Introduction to Robotics

Lecture 3

Jianwei Zhang, Lasse Einig
[zhang, einig]@informatik.uni-hamburg.de

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

April 22, 2016

Outline

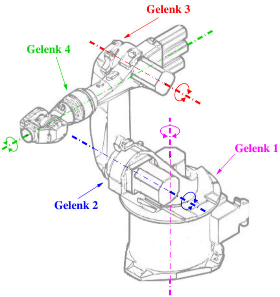
Recapitulation of DH-Parameter

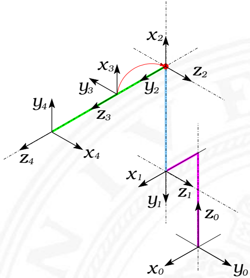
- universal minimal robot description
- based on frame transformations
- four parameters per frame transformation
- serial chain of transformations
- unique description of T_6

Drawbacks

- ambiguous convention
- only kinematic chain described
- missing information on geometry, physical constraints, dynamics, collisions, inertia, sensors, ...

Definition of joint coordinate systems





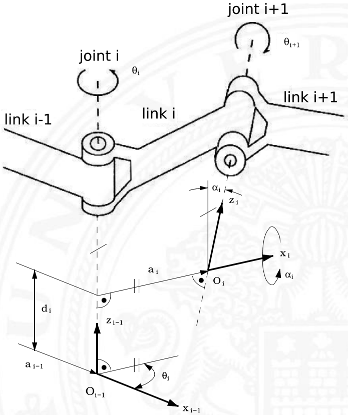
- CS_0 is the stationary origin at the base of the manipulator
- axis z_i is set along the axis of motion of the i^th link
- axis x_i is the common normal of $z_{i-1} \times z_i$
- axis y_i concludes a right-handed coordinate system

Parameters for description of two arbitrary links

Two parameters for the description of the link structure i

- a_i : shortest distance between the z_{i-1}-axis and the z_i-axis
- α_i : rotation angle around the x_i-axis, which aligns the z_{i-1}-axis to the z_i-axis

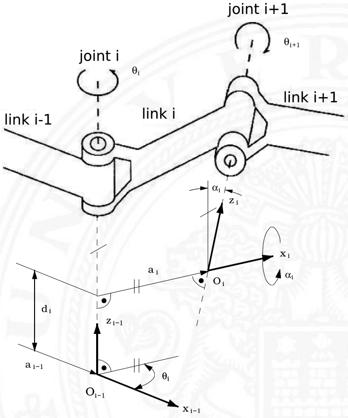
a_i and α_i are constant values due to construction



Parameters for description of two arbitrary links (cont.)

Two for relative distance and angle of adjacent links

- d_i: distance origin O_{i-1} of the $(i\text{-}1)^{st}$ CS to intersection of z_{i-1}-axis with x_i-axis
- θ_i: joint angle around z_{i-1}-axis to align x_{i-1}-parallel to x_i-axis into x_{i-1}, y_{i-1}-plane



θ_i and d_i are variable

- rotational: θ_i variable, d_i fixed
- translational: d_i variable, θ_i fixed

Universal Robot Description Format

Documentation

http://wiki.ros.org/urdf

http://wiki.ros.org/urdf/xml

- robot description format used in ROS[2]
- hierarchical description of components
- XML format representing robot model
  - kinematics and dynamics
  - visual
  - collision
  - configuration

[2] http://ros.org

URDF: Structure

links geometrical properties
- visual
- inertial
- collision

joints geometrical connections
- geometry
- structure
- config

sensors attached sensors

transmissions transmission properties

gazebo simulation properties

model_state robot state

URDF: XML Tree Structure

- Filename: robotname.urdf
- XML prolog:

```
<?xml version="1.0" encoding="utf-8"?>
```

- XML element types

```
<tag attribute="value"/>
```

```
<tag attribute="value">
  text or element(s)
</tag>
```

- XML comments

```
<!-- Comments are placed within these tags -->
```

URDF: XML Tree Structure (cont.)

