COV3 – Computer Vision
Camera and Imaging Systems

GERALD ZWETTLER

SOME CAMERA CALIBRATION GRAPHICS FROM AIP2/BVA2 LECTURE OF PROF. WERNER BACKFRIEDER

Topics

Mathematical Basics

Registration and Equation System Solving

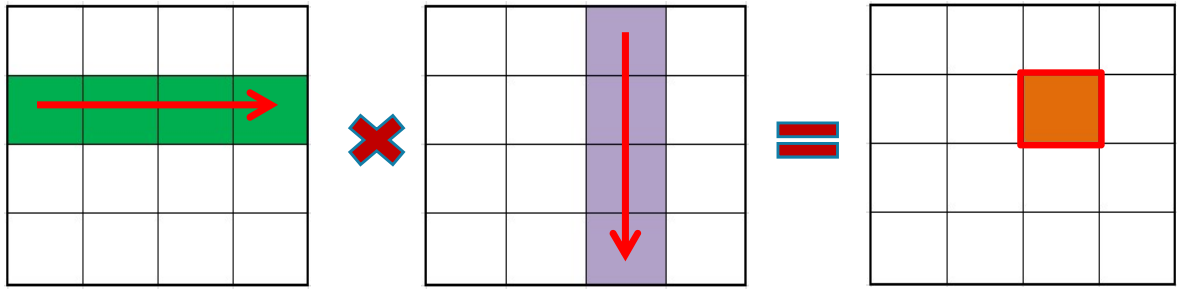
The Pinhole Camera (camera obscura)

Camera Calibration

Mathematical Basics

Matrix Operations

- Multiplication
  - To multiply matrix ***A*** by matrix ***B***, the number of columns in A and the number of rows in matrix B must match. This is always given for a 4x4 transformation matrix used for *affine transformations* (*translation, rotation, scale, shearing*)
  - Multiplication: a row from A is position-wise multiplied by a column of B, giving the overlapping position in result matrix ***C***. Operation is not commutative
  - Thus, dimensionality of result matrix C is determined by columns of A and rows in B.



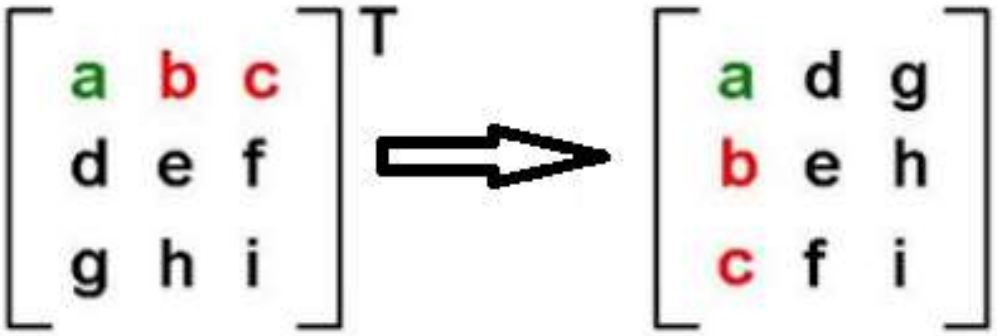
$$\text{mult: } \begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix} \times \begin{pmatrix} i & j \\ k & l \\ m & n \end{pmatrix} = \begin{pmatrix} w & x \\ y & z \end{pmatrix}$$

$$\text{example: } \begin{pmatrix} 1 & 2 & 3 \\ \dots & \dots & \dots \\ \dots & \dots & \dots \end{pmatrix} \times \begin{pmatrix} \dots & 4 & \dots \\ \dots & 5 & \dots \\ \dots & 6 & \dots \end{pmatrix} = \begin{pmatrix} \dots & 1*4+2*5+3*6 & \dots \\ \dots & \dots & \dots \\ \dots & \dots & \dots \end{pmatrix} = \begin{pmatrix} \dots & 32 & \dots \\ \dots & \dots & \dots \\ \dots & \dots & \dots \end{pmatrix}$$

Matrix Operations cont'd

Transpose

- transpose a matrix: $A' \equiv A^T$
  - The matrix that is obtained by interchanging the rows and columns of a matrix
  - Example: $A = \begin{bmatrix} 0 & 1 \\ 2 & 3 \\ -4 & -1 \end{bmatrix} \Rightarrow A^T = \begin{bmatrix} 0 & 2 & -4 \\ 1 & 3 & -1 \end{bmatrix}$
  - Common rules of transpose:
    - $(A + B)^T = A^T + B^T$
    - $(A^T)^T = A$
    - $(A\,B)^T = B^T A^T$
  - Symmetry of matrices:
    - $A^T = A$



Matrix Operations cont'd

Determinant

- used to check if an equation system represented in a quadratic coefficient matrix has a unique solution ($\det(A) = 0$) or if the matrix can be inverted ($\det(A) \neq 0$).
- calculation of determinant only "straight forward" up to size 3x3.
  - $A = [a_{1,1}] = [a], \det(A) = a$
  - $A = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \det(A) = a \cdot d - b \cdot c$
  - $A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}, \det(A) = a \cdot e \cdot i + b \cdot f \cdot g + c \cdot d \cdot h - c \cdot e \cdot g - f \cdot h \cdot a - i \cdot b \cdot d$
  - Calculation of determinant for arbitrary quadratic matrices via *Leibniz formula*.
  - Iterative calculation of determinant for arbitrary quadratic matrices via *Gaussian elimination algorithm*
- in case of multiplication on $n \times n$ matrics: $\det(A \cdot B) = \det(A) \cdot \det(B)$
- if matrix A is invertible, then $\det(A^{-1}) = \det(A)^{-1}$
- the determinant is invariant in case of transposing the matrix, thus $\det(A) = \det(A^T)$

Matrix Operations cont'd

Invert

- if a quadratic matrix is invertible, then identity matrix ***I*** (ones at first diagonal, zeros else) results

from $A \cdot A^{-1} = A^{-1} \cdot A = I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & 1 \end{bmatrix}$

- in 2D case: $A = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, A^{-1} = \frac{1}{\det(A)} \cdot \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$

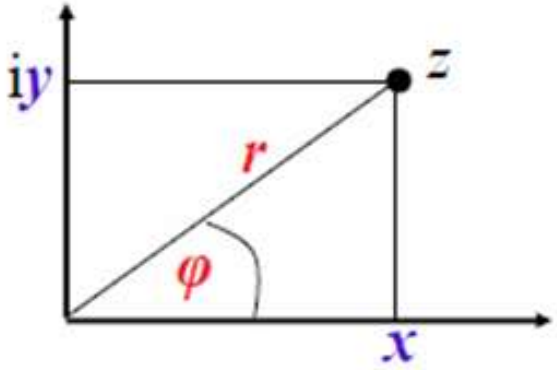
- in 3D case: $A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}, A^{-1} = \frac{1}{\det(A)} \cdot \begin{bmatrix} ei - fh & ch - bi & bf - ce \\ fg - di & ai - cg & cd - af \\ dh - eg & bg - ah & ae - bd \end{bmatrix}$

- general calculation e.g. utilizing *Gaussian elimination algorithm*
- *useful for re-forming mathematical equations, e.g. Ax=B ➔ $x=A^{-1}B$*

Matrix Operations cont'd

Complex Conjugate

- matrices of size $n \times m$ containing complex numbers can be complex conjugated
  - thereby, the matrix is first transposed
  - then, all entries are complex conjugated
  - matrix conjugate transpose denoted as *Hermitian transpose*
  - conjugation of a complex number $z = x + iy, z \in \mathbb{C}$ refers to conserving the real part and inverting the sign of the imaginary part, thus $\bar{z} = x - iy$
    - conjugation of complex numbers used to find roots of polynomials
  - complex conjugate of matrix $A^* = \bar{A}^T$
    - example: $A = \begin{bmatrix} 1 & 2 & 3+i \\ 4-i & 5 & 6 \end{bmatrix}, A^* = \begin{bmatrix} 1 & 4+i \\ 2 & 5 \\ 3-i & 6 \end{bmatrix}$



polar coordinates

Matrix Operations

- **Affine transformations in 2D**, such as *translation, rotation, scale* can be represented by 3x3 matrix multiplication. Besides translation, all operations could be expressed by 2x2 matrices
  - To extend from 2x2 to 3x3 matrices, input coordinates (x,y) lack a third dimension ➔ homogeneous coordinates, adding 1 as scale-neutral 3th component (*c.f. homography*).
  - Affine transformations: parallelism and proportion ratios are conserved

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Scaling

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

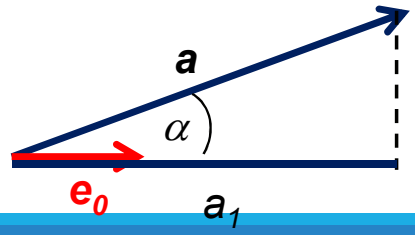
Rotation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & sh_x & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Shearing

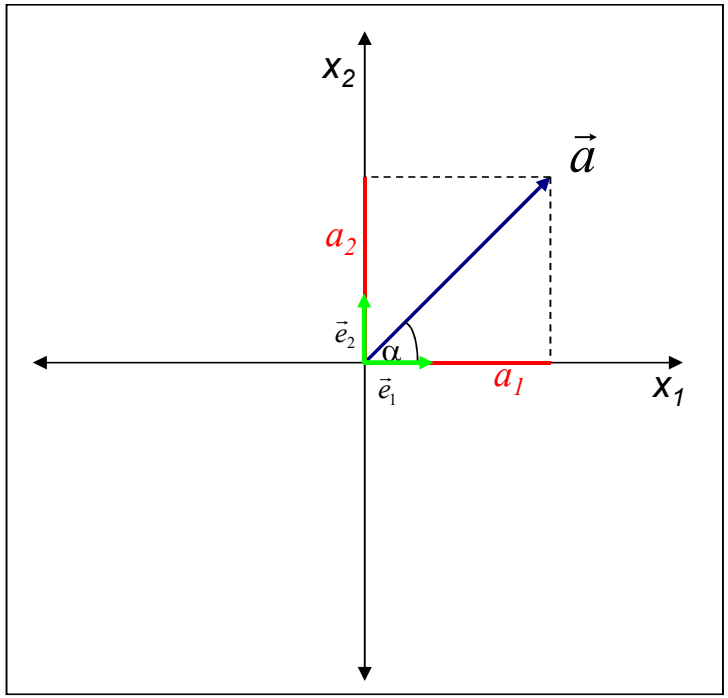
Coordinates and Vectors in Images

- **Vector**
  - entity defined by its direction and magnitude
  - independent from actual coordinate system, thus the same in *Cartesian* or *Polar* coordinates
  - definition of a vector: projection onto the base vectors of the coordinate system
  - a coordinate system is build by its base vectors, all pointing from the origin in direction of the coordinate system axis
  - with all base vectors, i.e. axis, mutually being **orthogonal** (right angle) and showing length of **one**, the coordinate system is called *orthonormal system*. This is true for *Cartesian* coordinate system.
  - scalar product: required for axis projection with
    - $a_1 = \overrightarrow{a} \cdot \overrightarrow{e_o} = |\vec{a}| \cdot |\overrightarrow{e_0}| \cos(\alpha)$ and in case of unit length $|\overrightarrow{e_0}| = 1$ then $a_1 = |\vec{a}| \cdot \cos(\alpha)$



