} \alpha x \\ \alpha y \\ \alpha \end{bmatrix} = \alpha \begin{bmatrix} Ax+By+C \\ Dx+Ey+F \\ Gx+Hy+1 \end{bmatrix} \tag{12}$$

which after inhomogenizing yields:

$$\begin{aligned} u &= \frac{Ax+By+C}{Gx+Hy+1} \\ v &= \frac{Dx+Ey+F}{Gx+Hy+1} \end{aligned} \tag{13}$$

The transform defined by the matrix $\mathbf{H}$ is called a *homography*, and effectively implements a projective transform as a linear operator. With properly chosen parameters, i.e. $G = H = 0$, it also implements an affine transform:

$$\begin{aligned} u &= Ax+By+C \\ v &= Dx+Ey+F \end{aligned} \tag{14}$$

With properly chosen parameters, it equals the similarity transform (scaling, rotation and translation:

$$\begin{aligned} u &= s\cos(\phi)\, x - s\sin(\phi)\, y + a \\ v &= s\sin(\phi)\, x + s\cos(\phi)\, y + b \end{aligned} \tag{15}$$

The homography is invariant to a change of scale. That is: $\beta\mathbf{H}\underline{\mathbf{x}} = \mathbf{H}\underline{\mathbf{x}}$ for any $\beta \neq 0$. Thus, $\beta\mathbf{H} \equiv \mathbf{H}$ for all $\beta \neq 0$.

The inverse transform of a homography is simply obtained by matrix inversion:

$$\underline{\mathbf{x}} = \mathbf{H}^{-1}\underline{\mathbf{u}} \tag{16}$$

3.2 Homogeneous representation of lines

An arbitrary line in 2D is defined by the equation:

$$ax + by + c = 0 \tag{17}$$

The three parameters of a line can be represented by a vector: $\underline{\mathbf{l}} = \begin{bmatrix} a & b & c \end{bmatrix}^T$. The set of all possible lines forms a projective space as it exhibits the same two properties as homogeneous points:

1) $\begin{bmatrix} a \\ b \\ c \end{bmatrix} \in \mathbb{R}^3 - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$ (each point in the 3D real space represents a line; except the origin) (18)

2) $\begin{bmatrix} a \\ b \\ c \end{bmatrix} \equiv \alpha \begin{bmatrix} a \\ b \\ c \end{bmatrix} \quad \forall \alpha \neq 0$ (two points that differ only by a scaling constant are equivalent)

A point lying on a line

In 2D, a point $\mathbf{x} = \begin{bmatrix} x & y \end{bmatrix}^T$ lies on a line if eq. (17) is satisfied. If the point is expressed in homogeneous coordinates: $\underline{\mathbf{x}} = \begin{bmatrix} \alpha x & \alpha y & \alpha \end{bmatrix}^T$, the inner product between the line and the point can be defined:

$$\underline{\mathbf{l}}^T\underline{\mathbf{x}} = \underline{\mathbf{x}}^T\underline{\mathbf{l}} = a\alpha x + b\alpha y + \alpha c \tag{19}$$

Thus the requirement for $\underline{\mathbf{x}}$ lying on $\underline{\mathbf{l}}$ is $\underline{\mathbf{l}}^T\underline{\mathbf{x}} = 0$.

The intersection of two lines

Suppose that we have two lines, $\underline{\mathbf{l}}_1 = \begin{bmatrix} a_1 & b_1 & c_1 \end{bmatrix}^T$ and $\underline{\mathbf{l}}_2 = \begin{bmatrix} a_2 & b_2 & c_2 \end{bmatrix}^T$. We wish to find their intersection point $\mathbf{v}$. Define a vector $\underline{\mathbf{v}} = \underline{\mathbf{l}}_1 \times \underline{\mathbf{l}}_2$ which is the cross product of the two lines. This vector is orthogonal to both $\underline{\mathbf{l}}_1$ and $\underline{\mathbf{l}}_2$. Therefore, $\underline{\mathbf{l}}_1\underline{\mathbf{v}}^T = 0$ and $\underline{\mathbf{l}}_2\underline{\mathbf{v}}^T = 0$. Consequently, $\underline{\mathbf{v}}$, regarded as an homogeneous point, lies both on $\underline{\mathbf{l}}_1$ and $\underline{\mathbf{l}}_2$. In other words, $\underline{\mathbf{v}}$ is the intersection point.

A line joining two points

To find a line $\underline{\mathbf{l}}$ that joins two given points $\underline{\mathbf{a}}$ and $\underline{\mathbf{b}}$ we can apply the same line of reasoning: that line is given by the cross product of the two points: $\underline{\mathbf{l}} = \underline{\mathbf{a}} \times \underline{\mathbf{b}}$.

