Universität
Bremen

Institute for Artificial Intelligence

Faculty 03

Mathematics &

Computer Science

Robot Programming with ROS

4. Motors and Kinematics

Arthur Niedźwiecki

07th May 2025

Artificial Intelligence

Overview

Overview

What makes a robot?

A Robot is an electro-mechanical device, composed of a collection of bodies (links), which are combined by joints. A robot is equipped with actuators (motors), that can move neighbouring links relative to each other by exerting forces.

What makes a robot?

A Robot is an electro-mechanical device, composed of a collection of bodies (links), which are combined by joints. A robot is equipped with actuators (motors), that can move neighbouring links relative to each other by exerting forces.

Robots are equipped with control programs that are designed to accomplish tasks by moving the body of the robot in a task-driven way.

The control software will be addressed in subsequent modules.

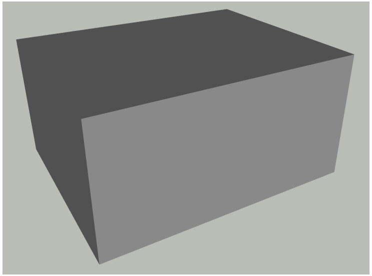
Basic Principles

- Position the End effector (Move the end effector in the room)
- Touch objects (Manipulation)
- Exercise a force on the ground relative to itself (Move in the plane)

Links

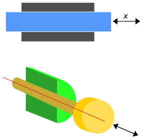
Rigid body with visual and collision features and known inertia properties.

```
<link name="${name}">
    <visual>
        <geometry>
            <box size="${base_length} ${base_width} ${base_height}"/>
        </geometry>
        <material name="LightGrey">
            <color rgba="0.7 0.7 0.7 1.0"/>
        </material>
    </visual>
    <collision>
        <geometry>
            <box size="${base_length} ${base_width} ${base_height}"/>
        </geometry>
    </collision>
    <inertial>
        <origin xyz="0 0 0.5" rpy="0 0 0"/>
        <mass value="100"/>
        <inertia ixx="100" ixy="0" ixz="0" iyy="100" iyz="0" izz="100"/>
    </inertial>
</link>
```

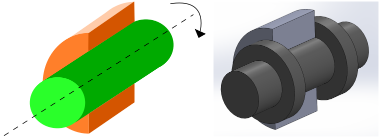


Joints

- Two type of joints: **revolute** (rotary motion) and **prismatic** (linear motion)
- Classical industrial application: high stiffness in links and joints
  - Advantage: position accuracy
  - Disadvantage: may lead to high forces exerted by robot → danger!
  - Note: link deflection under load or joint play might reduce precision in the real-world



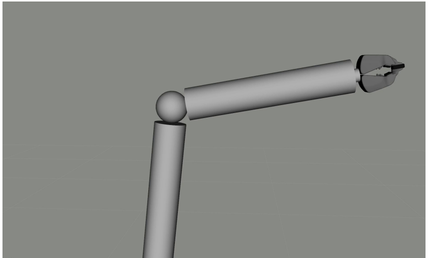
One translational degree of freedom



One rotary degree of freedom

Joints

```
<joint name="${prefix}shoulder_lift_joint" type="continuous">
    <parent link="${prefix}base_link"/>
    <child link = "${prefix}upper_arm_link"/>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <axis xyz="0 0 1"/>
    <limit lower="${shoulder_lift_lower_limit}" upper="${shoulder_lift_upper_limit}"
           effort="150.0" velocity="3.15"/>
    <dynamics damping="1.0" friction="100.0"/>
</joint>
```



Non-classical approach: soft robots

- Impedance control

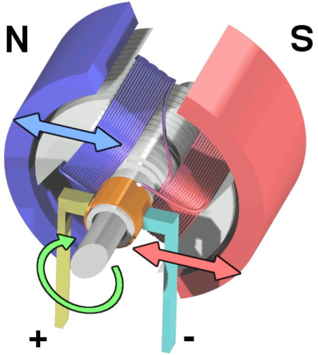
Summary

- Basic principles of robotics
- Description of links and joints

Overview

Brushed DC Motor

- cheap
- used in toys
- turns without motor controller
- commutation with brushes (may wear out)

