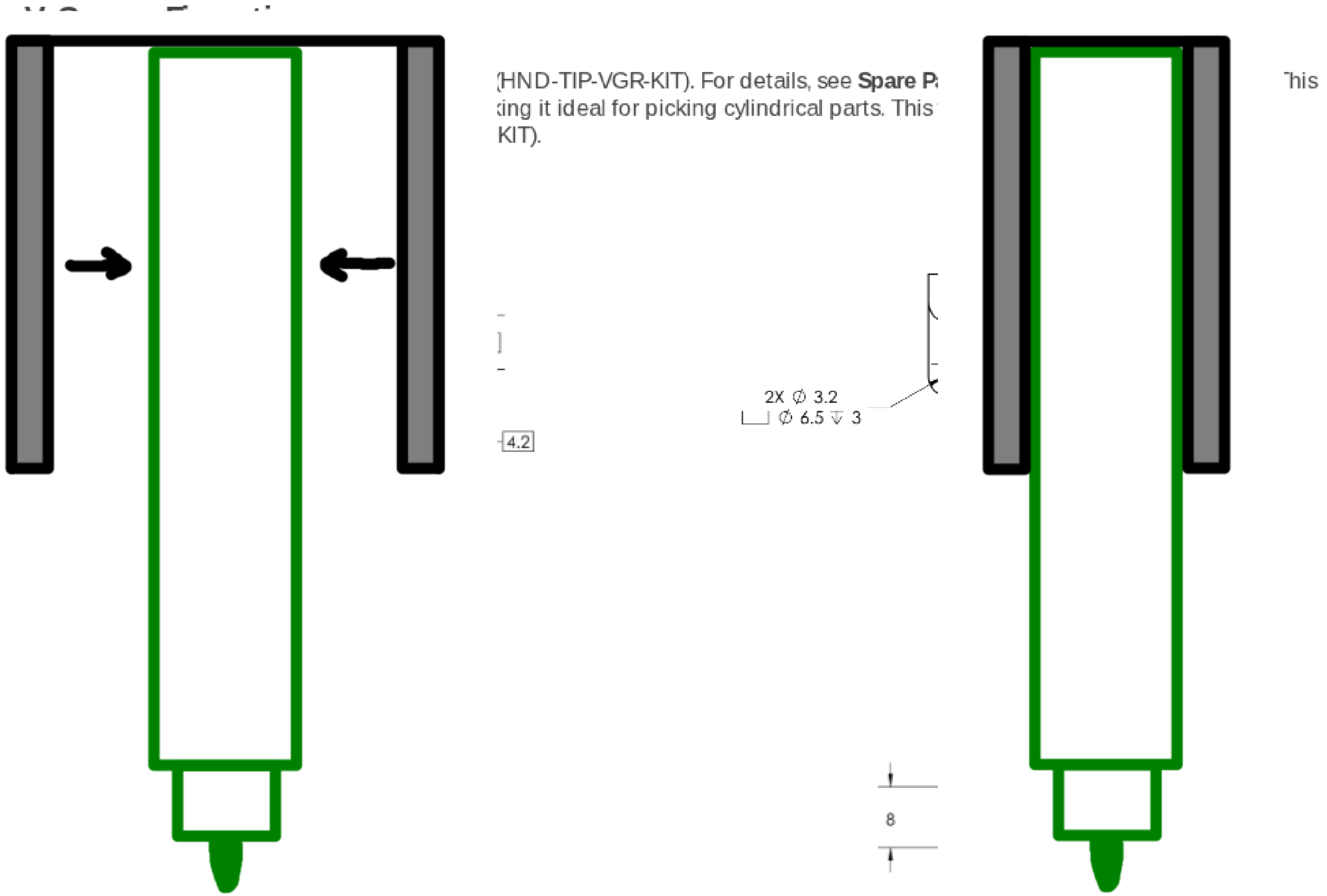
Robot arm

roadmap

20 / 02 / 2026

Cédric Mariéthoz
Pierre-Yves Savioz
Nathan Antonietti

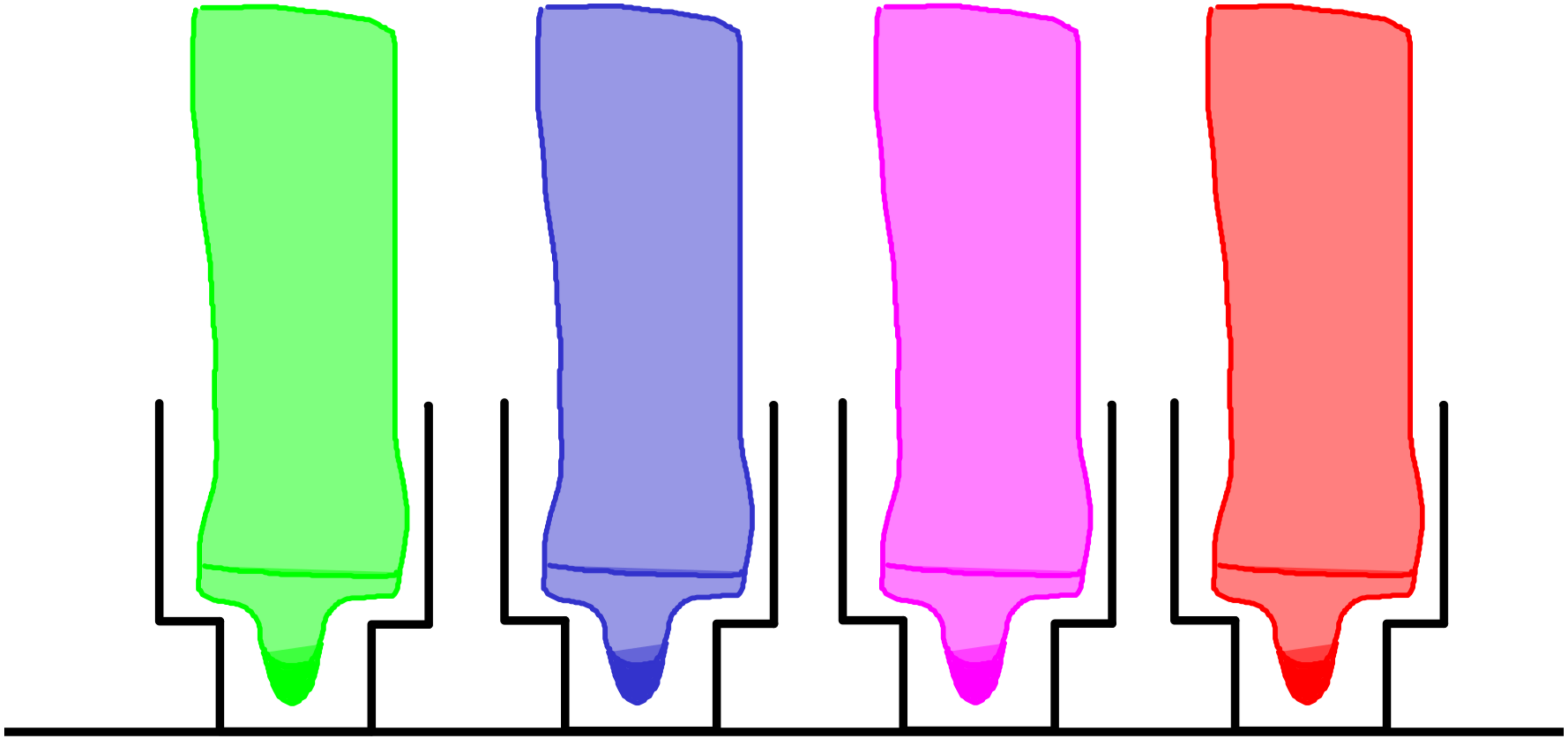
Gripper Fingers



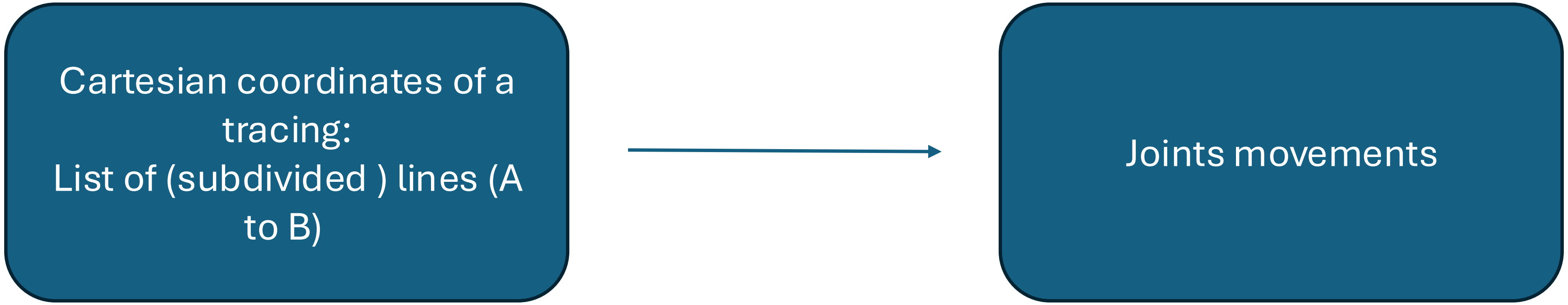
(HND-TIP-VGR-KIT). For details, see **Spare P** This
xing it ideal for picking cylindrical parts. This
KIT).

Fig. 5-9: Grooved fingertip

Pen Support



Robot Usage pipeline



Cartesian coordinates of a tracing:
List of (subdivided) lines (A to B)

Joints movements

The ur3e robotic arm

- 6 DOF arm with Hand-E

- In the Joints space:

J1 = { θ1, θ2, θ3, θ4, θ5, θ6}

J2 = {…. }

Python Lib:

