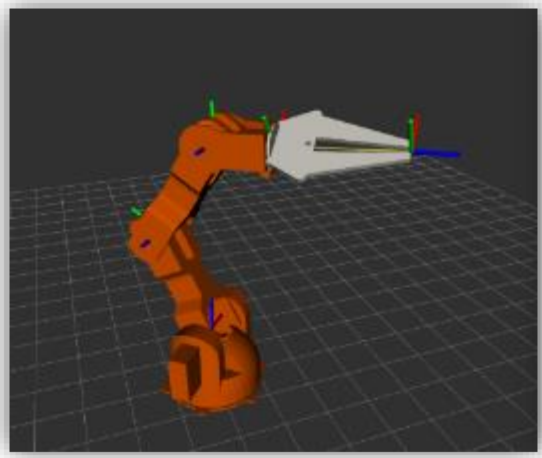
Forward and Inverse Kinematics

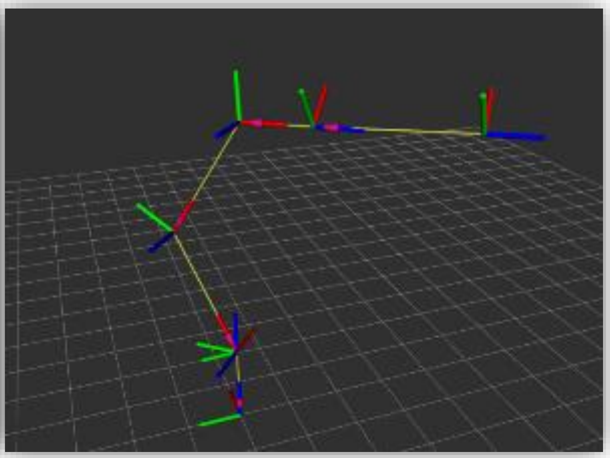
Dr Gerardo Aragon-Camarasa
gerardo.aragoncamarasa@glasgow.ac.uk

RF – University of Glasgow

Kinematics of Robot Manipulators

- It is the study of geometric and time properties of the **motion of robotic structures**, without reference to the causes producing it
  - It doesn't matter how the robot moves!

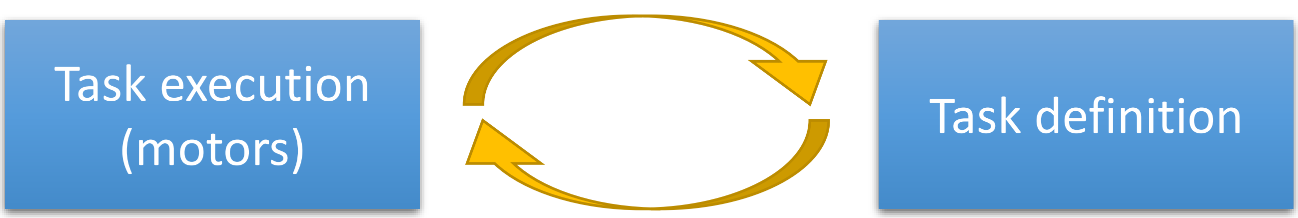




- A robot is seen as:
  - **"open kinematic chain of rigid bodies interconnected by (revolute or prismatic) joints"**

Why?

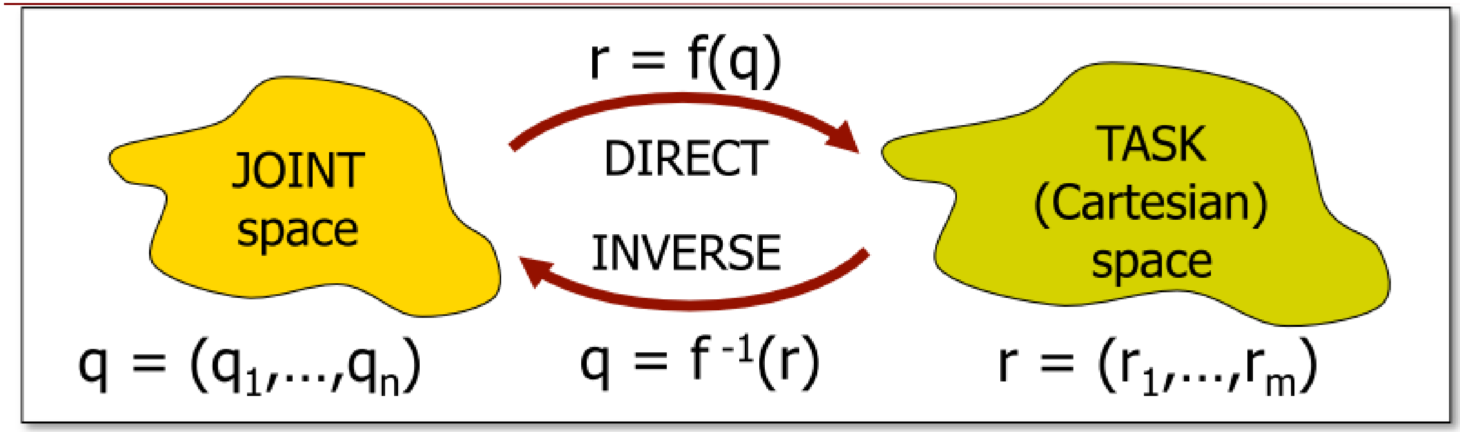
- Functional aspects:
  - Definition of a robot workspace
  - Calibration (virtual vs real)
- Operational aspects:
  - Links 2 different spaces related by kinematic and dynamic maps
  - Motion/Trajectory planning
  - Programming
  - Motion control



Why?

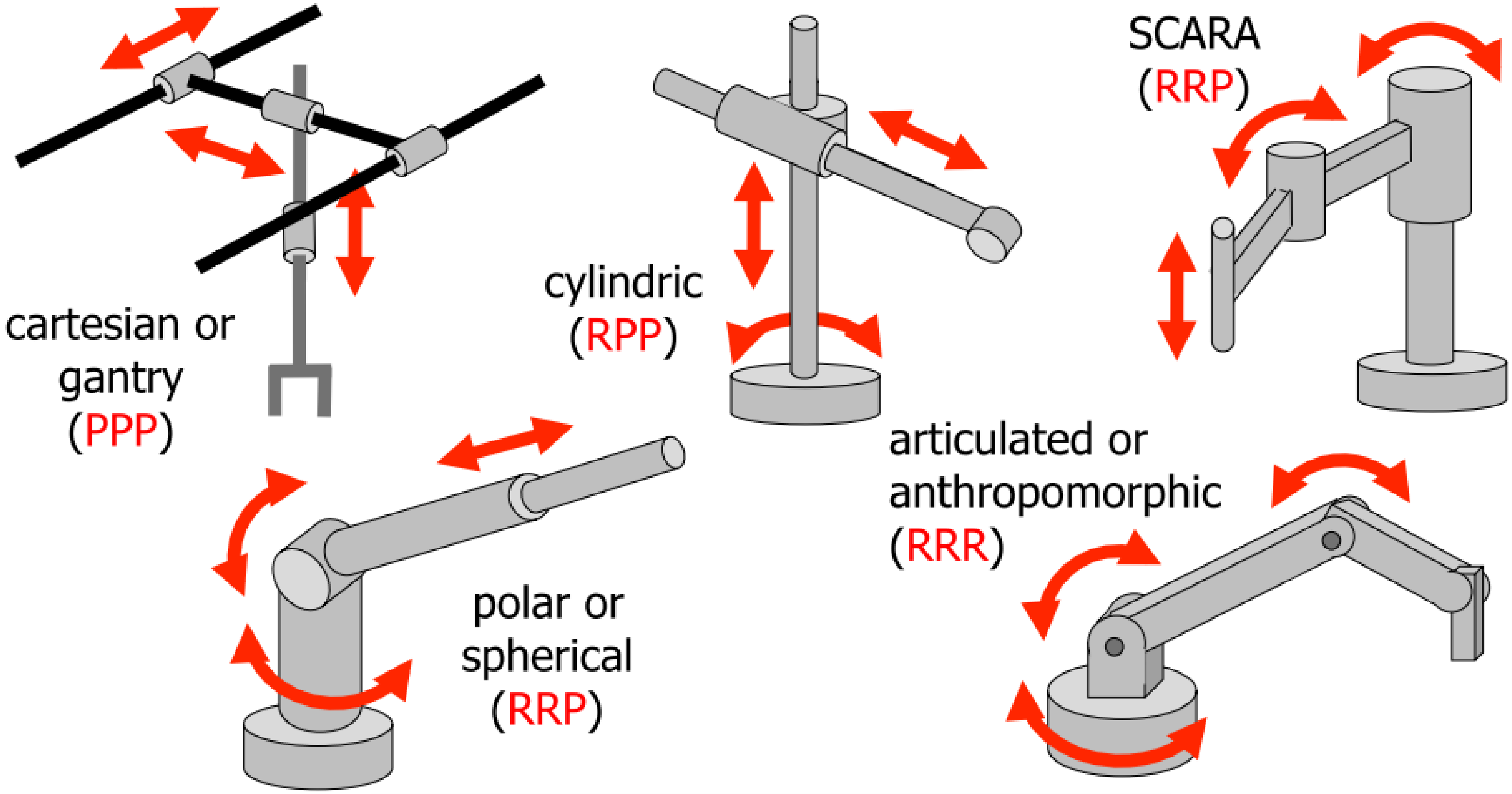
Why?

