Fundamentals of Manipulation

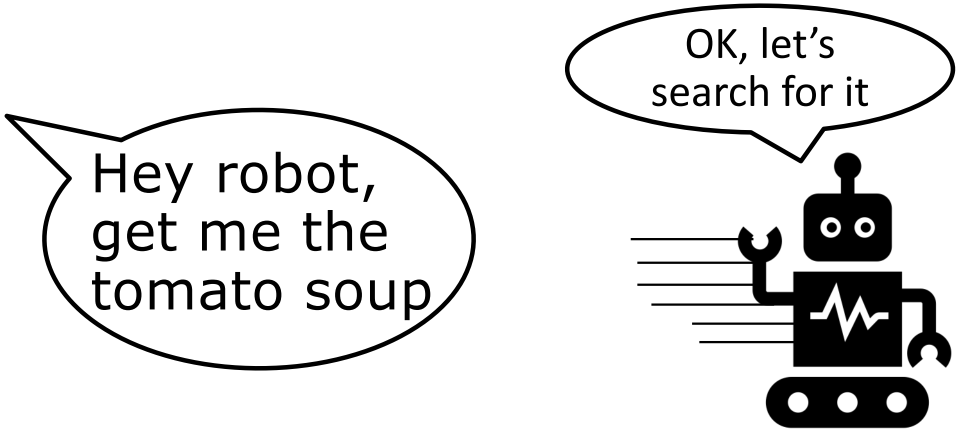
Maren Bennewitz, Nils Dengler
Humanoid Robots Lab, University of Bonn

Goal of This Chapter

- Learn the fundamentals of robotic manipulation
- Understand the concept of **kinematic chains**
- Learn how to calculate **forward** and **inverse kinematics**
- Understand **Denavit-Hartenberg parameters**
- Understand how to use **reachability maps** to compute the robot’s **kinematic capabilities**

Why Is Manipulation Needed?

- To act appropriately, robots must **observe** the world as shown in prior lectures
- However, **passively** observing the world is not enough
- Smart robots must integrate **perception and action** to gain relevant information and execute tasks

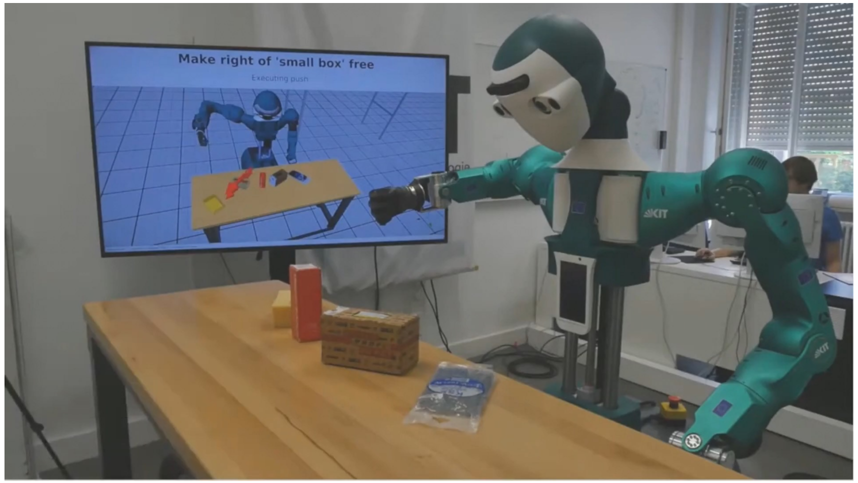


Use Manipulation Actions to...

... look behind occluding objects

Yao et al., ICRA 2025

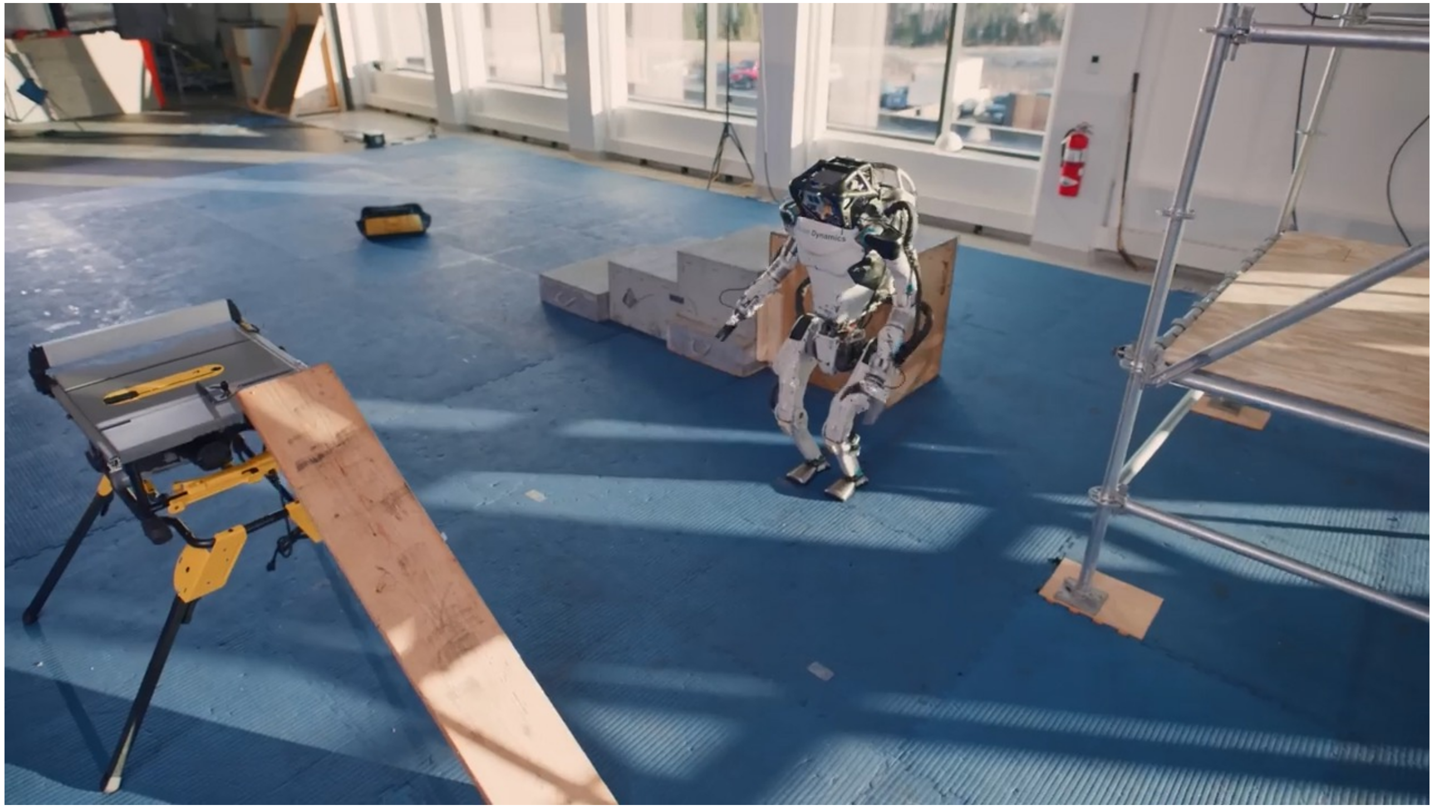
... re-arrange objects



Paus et al., ICRA 2020

Use Manipulation Actions to...

