Projective geometry and camera calibration

Wlodzimierz Kasprzak, Artur Wilkowski

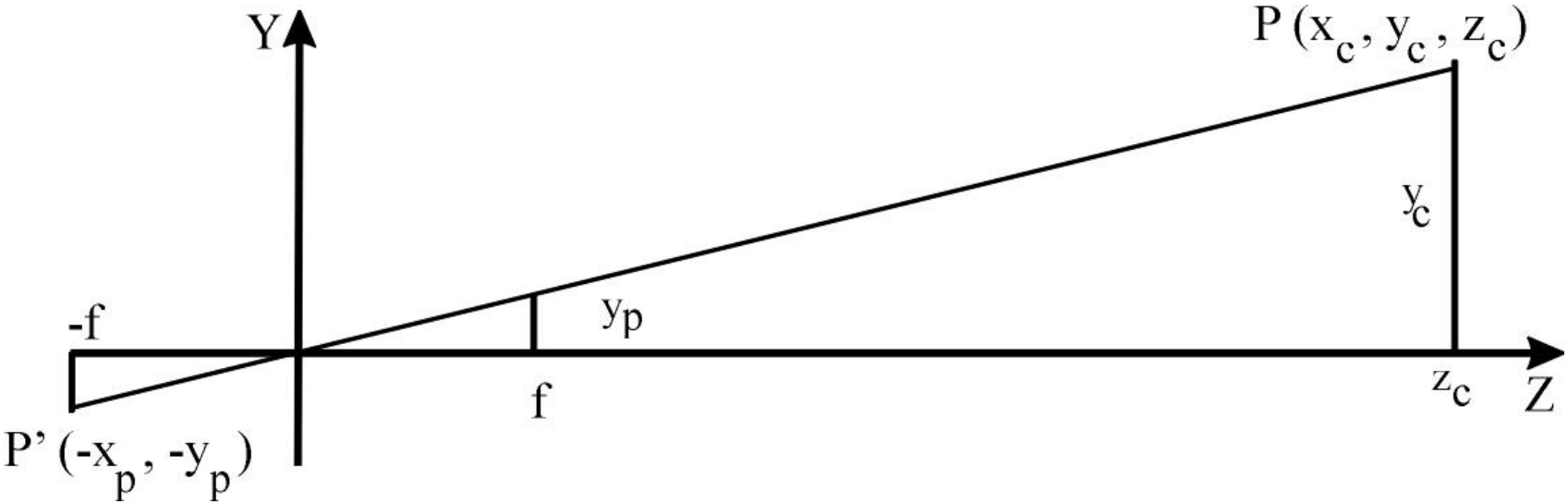
08 October 2021

Camera model

Projection types:

- Perspective projection
- Parallel projection

'Pinhole' model of the perspective projection



Camera model:

- $P' = (-x_p, -y_p)$ - image points in the reversed image,
- $P = (X_c, Y_c, Z_c)$ - 3D scene point (in camera coordinates), f - focal length

Camera model

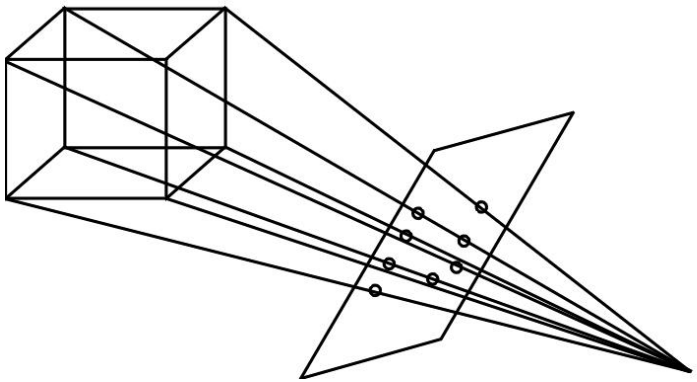
The relation between scene point and corresponding image point:

$$x_p = \frac{f \cdot X_c}{Z_c}, \quad y_p = \frac{f \cdot Y_c}{Z_c}$$

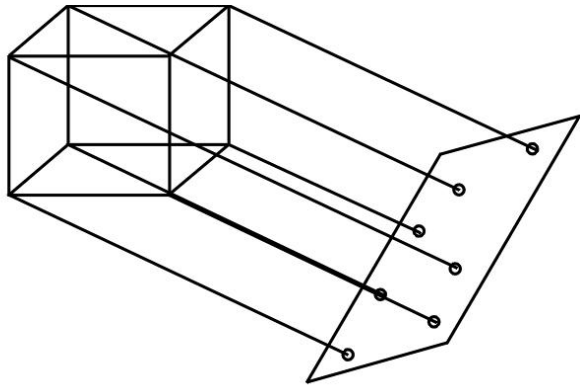
- Remark: the above equation is true if the co-ordinate system's origin is placed in the optical center of the lens

Parallel projection (ortographic projection) - is obtained from perspective projection when f tends to infinity

$$x_p = \lim_{f \approx Z_c \to \infty} \frac{f \cdot X_c}{Z_c} = X_c, \qquad \lim_{f \approx Z_c \to \infty} y_p = \frac{f \cdot Y_c}{Z_c} = Y_c$$



perspective projection



parallel projection

Camera model

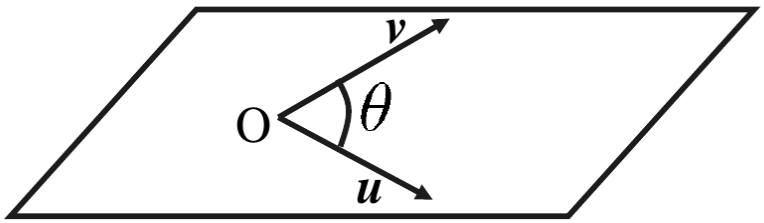
Perspective lenses
(perspective projection)

Telecentric lenses (parallel
projection)

Vector operations

Inner products of two vectors

$$\mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix}, \quad \mathbf{v} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}$$



$$< \mathbf{u}, \mathbf{v} >= \mathbf{u}^T \mathbf{v} = u_1 v_1 + u_2 v_2 + u_3 v_3$$

$$||\mathbf{u}|| = \sqrt{u_1^2 + u_2^2 + u_3^2}$$

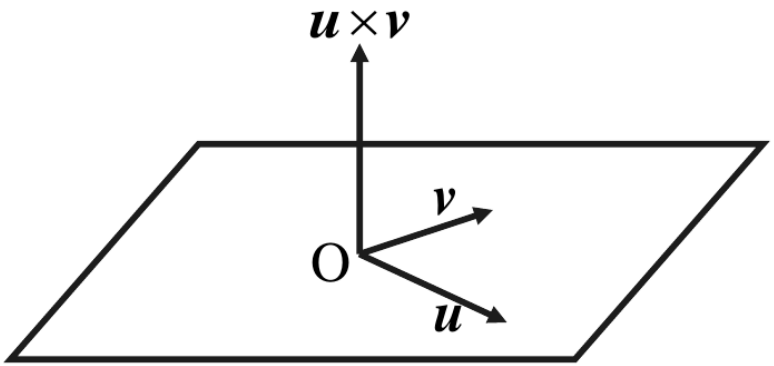
$$\cos\theta = \frac{< \mathbf{u}, \mathbf{v} >}{||\mathbf{u}||||\mathbf{v}||}$$

Cross product of two vectors

$$\mathbf{u} \times \mathbf{v} = \hat{U}\mathbf{v}$$

where

$$\hat{U} = \begin{pmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{pmatrix}$$



Homogenous coordinates I

Homogenous coordinates - in a projective geometry it is a method for representation of vectors (or points) in n-dimensional space using $(n+1)$-dimensional vectors (points). E.g. a point with 2D coordinates (x, y) can be represented as $(x, y, 1)$, or a point with 3D coordinates (x, y, z) can be given as $(x, y, z, 1)$.
Homogenous coordinates can be provided in *normalized* or *unnormalized* form.

- unnormalized form e.g. $(x, y, k)^T$ or $(x, y, z, k)^T$ and $k \neq 1$
- normalized form e.g. $(x, y, 1)^T$ or $(x, y, z, 1)^T$
  - normalized form can be obtained by dividing the unnormalized form by the last coordinate i.e. $(x, y, k) \rightarrow \left(\frac{x}{k}, \frac{y}{k}, 1\right)$ or $(x, y, z, k) \rightarrow \left(\frac{x}{k}, \frac{y}{k}, \frac{y}{k}, 1\right)$
  - coordinates of a point in n-dimensional space can be obtained by taking n first normalized coordinates of the point in $(n+1)$-dimensional homogenous space, so $(x, y, 1)^T \rightarrow (x, y)^T$ and $(x, y, z, 1)^T \rightarrow (x, y, z)^T$

Homogenous coordinates II

- thus, if points from $(n+1)$-dimensional homogenous space have identical normalized form, they represent the same point n-dimensional (non-homogenous) space, so for example $(10, 4, 2)^T$ corresponds to $(20, 8, 4)^T$, corresponds to $(5, 2, 1)^T$ and corresponds to $(5, 2)^T$ in 2D space.

