

Übungen zum Praktikum Control of Robot Manipulators

DH parameters and GLrooop

Compiling GLrooop

The following steps should guide you through the compiling process of Roboop.

- Download the newest version of GLrooop from the webpage (currently this is <http://www.cours.polymtl.ca/rooop/download.php>).
- Uncompress the downloaded file (currently `tar xvzf GLrooop-1.04.tar.gz`)
- Go to the GLrooop directory (`cd GLrooop`)
- Run the `gcc` makefile with `make -f makefile.gcc`

Pitfalls

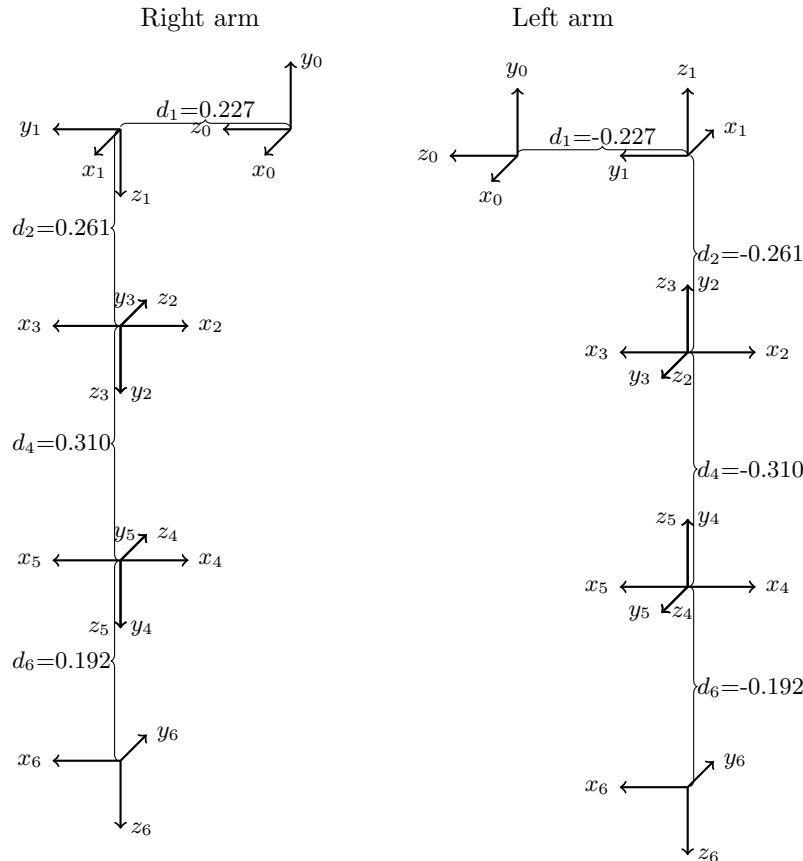
The makefile `makefile.gcc` of GLrooop is poorly written. It has hardcoded references to the directory `../rooop` for including header and library files of Roboop. So GLrooop expects that Roboop and GLrooop are in the same directory (e.g. `/programs/rooop` and `/programs/GLrooop`) and that the directory name of Roboop is `rooop`. This is usually not the case, because by default Roboop is extracted into a directory called `rooop-1.31`. The easiest way to fix this, is to change `makefile.gcc` in a way, that all links to `rooop` are replaced by links to `rooop-1.31`.

Using GLrooop

- Call `./GLrooop` to run GLrooop.
- GLrooop stores manipulator designs in `xyz.conf` files in its subdirectory `models`.
- You can load models easily by clicking on File → Open file and choosing one of the `xyz.conf` files.
- There are many popular manipulators predefined in the `models` subdirectory, which are great for studying the syntax that is used to describe a manipulator design. Important to know is that the filename of a manipulator must match its name in brackets (e.g. storing a manipulator in `powercube_right.conf` makes it necessary to call the manipulator `[powercube_right]`. The names of all links have to be called `[powercube_right_LINK1]`, `[powercube_right_LINK2]` and so on. The number of DOFs must match the number that you specify for the parameter `dof`. Note that GLrooop is not able to handle more than 6 DOFs.
- If you mess up an aspect of the syntax, GLrooop will not load your model.
- GLrooop allows you to specify the same 23 parameters like Roboop, where we will again use just the first 5 parameters `joint_type`, θ , d , a , and α .
- The visualization window can be zoomed with the keys `q` and `e`, rotated horizontally with `a` and `d`, and rotated vertically with `w` and `s`.

Homework

- The picture below shows a schematic drawing of the right and left powercube arm. If a translational value is not specified, like d_3 , d_5 or all a -values, the value equals '0'. Compute all DH parameters for both manipulators.



- Compile GLroboop. Use your computed DH parameters from task 1 to build the right manipulator in GLroboop. Test your manipulator design by changing the joint angles of all joints one after another. Compare the resulting arm movement in GLroboop's visualization window to the corresponding movement in Gazebo when you change the angle of the same joint with pi-actarray. Are the movements identical? If not, your DH parameters are wrong and you have to correct them. Another check to find out if your design is correct, you can enter different configuration space poses (given as $\langle \theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6 \rangle$) and compare the resulting workspace pose (given as $\langle X, Y, Z, Yaw, Pitch, Roll \rangle$). The configuration space pose $\langle -0.53, -3.28, 4.38, 3.02, 3.72, -4.14 \rangle$ for example should lead to the workspace pose $\langle 0.248, -0.512, 0.148, -1.969, -1.031, -1.507 \rangle$.

- Which joint angles are required to reach the following endeffector workspace poses (given as $\langle X, Y, Z, Yaw, Pitch, Roll \rangle$)? Use inverse kinematic queries in GLroboop and copy the displayed values for the joint angles.

Workspace poses:

- $\langle -0.19, -0.47, 0.09, -2.31, -0.62, -2.03 \rangle$
- $\langle 0.60, -0.00, 0.05, -2.67, -0.71, -0.31 \rangle$
- $\langle 0.31, 0.41, -0.19, 3.14, -0.27, 0.9 \rangle$
- $\langle 0.22, 0.21, 0.59, 0.79, -0.15, 2.71 \rangle$
- $\langle 0.33, -0.24, -0.25, 2.88, 0.15, 1.59 \rangle$