Formulation and Parameters



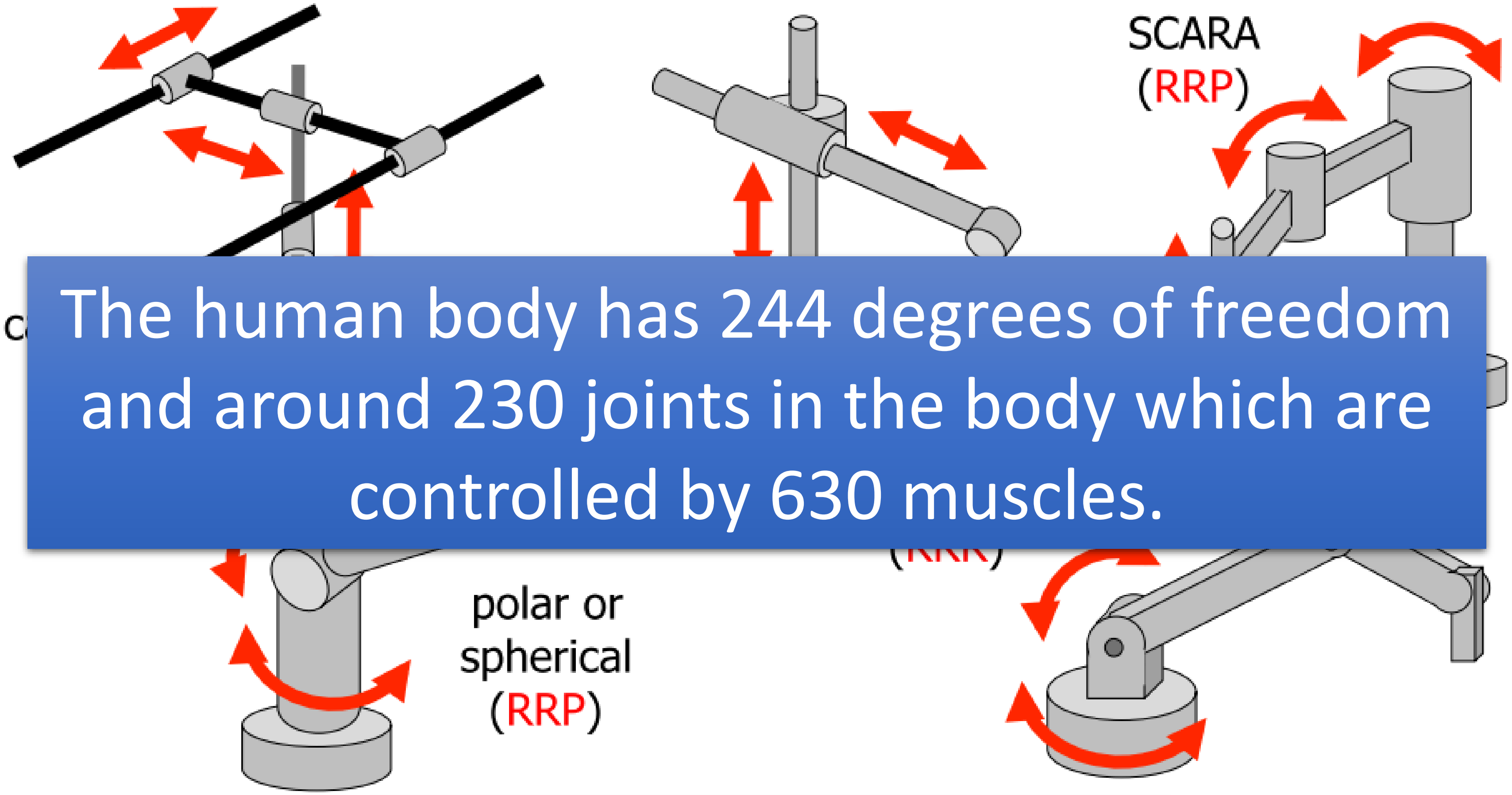
- Parameterisation of q
  - **Unambiguous** and **minimal** characterisation of the robot configuration
  - n = # degrees of freedom (DoF) = # robot joints (rotation and/or translation)
- Choice of parameterisation r
  - Compact description of **pose** to the required task
  - $m \leq 6$, and usually $m \leq n$ (but this is not needed)

Kinematic types – Classification



P = 1-DoF translation (prismatic) joint
R = 1-DoF rotation (revolute) joint

Kinematic types – Classification



The human body has 244 degrees of freedom and around 230 joints in the body which are controlled by 630 muscles.

P = 1-DoF translation (prismatic) joint
R = 1-DoF rotation (revolute) joint

Forward Kinematics

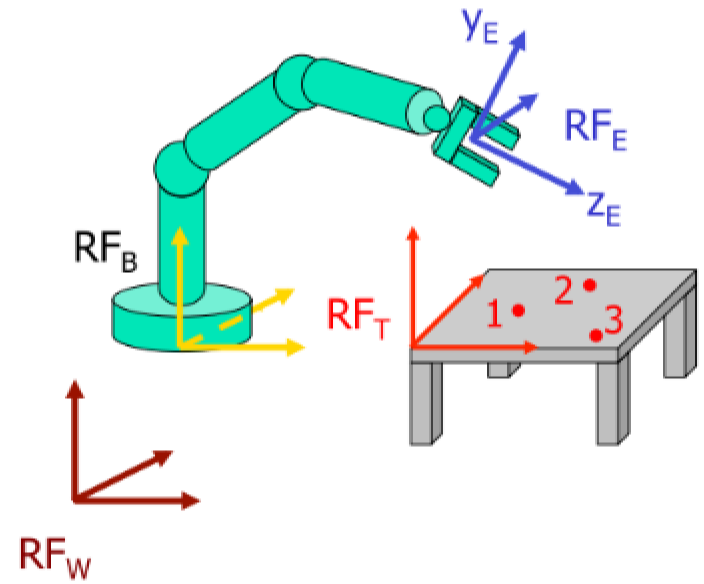
Forward Kinematics

- The structure of the forward kinematics function depends from the chosen r

$$r = f\ (q)$$

- Methods for computing $f\,(q)$:
  - Geometric (by inspection)
  - Systematic: Assigning coordinate frames to the robot's links and using homogenous transformation matrices
    - Denavit-Hartengerg (DH) – Frame assignment
    - or modified Denavit-Hartenberg (Craig, 2005)