The Finite Projective Camera model

Finite Projective Camera - an extension of the 'pinhole' camera model.
The necessary parameters required to perform a projection of some $\mathbf{P}$ onto its image $\mathbf{p}$ are called camera calibration parameters and they are usually composed of two subsets:

- Extrinsic camera parameters
- Intrinsic camera parameters

Extrinsic parameters The **extrinsic** camera parameters enable to transform the coordinates of a scene point $\mathbf{P}$ from **world** (or **object**) **coordinates** into **camera coordinates**.
In general the extrinsic parameters can be decomposed into two transformation steps:

- a rotation, specified by a 3×3 rotation matrix R and
- a translation, specified by a 1×3 translation vector $\mathbf{t}$.

This is a rigid transformation.

The Finite Projective Camera model

Let $\mathbf{P_w} = (X_w, Y_w, Z_w)^T$ be the world (or object) coordinates and $P_c = (X_c, Y_c, Z_c)$ the camera coordinates of some scene point $\mathbf{P_w}$. Then

$$\mathbf{P_c} = R\mathbf{P_w} + \mathbf{t}$$

Intrinsic parameters The **intrinsic** camera parameters describe:

1. a **perspective projection** expressed for points already given in camera coordinates (in both the 'pinhole' and FPC models) and
2. a transform from the **camera coordinates** into the **image coordinates** (**pixel units** in the image) – this is considered only in the FPC model.

Perspective projection

Perspective projection is given by the equation

$$Z_c \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = F\mathbf{P_c} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = \begin{pmatrix} fX_c \\ fY_c \\ Z_c \end{pmatrix}$$

where f - **focal length** - is the first intrinsic camera parameter.
In the above equation the projection splits into the *point-independent* part (the 3×3 matrix) and the *point-dependent* outer scaling by Z_c.

Remarks

1. The perspective projection is a non-linear transformation in 3-D space.
2. It can be defined in a form of 3×3 or 4×4 matrix when using 4D homogenous coordinate space.

Transformation from metrics to pixel units

In order to obtain pixel coordinates from metrical units, we must perform additional 2D scaling and translation.

$$\begin{pmatrix} x_p \\ y_p \\ 1 \end{pmatrix} = K \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$\text{where } K = \begin{pmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{pmatrix}$$

and the intrinsic camera parameters are:

- s_x, s_y are the scaling coefficients for pixels in horizontal and vertical direction (set to $(1, 1)$ in the pinhole model)
- s_θ is the *skew factor* (set to 0 in the pinhole model) - usually small and negligible
- (ox, oy) is the location of the optical center expressed in image coordinates (set to $(0, 0)$ in the pinhole model).

Camera calibration matrix

All the intrinsic parameters are represented in the camera calibration matrix K_{int}

$$K_{int} = KF = \begin{pmatrix} s_x f & s_\theta f & o_x \\ 0 & s_y f & o_y \\ 0 & 0 & 1 \end{pmatrix}$$

For the simplistic *pinhole* camera model where $K = I$ we have

$$K_{int} = F = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The transformation induced by all the *intrinsic camera parameters* is thus

$$Z_c \begin{pmatrix} x_p \\ y_p \\ 1 \end{pmatrix} = K_{int} \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = \begin{pmatrix} s_x f & s_\theta f & o_x \\ 0 & s_y f & o_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix}$$

Representation of 3D transformations

There are a handful of useful 3D transformations that can be easily expressed as a matrix multiplication

$$\mathbf{P}' = A\mathbf{P}$$

where A is 3×3 matrix, $\mathbf{P} = (P_x, P_y, P_z)^T$, $\mathbf{P}' = (P'_x, P'_y, P'_z)^T$

- Scaling of point coordinates by $(s_x, s_y, s_z)^T$

$$S = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{pmatrix}$$

Representation of 3D transformations

- Rotation of a point around z-axis by angle θ

$$R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- Rotation of a point around x-axis by angle α

$$R_\alpha = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix}$$

- Rotation of a point around y-axis by angle β

$$R_\beta = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix}$$

Dual operations - rotations of a coordinate system by $-\theta, -\alpha, -\beta$

Representation of 3D transformations

Some properties of the representation:

- The transformations expressed in the matrix form can be stacked, e.g.

$$\mathbf{P}' = A_n A_{n-1} \dots A_1 \mathbf{P}$$

  Note that innermost transformations are applied first!

- There exists some notable transformations that **cannot** be easily expressed by matrix multiplication!

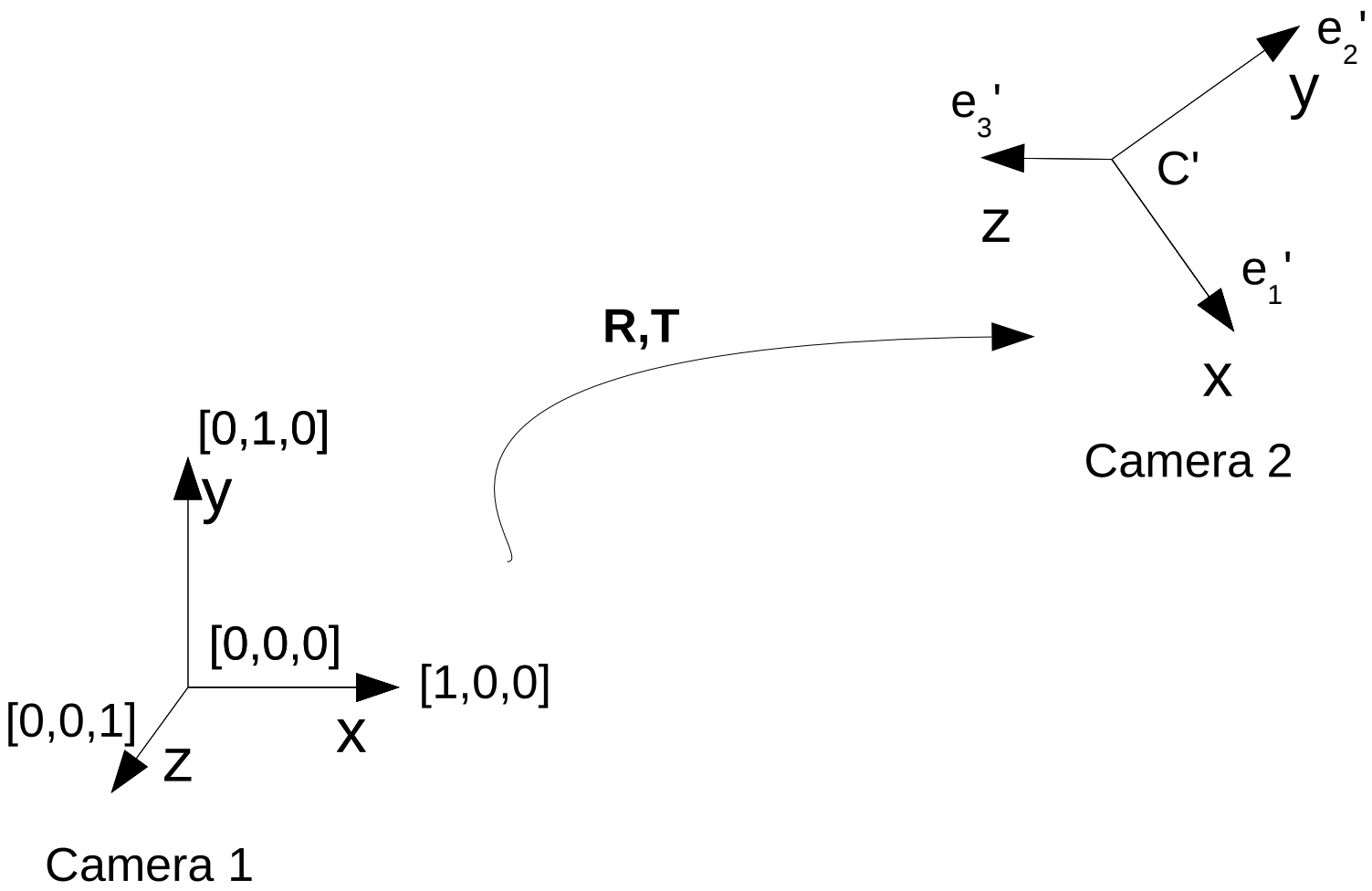
Important transformations that are **not** easily expressible as a matrix multiplication

- Translation of a point by a vector $\mathbf{t} = (t_x, t_y, t_z)^T$

$$P' = P + \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$$

- Perspective projection

Understanding transformations of coordinate systems



Understanding transformations of coordinate systems I

- Assume that the our reference coordinate system is the coord. system of Camera 1.
- This coordinate system can be full described by four points $c = (0,0,0)^T$, $e_1 = (1,0,0)^T$, $e_2 = (0,1,0)^T$ and $e_3 = (0,0,1)^T$
- Now let us transform rigidly these points (and consequently our reference system) using the transformation

$$\mathbf{x_i}' = \mathbf{R}\mathbf{x_i} + \mathbf{t}, \text{ where } \mathbf{x_i} \text{ means } \mathbf{c}, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$$

- Transformed points will form a new coordinate system - let it be associated with the Camera 2.

