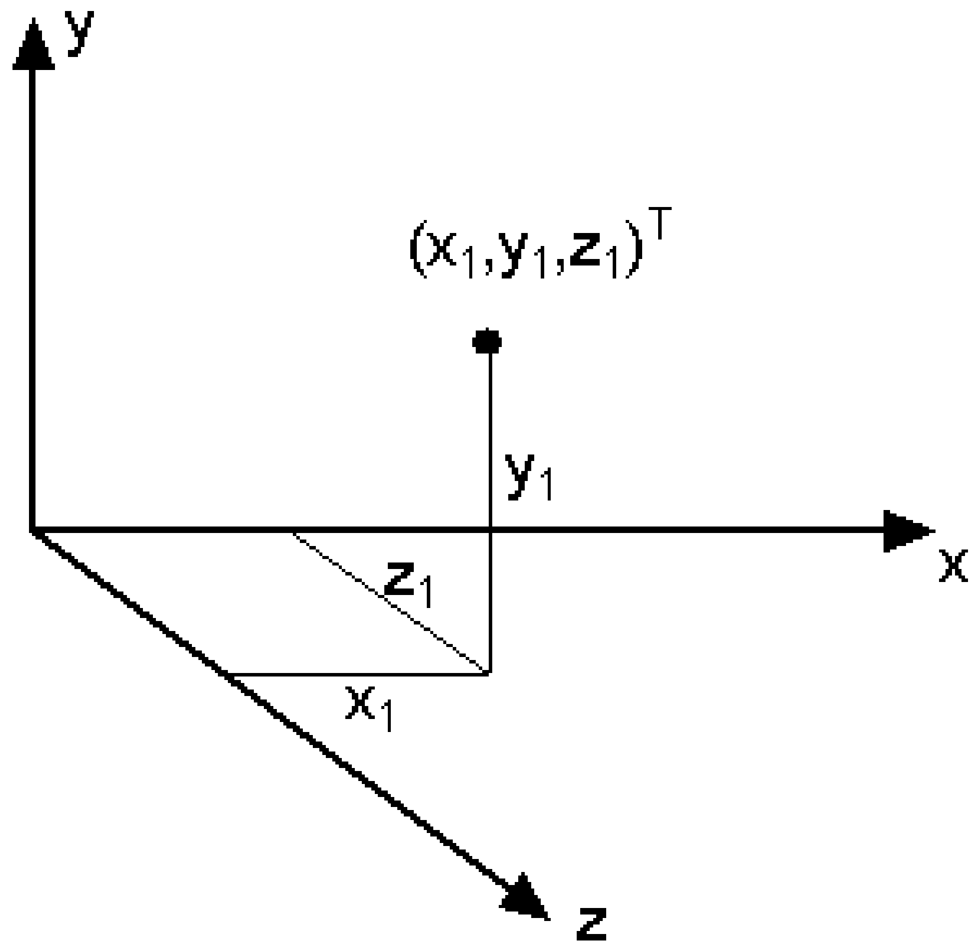
Spatial Representations & Transforms

Coordinate Systems

Cartesian Coordinates

- 3 unit vectors
  - x-axis: $(1,0,0)^T$
  - y-axis: $(0,1,0)^T$
  - z-axis: $(0,0,1)^T$
  - form basis
- origin: $\boldsymbol{o} = (0,0,0)^T$
- points: column vector $\boldsymbol{p}$
  - *x, y, z* coordinates

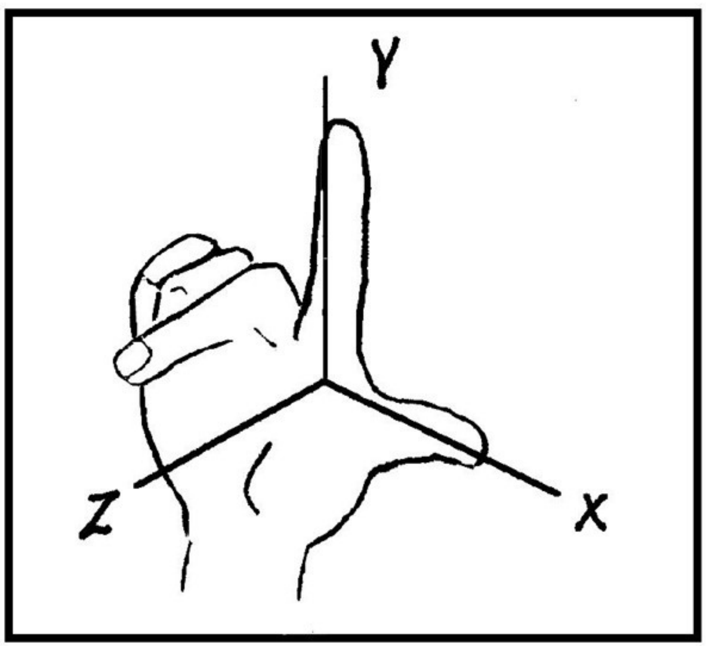


Conventions: Notations

- scalar: small letter, normal – e.g., x
- vector: small letter, bold – e.g., $\boldsymbol{x}$
- matrix: capital letters, bold – e.g., $\boldsymbol{A}$
- transpose matrix/vector: superscript T– e.g., $\boldsymbol{A}^{T}$

Conventions: Right Hand Rule

- Cartesian Coordinate System should ***always*** follow the right hand rule!!!
- Attention!!! There are quite some exceptions in hard- and software in robotics that do not

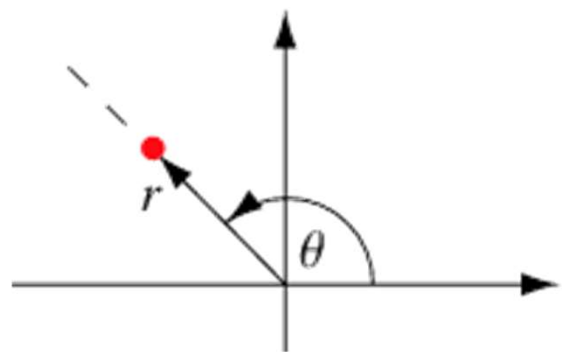


fingers of the ***right*** hand

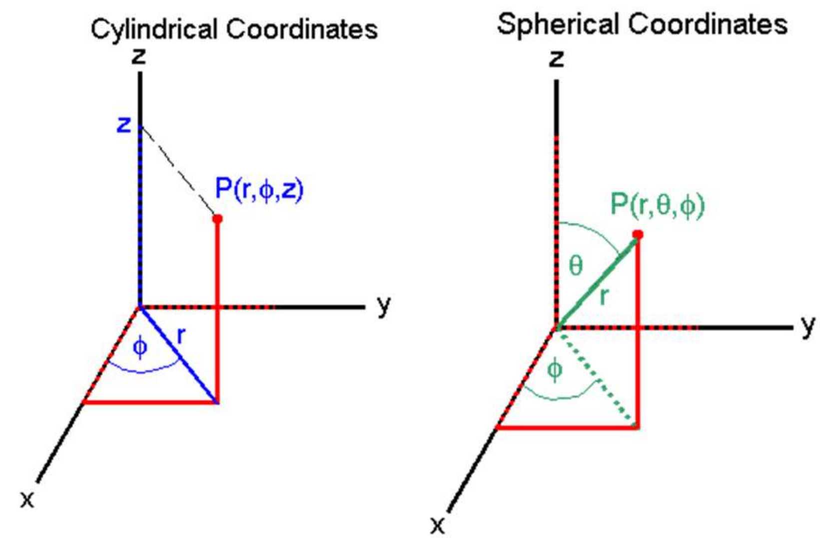
- x ~ thumb
- y ~ index finger
- z ~ middle finger

Angular Coordinates

- 2D polar coordinates
  - radial distance from origin
    - here: r
  - counterclockwise angle from x-axis
    - aka polar angle or azimuth
    - here: θ