Forward Kinematics: Geometric



Known once the robot is installed

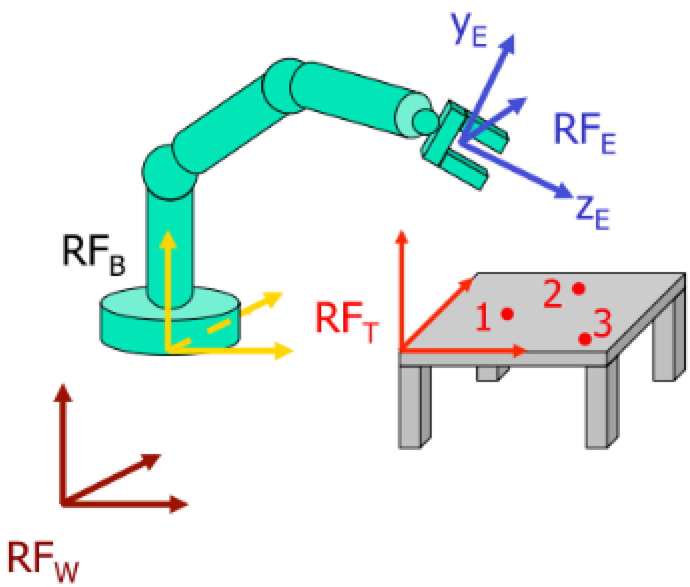
Task definition relative to the robot's end-effector

$$f_r(q) = {}^{B}T_E = ({}^{W}T_B)^{-1}\,{}^{W}T_{Task}\,({}^{E}T_{Task})^{-1}$$

Kinematics of the robot arm

Task definition

Forward Kinematics: Geometric



$$\boxed{{}^{W}T_{Task} = {}^{W}T_{B}\,{}^{B}T_{E}\,{}^{E}T_{Task}}$$

Known once the robot is installed

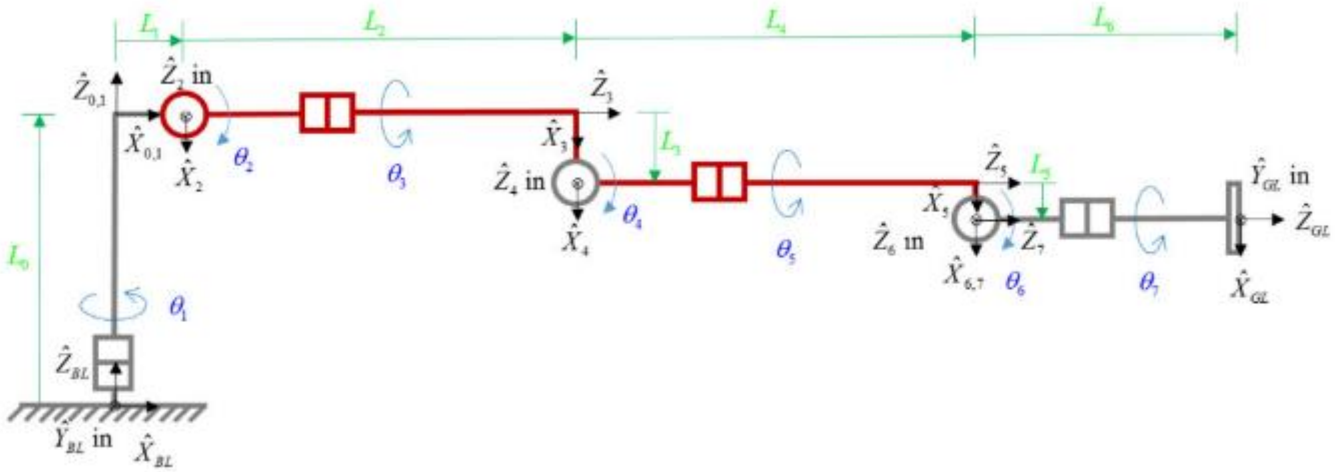
Task definition relative to the robot's end-effector

$$f_r(q) = {}^{B}T_{E} = \left({}^{W}T_{B}\right)^{-1}\,{}^{W}T_{Task}\left({}^{E}T_{Task}\right)^{-1}$$

Kinematics of the robot arm

Task definition

Baxter Forward Kinematics



Baxter left arm: 7-DoF kinematic diagram with coordinate frames

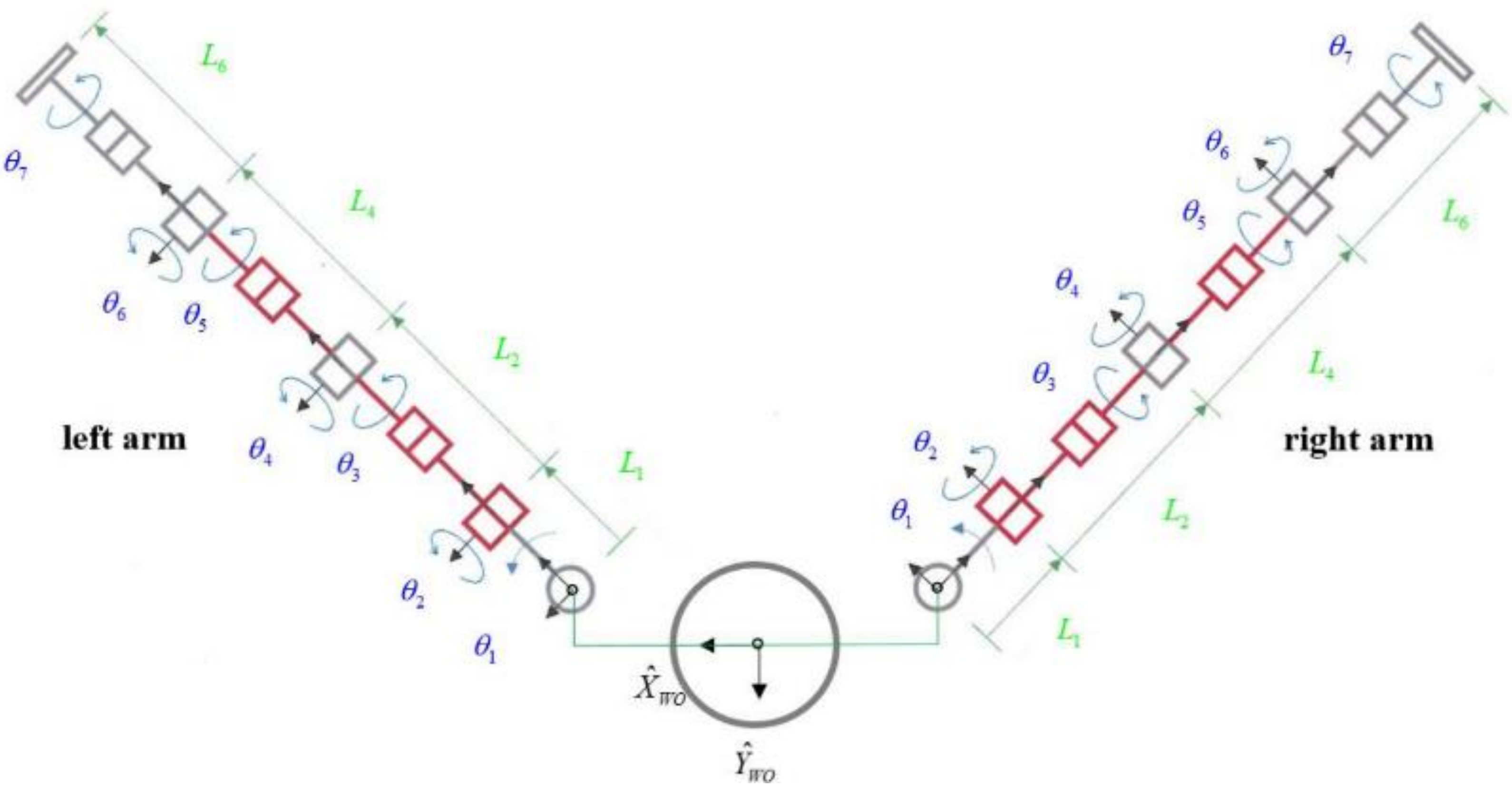
Joint Name	Joint Variable
S_0	θ_1
S_1	θ_2
E_0	θ_3
E_1	θ_4
W_0	θ_5
W_1	θ_6
W_2	θ_7

