

Controlling Robot Manipulators

Alexis Maldonado, Andreas Fedrizzi

Lehrstuhl für Informatik IX
Technische Universität München

Wintersemester 2008

Outline of the Course

- Introduction
 - Robot-Components
 - Manipulators
 - Configuration Space
- Software
 - Player
 - Gazebo
- Position Analysis
 - Vectors in Space
 - Representation of Rigid Bodies
 - Representation of Frame Transformations
 - Roboop
- Forward and Inverse Kinematics
- Modeling of Manipulators
 - DH Parameters
 - GLrooop
- Further topics
 - Trajectory Planning
 - Modern Control Methods

Outline

- 1 Transformation relative to the current body frame
- 2 Inverse of HT matrices
- 3 Forward Kinematics

Outline

- 1 Transformation relative to the current body frame
- 2 Inverse of HT matrices
- 3 Forward Kinematics

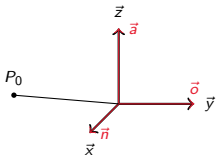
Transformation relative to the current body frame

- So far we have discussed transformations relative to a fixed reference frame, where all transformations had to be premultiplied.
- When you do a transformation relative to the current body frame, you have to **postmultiply** the corresponding HT matrix.
- The **result will be different** if you transform relative to the current frame instead of the reference frame (matrix multiplication is not commutative).
- Note that any point that is described relative to the moving frame is also always postmultiplied.

Example

- We revisit our example from the last lecture and transform a point P_0 at position $\langle 1, -1, 0.5 \rangle$ by the following operations.
 - Rotation of 50° about the a-axis.
 - Translation of $\langle 0.3, 1.2, -0.3 \rangle$.
 - Rotation of 150° about the n-axis.
 - Rotation of 20° about the o-axis.

$$P_0 = \text{trans} \begin{pmatrix} 1 \\ -1 \\ 0.5 \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

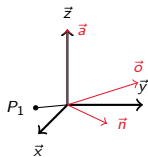
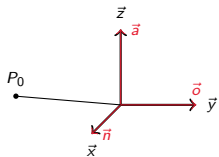


Example

- We revisit our example from the last lecture and transform a point P_0 at position $\langle 1, -1, 0.5 \rangle$ by the following operations.
 - Rotation of 50° about the a-axis.
 - Translation of $\langle 0.3, 1.2, -0.3 \rangle$.
 - Rotation of 150° about the n-axis.
 - Rotation of 20° about the o-axis.

$$P_0 = \text{trans} \begin{pmatrix} 1 \\ -1 \\ 0.5 \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P_1 = \text{rot}(a, 50^\circ) \times \text{trans} \begin{pmatrix} 1 \\ -1 \\ 0.5 \end{pmatrix} = \begin{bmatrix} 0.64 & -0.77 & 0 & 1.41 \\ 0.77 & 0.64 & 0 & 0.12 \\ 0 & 0 & 1 & 0.50 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

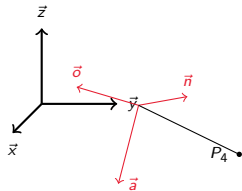
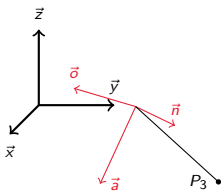
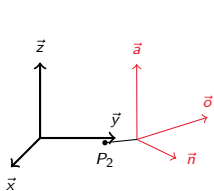


Example (cont.)