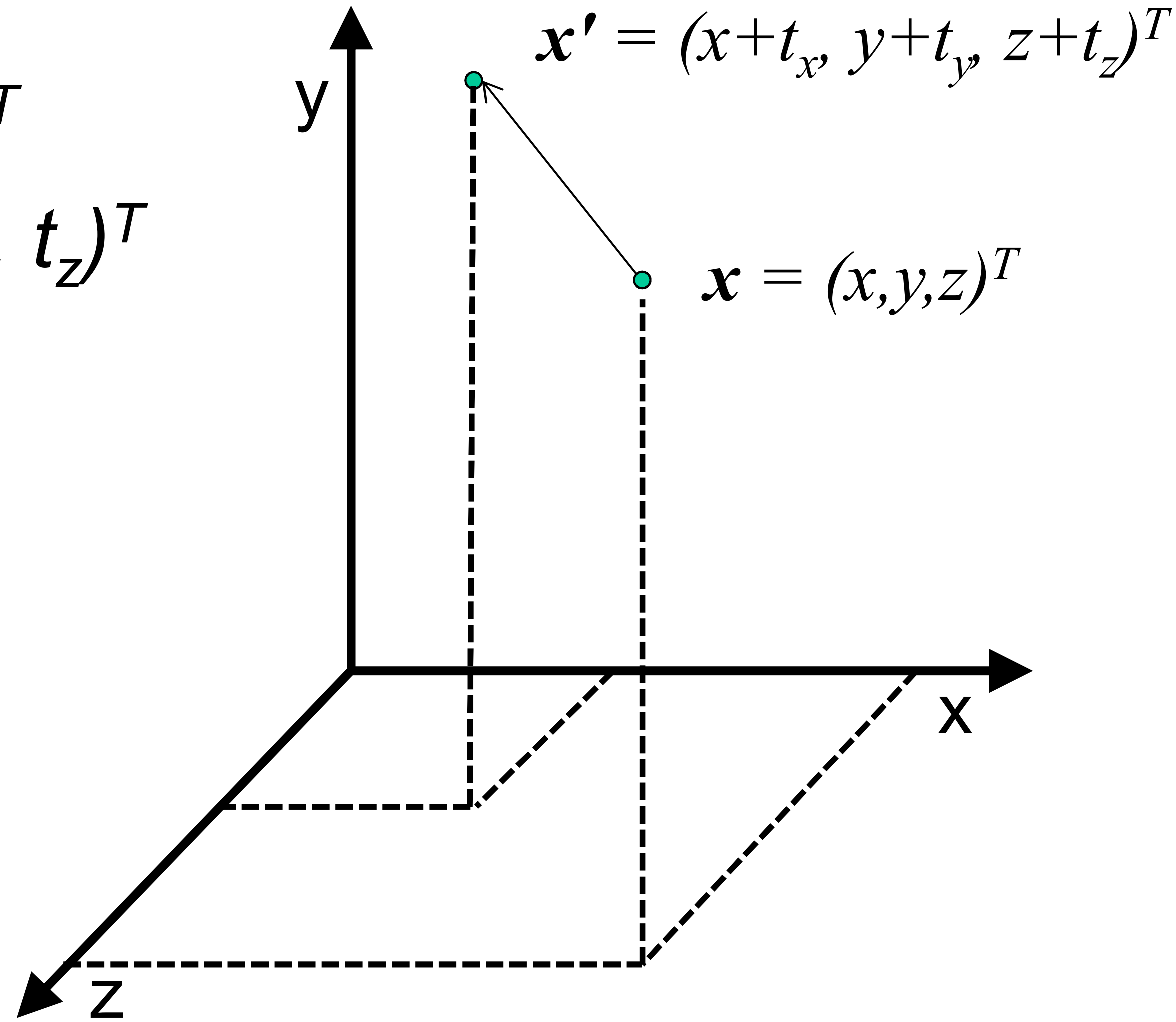
- 3D extension
  - cylindrical: polar + Cartesian z
  - spherical: polar + angle y-axis



Geometric Transformation: Translation

- **Translation**
- of point $\boldsymbol{x} = (x,y,z)^T$
- by vector $\boldsymbol{t} = (t_x, t_y, t_z)^T$
- through **addition**

$$\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{t}$$



Geometric Transformation: Rotation

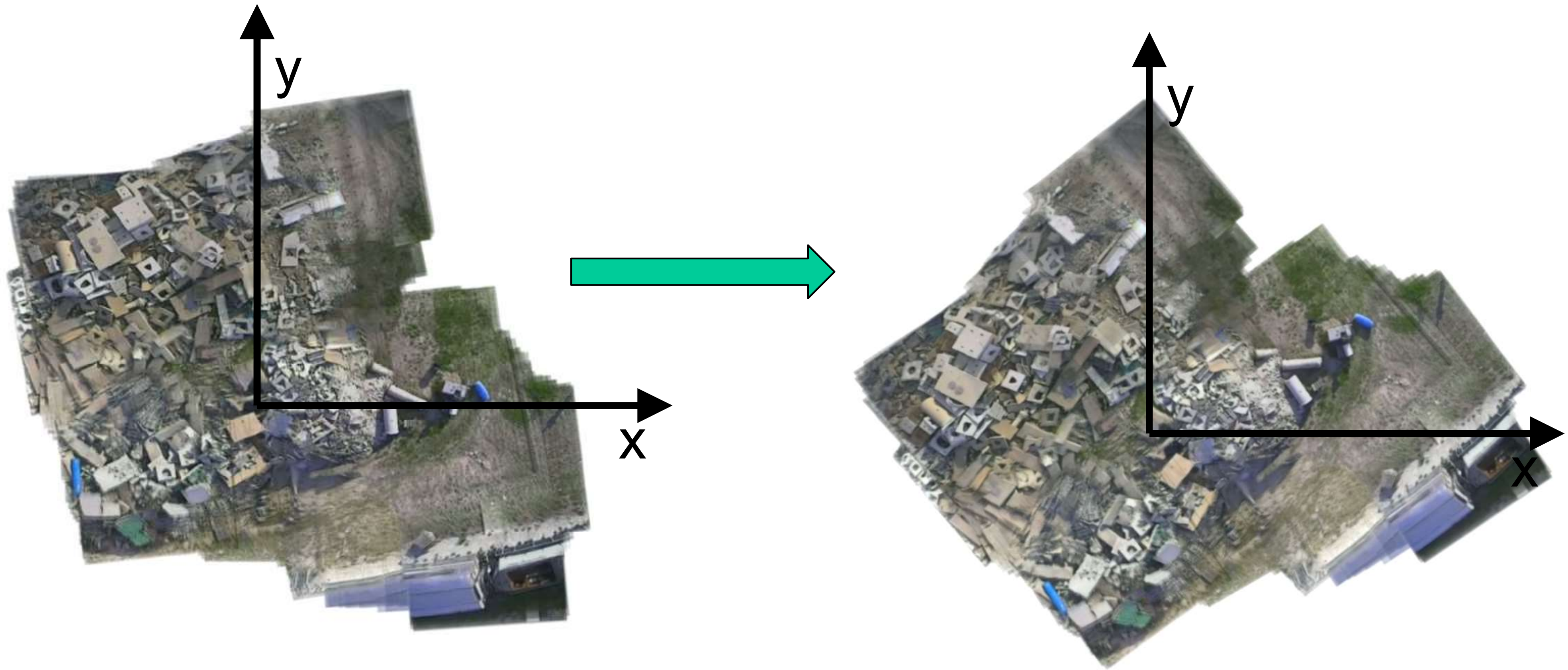
- **rotation**
  - in the plane (2D) around origine ***o***
  - by angle α according to right-hand-rule
- through **left-multiplication** with Rotationsmatrix $\mathbf{R_{2D}}$

$$\mathbf{R_{2D}} = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix}$$

$$\mathbf{x}' = \mathbf{R_{2D}}\ \mathbf{x}$$

Rotation

example: rotation by 30^o



$$\mathbf{R} = \begin{pmatrix} \cos(30^o) & -\sin(30^o) \\ \sin(30^o) & \cos(30^o) \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\sqrt{3} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2}\sqrt{3} \end{pmatrix}$$

Conventions: Rotations

- rotations ***should*** also ***always*** follow the **right-hand-rule**
- Attention!!! For this aspect, even more ***exceptions*** in soft- and hardware

fingers of the ***right*** hand

- rotation axis ~ thumb
- sense of rotation ~ bended fingers

(hence aka **cork-screw** rule)