Coordinates and Vectors in Images

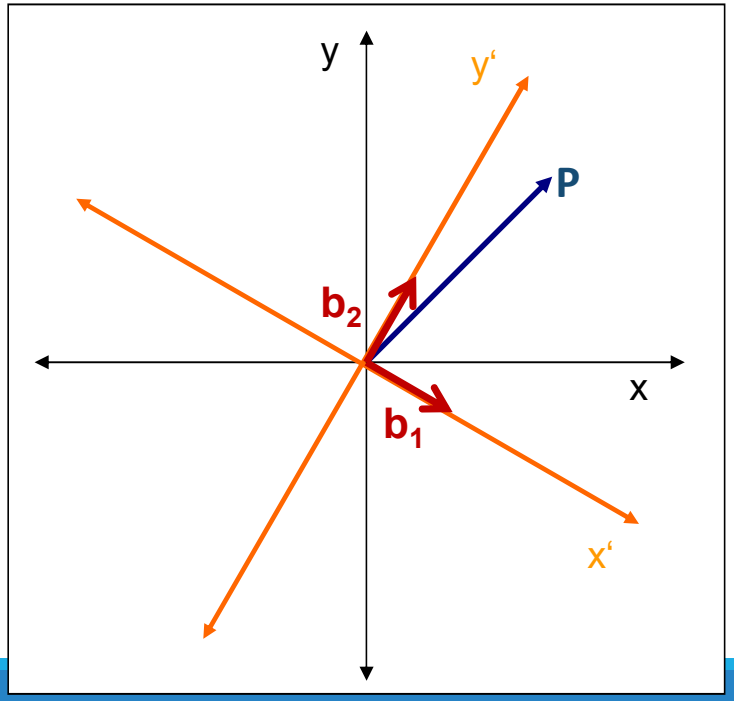
- **Coordinates**
  - Cartesian coordinates in a 2D plane can be represented by projection of the vector onto the coordinate axis.
  - the components a_1 and a_2 of a vector a, cf. *x/y* for 2D case, correspond to the inner product with the base vectors e_1 and e_2 respectively. These axis generally are denoted as x-axis and y-axis in 2D case.
  - with $a_i = \overrightarrow{a} \cdot \overrightarrow{e_i} = |\vec{a}| \cdot |\overrightarrow{e_i}| \cos(\alpha)$ the vector a can be composed by vector addition of its components per dimension $\vec{a} = \sum_{i=1}^{n} a_i \cdot \overrightarrow{e_i}$ with $n = 2$ for 2D case



according to AIP lecture by Prof. Werner Backfrieder

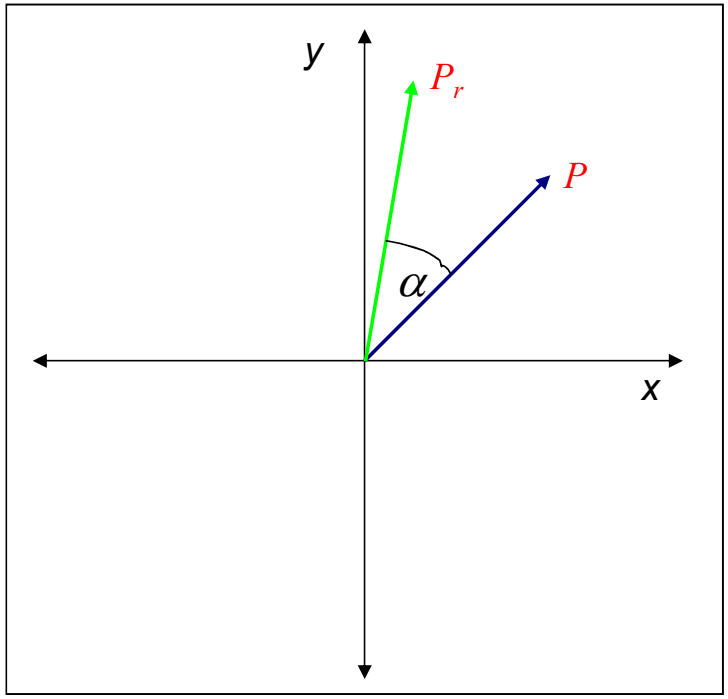
Coordinates and Vectors in Images

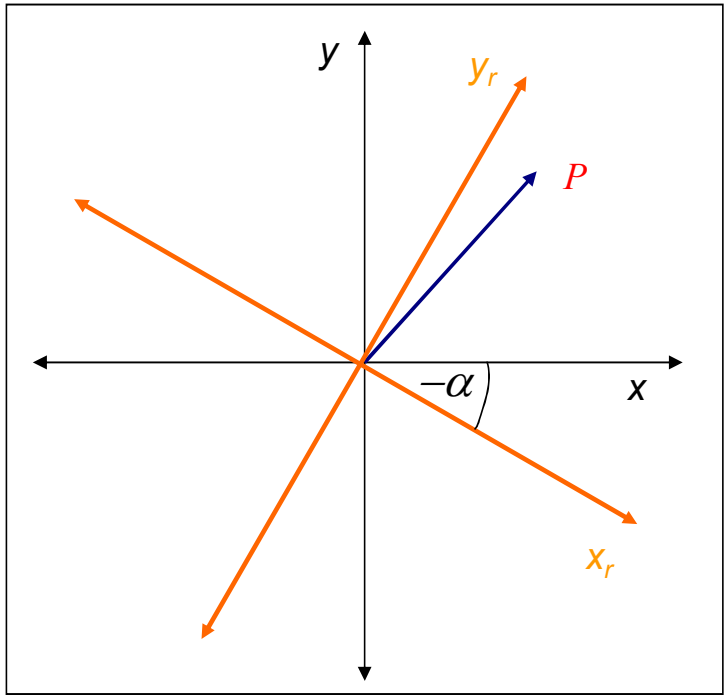
- **Coordinate Transform**
  - To rotate a point P, it can be transformed between the coordinate bases $\{x, y\}$ and $\{x', y'\}$
  - The coordinates of P referred to the base $\{x', y'\}$ are thereby expressed by the inner products with the base vectors $\boldsymbol{b_1}$ and $\boldsymbol{b_2}$ in $\{x, y\}$ representation.
  - the transform can be expressed as $\vec{p}' = U^T \cdot \vec{p}$, where the columns of U hold the base vectors
  - thus, matrix notation allows a compact expression of the rotation transform.



Coordinates and Vectors in Images

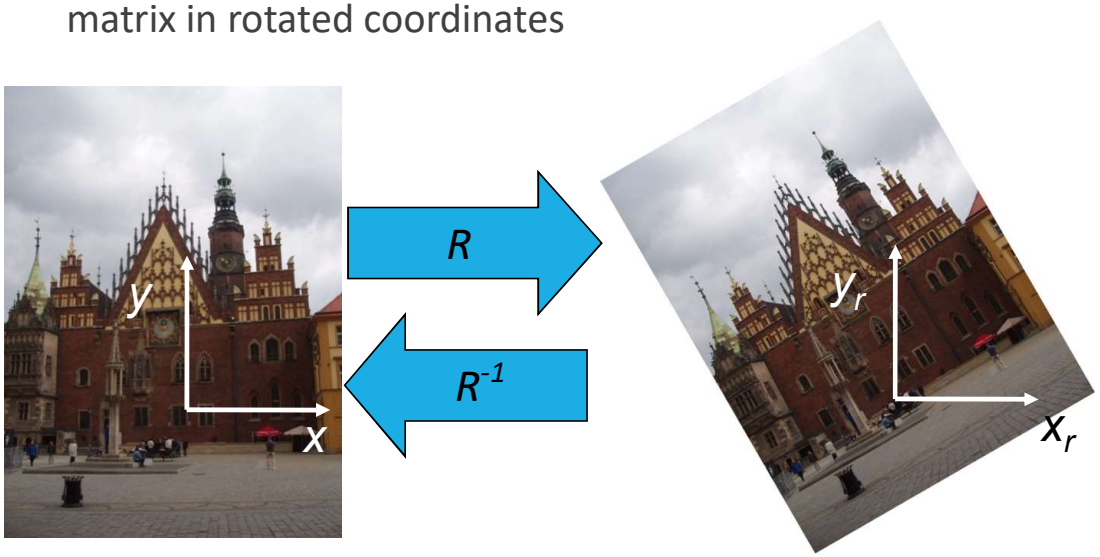
- **Coordinate Transform cont'd**
  - a counter-clockwise (CCW) rotation of a point, i.e. vector, can be achieved at two different ways:
    - **(a)** by keeping the base $\{x, y\}$ and calculating new coordinates for $\vec{p}'$ by rotating around the origin by angle α.
    - or **(b)** by keeping $\vec{p}'$ fixed at position $\vec{p}$ but rotating the whole coordinate system into opposite direction (by $-\alpha$). Thereby, the rotation is converted into a transform of P to the rotated coordinate system $\{x_r, y_r\}$.

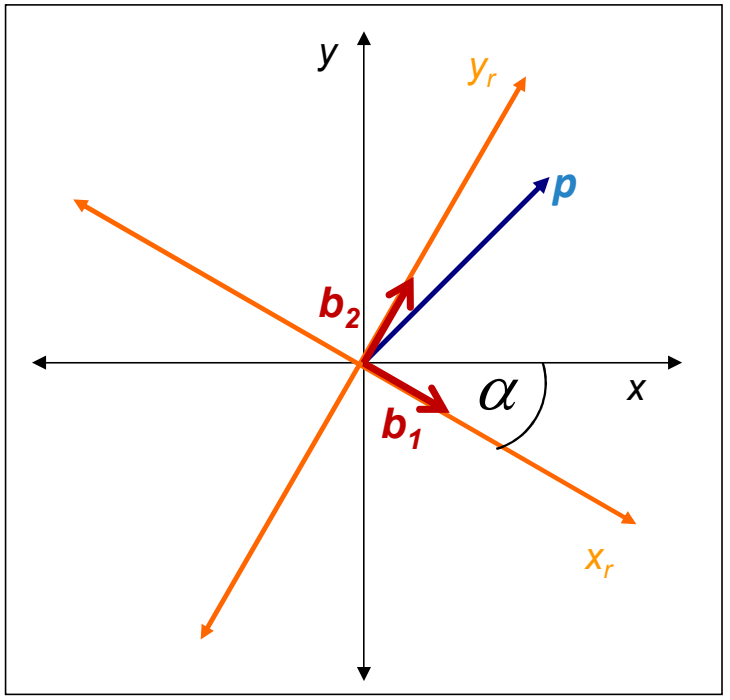




Coordinates and Vectors in Images

- Coordinate Transform cont'd
  - proof, that planar rotation can be calculated both way with identic outcome:
    - new coordinate system $\{b_1, b_2\}$ with $b_1 = \begin{pmatrix} \cos\alpha \\ -\sin\alpha \end{pmatrix}$, $b_2 = \begin{pmatrix} \sin\alpha \\ \cos\alpha \end{pmatrix}$, thus $U = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}$ and $U^T = R = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}$
    - thus, $\vec{p}_r = R \cdot \vec{p} = \begin{pmatrix} x \cdot \cos\alpha - y \cdot \sin\alpha \\ x \cdot \sin\alpha + y \cdot \cos\alpha \end{pmatrix}$ for $\vec{p} = \begin{pmatrix} x \\ y \end{pmatrix}$.
    - rotation matrix R thereby corresponds to the transform matrix in rotated coordinates





Matrix Operations cont'd

- **Affine transformations in 3D**, such as translation, rotation, scale can be represented by 4x4 matrix multiplication. Besides translation, all operations could be expressed by 3x3 matrices
  - To extend from 3x3 to 4x4 matrices, input coordinates (x,y,z) lack a fourth dimension ➔ homogeneous coordinates, adding 1 as scale-neutral 4th component.

X-Rotation in 3D

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Z-Rotation in 3D

$$\begin{bmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scale in 3D

$$\begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

for 3D Transformations, the rotation matrices T_x, T_y and T_z are required for operations around the particular axis

Y-Rotation in 3D

$$\begin{bmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Translation in 3D

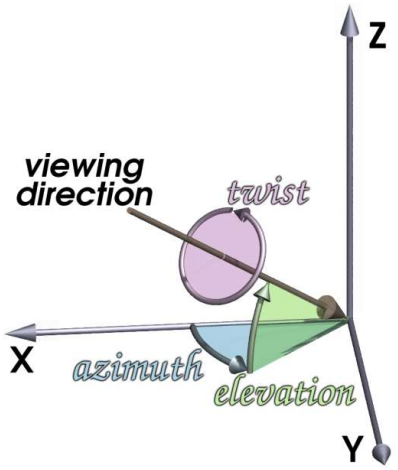
$$\begin{bmatrix} 1 & 0 & 0 & Tx \\ 0 & 1 & 0 & Ty \\ 0 & 0 & 1 & Tz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

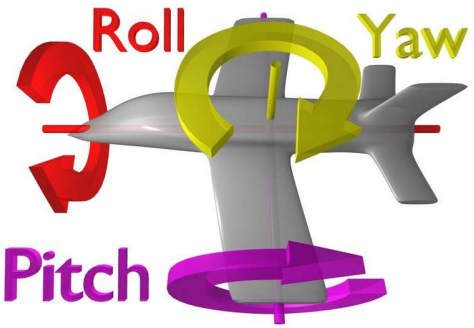
Matrix Multiplication

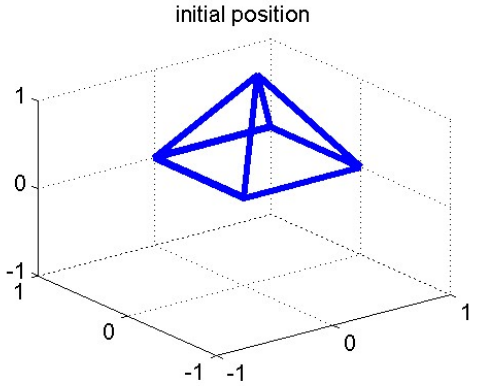
$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ q \end{bmatrix}$$