Planes in 3D

Generalizing this to 3D, a vector $\underline{\mathbf{x}} = \begin{bmatrix} a & b & c & w \end{bmatrix}^T$ could represent a plane:

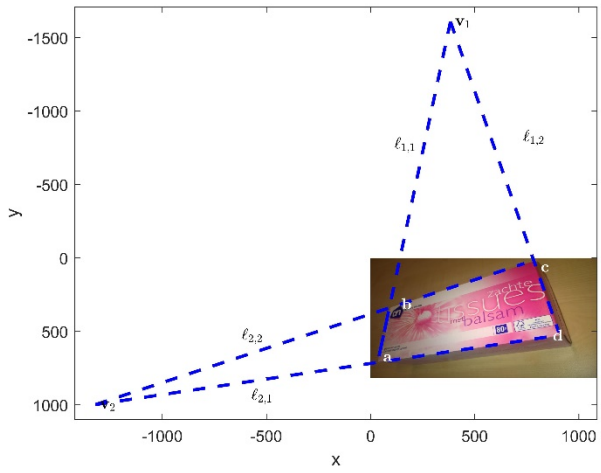
$$ax + by + cz + w = 0 \tag{20}$$

in the 3D space.

3.3 Rectifying perspective distortions

Consider the image below:

In 3D, the top of the box forms a plane. In the image, this plane is distorted due to the perspective projection. Parallel lines in the real world are now lines with a common intersection point; the *vanishing point*. The figure below shows the two vanishing points $\underline{\mathbf{v}}_1$ and $\underline{\mathbf{v}}_2$ of the top of the box. They are obtained in the way described above: two points define a line. Two intersecting lines define the vanishing point.



Rectifying an image means applying a projective transform such that the similarity properties are restored: 1) Parallel lines in the real world become parallel in the image. 2) Angles in the 3D plane are identical to angles in the image. 3) The ratio between distances in the 3D plane equals the ratio of these distances in the image.

To accomplish rectification, the two vanishing points must be shifted to points that are located at infinity in the horizontal and vertical directions, respectively. In homogeneous coordinates these new points are represented by $\underline{\mathbf{u}}_1 = \begin{bmatrix}1 & 0 & 0\end{bmatrix}^{\mathrm{T}}$ and $\underline{\mathbf{u}}_2 = \begin{bmatrix}0 & 1 & 0\end{bmatrix}^{\mathrm{T}}$,

To rectify the image, we must find a homography $\mathbf{H}$ such that:

$$\underline{\mathbf{u}}_1 = \mathbf{H}\underline{\mathbf{v}}_1 \qquad \underline{\mathbf{u}}_2 = \mathbf{H}\underline{\mathbf{v}}_2 \tag{21}$$

These are not sufficient conditions to find $\mathbf{H}$ as we can freely scale and shift a rectified image and still satisfy eq. (21). Therefore, in addition, we want to require that the origin will not shift. This leads to the additional requirement:

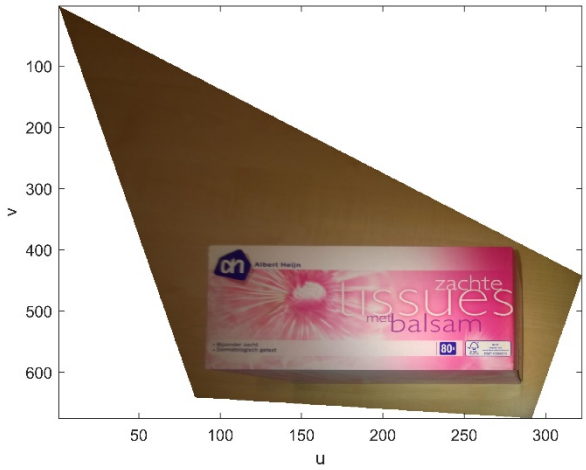
$$\begin{bmatrix}0\\0\\\beta\end{bmatrix} = \mathbf{H}\begin{bmatrix}0\\0\\1\end{bmatrix} \tag{22}$$

Expanding eq (21) and (22) and writing the matrix $\mathbf{H}$ as a vector, see eq (12), yield:

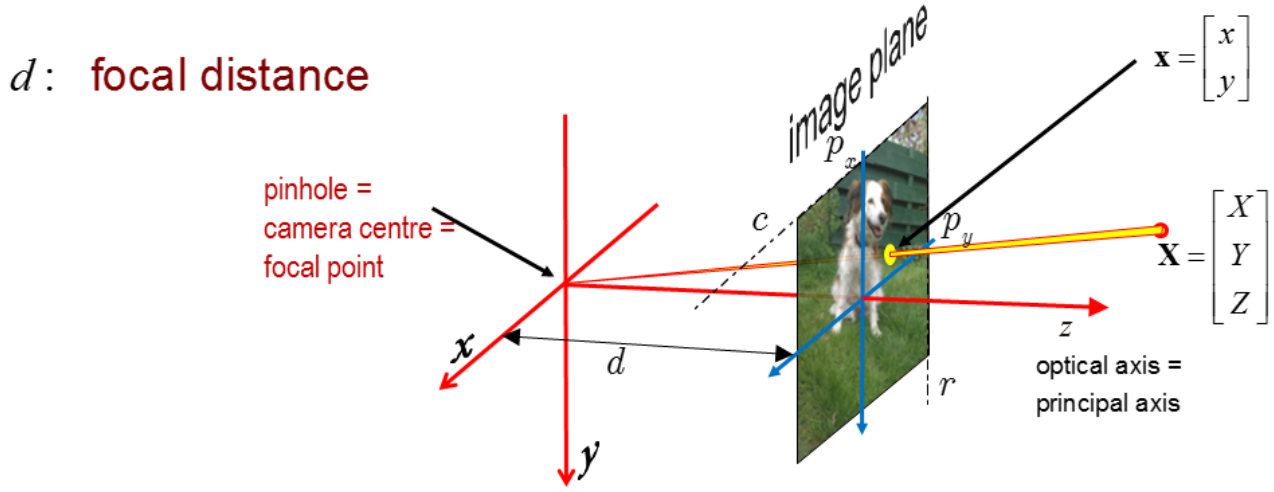
$$\begin{bmatrix}
v_{11} & v_{12} & v_{13} & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & v_{11} & v_{12} & v_{13} & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & v_{11} & v_{12} & v_{13}\\
v_{21} & v_{22} & v_{23} & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & v_{21} & v_{22} & v_{23} & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & v_{21} & v_{22} & v_{23}\\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}A\\B\\C\\D\\E\\F\\G\\H\\I\end{bmatrix}
=
\begin{bmatrix}\underline{\mathbf{u}}_1\\\\\underline{\mathbf{u}}_2\\\\0\\0\\\beta\end{bmatrix}$$

with $\underline{\mathbf{v}}_1 = \begin{bmatrix}v_{11} & v_{12} & v_{13}\end{bmatrix}^{\mathrm{T}}$ and $\underline{\mathbf{v}}_2 = \begin{bmatrix}v_{21} & v_{22} & v_{23}\end{bmatrix}^{\mathrm{T}}$ (23)

Solving this linear equation by inverting the 9x9 matrix provides us with a solution for $\mathbf{H}$. The parameter β controls the spatial scaling of the resulting rectified image. The figure below shows a result of rectification.



There is still an ambiguity in the solution of eq (23) since the point at infinity $\underline{\mathbf{u}}_1 = \begin{bmatrix}1 & 0 & 0\end{bmatrix}^{\mathrm{T}}$ is essentially equivalent to the point $\begin{bmatrix}-1 & 0 & 0\end{bmatrix}^{\mathrm{T}}$, Substitution of this last one in eq. (23) leads to another solution, namely the version of the



perspective projection:

$$c = \frac{Xd}{Z\Delta} + p_x \qquad r = \frac{Yd}{Z\Delta} + p_y$$

(r,c): Pixel coordinates or image coordinates
p_x, p_y: Principal point = image center
Δ: Pitch (distance between pixels)
Often, d is expressed in Δ, so that $\Delta = 1$

rectified image that is mirrored along the y-axis. The same holds for the point $\underline{\mathbf{u}}_2$. Altogether there are four different solutions. These correspond to the four ways that the image can be mirrored along the two axes. To get a non-mirrored solution, one has to select the appropriate signs in the points at infinity.

Another unresolved issue is the ratio between spatial scaling factors of the horizontal and vertical axes. This ratio is influenced by our choices of $\underline{\mathbf{u}}_1$ and $\underline{\mathbf{u}}_2$. For instance, by choosing $\underline{\mathbf{u}}_1 = [\alpha \quad 0 \quad 0]^T$ the scaling factor of the y-axis is controlled by the parameter α. We need some reference to obtain the correct factor α for getting a true similarity.

4 The pinhole camera model

The figure above presents an overview of the so-called pinhole model of an optical camera. The model is built in a 3D frame called the camera coordinate system (CCS). The origin of this frame is the *focal point* of the camera, which is also called the *camera center* or the *pinhole*. The so-called camera coordinates are denoted by (x, y, z). By convention, the z-axis is always the optical axis. Usually, the x-axis is the horizontal axis. That is, the direction of a row of pixels in the image. The y-axis is the vertical axis, i.e. the direction of a column.