Conventions: Rotations

positive angle ~
counter clockwise rotation
(CCR)

negative angle ~
clockwise rotation
(CWR)

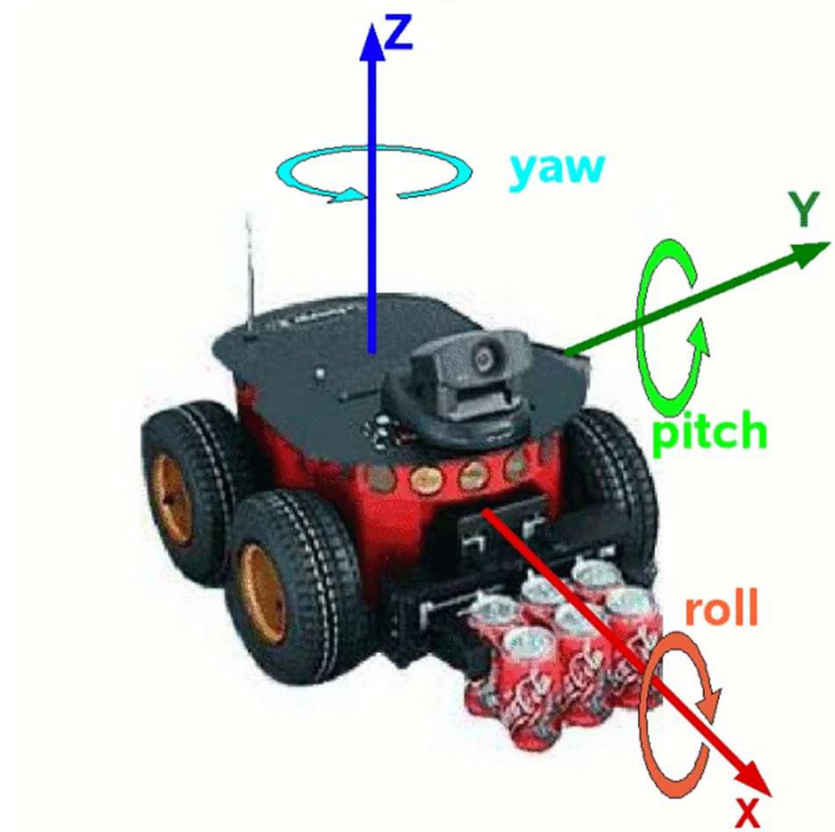
Rotations in 3D Space

- in a **Rotation Plane** (like in 2D)
- around perpendicular **axis of rotation** (defines the point of rotation)
- e.g., around x-, y-, z-axis

Roll

Pitch

Yaw



Rotations in 3D Space

- in a **Rotation Plane** (like in 2D)
- around perpendicular **axis of rotation** (defines the point of rotation)
- e.g., around x-, y-, z-axis

$$\mathbf{R_x} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{pmatrix}$$

$$\mathbf{R_y} = \begin{pmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{pmatrix} \quad \mathbf{R_z} = \begin{pmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Rotations in 3D Space

- arbitrary rotation:
- e.g., composition of rotations around x,y,z-axis

$$\mathbf{x}' = \mathbf{R_z R_y R_x \; x}$$

ATTENTION: matrix-multiplication is ***not commutativ***

hence: order of rotations is important!!!

(more about this later)

Linear Transformations

$$\mathbf{x} \in \mathbb{R}^n$$

$$\mathbf{A} \in \mathbb{R}^{n \times n}$$

$$f : \mathbb{R}^n \to \mathbb{R}^n$$

$$\mathbf{x} \mapsto \mathbf{A}\mathbf{x}$$

e.g.:

- **Rotation**
- Reflection
- Shear
- Scale

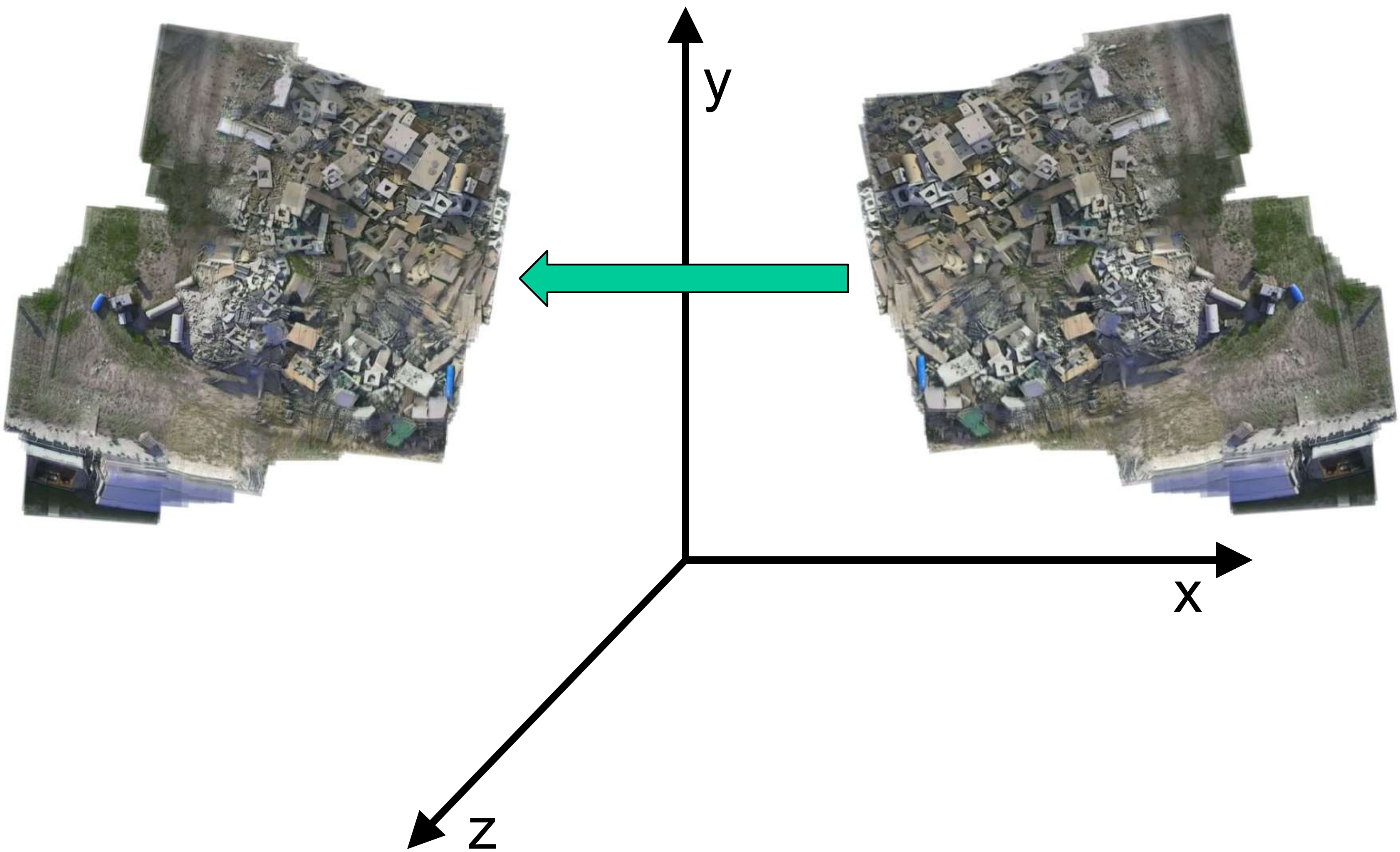
Reflection

along x-, y-, z-Achse

$$\mathbf{A}_{\mathbf{sp-x}} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{A}_{\mathbf{sp-y}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \mathbf{A}_{\mathbf{sp-z}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