Understanding transformations of coordinate systems II

- Now for each point with coordinates given in the coordinate system of Camera 2 as $\mathbf{P}^2$, its corresponding coordinates $\mathbf{P}^1$ in the Camera 1 system are given again by

$$\mathbf{P}^1 = \mathbf{R}\mathbf{P}^2 + \mathbf{t}$$

- Observe: Transformation of coordinate systems (eg. cameras) is an 'inverse' of transformation of points from these coordinate systems. I.e. the system of Camera 2 was obtained by a transformation of Camera 1 using $(\mathbf{R}, \mathbf{t})$, however by applying $(\mathbf{R}, \mathbf{t})$ to a point in Kamery 2 coordinate system gives corresponding coordinates in the Camera 2 coordinate system.
- *Exercise: What are coordinates $\mathbf{P}^2$ in the coordinate system of Camera 2 of a point which coordinates in the Camera 1 coordinate system are given as $\mathbf{P}^1$?*

Understanding rotations

Any matrix $\mathbf{R}$ - 3×3 can be a rotation matrix if

- $\mathbf{R}^T = \mathbf{R}^{-1}$ - orthogonality condition
- $det(\mathbf{R}) = 1$ - proper rotation (pure rotation, no reflection)

Let us assume that in our reference coordinate system (1) we choose arbitrary orthogonal vector's $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ forming right-handed coordinate system (2). Then the rotation matrix $\mathbf{R}$ transforming our reference system (1) into a new system (2) can be given simply as

$$\mathbf{R} = (\mathbf{e}_1 | \mathbf{e}_2 | \mathbf{e}_3)$$

Exercise: Show, why?

Alternative representation of rotation

Rotation matrix is 3×3 and has 9 parameters, while any rotation in 3D can be described by only 3 paramters. Such over-representation can cause problems when trying to estimate rotation from observable data. So for estimation of rotation a more compact representation such as *quaternions* or *angle axis* are more convenient.

According to the Euler's Rotation Theorem every 3D rotation can be represented as a rotation around some selected axis. Thus every rotation can be given by a rotation vector $\mathbf{r}$, which orientation specify the rotation axis and magnitude specify the rotation angle θ.

Alternative representation of rotation

The rotation can be applied using *Rodriguez Rotation Formula*

$$\theta := ||\mathbf{r}||_2,\ \mathbf{r} := \frac{\mathbf{r}}{||\mathbf{r}||_2}$$

$$\mathbf{P}' = \mathbf{P}\cos\theta + (\mathbf{r} \times \mathbf{P})\sin\theta + \mathbf{r}(\mathbf{r} \cdot \mathbf{P})(1 - \cos\theta)$$

If we apply the *Rodriguez Formula* to standard basis of a cartesian space we obtain a formula for a conversion from *axis-angle* representation to *rotation matrix* representation.

Alternative representation of rotation

- Conversion from angle-axis to rotation matrix.

$$\theta := ||\mathbf{r}||_2, \ \mathbf{r} := \frac{\mathbf{r}}{||\mathbf{r}||_2}$$

$$\mathbf{R} = \cos\theta \cdot \mathbf{I} + (1 - \cos\theta)\mathbf{r}\mathbf{r}^T + \sin\theta \begin{pmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{pmatrix}$$

- Conversion from rotation matrix to angle-axis - one need to solve an equation:

$$\sin\theta \begin{pmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{pmatrix} = \frac{\mathbf{R} - \mathbf{R}^T}{2}$$

Note: the OpenCV library provides a function *cv::Rodrigues* for automatic conversion

Perspective camera model in homogenous coordinates

Transformations of coordinate systems

All basic transformations of 3D coordinate systems can be expressed as matrix operations on 4D homogenous coordinates

Translation by $(t_x, t_y, t_z)^T$:

$$\mathbf{T} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Axis scaling by $(s_x, s_y, s_z)^T$:

$$\mathbf{S} = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation of a coordinate system by θ around z axis:

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

Transformations of coordinate systems

Rotation of a coordinate system by α around x axis:

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

Rotation of a coordinate system by β around y axis:

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

Note: operatins can be composed by multiplication (order of operations is important!), eg.

$$\mathbf{p}' = \mathbf{S}(\mathbf{R}_\theta(\mathbf{T} \cdot \mathbf{P})) = \mathbf{S}\mathbf{R}_\theta\mathbf{T} \cdot \mathbf{P} = \mathbf{A} \cdot \mathbf{p}$$

Perspective projection

Perspective projection can be expressed in 4D homogenous coordinates in the form of matrix multiplicatin

$$\mathbf{\Psi_f} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{f} & 0 \end{pmatrix}$$

Note: The matrix is not invertible! By multiplying homogenous coordinates of 3D point by a matrix $\mathbf{\Psi_f}$ we obtain familiar formulas of perspective projection

$$\mathbf{p_h} = \begin{pmatrix} x_p \\ y_p \\ f \\ 1 \end{pmatrix} = norm(\mathbf{\Psi_f} \cdot \mathbf{P_h}) = norm \left(\begin{pmatrix} X_c \\ Y_c \\ Z_c \\ \frac{Z_c}{f} \end{pmatrix} \right) = \begin{pmatrix} \frac{fX_c}{Z_c} \\ \frac{fY_c}{Z_c} \\ f \\ 1 \end{pmatrix}$$

The matrix Ψ_f of size 4×4 corresponds to the matrix F

Perspective projection

Scaling, translation (and application of skeweness) transforming from physical to image units is obtained by application of Ψ_s matrix.

$$\mathbf{\Psi_s} = \begin{pmatrix} s_x & s_\theta & 0 & o_x \\ 0 & s_y & 0 & o_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Ψ_s matrix correspond to matrix K. Cummulated matrix of intrinsic camera parameters Ψ_{int} can be now given as

$$\mathbf{\Psi_f} = \mathbf{\Psi_s}\mathbf{\Psi_f} = \begin{pmatrix} s_x & s_\theta & 0 & \frac{o_x}{f} \\ 0 & s_y & \frac{o_y}{f} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{f} & 0 \end{pmatrix}$$

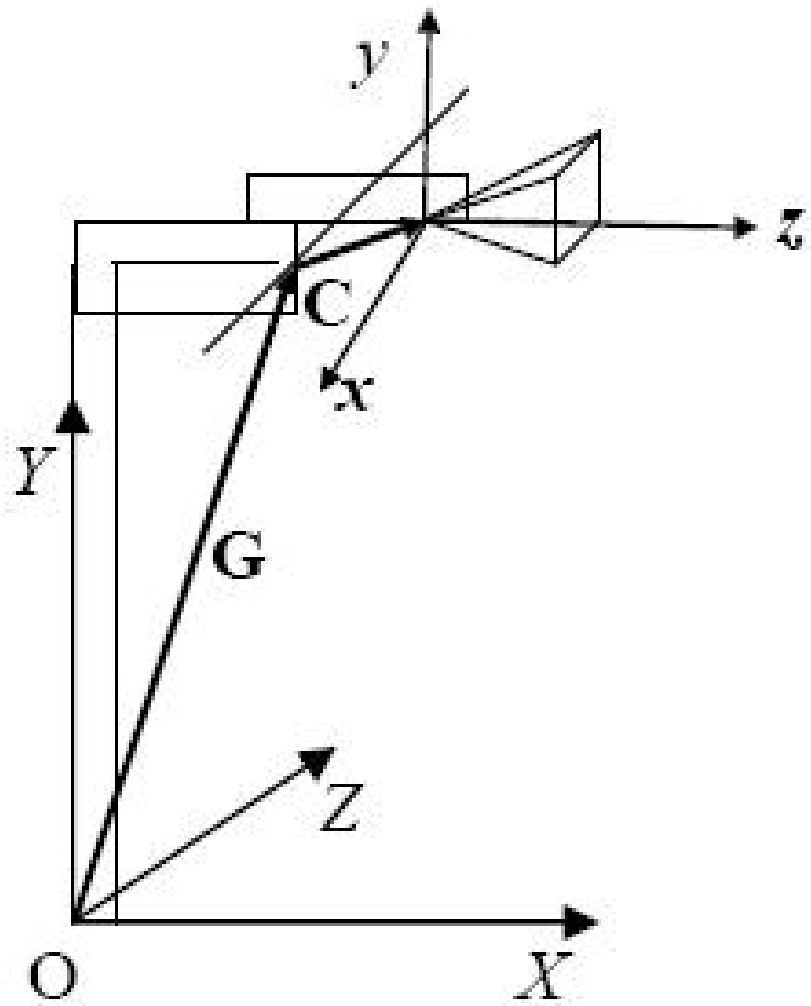
and corresponds to K_{int} matrix.