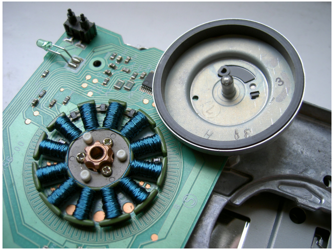


Video: Learn Engineering, *DC Motor, How it works?* Sep. 23, 2014

https://www.youtube.com/watch?v=LAtPHANEfQo

Brushless DC Motor

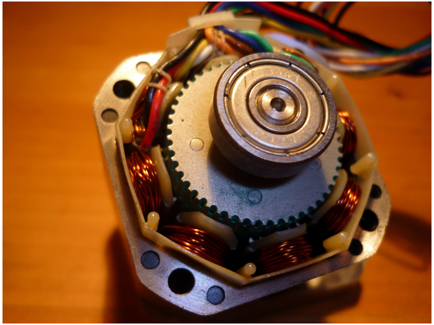
- cheap as well
- used e.g. in CD-ROM drives
- electronic commutation necessary (with sensors or sensorless)
- coils can be inside or outside (better cooling when outside)



Video: Learn Engineering, *Brushless DC Motor, How it works ?* Oct. 14, 2014 https://www.youtube.com/watch?v=bCEiOnuODac

Stepper Motor

- very standardized
- strong at low speeds
- used in printers
- moves repeatably
- feed-forward by nature
- “looses steps” when friction/inertia/... is too high ...
- ... or is moving at its resonance frequency



Video: Learn Engineering, *How does a Stepper Motor work ?* Oct. 19, 2016 https://www.youtube.com/watch?v=eyqwLiowZiU

Overview

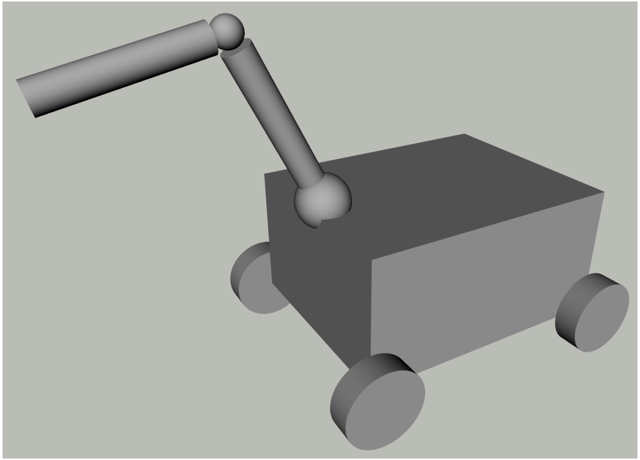
Robot Arm

Sequence of links connected by joints and moved relative to each other by actuators.

Robot Arm

```
<xacro:macro name="simple_arm" params="prefix parent *origin joint_limited:=true">

    <joint name="shoulder_pan_joint" type="continuous">
        <parent link="${parent}"/>
        <child link = "${prefix}base_link"/>
        <xacro:insert_block name="origin"/>
        <axis xyz="0 1 0"/>
        <limit lower="${shoulder_pan_lower_limit}" upper="${shoulder_pan_upper_limit}" effort="150.0" velocity="3.15
              "/>
        <dynamics damping="1.0" friction="100.0"/>
    </joint>

    <link name="${prefix}base_link">
        <visual>
            <origin xyz="0 0 0" rpy="0 0 ${base_correction}"/>
            <geometry>
                <sphere radius="${base_radius}"/>
            </geometry>
            <material name="LightGrey">
                <color rgba="0.7 0.7 0.7 1.0"/>
            </material>
        </visual>
        <collision>
            <origin xyz="0 0 0" rpy="0 0 ${base_correction}"/>
            <geometry>
                <sphere radius="${base_radius}"/>
            </geometry>
        </collision>
        <xacro:sphere_inertial radius="${base_radius}" mass="${base_mass}">
            <origin xyz="0.0 0.0 0.0" rpy="0 0 0"/>
        </xacro:sphere_inertial>
    </link>

[...]

</xacro:macro>
</robot>
```