Reflection



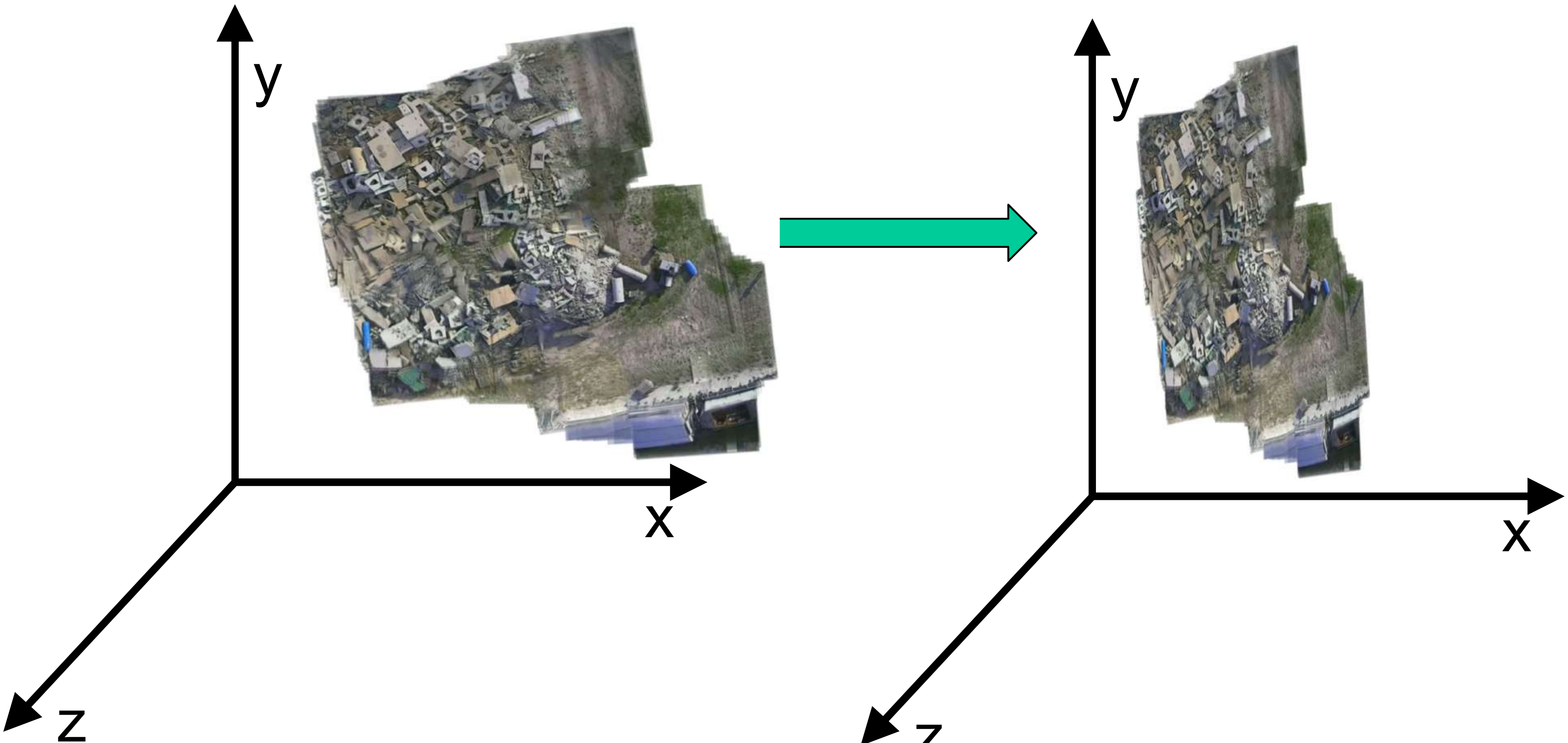
e.g.: along y-axis

Scale

with scaling factors s_i along x-, y-, z-axis

$$\mathbf{A}_{\mathbf{streck}} = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{pmatrix}$$

Scale



e.g.: scaling by ½ along x-axis

Shear

2-D: shear along x-, y-axis

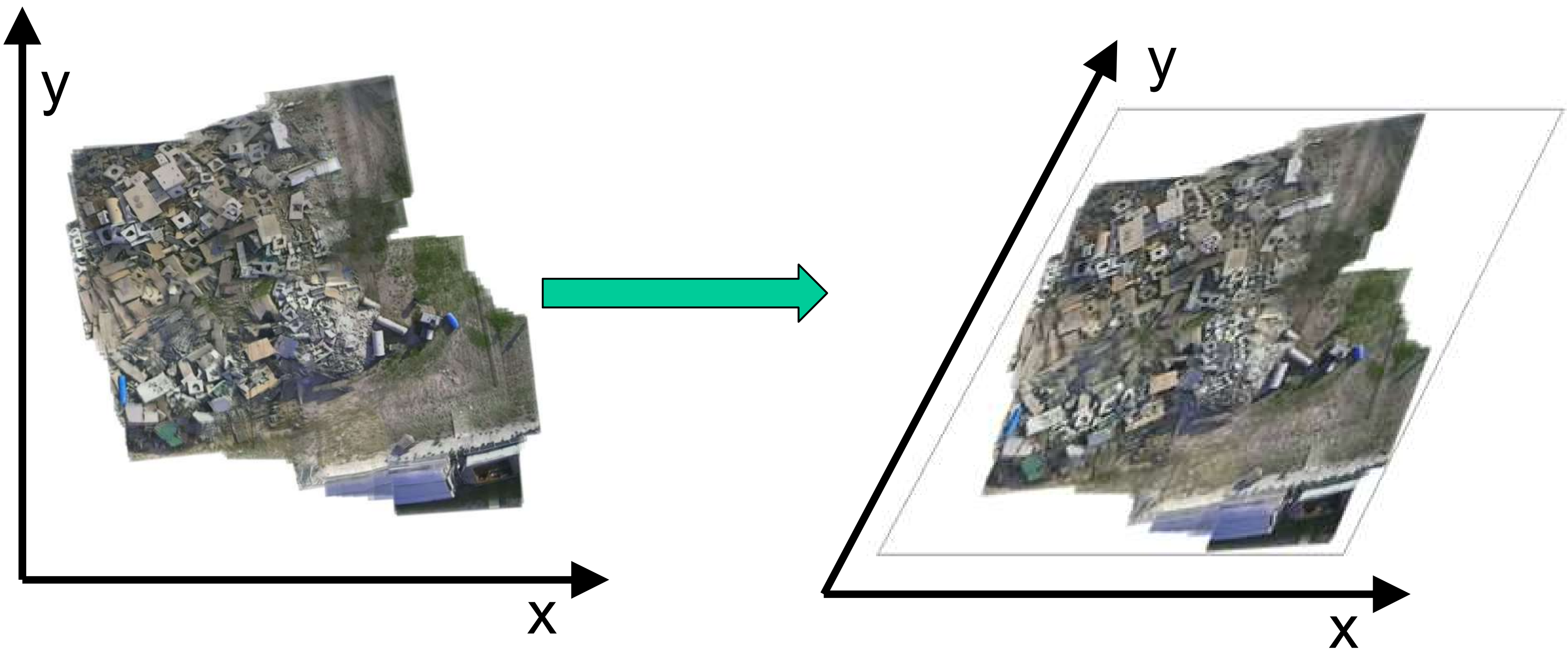
$$\mathbf{A}_{\mathbf{sch-x}} = \begin{pmatrix} 1 & e \\ 0 & 1 \end{pmatrix} \qquad \mathbf{A}_{\mathbf{sch-y}} = \begin{pmatrix} 1 & 0 \\ e & 1 \end{pmatrix}$$

3D: shear in xy-plane (xz, yz accordingly)

$$\mathbf{A}_{\mathbf{sch-xy}} = \begin{pmatrix} 1 & 0 & e_x \\ 0 & 1 & e_y \\ 0 & 0 & 1 \end{pmatrix}$$

Shear

e.g.: shear along x-axis



Affine Transformations

- linear transformation: matrix **A**
- plus translation: vector **t**

$$f : \mathbb{R}^n \rightarrow \mathbb{R}^n$$

$$\mathbf{x} \mapsto \mathbf{A}\mathbf{x} + \mathbf{t}$$

(from now on 3D, i.e., *n* = *3*)

Translation by Multiplication

- translations are non-linear
- but matrix-multiplication is very useful tool

trick: ***homogeous coordinates***

- additional dimension: $\mathbb{R}^3 \to \mathbb{R}^4$

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad \mapsto \quad \mathbf{x} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Notation

- it is often assumed that the use of 3D coordinates or of homogeneous coordinates is clear from the context
- hence, symbol ***x*** can refer to a point as a **3- or 4-dimensional** vector (depending on mathematical context)

