Waiter robot simulation

Robot mechanics exam project

A.A. 2023/2024

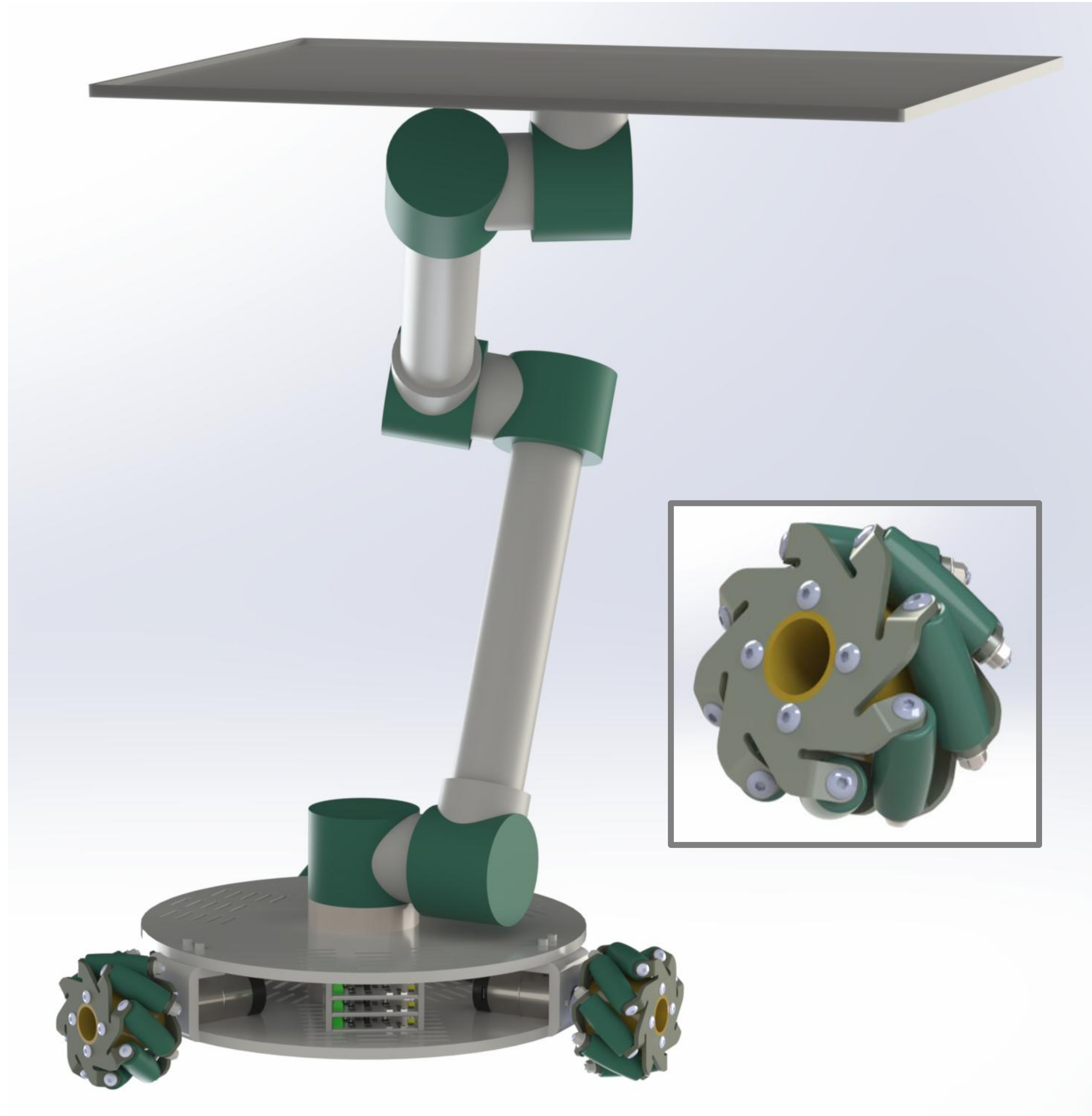
Authors:
Andrea Vivai
Enea Dragoni

Professor:
Marco Gabiccini

Waiter robot

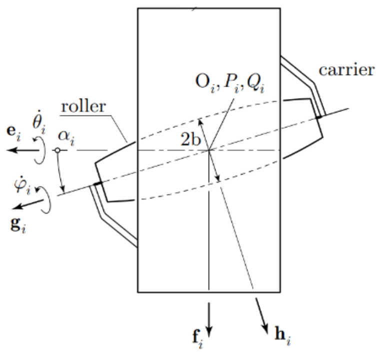
Summary:

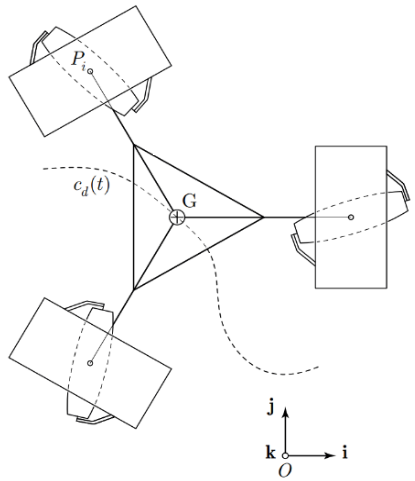
1. Kinematic modeling
2. Base CLIK control
3. Arm CLIK control
4. Task planning
5. Base motion planning
6. Holding tray in place
7. Waiter in action



Kinematic modeling

Base modeling





$\mathbf{Q_i}$: Center of the wheel axis
$\mathbf{O_i}$: Center of the roller axis
$\mathbf{P_i}$: Roller ground contact point
$\mathbf{G}$: Chassis center of mass

Let: $\mathrm{E} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$; $\mathrm{d}_i = \begin{bmatrix} GO_i \cdot \hat{i} \\ GO_i \cdot \hat{j} \end{bmatrix}$; Let: $\mathrm{U} = \begin{bmatrix} u_1^T \\ u_2^T \\ u_3^T \end{bmatrix}$; $\mathrm{u}_i = \begin{bmatrix} g_i \\ g_i^T E d_i \end{bmatrix}$;

The matrix U is invertible if the number of wheels is less than or equal to 3.

Let a be the wheel radius, $\mathrm{Id} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, and define the jacobian as $\mathrm{J} = \text{-a} \sin(\alpha) \cdot Id$

Under the assumption of non-slipping contact, a simple relationship between the wheels speed and the chassis twist can be derived:

$$\boxed{J\dot{\theta} = U\xi_G,} \quad \text{where } \xi_G = \begin{bmatrix} \dot{x}_G \\ \dot{y}_G \\ \omega_G \end{bmatrix} \in \mathbb{R}^3$$

global coordinates

Arm modeling

UNIVERSAL ROBOTS

Using the DH convention:

Link	a_i	θ_i	d_i	α_i
1	l_1	θ_1	0	$-\frac{\pi}{2}$
2	l_2	θ_2	0	$\frac{\pi}{2}$
3	l_3	θ_3	0	$\frac{\pi}{2}$
4	l_4	θ_4	0	$-\frac{\pi}{2}$
5	l_5	θ_5	0	$-\frac{\pi}{2}$
6	l_6	θ_6	0	$\frac{\pi}{2}$
7	l_7	θ_7	0	0

Base CLIK control

Let $\dot{\theta}_b$ be the vector of wheels rotation speeds.
Let J_b be the jacobian of the base vehicle.