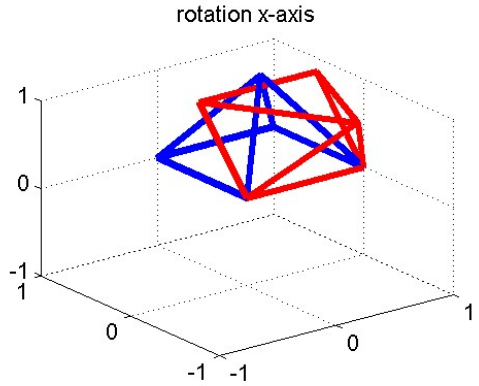
Matrix Operations cont'd

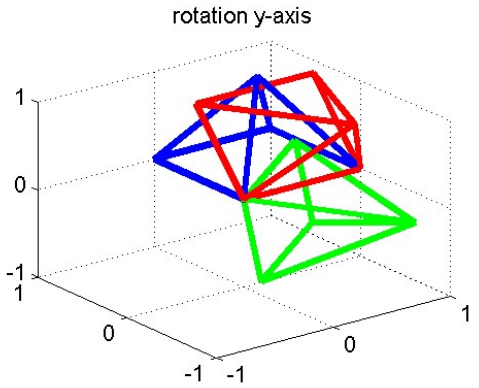
- **Rotation:** rotation around several stationary, i.e. fixed, axes represented as matrix multiplication with $\overrightarrow{x_r} = T_x \cdot T_y \cdot T_z \cdot \vec{x}$ (**extrinsic**)
  - example with $\phi_{T_x} = 45°$, $\phi_{T_y} = 30°$ and $\phi_{T_z} = 70°$
  - with rotations around fixed axis the outcome is not intuitive at all
  - thus, rotations around the object's axis instead are to be favored, utilizing *Euler angles* for **azimuth, elevation** and **spin** or **yaw, pitch** and **roll** as common in aviation

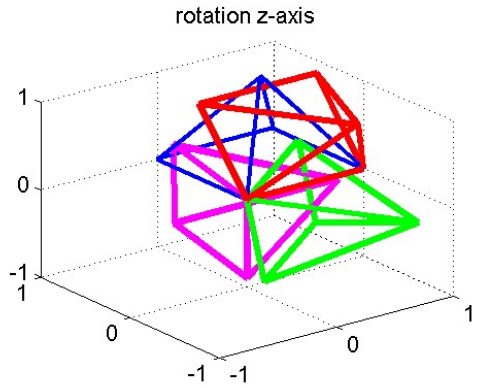






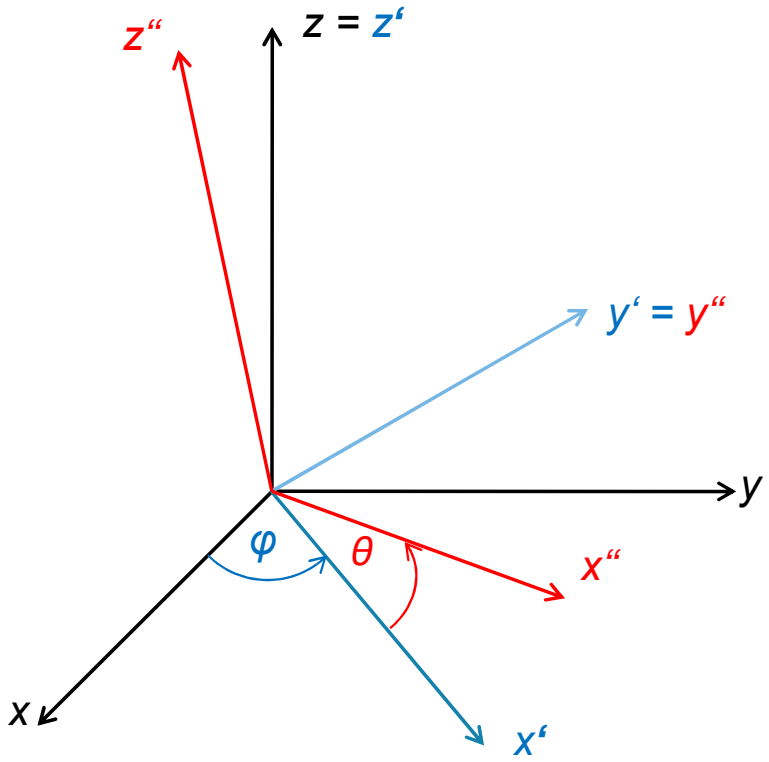






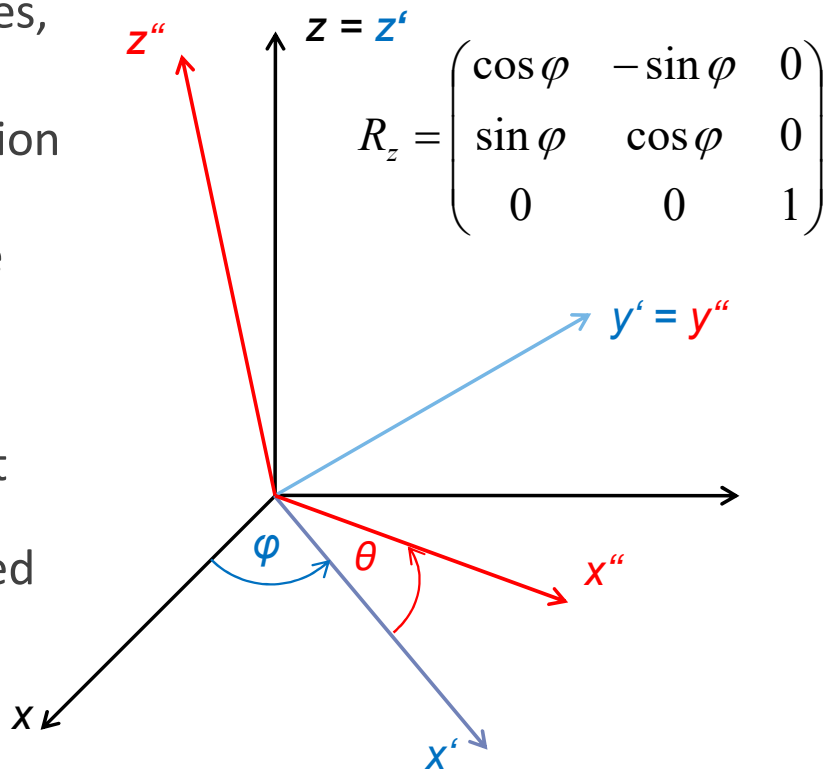
Object-Related Rotation (intrinsic)

- example with 2 object-related Euler angle rotations, namely azimuth phi φ and elevation theta ϑ. The rotation convention is defined with zy'.
  - rotation order
    - first rotation around z, leading to x', y' and conserved $z' = z$ (primed coordinate system all in blue)
    - then, rotation around y' axis transforms into the red double-primed coordinates with x'', $y'' = y'$ and z''



Object-Related Rotation cont'd

- evaluation of the example for point $\vec{p}$ leading to $\vec{p}' = R_z \cdot \vec{p}$
- but now not common R_y is applicable, as the y-axis has changes, thus $\vec{p}'' = R_{y'} \cdot \vec{p}'$
- the first rotation is executed around R_z while the second rotation is performed around $R_{y'}$ (*thus* $R = R_{y'} \cdot R_z$). To get the entire transformation, $R_{y'}$ needs to be expressed in "un-primed" base coordinate system as follows:
  - $R_{y'} = R_z \cdot R_y \cdot {R_z}^T$
  - entire transform $R = R_z \cdot R_y \cdot {R_z}^T \cdot R_z$ now as we know that ${R_z}^T \cdot R_z = I, \quad I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & 1 \end{bmatrix}$ the transform can be expressed by basic transform matrices in inverse order, namely
$$\boldsymbol{R = R_z \cdot R_y}$$
  - ***conclusion***: with object-related rotation the partial rotation matrices are applied in **inverse** order of the actual physical rotation.



Matrices for Solving Overdetermined Equation Systems

- in case of registration or calibration tasks, the problem is often defined by overdetermined equation systems resulting from n reference points stabilizing the outcome and approximation of a transformation matrix R.
  - an equation system generally is overdetermined, if there are more non-redundant equations compared to the number of variables to solve or approximate.
  - thus, we need to solve variables $x_1, x_2, \dots, x_n$ given some equations that contain some or all of these variable
  - for n variables given m equations with $m \gg n$ one can express:

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &\approx b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &\approx b_2 \\
\cdots \quad + \quad \cdots \quad + \cdots + \quad \cdots \quad &\approx \cdots \\
a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n &\approx b_i \\
\cdots \quad + \quad \cdots \quad + \cdots + \quad \cdots \quad &\approx \cdots \\
a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &\approx b_m
\end{aligned}
\quad \Rightarrow \boldsymbol{Ax} \approx \boldsymbol{b}
$$

for coeff. matrix A, solution vector $\boldsymbol{x}$ and $\boldsymbol{b}$

Matrices for Solving Overdetermined Equation Systems cont'd

- the equations generally do not allow a unique and accurate solution, thus an error r_i per equation is introduced with e.g.:
  - $r_1 = a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n - b_i$ as exact representation
- the overall error as function of squared differences $F(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{m} r_i^2$ thereby has to be minimized to get the best approximation.
- to get a solvable form: $A \cdot x = b$
  $\boldsymbol{A^T \cdot A \cdot x = A^T \cdot b}$, the *Gaussian normal equation cf. slide 6.*

$$x = (A^T \cdot A)^{-1} \cdot A^T \cdot b$$

- the Gaussian normal equation can be solved utilizing an extension of the „*Falk schema*“ of matrix multiplication
- this way overdetermined equation systems can be approximately solved. Nevertheless, inaccuracies due to matrix multiplication on real-value precision prevent this concept to be used for “really large” equation systems, e.g. reconstructing 3D computed tomography with $\approx 800{,}000$ equations per 512×512 slice

Matrices for Solving Overdetermined Equation Systems cont'd

- example with $\begin{bmatrix} 3 & 1 \\ -1 & -3 \\ 1 & 2 \\ -2 & 3 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \approx \begin{bmatrix} 8.1 \\ 0.2 \\ 0.9 \\ 2.9 \end{bmatrix}$ with $x_1 \approx 3$ and $x_2 \approx -1$ at some noise level
- approximation leads to $\begin{bmatrix} 15 & 2 \\ 2 & 23 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \approx \begin{bmatrix} 42.8 \\ -17.4 \end{bmatrix}$
- solving:
  - with Ax=B ➔ x=A^{-1}B
  - $A^{-1} = \begin{pmatrix} 0.06745 & -0.00587 \\ -0.0059 & 0.04399 \end{pmatrix}$
  - $A^{-1}B = \begin{pmatrix} 2.988856 \\ -1.016422 \end{pmatrix}$
  - thus $x_1 \approx \mathbf{2.988856}$ and $x_2 \approx \mathbf{-1.016422}$

$A =$

3	1	8,1
-1	-3	0,2
1	2	0,9
-2	3	-8,9

$= b$

$A^T =$

3	-1	1	-2	15	2	42,8
1	-3	2	3	2	23	-17,4

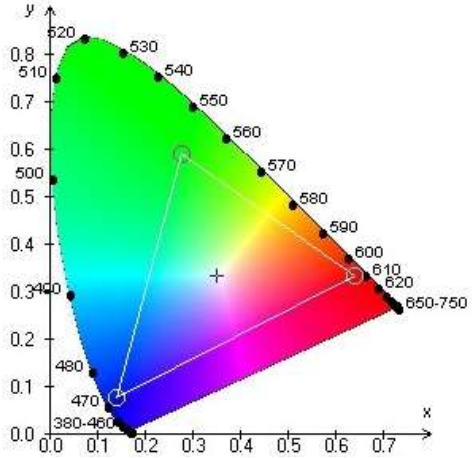
$= A^T \cdot b$

$= A^T \cdot A$

Camera Calibration

Initial Situation with Imaging Systems in General

- each imaging system suffers from inherent systematic errors, e.g. distortions, no color-truth, etc.
  - color-truth: brightening or darkening, no unique color representation and reproduction utilizing RGB ➔ CIE Lab diagram for unique definition of colors according to their wave lengths
  - focused projection at certain depth
  - image defects caused by aberrations of the lens system, e.g. rays in the outer section of the lens experiencing more refraction than those in the very center
- thus, camera calibration is needed to:
  - accurate assessment and measurement of the image data, e.g. forensic investigation
  - regularization of portrait photos, distortion correction