- 1^{st}-level structure

```
<robot name="samplerobot">
</robot>
```

- 2^{nd}-level structure

  link, joints, sensors, transmissions, gazebo, model_state
- 3^{rd}-level structure

  visual, inertia, collision, origin, parent, ...
- 4^{th}-level structure

  ⋮

URDF: Link

```
<link name="sample_link">
  <!-- describes the mass and inertial properties of
       the link -->
  <inertial/>

  <!-- describes the visual appearance of the link.
       can be describe using geometric primitives or
       meshes -->
  <visual>

  <!-- describes the collision space of the link.
       is described like the visual appearance -->
  <collision/>
</link>
```

URDF: Link – visual – primitives

Geometric primitives for describing visual appearance of the link

```
<link name="base_link">
  <visual>
    <origin xyz="0 0 0.01" rpy="0 0 0"/>
    <geometry>
      <box size="0.2 0.2 0.02"/>
    </geometry>
    <material name="cyan">
      <color rgba="0 1.0 1.0 1.0"/>
    </material>
  </visual>
</link>
```

- Geometric primitives: <box>, <cylinder>, <sphere>
- Materials: <color>, <texture>

URDF: Link – visual – meshes

3D meshes for describing visual appearance of the link

```
<link name="base_link">
  <visual>
    <origin xyz="0 0 0.01" rpy="0 0 0"/>
    <geometry>
      <mesh filename="meshes/base_link.dae"
    </geometry>
  </visual>
</link>
```

- the <collision> element is described identically to the <visual> element
- an additional <collision_checking> primitive can be used to approximate

URDF: Link – inertial

Parameters describing the physical properties of the link

```
<link name="base_link">
  <inertial>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <mass value="1">
    <inertia ixx="100" ixy="0" ixz="0"
             iyy="100" iyz="0" izz="100" />
  </inertial>
</link>
```

- center of gravity <origin xyz>
- object mass <mass value>
- inertia tensor <intertia>

URDF: Inertia

Inertial tensor describes the dynamic physical properties of the link

- orientation and position of the inertia CS described by `<origin>` tag
- tensor is a symmetric 3×3 matrix
- diagonal values describe main inertial axes `ixx`, `iyy`, `izz`
- `ixy`, `ixz`, `iyz` are 0 for geometric primitives
- rotations around largest and smallest inertial axis are most stable

URDF: Joint

```
<joint name="base_link_to_cyl" type="revolute">
  <!-- describes joint position and orientation -->
  <origin xyz="0 0 0.07" rpy="0 0 0"/>

  <!-- describes the related links -->
  <parent link="base_link"/>
  <child link="base_cyl"/>

  <!-- describes the axis of rotation-->
  <axis xyz="0 0 1"/>

  <!-- describes the joint limits-->
  <limit velocity="1.5707963267"
         lower="-3.1415926535" upper="3.1415926535"/>
</joint>
```

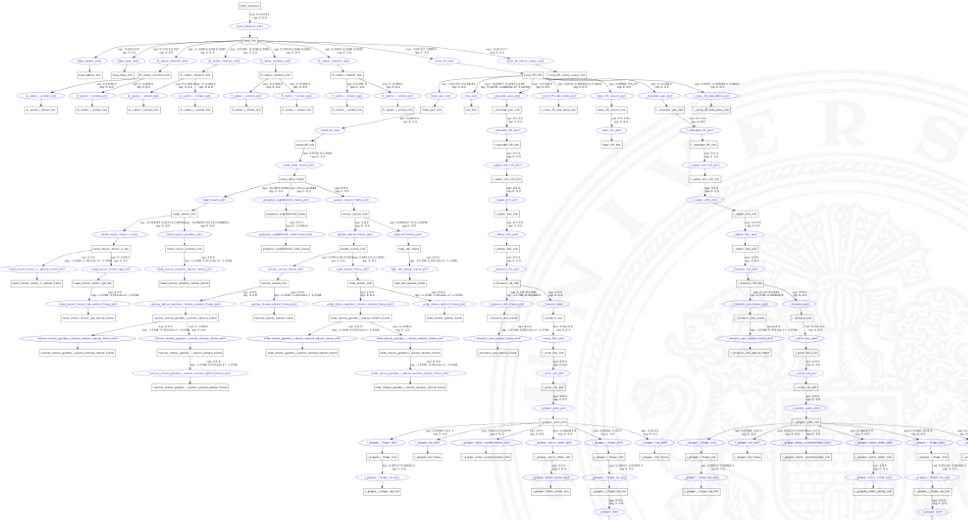
URDF: Joint (cont.)