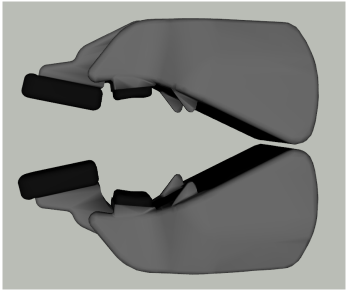
Robot Arm

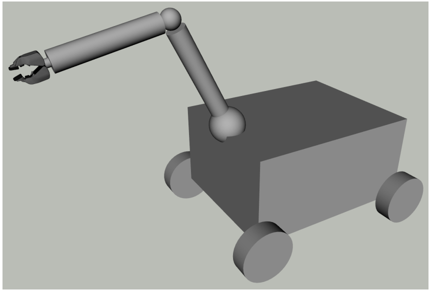


Overview

Gripper

Touch objects and interact with the environment.

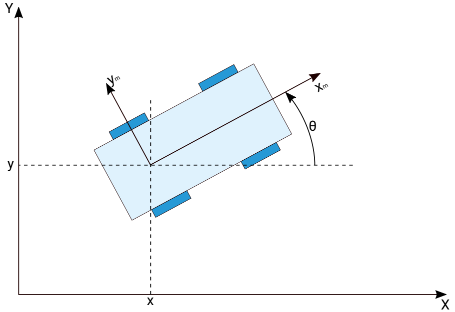




Overview

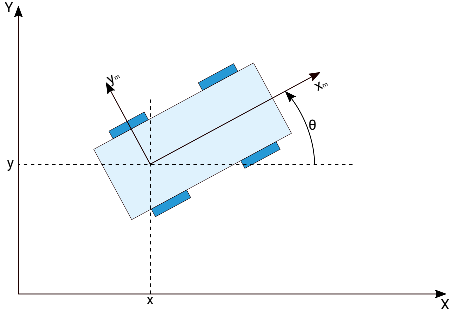
Wheeled Locomotion

Goal: Bring the robot to a desired pose (x, y, θ):
(position in **x**-axis, position in **y**-axis, **angle** with x-axis)
$\Rightarrow$ 3 Degrees of Freedom (DOF)



Wheeled Locomotion

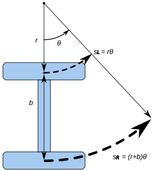
Goal: Bring the robot to a desired pose (x, y, θ):
(position in **x**-axis, position in **y**-axis, **angle** with x-axis)
$\Rightarrow$ 3 Degrees of Freedom (DOF)



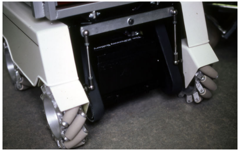
Wheeled motion typically has **non-holonomic constraints**

Wheel Types

Fixed wheels



Mecanum wheels



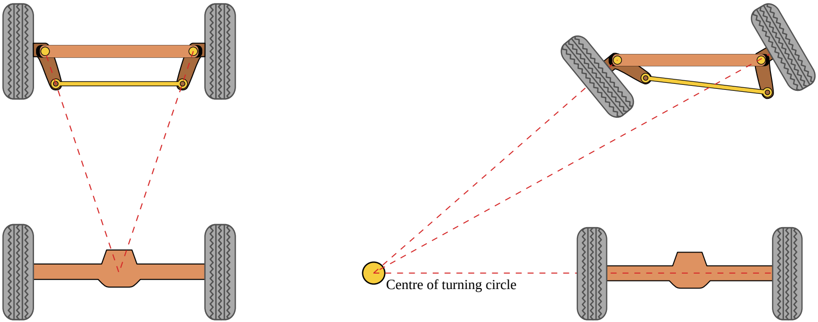
Omniwheels

Video

,

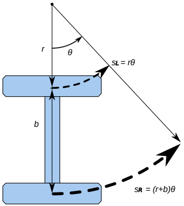
Ackerman steering

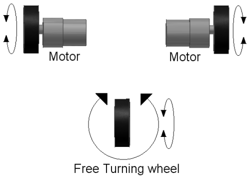
- Car-like steering
- \+ Robust
- \+ Outer wheels moves on a circle of different radius than inner wheel
- – But hard to control (parking!)