... combine with whole-body body motion planning

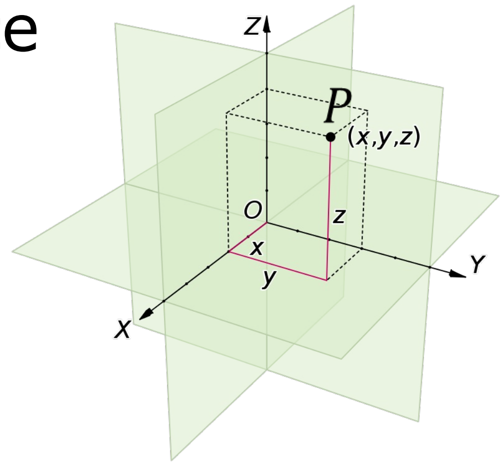


Boston Dynamics: https://www.youtube.com/watch?v=-e1_QhJ1EhQ

Recap: Transformation Basics

- **Cartesian Coordinates:**
  One way to represent the position in 3D space

$$P = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$



- **Homogeneous Coordinates**
  3D point represented by a 4D vector

$$P = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

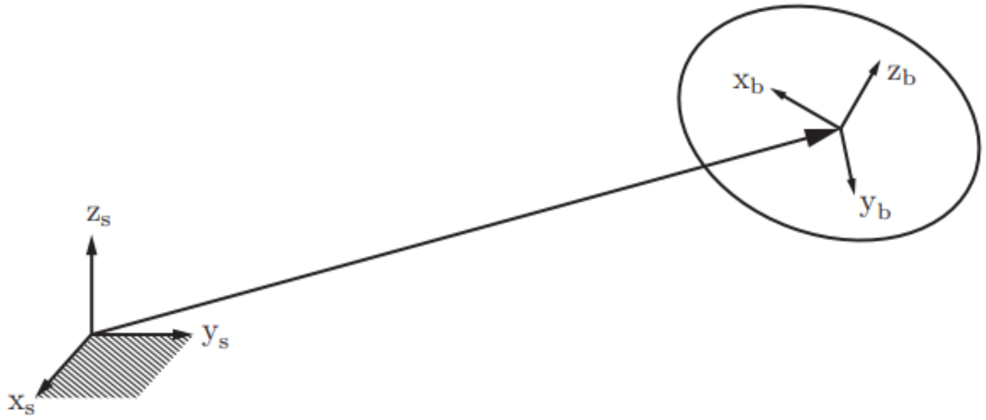
Recap: Transformation Basics

- Rigid body transformation in 3D can be described by a **translation** followed by a **rotation**

Translation

- Translation of a rigid body from A to B can be described by a 3D vector

$$r_{AB} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$



Recap: Transformation Basics

- Rigid body transformation in 3D can described by a **translation** followed by a **rotation**

Rotation

- 3x3 matrix to represent the rotation of a rigid body in 3D space
- Two important properties:
  - $R^{-1} = R^T \rightarrow R * R^T = I$
  - $\det(R) = 1$

Recap: Transformation Basics

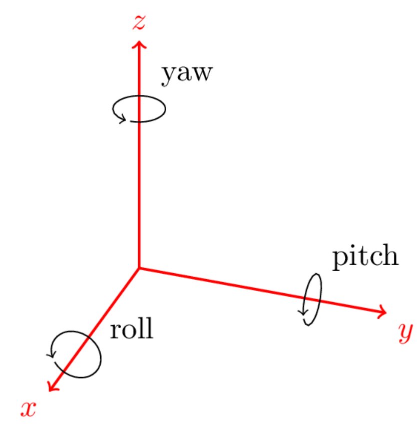
Euler Angles

- **One way** to represent a general rotation of a rigid body
- The rotation is described by a chain of rotations around 3 different axes
- We can obtain a general rotation matrix by using **matrix multiplication**

Recap: Transformation Basics

Example:

Using roll-pitch-yaw angles and **matrix multiplication**



$$R = R_z(\theta_z)\, R_y(\theta_y)\, R_x(\theta_x)$$

$$R = \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 \\ \sin(\theta_z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_y) & 0 & \sin(\theta_y) \\ 0 & 1 & 0 \\ -\sin(\theta_y) & 0 & \cos(\theta_y) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_x) & -\sin(\theta_x) \\ 0 & \sin(\theta_x) & \cos(\theta_x) \end{bmatrix}$$

Recap: Transformation Basics

Homogeneous Transformation Matrix

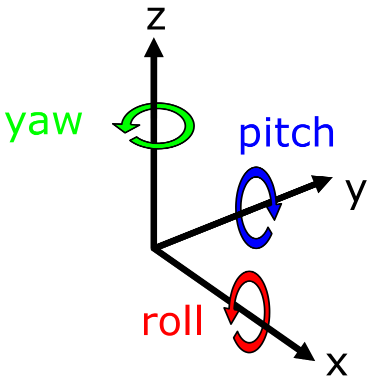
- Encodes the transformation (i.e., rotation and translation) of a rigid body in 4x4 matrix
- Element of SE(3)

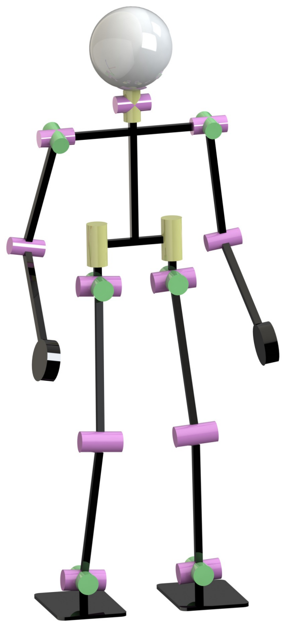
$$T = \begin{bmatrix} [R]_{3X3} & \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} \quad \rightarrow \quad T = \begin{bmatrix} R & P \\ 0 & 1 \end{bmatrix}$$

- The inverse can be calculated as: $T^{-1} = \begin{bmatrix} R^T & -R^T P \\ 0 & 1 \end{bmatrix}$

Kinematics

- The humanoid body is relatively complex, it has more than 20 degrees of freedom





Neck: yaw, pitch

Shoulder: roll, pitch

Elbow: pitch

Hip: roll, pitch, yaw

Knee: pitch

Ankle: roll, pitch

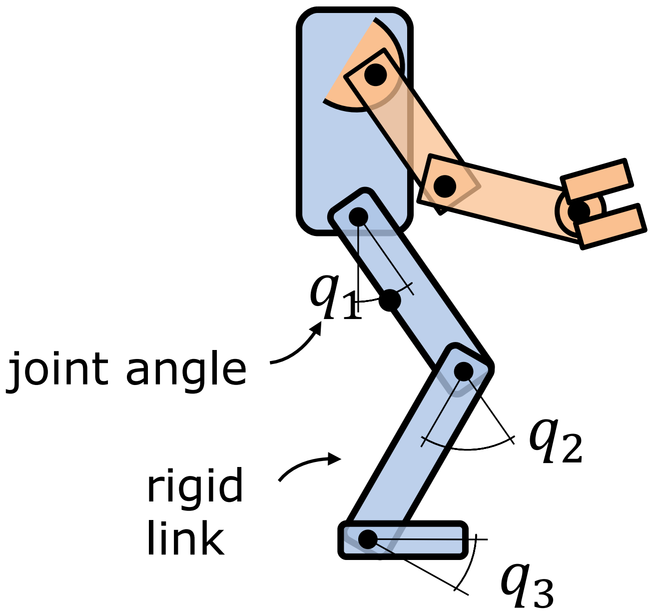
NimbRo-OP, AIS, Uni Bonn

Kinematic Chains

- Complex bodies are organized in **kinematic chains** of rigid links that are connected by joints