Robot camera calibration

Calibration setup

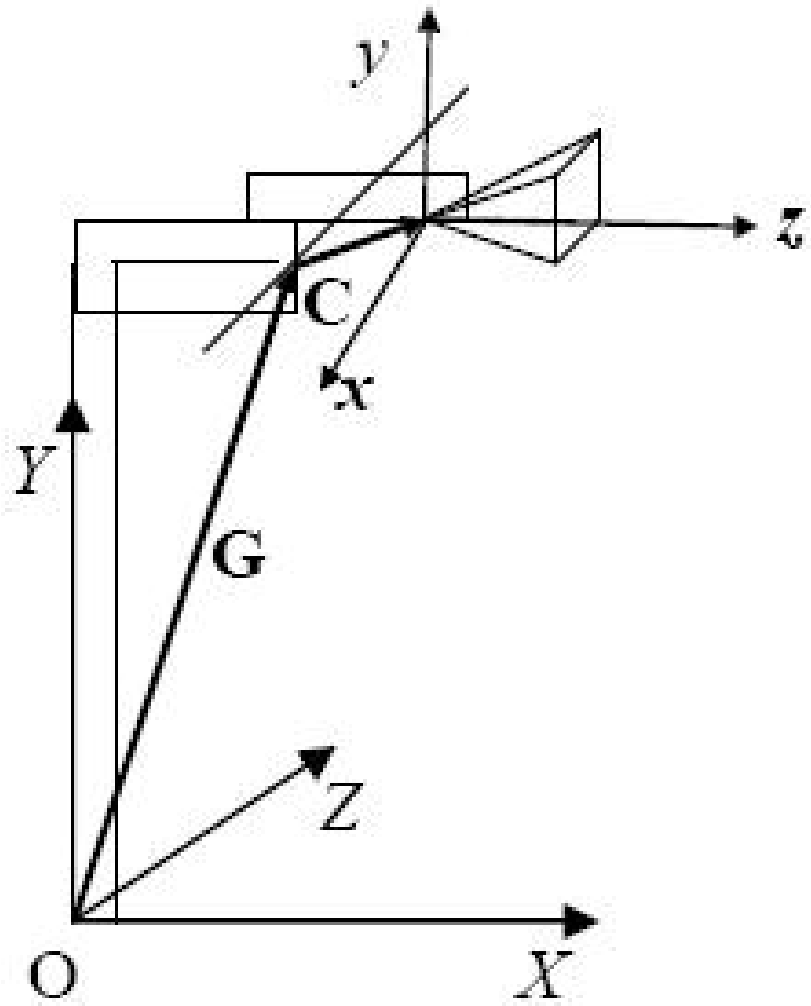
Extrinsic camera parameters:

- α - rotation around a global z axis
- β - rotation around a global y axis
- θ - rotation around a global x axis
- $\mathbf{G}$ - translation vector between the reference coordinate system and camera mount point
- $\mathbf{C}$ - translation vector between the camera mount points and optical center of the camera



Calibration setup

Note: In practice this setup can be described using smaller number of parameters, by merging $\mathbf{G}$ and $\mathbf{C}$ into a single translation vector $\mathbf{t}$.

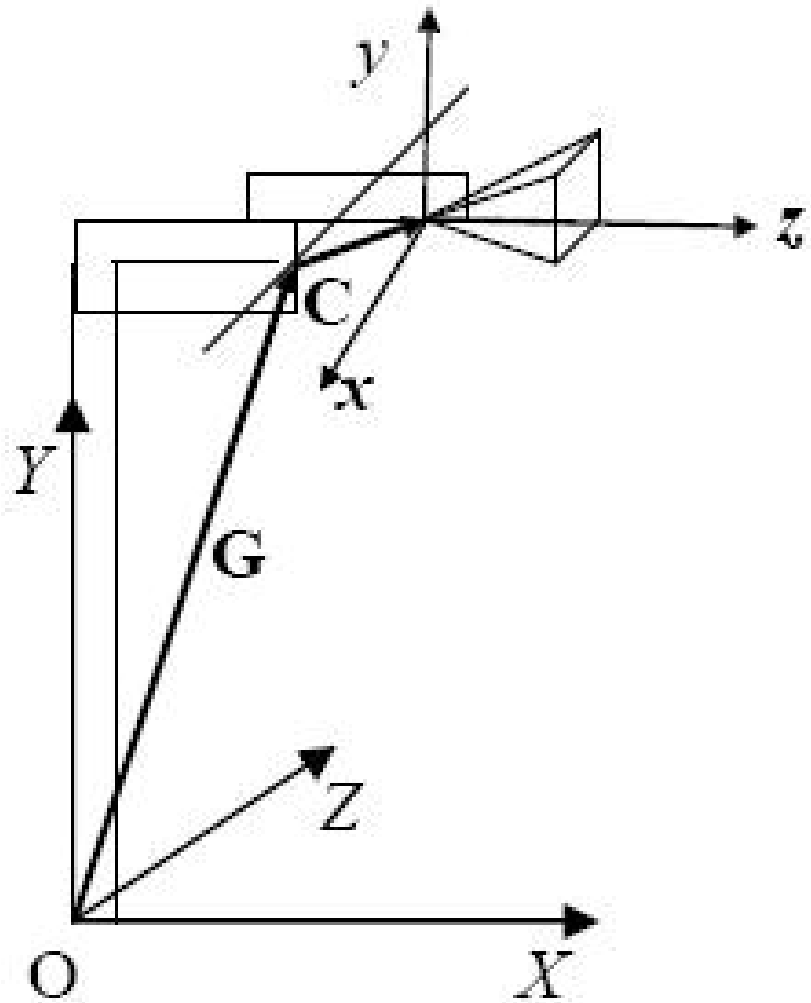


Calibration setup

Intrinsic camera parameters:

- f - focal length
- s_x, s_y - scaling coefficients for pixels
- o_x, o_y - camera optical center
- s_θ - skeweness coefficient

Note: In practice this setup can be described using fewer parameters. Instead of s_x, s_y, f and s_θ, we can use $s_x f$, $s_y f$ and $s_\theta f$, that appeare in K_{int}.



Full perspective projection formula

Projection of a point P_w expressed in global coordinates to image coordinates p

- Using transformation matrix in 4D homogenous coordinates

$$\mathbf{p} = (\mathbf{\Psi_s} \cdot \mathbf{\Psi_f} \cdot \mathbf{T_{-c}} \cdot \mathbf{R}_\theta \cdot \mathbf{R}_\beta \cdot \mathbf{R}_\alpha \cdot \mathbf{T_{-G}}) \cdot \mathbf{P_w} = A \cdot \mathbf{P_w}$$

- Using minimal matrix representation (11 degrees of freedom)

$$k \begin{pmatrix} x_p \\ y_p \\ 1 \end{pmatrix} = \begin{pmatrix} s_x f & s_\theta f & o_x \\ 0 & s_y f & o_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} \tag{1}$$

or

$$k\mathbf{p} = \mathbf{K_{int}}(\mathbf{R}|\mathbf{t})\mathbf{P_w}$$

Full perspective projection formula

$$k\mathbf{p} = \mathbf{K_{int}}(\mathbf{R}|\mathbf{t})\mathbf{P_w}$$

where

- $\mathbf{K_{int}}$ - intrinsic camera parameters
- $\mathbf{R}$ - rotation matrix describing rotation between global and camera coordinate systems
- $\mathbf{t}$ - translation vector describing translation between global and camera coordinate systems

The matrix $\mathbf{A} = \mathbf{K_{int}}(\mathbf{R}|\mathbf{t})$ is sometimes called a *projection matrix* or *camera matrix*, and the perspective projection formula assumes form $k\mathbf{p} = \mathbf{AP_w}$, i.e.

$$k \begin{pmatrix} x_p \\ y_p \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

Types of calibration

- *Explicit* calibration - we use known camera parameters (using e.g. manufacturer's data or physical measurements)
- Autocalibration - automatic calibration, camera parameters are estimated basing on observation of a known calibration pattern

DLT autocalibration

DLT - principle of operation

DLT (Direct Linear Transform) method. A purpose of the method is to estimate parameters of the $\mathbf{A}$ matrix basing on some number of **correspondences** between known 3D scene points and their projections onto image.
Calibration pattern is typically designed to facilitate detection of some characteristic points and improve precision of their localization.

Example: corners of the black and white chessboard become characteristic points of the calibration pattern

For each pair of points consisting of a 3D scene points $P_w = (X, Y, Z, 1)^T$ and image point $p = (x, y, 1)^T$ we can construct an equation of perspective projection, i.e.

$$k \begin{pmatrix} x_p \\ y_p \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

This matrix equation corresponds to 3 linear equations:

$$\begin{aligned} kx &= a_{11}X + a_{12}Y + a_{13}Z + a_{14} \\ ky &= a_{21}X + a_{22}Y + a_{23}Z + a_{24} \\ k &= a_{31}X + a_{32}Y + a_{33}Z + a_{34} \end{aligned}$$

After eliminating k we obtain 2 equations for each pair of points:

$$a_{11}X + a_{12}Y + a_{13}Z - a_{31}xX - a_{32}xY - a_{33}xZ - a_{34}x + a_{14} = 0$$
$$a_{21}X + a_{22}Y + a_{23}Z - a_{31}yX - a_{32}yY - a_{33}yZ - a_{34}y + a_{24} = 0$$

The set of equations

$$\begin{aligned} a_{11}X + a_{12}Y + a_{13}Z - a_{31}xX - a_{32}xY - a_{33}xZ - a_{34}x + a_{14} = 0 \\ a_{21}X + a_{22}Y + a_{23}Z - a_{31}yX - a_{32}yY - a_{33}yZ - a_{34}y + a_{24} = 0 \end{aligned} \tag{2}$$

is a linear set of equations with 12 variables a_{ij} for $i = 1, 2, 3$ i $j = 1, 2, 3, 4$.

Autocalibration procedure

1. Detect $m \geq 6$ image points ($p_i = (x_i, y_i)$, $i = 1, 2, \ldots, m$) which are projections of known 3D points - $P_i = (X_i, Y_i, Z_i)$, $i = 1, 2, \ldots, m$
2. For every pair of points $(\mathbf{p_i}, \mathbf{P_i})$, make two equations of form (2). In result we obtain $M \geq 11$ equations with 12 variables a_{11}-a_{32}, that form a system of linear equations in the form $\mathbf{Ba} = 0$, where $a = (a_{11}, a_{12}, \ldots, a_{34})$.
3. Solve the system of linear equations $\mathbf{Ba} = 0$, which is a **homogenous system of linear equations**.