Differential-Drive

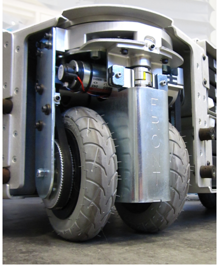
+ Turns on spot
+ Good choice for round robots
+ Parking is easier
– Cannot move sidewards

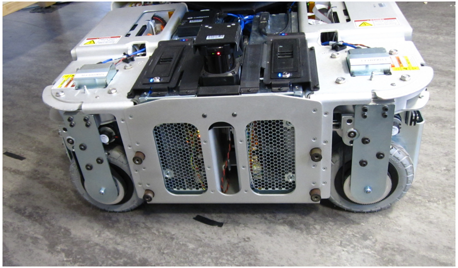




Turnable wheels

+ Omnidirectional (can drive forwards, sideways and turn)
– On change of direction, requires 'reconfiguration' of its wheels.
→ Controllers should not oscillate





PR2: Double wheel construction to reduce friction while turning the wheel

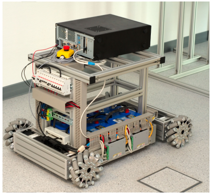
"Omniwheels"

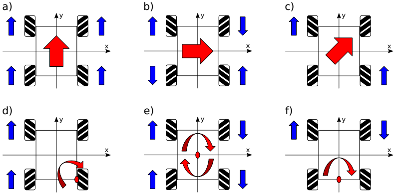
+ Omnidirectional (can drive forwards, sideways and turn)
- Wheels have free rollers at **90**^{o}
+ Three wheels are enough
– Hard to make them run smooth

Rotacaster
Source: https://en.wikipedia.org/w/index.php?title=Omni_wheel&oldid=1013193962

Mecanum-Wheels

+ Omnidirectional (can drive forwards, sideways and turn)
- Wheels have free rollers at **45**^{o}
+ No reconfiguration is involved
– Depending on wheels, requires flat ground





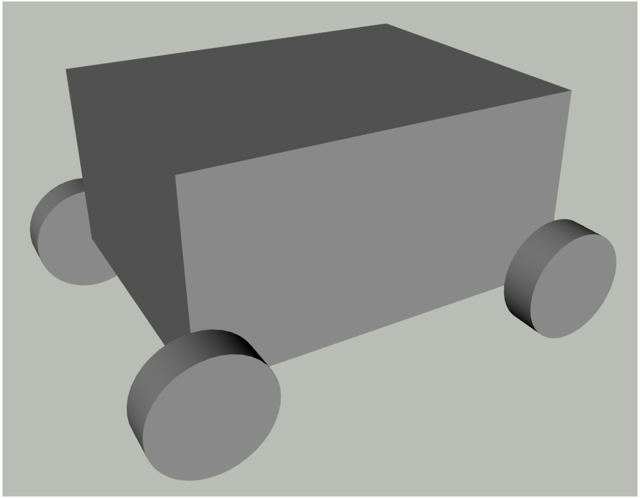
Linearity ⇒

A (linear) combination of cartesian movements can be achieved with the linear combination of the respective wheel velocities.

Mobile Base

```
<xacro:simple_robot_base name="base_link">
</xacro:simple_robot_base>

<xacro:omni_wheel name="omni_wheel_fl" parent="base_link">
    <origin xyz="${0.5*base_length} ${0.5*base_width + 0.5*wheel_thickness}
        ${-0.5*base_height}" rpy="${pi/2} 0 0"/>
</xacro:omni_wheel>

<xacro:omni_wheel name="omni_wheel_fr" parent="base_link">
    <origin xyz="${0.5*base_length} ${-0.5*base_width - 0.5*wheel_thickness}
        ${-0.5*base_height}" rpy="${pi/2} 0 0"/>
</xacro:omni_wheel>

<xacro:omni_wheel name="omni_wheel_bl" parent="base_link">
    <origin xyz="${-0.5*base_length} ${0.5*base_width + 0.5*wheel_thickness}
        ${-0.5*base_height}" rpy="${pi/2} 0 0"/>
</xacro:omni_wheel>

<xacro:omni_wheel name="omni_wheel_br" parent="base_link">
    <origin xyz="${-0.5*base_length} ${-0.5*base_width - 0.5*wheel_thickness}
```

Mobile Base



Summary

- Actuation of robot arms
- Wheeled locomotion

Overview

Kinematics

- Joint Space
- Task Space (Workspace)
- Forward Kinematics
- Inverse Kinematics

Configuration Space

Configuration

The **configuration** of a robot is a complete specification of the positions of every point of the robot.