Since (last slide):

$$J_b \, \dot{\theta}_b = U\xi_G$$

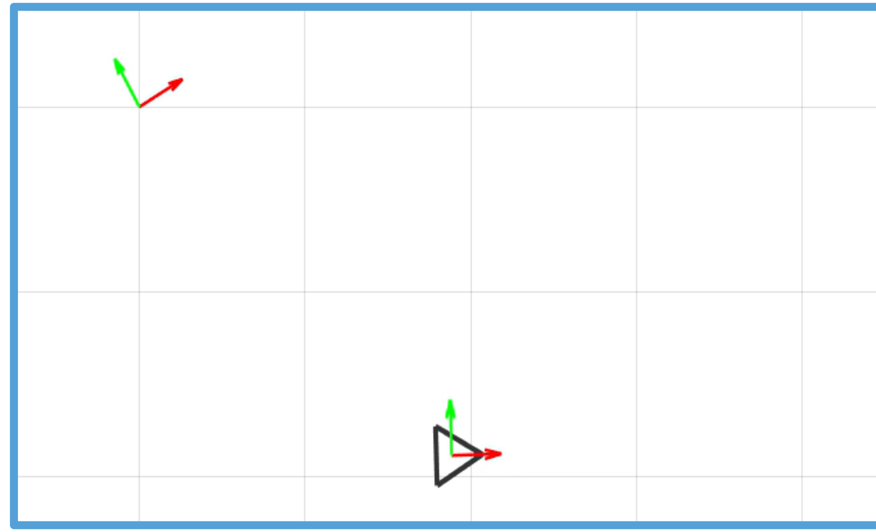
We can define a CLIK control as following:

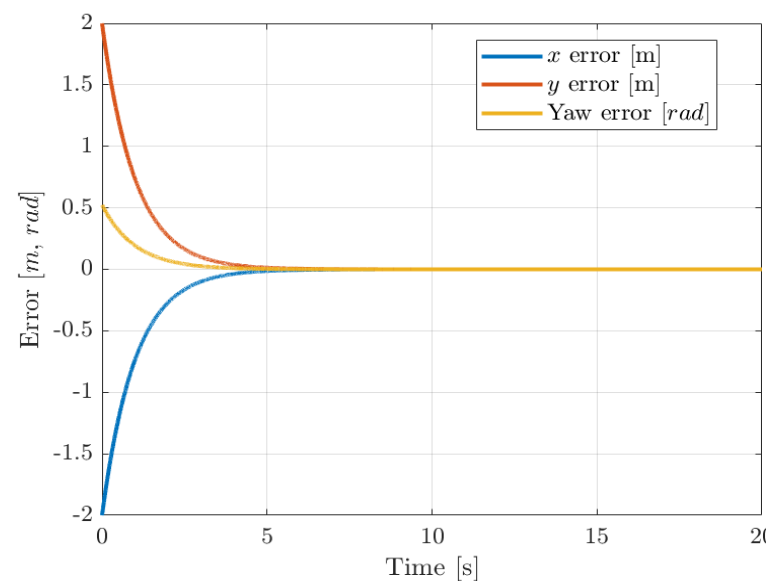
$$\dot{\theta}_b = {J_b}^{-1} U K_b e_b$$

With $e_b = \begin{bmatrix} x_{WG}^{(goal)} - x_{WG} \\ y_{WG}^{(goal)} - y_{WG} \\ yaw^{(goal)} - yaw \end{bmatrix}$

Where: W = «world» ; G = «mobile base». So the error is calculated in global coordinates

Note: The base can indipendently control its translation and rotation





Arm CLIK control

Let $\dot{\theta}_a$ be the vector of arm joints speeds.
Let J_a be the geometric jacobian of the arm.

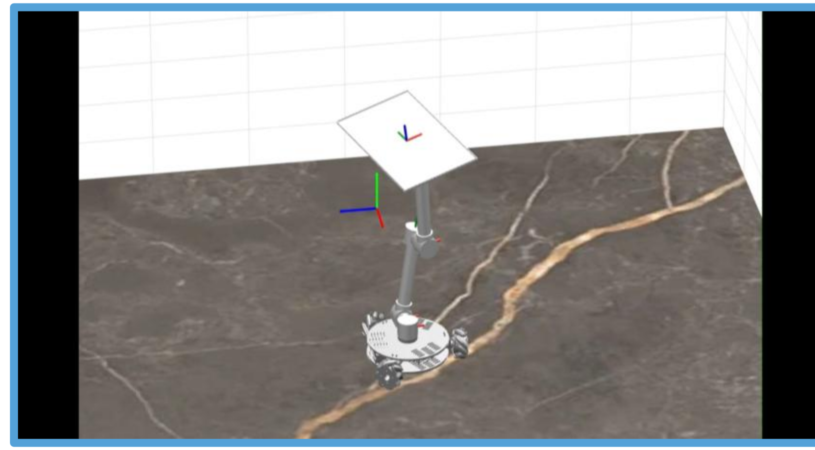
The controller is a CLIK with orientation error (e_{orient}) parametrized using **quaternions**:

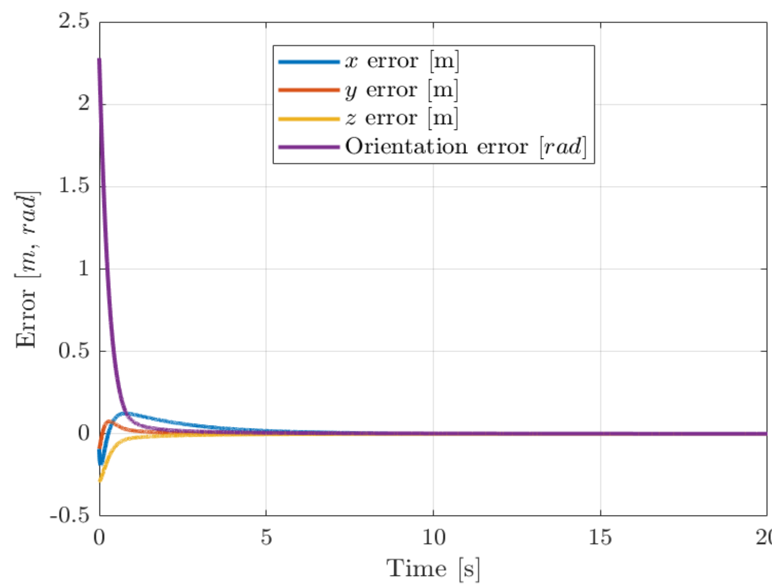
$$\dot{\theta}_a = {J_a}^{-1}(K_a e_a - \xi_{WE}^{base})$$

Where: E = «end effecrtor» ;

ξ_{WE}^{base}: twist of the end effector, relative to the world frame, written in world frame, **resulted assuming that the arm is a rigid body fixed to the base.** It can be thought as a «feedforward compensation»;

$e_a = \begin{bmatrix} e_{pos} \\ e_{orient} \end{bmatrix}$ error calculated assigning a desired end effector pose, in world coordinates



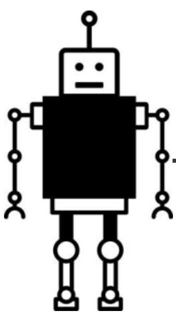


Task planning

DESIRED_POSE

CUSTOMERS POSES DATASET

Desired end effector poses, relative to the «World» R.F.



CURRENT_POSE

STATE ESTIMATION (WHOLE ROBOT)

ARM PLANNER

```
1. pose_in_reach ← check_if_within_reach(CURRENT_POSE)
2. if pose_in_reach:
3.     desired_arm_pose ← get_arm_pose_error(DESIRED_POSE, CURRENT_POSE)
4.     arm_control      ← CLIK_arm(arm_error)
5. else:
6.     arm_control      ← rest_position
```

BASE PLANNER

More on this in the next slide…

```
1. base_pose          ← get_base_pose(CURRENT_POSE)
2. desired_base_pose ← RRT*(DESIRED_POSE)
3. base_error         ← desired_base_pose - base_pose
4. wheels_control     ← CLIK_base(base_error)
```