Leg chain

hip
↓
knee
↓
ankle

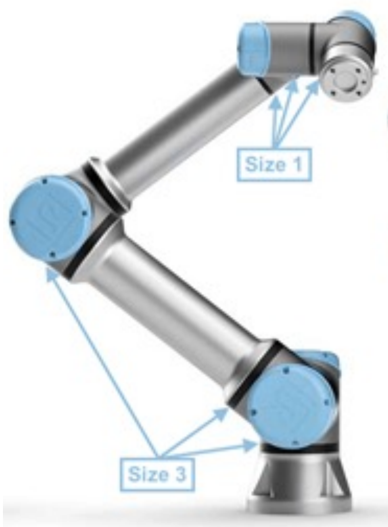


Arm chain

shoulder
↓
elbow
↓
wrist

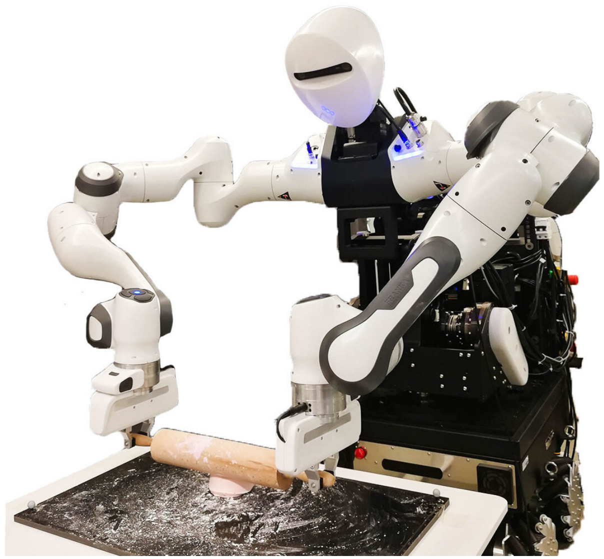
Types of Kinematic Chain

- Open kinematic chain (e.g., robotic arm)
- Closed kinematic chain (e.g., delta robot)
- Semi-closed kinematic chain (e.g., bimanual manipulation)



Courtesy: Universal Robots

Courtesy: Fanuc



Courtesy: Clover Lab

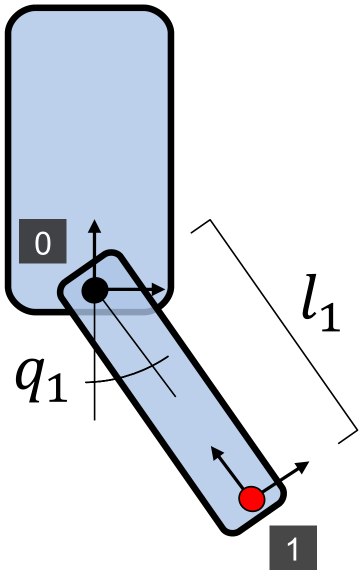
Kinematic Parameters

- Rigid body transformations can be described by **translation followed by a rotation**
- Each link of the kinematic chain is transformed relative to its parent link
- Each joint can be explained by:
  - **Joint parameter** (i.e., rotation)
  - **Relative transformation** to other joints

$$P^0 = \begin{bmatrix} P_x^0 \\ p_y^0 \end{bmatrix}$$

$$P^1 = ?$$

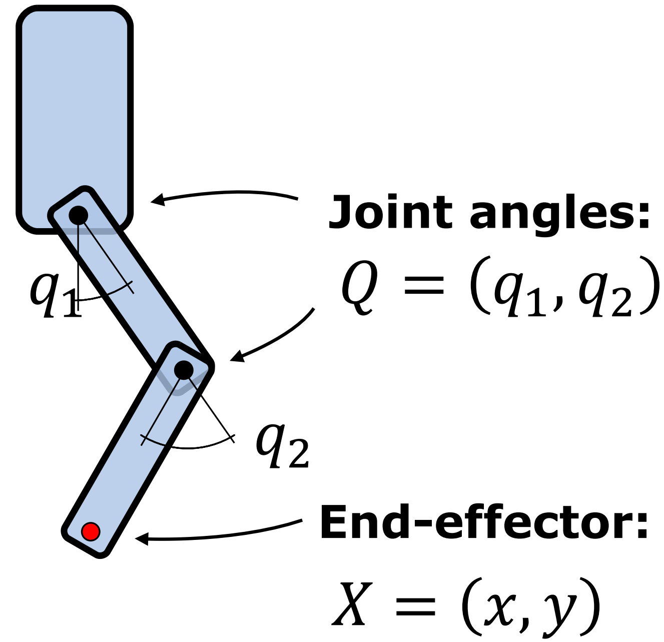
$$P^1 = \begin{bmatrix} cosq_1 & -sinq_1 \\ sinq_1 & cosq_1 \end{bmatrix} \left(P^0 + \begin{bmatrix} 0 \\ -l_1 \end{bmatrix} \right)$$



Kinematics

- Given the joint angles, what is the end-effector (EE) pose?
- Given an EE pose what are possible joint angles to reach it?

- **Forward Kinematics:** $X = f(Q)$
- **Inverse Kinematics:** $Q = f^{-1}(X)$



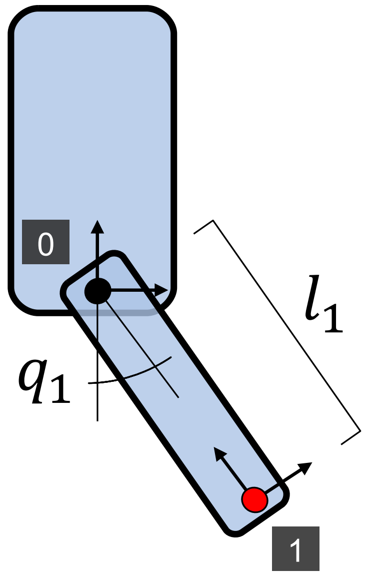
Forward Kinematics

- **Homogeneous coordinates** to represent the translation and the rotation as a matrix multiplication

$$P^0 \Rightarrow \begin{bmatrix} P_x^0 \\ P_y^0 \\ 1 \end{bmatrix} \quad P^1 \Rightarrow \begin{bmatrix} P_x^1 \\ P_y^1 \\ 1 \end{bmatrix}$$

$$P^1 = \begin{bmatrix} cosq_1 & -sinq_1 \\ sinq_1 & cosq_1 \end{bmatrix} \left(P^0 + \begin{bmatrix} 0 \\ -l_1 \end{bmatrix} \right)$$

$$\boxed{\begin{bmatrix} P_x^1 \\ P_y^1 \\ 1 \end{bmatrix} = \begin{bmatrix} cosq_1 & -sinq_1 & 0 \\ sinq_1 & cosq_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -l_1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x^0 \\ P_y^0 \\ 1 \end{bmatrix}}$$