Methods for solving a linear system $\mathbf{Ba} = 0$

- Fix value of one variable to arbitrary value (e.g. 1), but first make sure it is not 0! Then, depending on the number of equations directly solve the system of linear equations or apply a **Least Squares Method** - in order to reduce errors associated with each observation, or...
- Select first 11 equations. Such system of equations $\mathbf{Ba} = 0$ solve by transforming matrix $\hat{\mathbf{B}}$ to a *row-echelon form* (compute a null space of a matrix $\hat{\mathbf{B}}$), or ...
- Use all the available points and equations and solve a **Homogeonus Least Squares Problem** - reducing errors associated with each observation

DLT - extraction of camera parameters

Solution to the equation $\mathbf{Ba} = 0$, is a vector of parameters $\mathbf{a}$, which can be transformed into a projection matrix

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{pmatrix}$$

- Noted that matrix A has 12 parameters, however they are specified up to a scale, so in reality the number of degrees of freedom is 11.
- From the matrix $\mathbf{A}$ one can (practically) unambiguously retrieve 11 parameters of camera calibration.
- A matrix of intrinsic parameters is obtain by RQ decomposition of the first three columns of the matrix $\mathbf{A}$ (thus we obtain a matrix $\mathbf{K_{int}}$ i $\mathbf{R}$).

DLT - extraction of camera parameters

- The last column of $\mathbf{A}$ corresponds to $\mathbf{K_{int}}$ multiplied by the traslantion vector $\mathbf{t}$ (so $\mathbf{t}$ can be easily extracted)
- Initial scaling of $\mathbf{A}$ is not specified so $\mathbf{K_{int}}$ must be normalized to ensure that we have 1 in the bottom right corner of $\mathbf{K_{int}}$.

Least-Squares Problem

Least-Squares problem for linear equations

LS Problem for linear system of equations with two variables

Assume that we have a sequence of N observations (x_i, y_i) of some phenomenon. We know that according to a known model x i y are connected by linear relatios i.e.

$$y = ax + b \text{ or } ax + b - y = 0$$

- In order to establish model parameters a and b it is enough to have two pairs of measurement points.
- However, if the measurements are prone to significant random errors, parameter estimations will also exhibit errors of similar order.
- To prevent this, we can use more measurement points to 'average' estimations of a i b, and reduce random errors.

This error reduction process can be implemented using Least-Squares Method.

Least-Squares problem for linear equations

Let us write error equations for each pair of points.

$$ax_1 + b - y_1 = \varepsilon_1$$
$$ax_2 + b - y_2 = \varepsilon_2$$
$$\dots$$
$$ax_N + b - y_N = \varepsilon_N$$

Vector $\varepsilon = (\varepsilon_1, \varepsilon_2, \dots, \varepsilon_N)$ is an *error vector*.
Using the Least-Squares Method we look for such values of parameters (a i b here), that minimize a total error measure expressed as a sum of squares of individual errors, i.e.

$$U(a,b) = \varepsilon_1^2 + \varepsilon_2^2 + \dots + \varepsilon_N^2 = \sum \varepsilon_i = ||\varepsilon||^2$$

In our case we have

$$U(a,b) = \sum (ax_i + b - y_i)^2$$

Least-Squares problem for linear equations

$U(a,b)$ is a convex function, therefore its local minimum coincides with a global minimum and is located in a point meeting the conditions:

$$\frac{\partial U}{\partial a} = 0 \text{ oraz } \frac{\partial U}{\partial b} = 0$$

so

$$\frac{1}{2}\frac{\partial U}{\partial a} = \sum (ax_i + b - y_i)x_i = 0 \text{ oraz } \frac{1}{2}\frac{\partial U}{\partial b} = \sum (ax_i + b - y_i) = 0$$

Thus we obtain a system of two linear equations with two variables

$$\begin{aligned} a\sum x_i^2 + b\sum x_i &= \sum x_i y_i \\ a\sum x_i + bN &= \sum y_i \end{aligned}$$

Least-Squares problem for linear equations

The exact solution of the linear system is given by

$$\begin{pmatrix} a \\ b \end{pmatrix} = \frac{1}{N\sum x_i^2 - (\sum x_i)^2} \begin{pmatrix} N & -\sum x_i \\ -\sum x_i & \sum x_i^2 \end{pmatrix} \begin{pmatrix} \sum x_i y_i \\ \sum y_i \end{pmatrix}$$

Least-Squares problem for linear equations

LS Problem for linear system of equations with multiple variables

We have a linear equation in the form:

$$\mathbf{Ax} = \mathbf{b}$$

where $\mathbf{b} \in R^n, \mathbf{x} \in R^n$ i $\mathbf{A} \in R^{n \times m}$ i $n \geq m$. We search for the argument vector $\mathbf{x}$, that minimizes the following sum-of-squares error:

$$\hat{\mathbf{x}} = \text{argmin}_{\mathbf{x}} ||\mathbf{Ax} - \mathbf{b}||^2$$

The solution is a vector:

$$\hat{\mathbf{x}} = \mathbf{A}^{\dagger}\mathbf{b}$$

where $\mathbf{A}^{\dagger}$ is Moore-Penrose pseudo-inverse matrix, specified by the equation:

$$\mathbf{A}^{\dagger} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$$

LS Problem for a homogenous system of equations

We are given a homogenous system of equations $\mathbf{Ax} = 0$, with a matrix $\mathbf{A}^{n \times m}$ with $rank(\mathbf{A}) = m$ and $n \geq m$. Solution to the LS problem for such homogenous system of equations is a vector $\hat{\mathbf{x}}$ meeting conditions

- $\hat{\mathbf{x}} = \text{argmin}_{\mathbf{x}} ||\mathbf{Ax}||^2$ (minimization of a sum-of-squares error)
- $||\hat{\mathbf{x}}|| = 1$ (solution is searched among vectors of length 1)

It can be shonw that solution to the problem can be obtained by performing Singular Value Decomposition (SVD) of the matrix $\mathbf{A}$, i.e.

$$\mathbf{A} = \mathbf{UDV}^T$$

where $\mathbf{U}^{-1} = \mathbf{U}^T$, $\mathbf{V}^{-1} = \mathbf{V}^T$, a $\mathbf{D}$ is a diagonal matrix.
The result vector $\hat{\mathbf{x}}$ is a column of the matrix $\mathbf{V}$ corresponding to the smallest singular value from $\mathbf{D}$.

Least-Squares problem for non-linear equations

LS Problem for non-linear equations There is given a function $f(\mathbf{x})$, given in the vector form:

$$\mathbf{y} = f(\mathbf{p})$$

where $\mathbf{y} \in R^N$, $\mathbf{p} \in R^n$ and $N \geq n$.
For the given observation vector $\mathbf{y}$, a goal is to find such value of $\mathbf{p}$, that minimizes the difference $\mathbf{y} - f(\mathbf{p})$. This can be expressed as minimization of a sum of squares of the residual vector $\mathbf{r}$.

$$\Phi(\mathbf{p}) = \frac{1}{2} r^T(\mathbf{p}) r(\mathbf{p}) = \frac{1}{2} \sum_{i=1}^{N} r_i^2(\mathbf{p}) = \frac{1}{2} \sum_{i=1}^{N} (y_i - f(\mathbf{p}))^2$$

The function $\Phi(\mathbf{p})$ can be called a cost function.

Least-Squares problem for non-linear equations

Optimization of Φ is an unconstrained optimization and is solved iteratively. In order to find the most desirable direction of search, there is us the information on the first and the second derivative of the cost function Φ.

A gradient of Φ (a vector of 1-st derivatives) is computed as

$$\nabla\Phi(\mathbf{p}) = \sum_{i=1}^{N} r_i(\mathbf{p})\nabla r_i(\mathbf{p}) = \mathbf{J}(\mathbf{p})^T\mathbf{r}(\mathbf{p})$$

where

$$\{\mathbf{J}(\mathbf{p})\}_{ij} = \frac{\partial r_i(\mathbf{p})}{\partial p_j}$$

is the Jacobiego matrix of first derivatives of the residual vector $\mathbf{r}(\mathbf{p})$ (negation of Jacobi matrix of the original function $f(\mathbf{p})$).

Least-Squares problem for non-linear equations

Hessian matrix of Φ (second-derivative matrix) is computed as:

$$\mathbf{H}(\mathbf{p}) = \nabla^2\Phi(\mathbf{p}) = J(\mathbf{p})^T J(\mathbf{p}) + \sum_{i=1}^{N} r_i(\mathbf{p})\nabla^2 r_i(\mathbf{p})$$

In the case of the Least Squares method we are usually dealing with relatively small values of $\mathbf{r}$, so the Hessian matrix can be approximated by:

$$\mathbf{H}(\mathbf{p}) = \nabla^2\Phi(\mathbf{p}) = J(\mathbf{p})^T J(\mathbf{p})$$

Solution to the Least-Squares Problem for non-linear equations

1. **Steepest Descent Method**

   Current approximation of solution is updated using a rule

$$\mathbf{p}_{i+1} = \mathbf{p}_i - \lambda \nabla \Phi(\mathbf{p}_i)$$

   i.e. we travel in the direction of the largest local decline of the cost function value. Step size depends on the λ coefficient.

