Underactuated Robots: Humanoid Robot Control

Nicola Scianca

December 2025

MPC using a whole-body kinematic model

- it would be great if we could express the **centroidal quantities** (CoM and angular momentum) in terms of the **joint motion**
- this would let us build an MPC in which we don't need to neglect angular momentum, and we can have a better prediction of the ZMP
- we also don't need a whole-body controller anymore: the MPC gives us directly the **joint commands**

MPC using a whole-body kinematic model

- we can define a generic **position task** with acceleration input

$$\begin{pmatrix} \boldsymbol{r}^{i+1} \\ \dot{\boldsymbol{r}}^{i+1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I} & \delta_t \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{r}^{i} \\ \dot{\boldsymbol{r}}^{i} \end{pmatrix} + \begin{pmatrix} \delta_t^2 \boldsymbol{I}/2 \\ \delta_t \boldsymbol{I} \end{pmatrix} \ddot{\boldsymbol{r}}^{i}$$

- we can linearize $\ddot{\boldsymbol{r}}$ around the **previous MPC solution** $(\bar{\boldsymbol{q}}_0, \ldots, \bar{\boldsymbol{q}}_N)$, $(\bar{\boldsymbol{\nu}}_0, \ldots, \bar{\boldsymbol{\nu}}_N)$

$$\ddot{\boldsymbol{r}}^{i} = \boldsymbol{J}_r(\bar{\boldsymbol{q}}^{i})\dot{\boldsymbol{\nu}}^{i} + \dot{\boldsymbol{J}}_r(\bar{\boldsymbol{q}}^{i}, \bar{\boldsymbol{\nu}}^{i})\bar{\boldsymbol{\nu}}^{i}$$

- now we have a linear link between a generic task and the floating base and joint accelerations $\dot{\boldsymbol{\nu}}$

- if the robot is **position-controlled**, from the joint accelerations we can directly reconstruct the joint position commands (otherwise we find appropriate torques)

Belvedere et al.,"Joint-Level IS-MPC: a Whole-Body MPC with Centroidal Feasibility for Humanoid Locomotion", International Conference on Intelligent Robots and Systems, 2024

MPC using a whole-body kinematic model

- we can use this model to map accelerations $\dot{\nu}$ to all the relevant robot quantities:
  - center of mass
  - angular momentum
  - torso orientation
  - feet position and orientation
  - zero-moment point
  - hands position and orientation

- using these, we can formulate tasks, and use these tasks either as **cost function terms** or **constraints**

