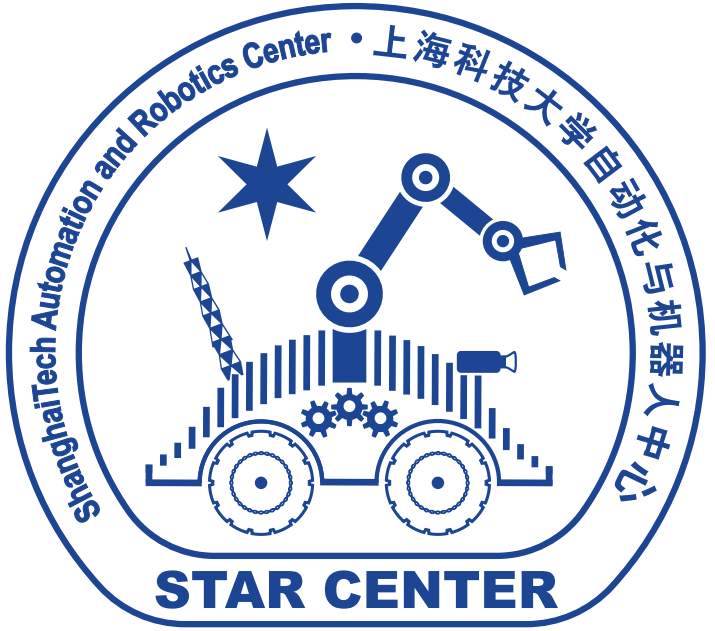
上海科技大学
ShanghaiTech University

CS289: **Mobile Manipulation Fall 2025**

Sören Schwertfeger

ShanghaiTech University



Outline

- Kinematics
- Planning

KINEMATICS

Kinematics

Forward Kinematics (angles to pose) (it is straight-forward -> easy)

What you are given: The constant arm parameters (e.g. DH parameters)
The angle of each joint

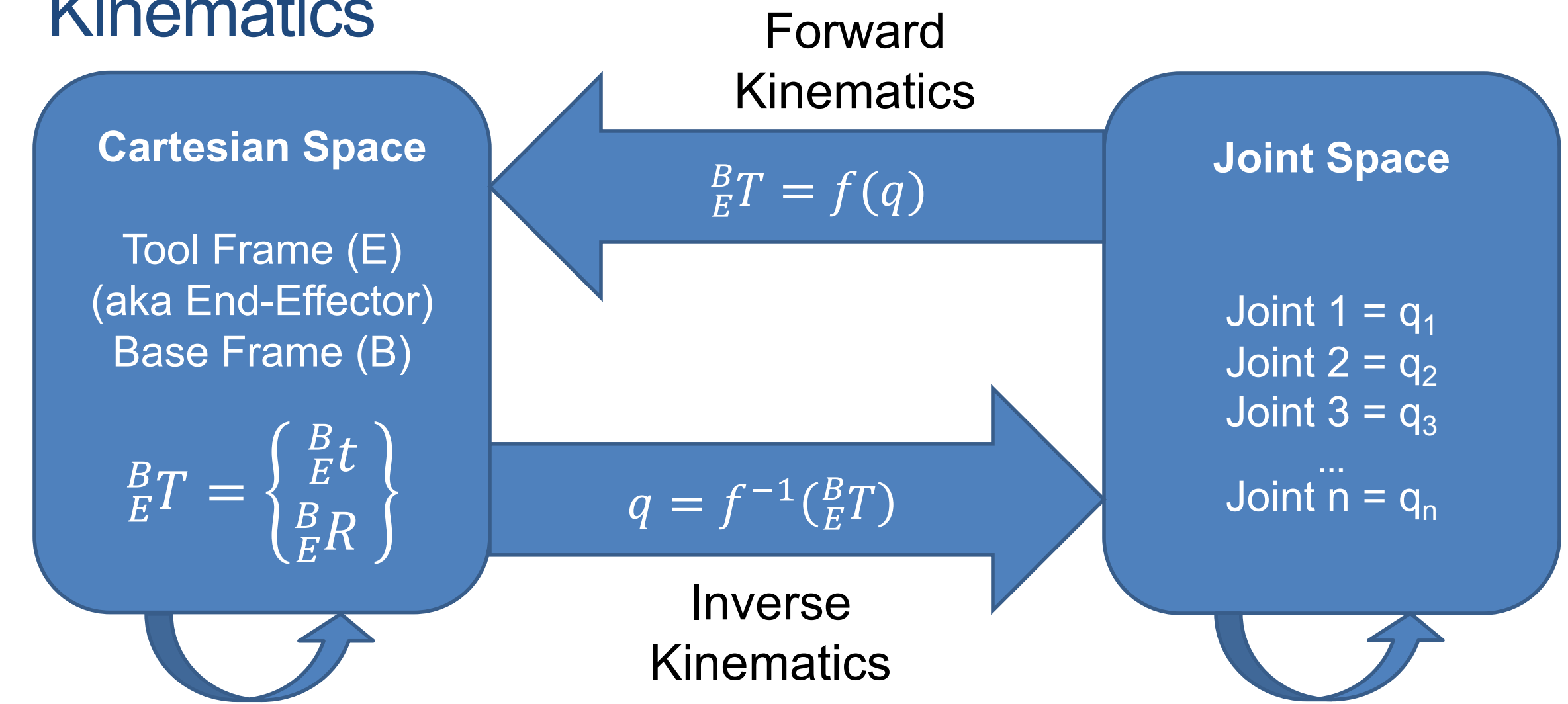
What you can find: The pose of any point (i.e. it's (x, y, z) coordinates & 3D orientation)