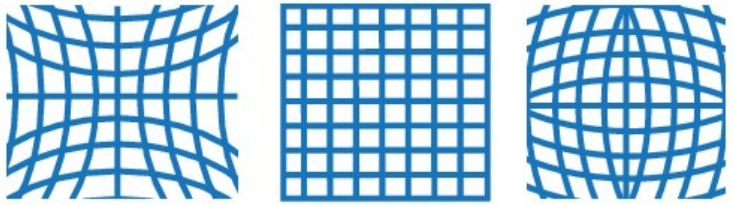


Camera Calibration

Fields of Application

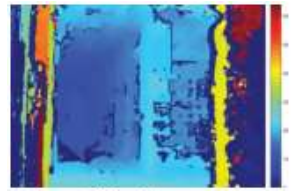
- camera calibration comprises the following aspects
  - *lens distortion*: while a pinhole camera system as theoretical concept does not address lens distortion at all, in real world optics a lens can not be perfectly manufactured, thus leading to different kinds of distortion
    - *pincushion*: negative radial distortion
    - *barrel*: positive radial distortion
    - furthermore, possible irregular local distortions (relevant for medical imaging systems)
  - planar 2D images lack stereoscopy and are not faithful w.r.t. to real-world distances
    - *reconstruct surface from several planar images*
    - *measure size of objects if calibrated for the particular distance*
  - combination of multiple camera systems
    - *e.g. Kinect with separate RGB and depth cam. Knowledge of physical construction and general lens distortion insufficient. Precise measurements necessitate camera calibration.*
    - *estimate depth with 3D cams*

Remove Lens Distortion




Measure Planar Objects


Estimate 3-D Structure from Camera Motion

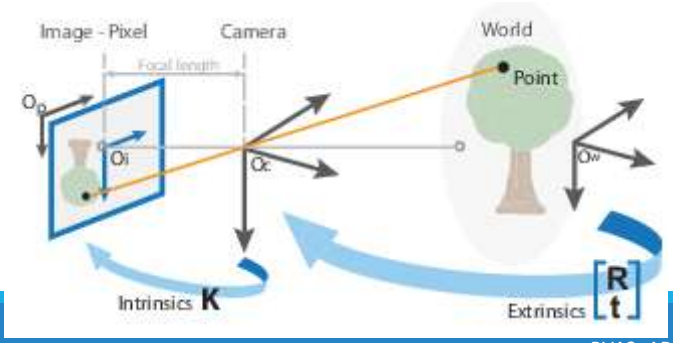

Estimate Depth
Using a Stereo Camera

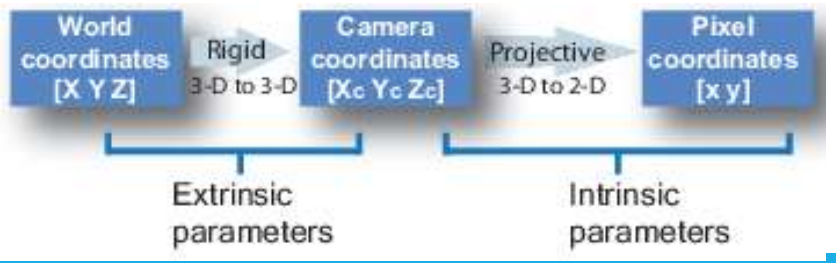
images from Mathworks available from https://de.mathworks.com/help/vision/ug/camera-calibration.html#bu1ag7t

Camera Calibration

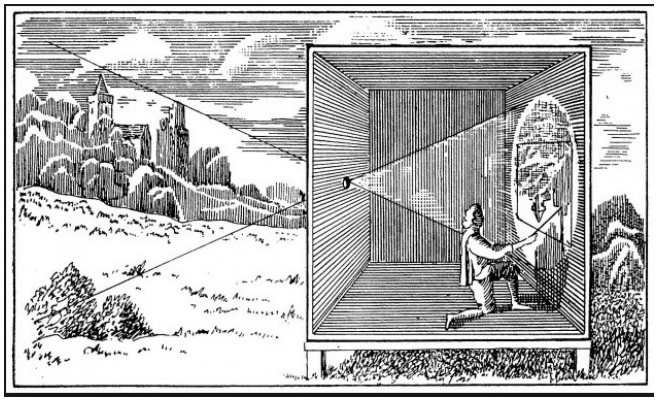
General Definition

- for camera calibration, static and regular test patterns of known geometry and size, such as a black/white chessboard texture, are utilized to determine the following coefficients based on a sufficient number of reference points derived from the test pattern
  - *intrinsic* parameters/coefficients (optical center and focal length for projection from 3D camera coordinates to 2D image plane)
  - *extrinsic* parameters/coefficients (rotation and translation in the real world relative to 3D camera coordinates). Calibration with intrinsic and extrinsic parameters introduced by [Zhang 2000].
  - *distortion* parameters/coefficients: to be additionally considered in the camera calibration process, see [Heikkila and Silven 1997] or [Tsai 1986] for barrel- and pincushion-shaped distortions.
- result of camera calibration is the knowledge of intrinsic, extrinsic and optionally the distortion coefficients, thus allowing mapping from real 3D world to planar image plane utilizing matrix operations



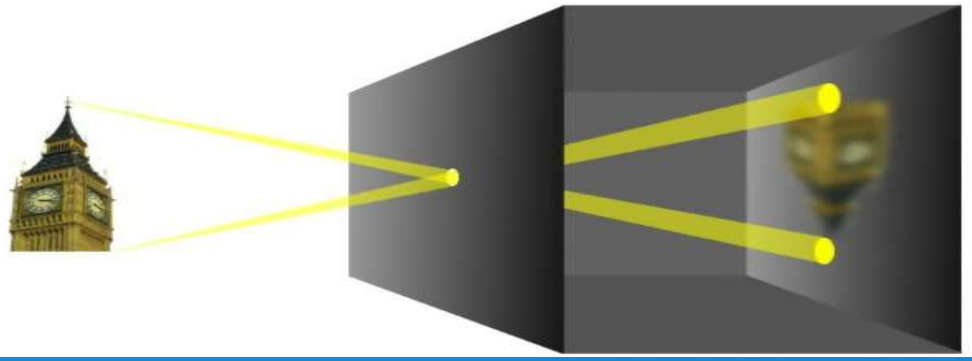


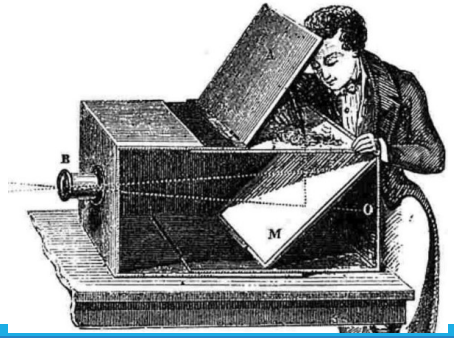
Camera Obscura



The pinhole camera – *camera obscura*

- most basic imaging system, known since ancient times
- light shines onto the back wall of a dark box or cabin/room.
- the object is reproduced (pictured) on the back image plane
- size of the pinhole balances between sharpness and brightness
  - each point of the world (like top of *"Elisabeth Tower"* including comprising *"Big Ben"*) of minimal size contributes to a circular area of the image plane depending on size of the pinhole
  - thus, as several color rays of real world contribute for each point on the image plane, blurring is introduced
- resulting images are inverted, i.e. mirrored top-down and left-right
- ratio of optical axis vs. distance from pinhole to the image plane determines wether the object is pictured larger or smaller compared to true size.

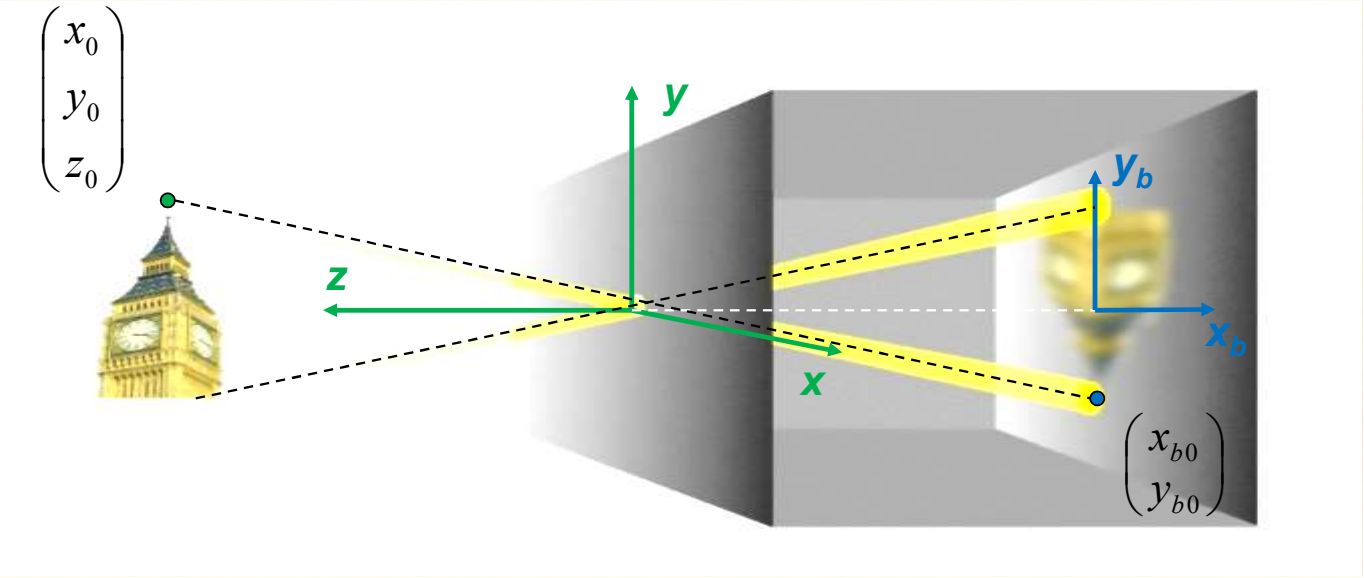




Camera Obscura

Coordinate Systems

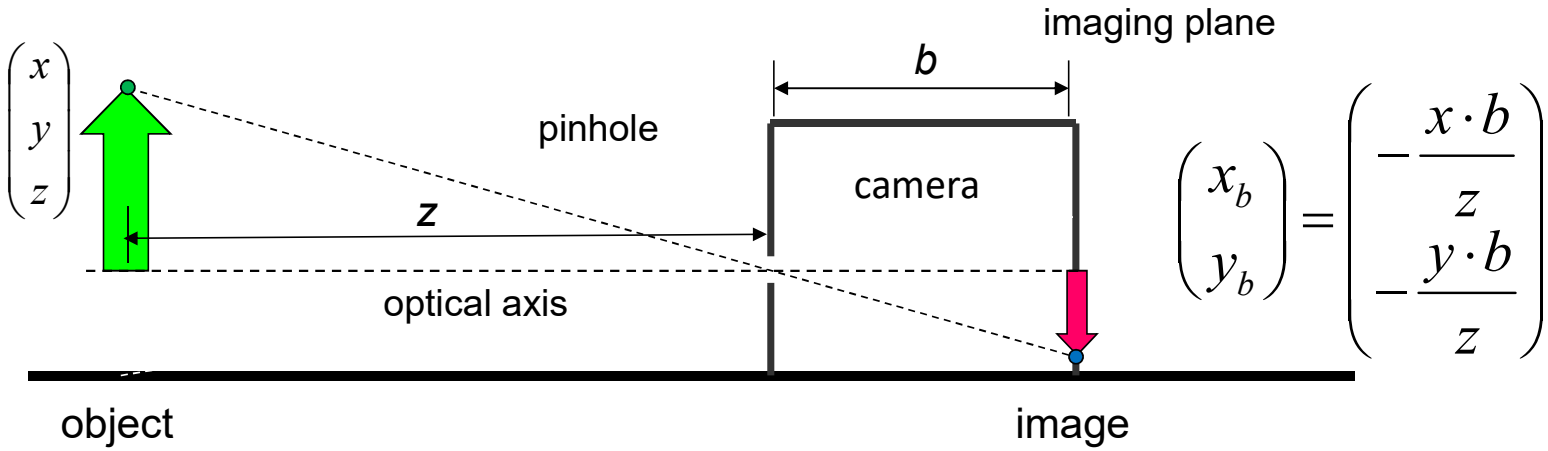
- **camera coordinates x, y, z**
  - fixed to the camera, 3D, orthonormal and right-handed
- **image coordinates x_b, y_b**
  - linked to the image plane, 2D, reflect projection geometry 3D➔2D