type	revolute, continuous, prismatic, fixed, floating, planar
parent_link	link which the joint is connected to
child_link	link which is connected to the joint
axis	joint axis relative to the joint CS. Represented using a normalized vector
limit	joint limits for motion (lower, upper), velocity and effort
dynamics	damping, friction
calibration	rising, falling
mimic	joint, multiplier, offset
safety_controller	soft_lower_limit, soft_upper_limit, k_position, k_velocity

URDF: Other elements

- sensor
  - position and orientation relative to link
  - sensor properties
    - update rate
    - resolution
    - minimum / maximum angle
- transmissions
  - relation of motor to joint motion
- gazebo
  - simulation properties
- model state
  - description of different robot configurations

URDF: Hierarchy



Outline

Outline (cont.)

Inverse kinematics for manipulators

Set of problems

- In the majority of cases the control of robot manipulators takes place in the *joint space*,
- The informations about objects are mostly given in the *cartesian space*.

For getting a specific tool frame T related to the world, joint values $\theta(t) = (\theta_1(t), \theta_2(t), ..., \theta_n(t))^T$ should be calculated in two steps:

1. Calculation of $T_6 = Z^{-1}BGE^{-1}$;
2. Calculation of $\theta_1, \theta_2, ..., \theta_n$ via T_6.

$\Longrightarrow$ In this case the inverse kinematics is more important than the forward kinematics.

The solution using the example of PUMA 560

$$T_6 = T'T'' = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where

$$n_x = C_1[C_{23}(C_4C_5C_6 - S_4S_6) - S_{23}S_5C_6] - S_1(S_4C_5C_6 + C_4S_6) \quad (13)$$

$$n_y = S_1[C_{23}(C_4C_5C_6 - S_4S_6 - S_{23}S_5S_6] + C_1(S_4C_5C_6 + C_4S_6) \quad (14)$$

$$n_z = -S_{23}[C_4C_5C_6 - S_4S_6] - C_{23}S_5C_6 \quad (15)$$

$$(16)$$

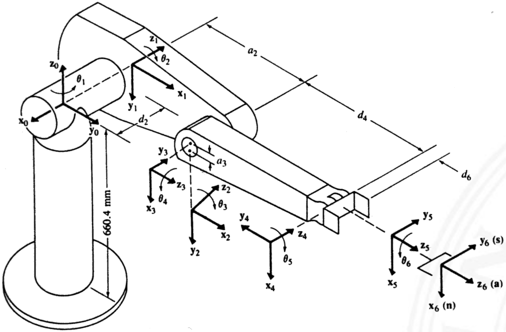
The solution using the example of PUMA 560 (cont.)

$$o_x = \ldots \tag{17}$$
$$o_y = \ldots \tag{18}$$
$$o_z = \ldots \tag{19}$$
$$a_x = \ldots \tag{20}$$
$$a_y = \ldots \tag{21}$$
$$a_z = \ldots \tag{22}$$
$$p_x = C_1[d_6(C_{23}C_4S_5 + S_{23}C_5) + S_{23}d_4 + a_3C_{23} + a_2C_2] - S_1(d_6S_4S_5 + d_2) \tag{23}$$
$$p_y = S_1[d_6(C_{23}C_4S_5 + S_{23}C_5) + S_{23}d_4 + s_3C_{23} + a_2C_2] + C_1(d_6S_4S_5 + d_2) \tag{24}$$
$$p_z = d_6(C_{23}C_5 - S_{23}C_4S_5) + C_{23}d_4 - a_3S_{23} - a_2S_2 \tag{25}$$

Remark

- Non-linear equations
- Existence of solutions:
  Workspace: the volume of space that is reachable for the tool of the manipulator.
  - "dexterous workspace"
  - "reachable workspace"
- Many joint positions that produce a similar TCP position using the example of PUMA 560:
  - Ambiguity of solutions for $\theta_1, \theta_2, \theta_3$ related to given $\mathbf{p}$.
  - For each solution of $\theta_4, \theta_5, \theta_6$ the alternative solution exists:

Remark (cont.)