At a distance d, the *focal distance*, the z-axis intersects the image plane. This intersection is orthogonally, which implies that the image plane is parallel with the xy-plane. Hence locating a 2D point in the image plane is done by (x, y). The intersection point of the image plane and the z-axis, is a 3D point with coordinates $(0,0,d)$. Regarded as a 2D point in the image plane, it has coordinates $(0,0)$. This point is called the *image center*, also called *principal point*.

According to the pinhole model, a point light source at a position (X,Y,Z) in front of the camera will be imaged at a point in the image plane at the intersection of a ray from (X,Y,Z) to the origin $(0,0,0)$. This ray is given by $\alpha[X \quad Y \quad Z]^T$ with $\alpha \in \mathbb{R}$. To find the intersection point, we have to equate:

$$\alpha \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} x \\ y \\ d \end{bmatrix} \tag{24}$$

From which the equations of a perspective projection follows:

$$x = \frac{Xd}{Z} \qquad y = \frac{Yd}{Z} \tag{25}$$

The next step is to introduce the sampling grid of the image. By convention, the x- and y-axes are aligned with the rows

and the columns of the image grid. This grid is indexed by row- and column subscripts (r,c). These are integers ranging from $r=1,\cdots,N_{row}$ and $c=1,\cdots,N_{col}$. The system (r,c) is called the *pixel subscripts* or *pixel indices*. Note the change in order: r is associated with y, and c with x. So, actually a notation (c,r) would be more logical. Despite of that, the usual order is (r,c) which stems from conventions in matrix algebra.

The relation between camera coordinates and pixel subscripts is:

$$\begin{aligned} x &= (c-p_x)\Delta \\ y &= (r-p_y)\Delta \end{aligned} \tag{26}$$

Here, (p_x,p_y) are the pixel coordinates of the principal point. Δ is the distance between neighboring pixels, the so-called *pitch*.

The back transformation involves a round-off procedure since pixel subscripts are integers:

$$\begin{aligned} r &= round\left(\frac{y}{\Delta}+p_y\right) \\ c &= round\left(\frac{x}{\Delta}+p_x\right) \end{aligned} \tag{27}$$

If we omit the round-off, the results are no longer (integer) indices. Nonetheless, this is useful. Without round-off, (c,r) are called *intrinsic coordinates* (note the order), also called *image coordinates* or *pixel coordinates*, denoted by $\mathbf{p}=\begin{bmatrix} c & r \end{bmatrix}^T$.

Substitution of (26) in (25) yields:

$$\begin{aligned} c &= \frac{Xd}{Z\Delta}+p_x \\ r &= \frac{Yd}{Z\Delta}+p_y \end{aligned} \tag{28}$$

Note 1:
We silently assumed that the pitch in the x-direction equals the one in the y-direction. If this is not the case, we have to introduce Δ_x and Δ_y.

Note 2:
Usually, the focal distance d is expressed in the unit Δ. For instance, $d=1000\ (\Delta)$ means that the focal distance is 1000Δ. With this convention, the Δ in the numerator and the Δ in the denominator cancel, If $\Delta_x \neq \Delta_y$, then eq (28) simplifies to:

$$\begin{aligned} c &= \frac{Xd_x}{Z}+p_x \\ r &= \frac{Yd_y}{Z}+p_y \end{aligned} \tag{29}$$

in which $d_x = d\ (\Delta_x)$ and $d_y = d\ (\Delta_y)$ are the focal distances expressed in the units Δ_x and Δ_y.

4.1 Normalized camera coordinates

We may map the camera coordinates in the image, i.e. (x,y), to normalized coordinates (x_n,y_n). These normalized coordinates are the coordinates at the virtual image plane that is at a distance of $d_n=1\ (\Delta)$. According to (25), the normalization is obtained by:

$$\begin{aligned} x_n &= \frac{x}{d_x} \\ y_n &= \frac{y}{d_y} \end{aligned} \tag{30}$$

Consequently, the 3D representation of a point in this virtual image plane is: $(x_n,y_n,1)$.

Substitution of (30) in (25) yields an interesting result:

$$\begin{aligned} x_n &= \frac{X}{Z} \\ y_n &= \frac{Y}{Z} \end{aligned} \tag{31}$$

In fact, if we regard the vector $\mathbf{X}=\begin{bmatrix} X & Y & Z \end{bmatrix}^T$ as a homogeneous coordinates of a 2D point, then

$$\mathbf{x}_n = \begin{bmatrix} X/Z \\ Y/Z \end{bmatrix} \tag{32}$$

is the de-homogenized version of $\mathbf{X}$.