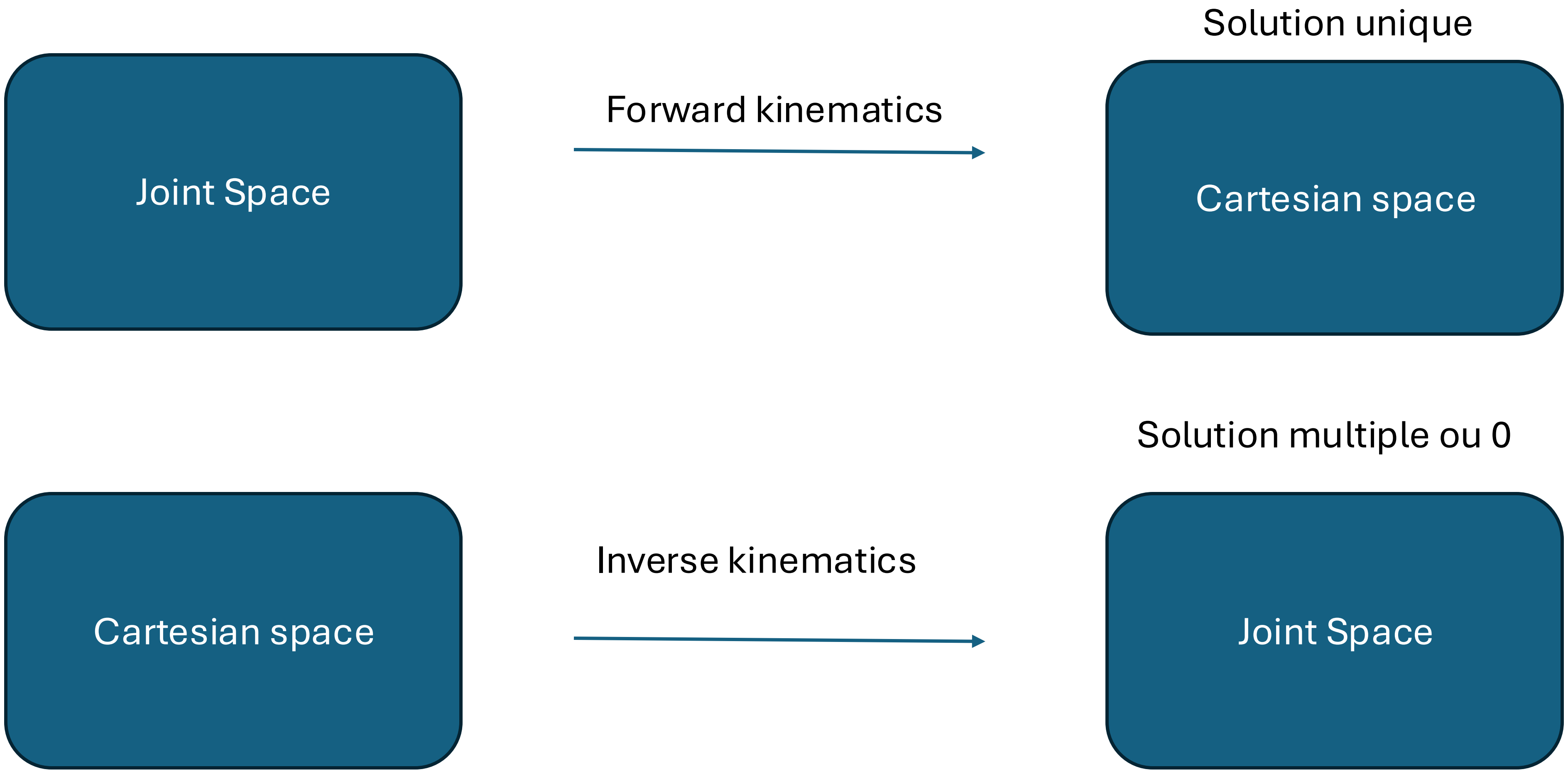
- movej

- In the cartesian space:
- P1 = {x, y, z, rx, ry, rz}
- P2 = …

Python lib:

- movel

Transformations



Solution unique

Joint Space

Forward kinematics

Cartesian space

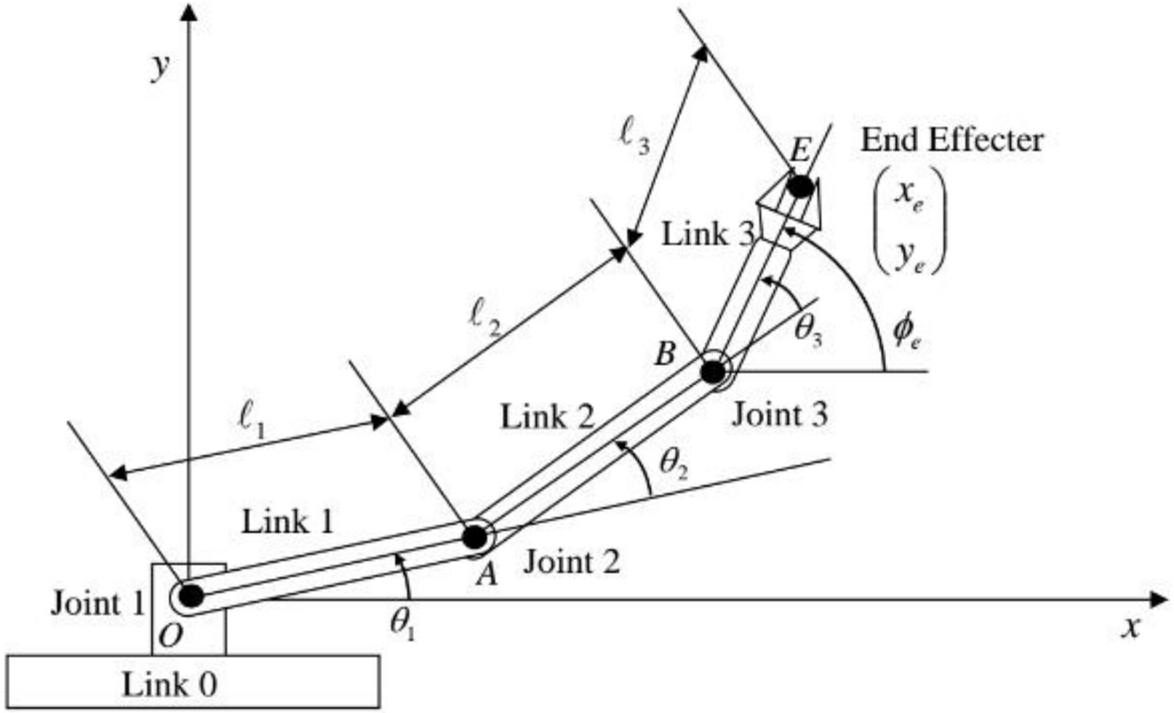
Solution multiple ou 0

Cartesian space

Inverse kinematics

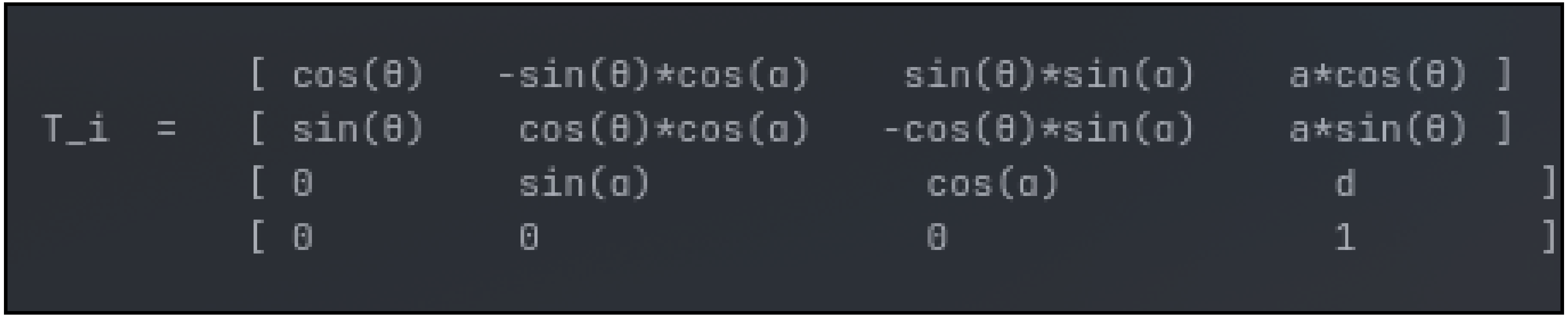
Joint Space

Forward kinematics

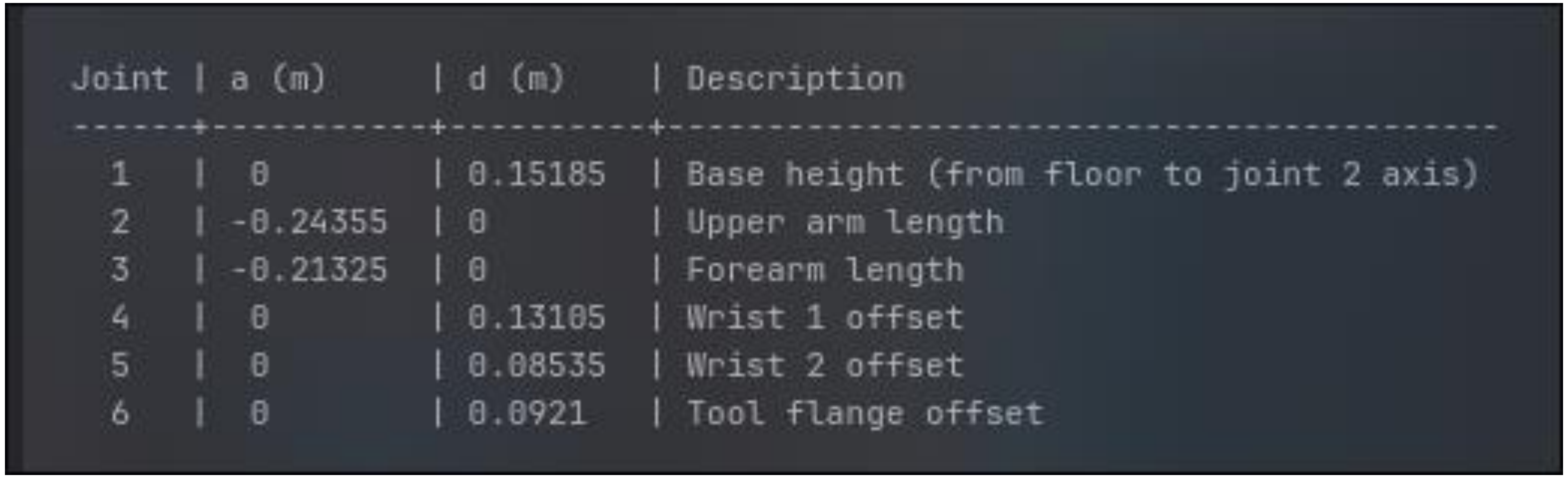


https://realitybytes.blog/2017/06/16/forward-and-inverse-kinematics-an-introduction/