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad \mapsto \quad \mathbf{x} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Translation with Homogeneous Coordinates

- translation by $\mathbf{t} = (t_x, t_z, t_z)^T$
- now via matrix-multiplication

$$\begin{pmatrix} x + t_x \\ y + t_y \\ z + t_z \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Affine Transformation with Homogeneous Coordinates

Affine Transformation $\mathbf{x} \mapsto \mathbf{A}\mathbf{x} + \mathbf{t}$

with related homogenous matrix **H**:

$$\mathbf{H} = \begin{pmatrix} & \mathbf{A} & & \mathbf{t} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$\mathbf{A} \in \mathbb{R}^{3\times 3}$

$\mathbf{t} \in \mathbb{R}^{3}$

$\mathbf{H} \in \mathbb{R}^{4\times 4}$

Affine Transformation with Homogeneous Coordinates

Affine Transformation $\mathbf{x} \mapsto \mathbf{A}\mathbf{x} + \mathbf{t}$

$$\mathbf{x}' = \underbrace{\begin{pmatrix} & \mathbf{A} & & \mathbf{t} \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathbf{H}} \mathbf{x}$$

=> arbitrary affine transformation via matrix multiplication

Affine Transformation with Homogeneous Coordinates

Affine Transformation $\mathbf{x} \mapsto \mathbf{A}\mathbf{x} + \mathbf{t}$

strictly speaking:

$$\begin{pmatrix} \mathbf{x}' \\ 1 \end{pmatrix} = \begin{pmatrix} & \mathbf{A} & & \mathbf{t} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}$$

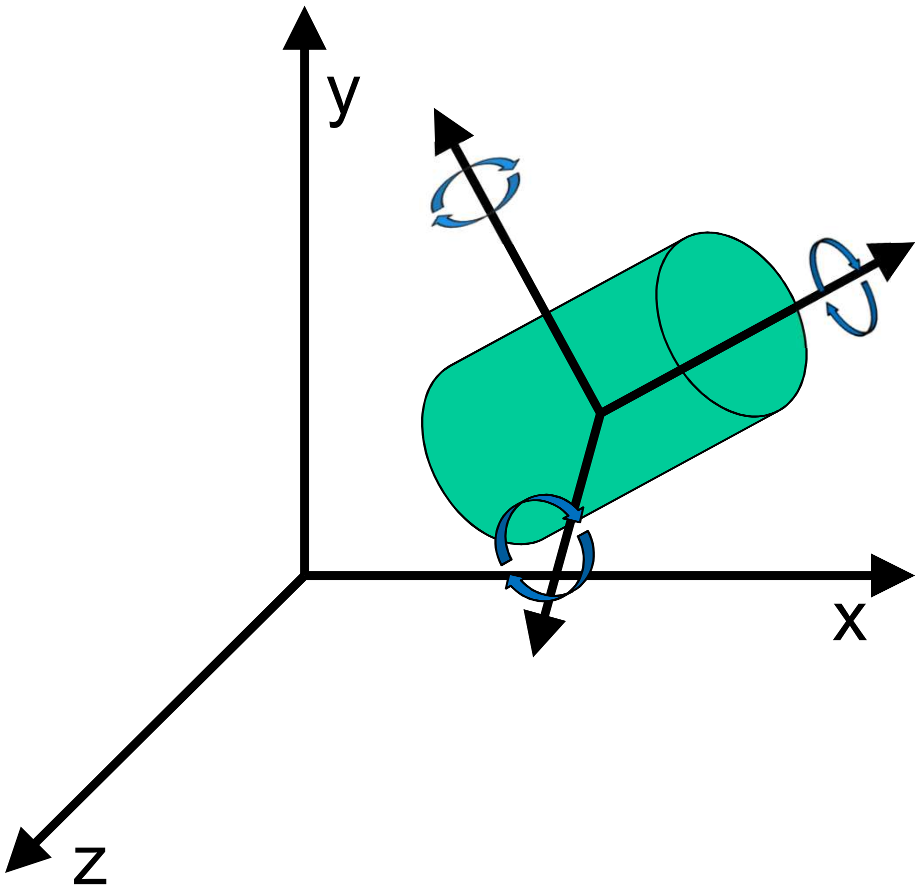
as already mentioned:

- same ***x*** for point in 3D and in homogenous coordinates
- proper semantics implicated by context

Rigid Body in 3D

- not only point ***x*** for position
- but also orientation

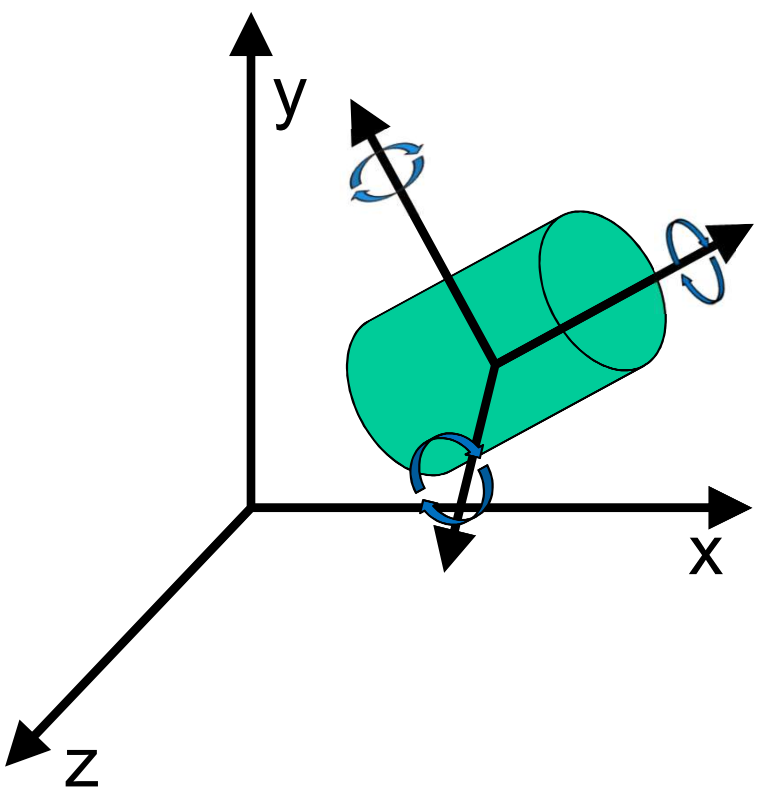
=> ***Pose*** (position & orientation)



e.g.: via Roll, Pitch, Yaw

Rigid Body in 3D

- **Degree of Freedom**, DoF:
- number of independent motion variables
- essential concept in the context of Kinematics



rigid body in 3D: 6 DoF

- 3 DoF translation (position)
- 3 DoF rotation (orientation)

Poses and Homogeneous Matrices

Homogeneous Matrix

- with Rotation **A**
- and Translation **t**

$$\mathbf{H} = \begin{pmatrix} & \mathbf{A} & & \mathbf{t} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