$$P_2 = \text{rot}(a, 50^\circ) \times \text{trans} \begin{pmatrix} 0.3 \\ 1.2 \\ -0.3 \end{pmatrix} \times \text{trans} \begin{pmatrix} 1 \\ -1 \\ 0.5 \end{pmatrix} = \begin{bmatrix} 0.64 & -0.77 & 0 & 0.68 \\ 0.77 & 0.64 & 0 & 1.12 \\ 0 & 0 & 1 & 0.20 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P_3 = \text{rot}(a, 50^\circ) \times \text{trans} \begin{pmatrix} 0.3 \\ 1.2 \\ -0.3 \end{pmatrix} \times \text{rot}(n, 150^\circ) \times \text{trans} \begin{pmatrix} 1 \\ -1 \\ 0.5 \end{pmatrix} = \begin{bmatrix} 0.64 & 0.66 & 0.38 & -0.56 \\ 0.77 & -0.56 & -0.32 & 2.16 \\ 0 & 0.5 & -0.87 & -1.17 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P_4 = \text{rot}(a, 50^\circ) \times \text{trans} \begin{pmatrix} 0.3 \\ 1.2 \\ -0.3 \end{pmatrix} \times \text{rot}(n, 150^\circ) \times \text{rot}(o, 20^\circ) \times \text{trans} \begin{pmatrix} 1 \\ -1 \\ 0.5 \end{pmatrix} = \begin{bmatrix} 0.47 & 0.66 & 0.58 & -0.63 \\ 0.83 & -0.56 & -0.04 & 2.37 \\ 0.30 & 0.5 & -0.81 & -0.91 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Outline

- 1 Transformation relative to the current body frame
- 2 Inverse of HT matrices**
- 3 Forward Kinematics

Inverse of HT matrices

- Inverses of HT matrices are important, because they **revert the direction** in which the matrix transforms a frame into another.
- If a HT matrix is given that represents the pose of the B_{21} robot relative to the world origin, the inverse matrix represents the pose of the world origin relative to the B_{21} robot:
 $({}^O T_{B_{21}})^{-1}$ is the same as ${}^{B_{21}} T_O$.
- To **compute the inverse** of a matrix you usually need to perform the following steps.
 - Calculate the determinant of the matrix.
 - Transpose the matrix.
 - Replace each element of the transposed matrix by its own minor (adjoint matrix).
 - Divide the converted matrix by the determinant.

Simple inversion for 4×4 HT matrices

- It can be shown that all **unitary matrices** can be inverted by just taking the **transposed** of the matrix.
- All 3×3 rotation matrices are unitary matrices.
- 4×4 transformation matrices can be inverted by splitting the matrix into two parts and
 - Transposing the 3×3 rotation matrix part.
 - Building the negative of the dot product of vector \vec{P} with each of the \vec{n} , \vec{o} , \vec{a} vectors for the 3×1 translation matrix part.

$$T = \begin{bmatrix} n_x & o_x & a_x & P_x \\ n_y & o_y & a_y & P_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T^{-1} = \begin{bmatrix} n_x & n_y & n_z & -\vec{P} \cdot \vec{n} \\ o_x & o_y & o_z & -\vec{P} \cdot \vec{o} \\ a_x & a_y & a_z & -\vec{P} \cdot \vec{a} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Example

- A matrix T is given by

$$T = \text{trans}(3, 2, 5) \times \text{rot}(y, 60^\circ) = \begin{bmatrix} 0.5 & 0 & 0.87 & 3 \\ 0 & 1 & 0 & 2 \\ -0.87 & 0 & 0.5 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

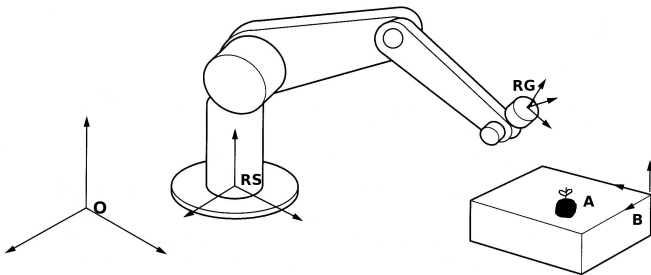
- Find the inverse matrix T^{-1}

$$T^{-1} = \begin{bmatrix} 0.5 & 0 & -0.87 & -(3 \cdot 0.5 + 2 \cdot 0 + 5 \cdot (-0.87)) \\ 0 & 1 & 0 & -(3 \cdot 0 + 2 \cdot 1 + 5 \cdot 0) \\ 0.87 & 0 & 0.5 & -(3 \cdot 0.87 + 2 \cdot 0 + 5 \cdot 0.5) \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} 0.5 & 0 & -0.87 & 2.85 \\ 0 & 1 & 0 & -2 \\ 0.87 & 0 & 0.5 & -5.11 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Application

- The B_{21} robot is asked to grasp an apple that is located in a box on a table.