Inverse Kinematics (pose to angles) (more difficult)

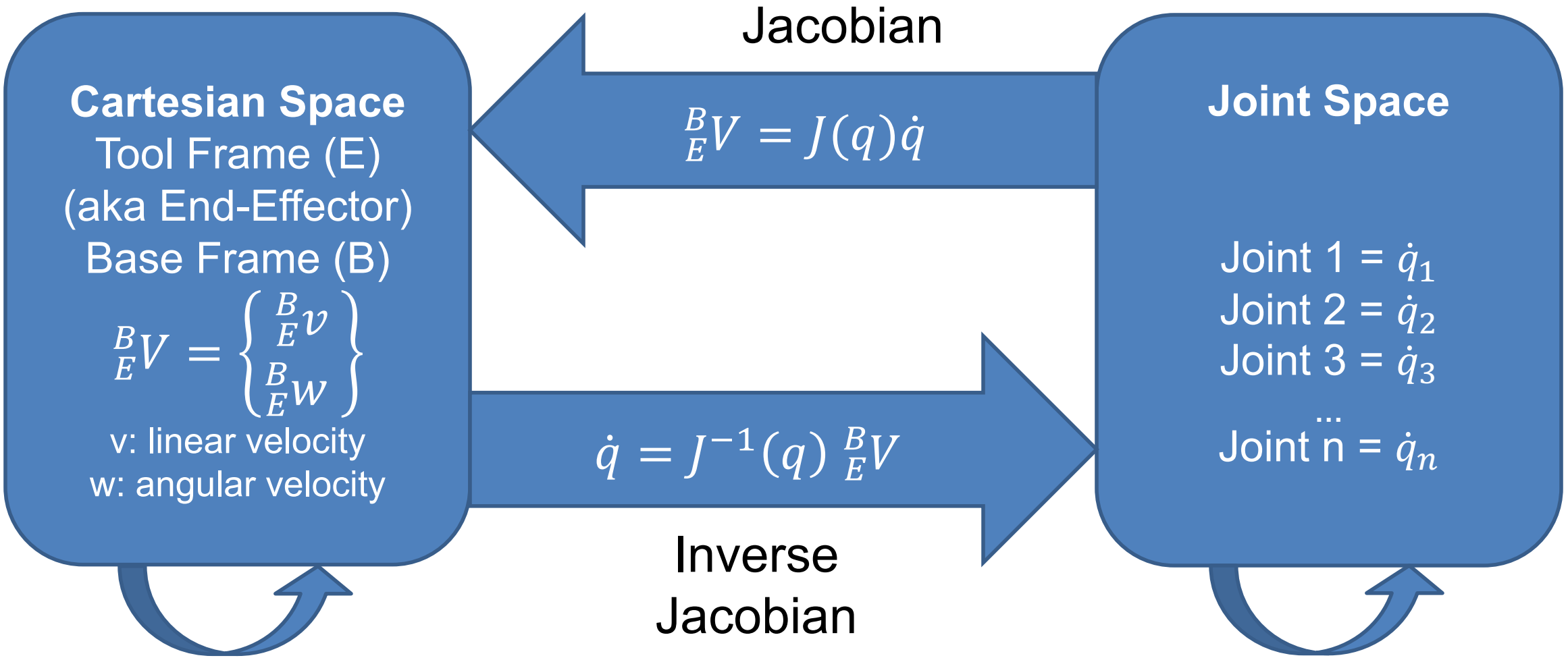
What you are given: The constant arm parameters (e.g. DH parameters)
The pose of some point on the robot

What you can find: The angles of each joint needed to obtain that pose

Kinematics


Forward Kinematics
$^{B}_{E}T = f(q)$
Cartesian Space
Tool Frame (E)
(aka End-Effector)
Base Frame (B)
$^{B}_{E}T = \begin{Bmatrix} ^{B}_{E}t \\ ^{B}_{E}R \end{Bmatrix}$
Joint Space
Joint 1 = q_1
Joint 2 = q_2
Joint 3 = q_3
...
Joint n = q_n
$q = f^{-1}(^{B}_{E}T)$
Inverse Kinematics
Rigid body transformation
Between coordinate frames
Linear algebra

Kinematics: Velocities



FORWARD KINEMATICS

Forward Kinematics

- DH Parameters to pose:

$$A = R_z(\Theta_j)\, t_z(d_j)\, t_x(a_j)\, R_x(\alpha_j) \quad \Rightarrow$$

$${}^{B}_{E}T = R_z(\Theta_1)\, t_z(d_1)\, t_x(a_1)\, R_x(\alpha_1) \quad R_z(\Theta_2)\, t_z(d_2)\, t_x(a_2)\, R_x(\alpha_2) \quad \cdots R_z(\Theta_N)\, t_z(d_N)\, t_x(a_N)\, R_x(\alpha_N)$$

- For revolute joints: only Θ varies – other three are constant for that robot
- For prismatic joints: only d varies – other three are constant for that robot

- For non-holonomic (mobile) robots: Chain the time-varying motion: Odometry

INVERSE KINEMATICS (IK)

Inverse Kinematics (IK)

- Given end effector pose, compute required joint angles
- In simple case, analytic solution exists
  - Use trig, geometry, and algebra to solve

- Possible Problems of Inverse Kinematics
  - Multiple solutions
  - Infinitely many solutions
  - No solutions
  - No closed-form (analytical solution)