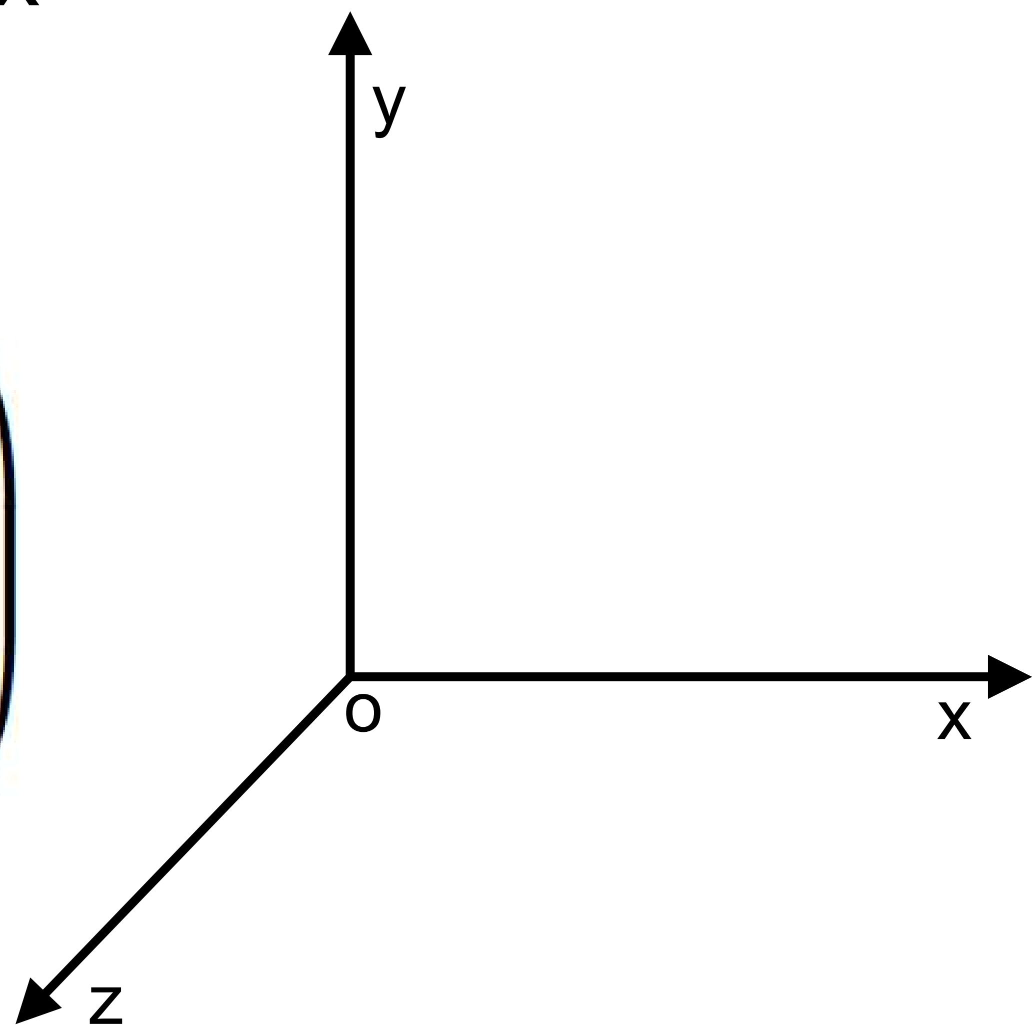
dual interpretation of **H**:

- geometric transformation
- coordinate system (aka ***frame***) with
  - – origin **t**
  - – x,y,z-axis along the columns in **A**

Poses and Homogeneous Matrices

example: identity matrix

$$\mathbf{I} = \begin{array}{c} \begin{matrix} \boldsymbol{x} & \boldsymbol{y} & \boldsymbol{z} & \boldsymbol{o} \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$



World and Local Frames

- World Frame (aka global frame)
  - system of reference to the real world, e.g.,
    - (mobile) robot pose at time *t=0*
    - origin: left lower corner lab room, axes along floor/walls
  - identity matrix $\boldsymbol{I}_{4x4}$ as canonical choice

- Local Frames
  - (arbitrary) reference systems for objects, z.B.
    - mobile robot
    - (part of) a robot arm
    - sensor on a robot
    - gripper on a robot arm
    - object seen by a sensor
    - to be manipulated object
    - etc.

World and Local Frames

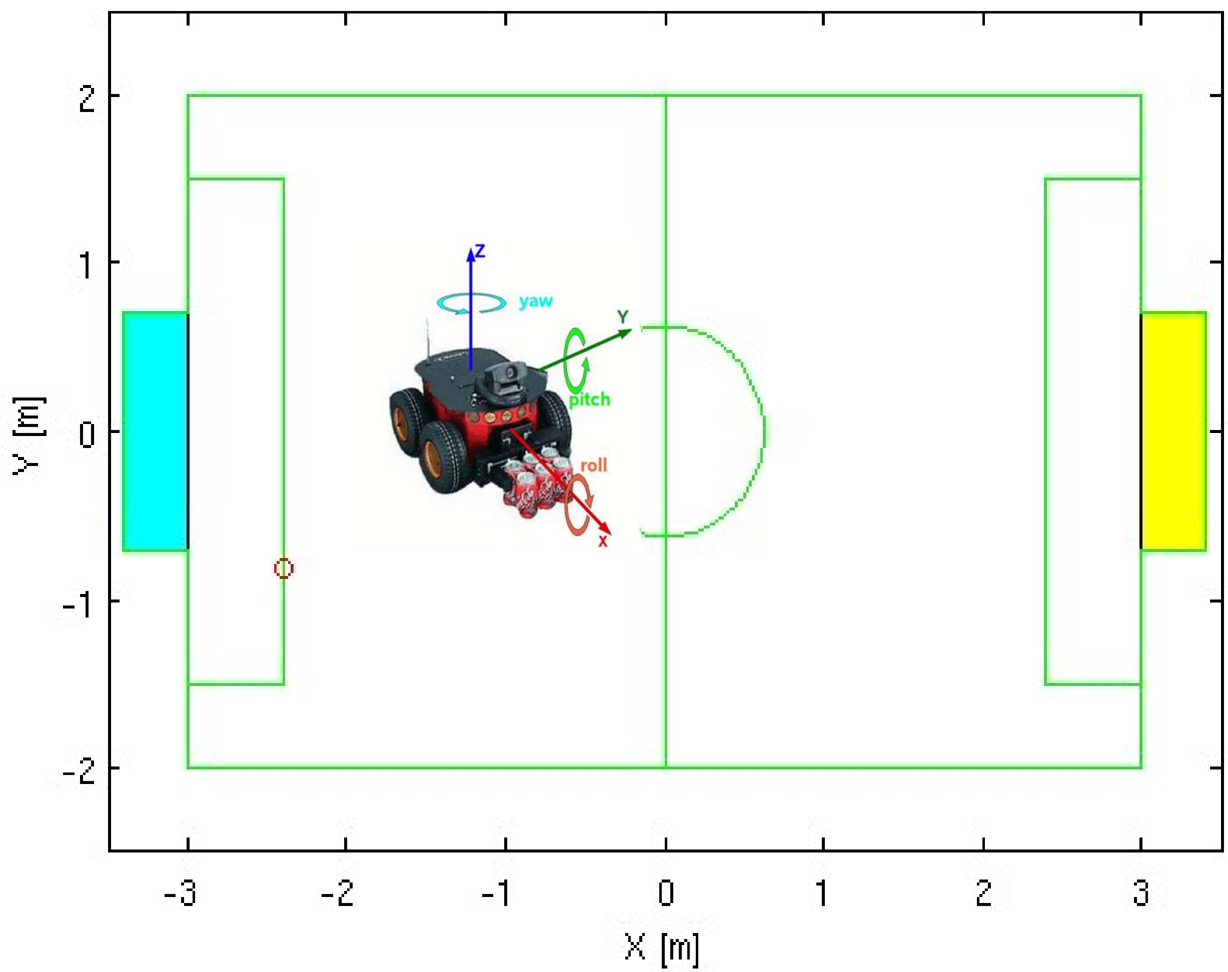
Example.: soccer robot

Frame F_0 (global)

- origin: kickoff
- x: direction opponent
- y: direction left side

Frame F_1 (Local)

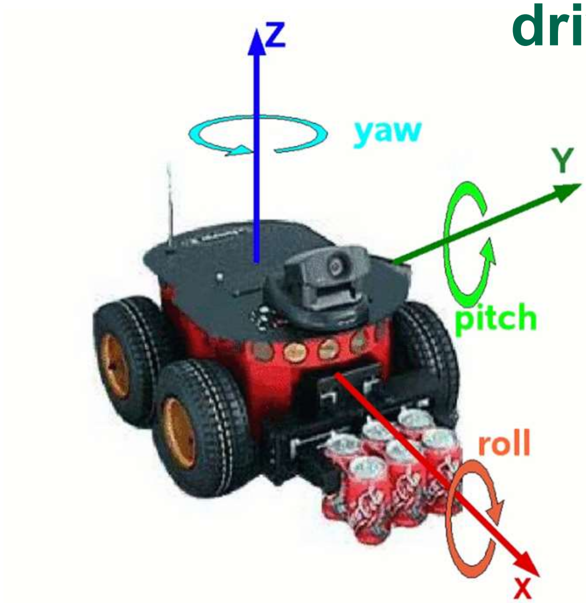
- robocentric frame



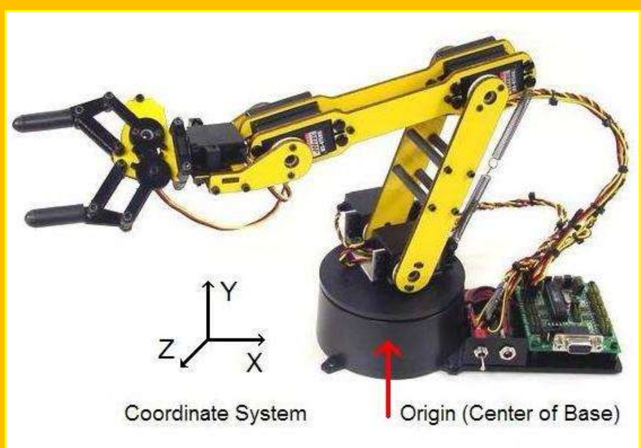
Conventions: Robocentric Frame

- x-axis pointing forward
- z-axis: follows gravitation
  - land-robots / arm: up
  - aerial / space / underwater: down
- and y-axis?