“How can we represent the configuration?”

Configuration Space

Degrees of Freedom

The number of **degrees of freedom** is the minimum number of independent parameters needed to represent the configuration of the robot.

It is equivalently called the **mobility** of the robot.

Configuration Space

Degrees of Freedom

The number of **degrees of freedom** is the minimum number of independent parameters needed to represent the configuration of the robot.

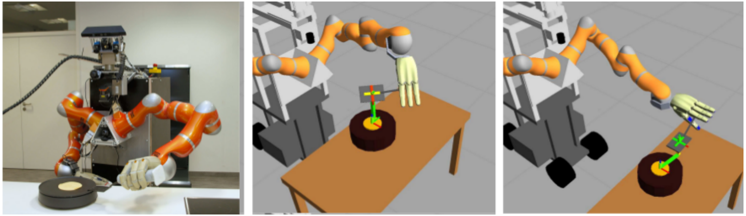
It is equivalently called the **mobility** of the robot.

Configuration Space

If the robot has n degrees of freedom, the n-dimensional space containing all possible configurations of the robot is called the **configuration space**.

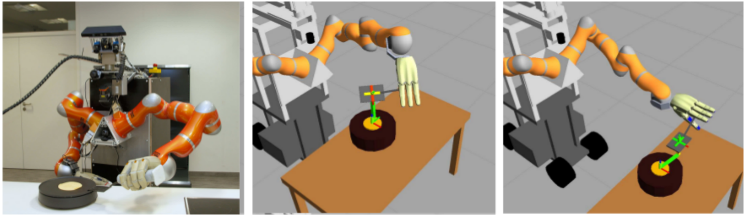
The robot's configuration is usually expressed in terms of joint variables, so the configuration space is also called the **joint space**.

Configuration Space vs. Task Space



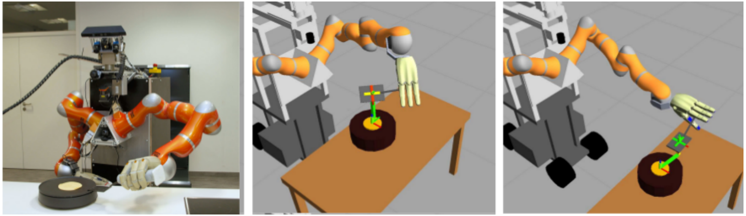
How can we get the *configuration/joint* trajectory to do this?

Configuration Space vs. Task Space



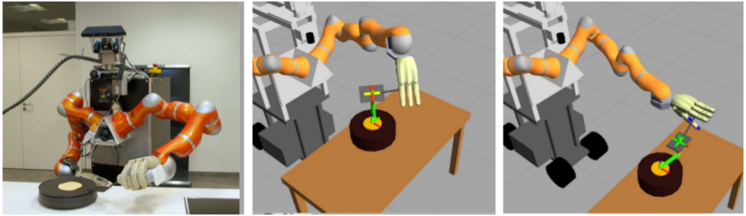
- The robot interacts with **objects** and **obstacles** which are better defined by an external set of coordinates (*e.g.,* (**x**, **y**, **z**) coordinates in a **global frame.**)

Configuration Space vs. Task Space



- The robot interacts with **objects** and **obstacles** which are better defined by an external set of coordinates (*e.g.,* (**x**, **y**, **z**) coordinates in a **global frame**.)
- The part interacting with environments such as obstacles and objects is called the **end-effector**.

Configuration Space vs. Task Space



- The robot interacts with **objects** and **obstacles** which are better defined by their coordinates (*e.g.,* (**x**, **y**, **z**) Euclidean 3D coordinates in a **world frame**.)
- The part interacting with environments such as obstacles and objects is called the **end-effector**.
- It is useful to attach a coordinate frame to the **end-effector** and command the pose of the end-effector in the world/object frame.