- Generally (more DOF) difficult
  - Use Newton’s method

- Analytic solution of 2-link inverse kinematics

$$x^2 + y^2 = a_1^2 + a_2^2 - 2a_1a_2\cos(\pi - \theta_2)$$

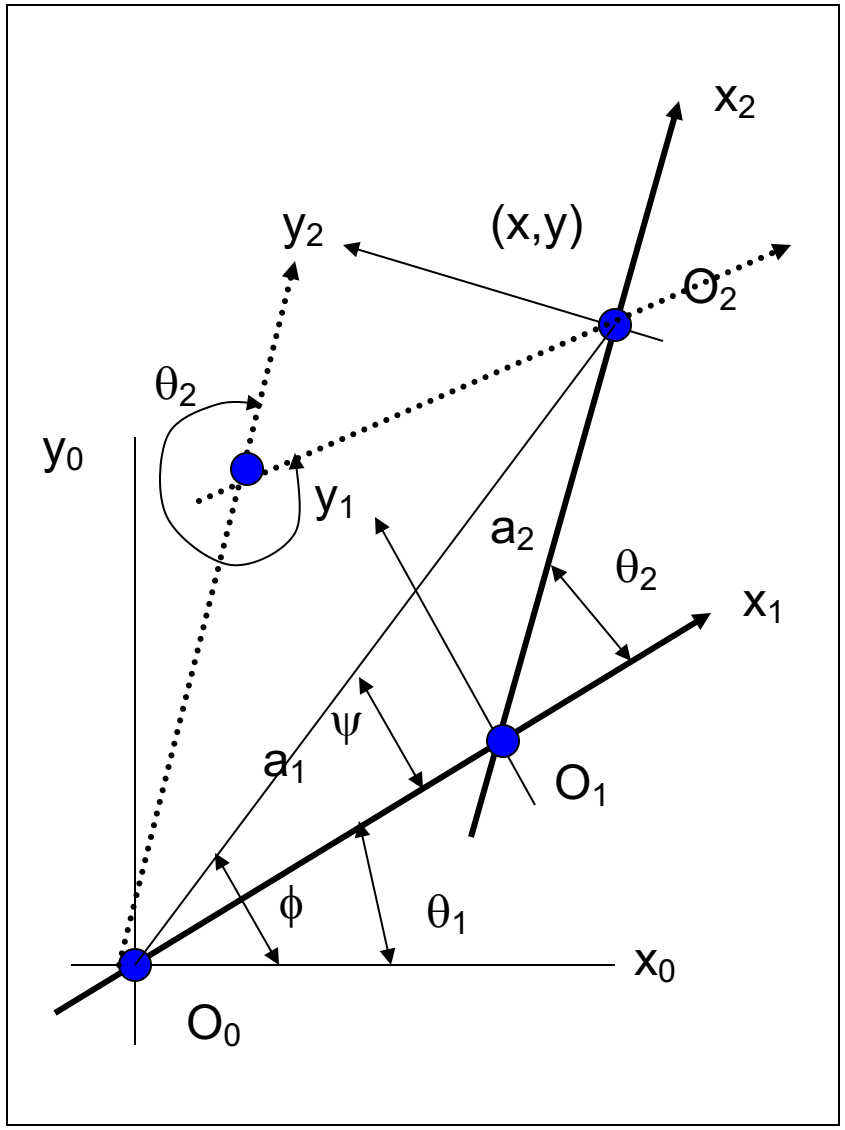
$$\cos\theta_2 = \frac{x^2 + y^2 - a_1^2 - a_2^2}{2a_1a_2}$$

for greater accuracy

$$\tan^2\frac{\theta_2}{2} = \frac{1-\cos\theta}{1+\cos\theta} = \frac{2a_1a_2 - x^2 - y^2 + a_1^2 + a_2^2}{2a_1a_2 + x^2 + y^2 - a_1^2 - a_2^2}$$

$$= \frac{\left(a_1^2 + a_2^2\right)^2 - \left(x^2 + y^2\right)}{\left(x^2 + y^2\right) - \left(a_1^2 - a_2^2\right)^2}$$

$$\theta_2 = \pm 2\tan^{-1}\sqrt{\frac{\left(a_1^2 + a_2^2\right)^2 - \left(x^2 + y^2\right)}{\left(x^2 + y^2\right) - \left(a_1^2 - a_2^2\right)^2}}$$



- Two solutions: elbow up & elbow down

Iterative IK Solutions

- Often analytic solution is infeasible
- Use Jacobian
- Derivative of function output relative to each of its inputs
- If y is function of three inputs and one output

$$y = f(x_1, x_2, x_3)$$

$$\delta y = \frac{\delta f}{\partial x_1} \cdot \delta x_1 + \frac{\delta f}{\partial x_2} \cdot \delta x_2 + \frac{\delta f}{\partial x_3} \cdot \delta x_3$$

- Represent Jacobian J(X) as a 1x3 matrix of partial derivatives

Jacobian

- In another situation, end effector has 6 DOFs and robotic arm has 6 DOFs
- $f(x_1, \ldots, x_6) = (x, y, z, r, p, y)$
- Therefore J(X) = 6x6 matrix

$$\begin{bmatrix} \frac{\partial f_x}{\partial x_1} & \frac{\partial f_y}{\partial x_1} & \frac{\partial f_z}{\partial x_1} & \frac{\partial f_r}{\partial x_1} & \frac{\partial f_p}{\partial x_1} & \frac{\partial f_y}{\partial x_1} \\ \frac{\partial f_x}{\partial x_2} & & & & & \\ \frac{\partial f_x}{\partial x_3} & & & & & \\ \frac{\partial f_x}{\partial x_4} & & & & & \\ \frac{\partial f_x}{\partial x_5} & & & & & \\ \frac{\partial f_x}{\partial x_6} & & & & & \end{bmatrix}$$