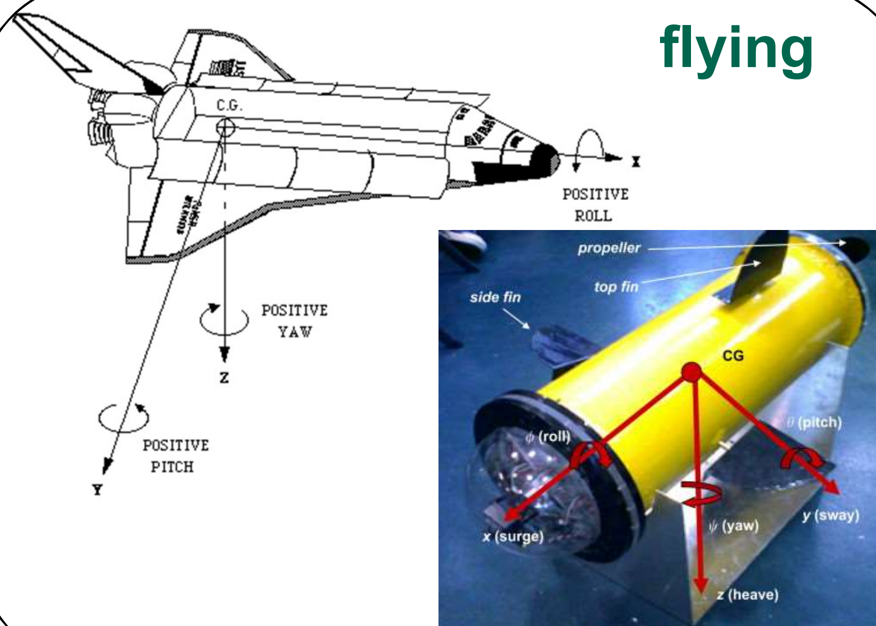
driving



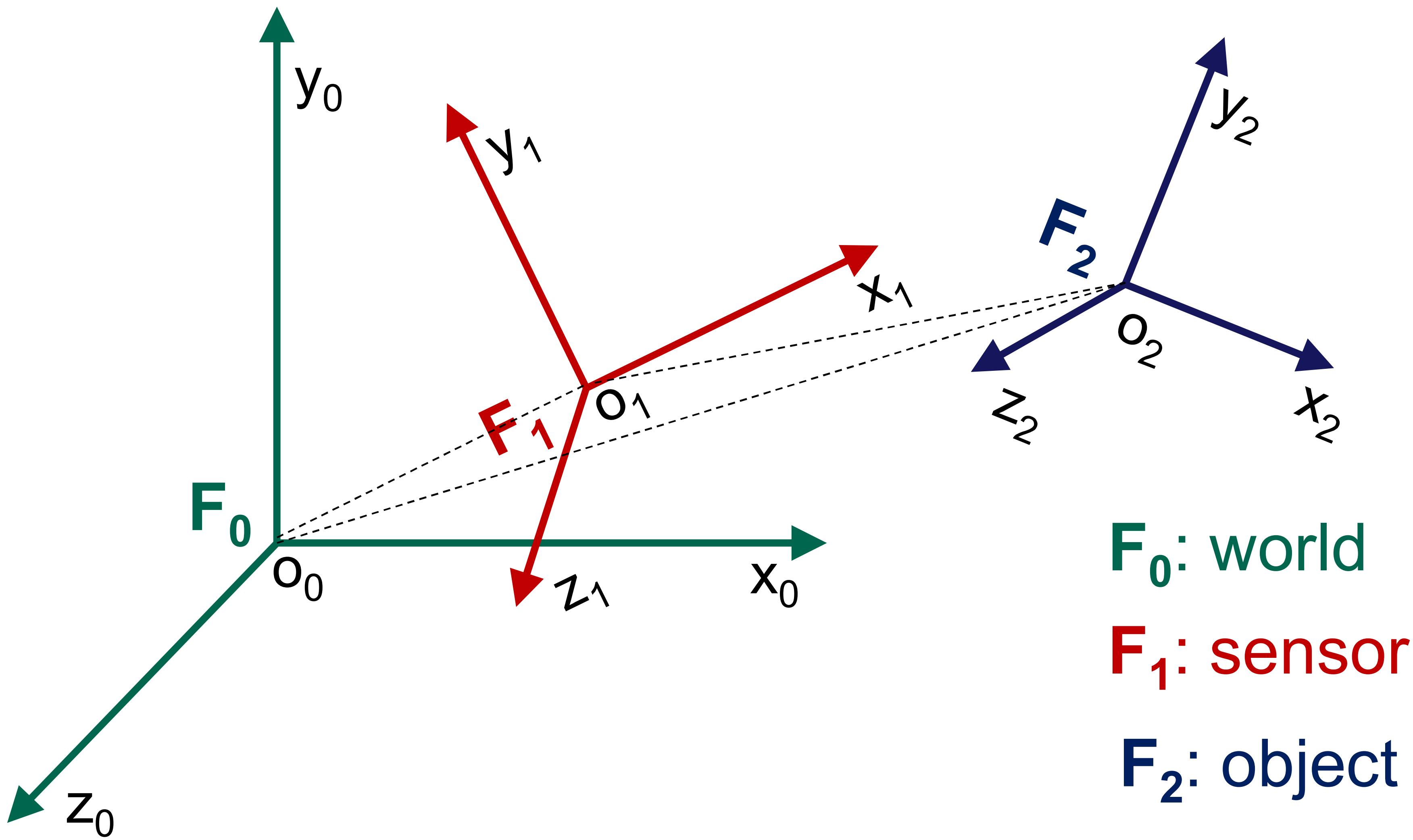
be aware of outliers!!!



flying



Change of References



Representation of F_2 in reference to F_1 versus F_0?

More about rotations

General Rotation Matrix

rotation matrix R

- 3 x 3 matrix
- orthogonal
  - R = (**x**, **y**, **z**), with **x**, **y**, **z** orthonormal
  - i.e., **x**, **y**, **z** are mutually perpendicular
  - $R^T R = R R^T = I$, i.e., $R^{-1} = R^T$
- determinant det(R) = +/-1
  - makes it right-handed (+1)
  - or left-handed (-1)

Recap: 2x2 / 3x3 Determinant

A= $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$, det(A) = |A| = ad – bc

B= $\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$, det(B) = a $\det \overset{\textit{minor}}{\begin{pmatrix} e & f \\ h & i \end{pmatrix}}$ – b $\det \begin{pmatrix} d & f \\ g & i \end{pmatrix}$ + c $\det \begin{pmatrix} d & e \\ g & h \end{pmatrix}$