MPC using a whole-body kinematic model

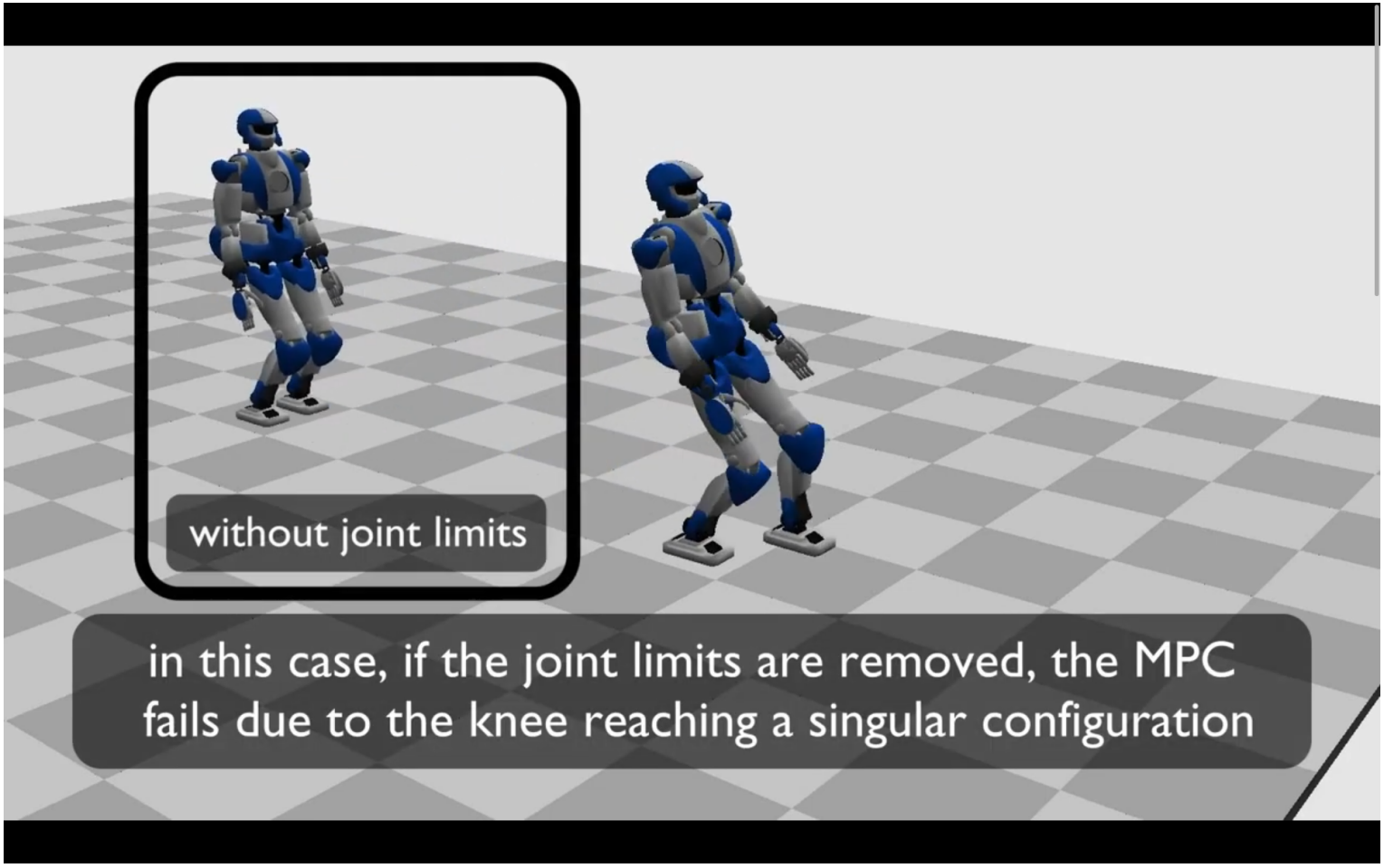
- the MPC looks like

$$\min_{\boldsymbol{w}^k} \sum_{i=k+1}^{k+C} \left\|\boldsymbol{r}^i - \boldsymbol{r}_d^i\right\|_{\boldsymbol{W}_r}^2 + \left\|\dot{\boldsymbol{q}}_j^i\right\|_{\boldsymbol{W}_{\dot{q}_j}}^2 + \left\|\boldsymbol{l}^i\right\|_{\boldsymbol{W}_l}^2 + \sum_{i=k}^{k+C-1} \left\|\dot{\boldsymbol{\nu}}^i\right\|_{\boldsymbol{W}_{\dot{\nu}}}^2$$

  subject to:
  - stability constraint,
  - ZMP constraints,
  - joint limits,
  - contact constraint

- we can also add a term that maximizes the MPC **feasibility** (i.e., the ability to find a solution to the optimization problem), which increases **robustness**

MPC using a whole-body kinematic model



https://www.youtube.com/embed/Fa6iy3mUcBY

LAAS-CNRS whole-body MPC

- some groups are experimenting with MPC using a **whole-body dynamic model**
- the optimization is usually solved using some **variant of DDP**, but instead of trying to reach convergence, a single DDP iteration is performed at every time-step
- this works because at every time-step the optimization is initialized using the **previous solution** as initial guess
- let's look at the implementation proposed by LAAS-CNRS [Dantec et al.]

Dantec et al., "Whole-Body Model Predictive Control for Biped Locomotion on a Torque-Controlled Humanoid Robot", International Conference on Humanoid Robots, 2022

LAAS-CNRS whole-body MPC

- the cost function includes:
  - state regularization: keep the state close to an **upright position**

$$(\boldsymbol{x} - \boldsymbol{x}^d)^T \boldsymbol{R}_x (\boldsymbol{x} - \boldsymbol{x}^d)$$

  - control regularization: keep the input close to **gravity-compensating torques**

$$(\boldsymbol{u} - \boldsymbol{u}^d)^T \boldsymbol{R}_u (\boldsymbol{u} - \boldsymbol{u}^d)$$

  - feet tracking: track predefined **feet trajectories** (using a log activation function)

$$a(\boldsymbol{p}(\boldsymbol{x}) - \boldsymbol{p}^d) \quad \text{with } a(\boldsymbol{r}) = \log\left(1 + \frac{||\boldsymbol{r}||}{\alpha}\right)$$