$$\theta_4' = \theta_4 + 180^\circ$$
$$\theta_5' = -\theta_5$$
$$\theta_6' = \theta_6 + 180^\circ$$

- Different solution strategy: closed solutions vs. numerical solutions

Different methods for solution finding

Closed form (analytical):

- `algebraic solution`
  - \+ accurate solution by means of equations
  - − solution is not geometrically representative
- `geometrical solution`
  - \+ case-by-case analysis of possible robot configurations
  - − robot specific

Numerical form:

- `iterative methods`
  - \+ the methods are transferable
  - − computationally intensive, for several exceptions the convergence can not be guaranteed

Methods for solution finding

Solvability

"The inverse kinematics for all systems with 6 DOF (translational or rotational joints) in a simple serial chain is always numerical solvable."

Analytical solvability of manipulator

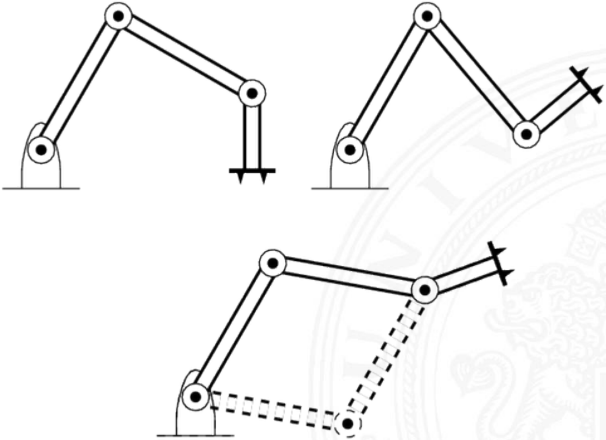
The closed solution exists if specific constraints (sufficient constraints) for the arm geometry are satisfied:

`If` 3 sequent axis intersect in a given point

`or if` 3 sequent axis are parallel to each other

- manipulators should be designed regarding these constraints
- most of them are
  - PUMA 560: axes 4, 5 & 6 intersect in a single point
  - Mitsubishi PA10, KUKA LWR, PR2
  - 3-DOF planar (RPC)

Example 1: a planar 3 DOF manipulator



Example 1: a planar 3 DOF manipulator (cont.)

Joint	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	0	l_1	0	θ_2
3	0	l_2	0	θ_3

$$
{}^0T_3 = \begin{bmatrix} C_{123} & -S_{123} & 0 & l_1 C_1 + l_2 C_{12} \\ S_{123} & C_{123} & 0 & l_1 S_1 + l_2 S_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

The algebraical solution for the example 1

Specification for the TCP: (x, y, ϕ). For such kind of vectors applies:

$$^0T_3 = \begin{bmatrix} C_\phi & -S_\phi & 0 & x \\ S_\phi & C_\phi & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Resultant four equations can be derived:

$$C_\phi = C_{123} \tag{26}$$

$$S_\phi = S_{123} \tag{27}$$

$$x = l_1 C_1 + l_2 C_{12} \tag{28}$$

$$y = l_1 S_1 + l_2 S_{12} \tag{29}$$

The algebraical solution for the example 1 (cont.)

The function *atan*2 is defined as:

$$\theta = atan2(y,x) = \begin{cases} 0 & \text{for} \quad x = 0, y = 0 \\ \pi/2 & \text{for} \quad x = 0, y > 0 \\ 3 * \pi/2 & \text{for} \quad x = 0, y < 0 \\ atan(y,x) & \text{for +x and +y} \\ 2\pi - atan(y,x) & \text{for +x und -y} \\ \pi - atan(y,x) & \text{for -x und +y} \\ \pi + atan(y,x) & \text{for -x und -y} \end{cases}$$

The algebraical solution for the example 1 (cont.)