DH parameters

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	-90°	L_1	0	θ_2+90°
3	90°	0	L_2	θ_3
4	-90°	L_3	0	θ_4
5	90°	0	L_4	θ_5
6	-90°	L_5	0	θ_6
7	90°	0	0	θ_7

No questions about this in the exam!

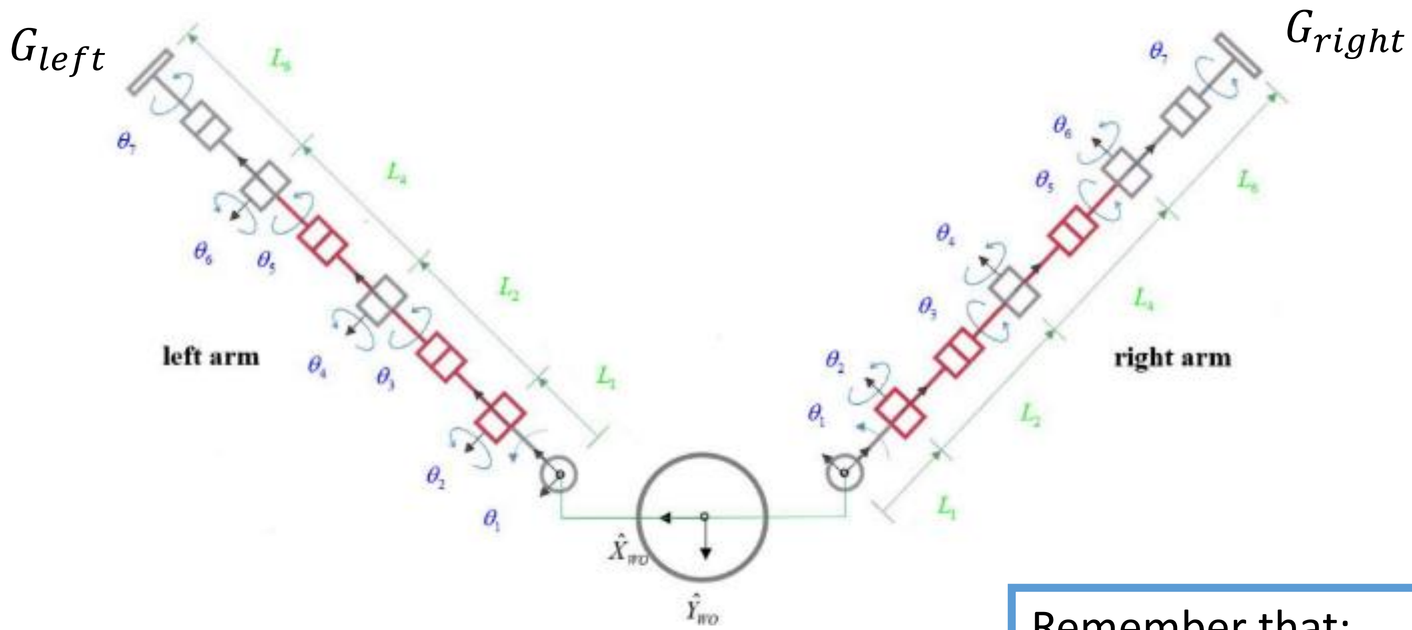
Baxter Forward Kinematics



Baxter left and right arms kinematic diagrams
(Top view, zero joint angles)

No questions about this in the exam!

Baxter Forward Kinematics



Remember that:

$$^{A}T_{B} = \begin{bmatrix} & {}^{A}R_{B} & & {}^{A}p_{AB} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Given $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7,)$ – joint angles
  - ${}^{0}T_{7} = {}^{WO}T_{G}$ (either left or right arm):

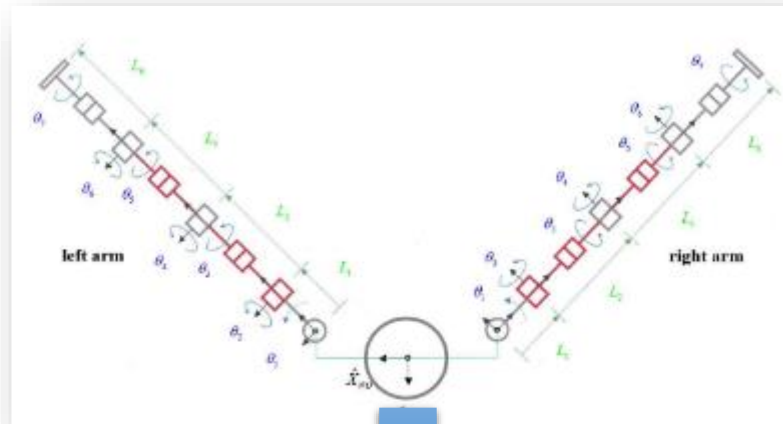
$${}^{0}T_{7} = {}^{0}T_{1}\ {}^{1}T_{2}\ {}^{2}T_{3}\ {}^{3}T_{4}\ {}^{4}T_{5}\ {}^{5}T_{7}\ {}^{6}T_{7}$$
$${}^{7}T_{0} = \left({}^{0}T_{7}\right)^{-1}$$