2. Gauss-Newton Method

We assume that the function is locally similar to the quadratic function in some neighbourhood of the point. We apply a Taylor series expansion of the gradient of Φ

$$\nabla\Phi(\mathbf{p}) = \nabla\Phi(\mathbf{p}_0) + (\mathbf{p} - \mathbf{p}_0)^T\nabla^2\Phi(\mathbf{p}_0) + \dots$$

We are leaving only the first two components, and solve the equation $\nabla\Phi(p_{opt}) = 0$ to obtain the iterative update rule:

$$\mathbf{p}_{i+1} = \mathbf{p}_i - (\nabla^2\Phi(\mathbf{p}_i))^{-1}\nabla\Phi(\mathbf{p}_i) = \mathbf{p}_i - (\mathbf{H}(\mathbf{p_i}))^{-1}\nabla\Phi(\mathbf{p}_i)$$

3. Levenberg-Marquardt method

- Steepest Descent Method provides one of the most stable convergence, but is rather slow.
- Gauss-Newton Method is different - it is capable of a very fast convergence (it aims directly at the estimated minimum of the cost function), but the convergence is less stable
- Levenberg-Marquardt method 'switches' between these to methods depending on circumstances

In the simpler version, the next approximation is computed as:

$$\mathbf{p_{i+1}} = \mathbf{p_i} - (\mathbf{H}(\mathbf{p_i}) + \lambda I)^{-1} \nabla \Phi(\mathbf{p_i})$$

Observe, that for large λ's the method is mostly Steepest Descent Method and for small λ's mostly Gauss-Newton method.

Least-Squares problem for non-linear equations

It was established by the autors, that even when the method resembles mostly the Steepest Descent Method it is profitable to use local curvatures to regulate the step of the method. I.e. elements of the gradient are scaled by the curvature coefficients from the Hessian Matrix diagonal, using the rule that the smaller the curvature, the larger step is allowed in this direction. Iterative update rule is then given as:

$$\mathbf{p_{i+1}} = \mathbf{p_i} - (\mathbf{H}(\mathbf{p_i}) + \lambda diag(\mathbf{H}(\mathbf{p_i})))^{-1} \nabla \Phi(\mathbf{p_i})$$

Least-Squares problem for non-linear equations

Optimizatin procedure in the **LM** algorithm can be given as:

1. Compute candidate for a new approximation $\mathbf{p_{i+1}}$ basing on $\mathbf{p_i}$
2. Observe the error $\mathbf{r_{i+1}}$
3. If the error is larger than the error for $\mathbf{r_i}$ increase λ k -fold (e.g. k $= 10$) and go back to the point 1 without update of $\mathbf{p}_i$ (we increase the influence of the Steepest Descent Method.
4. Upadte $\mathbf{p_i} := \mathbf{p_{i+1}}$ and decrease λ k-times and return to the point 1(we increase influence of the Gaussa-Newtona method.)

Sensor distortions

Radial and tangential distortions

The perspective camera model is simplified. Real cameras always introduce non-linear distortions, that must be taken into account in more realistic model. Two types of distortions are most important:

- radial distortions (more pronounced, up to $0.01mm$)
- tangential distortions (usually smaller than $3\mu m$)

Modified perspective projection model

Taking into account distortions, we have to modify the projection model described in (1).

$$k \begin{pmatrix} x_p \\ y_p \\ 1 \end{pmatrix} = \begin{pmatrix} s_x f & s_\theta f & o_x \\ 0 & s_y f & o_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

Modified perspective projection model

Modified perspective projection model:

1. Transformation of a point from the global to the camera coordinate system (according to (1))

$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

2. Normalization of a point $(X_c, Y_c, Z_c)^T$ expressed in 3D space of homogenous coordinates (the actual perspective projection)

$$\begin{pmatrix} x_n \\ y_n \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{X_c}{Z_c} \\ \frac{Y_c}{Z_c} \\ 1 \end{pmatrix}$$

(x_n, y_n) are described as *normalized coordinates*.

Modified perspective projection model

3. Application of the selected model of radial and tangential distortions

$$\begin{pmatrix} x_n \\ y_n \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} x_d \\ y_d \\ 1 \end{pmatrix}$$

(check next slides for the exact formula)

4. Transformation to image coordinates using K_{int} matrix (similar to (1)).

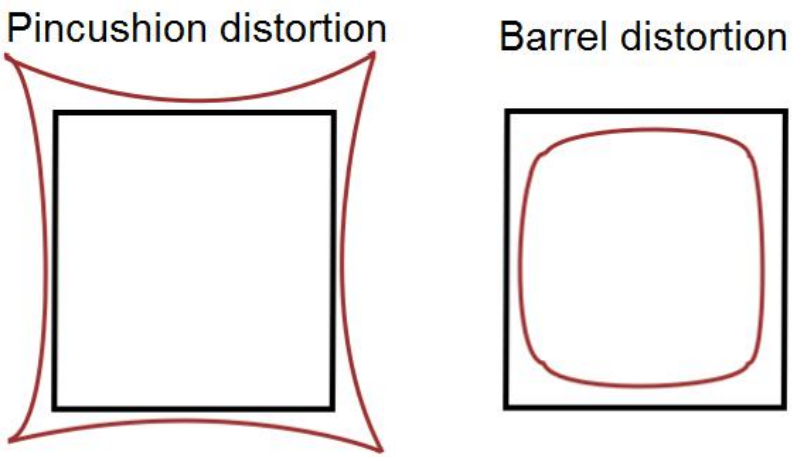
$$\begin{pmatrix} x_p \\ y_p \\ 1 \end{pmatrix} = \begin{pmatrix} s_x f & s_\theta f & o_x \\ 0 & s_y f & o_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_d \\ y_d \\ 1 \end{pmatrix} = K_{int} \begin{pmatrix} x_d \\ y_d \\ 1 \end{pmatrix}$$

Radial and tangential distortions

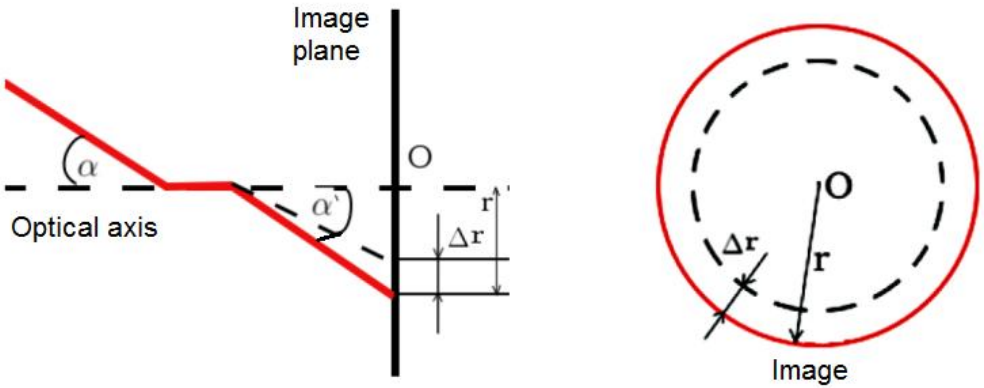
Radial distortions:

- Radial distortions are a result of unwanted deflection of the light ray going through the lenses
- The larger the angle between the ray and the optical axis, the more pronounced the radial distortion (distortions are particularly visible near image borders).
- Basic types of radial distortions:
  - Barrel distortions - pixels are 'repelled' from the optical center of the image
  - Pincushion distortions - pixels are 'attracted' to the optical center of the image

Radial and tangential distortions



Radial distortions in the image



Influence of the ray angle on the radial distortions

Radial and tangential distortions

Models of radial disotortions:

- Model Brown-Conrady

$$x_d = x_n(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 \dots)$$
$$y_d = y_n(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 \dots)$$

- Model Brown-Conrady + division model

$$x_d = x_n \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \dots}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6 + \dots}$$
$$y_d = y_n \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \dots}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6 + \dots}$$

k_1,k_2,k_3,k_4,k_5,k_6 are distortion parameters and $r^2 = x_n^2 + y_n^2$.

Radial and tangential distortions

A source of tangential distortions is a lack of a perfect perpendicular alignment of the optical axis with respect to the image plane (due to imperfect alignment of the lenses)

A model of tangential distortions:

$$x_d = x_n + 2p_1 x_n y_n + p_2(r^2 + 2x_n^2)$$
$$y_d = y_n + p_1(r^2 + 2y_n^2) + 2p_2 x_n y_n$$

where p_1 i p_2 are distortion parameters and $r^2 = x_n^2 + y_n^2$.

A joint (practical) distortion model

$$x_d = x_n \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + 2p_1 x_n y_n + p_2(r^2 + 2x_n^2)$$
$$y_d = y_n \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + p_1(r^2 + 2y_n^2) + 2p_2 x_n y_n$$

where k_i,p_i are distortion parameters i $r^2 = x_n^2 + y_n^2$.

