

Controlling Robot Manipulators

Alexis Maldonado, Andreas Fedrizzi

Lehrstuhl für Informatik IX
Technische Universität München

Sommersemester 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
- Software
 - Roboop
 - GLroboop
- Forward and Inverse Kinematics
- Modeling of Manipulators
 - DH Parameters
- Further topics
 - Trajectory Planning
 - Modern Control Methods

Outline

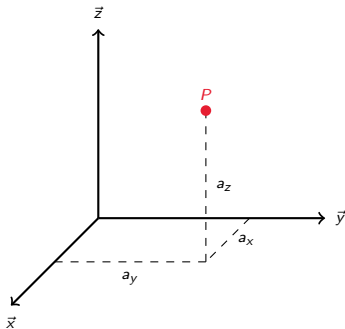
- 1 Vectors in Space
- 2 Representation of Rigid Bodies
- 3 Representation of Frame Transformations
- 4 Introduction to Roboop and GLrooop

Outline

- 1 Vectors in Space
- 2 Representation of Rigid Bodies
- 3 Representation of Frame Transformations
- 4 Introduction to Roboop and GLrooop

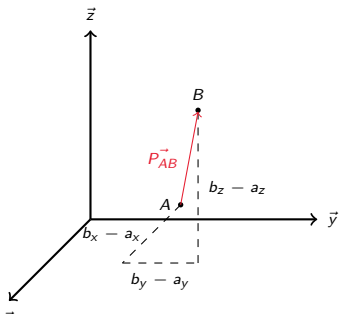
Representation of a point in space

- Described by three coordinates relative to a reference frame.
- $P = a_x \vec{x} + a_y \vec{y} + a_z \vec{z}$



Representation of a vector in space

- Described by three coordinates for the tail and three coordinates for the head relative to a fixed reference frame.
- $\vec{P}_{AB} = (b_x - a_x)\vec{x} + (b_y - a_y)\vec{y} + (b_z - a_z)\vec{z}$
- Corresponding matrix form: $P_{AB}^{\vec{}} = \begin{pmatrix} b_x - a_x \\ b_y - a_y \\ b_z - a_z \end{pmatrix}$



Scale factor

- For a more compact notation we will set
 $p_x = b_x - a_x$, $p_y = b_y - a_y$ and $p_z = b_z - a_z$
- A vector can be scaled by a scale factor to **increase or decrease its length**.
- For this we add a fourth component w to the vector, so that it can be written as

$$P'_{AB} = \begin{pmatrix} p'_x \\ p'_y \\ p'_z \\ w \end{pmatrix}, \text{ where } p'_x = p_x * w, p'_y = p_y * w \text{ and } p'_z = p_z * w.$$

Scale factor

- For a more compact notation we will set $p_x = b_x - a_x$, $p_y = b_y - a_y$ and $p_z = b_z - a_z$
- A vector can be scaled by a scale factor to **increase or decrease its length**.
- For this we add a fourth component w to the vector, so that it can be written as

$$P'_{AB} = \begin{pmatrix} p'_x \\ p'_y \\ p'_z \\ w \end{pmatrix}, \text{ where } p'_x = p_x * w, p'_y = p_y * w \text{ and } p'_z = p_z * w.$$

- **Example:** A vector is described as $\vec{P} = 3\vec{x} + 5\vec{y} + 2\vec{z}$. Compute the corresponding vector \vec{P}' with a scale factor of 2.

Scale factor

- For a more compact notation we will set
 $p_x = b_x - a_x$, $p_y = b_y - a_y$ and $p_z = b_z - a_z$
- A vector can be scaled by a scale factor to **increase or decrease its length**.
- For this we add a fourth component w to the vector, so that it can be written as

$$P'_{AB} = \begin{pmatrix} p'_x \\ p'_y \\ p'_z \\ w \end{pmatrix}, \text{ where } p'_x = p_x * w, p'_y = p_y * w \text{ and } p'_z = p_z * w.$$

- **Example:** A vector is described as $\vec{P} = 3\vec{x} + 5\vec{y} + 2\vec{z}$. Compute the corresponding vector \vec{P}' with a scale factor of 2.
- **Solution:** $\vec{P}' = \begin{pmatrix} 6 \\ 10 \\ 4 \\ 2 \end{pmatrix}$

Unit vector

- The unit vector is a normalized version of a vector that **keeps the orientation**, but **changes the length** so that it equals unity.
- To do this, each component of the vector will be divided by the square root of the sum of the squares of the three components.