4.2 Intrinsic camera parameters

We consider the 3D point $\mathbf{X}=\begin{bmatrix} X & Y & Z \end{bmatrix}^T$ again, and its homogenized version $\underline{\mathbf{X}}=\begin{bmatrix} \alpha X & \alpha Y & \alpha Z & \alpha \end{bmatrix}^T$ in the 3D projective space. Equation (29) can then be expressed in matrix-vector form:

$$\underline{\mathbf{p}} = \mathbf{K}\begin{bmatrix} \mathbf{I}_{3\times 3} & \mathbf{0}_{3\times 1} \end{bmatrix}\underline{\mathbf{X}} \tag{33}$$

Here, $\mathbf{I}_{3\times 3}$ is the 3×3 unity matrix, and $\mathbf{0}_{3\times 1}$ is a 3×1 matrix filled with zeroes. In other words:

$$\begin{bmatrix}\mathbf{I}_{3\times3} & \mathbf{0}_{3\times1}\end{bmatrix} = \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right] \tag{34}$$

The matrix $\mathbf{K}$ is the *calibration matrix* defined as:

$$\mathbf{K} = \begin{bmatrix} d_x & 0 & p_x \\ 0 & d_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{35}$$

To proof (33), expand the equation:

$$\begin{bmatrix}\mathbf{I}_{3\times3} & \mathbf{0}_{3\times1}\end{bmatrix}\begin{bmatrix} \alpha X \\ \alpha Y \\ \alpha Z \\ \alpha \end{bmatrix} = \begin{bmatrix} \alpha X \\ \alpha Y \\ \alpha Z \end{bmatrix}$$

So:

$$\mathbf{K}\begin{bmatrix}\mathbf{I}_{3\times3} & \mathbf{0}_{3\times1}\end{bmatrix}\underline{\mathbf{X}} = \mathbf{K}\begin{bmatrix} \alpha X \\ \alpha Y \\ \alpha Z \end{bmatrix} \tag{36}$$

However:

$$\mathbf{K}\begin{bmatrix} \alpha X \\ \alpha Y \\ \alpha Z \end{bmatrix} = \begin{bmatrix} d_x & 0 & p_x \\ 0 & d_y & p_y \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \alpha X \\ \alpha Y \\ \alpha Z \end{bmatrix} = \begin{bmatrix} \alpha d_x X + \alpha p_x Z \\ \alpha d_y Y + \alpha p_y Z \\ \alpha Z \end{bmatrix}$$

which is actually $\underline{\mathbf{p}}$

After de-homogenizing we get:

$$\mathbf{p} = \begin{bmatrix} \frac{d_x X}{Z} + p_x \\ \frac{d_y Y}{Z} + p_y \end{bmatrix} \tag{37}$$

This is fully in accordance with (29). The advantage of (33) is that we now have a linear equation which relates the 3D point in space with the 2D point on the image plane.

4.3 Virtual rotation of a camera

If we homogenize the intrinsic coordinates $\mathbf{p}$ by defining $\underline{\mathbf{p}} = \begin{bmatrix} c & r & 1 \end{bmatrix}^{\mathrm{T}}$ we can calculate:

$$\mathbf{K}^{-1}\underline{\mathbf{p}} = \begin{bmatrix} X/Z \\ Y/Z \\ 1 \end{bmatrix} \tag{38}$$

These are, in fact, the 3D coordinates of the original 3D point, projected on the virtual image plane with $d = 1$. Suppose that we virtually rotate the camera. The rotation is mathematically represented by a 3×3 rotation matrix $\mathbf{R}$ that transforms the original coordinates $\mathbf{X}$ such that in the new CCS (after rotation) they are represented by $\mathbf{RX}$. If we apply this rotation to the points in the virtual image plane, then the result, expressed in the new CCS, will be $\mathbf{RK}^{-1}\underline{\mathbf{p}}$. If finally these points are projected on the rotated image plane, the coordinates become $\mathbf{KRK}^{-1}\underline{\mathbf{p}}$. Therefore, a rotation of a camera can be simulated on an observed image by applying a projective transform with an homography:

$$\mathbf{H} = \mathbf{KRK}^{-1} \tag{39}$$

4.4 Extrinsic camera calibration parameters

Up to now, the discussion was limited to just one 3D frame: the CCS. Cameras are situated in the real world which may have its own frame: the world coordinate system (WCS). To distinguish points that are represented in one frame from representations in another frame, we introduce the notation ${}^{c}\mathbf{X}$ (CCS) and ${}^{w}\mathbf{X}$ (WCS).