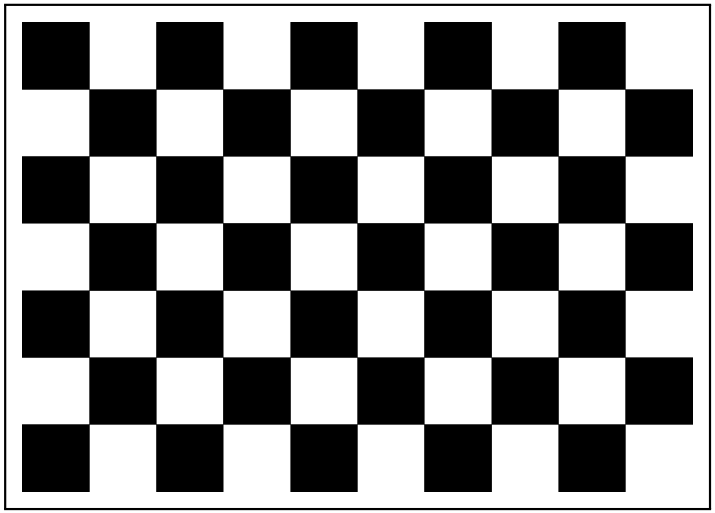
Practical calibration algorithm - Zhang calibration

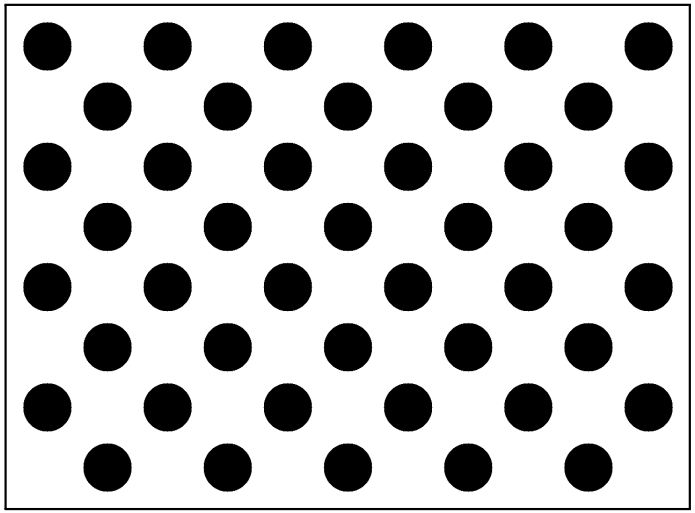
Zhang calibration

Zhang calibration is performed using multiple views of the calibration pattern.

Source: `https://kushalvyas.github.io/calib.html`

Zhang calibration





Calibration patterns (OpenCV)

Source: https://www.oreilly.com/library/view/learning-opencv-3/9781491937983/app03.html

Zhang algorithm

1. Estimation of the intrinsic parameters of the camera using all views. For each view there is estimated a homography between the pattern and its image. From each homography one can obtain 2 equations related only to intrinsic parameters of the camera
2. Estimation of inintial camera positions using 2D-3D correspondences. There is used a *Perspective-n-point* together with Levenberg-Marquardt optimization.
3. Application of the global Levenberg-Marquardt algorithm in order to obtain: $\mathbf{K_{int}}$, $\mathbf{R_i}$, $\mathbf{t_i}$, distortion coefficients $[k_1, k_2, p_1, p_2[, k_3[, k_4, k_5, k_6]]]$.
   - There is minimized a reprojection error, which means the difference between coordinates (x_p, y_p) computed from 3D points using current camera parameters and $(\tilde{x}_p, \tilde{y}_p)$ - measured image points.

Camera calibration OpenCV

```
double calibrateCamera(
        InputArrayOfArrays objectPoints,
        InputArrayOfArrays imagePoints,
        Size imageSize,
        InputOutputArray cameraMatrix,
        InputOutputArray distCoeffs,
        OutputArrayOfArrays rvecs,
        OutputArrayOfArrays tvecs,
        int flags=0,
        TermCriteria
        criteria = TermCriteria(TermCriteria::COUNT+
                TermCriteria::EPS, 30, DBL_EPSILON) );
```

Camera calibration OpenCV

Intrinsic camera parameters:

- objectPoints i imagePoints - points of 3D pattern and corresponding image points organized as an array of arrays (one for each view of the pattern)
- cameraMatrix - estmiated $\mathbf{K_{int}}$ matrix
- distCoeffs - estimated distortion coefficients: 4, 5 or 8 elements $[k_1, k_2, p_1, p_2[, k_3[, k_4, k_5, k_6]]]$

Extrinsic camera parameters:

- rvecs - transformation from the pattern coordinate system to the camera coordinate system - rotation vectors use *axis angle* and correspond to $\mathbf{R_i}$
- tvecs - transformation from the pattern coordinate system to the camera coordinate system - translation vectors correspond to $\mathbf{t_i}$

Examples

E.1 DLT calibration I

There are given 3 points in 3D space specified by the following global coordinates

Point	X_i	Y_i	Z_i
P1	5	6	5
P2	0	2	-3
P3	4	3	6

Their perspective projections onto the image plane have been detected as:

Point	x_i	y_i
P1	3	3.9
P2	-0.4	-2
P3	2.2	2.2

E.1 DLT calibration II

Assuming there is no rotation between the global and camera coordinate systems and identity matrix of camera intrinsic parameters $\mathbf{K} = I$ (so the only estimated parameter is the focal length f in $\mathbf{F}$):

1. Show, that 3 points are enough to perform self-calibration using the DLT method.
2. Calibrate the camera using DLT method (create a system of equations, compute pseudeo-inverse matrix and verify its determinant, extract final camera parameters).
3. How many points are required to perform calibration assuming non-zero rotation around z axis? Justify your answer.

Solution 1.1

Recall the general camera projection equation:

$$k \begin{pmatrix} x_p \\ y_p \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

$$a_{11}X + a_{12}Y + a_{13}Z - a_{31}xX - a_{32}xY - a_{33}xZ - a_{34}x + a_{14} = 0$$
$$a_{21}X + a_{22}Y + a_{23}Z - a_{31}yX - a_{32}yY - a_{33}yZ - a_{34}y + a_{24} = 0$$

The projection matrix $\mathbf{A}$ is given as

$$\mathbf{A} = \mathbf{K_{int}}(\mathbf{R}|\mathbf{t})$$

According to the assumptions for this task we have:

$$K_{int} = KF = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} = f \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{1}{f} \end{pmatrix}$$

Regarding projection each multiplicity of $\mathbf{K_{int}}$ does as well as the original $\mathbf{K}$ int so we may drop f multiplier from the equation

$$K^1_{int} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{1}{f} \end{pmatrix}$$

Then the projection matrix $\mathbf{A}_1$ can be given as:

$$\mathbf{A} = \mathbf{K^1_{int}}(\mathbf{R}|\mathbf{t}) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{1}{f} \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1/f & t_z/f \end{pmatrix}$$

Hence, the number of unknowns is reduced from 12 to 4 in this situation, as:

$$a_{11} = 1,\ a_{12} = a_{13} = 0,$$
$$a_{21} = 0,\ a_{22} = 1,\ a_{23} = 0,$$
$$a_{31} = 0,\ a_{32} = 0.$$

The simplified equations for each point pair are:

$$X - a_{33}xZ - a_{34}x + a_{14} = 0$$
$$Y - a_{33}yZ - a_{34}y + a_{24} = 0$$

It is a standard (non-homogenous) system of linear equations with 4 unknowns. So 3 point pairs should be sufficient for self-calibration.

Solution 1.2. Calibrate the camera using DLT method

The system of equations can be written in a matrix form as

$$\begin{pmatrix} -x_1Z_1 & -x_1 & 1 & 0 \\ -y_1Z_1 & -y_1 & 0 & 1 \\ -x_2Z_2 & -x_2 & 1 & 0 \\ -y_2Z_2 & -y_2 & 0 & 1 \\ -x_3Z_3 & -x_3 & 1 & 0 \\ -y_3Z_3 & -y_3 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_{33} \\ a_{34} \\ a_{14} \\ a_{24} \end{pmatrix} = \begin{pmatrix} -X_1 \\ -Y_1 \\ -X_2 \\ -Y_2 \\ -X_3 \\ -Y_3 \end{pmatrix}$$

Using the actual numerical values we have

$$\begin{pmatrix} -15 & -3 & 1 & 0 \\ -19.5 & -3.9 & 0 & 1 \\ -1.2 & 0.4 & 1 & 0 \\ -6 & 2 & 0 & 1 \\ -13.2 & -2.2 & 1 & 0 \\ -13.2 & -2.2 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_{33} \\ a_{34} \\ a_{14} \\ a_{24} \end{pmatrix} = \begin{pmatrix} -5 \\ -6 \\ 0 \\ -2 \\ -4 \\ -3 \end{pmatrix}$$

Solution to the above set of equations does not exist due to noise. More realistic is LSE formulation.

$$\begin{pmatrix} -15 & -3 & 1 & 0 \\ -19.5 & -3.9 & 0 & 1 \\ -1.2 & 0.4 & 1 & 0 \\ -6 & 2 & 0 & 1 \\ -13.2 & -2.2 & 1 & 0 \\ -13.2 & -2.2 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_{33} \\ a_{34} \\ a_{14} \\ a_{24} \end{pmatrix} = \begin{pmatrix} -5 \\ -6 \\ 0 \\ -2 \\ -4 \\ -3 \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \end{pmatrix}$$

Squared error function is given by

$$U = \sum_{i=1}^{6} \varepsilon_i^2$$

The minimum of U is achieved if:

$$\frac{1}{2}\frac{\partial U}{\partial a_{33}} = 0;\ \frac{1}{2}\frac{\partial U}{\partial a_{34}} = 0;\ \frac{1}{2}\frac{\partial U}{\partial a_{14}} = 0;\ \frac{1}{2}\frac{\partial U}{\partial a_{24}} = 0;$$