Forward kinematics

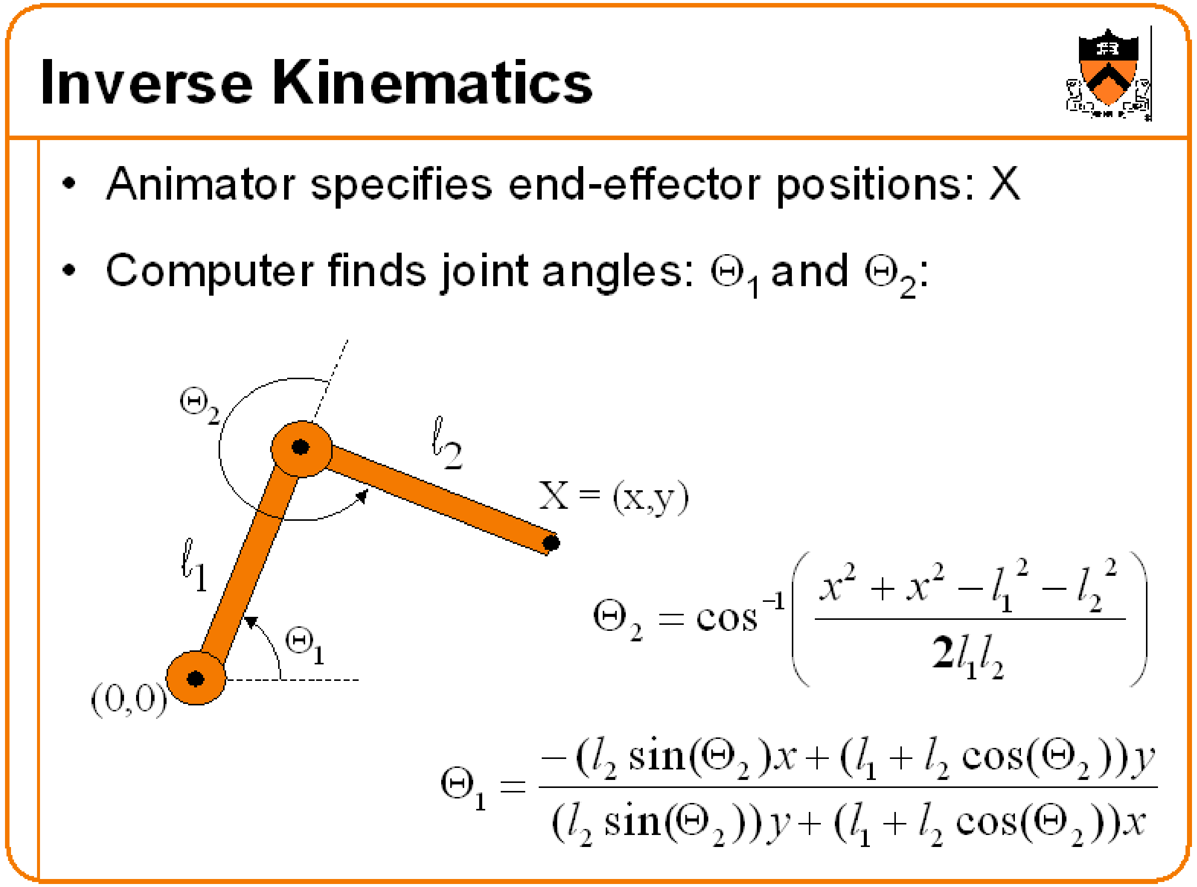


```
         [ cos(θ)   -sin(θ)*cos(α)    sin(θ)*sin(α)    a*cos(θ) ]
T_i  =   [ sin(θ)    cos(θ)*cos(α)   -cos(θ)*sin(α)    a*sin(θ) ]
         [ 0         sin(α)            cos(α)            d        ]
         [ 0         0                 0                 1        ]
```



```
Joint | a (m)     | d (m)     | Description
------+-----------+-----------+------------------------------------------
  1   |  0        | 0.15185   | Base height (from floor to joint 2 axis)
  2   | -0.24355  | 0         | Upper arm length
  3   | -0.21325  | 0         | Forearm length
  4   |  0        | 0.13105   | Wrist 1 offset
  5   |  0        | 0.08535   | Wrist 2 offset
  6   |  0        | 0.0921    | Tool flange offset
```

Inverse kinematics



Inverse Kinematics

- Animator specifies end-effector positions: X
- Computer finds joint angles: Θ_1 and Θ_2:

$$\Theta_2 = \cos^{-1}\left(\frac{x^2 + x^2 - l_1{}^2 - l_2{}^2}{\mathbf{2}l_1 l_2}\right)$$

$$\Theta_1 = \frac{-(l_2 \sin(\Theta_2)x + (l_1 + l_2 \cos(\Theta_2))y}{(l_2 \sin(\Theta_2))y + (l_1 + l_2 \cos(\Theta_2))x}$$

https://www.cs.princeton.edu/courses/archive/fall99/cs426/lectures/kinematics/sld008.htm

