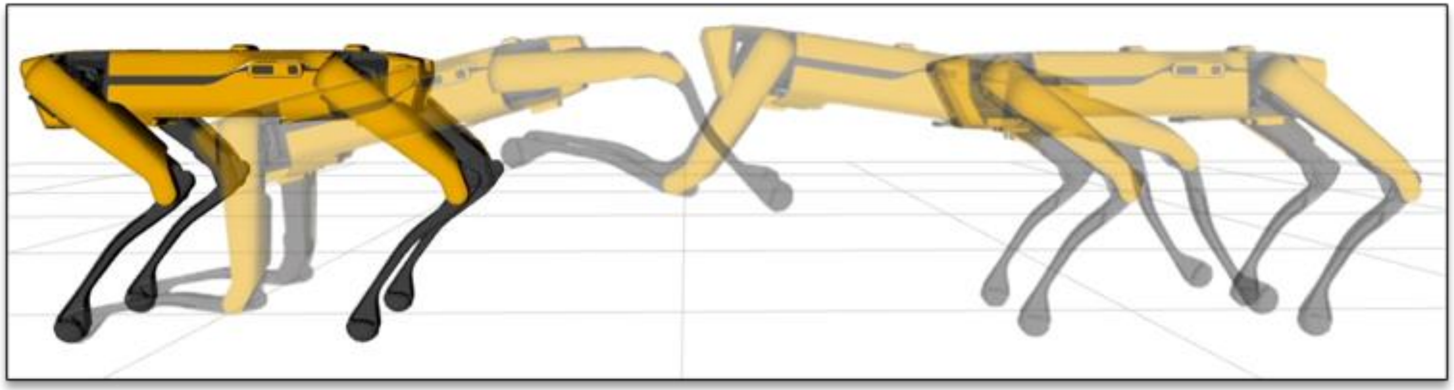
Horizon:
A Trajectory Optimization Framework for Robotic Systems



ISTITUTO ITALIANO
DI TECNOLOGIA
HUMANOIDS AND HUMAN
CENTERED MECHATRONICS

Features

Robotics focus: built-in methods and robotics-oriented utilities

User-friendly interfaces: simplify the formulation process, set up an optimization with minimal effort

Comprehensive pipeline: all the modules to generate a complete robot motion only requiring standard inputs such as XML files

Versatility: generic enough to include all the necessary tools to prototype:

- offline dynamic motion
- receding horizon walking gait
- co-design a robot structure

- **Open-source**: based on open-source packages, and freely available itself

NLP formulation

$$\min_{\mathbf{x},\mathbf{u}} \int_{t_0}^{t_f} \ell(\mathbf{x},\mathbf{u};\mathbf{p},t)\mathrm{d}t + \ell_f\left(\mathbf{x}_f;\mathbf{p},t_f\right)$$ → objective function to minimize

subject to: $$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u};\mathbf{p},t)$$ → dynamics of the system

$$\phi^{\min} \le \phi(\mathbf{x},\mathbf{u};\mathbf{p},t) \le \phi^{\max}$$ → desired constraints

⬇

NLP problem set up using the **symbolic framework** provided by CasADi:

- cost and constrain functions are defined using symbolic expressions
- state-of-the-art implementation of algorithmic differentiation (AD)

Modules

Robot model acquisition
↓
acquires robot model parsing the URDF

Problem formulation
↓
concise syntax to specify costs and constraints, and distribute them over horizon

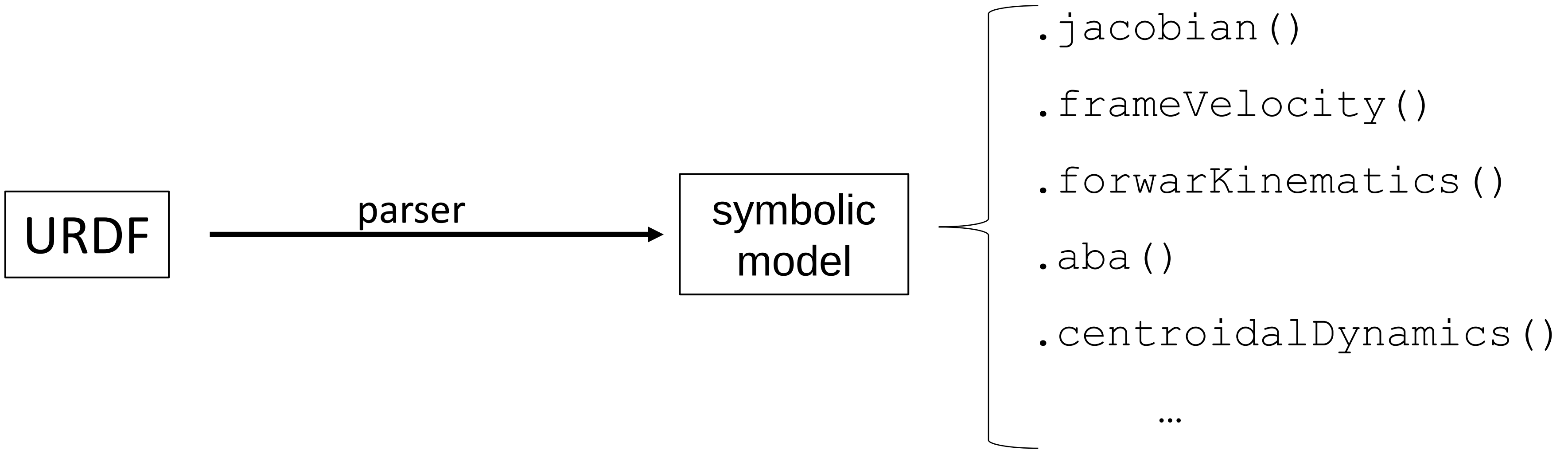
Solver selection
↓
different solvers available to meet different requirements

Receding horizon
↓
tools for cheaply relocating costs and constraints to different nodes