Recall that the overdetermined system of linear equations $\mathbf{Ax} = \mathbf{b}$ we can solve using LSE approach by solving $\mathbf{A^T Ax} = \mathbf{A^T b}$. Thus we have

$$\begin{pmatrix} 991.17 & 166.65 & -29.4 & -38.7 \\ 166.65 & 38.05 & -4.8 & -4.1 \\ -29.4 & -4.8 & 3 & 0 \\ -38.7 & -4.1 & 0 & 3 \end{pmatrix} \begin{pmatrix} a_{33} \\ a_{34} \\ a_{14} \\ a_{24} \end{pmatrix} = \begin{pmatrix} 296.4 \\ 49.8 \\ -9 \\ -11 \end{pmatrix}$$

The determinant of $\mathbf{A}$ is $5367.5 \gg 0$. Hence, there exists a good solution to this system of equations. The solution is

$$\begin{pmatrix} a_{33} \\ a_{34} \\ a_{14} \\ a_{24} \end{pmatrix} = \begin{pmatrix} 0.4748 \\ -0.4568 \\ 0.9223 \\ 1.8341 \end{pmatrix}$$

We can now extract specific camera parameters

$$\begin{cases} f = 1/a_{33} = 2.1061 \\ t_x = a_{14} = 0.9233 \\ t_y = a_{24} = 1.8341 \\ t_z = a_{34} f = -0.9620 \end{cases}$$

Solution 1.3 If a non-zero rotation has to be considered then the camera projection matrix is:

$$\mathbf{A} = \mathbf{K}^1_{\mathbf{int}}(\mathbf{R}|\mathbf{t}) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{1}{f} \end{pmatrix} \cdot \begin{pmatrix} \cos(\alpha) & \sin(\alpha) & 0 & t_x \\ -\sin(\alpha) & \cos(\alpha) & 0 & t_y \\ 0 & 0 & 1 & t_z \end{pmatrix} =$$

$$\begin{pmatrix} \cos(\alpha) & \sin(\alpha) & 0 & t_x \\ -\sin(\alpha) & \cos(\alpha) & 0 & t_y \\ 0 & 0 & 1/f & t_z/f \end{pmatrix}$$

In adition to the 4 unknowns from the previous case: a_{14}, a_{24}, a_{33}, a_{34} there are 2 other (partially independent) parameters: a_{11} $(= a_{22})$ and a_{21} $(= -a_{12})$

Hence, in total there are 6 unknowns. Solution requires at least 3 pairs of points to be measured in the calibration process.

Remark: This time we will end-up with a set of homogenous equations

E.2 Defining a coordinate system

There are given 3 points in the global coordinate system: $\mathbf{A} = (1, 1, 1)^T$, $\mathbf{B} = (1, 1, 2)^T$ and $\mathbf{C} = (1, 2, 1)^T$. These points define a plane P_{ABC}.

1. Construct a new coordinate system, that meets the following conditions:
   - the plane XY of the new coordinate system coincides with the plane P_{ABC},
   - the axis X coincides with the vector $\overrightarrow{BC}$, and $\mathbf{B}$ is the origin of the new coordinate system

   and specify its origin and unit basis vectors.
2. Find a rigid transformation transformation of point coordinates from the new coordinate system to the global coordinate system.

Solution 2.1

Since axes X and Y lie in the plane P_{ABC}, the axis Z is perpendicular to the plane P_{ABC} that is defined by the vectors $\overrightarrow{BC}$ and $\overrightarrow{BA}$, so it is perpendicular to these vectors. We use a cross product to compute the basis vector of Z axis.

$$\mathbf{e_z} \propto \overrightarrow{BC} \times \overrightarrow{BA} = (0,1,-1)^T \times (0,0,-1)^T$$
$$= \left\{ \begin{vmatrix} i & j & k \\ 0 & 1 & -1 \\ 0 & 0 & -1 \end{vmatrix} = -i \right\} = (-1,0,0)^T$$

Note: the oposite diretion of $\mathbf{e_z} = (1,0,0)^T$ is also correct, but we won't consider this case

The X axis and its basis vector coincides with $\overrightarrow{BC}$ so

$$\mathbf{e_x} \propto (0,1,-1)^T$$

The remaining Y axis basis vector must be perpendicular to both $\mathbf{e_x}$ and $\mathbf{e_z}$

$$\mathbf{e_y} \propto \mathbf{e_z} \times \mathbf{e_x} = (-1, 0, 0)^T \times (0, 1, -1)^T = (0, -1, -1)^T$$

Note: the order z-x is important to obtain a right-handed coordinate system. Since in general we want our basis vectors to have length 1 (**unit** basis vectors), we have to normalize them:

$$\mathbf{e_x} = \left(0, \frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}\right)^T, \mathbf{e_y} = \left(0, -\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}\right)^T, \mathbf{e_y} = (-1, 0, 0)^T$$

The origin of the new coordinate system is obviously

$$\mathbf{O} = \mathbf{B} = (1, 1, 2)^T$$

Solution 2.2

The rotation matrix transforming axes (axes' basis vectors) of the global coordinate system into axes (axes' basis vectors) of the new coordinate system is given as:

$$\mathbf{R} = (\mathbf{e_x}|\mathbf{e_y}|\mathbf{e_z}) = \begin{pmatrix} 0 & 0 & -1 \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \end{pmatrix}$$

and the corresponding translation vector

$$\mathbf{t} = \mathbf{B} = (1, 1, 2)^T$$

The full rigid transformation of a point from the new to the global coordinate system is given by

$$\mathbf{P^g} = \mathbf{R}\mathbf{P^{new}} + \mathbf{t}$$

where $\mathbf{P^g}$ and $\mathbf{P^{new}}$ are coordinates of a point in the global and the new coordinate system respectively.

Let $\mathbf{P^i}$ be the coordinates of a point $\mathbf{P}$ in the i-th coordinate system. The coordinate systems 1 and 3 are related by equations

$$\mathbf{P}^2 = \mathbf{R}_2\mathbf{P}^3 + \mathbf{t}_2,\ \mathbf{P}^1 = \mathbf{R}_1\mathbf{P}^2 + \mathbf{t}_1$$

1. If we are given a point $\mathbf{P}^3$ given int the 3-rd coordinate system, what are its coordinates int the 1-st coordinate system ($\mathbf{P}^1$)?
2. Inversely, if we have a point $\mathbf{P}^1$ in the 1-st coordinate system, what would be the coordinates of the same point in the 3-rd coordinate system ($\mathbf{P}^3$).

Solution 3.1

Directly from the relations provided we have:

$$\mathbf{P}^1 = \mathbf{R}_1\mathbf{P}^2 + \mathbf{t}_1 = \mathbf{R}_1(\mathbf{R}_2\mathbf{P}^3 + \mathbf{t}_2) + \mathbf{t}_1 = \mathbf{R}_1\mathbf{R}_2\mathbf{P}^3 + \mathbf{R}_1\mathbf{t}_2 + \mathbf{t}_1$$

Solution 3.2

Directly from the relations provided we have:

$$\mathbf{P}^3 = \mathbf{R}_2{}^T(\mathbf{P}^2 - \mathbf{t}_2),\ \mathbf{P}^2 = \mathbf{R}_1{}^T(\mathbf{P}^1 - \mathbf{t}_1),\ \text{note: } \mathbf{R}^{-1} = \mathbf{R}^{\mathbf{T}}$$

hence

$$\mathbf{P}^3 = \mathbf{R}_2{}^T(\mathbf{R}_1{}^T(\mathbf{P}^1 - \mathbf{t}_1) - \mathbf{t}_2) = \mathbf{R}_2{}^T\mathbf{R}_1{}^T\mathbf{P}^1 - \mathbf{R}_2{}^T\mathbf{R}_1{}^T\mathbf{t}_1 - \mathbf{R}_2{}^T\mathbf{t}_2$$

E.4 Coordinate systems and a projection matrix

The camera coordinate system was first aligned with the global coordinate system so the projection matrix had a form:

$$\mathbf{A}_1 = \mathbf{K}_{\mathbf{int}}(\mathbf{I}|0)$$

Then the camera was rotated by $\mathbf{R}$ and translated by $\mathbf{t}$. What is the new camera projection matrix $\mathbf{A}_2$ after these transformations?

Let denote as $\mathbf{P}$ some point coordinates in the global coordinate system and let $\mathbf{P}^1$ and $\mathbf{P}^2$ be its coordinates in the 1-st and the 2-nd camera frame respectively. According to the projection matrix provided, coordinates of the point $\mathbf{P}$ in the 1-st camera frame are

$$\mathbf{P}^1 = \mathbf{R}_1\mathbf{P} + \mathbf{t}_1 = I\mathbf{P} + 0 = \mathbf{P}$$

For the 2-nd camera wa seek for the relation $\mathbf{P}^2 = \mathbf{R}_2\mathbf{P} + \mathbf{t}_2$, where $\mathbf{R}_2$ and $\mathbf{t}_2$ are still unknown parameters. Since the 2-nd camera is the rotated and translated version of the 1-st camera we have $\mathbf{P}^1 = \mathbf{R}\mathbf{P}^2 + \mathbf{t}$

Recall that the transformation of coordinate systems is opposite the transformation of respective points in these coordinate systems.

After basic transformations we have

$$\mathbf{P}^2 = \mathbf{R}^T\mathbf{P}^1 - \mathbf{R}^T\mathbf{t} = \mathbf{R}^T\mathbf{P} - \mathbf{R}^T\mathbf{t}$$

and the projection matrix for the 2-nd camera is

$$\mathbf{A}_2 = \mathbf{K_{int}}(\mathbf{R^T}| - \mathbf{R}^T\mathbf{t})$$