  - kinematic limits: since DDP doesn't allow for constraints, limits are added using **barrier functions**

$$\frac{1}{2}||\max(\boldsymbol{x} - \boldsymbol{x}_{\max}, \mathbf{0})||^2 + \frac{1}{2}||\min(\mathbf{x} - \mathbf{x}_{\min}, \mathbf{0})||^2$$

LAAS-CNRS whole-body MPC

- the cost function includes:
  - since DDP does not allow for constraints, inequality constraints on the **contact wrenches** $\boldsymbol{w}$ are substituted with a **tracking** term on the center of the constraints $\boldsymbol{w}_d$

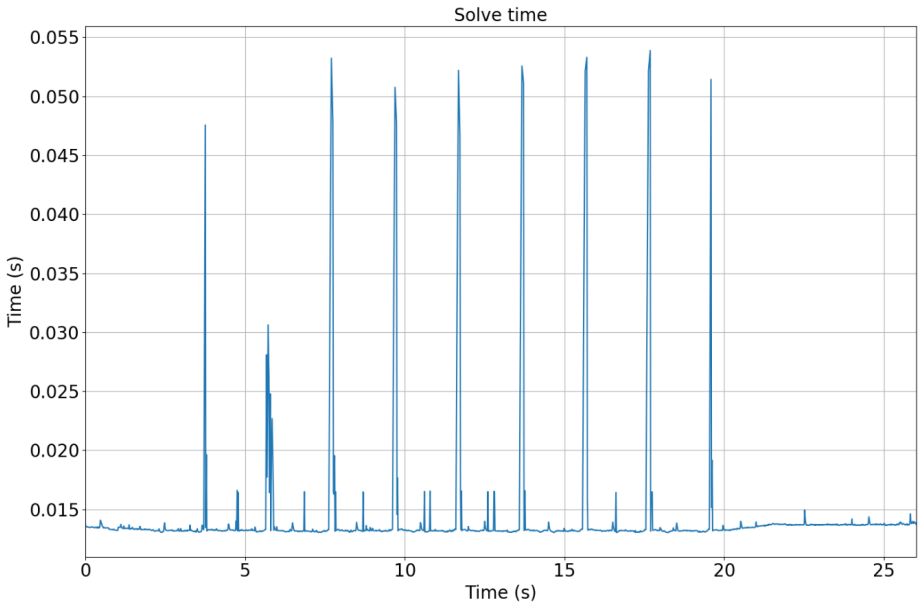
$$\begin{aligned} f_z &\geq 0, \\ |f_x| &\leq \bar{\mu} f_z, \\ |f_y| &\leq \bar{\mu} f_z, \\ |\tau_x| &\leq d_y f_z, \\ |\tau_y| &\leq d_x f_z, \end{aligned} \qquad \Longrightarrow \qquad ||\boldsymbol{A}(\boldsymbol{w} - \boldsymbol{w}_d(t))||^2$$

  - additionally, a tracking term is added to bring the **ZMP** close to the center of the foot