Task Space aka Operational Space

Operational Space

Given a robot with a reference frame attached to its end-effector, the **operational space** is the set of all positions and/or orientations achievable by the end-effector frame.

Required degrees of freedom

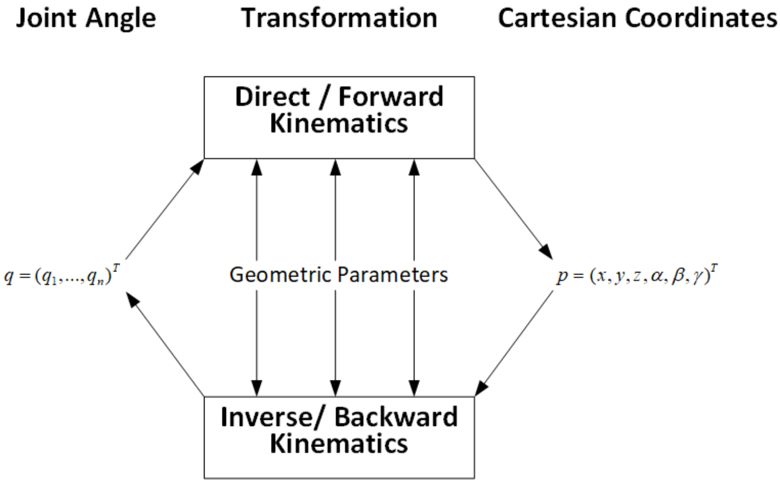
How many degrees of freedom do we need?

Required degrees of freedom

How many degrees of freedom do we need? (It depends on the task!)

- pointing a camera: 2 DOF
- placing an object (only position is important): 3 DOF
- placing an object (position and orientation): 6 DOF
- imitating a human arm: 7 DOF (from shoulder ball joint)

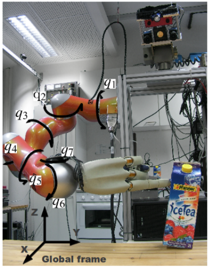
Robot kinematics



Forward Kinematics

Forward Kinematics Problem

Given an open-chain robot arm with a prescribed task frame, the goal is to determine the task frame's position and orientation as a function of the joint values.

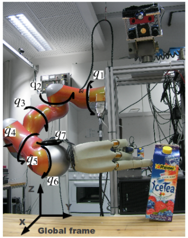


Given: $(q_1, q_2, \cdots, q_7)$

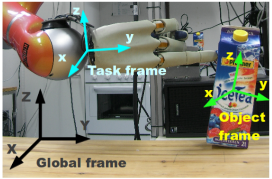
Forward Kinematics

Forward Kinematics Problem

Given an open-chain robot arm with a prescribed task frame, the goal is to determine the task frame's position and orientation as a function of the joint values.



Forward kinematics



Given: $(q_1, q_2, \cdots, q_7)$

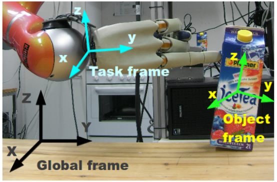
Calculate: $(x, y, z, \alpha, \beta, \gamma)$

For a given set of joint angles there will usually exist a **unique** end-effector position

Inverse Kinematics

Inverse Kinematics Problem

Given a desired position and orientation of the end-effector frame, one seeks to determine the set of joint angles that achieves this desired end-effector configuration.

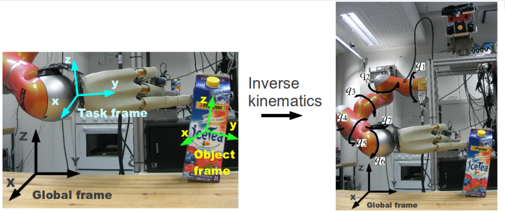


Given: $(x, y, z, \alpha, \beta, \gamma)$

Inverse Kinematics

Inverse Kinematics Problem

Given a desired position and orientation of the end-effector frame, one seeks to determine the set of joint angles that achieves this desired end-effector configuration.