Jacobian Transpose Method

- Relates velocities in parameter space to velocities of outputs

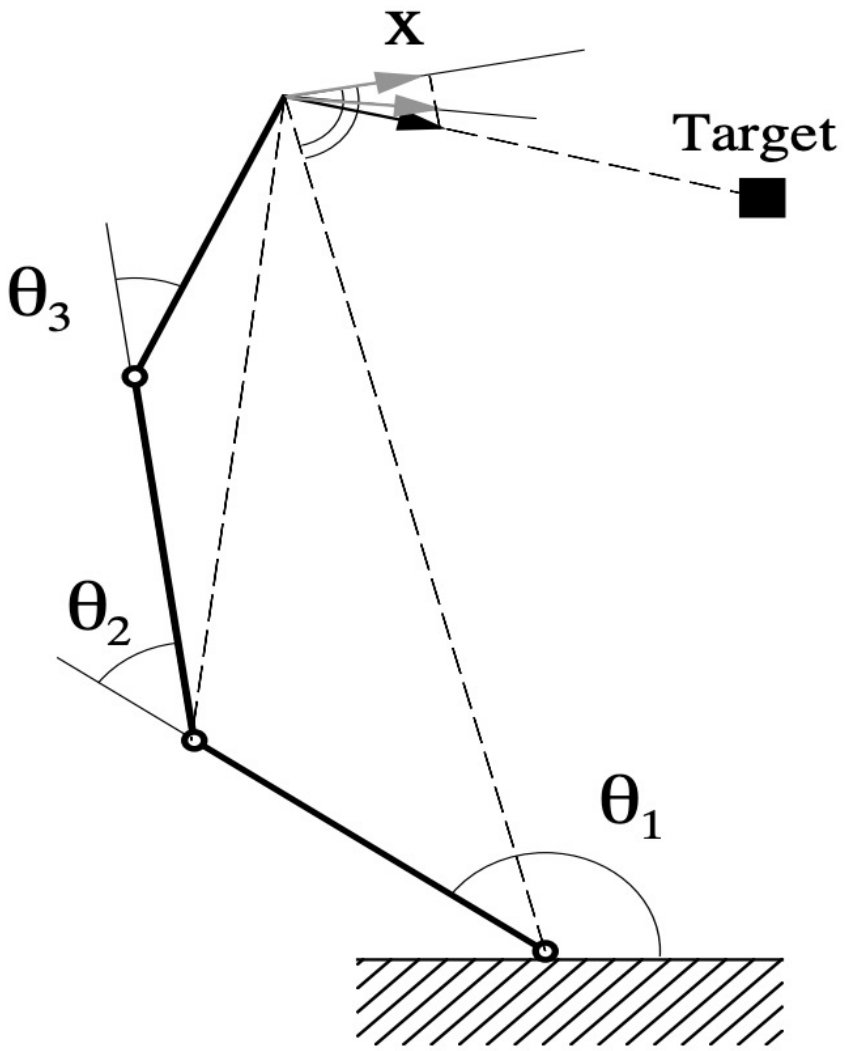
$$\dot{Y} = J(X) \cdot \dot{X}$$

- If we know $Y_{current}$ and $Y_{desired}$, then we subtract to compute $\dot{Y}$

- Invert Jacobian and solve for $\dot{X}$:

$$\dot{X} = \alpha J^T(X) \cdot \Delta Y$$

- Projects difference vector ΔY to those dimensions which can reduce it the most
- Disadvantages:
  - Needs many iterations until convergence in certain configurations (e.g., Jacobian has very small coefficients)
  - Unpredictable joint configurations



Stefan Schaal; USC, Max Planck

Iterative Solution to Inverse Kinematics

- Only holds for high sampling rates or low Cartesian velocities
- “a local solution” that may be “globally” inappropriate
- Problems with singular postures
- Can be used in two ways:
  - As an instantaneous solutions of “which way to take “
  - As an “batch” iteration method to find the correct configuration at a target

ROBOT DESCRIPTION

URDF+Xacro

Unified Robot Description Format (**URDF**) is an XML format for representing a robot model.

It enable to describe kinematic, visual and dynamic properties of a manipulator.

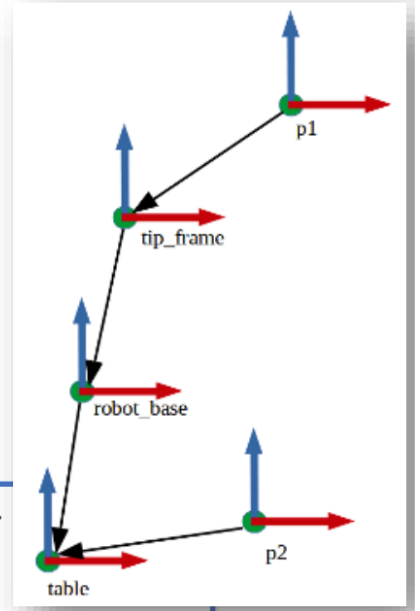
http://wiki.ros.org/urdf

Xacro is an XML macro language: enable construction of shorter and more readable XML files by using macros that expand to larger XML expressions.

http://wiki.ros.org/xacro

ROS provides parsing tools for reading and checking URDF files:

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



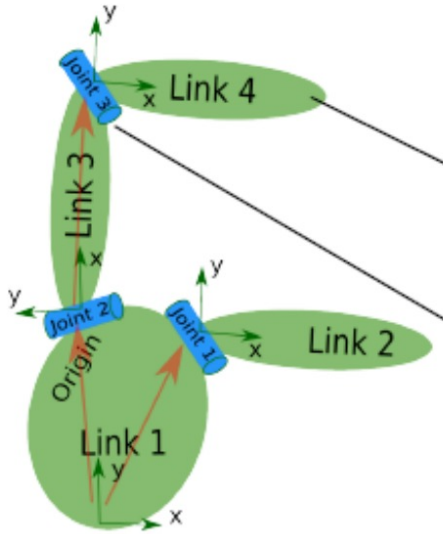

```
<robot name="test_robot">
  <link name="link1" />
  <link name="link2" />
  <link name="link3" />
  <link name="link4" />

  <joint name="joint1" type="continuous">
    <parent link="link1"/>
    <child link="link2"/>
  </joint>
  <joint name="joint2" type="continuous">
    <parent link="link1"/>
    <child link="link3"/>
  </joint>
  <joint name="joint3" type="continuous">
    <parent link="link3"/>
    <child link="link4"/>
  </joint>
</robot>
```

Diego Dall'Alba https://atlas-itn.eu/wp-content/uploads/Presentations/NTA3/univr2_intro_ros_B1.pdf

URDF Simple Example

- 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 Joint types

1. Fixed: rigid connection
2. Revolute: 1D rotation
3. Continuous: unlimited revolute
4. Prismatic: 1D translation
5. Planar: 2D translation
6. Floating: unlimited 6D

Following material partly from: https://ocw.tudelft.nl/courses/hello-real-world-ros-robot-operating-system/

Standardization in ROS

- ROS Enhancement Proposals (REPs)
  - https://ros.org/reps/rep-0000.html

- REP 103 - *Standard Units of Measure and Coordinate Conventions*
- Right-handed coordinate system
- X+ (forward) and Y+ (left) → Z+ (up)
- SI units:
  - Lengths: meters
  - Angles: radians

Limitations of URDF (1)

1. Robot descriptions cannot be changed
    1. Robot description read only once from parameter server
    2. No standardized way to notify consumers of changes
    3. Risk of *desynchronization* of nodes

2. Only tree structures
    1. Joints have only a single parent and child
    2. Only acyclic, directed graphs (or: trees) can be modeled
    3. Real-world impact: dual-arm manipulation, parallel grippers, etc

Limitations of URDF (2)

3. No `<sensor></sensor>`
   1. Sensor meta-data cannot be incorporated into URDF directly
   2. Alternatives exist, but data is distributed
   3. Diminishes value of URDF as *central* robot description

4. Low reusability of URDFs
   1. Only a single `<robot>` tag in a URDF
   2. No support for import of remote files
   3. Composite scenes have to be merged manually
   4. No way to compose multiple URDFs

Solution: XACRO (XML Macros)

- Programmatic URDF generation
- Templates
- Parameters

arm_macro.xacro

```
<xacro:macro name="arm" params="parent arm_name">
  <link name="${arm_name}_link_1" />
  <joint name="${arm_name}_joint_1" type="..">
    <parent link="${parent}" />
    <child link="${arm_name}_link_1" />
  </joint>
</xacro:macro>
```

robot.xacro

```
[..]
  <link name="torso" />
[..]
  <xacro:arm parent="torso" arm_name="left" />
  <xacro:arm parent="torso" arm_name="right" />
[..]
```

XACRO

- Import macros from other files
- Parameterize templates
- Composite robots & scenes easier:
  - Import macro from file
  - Invoke macro

- Disadvantage: XACRO not directly compatible with URDF
- Transformation needed
- Command:

```
$ rosrun xacro xacro /path/to/robot.xacro > robot.urdf
```

- Also checks for a valid XACRO file

Validate URDFs

- Validate URDFs: check_urdf
- checks syntax (ie: legal combinations of URDF keywords)
- not semantics (ie: meaning or real-world correctness)
- Command:

tiny_robot.urdf

```
<robot name="tiny_robot">
  <link name="link_1" />
  <link name="link_2" />
  <joint name="joint_1" type="continuous">
    <parent link="link_1" />
    <child link="link_3" />
  </joint>
</robot>
```

```
$ check_urdf tiny_robot.urdf
```

```
Error: Failed to build tree:
  child link [link_3] of joint [joint_1] not found
ERROR: Model Parsing the xml failed
```

2. state_publisher node

2.1 ROS API

2.1.1 Subscribed topics

`joint_states` (sensor_msgs/JointState)

joint position information

2.1.2 Parameters

`robot_description` (`urdf map`)

The urdf xml robot description. This is accessed via `urdf_model::initParam`

`tf_prefix` (`string`)

Set the tf prefix for namespace-aware publishing of transforms. See tf_prefix for more details.

`publish_frequency` (`double`)

Publish frequency of state publisher, default: 50Hz.

`ignore_timestamp` (`bool`)