$$\left|\frac{\tau_y}{f_z}\right|^2 + \left|\frac{\tau_z}{f_z}\right|^2$$

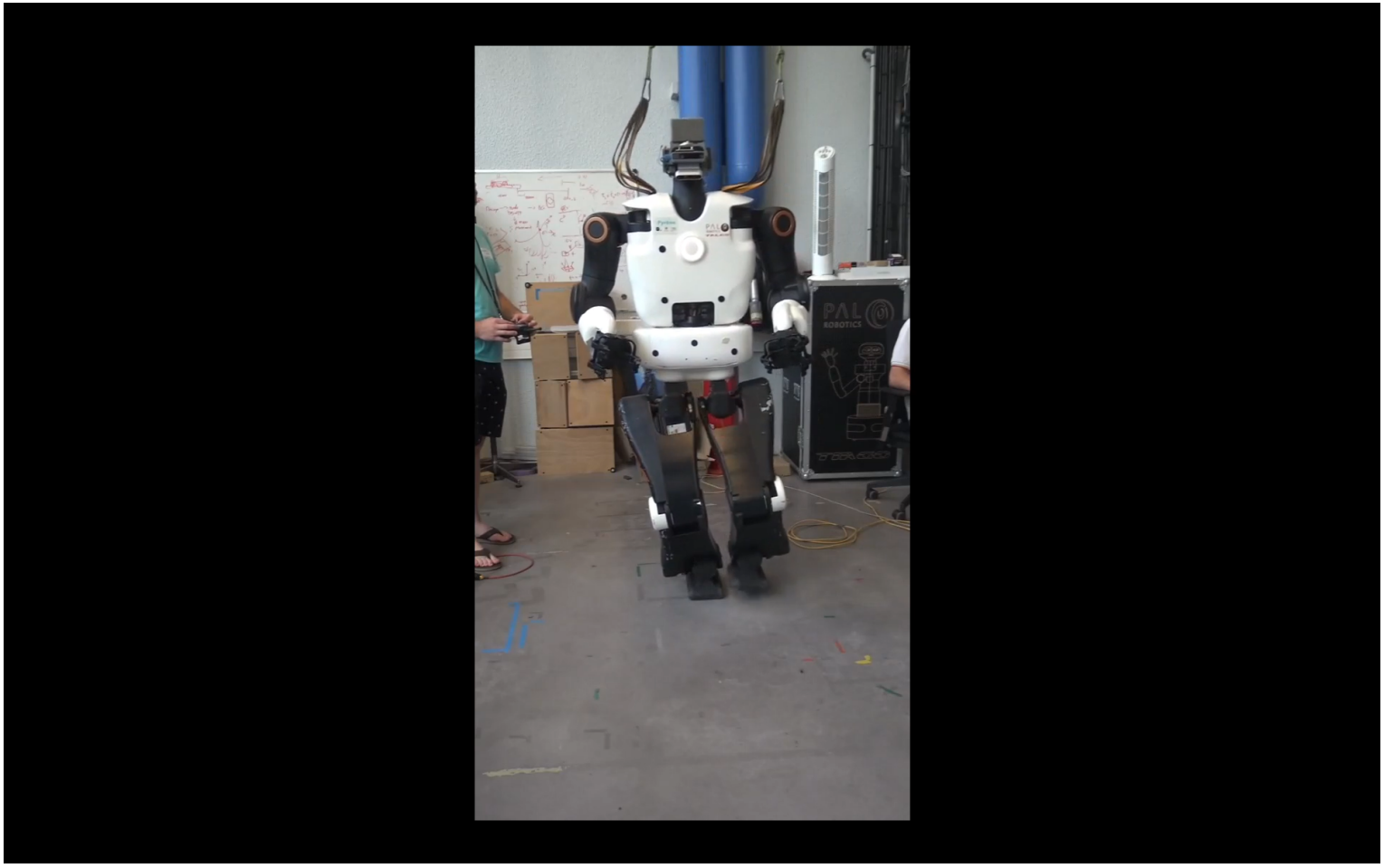
LAAS-CNRS whole-body MPC

- **one DDP iteration** iteration is performed at each cycle
- in this way, the authors achieve fast solution time (≈15 ms), but with occasional spikes (≈50 ms)



- to avoid problems during the spikes, the authors implemented a policy that sends commands at a higher frequency, using the **Riccati gains** found during the DDP iteration

LAAS-CNRS whole-body MPC



https://peertube.laas.fr/w/r4uA52hrEHfErjdTN5GyZV

MIT whole-body MPC

- solve a **single** NMPC problem over the **full robot state** (floating base + joints) and **contact forces**
- **algorithmic differentiation** of the dynamics, cost and constraints using **casADi**
- fast solution: linearize the problem and iterate only once, using a general purpose **sparse** QP solver: OSQP

Khazoom et al., "Tailoring Solution Accuracy for Fast Whole-body Model Predictive Control of Legged Robots", IEEE RA-L, 2024

MIT whole-body MPC

- decision variables over the horizon ($k = 0, \ldots, N-1$):

$$\mathcal{X} := \{(\boldsymbol{q}_k, \boldsymbol{v}_k),\ \boldsymbol{\tau}_k,\ \{\boldsymbol{F}_{c,k}\}_{c \in \mathcal{C}}\}$$