Application

- To perform the task the robot first drives to the table. To grasp the apple, the robot has to know the HT matrix ${}^{RS}T_{Apple}$, because to grasp the apple, the frame ${}^{RS}T_{RG}$ must be equal to ${}^{RS}T_{Apple}$.
- When the frame ${}^{RS}T_{Apple}$ is known, the robot can compute the inverse kinematics solution for his manipulator to make ${}^{RS}T_{RG}$ equal to ${}^{RS}T_{Apple}$.
- The frame ${}^O T_{Apple}$ can be given in two independent paths

$${}^O T_{B_{21}} \times {}^{B_{21}} T_{RS} \times {}^{RS} T_{Apple} = {}^O T_{Table} \times {}^{Table} T_{Box} \times {}^{Box} T_{Apple}$$

Application

- To perform the task the robot first drives to the table. To grasp the apple, the robot has to know the HT matrix ${}^{RS}T_{Apple}$, because to grasp the apple, the frame ${}^{RS}T_{RG}$ must be equal to ${}^{RS}T_{Apple}$.
- When the frame ${}^{RS}T_{Apple}$ is known, the robot can compute the inverse kinematics solution for his manipulator to make ${}^{RS}T_{RG}$ equal to ${}^{RS}T_{Apple}$.
- The frame ${}^O T_{Apple}$ can be given in two independent paths
 ${}^O T_{B_{21}} \times {}^{B_{21}} T_{RS} \times {}^{RS} T_{Apple} = {}^O T_{Table} \times {}^{Table} T_{Box} \times {}^{Box} T_{Apple}$
- We usually know all frames except ${}^{RS}T_{Apple}$.

Using inverse matrices to compute ${}^{RS}T_{Apple}$

$$(1) \quad {}^O T_{B_{21}} \times {}^{B_{21}} T_{RS} \times {}^{RS} T_{Apple} = {}^O T_{Table} \times {}^{Table} T_{Box} \times {}^{Box} T_{Apple}$$

Matrices do not allow division, so we need to multiply their inverse matrices to cancel out ${}^O T_{B_{21}}$ and ${}^{B_{21}} T_{RS}$.

$$(2) \quad ({}^{B_{21}} T_{RS})^{-1} \times ({}^O T_{B_{21}})^{-1} \times {}^O T_{B_{21}} \times {}^{B_{21}} T_{RS} \times {}^{RS} T_{Apple} = \\ ({}^{B_{21}} T_{RS})^{-1} \times ({}^O T_{B_{21}})^{-1} \times {}^O T_{Table} \times {}^{Table} T_{Box} \times {}^{Box} T_{Apple}$$

$$(3) \quad {}^{RS} T_{Apple} = {}^{RS} T_{B_{21}} \times {}^{B_{21}} T_O \times {}^O T_{Table} \times {}^{Table} T_{Box} \times {}^{Box} T_{Apple}$$

Remember that $({}^O T_{B_{21}})^{-1}$ is the same as ${}^{B_{21}} T_O$.

\Rightarrow Because we usually know ${}^O T_{B_{21}}$ (from the localization device) and ${}^{B_{21}} T_{RS}$ (it is fixed), we need to be able to compute their inverse matrices to get ${}^{RS} T_{Apple}$.

Outline

- 1 Transformation relative to the current body frame
- 2 Inverse of HT matrices
- 3 Forward Kinematics**

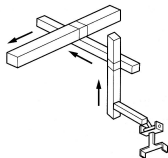
Forward Kinematics

- **Task:** Compute the manipulator's endeffector pose in cartesian space, when all joint angles and link lengths are given.
- **Input:** All joint angles and link lengths of a manipulator.
- **Output:** Position and orientation (=pose) of the endeffector.
- **Examples**
 - Cartesian robot
 - Cylindrical robot
 - Spherical robot (homework)
 - Roll, Pitch, Yaw angles
 - Euler angles

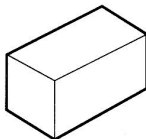
Forward Kinematics for a cartesian manipulator

- Robot design: Robot design: 3 mutually perpendicular, translational joints with a joint state of $Cart(t_n, t_o, t_a) = \langle l_1, l_2, l_3 \rangle$.