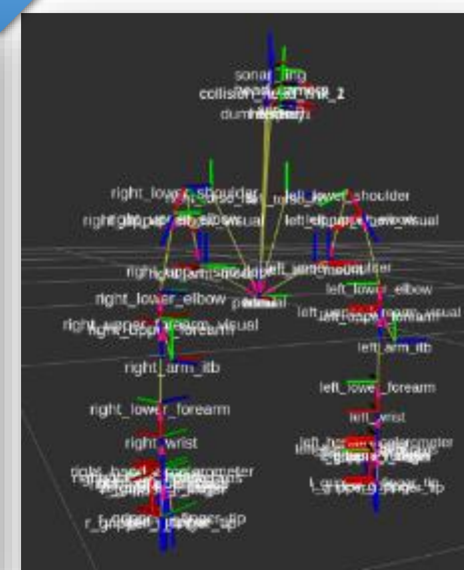
Forward Kinematics

Modelling Robots in ROS

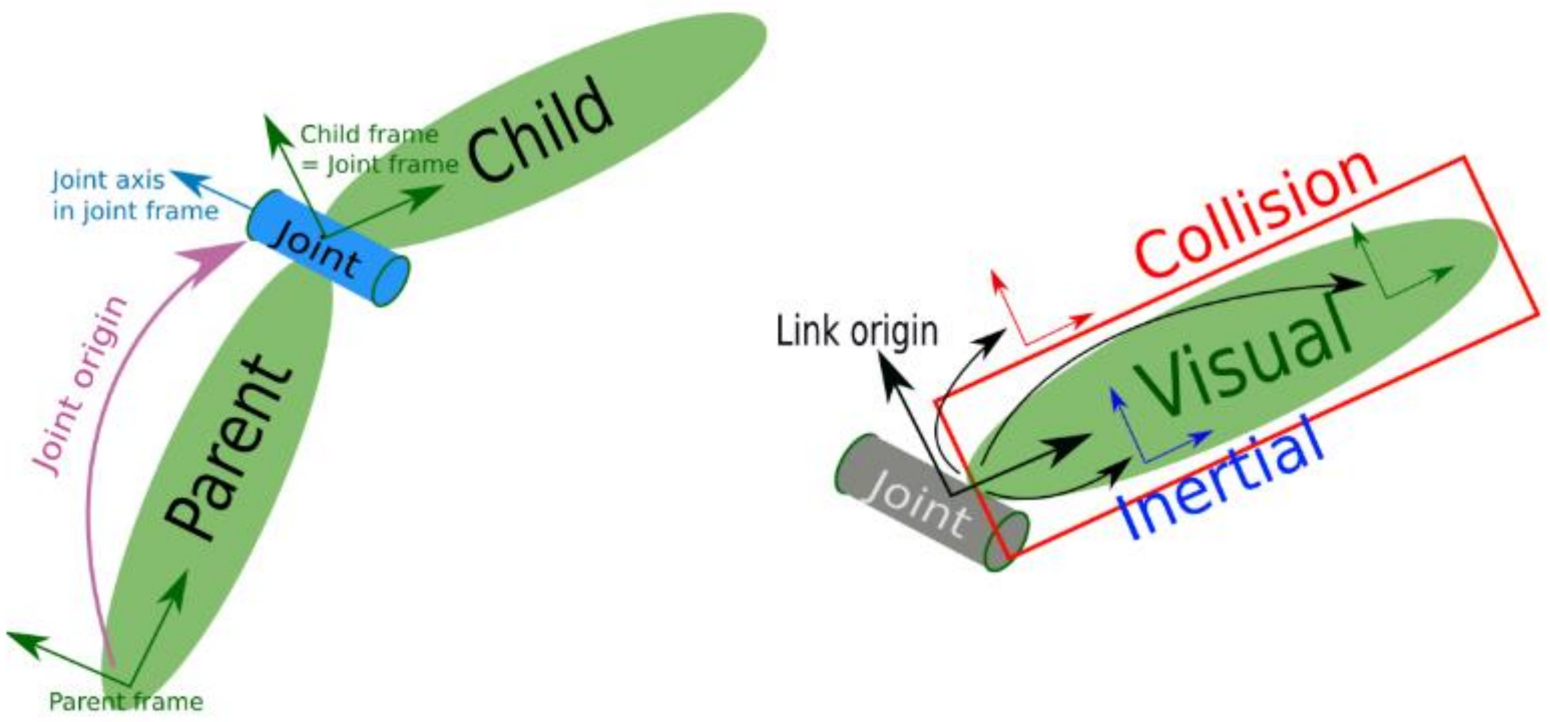
Unified Robot Description Format

- What:
  - Unified Robot Description Format; aka **URDF**
  - Kinematic and basic physics description of a robot
- How:
  - XML format
  - Tags: link, joint, transmission, ...
  - Kinematic tree structure
  - Order in the file does not matter





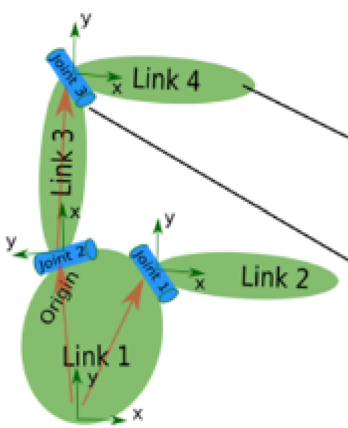
URDF: Link and Joint Representation




```
rosrun tf tf_echo /parent /child
```

URDF: Modelling Robots

- Description consists of a set of link elements and a set of joint elements
- Joints connect the links together



robot.urdf

```
<robot name="robot">
  <link> ... </link>
  <link> ... </link>
  <link> ... </link>

  <joint> .... </joint>
  <joint> .... </joint>
  <joint> .... </joint>
</robot>
```

```
<link name="link_name">
  <visual>
    <geometry>
      <mesh filename="mesh.dae"/>
    </geometry>
  </visual>
  <collision>
    <geometry>
      <cylinder length="0.6" radius="0.2"/>
    </geometry>
  </collision>
  <inertial>
    <mass value="10"/>
    <inertia ixx="0.4" ixy="0.0" …/>
  </inertial>
</link>
```

```
<joint name="joint_name" type="revolute">
  <axis xyz="0 0 1"/>
  <limit effort="1000.0" upper="0.548" … />
  <origin rpy="0 0 0" xyz="0.2 0.01 0"/>
  <parent link="parent_link_name"/>
  <child link="child_link_name"/>
</joint>
```

More info:
http://wiki.ros.org/urdf/XML/model

URDF: Usage in ROS

- The robot description (URDF) is stored in the parameter server (typically) under `/robot_description`
- You can visualize the robot model in Rviz with the `RobotModel` plugin as in Lab 1

baxter.urdf

```xml
<link name="base">
  </link>
  <link name="torso">
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://baxter_description/meshes/torso/base_link.DAE"/>
      </geometry>
      <material name="darkgray">
        <color rgba=".2 .2 .2 1"/>
      </material>
    </visual>
    <collision>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://baxter_description/meshes/torso/base_link_collision.DAE"/>
      </geometry>
    </collision>
    <inertial>
      <origin rpy="0 0 0" xyz="0.000000 0.000000 0.000000"/>
      <mass value="35.336455"/>
      <inertia ixx="1.849155" ixy="-0.000354" ixz="-0.154188" iyy="1.662671" iyz="0.003292" izz="0.802239"/>
    </inertial>
  </link>
```

URDF: Type of joints

- **revolute** - a hinge joint that rotates along the axis and has a limited range specified by the upper and lower limits
- **continuous** - a continuous hinge joint that rotates around the axis and has no upper and lower limits
- **prismatic** - a sliding joint that slides along the axis, and has a limited range specified by the upper and lower limits
- **fixed** - This is not really a joint because it cannot move
- **floating** - This joint allows motion for all 6 degrees of freedom
- **planar** - This joint allows motion in a plane perpendicular to the axis