Unit vector

- The unit vector is a normalized version of a vector that **keeps the orientation**, but **changes the length** so that it equals unity.
- To do this, each component of the vector will be divided by the square root of the sum of the squares of the three components.
- **Example:** A vector is described as $\vec{P} = 3\vec{x} + 5\vec{y} + 2\vec{z}$. Compute the corresponding unit vector P_{unit} .

Unit vector

- The unit vector is a normalized version of a vector that **keeps the orientation**, but **changes the length** so that it equals unity.
- To do this, each component of the vector will be divided by the square root of the sum of the squares of the three components.
- **Example:** A vector is described as $\vec{P} = 3\vec{x} + 5\vec{y} + 2\vec{z}$. Compute the corresponding unit vector $P_{unit}^{\vec{}}$.
- **Solution:** $l_P = \sqrt{p_x^2 + p_y^2 + p_z^2} = 6.16$

$$\Rightarrow P_{unit}^{\vec{}} = \begin{pmatrix} 0.487 \\ 0.811 \\ 0.324 \end{pmatrix}$$

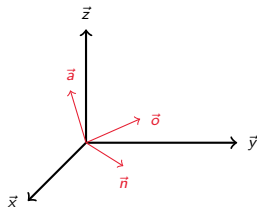
Outline

- 1 Vectors in Space
- 2 Representation of Rigid Bodies**
- 3 Representation of Frame Transformations
- 4 Introduction to Roboop and GLrooop

Representation of a frame at the origin of a fixed reference frame

- A dependend frame that is centered at the origin of a fixed reference frame is described by three unit vectors, which represent the **orientation**.
- The unit vectors are **perpendicular** to each other, and are called \vec{n} , \vec{o} , \vec{a} for normal, orientation and approach.
- Each unit vector in the dependend coordinate frame is represented by its three components in the reference frame.

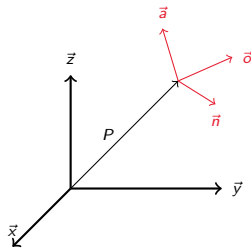
$$F = \begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix}$$



General Representation of a frame in a fixed reference frame

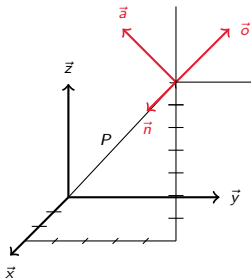
- Beside the rotation, we have to express the **position of the origin** of the dependend frame relative to a fixed reference frame.
- For achieving this, we use a vector that goes from the origin of the reference frame to the origin of the dependend frame. This vector is expressed in the coordinates of the reference frame.

$$F = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



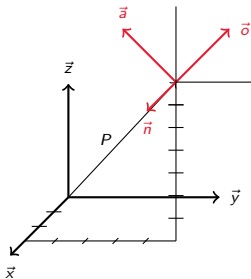
Example

- The dependend frame F shown in the figure is located at $\langle 3, 5, 7 \rangle$. Its n -axis is parallel to the x -axis of the reference frame, its o -axis is the y -axis of the reference frame, rotated by 45° around the x -axis and its a -axis is the z -axis of the reference frame rotated by 45° around the x -axis.
- How can the frame be described in a matrix?



Example

- The dependend frame F shown in the figure is located at $\langle 3, 5, 7 \rangle$. Its n -axis is parallel to the x -axis of the reference frame, its o -axis is the y -axis of the reference frame, rotated by 45° around the x -axis and its a -axis is the z -axis of the reference frame rotated by 45° around the x -axis.
- How can the frame be described in a matrix?