Movement

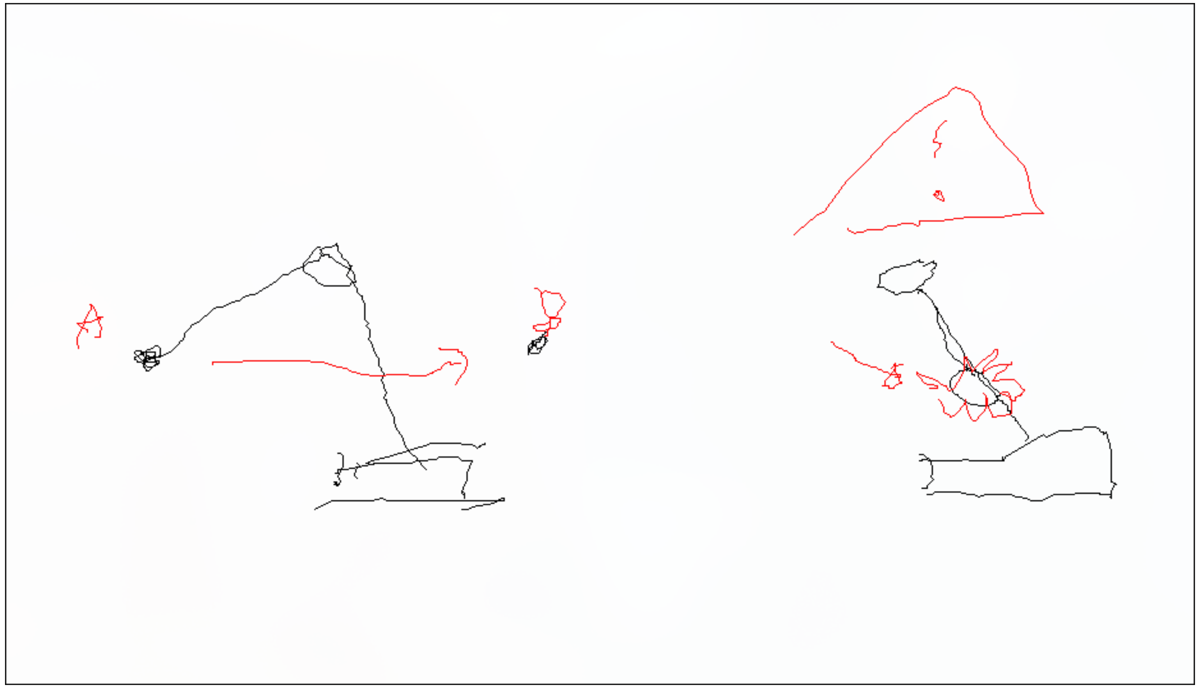
- From J1 to J2

- In the Joints space:
- J1 = { θ1, θ2, θ3, θ4, θ5, θ6}
- J2 = {…. }

- Minimalise the theta difference -> movej
- Movel is like a wrapper around movej

Movement problem

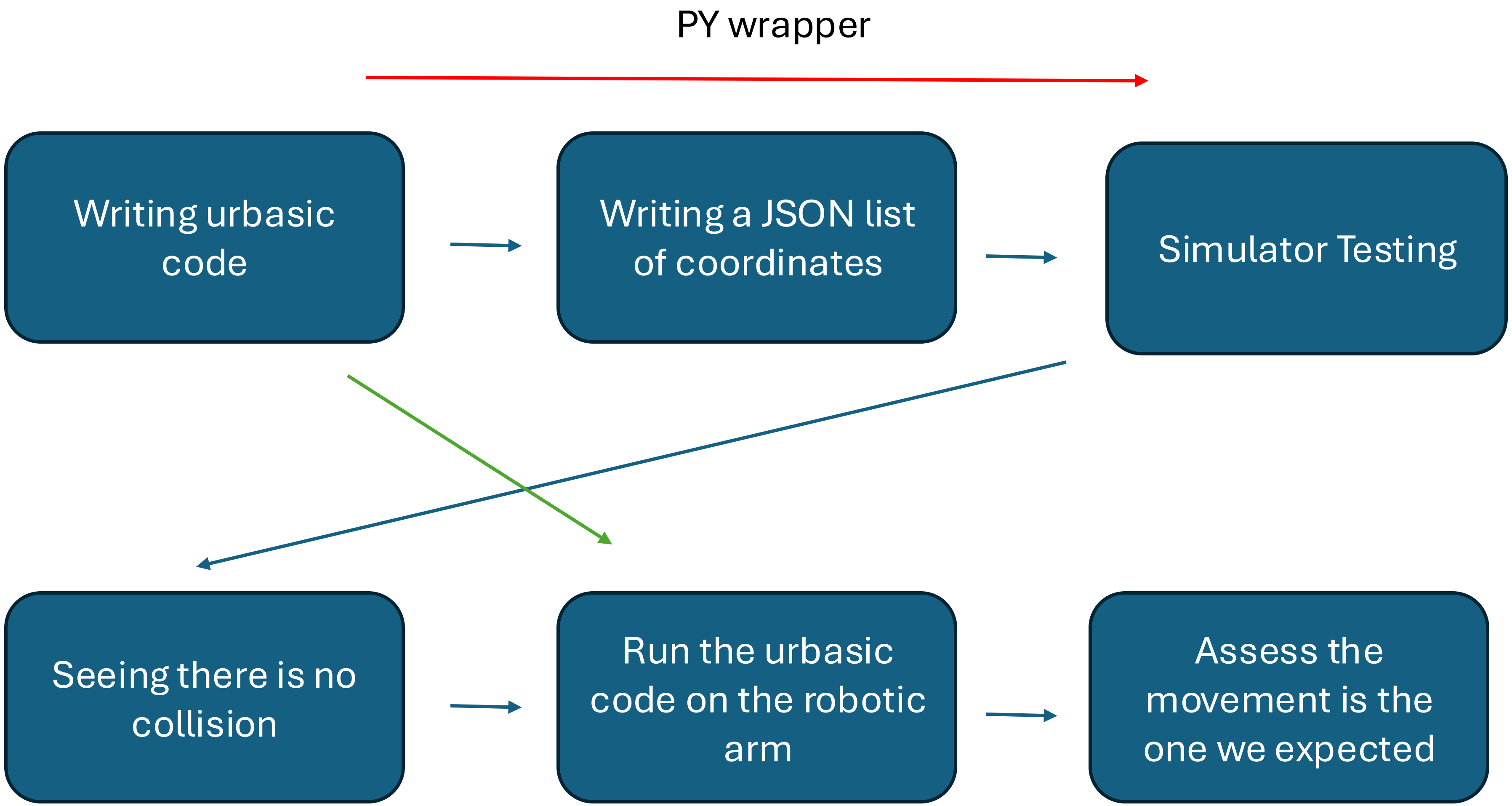
Builtn in collision detection in the robotic internal software, not exposed in URBasic