Forward Kinematics

- Combine translation and rotation into one transformation matrix and use a **symbolic notation**

$$P^1 = R(q_1) \cdot \mathrm{t}(l_1) \cdot P^0$$

$$\begin{bmatrix} P_x^1 \\ P_y^1 \\ 1 \end{bmatrix} = \begin{bmatrix} cosq_1 & -sinq_1 & 0 \\ sinq_1 & cosq_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -l_1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x^0 \\ P_y^0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} P_x^1 \\ P_y^1 \\ 1 \end{bmatrix} = \begin{bmatrix} cosq_1 & -sinq_1 & l_1 sinq_1 \\ sinq_1 & cosq_1 & -l_1 cosq_1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x^0 \\ P_y^0 \\ 1 \end{bmatrix}$$

To

$$\boldsymbol{P^1 = T_0^1(q_1, l_1) \cdot P^0}$$

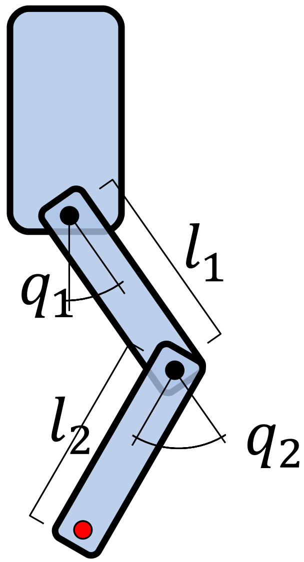
From

Forward Kinematics

- Now, transformations can be easily **concatenated**
- The order of the transformations follows the hierarchy along the kinematic chain

$$P^2 = T_1^2(q_2, l_2) \cdot T_0^1(q_1, l_1) \cdot P^0$$

$$P^2 = T_0^2 \cdot P^0$$



$$P^0 = \begin{bmatrix} P_x^0 \\ P_y^0 \end{bmatrix}$$

$$P^2 = ?$$

Inverse Kinematics (IK)

- IK computes the **joint angle values** so that the **end-effector reaches a desired pose**
- IK is challenging and cannot be as easily computed as FK
- There might exist several possible solutions, or there may be no solution at all

Inverse Kinematics (IK)

- Many different approaches to solving IK problems exist
- **Analytical methods**: closed-form solution
- **Numerical methods**: iteratively calculate a sequence of configurations that approach target pose

Analytical IK Methods

- Closed-form solution (using trigonometry, geometry)
- Computes all IK solutions, determines whether or not a solution exists
- Once the equations are derived, solutions are **very fast to compute**
- No need to define solution parameters or initial guesses
- Often **difficult to define**, must be derived
- Individual equations for robots with different kinematic structures

Numerical IK Methods

- Given a start configuration, **iteratively calculate a sequence of configurations** to reach end-effector pose
- Use the **Jacobian** and try to converge to a solution
- Usually **much slower but also more general**

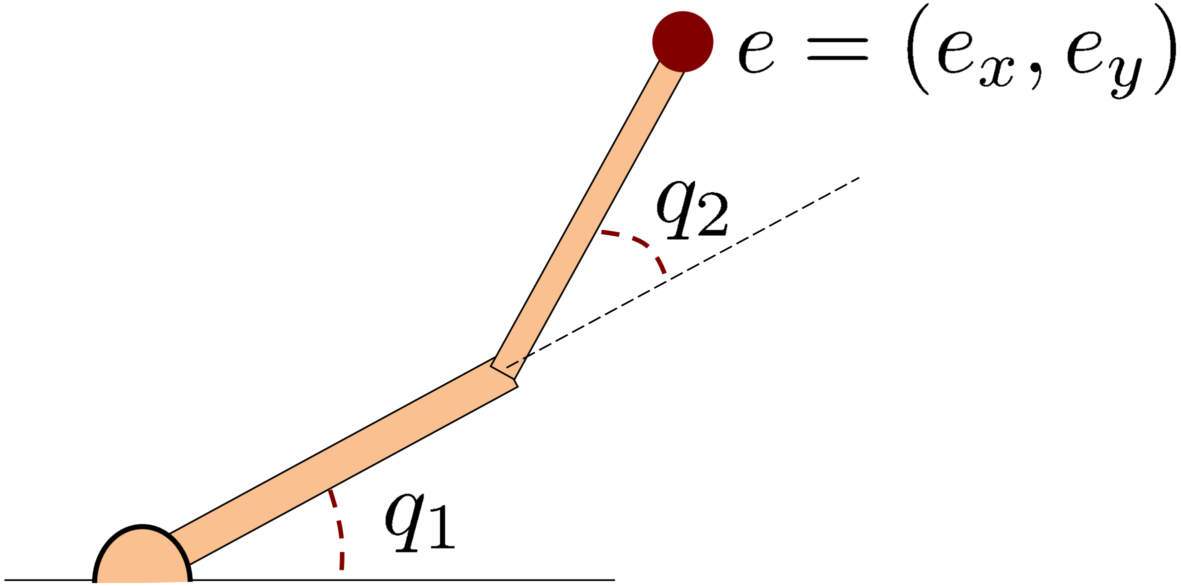
Existing Inverse Kinematics Solvers

- **Analytical solvers**:
  - IKFast: https://github.com/rdiankov/openrave
  - EAIK: https://ostermd.github.io/EAIK/

- **Numerical methods:**
  - Kinematics and Dynamics Library (KDL) http://wiki.ros.org/kdl
  - Robotics Toolbox https://petercorke.github.io/robotics-toolbox-python/IK/ik.html