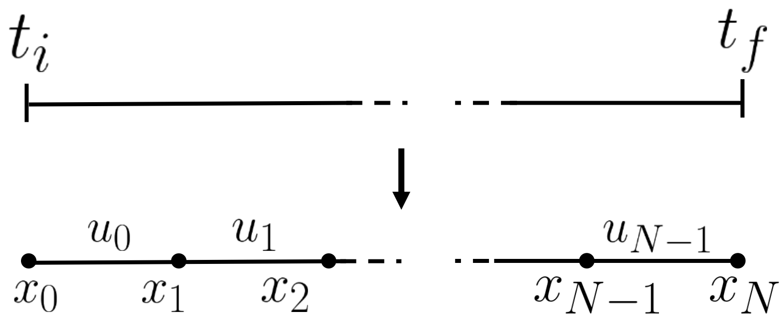
Robot model acquisition

Pinocchio
Efficient and versatile rigid body dynamics algorithms

- Parse the URDF file generating a ready-to-use model in Horizon
- From XML file to *symbolic description* compatible with the framework
- Pinocchio library for **robot kinematics and dynamics**



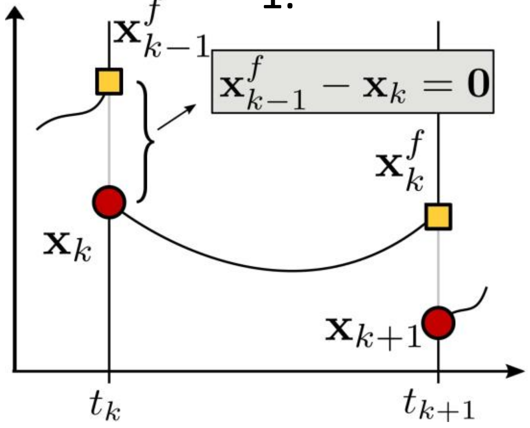
Problem formulation



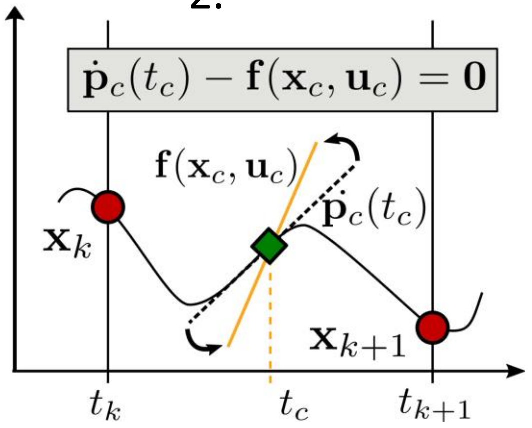
- **Duration** and **discretization** of time horizon

```
prb = Problem(nodes=50)
prb.setDt(dt)
```

1.



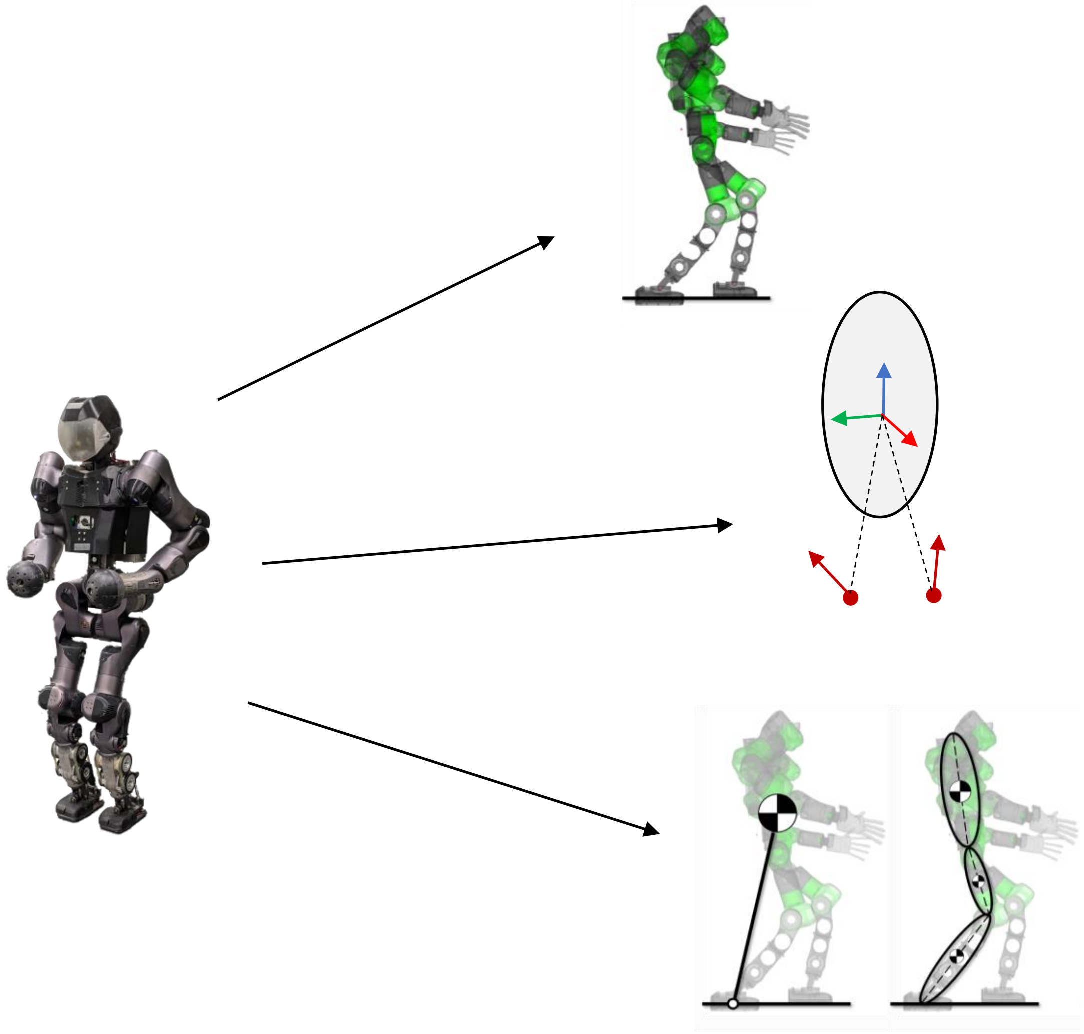
2.