Camera Obscura

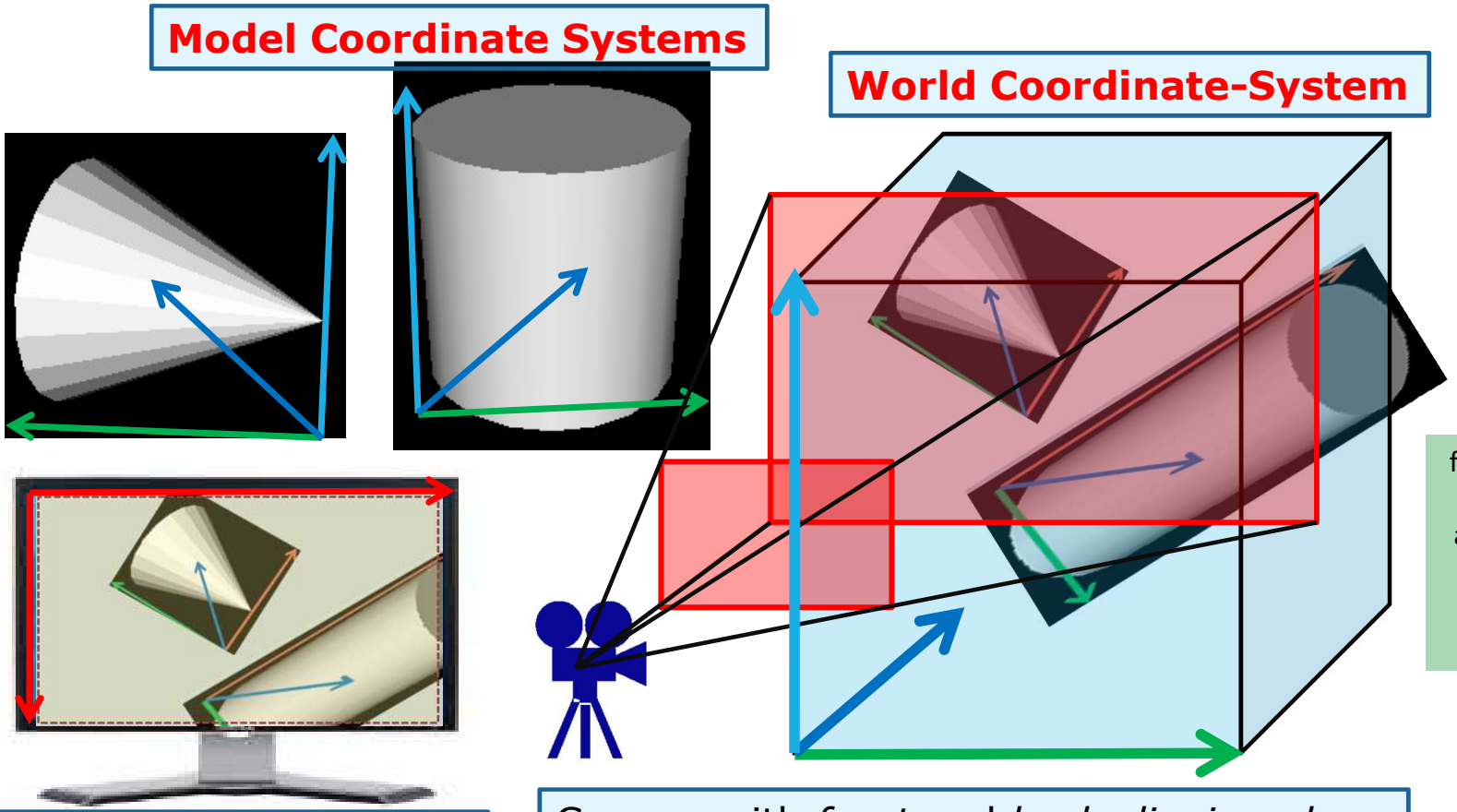
Projection Geometry of the Imaging System

- negative sign, i.e. mirroring/inversion around the optical axis
- in case $z < b$, thus distance to the object smaller than to the image
  - 2D image of the object is enlarged
- in case $z > b$, thus distance to the object larger than to the image
  - 2D image of the object is scaled down



Excursus: Coordinate Systems in Computer Graphics

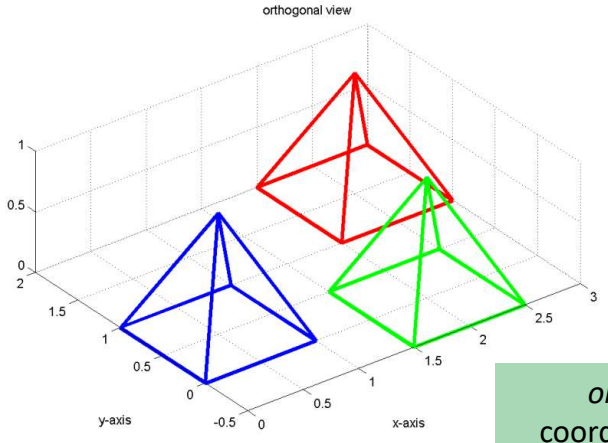
generally: perspective coordinate systems, showing divergent rays



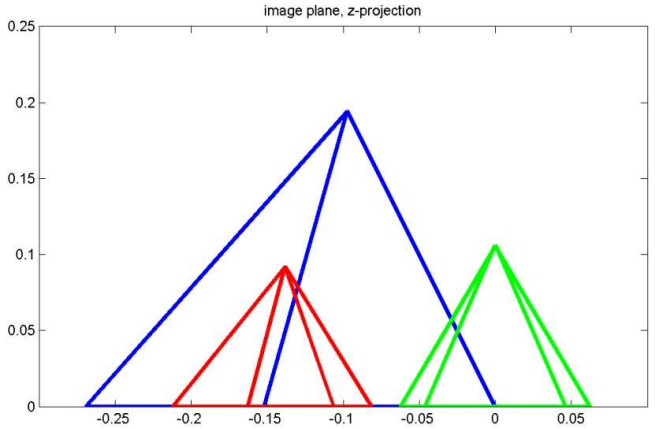
from computer graphics point of view: image plane on viewing axis generally between near/far clipping plane, thus no inversion/mirroring of the resulting 2D image

Excursus: Coordinate Systems in Computer Graphics cont'd

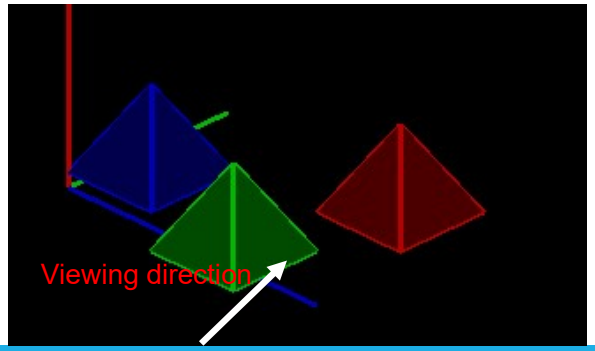
- in case of orthogonal coordinate systems, three pyramids of identical size are imaged at same size despite depth distance from the camera system, while for perspective coordinate systems size scales according to distance, c.f. human eye vision.

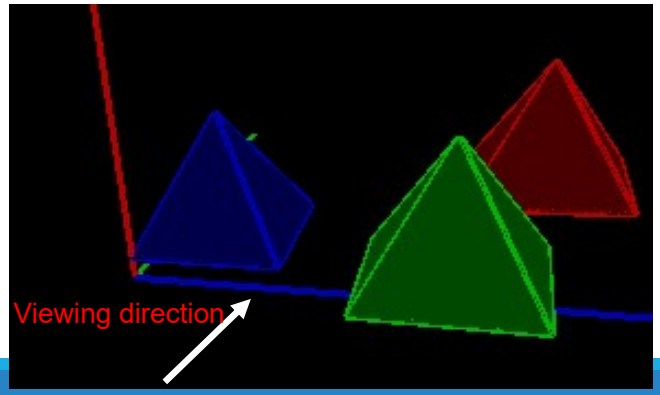


orthogonal coordinate system



perspective coordinate system; in case of orthonormal viewing direction: **central projection**

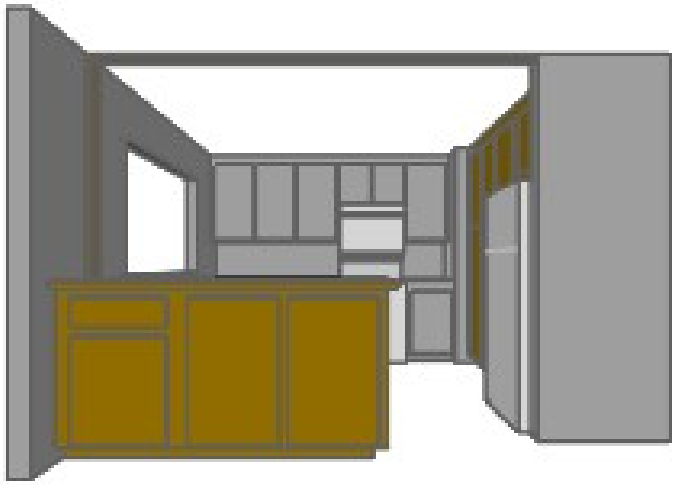


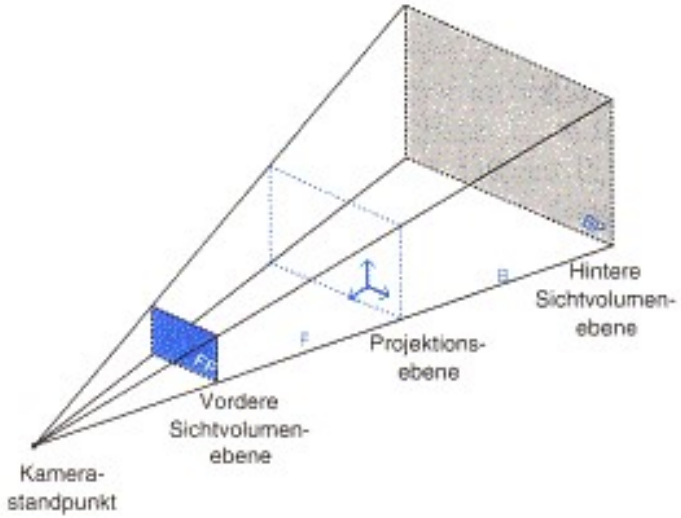


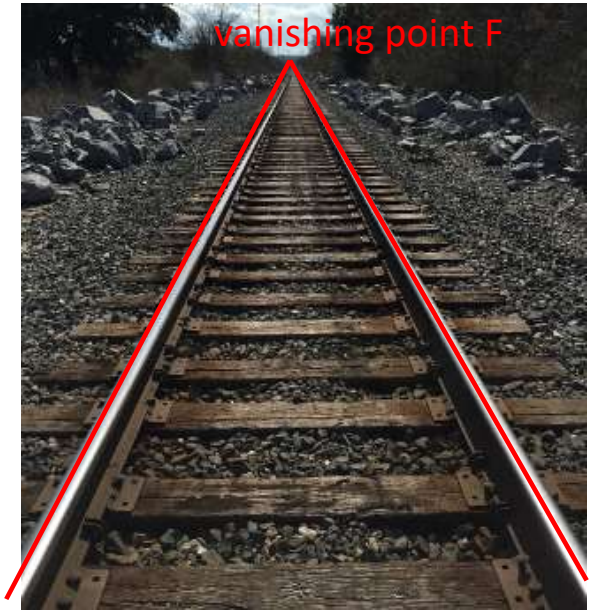
Excursus: Coordinate Systems in Computer Graphics cont'd

perspective coordinate systems in computer graphics and real world

- focus point as well as camera position or eye of human observer virtually at same point.
  - perspective imaging: observer and vanishing point both on focus line