-> Have to solve it ourself

-> Have to test and implement in the simulator

Testing piepline



PY wrapper

Writing urbasic code

Writing a JSON list of coordinates

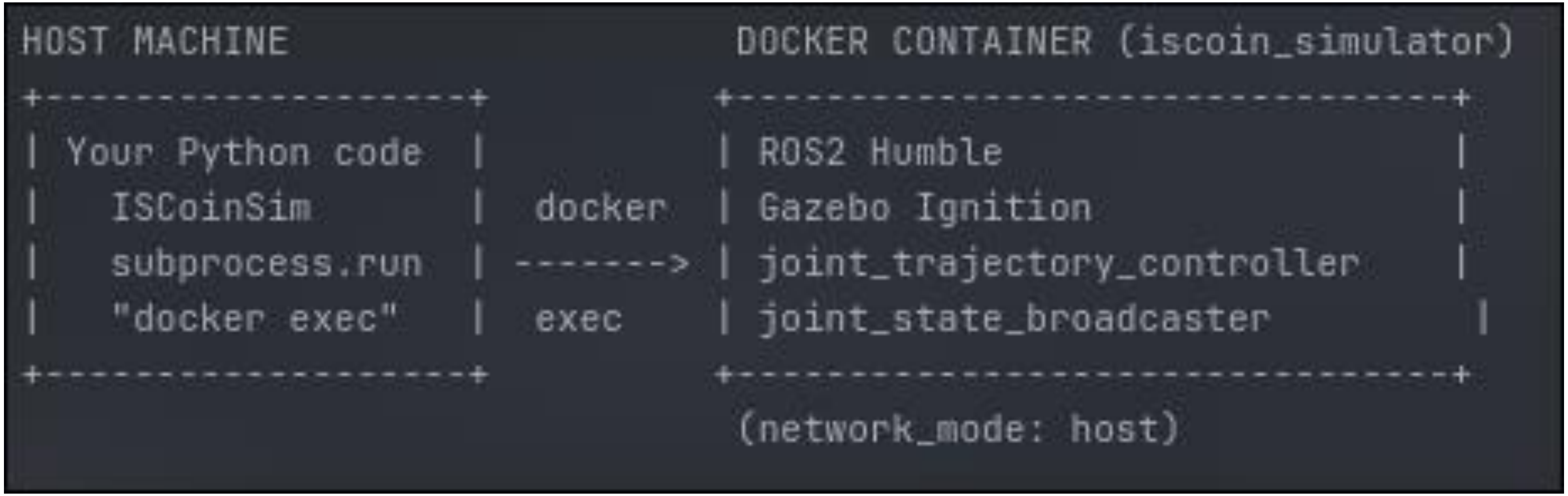
Simulator Testing

Seeing there is no collision

Run the urbasic code on the robotic arm

Assess the movement is the one we expected

URBasic to Gazebo



```
HOST MACHINE                    DOCKER CONTAINER (iscoin_simulator)

+--------------------+          +----------------------------------+
| Your Python code   |          | ROS2 Humble                      |
|   ISCoinSim        |  docker  | Gazebo Ignition                  |
|   subprocess.run   | -------> | joint_trajectory_controller      |
|   "docker exec"    |  exec    | joint_state_broadcaster          |
+--------------------+          +----------------------------------+
                                 (network_mode: host)
```