Square and add (28) and (29)

$$x^2 + y^2 = l_1^1 + l_2^2 + 2 l_1 l_2 C_2$$

using

$$C_{12} = C_1 C_2 - S_1 S_2, S_{12} = C_1 S_2 + S_1 C_2$$

giving

$$C_2 = \frac{x^2 + y^2 - l_1^2 - l_2^2}{2 l_1 l_2}$$

for goal in workspace

$$S_2 = \pm\sqrt{1 - C_2^2}$$

solution

$$\theta_2 = atan2(S_2, C_2)$$

The algebraical solution for the example 1 (cont.)

solve (28) and (29) for θ_1

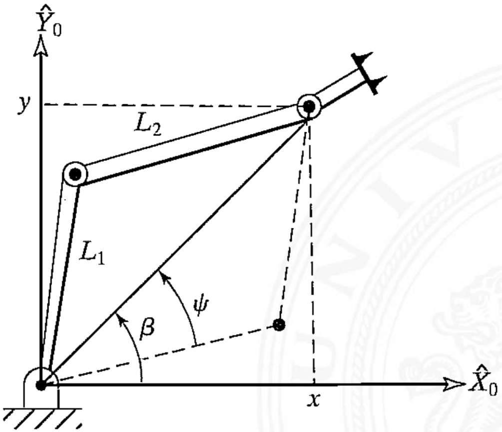
$$\theta_1 = atan2(y, x) - atan2(k_2, k_1)$$

where $k_1 = l_1 + l_2 C_2$ and $k_2 = l_2 S_2$.

solve θ_3 from (26) and (27)

$$\theta_1 + \theta_2 + \theta_3 = atan2(S_\phi, C_\phi) = \phi$$

The geometrical solution for the example 1



The geometrical solution for the example 1 (cont.)

Calculate θ_2 via the "law of cosines":

$$x^2 + y^2 = l_1^2 + l_2^2 - 2 l_1 l_2 \cos(180 + \theta_2)$$

The solution:

$$\theta_2 = \pm cos^{-1} \frac{x^2 + y^2 - l_1^2 - l_2^2}{2 l_1 l_2}$$

$$\theta_1 = \beta \pm \psi$$

where:

$$\beta = atan2(y, x), \quad \cos\psi = \frac{x^2 + y^2 - l_1^2 - l_2^2}{2 l_1 \sqrt{x^2 + y^2}}$$

For $\theta_1, \theta_2, \theta_3$ applies:

$$\theta_1 + \theta_2 + \theta_3 = \phi$$

Algebraical solution (with polynomial conversion)

The following substitutions are used for the polynomial conversion of transcendental equations:

$$u = tan\frac{\theta}{2}$$

$$\cos\theta = \frac{1-u^2}{1+u^2}$$

$$\sin\theta = \frac{2u}{1+u^2}$$

Algebraical solution (with polynomial conversion) (cont.)

Example:
The following transcendental equation is given:

$$a \cos\theta + b \sin\theta = c$$

After the polynomial conversion:

$$a(1 - u^2) + 2bu = c(1 + u^2)$$

The solution for u:

$$u = \frac{b \pm \sqrt{b^2 - a^2 - c^2}}{a + c}$$

Then:

$$\theta = 2tan^{-1}\left(\frac{b \pm \sqrt{b^2 - a^2 - c^2}}{a + c}\right)$$

Algebraic solution using the PUMA 560

Calculation of $\theta_1, \theta_2, \theta_3$:
The first three joint angles $\theta_1, \theta_2, \theta_3$ affect the position of the TCP $(p_x, p_y, p_z)^T$ (in case $d_6 = 0$).

$$p_x = C_1[S_{23}d_4 + a_3C_{23} + a_2C_2] - S_1d_2 \quad (30)$$
$$p_y = S_1[S_{23}d_4 + a_3C_{23} + a_2C_2] + C_1d_2 \quad (31)$$
$$p_z = C_{23}d_4 - a_3S_{23} - a_2S_2 \quad (32)$$

The outcome of this is:

$$\theta_1 = tan^{-1}\left(\frac{\mp p_y\sqrt{p_x^2 + p_y^2 - d_2^2} - p_xd_2}{\mp p_x\sqrt{p_x^2 + p_y^2 - d_2^2} + p_yd_2}\right)$$

Algebraic solution using the PUMA 560 (cont.)

$$\theta_3 = tan^{-1}\left(\frac{\mp A_3\sqrt{A_3^2 + B_3^2 - D_3^2} + B_3 D_3}{\mp B_3\sqrt{A_3^2 + B_3^2 - D_3^2} + A_3 D_3}\right)$$

where

$$\begin{aligned} A_3 &= 2a_2 a_3 \\ B_3 &= 2a_2 d_4 \\ D_3 &= p_x^2 + p_y^2 + p_z^2 - a_2^2 - a_3^2 - d_2^2 - d_4^2 \end{aligned}$$