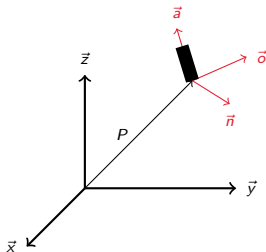


$$F = \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 0.707 & -0.707 & 5 \\ 0 & 0.707 & 0.707 & 7 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Representation of a rigid body

- A rigid body can be represented in space by **attaching a frame**.
- Because the body frame is always fixed in the same position and orientation of the rigid body, the position and orientation of the rigid body relative to the reference frame is always known by comparing its body frame with the reference frame.
- So representing the position and orientation of a rigid body in space is like representing a frame in a fixed reference frame.

$$F_{body} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Representation of a rigid body

- A rigid body is described by **6 values** (DOFs).
- The matrix has 16 values, 3 for positioning, 9 for rotation and 4 for filling up to a 4×4 matrix.
- There are 6 constraints on the 3×3 rotation matrix, so that many matrix configurations are forbidden.
- Each of the 3 **vector's length** must be equal to unity
 $\rightarrow |\vec{n}| = 1, |\vec{o}| = 1, \text{ and } |\vec{a}| = 1.$
- The 3 unit vectors $\vec{n}, \vec{o}, \vec{a}$ are **mutually perpendicular**
 $\rightarrow \vec{n} \cdot \vec{o} = 0, \vec{n} \cdot \vec{a} = 0, \text{ and } \vec{a} \cdot \vec{o} = 0.$
- Instead of the dot product, the **cross product** can be used to express the last constraint
 $\rightarrow \vec{n} \times \vec{o} = \vec{a}, \vec{o} \times \vec{a} = \vec{n}, \text{ and } \vec{a} \times \vec{n} = \vec{o}.$

Representation of a rigid body (cont.)

- As a result of the constraints, the rotation matrix is defined by 3 values instead of 9 values.
- So for any rotation matrix where 3 values are given, the remaining 6 values can be computed by solving the constraint equations.
- It is possible, that there are more possibilities for choosing the remaining values.

$$F = \begin{bmatrix} ? & 0 & ? & 5 \\ 0.707 & ? & ? & 3 \\ ? & ? & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \Rightarrow F = \begin{bmatrix} 0.707 & 0 & 0.707 & 5 \\ 0.707 & 0 & -0.707 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{ or}$$

$$F = \begin{bmatrix} -0.707 & 0 & -0.707 & 5 \\ 0.707 & 0 & -0.707 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Outline

- 1 Vectors in Space
- 2 Representation of Rigid Bodies
- 3 Representation of Frame Transformations**
- 4 Introduction to Roboop and GLrooop

Representation of Frame Transformations

- A transformation is defined as making a **movement in space**.
- A transformation can be
 - a pure translation
 - a pure rotation about an axis
 - a combination of translations and rotations
- We use **homogenous transformation matrices** for representing transformations, which are 4×4 matrices that are like the matrices that we used for frame representation.
- Square matrices have useful properties like
 - multiplication in any order is possible
 - calculation of the inverse matrix is easier (an example will follow)

Naming conventions

- For general homogenous transformation matrices
 - Homogenous Transformation (HT) of frame R relative to the reference frame O (world origin): ${}^O T_R$
 - Special cases of a HT
 - Pure translation: $Trans(d_x, d_y, d_z)$
 - Pure rotation around one axis: $Rot(x, \theta)$, $Rot(y, \theta)$, $Rot(z, \theta)$
 - Rotation matrix defined through the angles roll, pitch, yaw: $RPY(\phi_a, \phi_o, \phi_n)$
- For coordinate frames in our practical course
 - World origin: O
 - Base of B21: $B21$
 - Right shoulder of B21: RS
 - Right gripper of B21: RG
 - Right laser range finder of B21: RL
 - Frames on the left side are named LS , LG , LL accordingly

Representation of a pure translation relative to a reference frame

- Only the position of the origin of the body frame changes, while the directional unit vectors point into the same direction.
- The translation matrix that translates the body frame's origin is given relative to the reference frame.
- In matrix form, the new body frame representation can be found by **premultiplying** the body frame with the translation matrix.
- Computation: Current body frame: F_{cur}

Translation matrix: $Trans(d_x, d_y, d_z)$

$$F_{new} = Trans(d_x, d_y, d_z) \times F_{cur}$$

$$F_{new} = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n_x & o_x & a_x & p_x + d_x \\ n_y & o_y & a_y & p_y + d_y \\ n_z & o_z & a_z & p_z + d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Representation of a pure rotation about an axis of a reference frame

- Matrix for rotating a point by θ around the x-axis.

$$Rot(x, \theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Matrix for rotating a point by θ around the y-axis.