- **Problem transcription**:
    1. Direct Multiple Shooting
    2. Direct Collocation

```
Transcriptor.make("multiple_shooting", prb)
```

Problem formulation



- **System dynamics**:

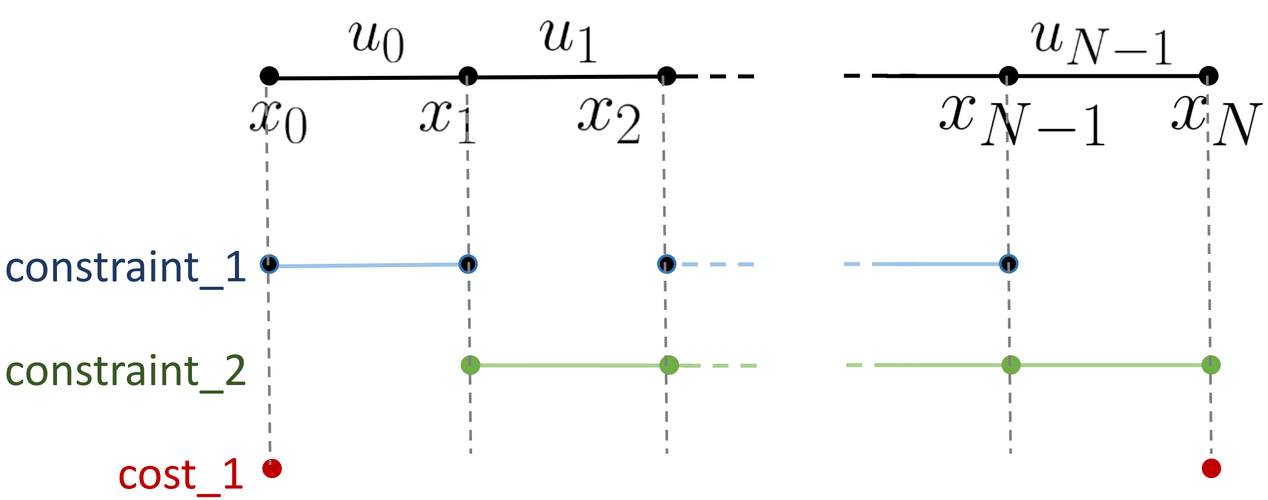
  1. Built-in models:
     - *Full body dynamics*
     - *Single Rigid Body Dynamics*
     - *Centroidal Dynamics*

  2. Custom implementation:
     as differential-algebraic system of equation (DAE)

```
x = prb.createStateVariable("x", dim)
u = prb.createInputVariable("u", dim)

prb.setDynamics(x_dot)
```

Problem formulation



- **Functions and bounds definition**:
  - *constraints* and *costs* defined on desired nodes
  - *bounds* for variables and constraints

```
prb.createCost("qddot",
               cs.sumsqr(u),
               nodes=[1,2,15])

prb.createConstraint("frict_cones", f_c)

x.setBounds(ub, lb, nodes)
```

Solvers

Solver selection depending on the user's requirements.

Custom implementations:

- Gauss-Newton Sequential Quadratic Programming (**GN-SQP**)
- multiple-shooting Iterative Linear-Quadratic Regulator (**ILQR**)

→ Fast, useful for receding horizon formulation

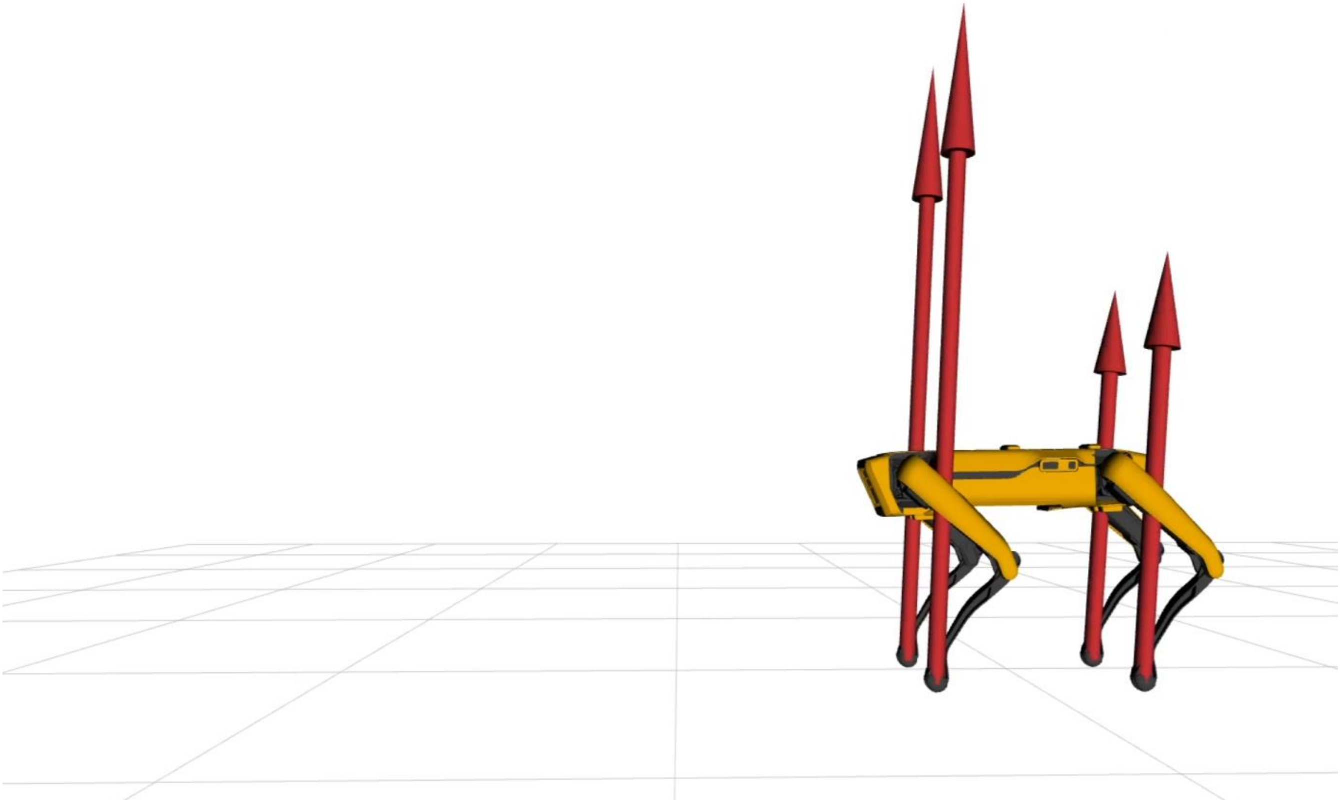
Off-the-shelf solvers available through CasADi interface:

- **IPOPT**/BONMIN,
- BlockSQP
- WORHP
- KNITRO and
- SNOPT

→ Large-scale nonlinear optimization, interior-point methods

Robot motion design

- Optimized trajectory: combined result of bounds, constraints and cost functions
- The desired behavior can be achieved by tuning the formulation of the problem
- High performance index corresponds to the closeness of the trajectory to the desired behavior



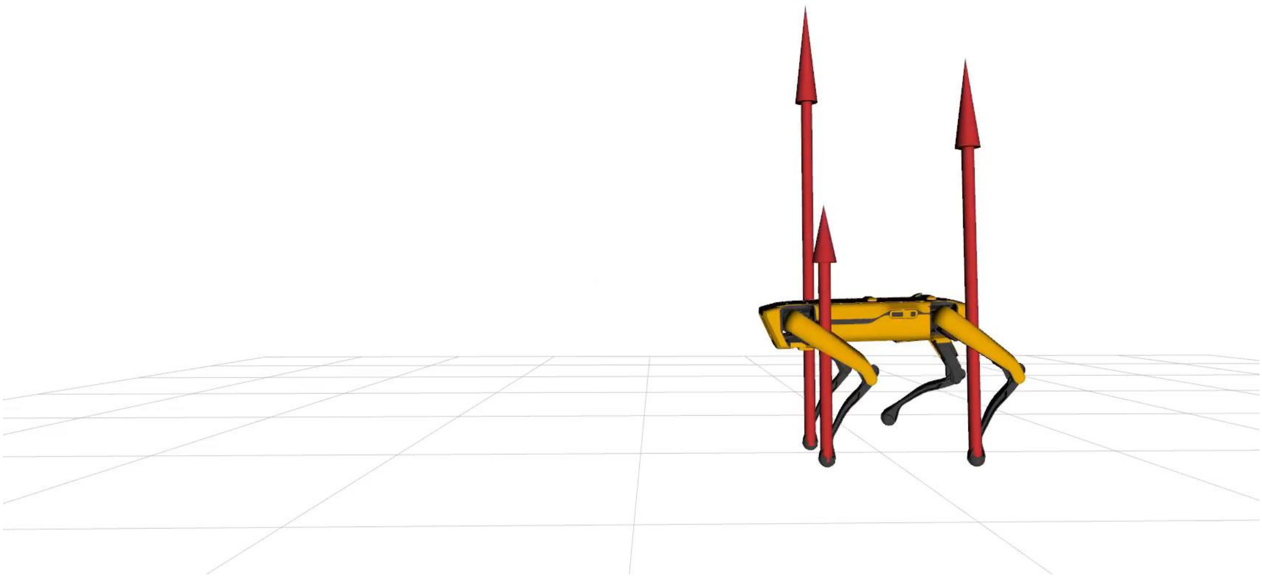
Receding horizon

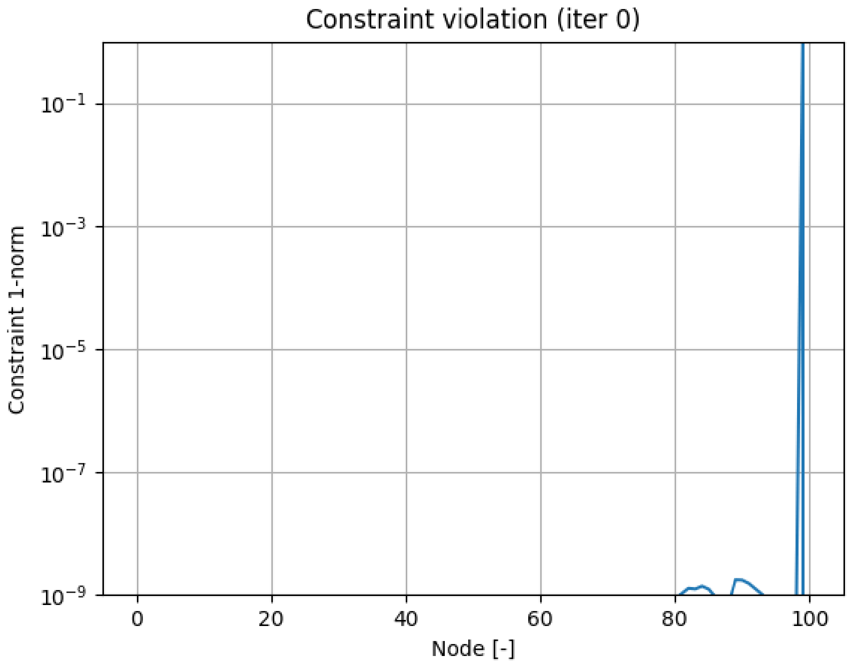
Online scenario:

- real *system state* is changing
- *user references* are changing

More effective if user references are changed "close to the tail"

Idea: update system state and user inputs before each iteration





Spot’s receding horizon walking

Problem options:

- Full body dynamics
- N_nodes = 100
- dt = 0.1 s

Constraints:

- [*stance*] zero contact velocity
- [*stance*] unilateral contact force
- [*swing*] zero force
- [*swing*] prescribed z-trajectory
- force-acceleration consistency

Costs:

- reference base link velocity
- postural
- regularization

Parameters:

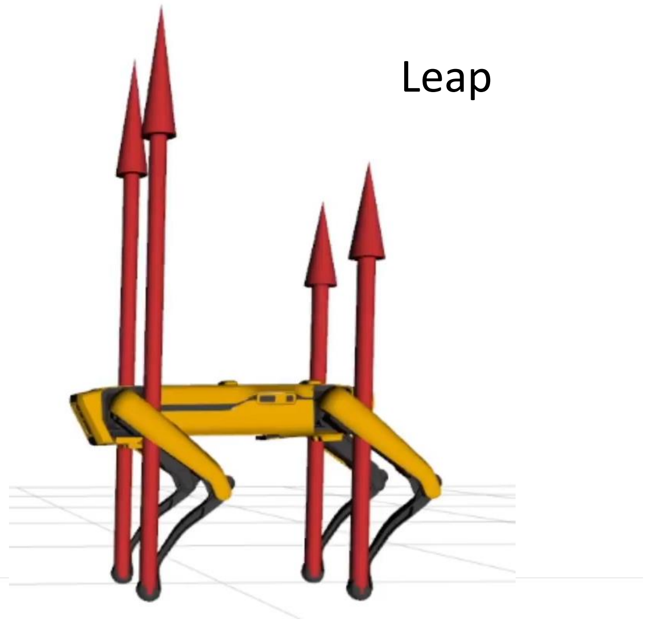
- reference velocity
- contact schedule
- z trajectory