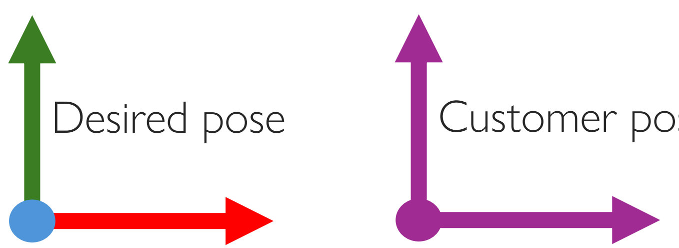
BASE MOTION PLANNING

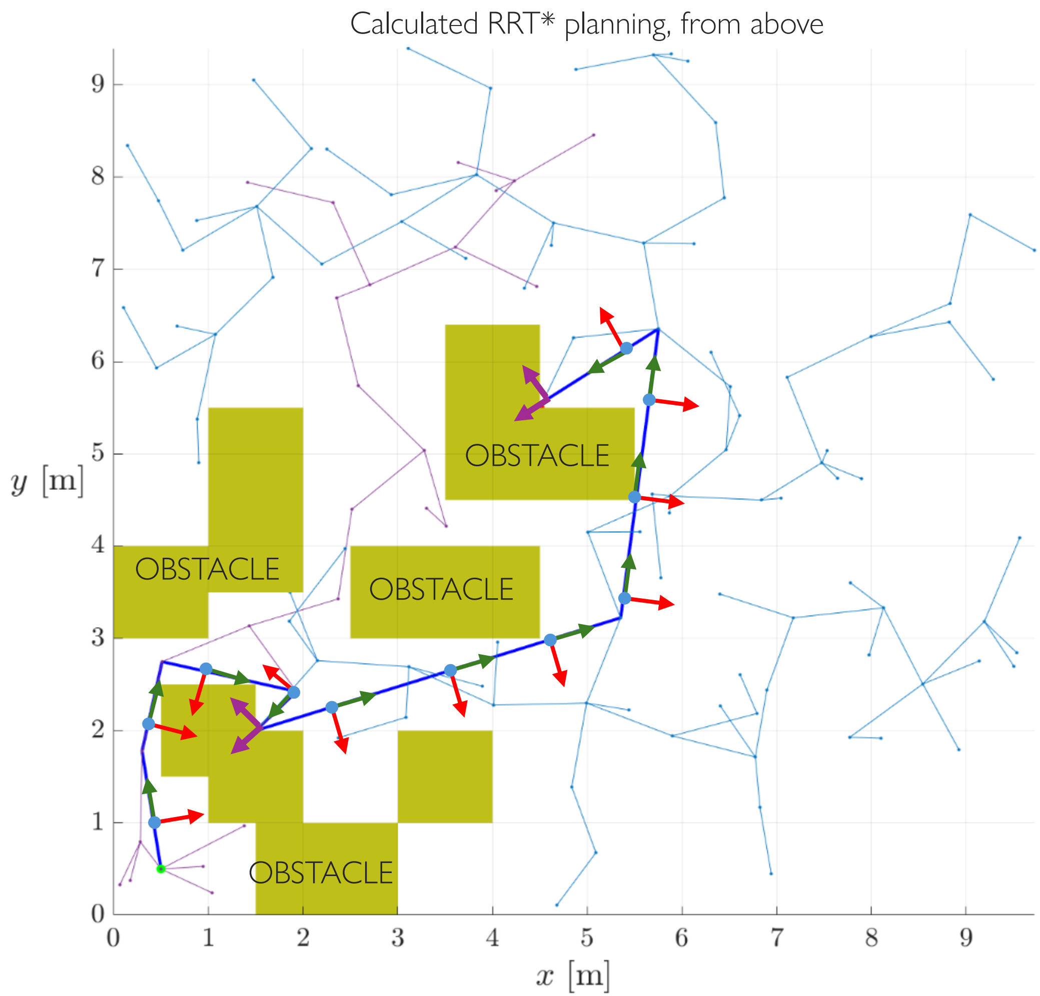
The goal of the base is to move the arm in a position where the it can reach the customer.

Given an end effector pose, the base goal is to move towards a desired position, that is the same as the end effector pose, but translated on the ground.