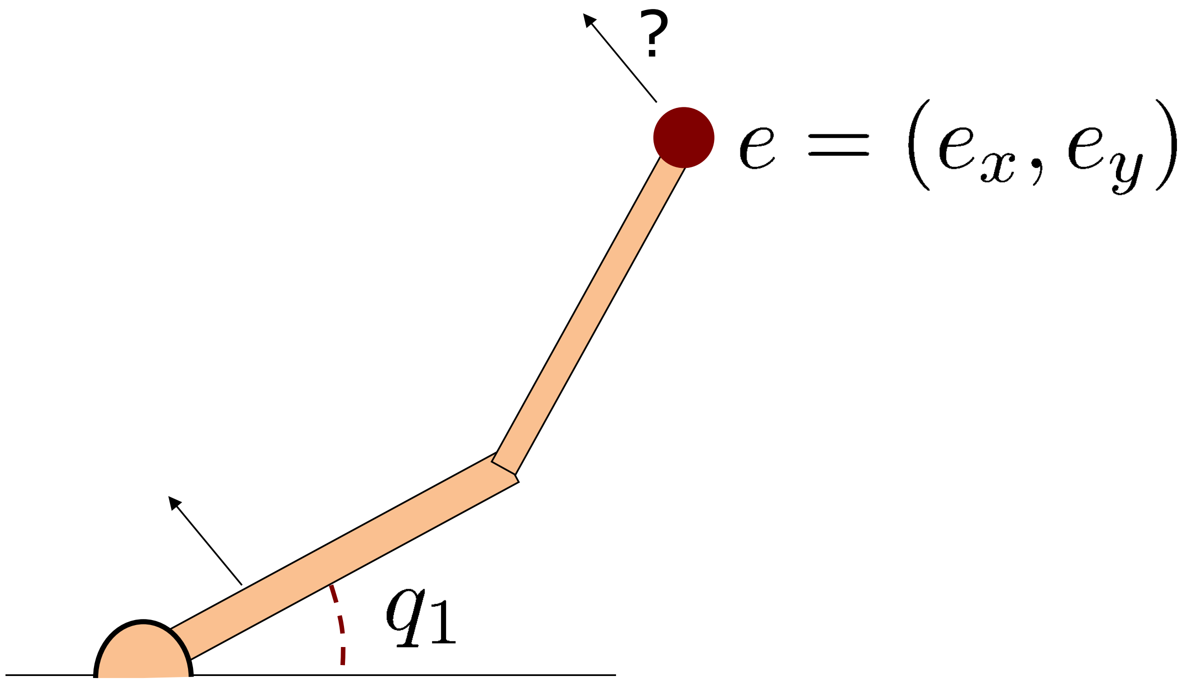
Inverse Kinematics: Example

- Consider a simple 2D robot arm with two 1-DOF joints
- Given a desired end-effector pose $\mathbf{e}$
- Compute joint angles q_1 and q_2



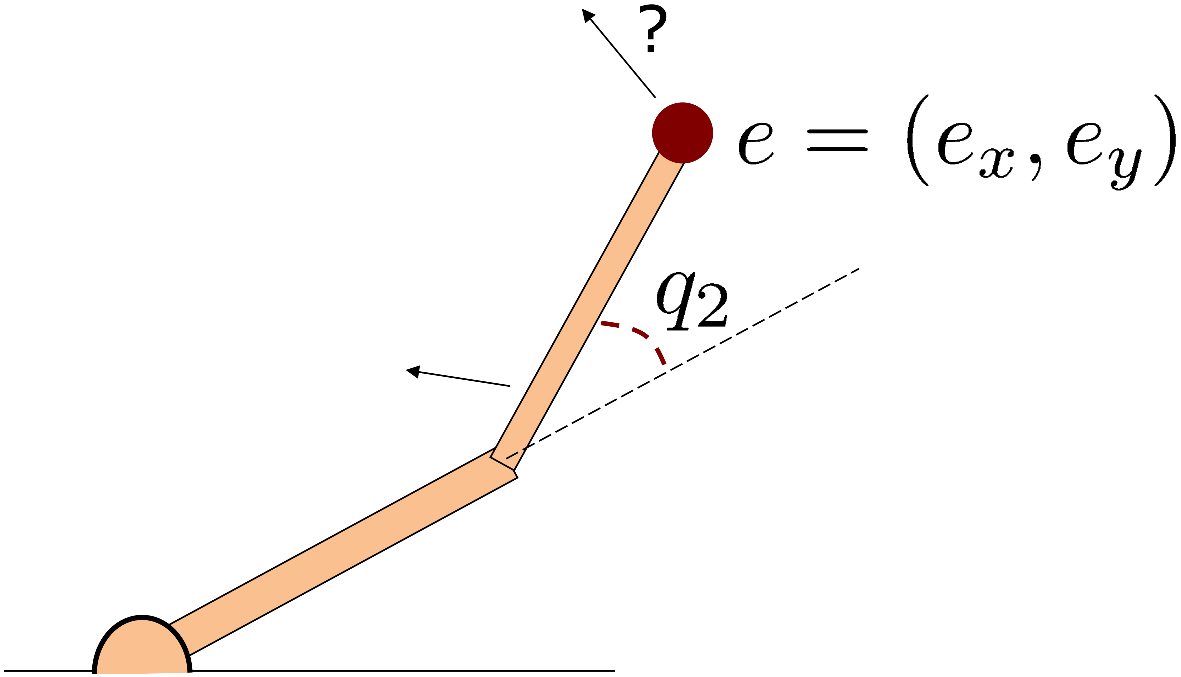
Inverse Kinematics: Example

- If we increased q_1 by a small amount, what would happen to $\mathbf{e}$?



Inverse Kinematics: Example

- If we increased q_1 by a small amount, what would happen to $\mathbf{e}$?



Numerical Approach Using the Jacobian

- Jacobian matrix for this simple example would look like:

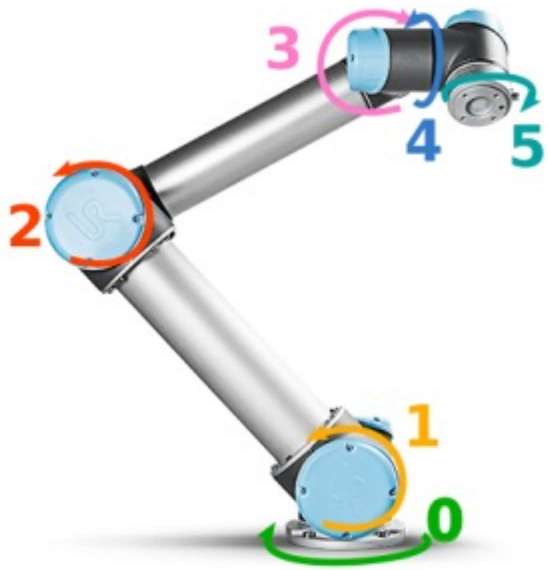
$$\mathbf{J}(\mathbf{e}, \mathbf{q}) \;=\; \begin{pmatrix} \frac{\partial e_x}{\partial q_1} & \frac{\partial e_x}{\partial q_2} \\ \\ \frac{\partial e_y}{\partial q_1} & \frac{\partial e_y}{\partial q_2} \end{pmatrix}$$

- Defines how each component of $\mathbf{e}$ changes w.r.t. joint angle changes
- For any given vector of joint values, we can compute the components of the Jacobian

Numerical Approach Using the Jacobian

- Usually, the Jacobian will be an 6x*N* matrix where *N* is the number of joints
- Jacobian can be computed based on the equations of FK

$$\mathbf{J}(\mathbf{e},\mathbf{q}) = \begin{pmatrix} \frac{\partial e_x}{\partial q_0} & \cdots & \frac{\partial e_x}{\partial q_5} \\ \frac{\partial e_y}{\partial q_0} & \cdots & \frac{\partial e_y}{\partial q_5} \\ \frac{\partial e_z}{\partial q_0} & \cdots & \frac{\partial e_z}{\partial q_5} \\ \frac{\partial e_\phi}{\partial q_0} & \cdots & \frac{\partial e_\phi}{\partial q_5} \\ \frac{\partial e_\theta}{\partial q_0} & \cdots & \frac{\partial e_\theta}{\partial q_5} \\ \frac{\partial e_\psi}{\partial q_0} & \cdots & \frac{\partial e_\psi}{\partial q_5} \end{pmatrix}$$