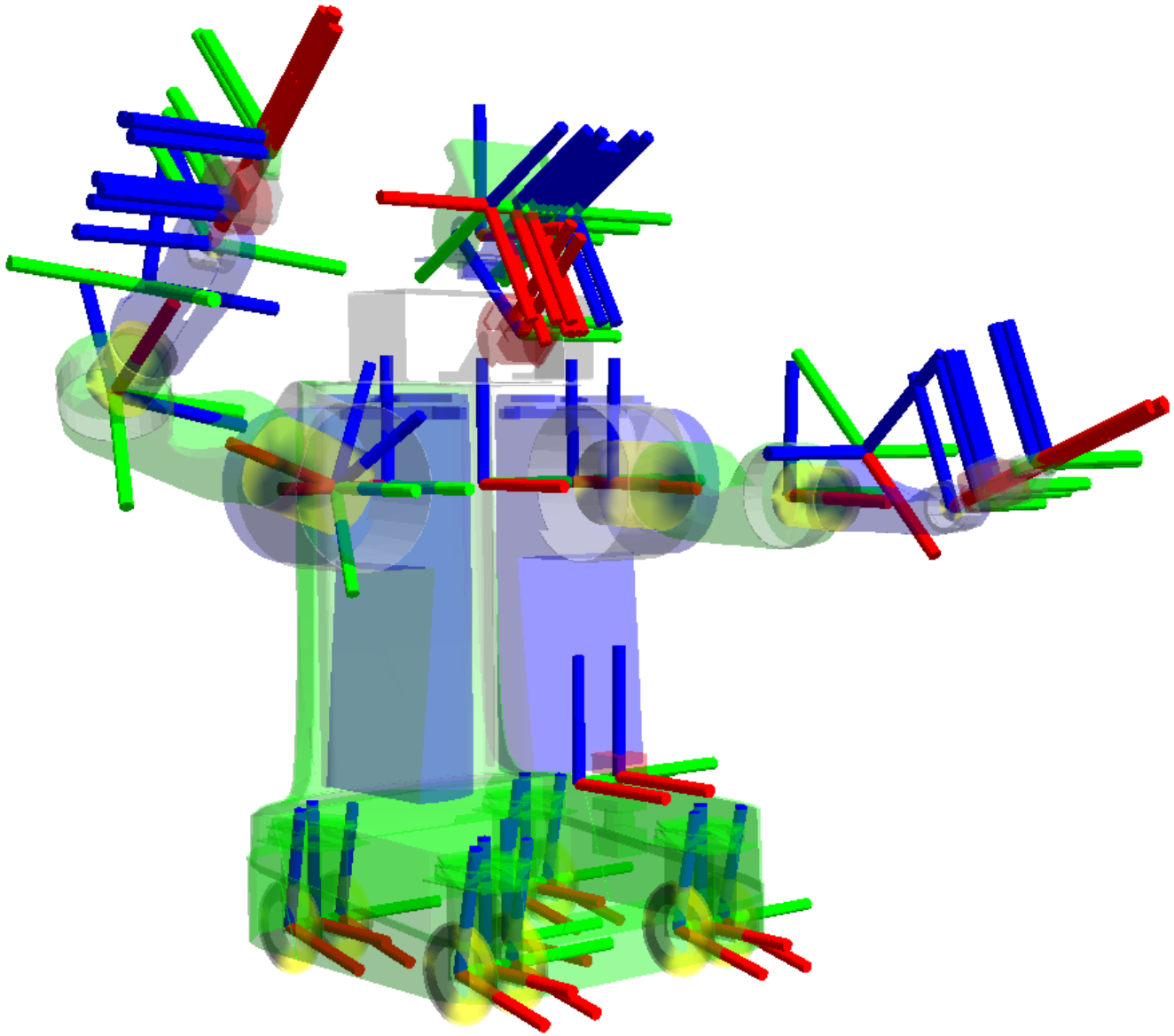
If true, ignore the publish_frequency and the timestamp of joint_states and publish a tf for each of the received joint_states. Default is "false".

`use_tf_static` (`bool`)

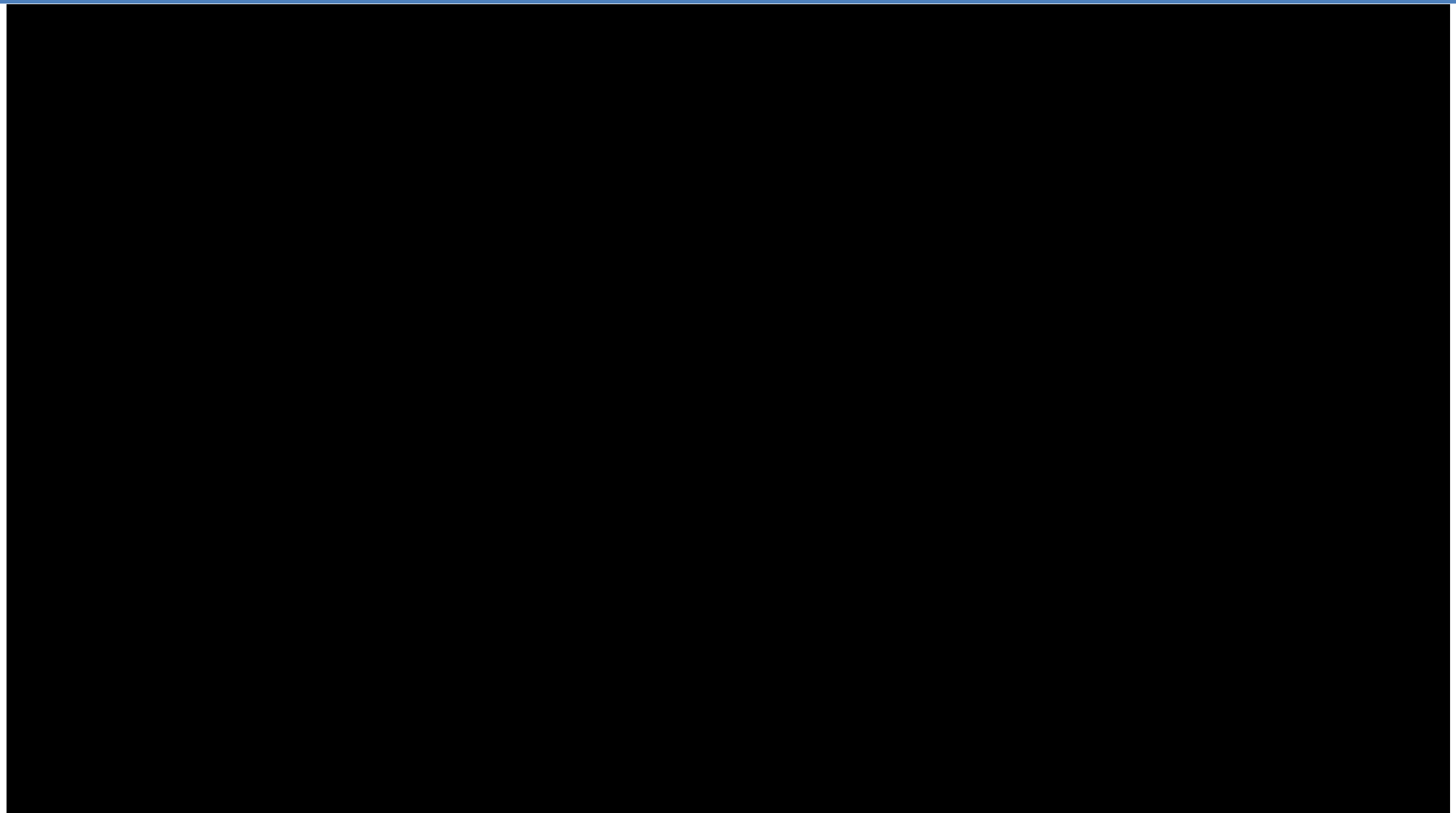
Set whether to use the /tf_static latched static transform broadcaster. Default: true.