Robot Design



Workspace



- HT matrix for cylindrical manipulator:

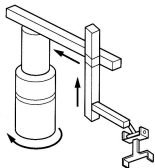
$$HT_{cart} = \text{trans} \begin{pmatrix} l_1 \\ 0 \\ 0 \end{pmatrix} \times \text{trans} \begin{pmatrix} 0 \\ l_2 \\ 0 \end{pmatrix} \times \text{trans} \begin{pmatrix} 0 \\ 0 \\ l_3 \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & l_1 \\ 0 & 1 & 0 & l_2 \\ 0 & 0 & 1 & l_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\Rightarrow \text{Position}_{cart} = \begin{pmatrix} l_1 \\ l_2 \\ l_3 \end{pmatrix} \Rightarrow \text{Rotation}_{cart} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

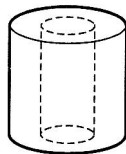
Forward Kinematics for a cylindrical manipulator

- Robot design: translational joint along the a-axis, rotational joint around the a-axis, and translational joint along the n-axis with a joint state of $Cyl(t_a, r_a, t_n) = \langle l_1, \alpha_2, l_3 \rangle$.

Robot Design



Workspace



- HT matrix for cylindrical manipulator:

$$HT_{cyl} = \text{trans} \begin{pmatrix} 0 \\ 0 \\ l_1 \end{pmatrix} \times \text{rot}(a, \alpha_2) \times \text{trans} \begin{pmatrix} l_3 \\ 0 \\ 0 \end{pmatrix} = \begin{bmatrix} C\alpha_2 & -S\alpha_2 & 0 & l_3 C\alpha_2 \\ S\alpha_2 & C\alpha_2 & 0 & l_3 S\alpha_2 \\ 0 & 0 & 1 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\Rightarrow \text{Position}_{cyl} = \begin{pmatrix} l_3 C\alpha_2 \\ l_3 S\alpha_2 \\ l_1 \end{pmatrix} \quad \Rightarrow \text{Rotation}_{cyl} = \begin{bmatrix} C\alpha_2 & -S\alpha_2 & 0 \\ S\alpha_2 & C\alpha_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Equations for orientation

- We assume that the robot endeffector has already moved to a certain position.
- The task is now to rotate the endeffector appropriately so that it can grasp an object at that position.
- There are several notations for expressing orientations.
 - Roll, Pitch, Yaw angles (RPY)
 - Euler angles
 - Articulated Joints
 - Quaternions

Forward Kinematics for RPY angles

- RPY (Roll, Pitch, Yaw) angles specify a sequence of 3 rotations around the a-axis, the o-axis, and the n-axis in this order. The joint state is $RPY(r_a, r_o, r_n) = \langle \alpha_1, \alpha_2, \alpha_3 \rangle$.
- All rotations are postmultiplied, because we do not want to change the position of the origin of the endeffector frame and therefore all rotations are relative to the moving body frame.
- HT matrix for RPY angles:

$$HT_{rpy} = rot(a, \alpha_1) \times rot(o, \alpha_2) \times rot(n, \alpha_3) =$$

$$\begin{bmatrix} C_{\alpha_1} C_{\alpha_2} & C_{\alpha_1} S_{\alpha_2} S_{\alpha_3} - S_{\alpha_1} C_{\alpha_3} & C_{\alpha_1} S_{\alpha_2} C_{\alpha_3} + S_{\alpha_1} S_{\alpha_3} & 0 \\ S_{\alpha_1} C_{\alpha_2} & S_{\alpha_1} S_{\alpha_2} S_{\alpha_3} + C_{\alpha_1} C_{\alpha_3} & S_{\alpha_1} S_{\alpha_2} C_{\alpha_3} - C_{\alpha_1} S_{\alpha_3} & 0 \\ -S_{\alpha_2} & C_{\alpha_2} S_{\alpha_3} & C_{\alpha_2} C_{\alpha_3} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\Rightarrow Pos_{rpy} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\Rightarrow Rot_{rpy} = \begin{bmatrix} C_{\alpha_1} C_{\alpha_2} & C_{\alpha_1} S_{\alpha_2} S_{\alpha_3} - S_{\alpha_1} C_{\alpha_3} & C_{\alpha_1} S_{\alpha_2} C_{\alpha_3} + S_{\alpha_1} S_{\alpha_3} \\ S_{\alpha_1} C_{\alpha_2} & S_{\alpha_1} S_{\alpha_2} S_{\alpha_3} + C_{\alpha_1} C_{\alpha_3} & S_{\alpha_1} S_{\alpha_2} C_{\alpha_3} - C_{\alpha_1} S_{\alpha_3} \\ -S_{\alpha_2} & C_{\alpha_2} S_{\alpha_3} & C_{\alpha_2} C_{\alpha_3} \end{bmatrix}$$