The transformation from one representation to another involves a rotation and a translation. Suppose that ${}^{c}\mathbf{R}_{w}$ is the rotation matrix that aligns the WCS to the CCS, and that ${}^{c}\mathbf{t}_{w}$ is the origin of the WCS expressed in CCS. Then the conversion is given by:

$${}^{c}\mathbf{X} = {}^{c}\mathbf{R}_{w}\,{}^{w}\mathbf{X} + {}^{c}\mathbf{t}_{w} \tag{40}$$

In homogeneous coordinates, this transformation simplifies to a matrix-vector multiplication. Define the following 4×4 transformation matrix:

$${}^{c}\mathbf{T}_{w} = \left[\begin{array}{ccc|c} & {}^{c}\mathbf{R}_{w} & & {}^{c}\mathbf{t}_{w} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{41}$$

The conversion becomes:

$${}^{c}\underline{\mathbf{X}} = {}^{c}\mathbf{T}_{w}\,{}^{w}\underline{\mathbf{X}} \tag{42}$$

Wrapping up the results, we have:

$$\begin{aligned}\underline{\mathbf{p}} &= \mathbf{K}\begin{bmatrix}\mathbf{I}_{3\times3} & \mathbf{0}_{3\times1}\end{bmatrix}\left[\begin{array}{ccc|c} & {}^{c}\mathbf{R}_{w} & & {}^{c}\mathbf{t}_{w} \\ \hline 0 & 0 & 0 & 1 \end{array}\right]{}^{w}\underline{\mathbf{X}} \\ &= \mathbf{K}\begin{bmatrix} {}^{c}\mathbf{R}_{w} & {}^{c}\mathbf{t}_{w}\end{bmatrix}{}^{w}\underline{\mathbf{X}}\end{aligned} \tag{43}$$

The four parameters inside the calibration matrix $\mathbf{K}$, together with the parameters for nonlinear lens distortion (two to four parameters), are called the *intrinsic camera*

parameters. The rotation matrix and translation vector are called the *extrinsic camera parameters*. Since the degrees of freedom of a rotation matrix is three, e.g. the Euler angles, there are six extrinsic parameters. In full, we have twelve to fifteen calibration parameters.

5 Camera calibration

Camera calibration is the estimation of the camera parameters, intrinsic and extrinsic, using images from a scene with reference objects. The most well-known procedure uses a checker board as reference.

Ignoring for a moment the nonlinear lens distortion, the model that relates 3D points in the scene with 2D points in the image is given by (43) which we rewrite in:

$$\underline{\mathbf{p}} = \mathbf{M}\,{}^{w}\underline{\mathbf{X}} \quad \text{with} \quad \mathbf{M} = \mathbf{K}\begin{bmatrix} {}^{c}\mathbf{R}_{w} & {}^{c}\mathbf{t}_{w} \end{bmatrix} \tag{44}$$

The matrix $\mathbf{M}$ is a 3x4 matrix with elements m_{ij}. This matrix is the so-called *perspective projection matrix*.

The estimation is done in three steps. First, the matrix $\mathbf{M}$ is estimated using the knowledge of $\underline{\mathbf{p}}$ and ${}^{w}\underline{\mathbf{X}}$ (being the reference points in the reference object). The second step is to retrieve the camera parameters from $\mathbf{M}$. The last step is a refinement step in which also the lens distortion parameters are estimated.

5.1 Estimation of the perspective projection matrix

In the first step, we assume that we have knowledge of N reference points in the 3D scene. These points are denoted in homogeneous coordinates by ${}^{w}\underline{\mathbf{X}}_n = \begin{bmatrix} X_n & Y_n & Z_n & 1 \end{bmatrix}^{\mathrm{T}}$ with $n = 1, \cdots, N$. We also assume that the corresponding image points $\underline{\mathbf{p}}_n = \alpha_n \begin{bmatrix} c_n & r_n & 1 \end{bmatrix}^{\mathrm{T}}$ can be found either by manual pinpointing, or by automatic image processing.

For each point pair (${}^{w}\underline{\mathbf{X}}_n$, $\underline{\mathbf{p}}_n$) eq (44) applies:

$$\alpha_n \begin{bmatrix} c_n \\ r_n \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X_n \\ Y_n \\ Z_n \\ 1 \end{bmatrix} \tag{45}$$

From the last row we solve α_n:

$$\alpha_n = m_{31}X_n + m_{32}Y_n + m_{33}Z_n + m_{34} \tag{46}$$

Substitution in the first two rows yields two equations that can be written as follows:

$$\begin{bmatrix} X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -c_nX_n & -c_nY_n & -c_nZ_n & -c_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -r_nX_n & -r_nY_n & -r_nZ_n & -r_n \end{bmatrix} \mathbf{m} = \mathbf{0} \tag{47}$$

where $\mathbf{m} = \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{34} \end{bmatrix}^{\mathrm{T}}$. For N point pairs, we have

$$\mathbf{G}\mathbf{m} = \mathbf{0} \tag{48}$$

The matrix $\mathbf{G}$ is a $2N \times 12$ matrix that for each point pair contains 2 rows like in eq (47). Since the vecrtor $\mathbf{m}$ has 12 elements m_{ij}, and each point gives 2 equations, we need at least $N \geq 6$ point pairs.