ROS: 3D Transforms : TF

- http://wiki.ros.org/tf
- http://wiki.ros.org/tf/Tutorials



PICK & PLACE



Kinematic Problems for Manipulation

- Reliably position the tip - go from one position to another position
- Don’t hit anything, avoid obstacles
- Make smooth motions
  - at reasonable speeds and
  - at reasonable accelerations
- Adjust to changing conditions -
  - i.e. when something is picked up *respond to the change in weight*

Planning Problem

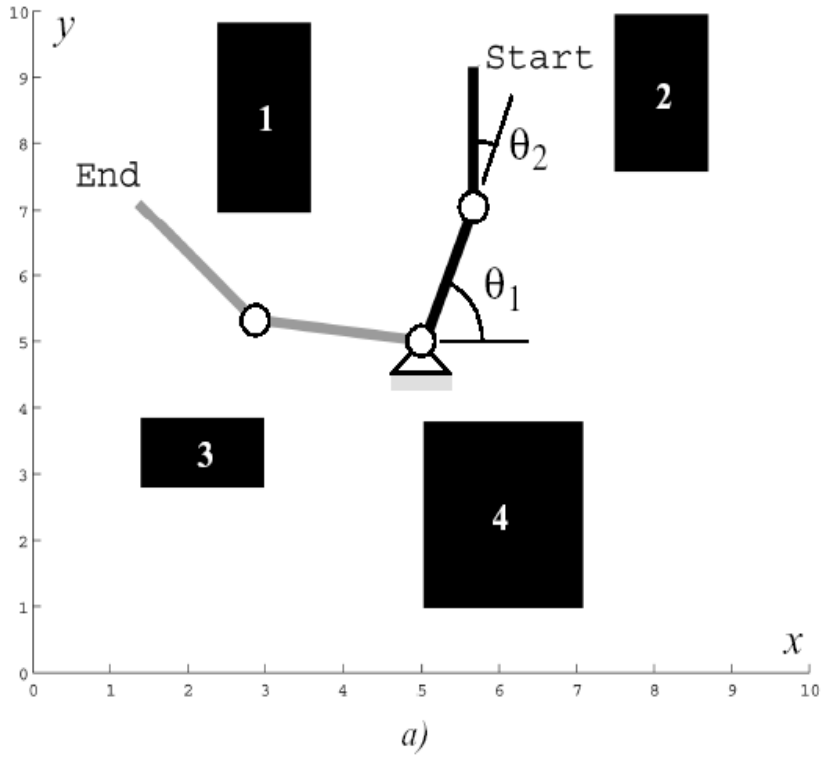
- (Arm) Pose: Set of join values
- (Arm) Trajectory:
  - Given a start pose and an end pose
  - A list of intermediate poses
  - That should be reached one after the other
  - With associated (desired) velocities and accelerations (maxima)
    - Without time (without velocity and acceleration): path! So:
    - Path: poses; Trajectory: poses with speeds (and maybe accelerations)
- Constrains:
  - Don’t collide with yourself
  - Don’t collide with anything else (except: fingers with the object to manipulate!!!)
  - Additional possible constrains:
    - Maximum joint velocities or accelerations
    - Keep global orientation of a joint (often end-effector) within certain boundaries