Euler angles

- Euler angles are a **generalization of RPY angles**, where the axes around which the 3 rotations are performed can be chosen freely.
- RPY always rotates around the a-axis, then around the o-axis, and finally around the n-axis.
- RPY would correspond to the Euler angle $EU(a,o,n)$.
- Although the Euler angles rotate around the moving body frame, they are usually **specified with x- y- and z-axis**, so that RPY corresponds to $EU(z,y,x)$ in the common notation.
- Euler angles allow **12 meaningful combinations** of rotations:
 $EU(x,y,x)$, $EU(x,y,z)$, $EU(x,z,x)$, $EU(x,z,y)$,
 $EU(y,x,y)$, $EU(y,x,z)$, $EU(y,z,x)$, $EU(y,z,y)$,
 $EU(z,x,y)$, $EU(z,x,z)$, $EU(z,y,x)$, $EU(z,y,z)$
- Note that it is allowed to rotate around the same axis twice.
- The most frequently used Euler angle is $EU(z,x,z)/EU(a,n,a)$.

Forward Kinematics for EU(a,n,a) / EU(z,x,z)

- The joint state for the EU(a,n,a) or EU(z,x,z) angle is $EU(r_a, r_n, r_a) = \langle \alpha_1, \alpha_2, \alpha_3 \rangle$.
- HT matrix for EU(a,n,a).

$$HT_{eu(a,n,a)} = rot(a, \alpha_1) \times rot(n, \alpha_2) \times rot(a, \alpha_3) =$$

$$\begin{bmatrix} C_{\alpha_1} C_{\alpha_3} - S_{\alpha_1} C_{\alpha_2} S_{\alpha_3} & -C_{\alpha_1} S_{\alpha_3} - S_{\alpha_1} C_{\alpha_2} C_{\alpha_3} & S_{\alpha_1} S_{\alpha_2} & 0 \\ S_{\alpha_1} C_{\alpha_3} + C_{\alpha_1} C_{\alpha_2} S_{\alpha_3} & -S_{\alpha_1} S_{\alpha_3} + C_{\alpha_1} C_{\alpha_2} C_{\alpha_3} & -C_{\alpha_1} S_{\alpha_2} & 0 \\ S_{\alpha_2} S_{\alpha_3} & S_{\alpha_2} C_{\alpha_3} & C_{\alpha_2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\Rightarrow Pos_{eu(a,n,a)} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\Rightarrow Rot_{eu(a,n,a)} = \begin{bmatrix} C_{\alpha_1} C_{\alpha_3} - S_{\alpha_1} C_{\alpha_2} S_{\alpha_3} & -C_{\alpha_1} S_{\alpha_3} - S_{\alpha_1} C_{\alpha_2} C_{\alpha_3} & S_{\alpha_1} S_{\alpha_2} \\ S_{\alpha_1} C_{\alpha_3} + C_{\alpha_1} C_{\alpha_2} S_{\alpha_3} & -S_{\alpha_1} S_{\alpha_3} + C_{\alpha_1} C_{\alpha_2} C_{\alpha_3} & -C_{\alpha_1} S_{\alpha_2} \\ S_{\alpha_2} S_{\alpha_3} & S_{\alpha_2} C_{\alpha_3} & C_{\alpha_2} \end{bmatrix}$$