Given: $(x, y, z, \alpha, \beta, \gamma)$ Calculate: $(q_1, q_2, \cdots, q_7)$

For a particular end-effector position and orientation, there may exist **multiple** solutions, or even **none** at all. (*e.g.*, *elbow-up* and *elbow-down* configurations)

Kinematics – An Example

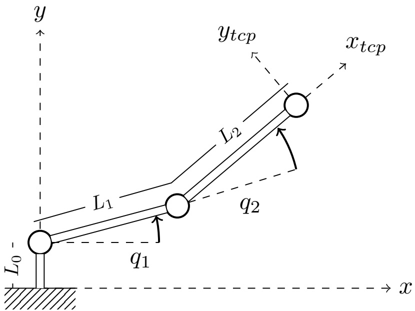


Figure: Two links planar manipulator

- Forward kinematics :

$$\begin{pmatrix} x_{tcp} \\ y_{tcp} \end{pmatrix} = \begin{pmatrix} L_1 \cos q_1 + L_2 \cos(q_1 + q_2) \\ L_1 \sin q_1 + L_2 \sin(q_1 + q_2) \end{pmatrix}$$

Kinematics – An Example

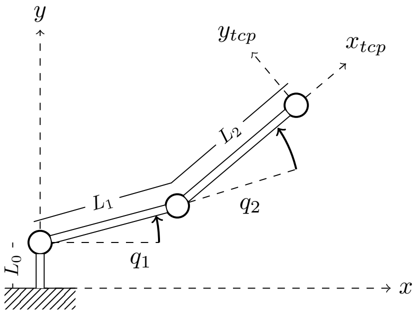


Figure: Two links planar manipulator

- Forward kinematics :

$$\begin{pmatrix} x_{tcp} \\ y_{tcp} \end{pmatrix} = \begin{pmatrix} L_1 \cos q_1 + L_2 \cos(q_1 + q_2) \\ L_1 \sin q_1 + L_2 \sin(q_1 + q_2) \end{pmatrix}$$

- Inverse kinematics :

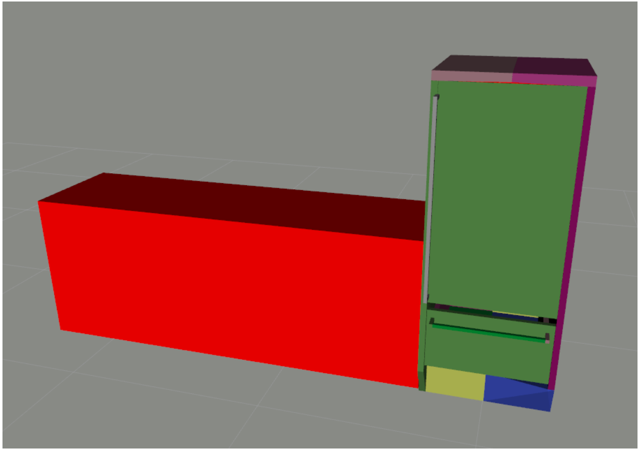
$$\alpha = \cos^{-1}\left(\frac{x_{tcp}^2 + y_{tcp}^2 + L_1^2 - L_2^2}{2L_1\sqrt{x_{tcp}^2 + y_{tcp}^2}}\right), \beta = \cos^{-1}\left(\frac{L_1^2 + L_2^2 - x_{tcp}^2 - y_{tcp}^2}{2L_1 L_2}\right)$$

$$q_1 = \tan^{-1}\frac{y_{tcp}}{x_{tcp}} - \alpha, q_2 = \pi - \beta$$

$$q_1 = \tan^{-1}\frac{y_{tcp}}{x_{tcp}} + \alpha, q_2 = \pi + \beta$$

Overview

The Representation Of The Environment



Summary

- Configuration and task space
- Robot kinematics

More in the URDF tutorial

`https://binder.intel4coro.de/v2/gh/IntEL4CoRo/jupyter-ros2/HEAD?labpath=tutorials%2FURDF.ipynb`

Overview

Assignment and dates

- Assignment 3:
  `https://github.com/artnie/rpwr-assignments`
- Grades: 15 points for this assignment on ROS
- Due: 20.05., 23:59 German time
- Tutorium: today after lunch 14:15
- Next class: 14.05., 12:15

Thanks for your attention!

Special thanks to the IAI team for the content of this lecture!