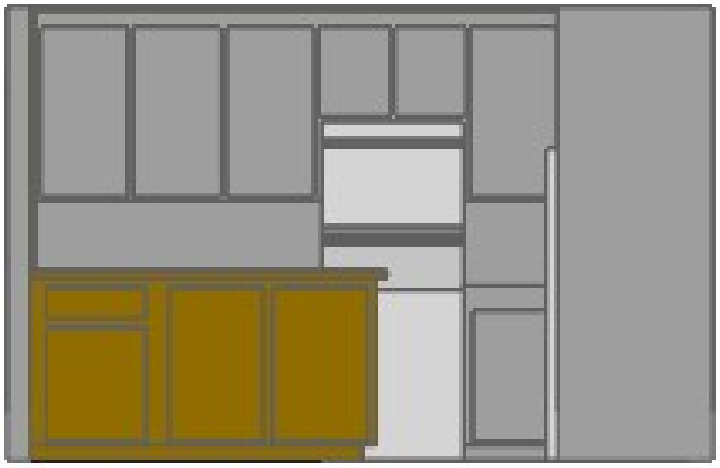


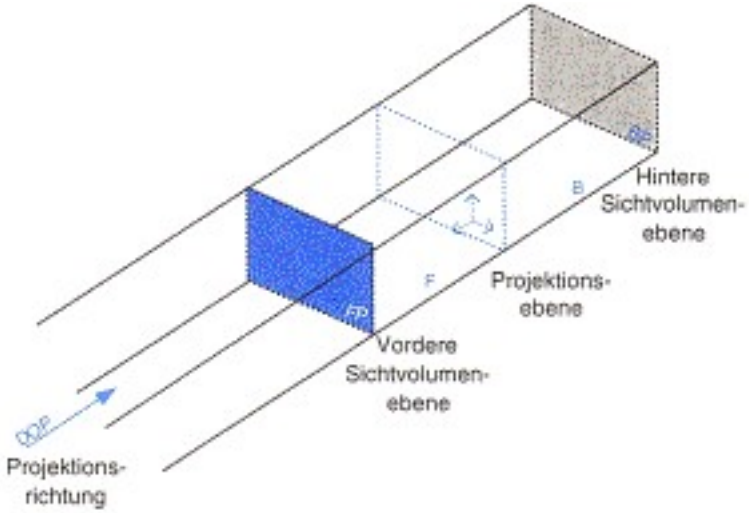


Excursus: Coordinate Systems in Computer Graphics cont'd

parallel coordinate systems in computer graphics and real world

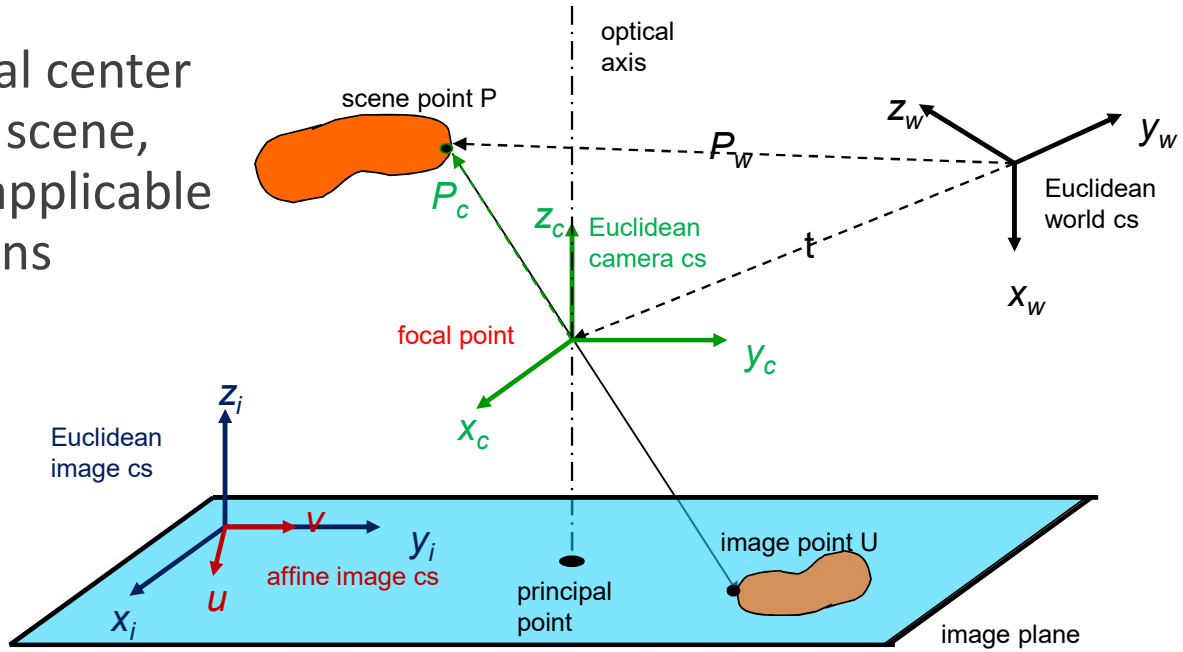
- (pinhole) camera in case of parallel projection in infinitive distance from the scene
  - thus, objects far away are imaged at the same size compared to near objects.
  - thus, the object distance cannot be derived from size





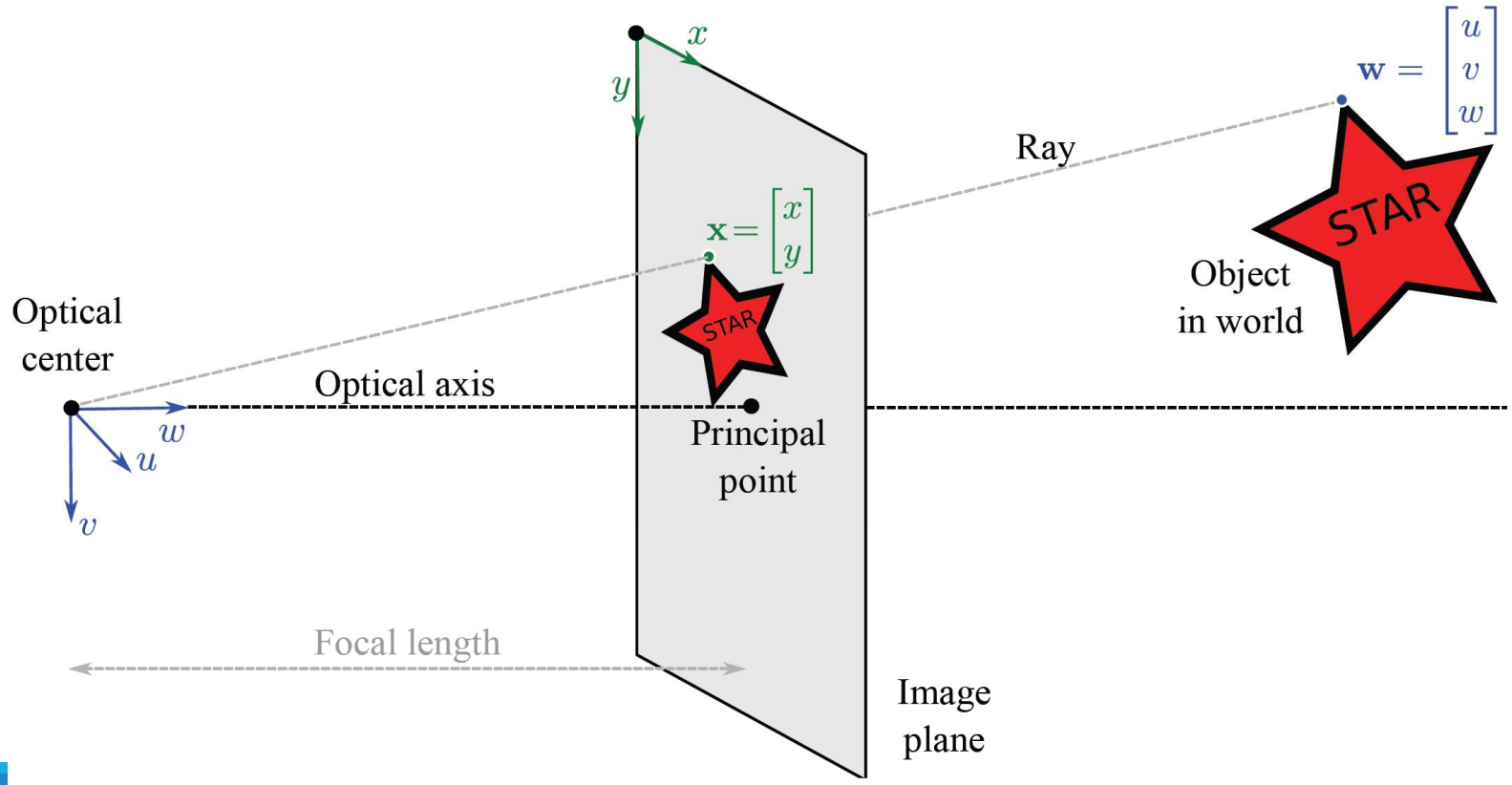
Camera Calibration

- several coordinate systems included, to transform Point P_w from world coordinate system to P_c in camera coordinate system before projecting to affine 2D image plane resulting in pixel position U.
  - given the rotation matrix R, the point can be transformed to the 3D camera coordinate system with $\overrightarrow{P_c} = \begin{pmatrix} u_c \\ v_c \\ w_c \end{pmatrix} = R \cdot (\overrightarrow{P_w} - \vec{t})$
  - furthermore, interpreting the optical center "behind" the image plane w.r.t. the scene, a simple pinhole camera model is applicable for camera calibration considerations



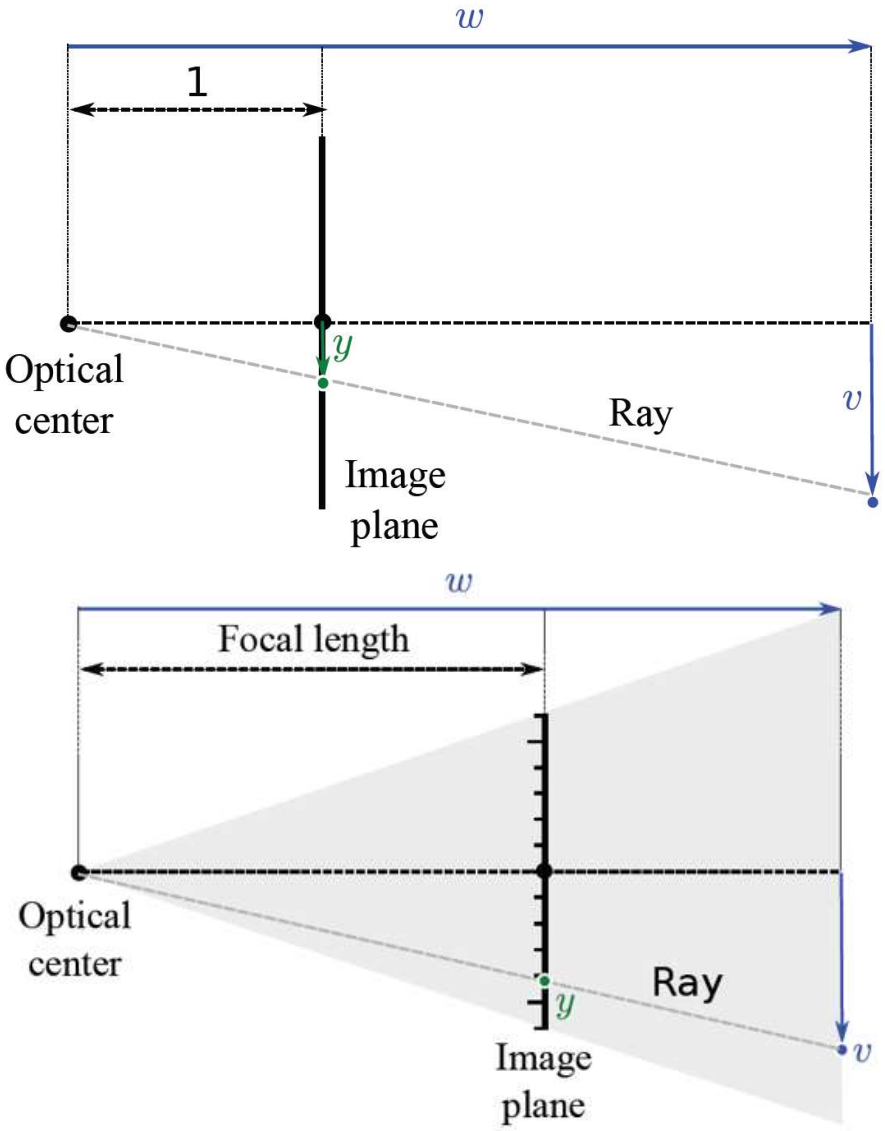
Camera Calibration cont'd

- pragmatic camera model with optical center behind the image plane
  - focal length determines size of the imaged objects, i.e. zoom behavior
  - (at first) no discrimination between world and camera 3D coordinate systems



Camera Calibration cont'd

- focal length parameters
  - considering a focal length of phi $\phi = 1$, the coordinates in the image plane are easily calculated utilizing triangle similarities with $x = \frac{u}{w}$ and $y = \frac{v}{w}$
  - for an arbitrary focal length, the coordinates are to be calculated with $x = \frac{\phi \cdot u}{w}$ and $y = \frac{\phi \cdot v}{w}$
  - as the receptors and images in practice are not necessarily square, utilization of two focal length parameters ϕ_x and ϕ_y is useful, thus $x = \frac{\phi_x \cdot u}{w}$ and $y = \frac{\phi_y \cdot v}{w}$.
  - utilizing this formula, the origin $(0,0)$, i.e. where the principal ray is orthonormal to the image plane, lies in the exact center and not in the top left corner as common for image data.
    Thus, a translation offset is introduced with $\boldsymbol{x = \frac{\phi_x \cdot u}{w} + \delta_x}$ and $\boldsymbol{y = \frac{\phi_y \cdot v}{w} + \delta_y}$.