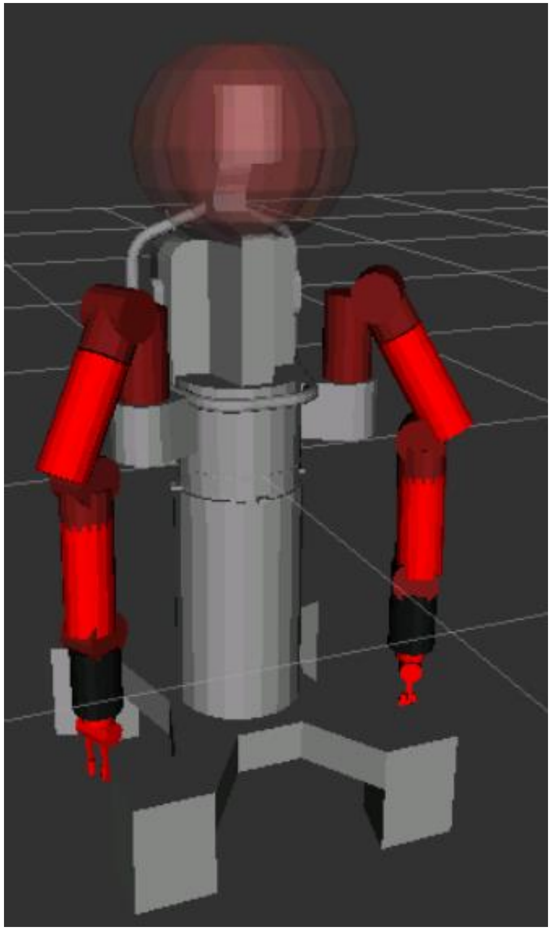
URDF: Collision and Physical properties

- To simulate a robot in Gazebo or to use the URDF with a motion planner
  - it is necessary to add physical and collision properties

- To do this:
  - Set on every link the dimension of the geometry to calculate possible collisions
  - the weight that will give us the inertia
  - Etc.

URDF: Collision and Physical properties

- The collision element:
  - is a direct sub element of the link object, at the same level as the visual tag
  - defines its shape, the same way the visual element does, with a geometry tag
- The format for the geometry tag is exactly the same as with the visual.
  - The origin is defined a as a sub element of the collision tag (as with the visual)



XML Macros

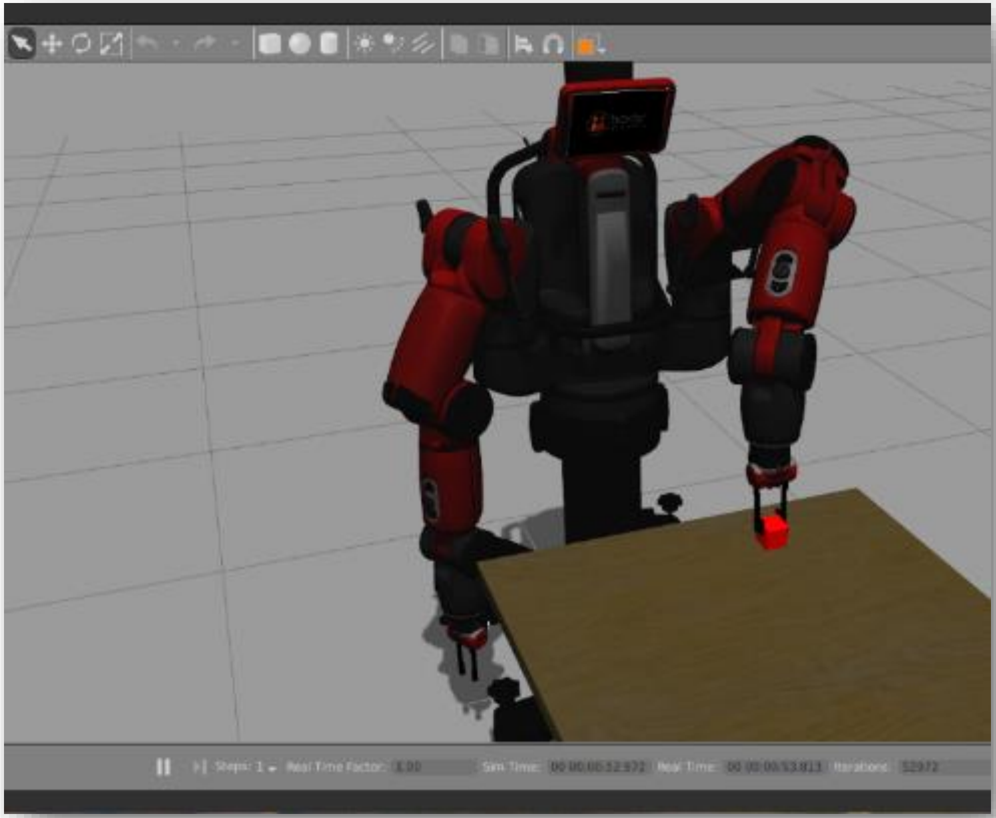
- What:
  - XML Macro used for URDF simplification
    - **XACRO**
  - Increase modularity
  - Reduce redundancy
  - Permit Parametrization
  - Generate URDFs on-the-fly
- How:
  - Includes (like imports in Python or Django template system)
  - Macros
  - Properties
  - Parameters
  - Command line and output to stdout

XACRO: Typical use

- Reduce redundant code
  - Repeated links should be defined as macros and called with parameters
- Parametrized entities
  - Use parameters for length of links
  - Use math for origin or inertia calculation
  - Shape parameters according to length
- Modularity:
  - Generic code can be put as include, to be reused in other files
  - Separate concerns to easily deactivate parts of the URDF
    - e.g. remove gazebo tags

Simulation Description Format (SDF)

- Defines an XML standard to describe
  - Environments (lighting, gravity etc.)
  - Objects (static and dynamic)
  - Sensors
  - Robots
- SDF is the standard format for Gazebo
- Gazebo converts a URDF to SDF automatically



More info: http://sdformat.org

URDF, XACRO & SDF

- More on these in the coming labs!

Lecture starts 12:05

Inverse Kinematics

Inverse kinematics

- Forward kinematics are always unique!
- Not the same case for inverse kinematics….
- Problem statement:
  - Given a desired end-effector pose (position + orientation), find the values of the joint angles

Forward kinematics:

$$r = f\,(q)$$

Inverse kinematics:

$$q = f^{-1}\,(r)$$

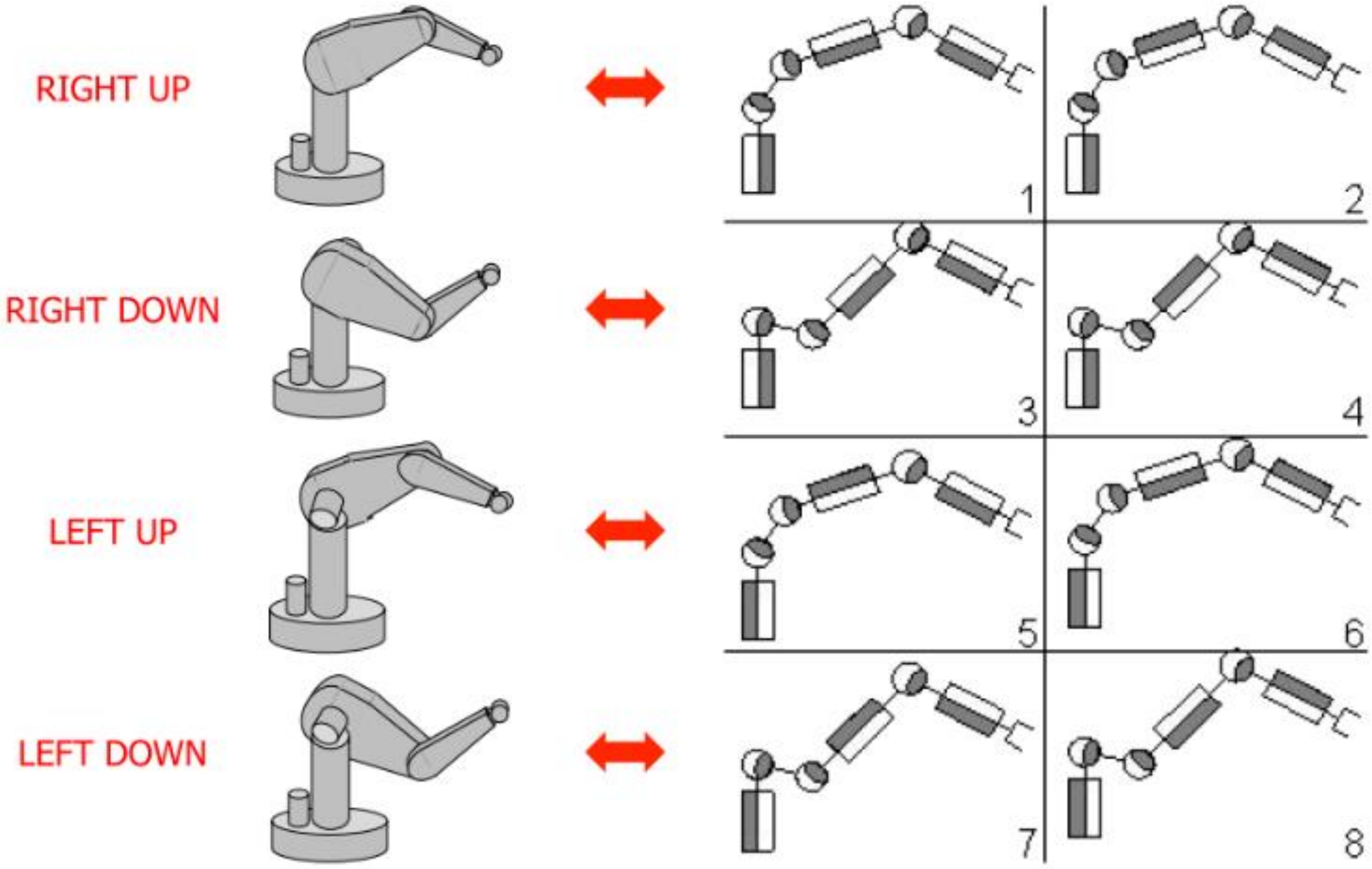
Inverse kinematics

$$\boxed{\boldsymbol{q} = \boldsymbol{f}^{-1}(\boldsymbol{r})}$$

- It is a nonlinear problem!
  - existence of a solution (workspace definition)
  - uniqueness/multiplicity of solutions

- Different solution methods
  - Analytical,
  - Numerical (Iterative, sampling-based)
  - Machine learning
  - etc.

Inverse kinematics



Inverse kinematics

- **Primary Workspace $\boldsymbol{WS_1}$**: set of all positions, p, that can be reached with at least one orientation (θ)
  - out of WS_1, there is no solution to the problem
  - when $p \in WS_1$, there is a suitable θ for which a solution exists
- **Secondary (i.e. Dexterous) Workspace $\boldsymbol{WS_2}$**: set of positions p that can be reached with any orientation within the reach of the robot
  - when $p \in WS_2$, there exists a solution for any θ