Algebraic solution using the PUMA 560 (cont.)

and

$$\theta_2 = tan^{-1}\left(\frac{\mp B_2\sqrt{p_x^2 + p_y^2 - d_2^2} + A_2 p_z}{\mp A_2\sqrt{p_x^2 + p_y^2 - d_2^2} + B_2 p_z}\right)$$

where

$$A_2 = d_4 C_3 - a_3 S_3$$
$$B_2 = -a_3 C_3 - d_4 S_3 - a_2$$

The solution for RPY angles

$$T = R_{z,\phi} R_{y,\theta} R_{x,\psi}$$

The solution for following equation is sought:

$$R_{z,\phi}^{-1} T = R_{y,\theta} R_{x,\psi}$$

$$\begin{bmatrix} f_{11}(\mathbf{n}) & f_{11}(\mathbf{o}) & f_{11}(\mathbf{a}) & 0 \\ f_{12}(\mathbf{n}) & f_{12}(\mathbf{o}) & f_{12}(\mathbf{a}) & 0 \\ f_{13}(\mathbf{n}) & f_{13}(\mathbf{o}) & f_{13}(\mathbf{a}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C\theta & S\theta S\psi & S\theta C\psi & 0 \\ 0 & C\psi & -S\psi & 0 \\ -S\theta & C\theta S\psi & C\theta C\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where

$$f_{11} = C\phi x + S\phi y$$
$$f_{12} = -S\psi x + C\phi y$$
$$f_{13} = z$$

The solution for RPY angles (cont.)

The equation for $f_{12}(\mathbf{n})$ leads to:

$$-S\phi n_x + C\phi n_y = 0$$

$\Longrightarrow$

$$\phi = atan2(n_y, n_x)$$

and

$$\phi = \phi + 180^\circ$$

The solution with the elements f_{13} and f_{11} are as appropriate:

$$-S\theta = n_z$$

The solution for RPY angles (cont.)

and

$$C\theta = C\phi n_x + S\phi n_y$$

$\Longrightarrow$

$$\theta = atan2(-n_z, C\phi n_x + S\phi a_y)$$

The solution with the elements f_{23} and f_{22} are as appropriate:

$$-S\psi = -S\phi a_x + C\phi a_y$$

$$C\psi = -S\phi o_x + C\phi o_y$$

$\Longrightarrow$

$$\psi = atan2(S\phi a_x - C\phi a_y, -S\phi o_x + C\phi o_y)$$

Solution for arm configurations

Definition of different arm configurations

- **shoulder** RIGHT-arm, LEFT-arm
- **elbow** ABOVE-arm, BELOW-arm
- **wrist** WRIST-down, WRIST-up

Solution for arm configurations (cont.)

Adapted from this following variable can be defined:

$$ARM = \begin{cases} +1 & \text{RIGHT-arm} \\ -1 & \text{LEFT-arm} \end{cases}$$

$$ELBOW = \begin{cases} +1 & \text{ABOVE-arm} \\ 1 & \text{BELOW-arm} \end{cases}$$

$$WRIST = \begin{cases} +1 & \text{WRIST-down} \\ -1 & \text{WRIST-up} \end{cases}$$

The complete solution for the inverse kinematics can be achieved by analysis of such arm configurations.

Technical difficulties for control software

Problem

- Software was hard-coded for a certain robot model / type.
- Software specialized on the robot skills and geometry
- Consequently, the extending and porting software to new hardware was difficult and time consuming

Solution

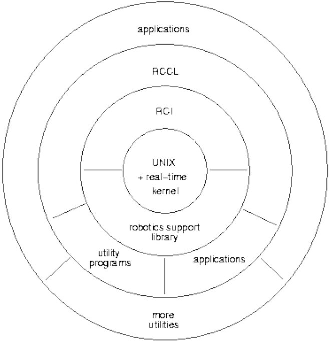
Develop a control software with the following capabilities