Camera Calibration cont'd

- if the image planes are not perfectly perpendicular, i.e. orthonormal, some *skewness* is introduced (cf. affine transformation *shearing*).
  - skew-transform represents a rotation gamma $\boldsymbol{\gamma}$ with one fixed axis leading to rhombic shape by introducing coordinate-correlation. Thus, we get $x = \frac{\phi_x \cdot u + \boldsymbol{\gamma} \cdot \boldsymbol{v}}{w} + \delta_x$ and $y = \frac{\phi_y \cdot v}{w} + \delta_y$.
- as discussed before, a position $P_w = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$ from real world will generally not be expressed in the camera coordinate system. Thus transformation is required with $\overrightarrow{P_w}' = \begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = R \cdot \begin{bmatrix} u \\ v \\ w \end{bmatrix} + \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix}$

where transform matrix R is rather unknown with $R = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}$ and translation vector.

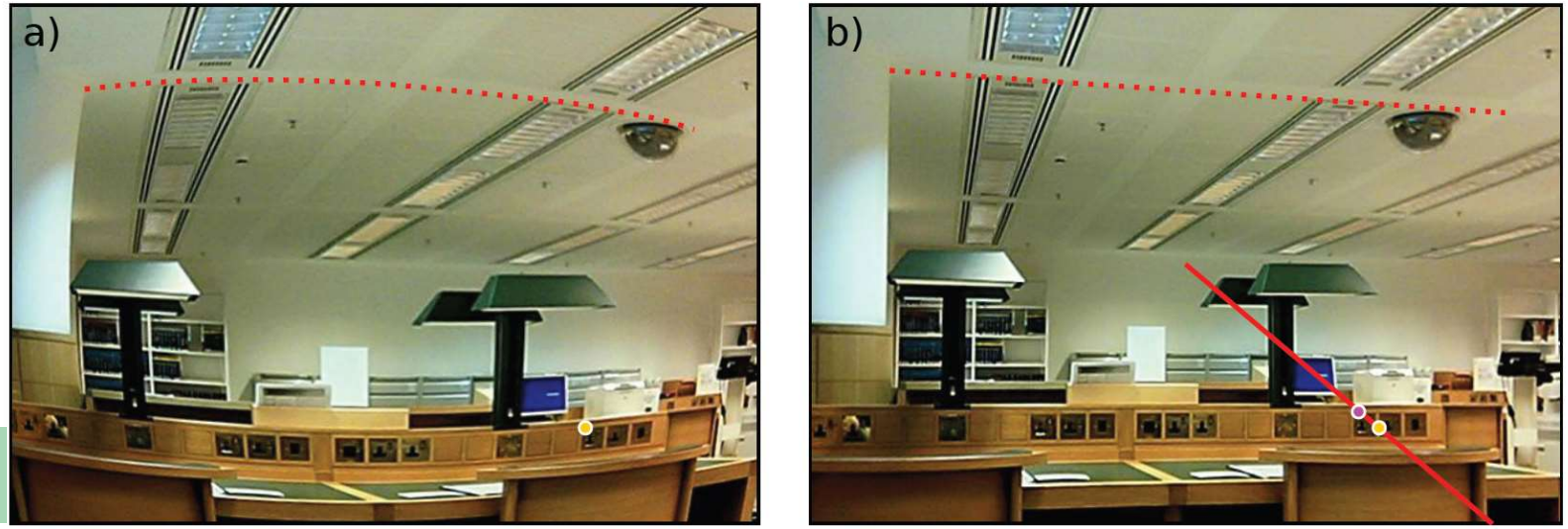
- in short: $\boldsymbol{w'} = \boldsymbol{\Omega w} + \boldsymbol{\tau}$.

Camera Calibration cont'd

- combining world coordinate system transform $\boldsymbol{w'} = \boldsymbol{\Omega w} + \boldsymbol{\tau}$ and the affine image plane projection $\boldsymbol{x} = \frac{\boldsymbol{\phi_x \cdot u + \gamma \cdot v}}{\boldsymbol{w}} + \boldsymbol{\delta_x}$ and $\boldsymbol{y} = \frac{\boldsymbol{\phi_y \cdot v}}{\boldsymbol{w}} + \boldsymbol{\delta_y}$, one gets
- $x = \frac{\boldsymbol{\phi_x}(w_{11}\boldsymbol{u} + w_{12}v + w_{13}\boldsymbol{w} + \tau_x) + \gamma(w_{21}\boldsymbol{u} + w_{22}v + w_{23}\boldsymbol{w} + \tau_y)}{w_{31}\boldsymbol{u} + w_{32}v + w_{33}\boldsymbol{w} + \tau_z} + \boldsymbol{\delta_x}$
- $y = \frac{\boldsymbol{\phi_y}(w_{21}\boldsymbol{u} + w_{22}v + w_{23}\boldsymbol{w} + \tau_y)}{w_{31}\boldsymbol{u} + w_{32}v + w_{33}\boldsymbol{w} + \tau_z} + \boldsymbol{\delta_y}$
  - consequently, **5** scalar intrinsic parameters are relevant, namely $\{\boldsymbol{\phi_x}, \boldsymbol{\phi_y}, \gamma, \boldsymbol{\delta_x}, \boldsymbol{\delta_y}\}$ to be represented in a intrinsic transformation matrix lambda $\boldsymbol{\Lambda} = \begin{bmatrix} \boldsymbol{\phi_x} & \gamma & \boldsymbol{\delta_x} \\ 0 & \boldsymbol{\phi_y} & \boldsymbol{\delta_y} \\ 0 & 0 & 1 \end{bmatrix}$.
  - furthermore, there are extrinsic parameter tensors $\{\boldsymbol{\Omega}, \boldsymbol{\tau}\}$ with **6** parameters for pose.
- overall in short: $\mathbf{w'} = \mathbf{pinhole}[\mathbf{w}, \boldsymbol{\Lambda}, \boldsymbol{\Omega}, \boldsymbol{\tau}]$.

Camera Calibration cont'd

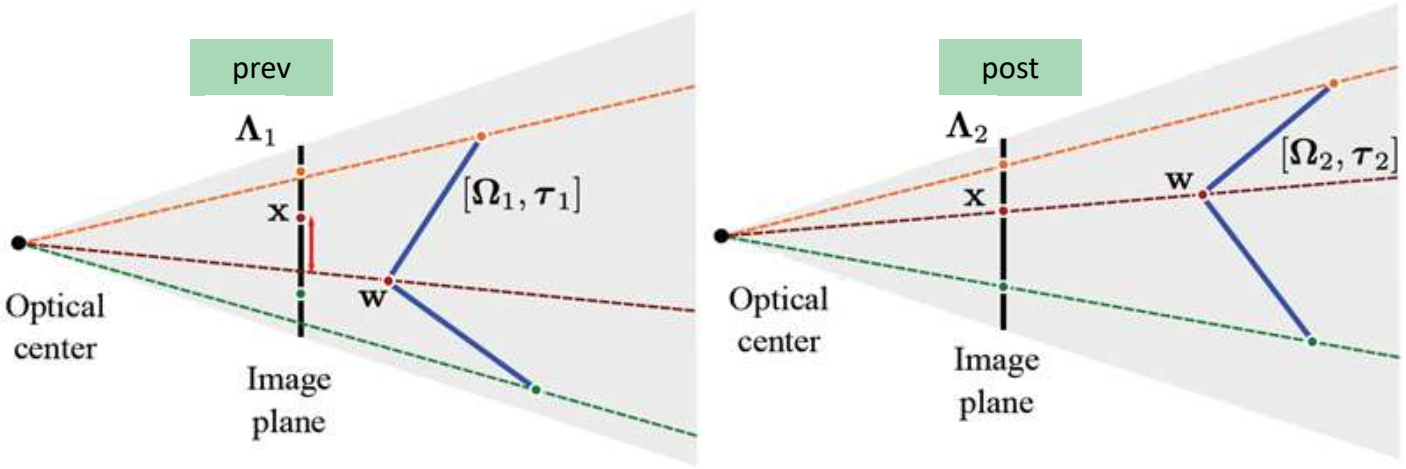
- the camera model developed so far will lead to discrepancies between observed positions and true world geometry due to inaccuracies of the lens and other deficiencies of the model and the optics domain.
  - thus, to address this deviation from the expected results, a noise term is added to the pinhole camera model with $P(\mathbf{w}'|\mathbf{w}, \mathbf{\Lambda}, \mathbf{\Omega}, \boldsymbol{\tau}) = Norm_{\mathbf{w}'}[\mathbf{pinhole}[\mathbf{w}, \mathbf{\Lambda}, \mathbf{\Omega}, \boldsymbol{\tau}], \sigma^2]$.
- radial distortion not considered so far, introducing distortion coefficients β_1 and β_2 for modelling increasing distortion when distance r from center increases too.
  - $x' = x \cdot (1 + \beta_1 r^2 + \beta_2 r^4)$
  - $y' = y \cdot (1 + \beta_1 r^2 + \beta_2 r^4)$



2D image before and after distortion correction

Camera Calibration cont'd

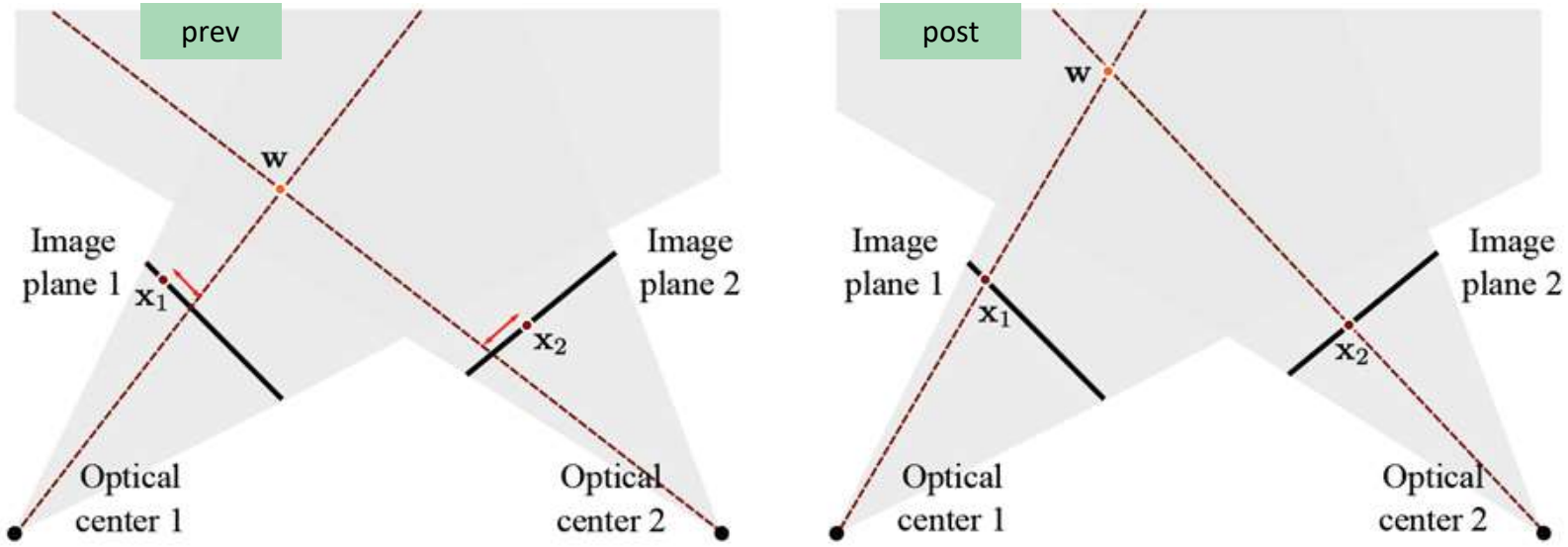
- **Problem Statement**: *what to calibrate?* Generally three different geometric problems can/need to be solved based on the introduced model for the pinhole camera, namely:



- (a) approximate/learn the extrinsic parameters, i.e. the exterior orientation of the scene relative to the camera – to be calculated with maximum likelihood: $\widehat{\boldsymbol{\Omega}}, \hat{\boldsymbol{\tau}} = \underset{\boldsymbol{\Omega},\boldsymbol{\tau}}{\operatorname{argmax}} \left[\sum_{i=1}^{I} log\left(P(\boldsymbol{x_i}|\boldsymbol{w_i}, \boldsymbol{\Lambda}, \boldsymbol{\Omega}, \boldsymbol{\tau})\right)\right]$ for I reference points
- (b) approximate/learn the intrinsic parameters, i.e. distortion, focal length camera,... – to be calculated with maximum likelihood: $\widehat{\boldsymbol{\Lambda}} = \underset{\boldsymbol{\Lambda}}{\operatorname{argmax}} \left[\max_{\boldsymbol{\Omega},\boldsymbol{\tau}} \sum_{i=1}^{I} log\left(P(\boldsymbol{x_i}|\boldsymbol{w_i}, \boldsymbol{\Lambda}, \boldsymbol{\Omega}, \boldsymbol{\tau})\right)\right]$ for I reference points.