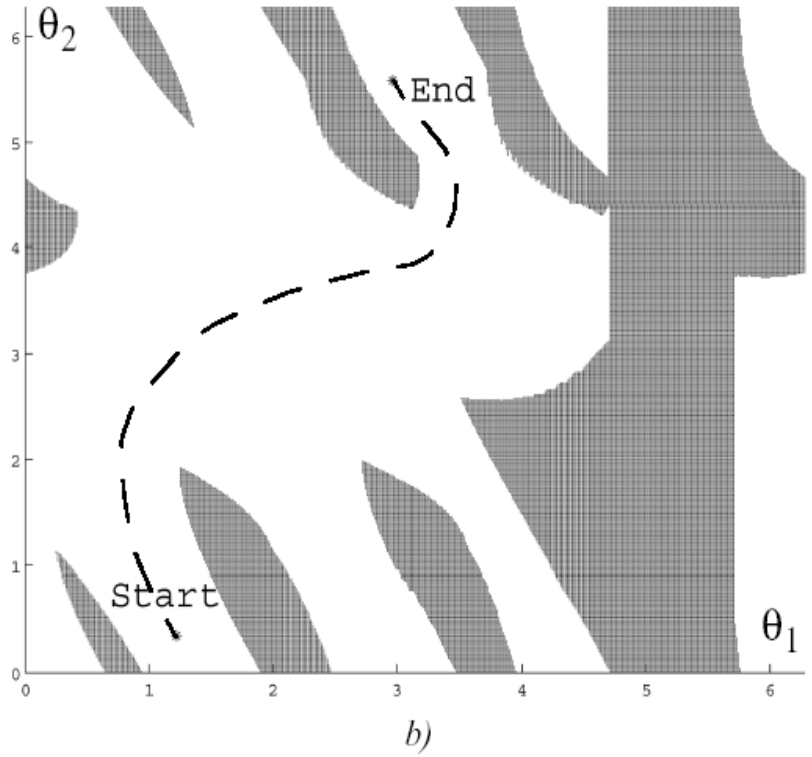
Planning Problem cont.

- Often the goal specified in Cartesian space (not joint space)
- => use IK to get joint space
- => often multiple (even infinitely many) solutions
  - Which one select for planning?
  - Plan for several solutions and select best!?

Work Space (Map) → Configuration Space

- State or configuration q can be described with k values q_i



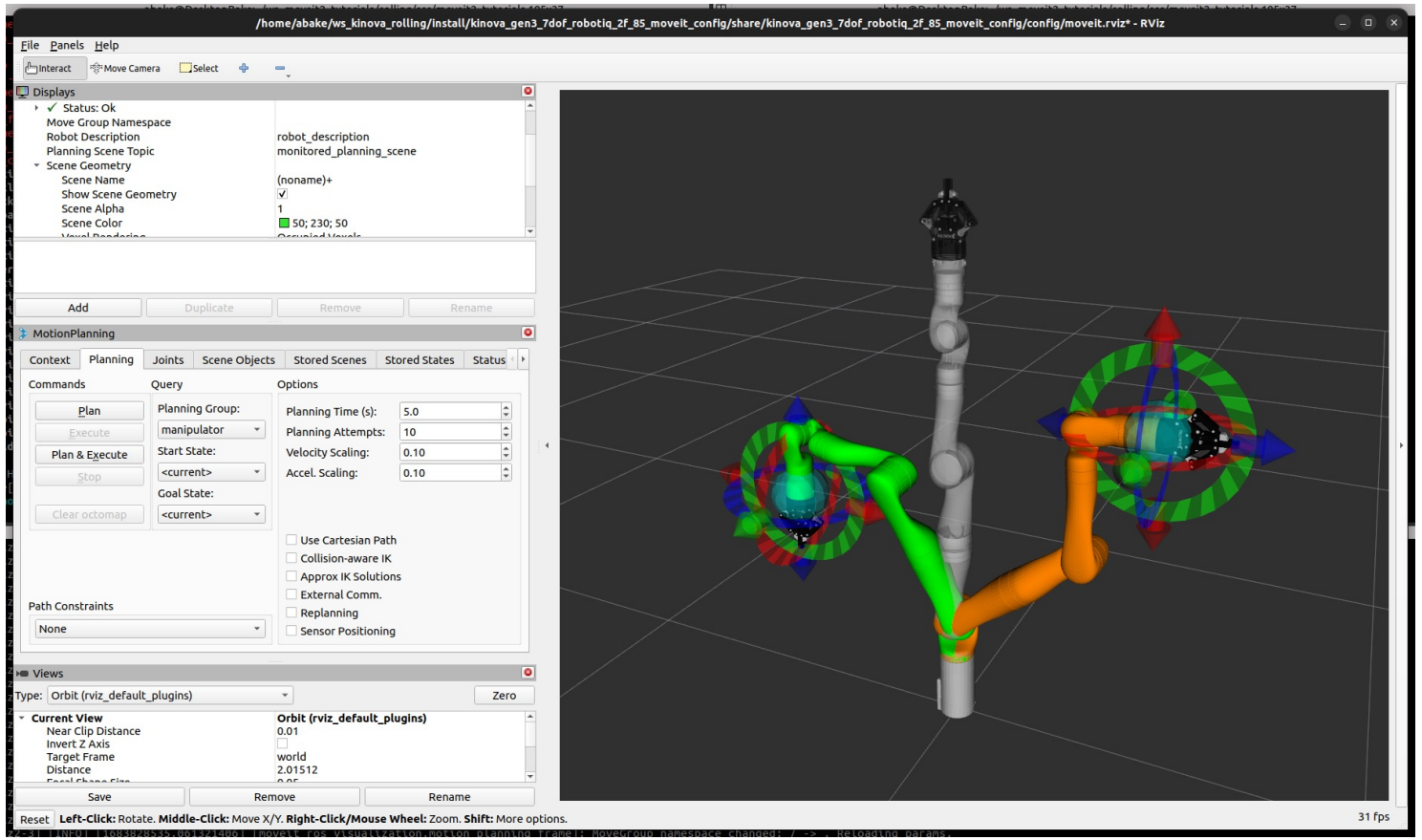


Work Space
Dimension depends on map
Dimension – typically 2D or 3D

Configuration Space:
the dimension of this
space is equal to the Degrees of Freedom (DoF)
of the robot

MOVEIT

MoveIt!



- Software to Control & Plan with robot arms
- Part of ROS
- https://moveit.picknik.ai/
- First let’s explore how to tell MoveIt! what kind of arm you have…

System Architecture

https://moveit.picknik.ai/main/doc/concepts/concepts.html