Spot results

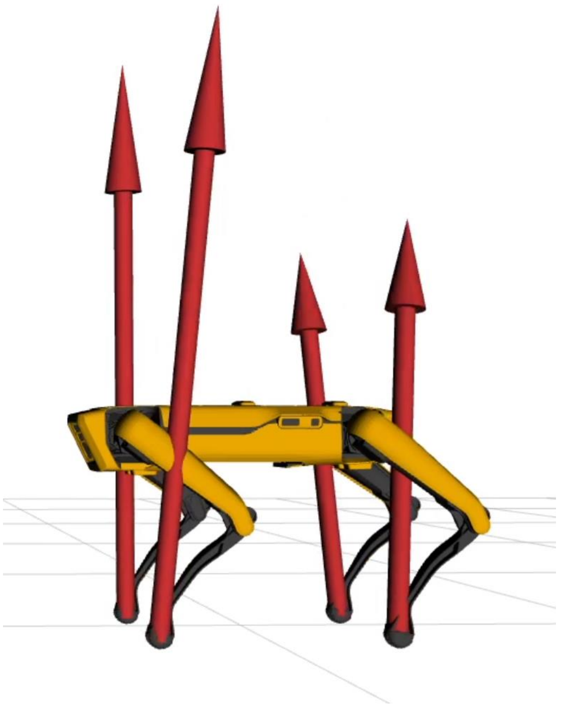
Formulation of the problem is similar between the different motions:

- schedule and duration of the contacts

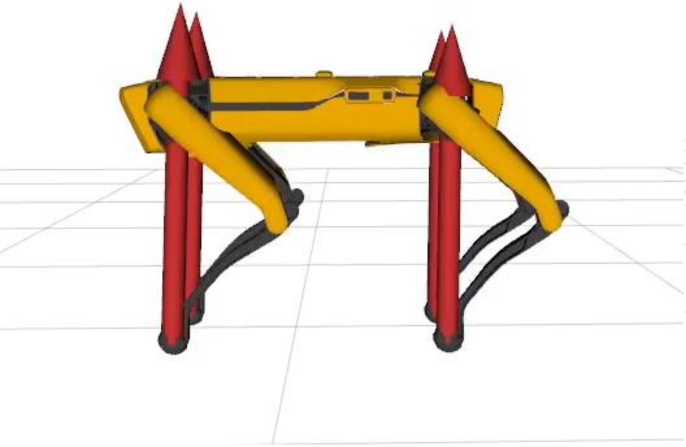
Leap



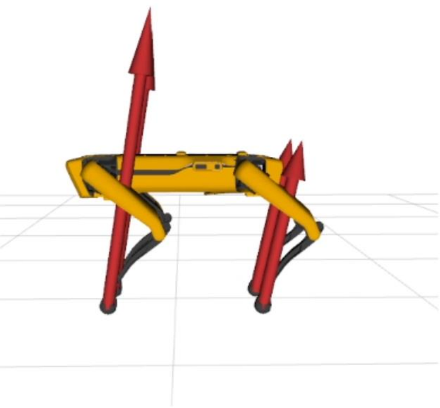
Jump and turn

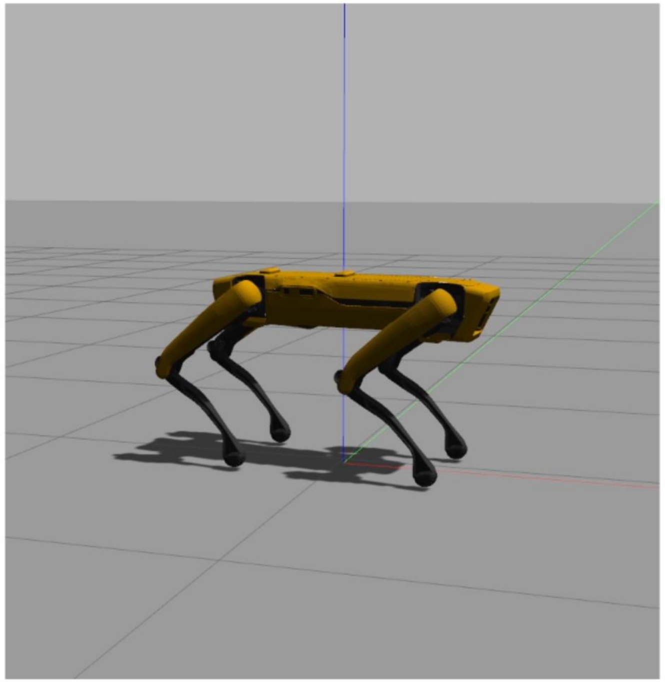


Wheelie

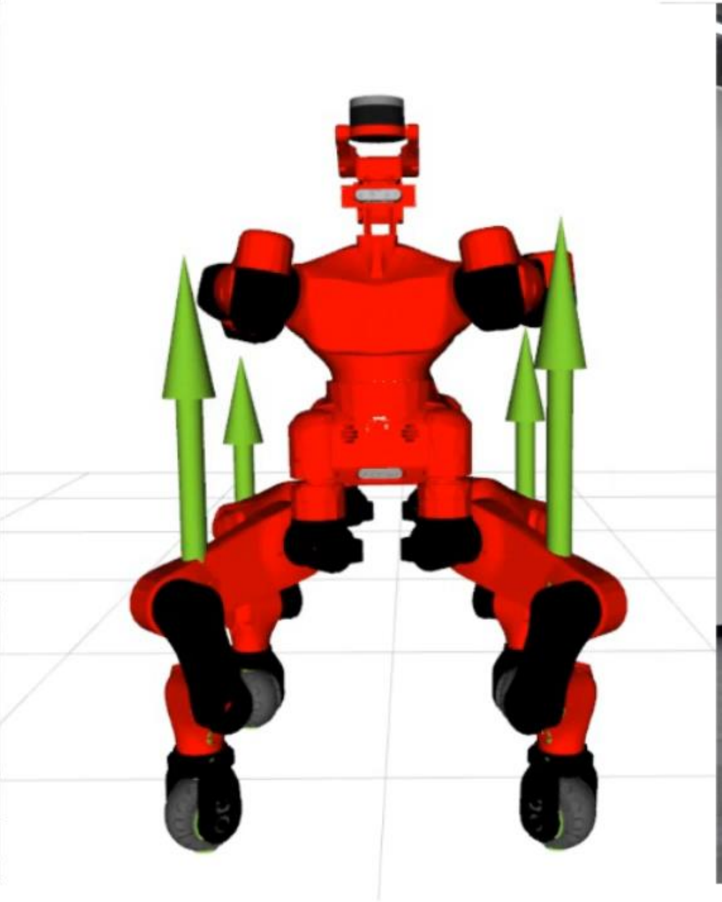


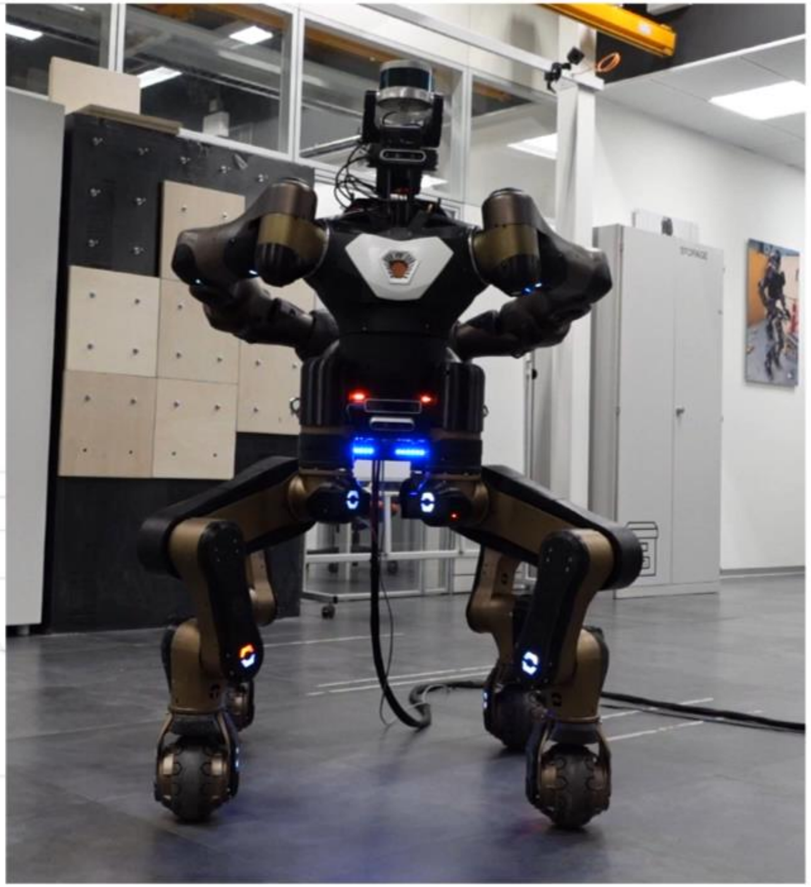
Jump





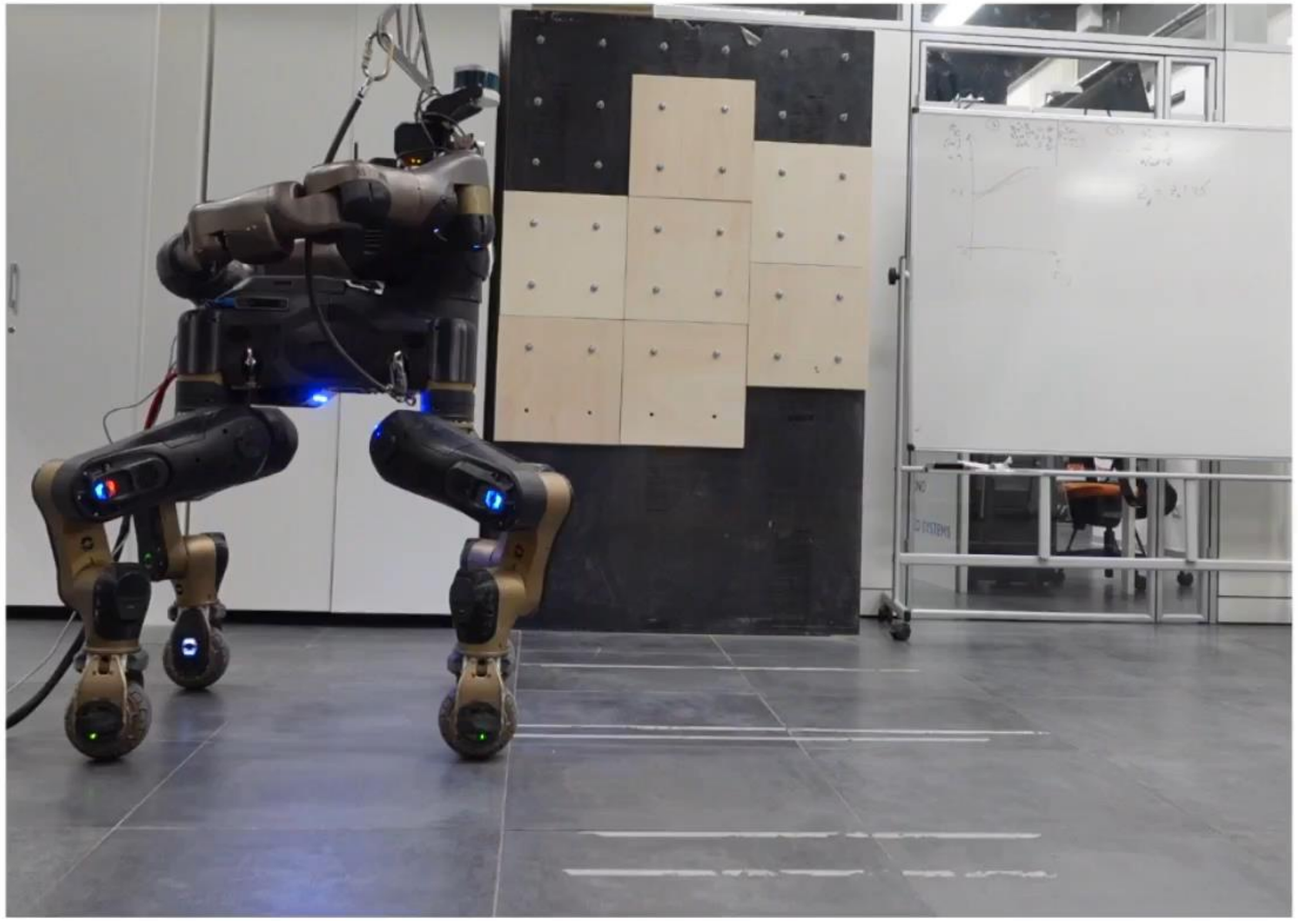
Centauro robot performing a series of steps to change its heading:

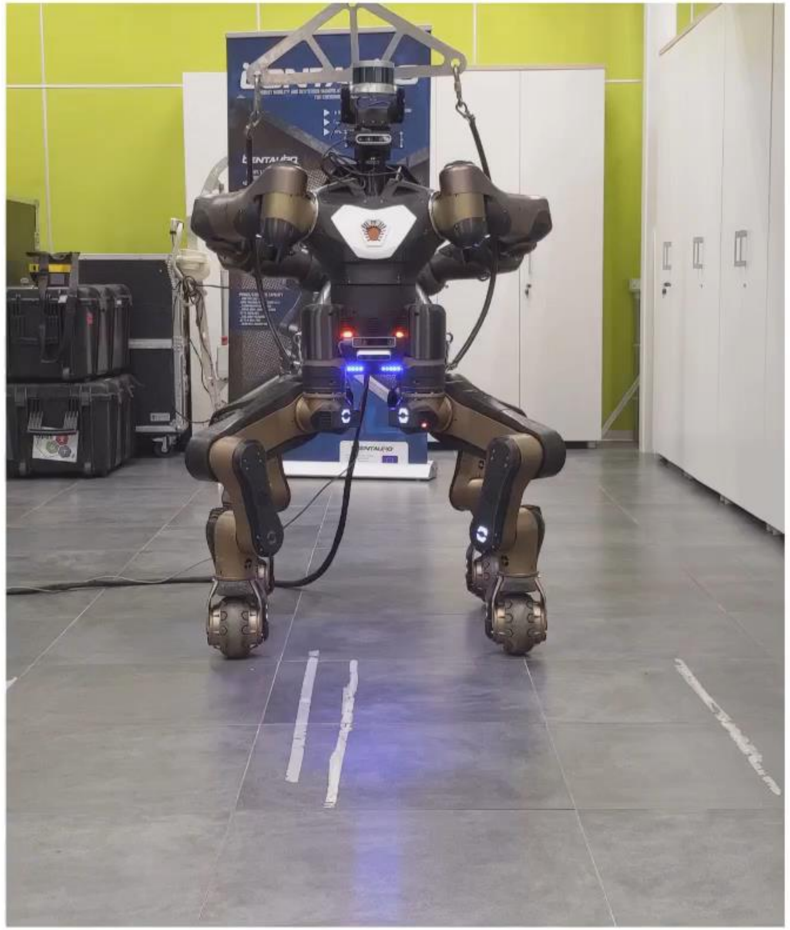




crawling gait on the robot Centauro:

due to the gait scheduling and the final pose, very large strides are required





Dynamic trajectories for a 2-DoF prototype robotic leg for the initial design of a leg intended for agile motions

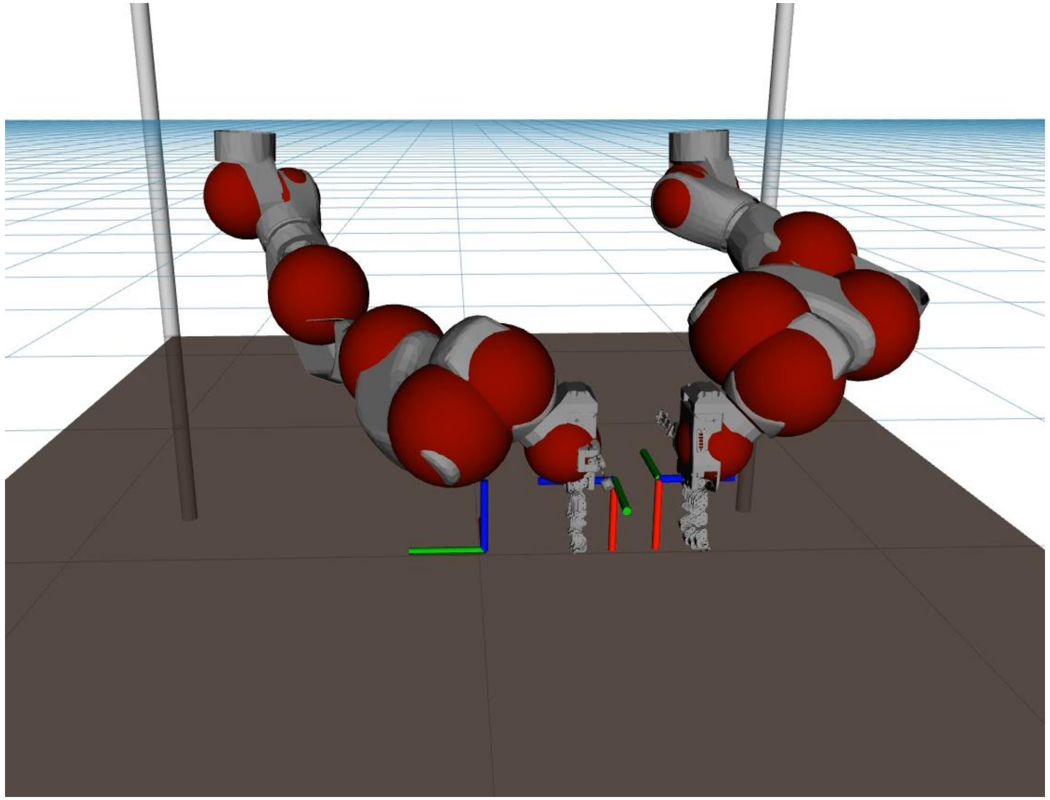
analyze relevant parameters:

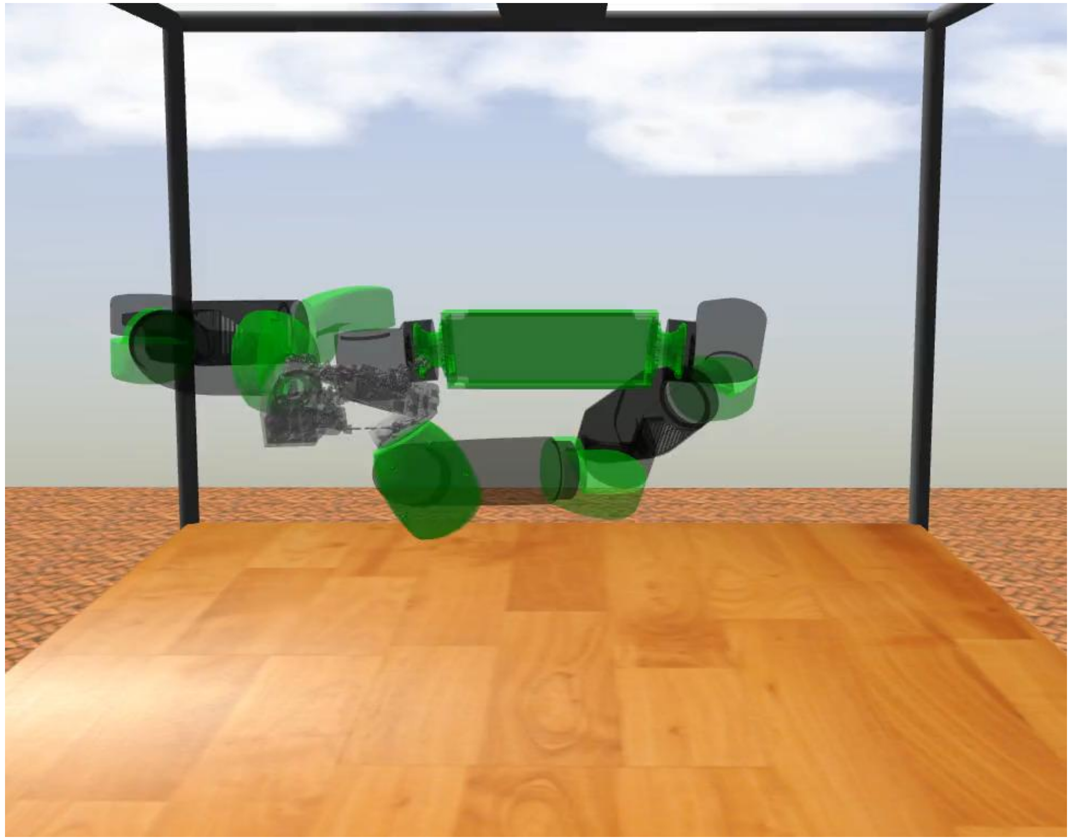
- maximum current
- torque
- angular velocity

to carry out the sizing of the motors

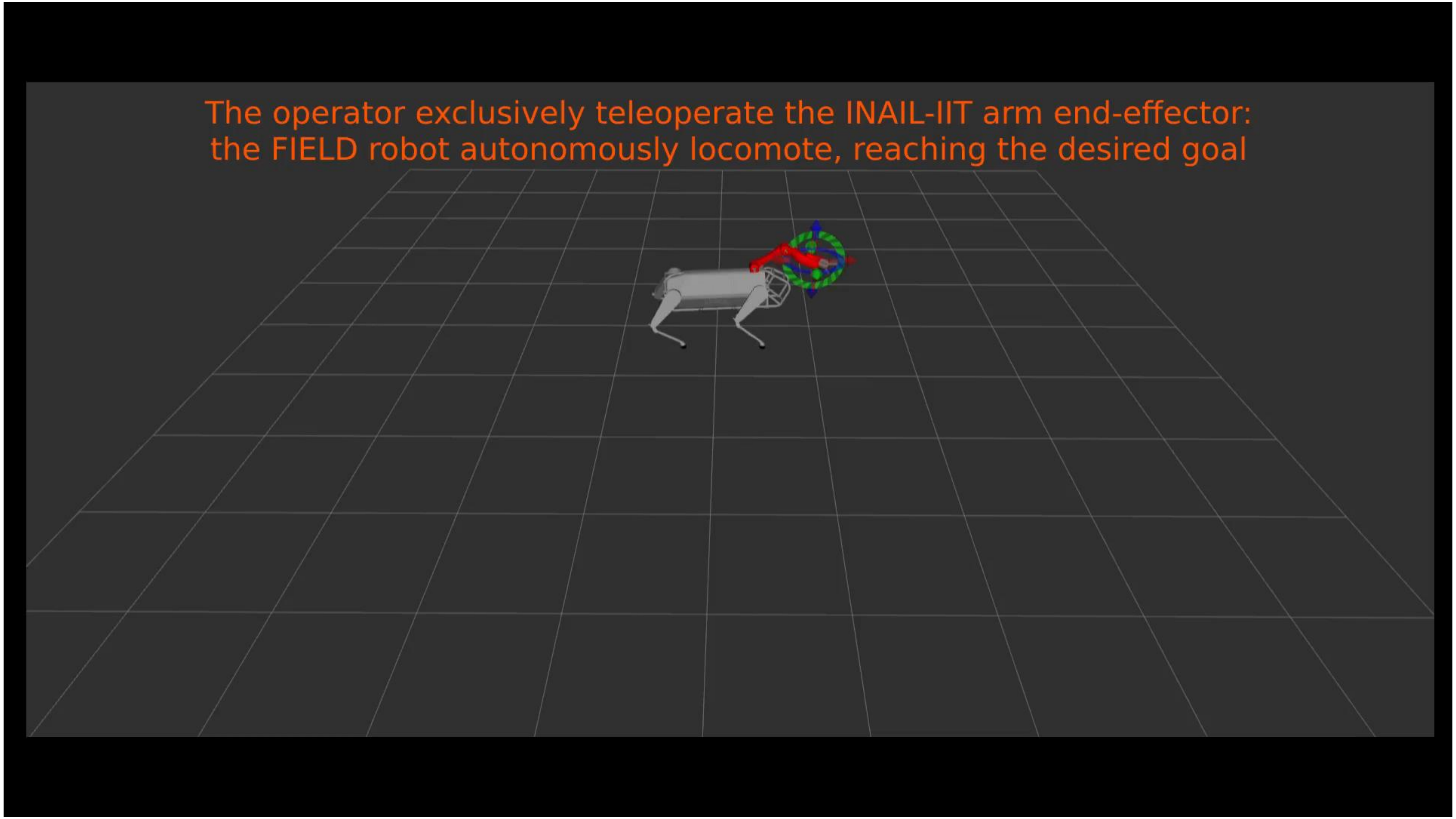
Co-design of bimanual manipulator

- Optimize link length and joint orientation to maximize manipulability





Intuitive robot operation through end-effector Cartesian control



Thank you for your attention!