Numerical Approach Using the Jacobian

- Given a desired incremental change in the end-effector configuration, we can compute the corresponding incremental change of $\mathbf{q}$:

$$\mathbf{J}\Delta\mathbf{q} = \Delta\mathbf{e}$$
$$\Delta\mathbf{q} = \mathbf{J}^{-1}\Delta\mathbf{e}$$

- As $\mathbf{J}$ cannot be inverted in the general case, it is replaced by the pseudoinverse or by the transpose in practice

Numerical Approach Using the Jacobian

- Forward kinematics is a **nonlinear function**
- Thus, we have an approximation that is only valid near the current configuration
- Until the end-effector is close to the desired pose, repeat:
  - Compute the Jacobian
  - Take a small step towards the goal

End-Effector Goal and Step Size

- Let $\mathbf{e}$ represent the current end-effector pose and $\mathbf{g}$ represent its desired goal pose
- Choose a value for $\Delta\mathbf{e}$ that will move $\mathbf{e}$ closer to $\mathbf{g}$, theoretically:

$$\Delta\mathbf{e} = \mathbf{g} - \mathbf{e}$$

- But non-linearity prevents the end-effector to reach the goal exactly
- Thus, to avoid oscillation, take a smaller step:

$$\Delta\mathbf{e} = \alpha(\mathbf{g} - \mathbf{e}), 0 \le \alpha \le 1$$

Basic Jacobian IK Algorithm

while (($\mathbf{g}$ - $\mathbf{e}$) > Threshold):

Compute $\mathbf{J}(\mathbf{e}, \mathbf{q})$ for the current configuration $\mathbf{q}$

Compute $\mathbf{J}^{-1}$

$\Delta\mathbf{e} = \alpha(\mathbf{g} - \mathbf{e})$ // choose a step to take

$\Delta\mathbf{q} = \mathbf{J}^{-1}\Delta\mathbf{e}$ // compute required change in joints

$\mathbf{q} = \mathbf{q} + \Delta\mathbf{q}$ // apply change to joints

Compute resulting $\mathbf{e}$ // by FK

Denavit-Hartenberg (DH) Convention

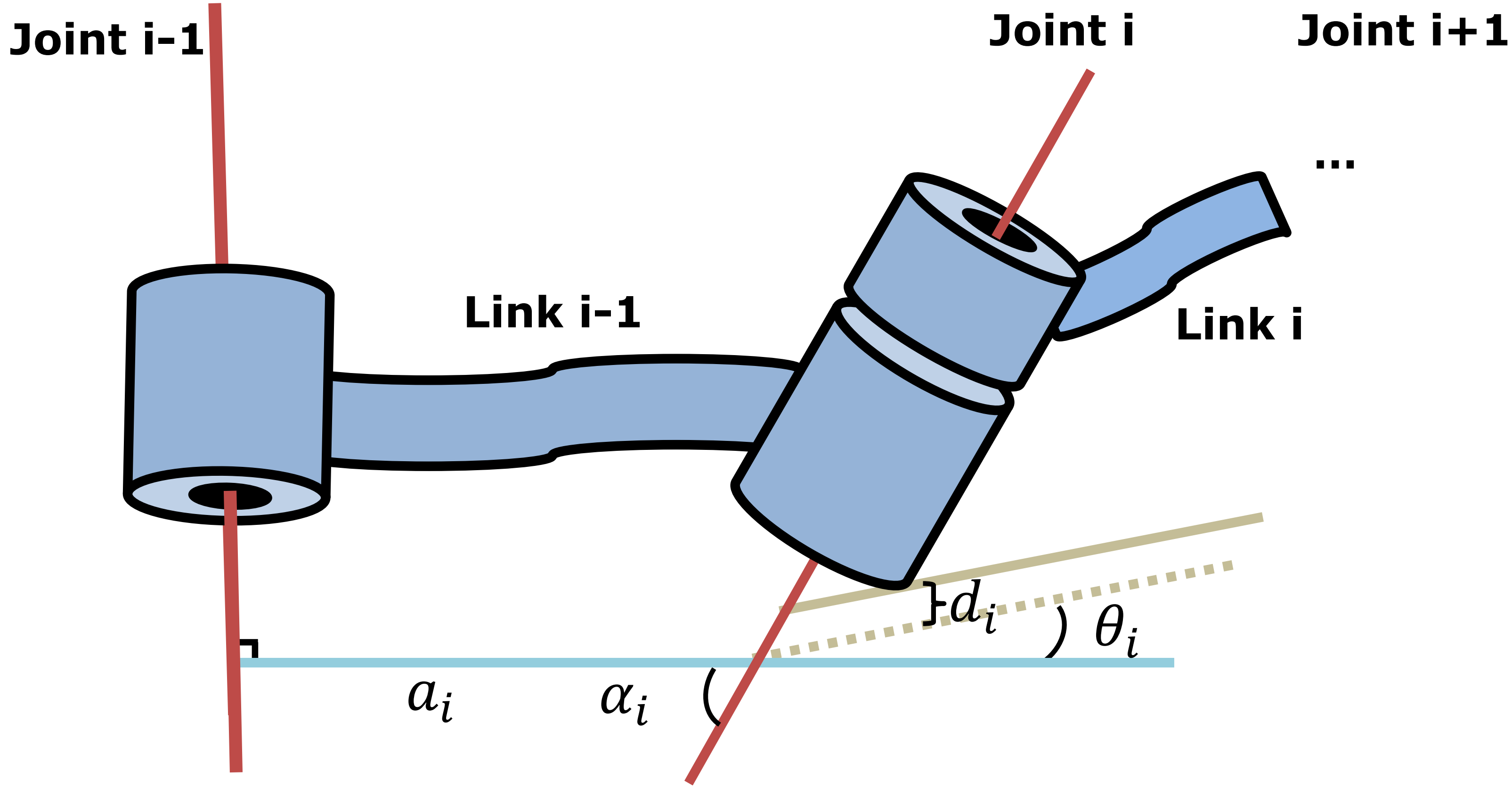
- **Goal:** Reduction of joint and link describing parameters
- **Approach:**
  - Systematic description of translation and rotation between neighbor links
  - Only 4 parameters

Denavit-Hartenberg Parameters

Describe joints via 4 DH-parameters:

- a_i: link length, or distance between two consecutive *z*-axes, measured along the x_{i-1}-axis
- α_i: angle between the *z*-axes of two consecutive joints measured about the x_{i-1}-axis
- d_i: link offset, distance between two consecutive *x*-axes measured along z_i-axes
- θ_i: angle between two consecutive x-axes, measured about the z_i-axis

Denavit-Hartenberg Parameters



- α_i and a_i **describe the joint**
- d_i and θ_i **describe the connection to the next joint**