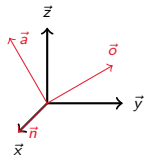
$$Rot(y, \theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Matrix for rotating a point by θ around the z-axis.

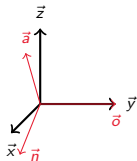
$$Rot(z, \theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Visualization of rotations

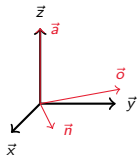
$$\text{Rot}(x, 30\text{deg}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.866 & -0.5 & 0 \\ 0 & 0.5 & 0.866 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$\text{Rot}(y, 30\text{deg}) = \begin{bmatrix} 0.866 & 0 & 0.5 & 0 \\ 0 & 1 & 0 & 0 \\ -0.5 & 0 & 0.866 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$\text{Rot}(z, 30\text{deg}) = \begin{bmatrix} 0.866 & -0.5 & 0 & 0 \\ 0.5 & 0.866 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Representation of a combined transformation relative to a reference frame

- Consists of a number of **successive translations and rotations** about the fixed reference frame or the moving frame's axes.
- Any transformation can be resolved into a set of translations and rotations in a particular order.
- The **multiplication order** of the transformation matrices is important.
- It is a difference to first rotate a frame and then translating it, or first translating a frame and then rotating it.

Visualization of transformation order

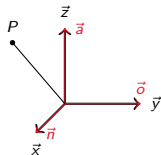
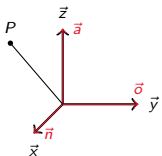
The position $\langle 0.5, -0.5, 1 \rangle$ of a point P is given relative to the reference frame and transformed by a translation and rotation.

Translation followed by rotation

$$P'' = \text{rot}(x, 45^\circ) \times \text{trans}(1, 1, 0) \times P$$

Rotation followed by translation

$$P'' = \text{trans}(1, 1, 0) \times \text{rot}(x, 45^\circ) \times P$$

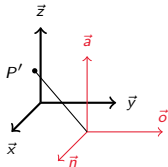
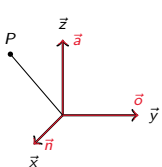


Visualization of transformation order

The position $\langle 0.5, -0.5, 1 \rangle$ of a point P is given relative to the reference frame and transformed by a translation and rotation.

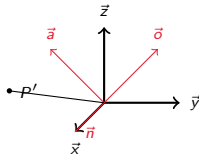
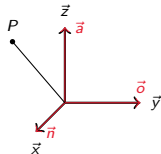
Translation followed by rotation

$$P'' = \text{rot}(x, 45^\circ) \times \text{trans}(1, 1, 0) \times P$$



Rotation followed by translation

$$P'' = \text{trans}(1, 1, 0) \times \text{rot}(x, 45^\circ) \times P$$



Visualization of transformation order

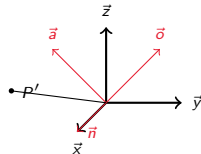
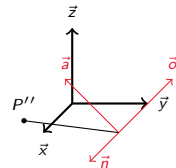
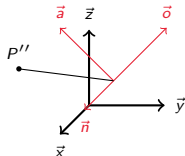
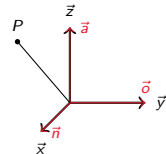
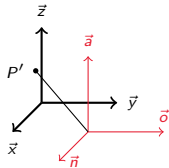
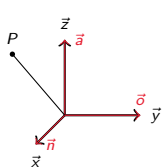
The position $\langle 0.5, -0.5, 1 \rangle$ of a point P is given relative to the reference frame and transformed by a translation and rotation.

Translation followed by rotation

$$P'' = \text{rot}(x, 45^\circ) \times \text{trans}(1, 1, 0) \times P$$

Rotation followed by translation

$$P'' = \text{trans}(1, 1, 0) \times \text{rot}(x, 45^\circ) \times P$$

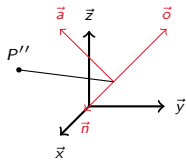
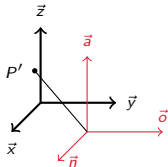
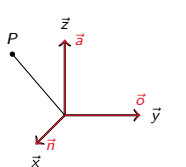


Visualization of transformation order

The position $\langle 0.5, -0.5, 1 \rangle$ of a point P is given relative to the reference frame and transformed by a translation and rotation.