Camera Calibration cont'd

- (c) 3D reconstruction from correlating pairs of points from multiple views, thus an unknown point w in the 3D scene is altered to minimize the discrepancy between measured (affine 2D image) and theoretic image coordinates: $\widehat{\boldsymbol{w}} = \underset{\boldsymbol{w}}{\operatorname{argmax}} \left[\sum_{j=1}^{J} log\left(P\left(\boldsymbol{x_j}|\boldsymbol{w}, \boldsymbol{\Lambda_j}, \boldsymbol{\Omega_j}, \boldsymbol{\tau_j}\right)\right)\right]$ utilizing j camera systems.

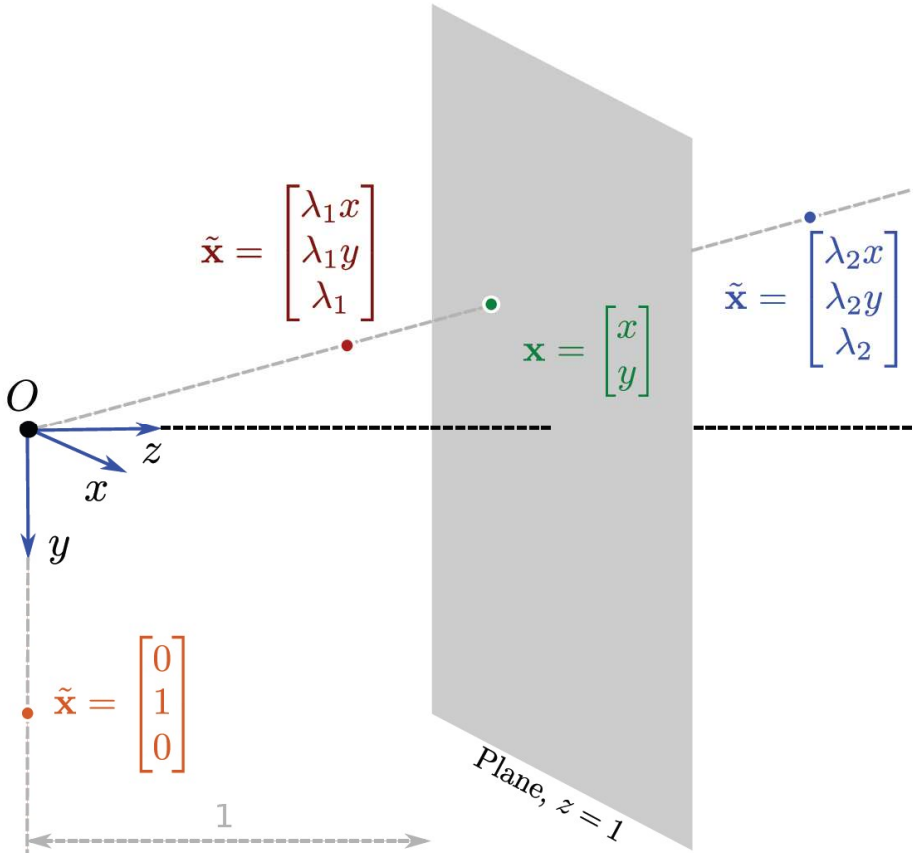


Camera Calibration cont'd

How to solve approximation of extrinsic/intrinsic coefficients or 3D reconstruction?

- problems cannot be solved in closed form, i.e. being solved in a finite number of operations
- no global best solution
- camera parameters all continuous
- heuristic search as non-linear optimization might get stuck in local minima
- thus: first the algebraic error is minimized, then non-linear optimization is applicable to further improve the results.

- anyway, for calculation mismatch between 2D and 3D coordinates is a problem, thus introduction of ***homogeneous*** coordinates with
  - $\tilde{\mathbf{x}} = \begin{bmatrix} \lambda x \\ \lambda y \\ \lambda \end{bmatrix}$ to convert from 2D ➔ 3D and $x = \frac{\tilde{x}}{\tilde{z}}$ and $y = \frac{\tilde{y}}{\tilde{z}}$
  - with $z = 1$, the image plane is virtually "1 unit" away from camera coordinate system origin



Camera Calibration cont'd

Retrospective of pinhole camera model in the light of homogeneous coordinates

- we get $\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\phi_x} & \gamma & \boldsymbol{\delta_x} & 0 \\ 0 & \boldsymbol{\phi_y} & \boldsymbol{\delta_y} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix}$
- incorporating the extrinsic parameters we get

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\phi_x} & \gamma & \boldsymbol{\delta_x} & 0 \\ 0 & \boldsymbol{\phi_y} & \boldsymbol{\delta_y} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} w_{11} & w_{12} & w_{13} & \tau_x \\ w_{21} & w_{22} & w_{23} & \tau_y \\ w_{31} & w_{32} & w_{33} & \tau_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix}$$

- or shorter $\lambda \tilde{\mathbf{x}} = [\Lambda \quad 0] \begin{bmatrix} \Omega & \tau \\ 0^T & 1 \end{bmatrix} \widetilde{w}$
- or even shorter $\lambda \tilde{\mathbf{x}} = \Lambda [\Omega \quad \tau] \widetilde{w}$
- to calculate **exterior orientation**, first both sides are multiplied by the *inverse* of the camera matrix

for each point $\mathbf{x_i}$: $\lambda_i \begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & w_{13} & \tau_x \\ w_{21} & w_{22} & w_{23} & \tau_y \\ w_{31} & w_{32} & w_{33} & \tau_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ w_i \\ 1 \end{bmatrix}$

Camera Calibration cont'd

Exterior Orientation cont'd

- based on $\lambda_i \begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & w_{13} & \tau_x \\ w_{21} & w_{22} & w_{23} & \tau_y \\ w_{31} & w_{32} & w_{33} & \tau_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ w_i \\ 1 \end{bmatrix}$ the third equation contains no coordinates of the image plane and thus gives a numeric expression for λ_i with $\lambda_i = w_{31}u_i + w_{32}v_i + w_{33}w_i + \tau_z$
- now this expression for λ_i can be substituted for the remaining two equations giving:
  - $\begin{bmatrix} (w_{31}u_i + w_{32}v_i + w_{33}w_i + \tau_z)x_i' \\ (w_{31}u_i + w_{32}v_i + w_{33}w_i + \tau_z)y_i' \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & w_{13} & \tau_x \\ w_{21} & w_{22} & w_{23} & \tau_y \end{bmatrix} \cdot \begin{bmatrix} u_i \\ v_i \\ w_i \\ 1 \end{bmatrix}$
  - multiplying we get two equations, namely

$$(w_{31}u_i + w_{32}v_i + w_{33}w_i + \tau_z)x_i' = w_{11}u_i + w_{12}v_i + w_{13}w_i + \tau_x$$
$$(w_{31}u_i + w_{32}v_i + w_{33}w_i + \tau_z)y_i' = w_{21}u_i + w_{22}v_i + w_{23}w_i + \tau_y$$

  - with 12 unknown variables $w_{11}, w_{12}, w_{13}, w_{21}, w_{22}, w_{23}, w_{31}, w_{32}, w_{33}, \tau_x, \tau_y, \tau_z$ one can write

$$\boldsymbol{w_{11}}u_i + \boldsymbol{w_{12}}v_i + \boldsymbol{w_{13}}w_i + \boldsymbol{w_{21}} \cdot 0 + \boldsymbol{w_{22}} \cdot 0 + \boldsymbol{w_{23}} \cdot 0 - \boldsymbol{w_{31}}u_i x_i' - \boldsymbol{w_{32}}v_i x_i' - \boldsymbol{w_{33}}w_i x_i' + \boldsymbol{\tau_x} + \boldsymbol{\tau_y} \cdot 0 - \boldsymbol{\tau_z}x_i' = 0$$
$$\boldsymbol{w_{11}} \cdot 0 + \boldsymbol{w_{12}} \cdot 0 + \boldsymbol{w_{13}} \cdot 0 + \boldsymbol{w_{21}}u_i + \boldsymbol{w_{22}}v_i + \boldsymbol{w_{23}}w_i - \boldsymbol{w_{31}}u_i y_i' - \boldsymbol{w_{32}}v_i y_i' - \boldsymbol{w_{33}}w_i y_i' + \boldsymbol{\tau_x} \cdot 0 + \boldsymbol{\tau_y} - \boldsymbol{\tau_z}y_i' = 0$$

Camera Calibration cont'd

Exterior Orientation cont'd

- now as for each point $\mathbf{x_i}$ two equations arise, comprising the unknown 12 variables w_{11}, w_{12}, w_{13}, w_{21}, w_{22}, w_{23}, w_{31}, w_{32}, w_{33}, τ_x, τ_y, τ_z one can write

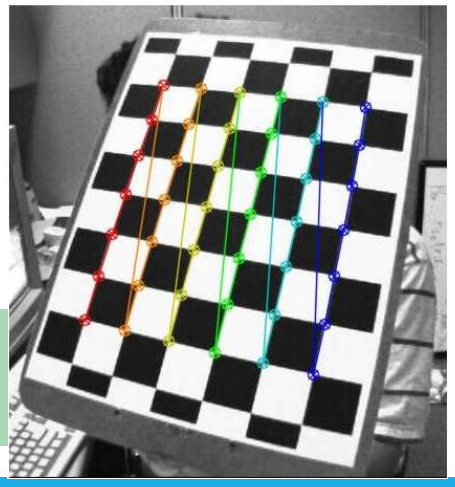
$$\begin{bmatrix} u_1 & v_1 & w_1 & 0 & 0 & 0 & -u_1x_1' & -v_1x_1' & -w_1x_1' & 1 & 0 & -x_1' \\ 0 & 0 & 0 & u_1 & v_1 & w_1 & -u_1y_1' & -v_1y_1' & -w_1y_1' & 0 & 1 & -y_1' \\ u_2 & v_2 & w_2 & 0 & 0 & 0 & -u_2x_2' & -v_2x_2' & -w_2x_2' & 1 & 0 & -x_2' \\ 0 & 0 & 0 & u_2 & v_2 & w_2 & -u_2y_2' & -v_2y_2' & -w_2y_2' & 0 & 1 & -y_2' \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_n & v_n & w_n & 0 & 0 & 0 & -u_nx_n' & -v_nx_n' & -w_nx_n' & 1 & 0 & -x_n' \\ 0 & 0 & 0 & u_n & v_n & w_n & -u_ny_n' & -v_ny_n' & -w_ny_n' & 0 & 1 & -y_n' \end{bmatrix} \cdot \begin{bmatrix} w_{11} \\ w_{12} \\ w_{13} \\ w_{21} \\ w_{22} \\ w_{23} \\ w_{31} \\ w_{32} \\ w_{33} \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = 0$$

- to be solved utilizing SVD (singular value decomposition)
- *problems*: scale is arbitrary and rows/columns of rotation matrix not necessarily orthogonal ➔ only first rough estimate for non-linear optimization to follow

Camera Calibration cont'd

Technical setup

- regular 2D or 3D test patterns required, generally checkerboard.
  - If the pattern is regular, only dimensionality, e.g. 7x6 reference points, is required
  - If the test pattern is 2D, z is set to zero.
  - The reference grid is thus specified with (0,0), (1,0),... or true size, e.g. in mm with (0,50), (0,100),... making no difference w.r.t. calculating the intrinsic parameters.
  - as shown in the following slides, each pair of points $(\tilde{x}_i, \tilde{w}_i)$ contributes to overall equation system with 2 equations. As 12 variables need to be solved, at least **6** pairs of points are required. In practice, a significantly larger amount of points is utilized, repeating the calibration from different points of view for optimization.



calibration image with detected 7 × 6 test pattern

undistorted test image

References

- [Heikkila and Silven 1997] Heikila, J., and Silven, O., 1997. *A Four-step Camera Calibration Procedure with Implicit Image Correction*. In: IEEE Int. Conf. on Computer Vision and Pattern Recognition.
- [Jackson et al. 2017] Jackson, A.S., Bulat, A., Argyriou, V., and Tzimiropoulos, G., 2017. *Large Pose 3D Face Reconstruction from a Single Image via Direct Volumetric CNN Regression*. In: Proc. of Int. Conf. on Computer Vision.
- [Tsai 1986] Tsai, R.Y., 1986. An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, Florida, USA, pp.364-274.
- [Zhang 2000] Zhang, Z., 2000. “A Flexible New Technique for Camera Calibration.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22(11), pp. 1330–1334.