Denavit-Hartenberg Parameters

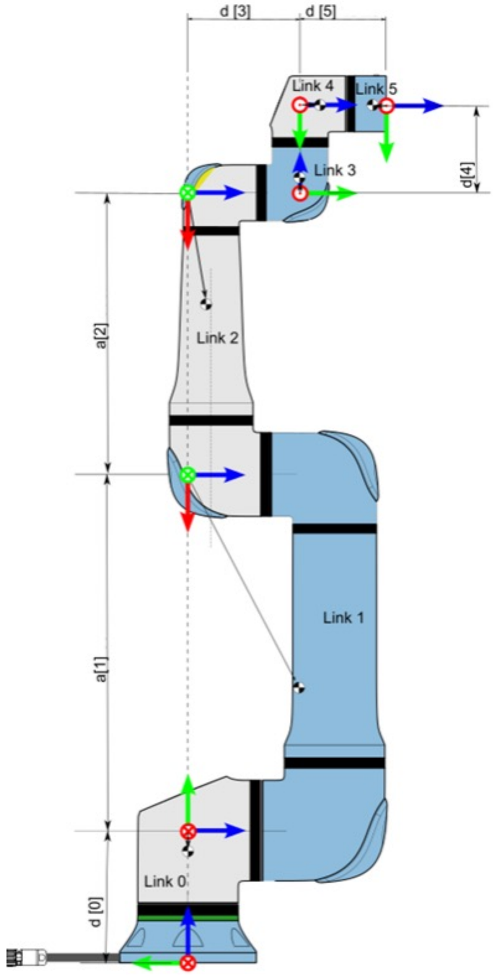
- Simplify computation of forward and inverse kinematics by using homogeneous transformations to describe each joint’s effect to the next one

$$T_{n-1}^{n} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) * \cos(\alpha_i) & \sin(\theta_i) * sin(\alpha_i) & a_i * \cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) * \cos(\alpha_i) & -\cos(\theta_i) * sin(\alpha_i) & a_i * sin(\theta_i) \\ 0 & sin(\alpha_i) & cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

J. Denavit and R. S. Hartenberg, A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices, 1955

DH Example: UR5 Arm

UR5				
Kinematics	theta [rad]	a [m]	d [m]	alpha [rad]
Joint 1	0	0	0.089159	π/2
Joint 2	0	-0.425	0	0
Joint 3	0	-0.39225	0	0
Joint 4	0	0	0.10915	π/2
Joint 5	0	0	0.09465	-π/2
Joint 6	0	0	0.0823	0



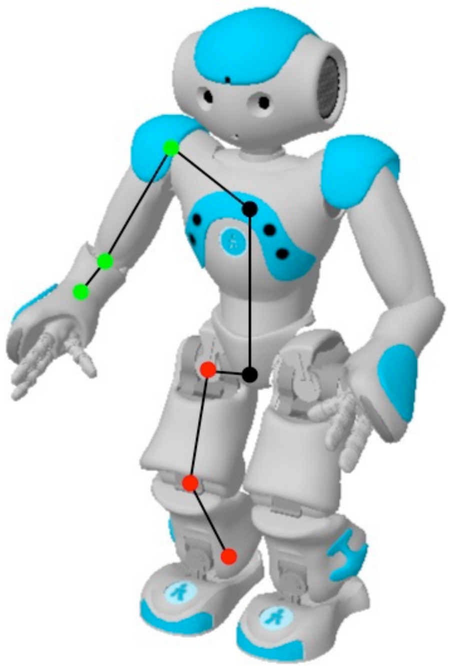
https://www.universal-robots.com/articles/ur/application-installation/dh-parameters-for-calculations-of-kinematics-and-dynamics/

Capability Maps

- Measures the **kinematic capabilities** of a robot, representing its workspace under certain quality measures
- Acceleration of online motion planning, due to **pre-calculation** of the capability measures
- **Common measures**:
  - Reachability
  - Manipulability
  - Robot base placeability (inverse reachability)
- Capability maps **always consist** of a reachability map

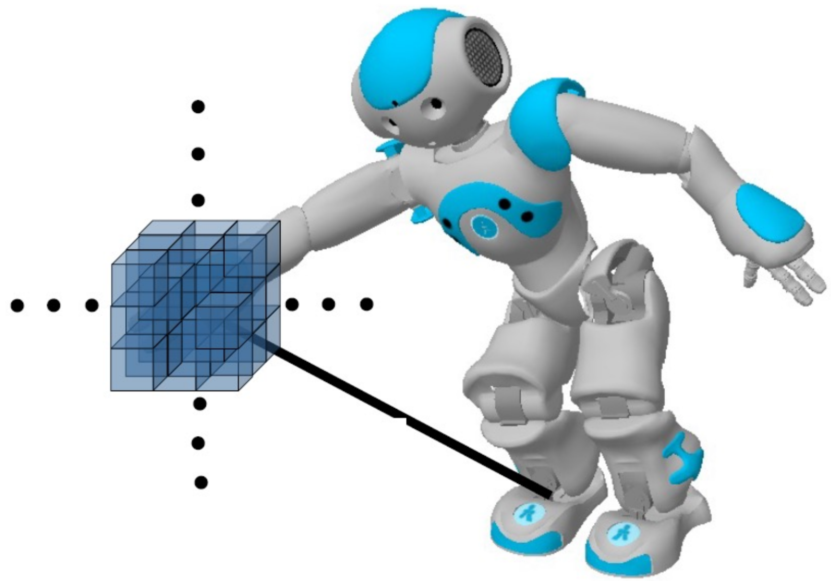
Reachability Map (RM)

- Constructed by systematic **sampling joint configurations** of a kinematic chain
- **Example:** Chain of joints between the right foot and the gripper link



Reachability Map (RM)

- FK to determine the **corresponding voxel** containing the end-effector pose

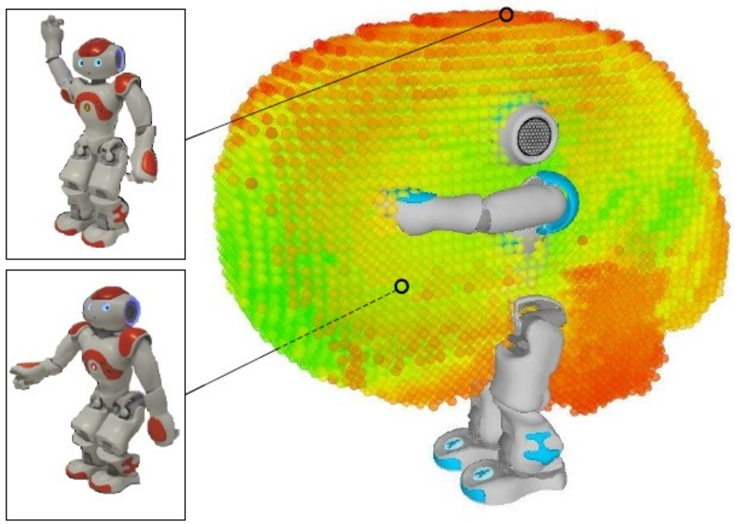


Reachability Map (RM)

- Configurations are added to the RM if they are **statically stable** and **self-collision free**
- Result: Representation of reachability, voxels contain configurations and **corresponding quality measures**
- Generating the RM is time-consuming, **but needs to be done only once offline**

Manipulability Measure

- Penalize configurations with limited maneuverability
- **Singular configurations:** Certain EE movements are not possible, i.e., **small desired changes** in EE poses lead to **large joint angle changes**
- **Consider:** Distance to singular configurations and joint limits, self-distance, ...