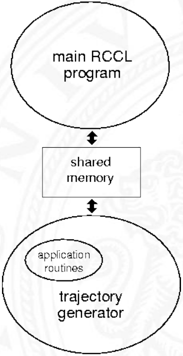
- Possibility to control low-level hardware properties
- Maximum portability to different platforms
- Maximum flexibility for fast programming of applications

A Framework for robots under UNIX: RCCL

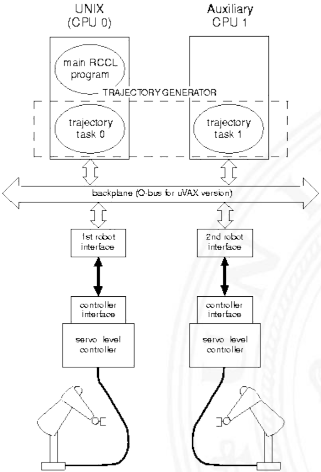
RCCL

Robot Control C Library

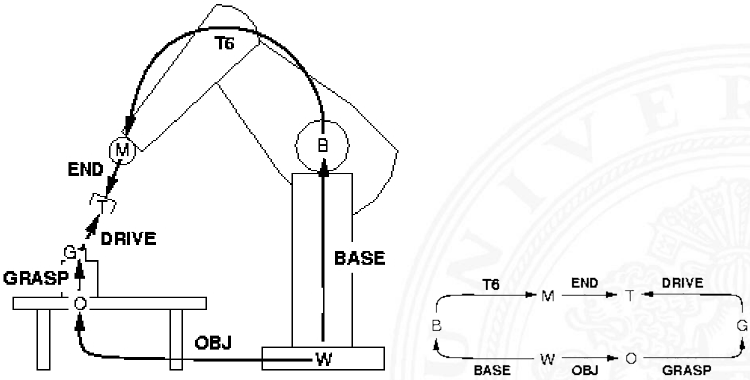




Ability to control multiple robots



Motion description with position equations



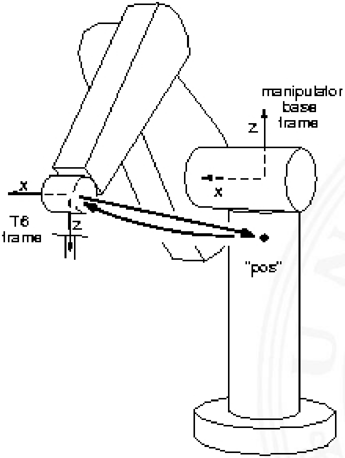
Code sample for robot control in RCCL

```
#include <rccl.h>
#include "manex.560.h"

main()
{
        TRSF_PTR p, t;                                                    /*#1*/
        POS_PTR pos;                                                      /*#2*/
        MANIP *mnp;                                                       /*#3*/
        JNTS rcclpark;                                                    /*#4*/
        char *robotName;                                                  /*#5*/

        rcclSetOptions (RCCL_ERROR_EXIT);                                 /*#6*/
        robotName = getDefaultRobot();                                    /*#7*/
        if (!getRobotPosition (rcclpark.v, "rcclpark", robotName))
         { printf (''position 'rcclpark' not defined for robot\n'');
           exit(-1);
         }
                                                                          /*#8*/
        t = allocTransXyz ("T", UNDEF, -300.0, 0.0, 75.0);
        p = allocTransRot ("P", UNDEF, P_X, P_Y, P_Z, xunit, 180.0);
        pos = makePosition ("pos", T6, EQ, p, t, NULL);                   /*#9*/
```

Code sample for robot control in RCCL (cont.)

```
        mnp = rcclCreate (robotName, 0);                     /*#10*/
        rcclStart();

        movej (mnp, &rcclpark);                              /*#11*/

        setMod (mnp, 'c');                                   /*#12*/
        move (mnp, pos);                                     /*#13*/
        stop (mnp, 1000.0);

        movej (mnp, &rcclpark);                              /*#14*/
        stop (mnp, 1000.0);

        waitForCompleted (mnp);                              /*#15*/
        rcclRelease (YES);                                   /*#16*/
}
```

Code sample for robot control in RCCL (cont.)



Evolution of Robot Frameworks

Robot Operating System (ROS)

↑

Player/Stage Framework

↑

RCCL

↑

Hard-coded robot software