To solve eq (48) we need a constraint, like $\|\mathbf{m}\| = 1$ to escape from the trivial solution $\mathbf{m} = \mathbf{0}$. The optimization problem can then be formulated as:

$$\arg\min_{\mathbf{m}} \|\mathbf{G}\mathbf{m}\|^2 \quad \text{subject to} \quad \|\mathbf{m}\| = 1 \tag{49}$$

That is, find the $\mathbf{m}$ that minimizes the squared error $\|\mathbf{G}\mathbf{m}\|^2$ under the constraint that $\|\mathbf{m}\|$ =1. Singular Value decomposition provides the solution for such a problem. Simply stated: $\mathbf{m}$ is the eigenvector of $\mathbf{G}^T\mathbf{G}$ associated with the smallest eigenvalue. From that, $\mathbf{M}$ follows.

5.2 Recovering the camera parameters

Once $\mathbf{m}$, and thus $\mathbf{M}$, has been estimated, the intrinsic parameters are estimated as follows. Define $\mathbf{M} = \begin{bmatrix} \mathbf{m}_1 & \mathbf{m}_2 & \mathbf{m}_3 & \mathbf{m}_4 \end{bmatrix}$. That is, $\mathbf{m}_i$ is the i th column from $\mathbf{M}$. Also define $\mathbf{A} = \begin{bmatrix} \mathbf{m}_1 & \mathbf{m}_2 & \mathbf{m}_3 \end{bmatrix} = \mathbf{K}\,{}^{c}\mathbf{R}_{w}$. From (44):

$$\mathbf{A}\mathbf{A}^T = \mathbf{K}\,{}^{c}\mathbf{R}_{w}\left(\mathbf{K}\,{}^{c}\mathbf{R}_{w}\right)^T = \mathbf{K}\,{}^{c}\mathbf{R}_{w}\,{}^{c}\mathbf{R}_{w}^T\mathbf{K}^T = \mathbf{K}\mathbf{K}^T \tag{50}$$

Defining $\mathbf{B} = \mathbf{A}\mathbf{A}^T$ and substitution of (35) yield:

$$\mathbf{B} = \begin{bmatrix} d_x^2 + p_x^2 & p_x p_y & p_x \\ p_x p_y & d_y^2 + p_y^2 & p_y \\ p_x & p_y & 1 \end{bmatrix} \tag{51}$$

From here, we conclude that:

$$\begin{aligned} p_x &= b_{13} \\ p_y &= b_{23} \\ d_x &= \sqrt{b_{11} - b_{13}^2} \\ d_y &= \sqrt{b_{22} - b_{23}^2} \end{aligned} \tag{52}$$

Because $\mathbf{M}$ is defined up to a scale factor, the last element of $\mathbf{B}$, i.e. b_{33} is usually not equal to 1, so we have to normalize $\mathbf{B}$ such that $b_{33}=1$.

The next step is to find the extrinsic parameters. These follow readily from:

$$\begin{aligned} {}^{c}\mathbf{R}_{w} &= \mathbf{K}^{-1}\mathbf{A} \\ {}^{c}\mathbf{t}_{w} &= \mathbf{K}^{-1}\mathbf{m}_{4} \end{aligned} \tag{53}$$

5.3 Fine tuning

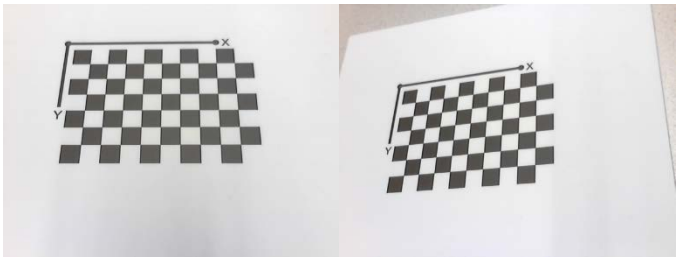
The procedure described above is sensitive to noise and it does not include the lens distortion parameters. The results can be refined by a numerical optimization procedure. Suppose that all the camera parameters are stacked in a vector $\boldsymbol{\alpha}$. If all these parameters were known, then application of the camera model to a 3D point in space, i.e. conversion from world coordinates (42), lens distortion (5) and (6), and projection (29) would predict the pixel coordinates of the reference 3D points. Denote this prediction by $\hat{\mathbf{p}}_n(\boldsymbol{\alpha})$. These predictions are called the *reprojected points*. Our goal would then be to minimize the difference between observed pixel positions $\mathbf{p}_n$ and the reprojected one. A suitable criterion for that is the mean square of the differences:

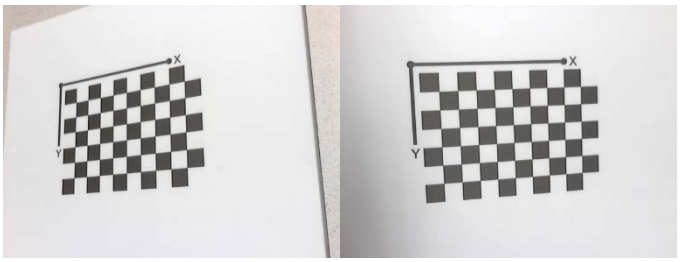
$$J(\boldsymbol{\alpha}) = \frac{1}{N}\sum_{n=1}^{N}\left\|\mathbf{p}_n - \hat{\mathbf{p}}_n(\boldsymbol{\alpha})\right\|^2 \tag{54}$$

The final estimate is the $\boldsymbol{\alpha}$ that minimizes this criterion. It is a nonlinear problem that must be solved numerically. The solution from the previous section serves as a good initial guess. This greatly facilitates the numerical optimization.

5.4 Calibration according to Zhang and Bouguet

A popular calibration object is the chess board. This is proposed by Zhang [1]. A Matlab implementation has been provided by Bouget [2], and this has been embedded in the computer vision toolbox of Matlab.





The chess board defines the world coordinate system. The corners of the fields, which are easily found and localized in an image, are the reference points with known positions in the WCS. However, all the 3D reference points are located in just one plane, the $Z=0$ plane. The geometrical relation between that plane and the image plane is a homography. As we know from Section 3.1, a homography is defined by 9 parameters, but it is invariant to scaling. So actually, the degrees of freedom is 8. The perspective projection matrix $\mathbf{M}$ contains 12 parameters. As a result, the matrix $\mathbf{G}^T\mathbf{G}$ in Section 5.1 will possess 4 eigenvalues that are zero. The solution space is not unique.

Zhang [2] proposed to use multiple images of the same chess board taken from multiple viewing angles. He developed a method, analogously to the approach described in section 5.1 up to 5.3 to find the intrinsic parameters, and for each image the extrinsic parameters.

A further remark is that the camera model might be extended by a *shear* constant s. The calibration matrix is then defined by:

$$\mathbf{K} = \begin{bmatrix} d_x & s & p_x \\ 0 & d_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{55}$$

This model takes into account the possible shearing of the grid of pixels. This may happen during the fabrication of the image sensor. However, often the shear constant is negligible.

6 Some useful functions in Matlab

The computer vision toolbox of Matlab provides the software of Bouguet in an easy-to-use app. The output of this app is a Matlab object from the class `cameraParameters` that contains all the relevant parameters and functions. The house-made function `ut_plot_lens_distortion`, which is a modification of one of the functions from the Bouguet's toolbox, provides a visualization of the lens distortion.

Geometrical transforms are implemented in two different toolboxes. The image processing toolbox supports the function `imtransform` with a number of supporting function. However, this set of functions has become obsolete with the introduction a new function called `imwarp`, with the introduction of the computer vision toolbox that contains supporting functions and classes. One of the supporting classes is `imref2D` which is helpful to set up the relation between pixel coordinates and camera coordinates.

References

Further reading:
A much more elaborated treatment of projective spaces is given in [3]. This textbook also contains chapters on camera models. A general outline of the camera calibration is given in [4].

1. Zhang, Z., *A Flexible New Technique for Camera Calibration.* IEEE Transaction on PAMI, 2000. **22**(11): p. 1330-1334.
2. Bouguet, J. *Camera Calibration Toolbox for Matlab*. [cited 2016 09/10/2016]; Available from: https://www.vision.caltech.edu/bouguetj/calib_doc/.
3. Hartley, R. and A. Zisserman, *Multiple View Geometry in Computer Vision*. 2003: Cambridge University Press. 700.
4. Zhang, Z., *Camera Calibration*, in *Emerging Topics in Computer Vision*, G. Medioni and S.B. Kang, Editors. 2004, Prentice Hall.