Translation followed by rotation

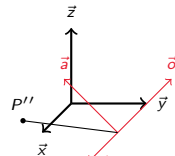
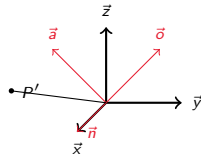
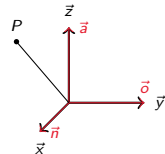
$$P'' = \text{rot}(x, 45^\circ) \times \text{trans}(1, 1, 0) \times P$$



$$P'' = \begin{bmatrix} 1.50 \\ -0.35 \\ 1.06 \end{bmatrix}$$

Rotation followed by translation

$$P'' = \text{trans}(1, 1, 0) \times \text{rot}(x, 45^\circ) \times P$$

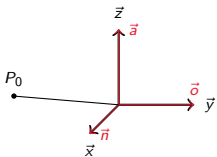


$$P'' = \begin{bmatrix} 1.50 \\ -0.06 \\ 0.35 \end{bmatrix}$$

Example

- The goal is to transform a point P_0 at position $\langle 1, -1, 0.5 \rangle$ by the following operations.
 - Rotation of 50° about the z-axis.
 - Translation of $\langle 0.3, 1.2, -0.3 \rangle$.
 - Rotation of 150° about the x-axis.
 - Rotation of 20° about the y-axis.

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

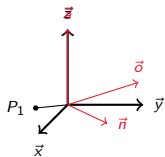
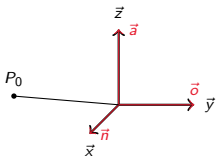


Example

- The goal is to transform a point P_0 at position $\langle 1, -1, 0.5 \rangle$ by the following operations.
 - Rotation of 50° about the z-axis.
 - Translation of $\langle 0.3, 1.2, -0.3 \rangle$.
 - Rotation of 150° about the x-axis.
 - Rotation of 20° about the y-axis.

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

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



Outline

- 1 Vectors in Space
- 2 Representation of Rigid Bodies
- 3 Representation of Frame Transformations
- 4 Introduction to Roboop and GLrooop**

Roboop

- Object oriented toolbox in C++ for **manipulator simulation**.
- Wrapper for the **newmat** library that implements numerous efficient matrix operations like
 - multiplication, addition, subtraction
 - computing the dot product of two vectors
 - finding the inverse or transposed of a matrix
 - and many more
- Allows the specification of robotic manipulators in **standard DH** and **modified DH** notation.
- Performs **forward kinematic** and **inverse kinematic** computations.
- Web resources concerning Roboop
 - <http://www.cours.polymtl.ca/roboop/>
(Roboop and GLRoboop)
 - http://www.robertnz.net/nm_intro.htm (newmat)

Creating a translation matrix in Roboop

```

#include <iostream>
#include <utils.h>
#include <newmat.h>
using namespace std;

int main(int argc, const char **argv)
{
    ColumnVector translation(3);
    Matrix oTb21(4,4);

    translation << 1.0 << 1.0 << 1.0;
    oTb21 = trans(translation);

    cout << setprecision(3); // show only 3 decimal digits
    cout << "oTb21" << endl << oTb21 << endl;
    cout << "oTb21*oTb21.i()" <<endl<< oTb21*oTb21.i() <<endl;
}

```

Compile with: `g++ test.cpp -lroboop -lnewmat -o test`

Set the `-I` and `-L` flags correct, if headers or libraries are not found.

Creating a homogenous transformation matrix

```
#include <rooop/ utils .h>
#include <rooop/newmat.h>
#define DTOR(d) ((d) * M_PI / 180)
using namespace std;

int main(int argc, const char **argv)
{
    ColumnVector translation(3), rot_axis(3);
    Matrix oTb21(4,4);

    translation << 3.0 << 5.0 << 7.0;
    oTb21 = trans(translation) * rotx(DTOR(45.0));
}
```

This piece of code creates the matrix from slide 16.

GLroboop

- OpenGL interface using the package ROBOOP.
- Reads manipulator models specified through DH parameters and **visualizes the manipulator structure**.
- Inverse kinematic queries are solved and the resulting manipulator configuration is displayed.
- Great tool for manually observing and evaluating the correctness of manipulator designs.