red=low
green=high

Inversion of the RM

- Invert the precomputed reachable workspace: **inverse reachability map (IRM)**
- Invert the FK transform for each configuration to get the **pose of the robot's base** (e.g., foot) **w.r.t. the EE frame**
- Determine the voxel in the IRM **containing the base pose**
- Store configurations and manipulability measures from the RM in the corresponding IRM voxels

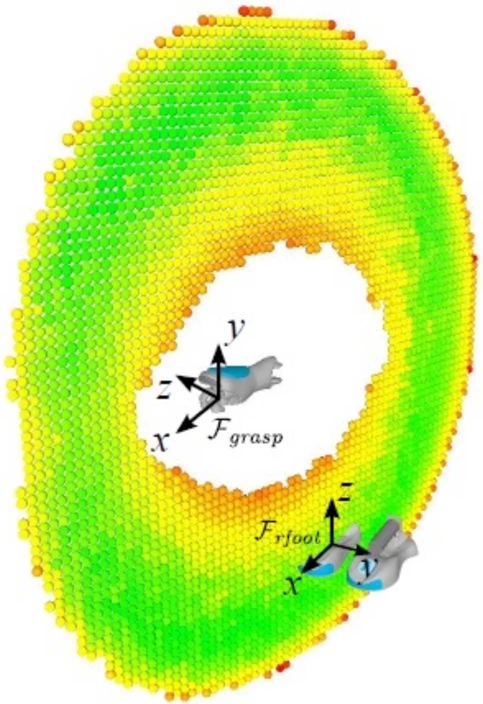
Inverse Reachability Map (IRM)

- The IRM represents the set of **potential base poses** relative to the EE frame
- Allows for selecting an optimal base pose for a given grasping target
- Can be pre-computed

red=low

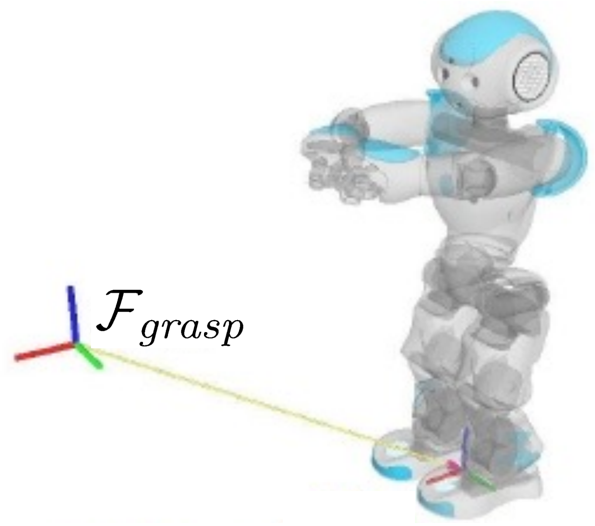
green=high

Cross section through the IRM showing potential feet locations



Determining the Optimal Stance Pose Given a Grasp Pose

- Given a desired 6D end-effector pose with transform $\mathcal{F}_{grasp}$
- How to determine the optimal stance pose?



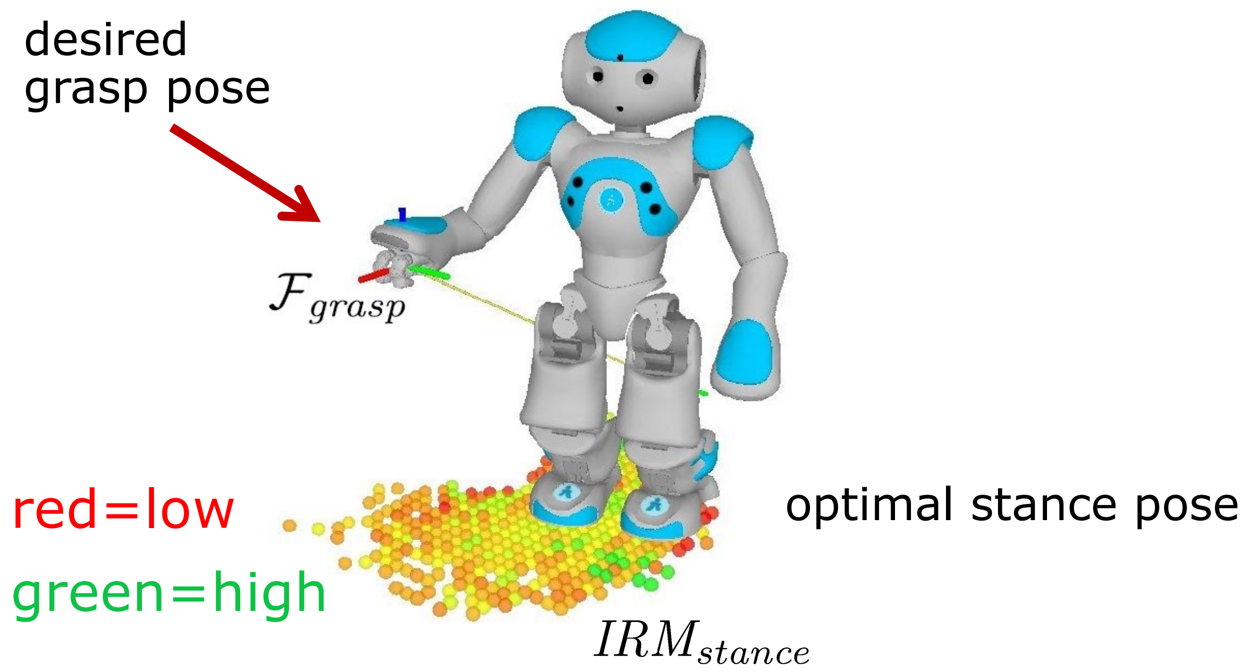
Determining the Optimal Stance Pose Given a Grasp Pose

- Transform the IRM and determine valid configurations of the feet on the ground
- Align the origin of the IRM with the grasp frame $\mathcal{F}_{grasp}$ to get the transformed IRM $tIRM$
- Intersect $tIRM$ with the floor plane *F*:

$$IRM_{floor} = tIRM \cap F$$

- Remove unfeasible configurations from IRM_{floor} to get IRM_{stance}

Determining the Optimal Stance Pose: Example



Select the optimal stance pose
from the voxel with the highest manipulability measure

Summary (1)

- Forward kinematics compute the end-effector pose given joint angles along the chain
- Inverse kinematics computes the joint angles so that the end-effector reaches a desired goal pose
- Several approaches for IK exist (analytical/numerical)
- Basic Jacobian IK technique iteratively adapts the joint angles to reach the end-effector goal pose
- Denavid-Hartenberg parameters reduce the joint and link describing parameters

Summary (2)

- Capability maps represent “how well” the robot can interact with the world
- Reachability maps represent reachable end-effector poses using FK
- Inverse RMs used to determine the optimal base pose for a desired end-effector pose
- Both computed offline only once

Literature

- Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least methods
  S.R. Buss, University of California, 2009
- Introduction to Robotics: Mechanics and Control
  John J. Craig, Pearson Prentice Hall, 2005
- Stance Selection for Humanoid Grasping Tasks by Inverse Reachability Maps
  F. Burget and M. Bennewitz,
  Proc. of the IEEE International Conference on Robotics & Automation (ICRA), 2015