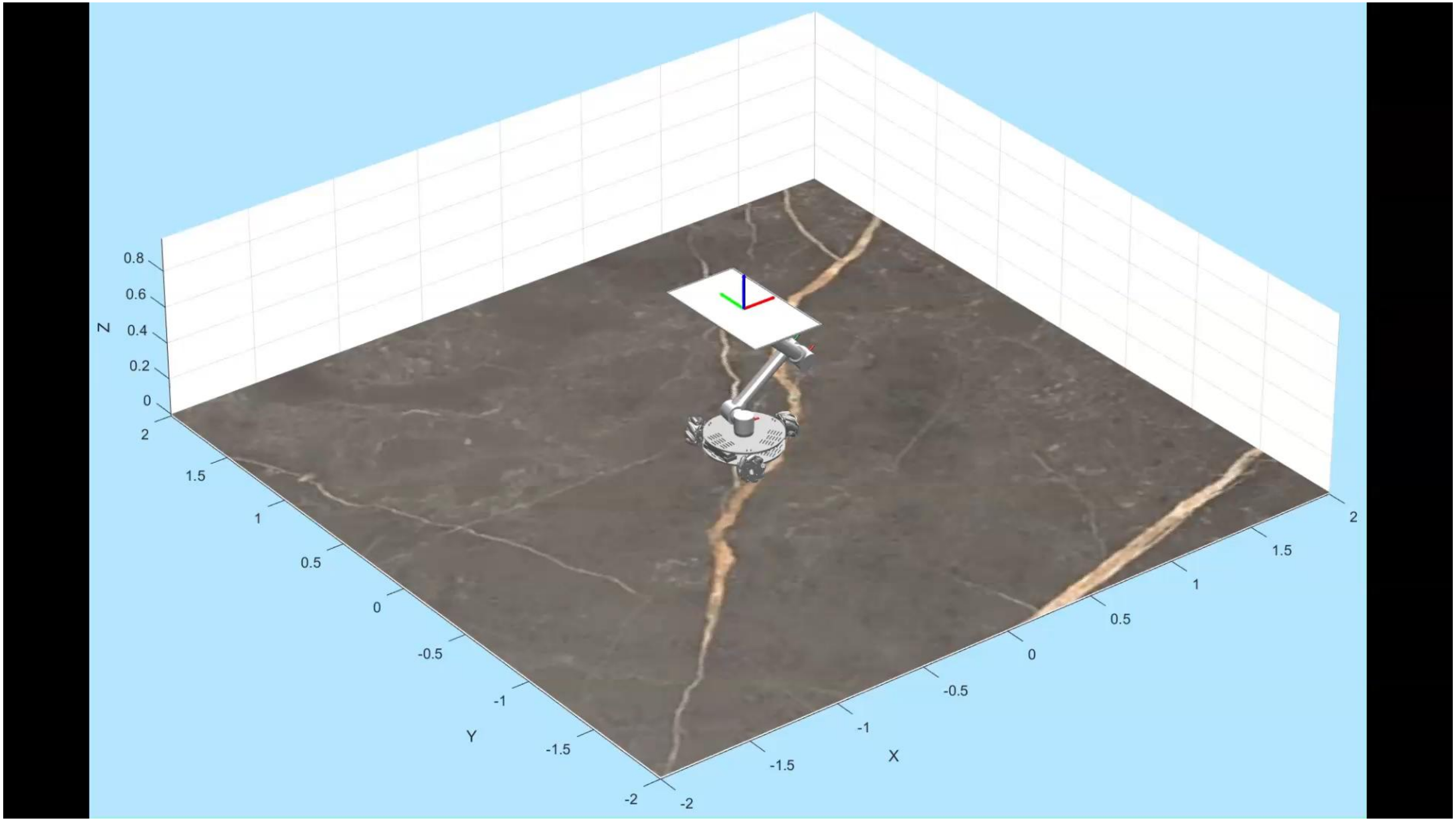
Using RRT*, a dataset of base poses are calculated, to allow the base to navigate the enviroment, and reach the final desired pose.

The base then is controlled giving a succession of desired poses (position + yaw) from the dataset.





Holding tray in place



Waiter in action