$$WS_2 \subseteq WS_1$$

Baxter Workspace

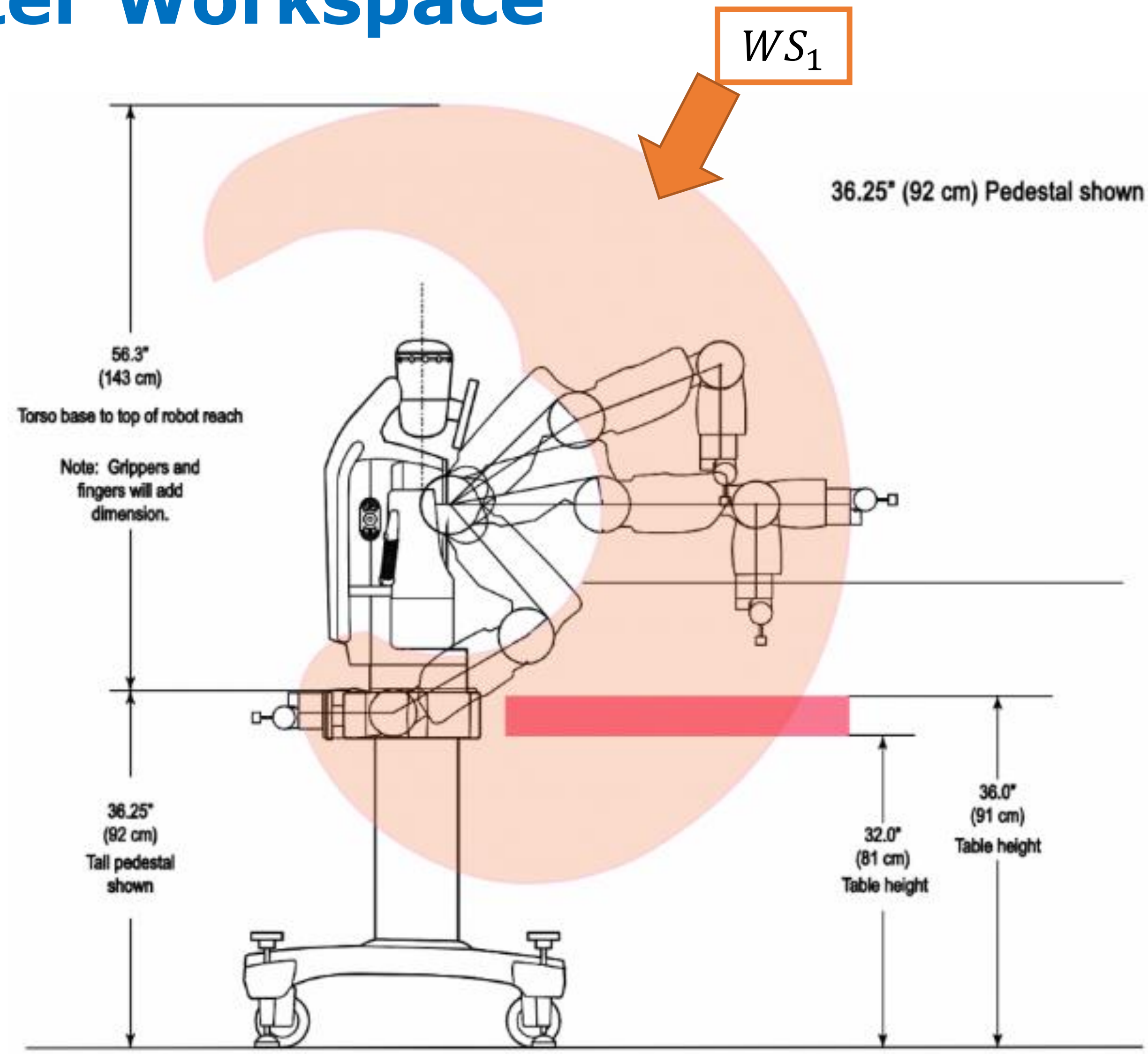


Image from: http://sdk.rethinkrobotics.com/wiki/Workspace_Guidelines

Baxter Workspace

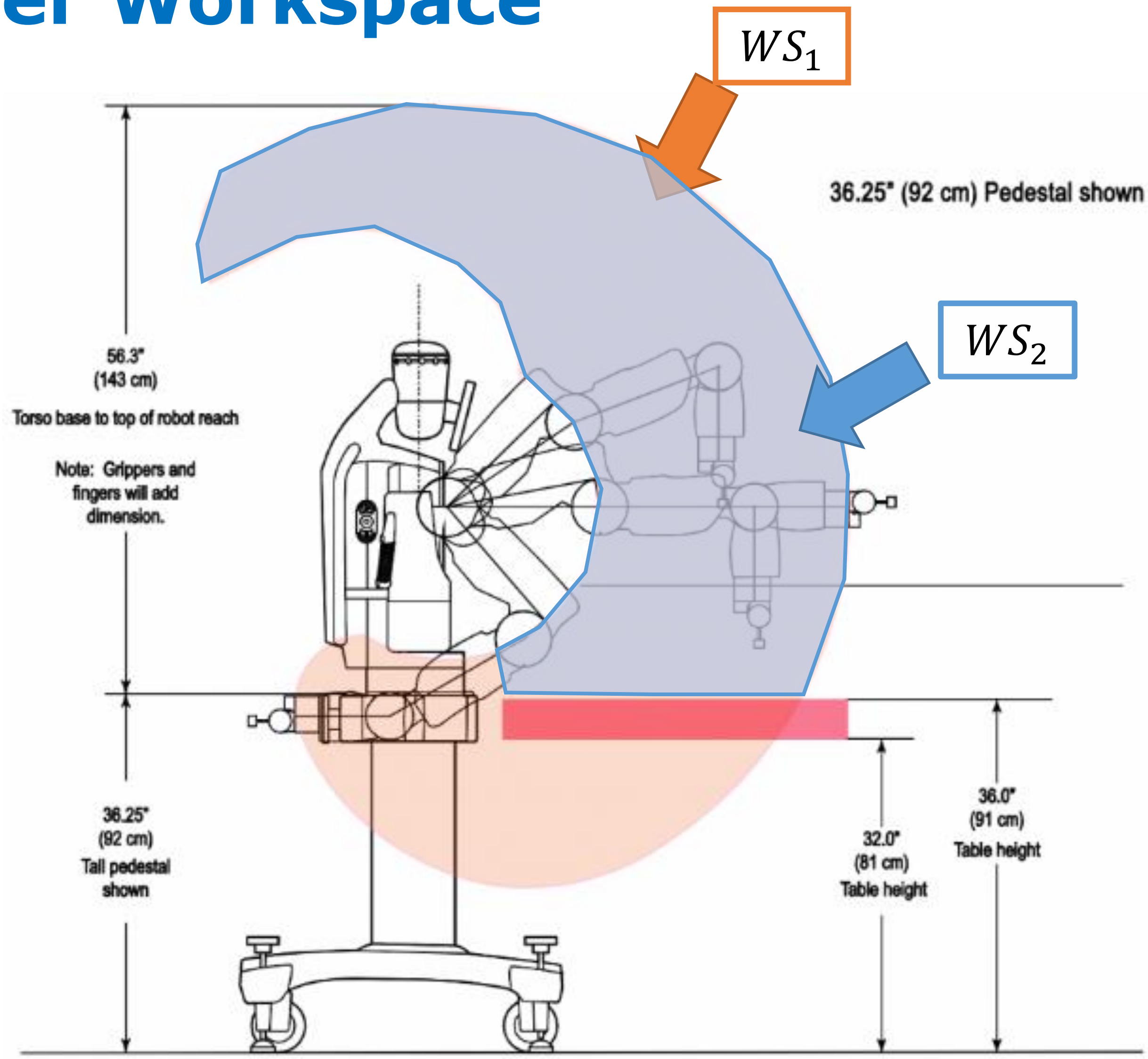


Image from: http://sdk.rethinkrobotics.com/wiki/Workspace_Guidelines

Baxter Workspace

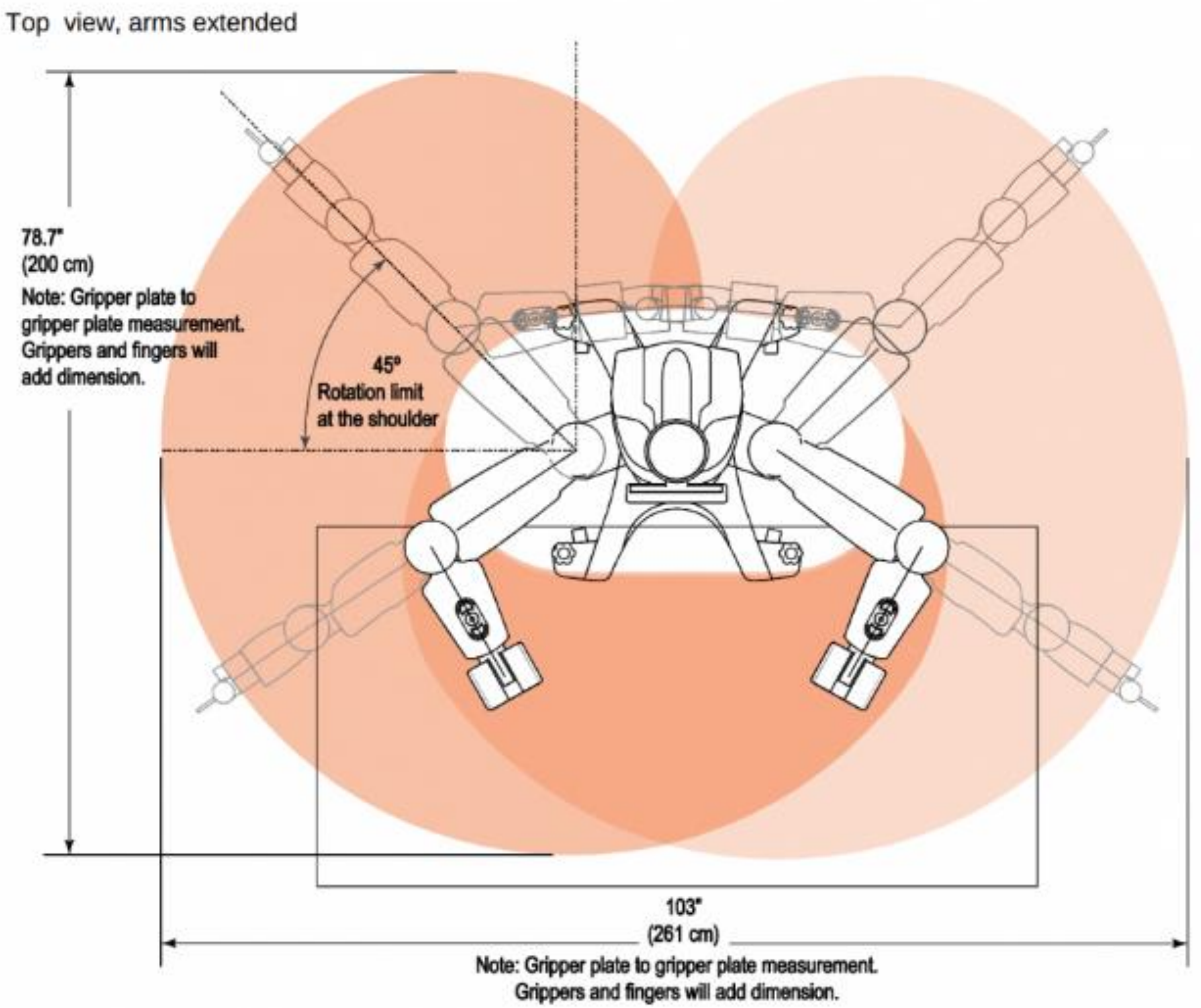


Image from: http://sdk.rethinkrobotics.com/wiki/Workspace_Guidelines

Summary: Inverse kinematics

- Next week we'll look at how modern motion planners deal with the inverse kinematic problem
- Baxter SDK → provides an analytical solution for both arms
  - **BUT it doesn't take into account collisions!**
  - More about Baxter Inverse Kinematics in labs

Next lecture

- Suggested reading from RVC book
  - **5.2** → Mobile robots but algorithms used for IK and motion planning!
  - **7.2** → IK analytical and numerical solutions
  - **7.3** → Rationale for motion planning in robotic arms