= a $(ei - fh)$ – b $(di - fg)$+ c $(dh - eg)$

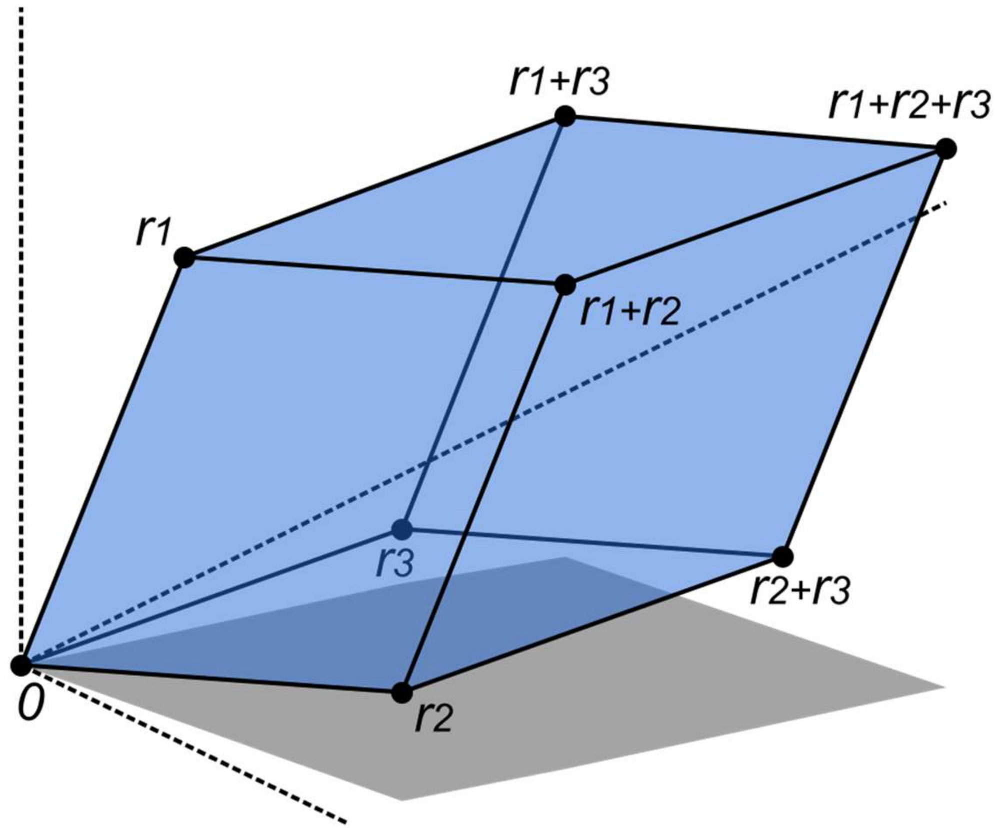
= aei + bfg + cdh − ceg − bdi − afh

Rule of Sarrus

det(A), 3x3

$$\begin{array}{ccc:cc} a_{11} & a_{12} & a_{13} & a_{11} & a_{12} \\ a_{21} & a_{22} & a_{23} & a_{21} & a_{22} \\ a_{31} & a_{32} & a_{33} & a_{31} & a_{32} \end{array}$$

Rotation Matrix & its Determinant



- determinant = volume span by the column vectors
- axis are unit (length 1)
- and orthogonal
  - reflection of 1 axis -> left-handed
  - reflection of 2 axes -> right-handed again
  - and so on...

Inverses of Rotation Matrices

- inverted rotation matrices
- are simply transposed matrices: $\mathbf{R}^{-1} = \mathbf{R}^T$

- hence for composed rotations:

$$(\mathbf{R}_1 \cdot \mathbf{R}_2)^{-1} = (\mathbf{R}_1 \cdot \mathbf{R}_2)^T = {\mathbf{R}_2}^T \cdot {\mathbf{R}_1}^T$$

A Few Fun Facts...

rotation matrix R always invertible?

yes!!!

R^{-1} exists <=> det(R) is non-Zero

<= det(R) = +/-1

(makes it right / left handed)

A Few Fun Facts...

proof: $A^{-1} = A^T$

- consider 2D (same game 3D)
- check that $A\,A^T = I$

$$A = \begin{pmatrix} cos\alpha & -sin\alpha \\ sin\alpha & cos\alpha \end{pmatrix} \quad A^T = \begin{pmatrix} cos\alpha & sin\alpha \\ -sin\alpha & cos\alpha \end{pmatrix}$$

use: $sin^2(x) + cos^2(x) = 1$

General Rotation (combined x,y,z rot.)

$$\begin{aligned}
& R(\alpha, \beta, \gamma) \\
= \quad & R_x(\alpha) \cdot R_y(\beta) \cdot R_z(\gamma) \\
= \quad & \begin{pmatrix} \cos(\alpha)\cos(\beta) & \cos(\alpha)\sin(\beta)\sin(\gamma) - \sin(\alpha)\cos(\gamma) & \cos(\alpha)\sin(\beta)\cos(\gamma) + \sin(\alpha)\sin(\gamma) \\ \sin(\alpha)\cos(\beta) & \sin(\alpha)\sin(\beta)\sin(\gamma) + \cos(\alpha)\cos(\gamma) & \sin(\alpha)\sin(\beta)\cos(\gamma) - \cos(\alpha)\sin(\gamma) \\ -\sin(\beta) & \cos(\beta)\sin(\gamma) & \cos(\beta)\cos(\gamma) \end{pmatrix}
\end{aligned}$$

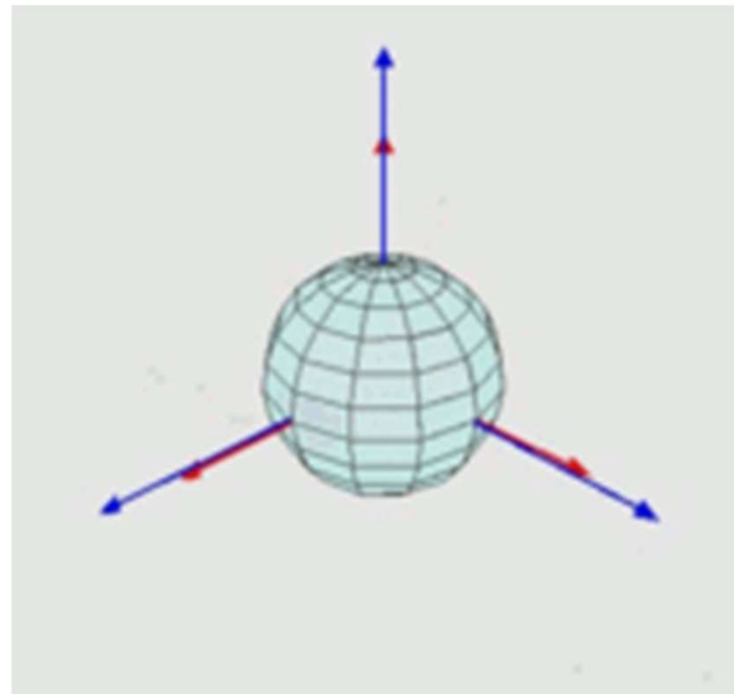
note: remember that matrix multiplication is ***not commutative***

=> order matters

=> bunch of possible angle orders

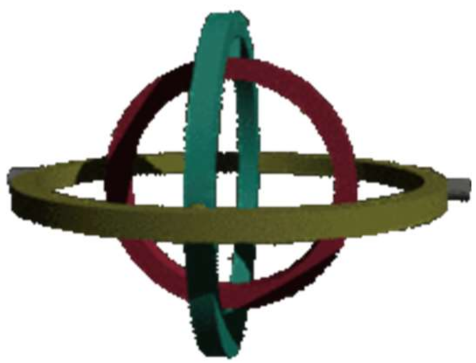
Euler angles

- family of conventions
- 3D orientation through sequence of 3 rotation angles
  – several possible rotation axes
    - X Y Z at start
    - x’ y’ z’ / x’’ y’’ z’’ after 1st / 2nd rotation
  – plus permutations
    - e.g., order X-Y-Z or Y-X-Z



Problem: Gimbal Lock

- no matter, which convention is used for Euler angles
- there is always a sequence that leads to a loss of 1 DoF
- as later axis of rotation gets co-aligned with previous one



gimbal lock: axes co-aligned => 1 DoF lost
(happens for any representation of rotation
with just 3 parameters)

solution: >3 parameters

- quaternions (4 parameters)
- rotation matrix (9 parameters)

General Rotation Matrix

NOTE: general rotation matrix

⇒in theory overdetermined, can avoid gimbal lock

⇒rotation e.g., via axis/angle (**v**,a), i.e., 4 parameters

⇒but needs conversions: matrix<-> axis/angle to do a rotation