URBasic

- Movej
- Movel
- Movec
- Movep
- Get_TCP_
- Set_TCP
- Gripper…
- Freemotion…

Calibration

📷

Camera

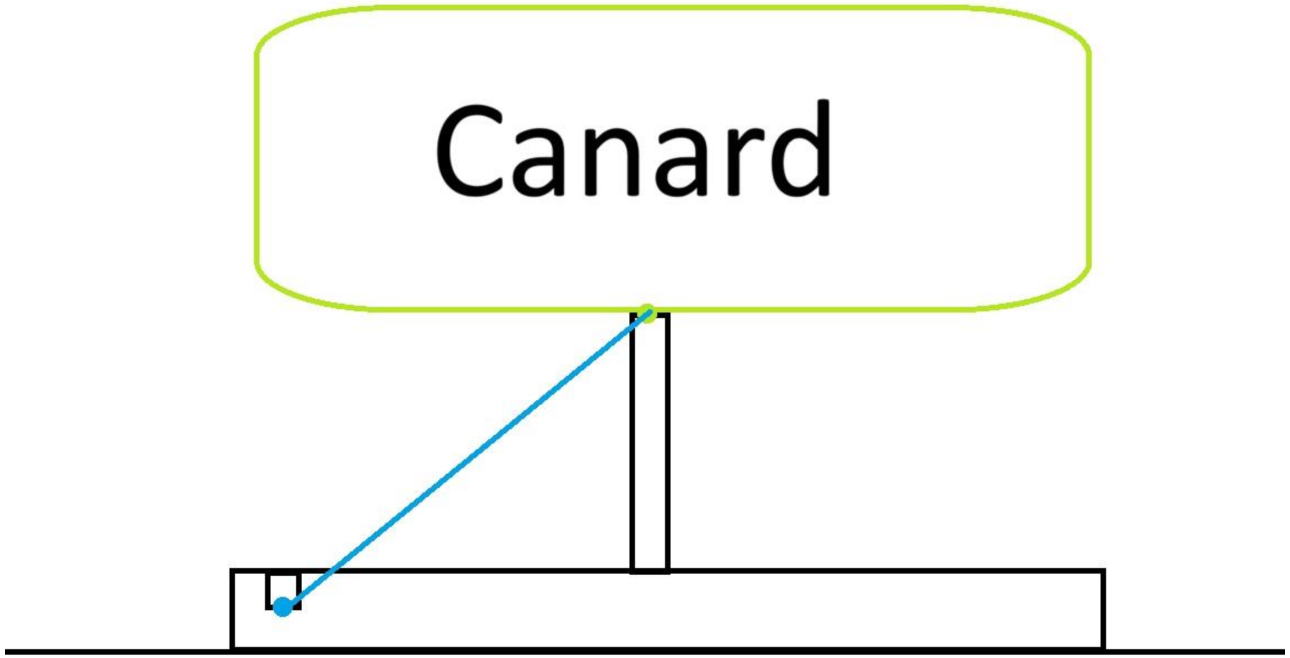
✓

TCP

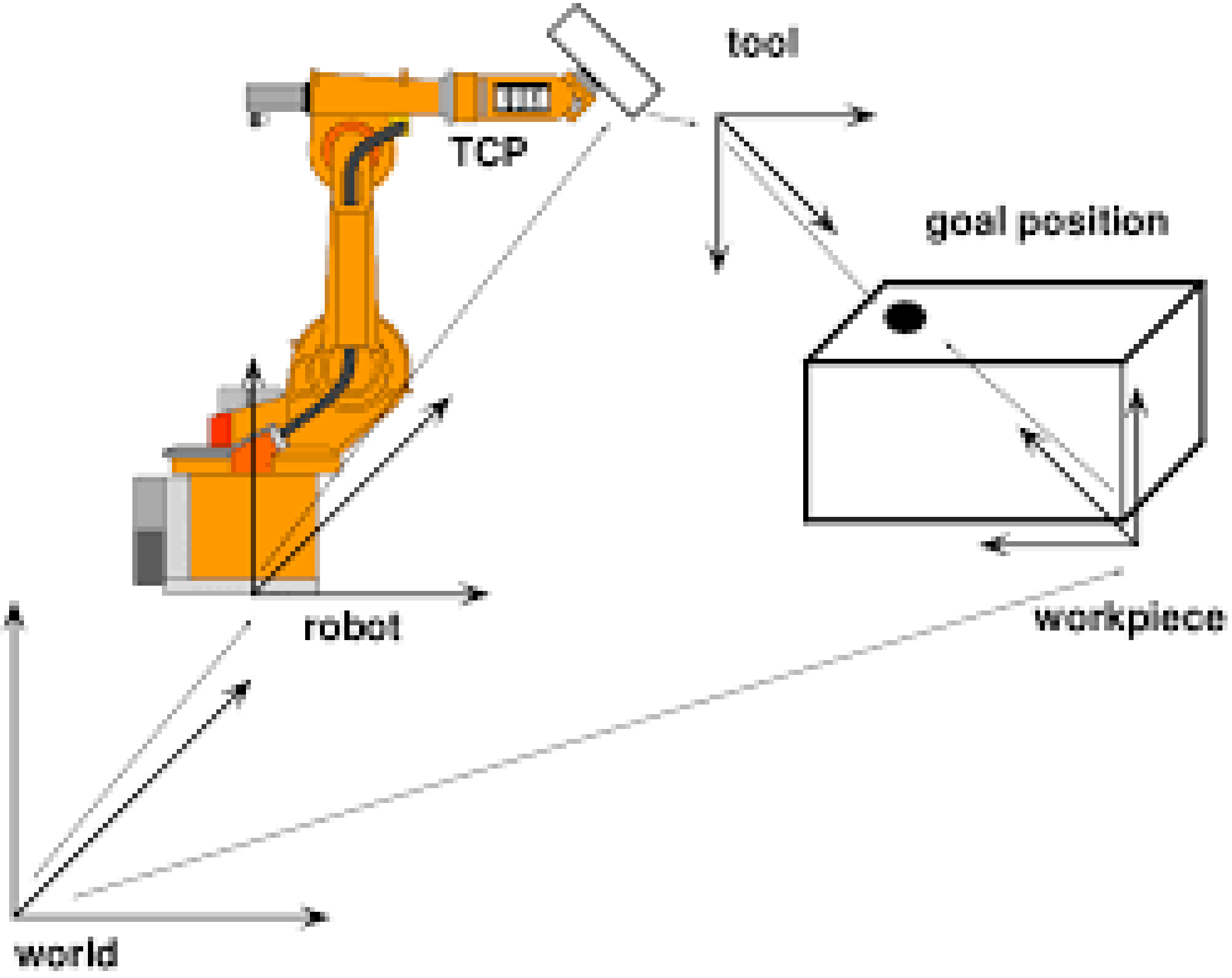
TCP

- freedrive_mode
- Collect multiple TCP points
    - get_actual_tcp_pose
- Solve TCP
- set_tcp
- Validation
    - Rotations of axis (~28°)
    - Success if not move

Workspace Definition



Workspace Definition


tool
TCP
goal position
robot
workpiece
world