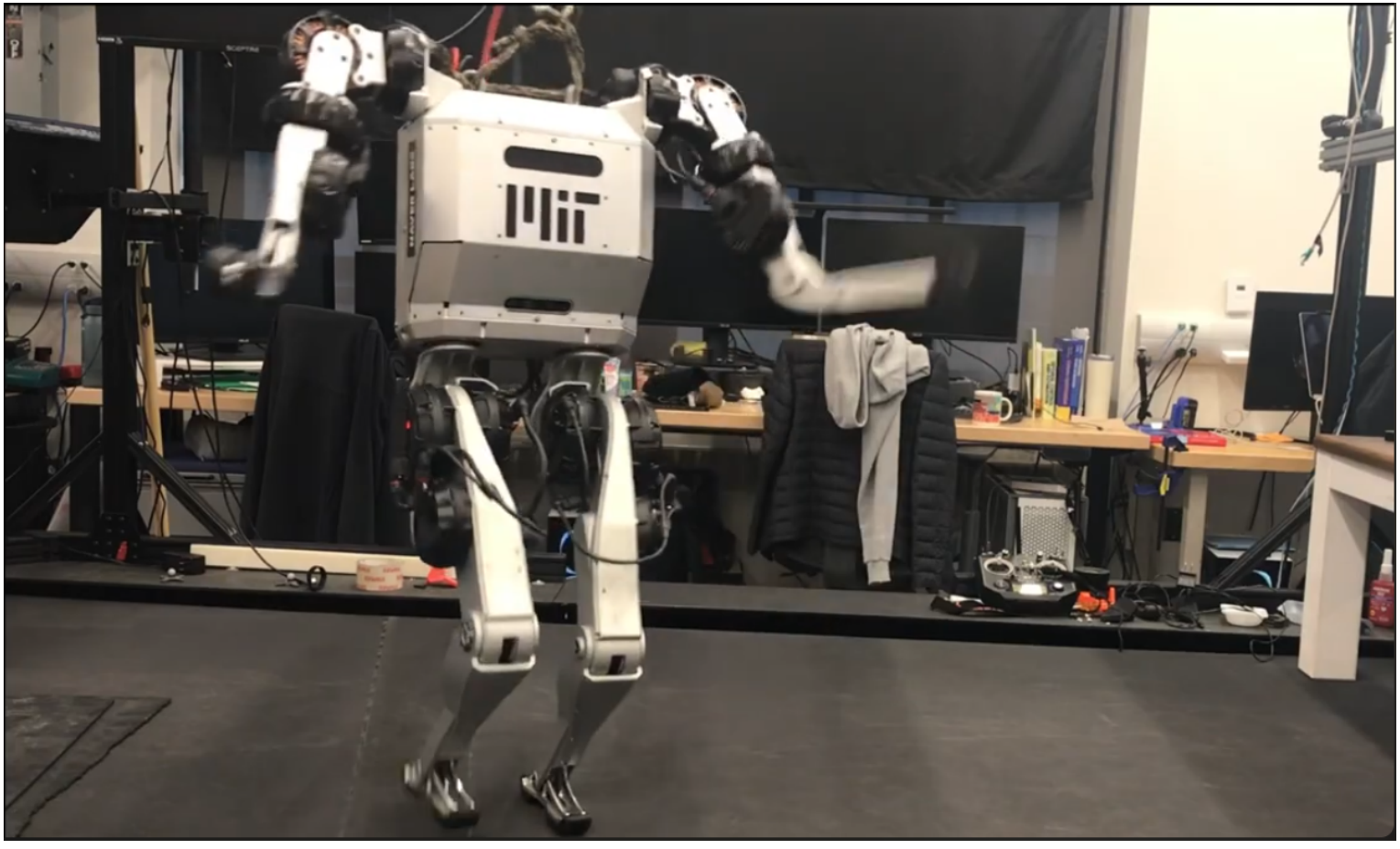
  - $\boldsymbol{q}$: floating-base pose + joint positions, $\boldsymbol{v}$: generalized velocity
  - $\boldsymbol{\tau}$: joint torques, $\boldsymbol{F}_c$: contact forces (multiple contact points per foot)

- constraints include (typical examples):
  - discretized dynamics $\boldsymbol{q}_{k+1} = f(\boldsymbol{q}_k, \boldsymbol{v}_k)$
  - inverse dynamics enforced via RNEA, with contact Jacobians
  - stance/swing constraints and friction cone inequalities
  - box constraints on $\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\tau}$ and **self-collision inequalities** $h(\boldsymbol{q}_k) \geq 0$

MIT whole-body MPC

- increasing QP accuracy (more ADMM iterations) **does not always improve** real-robot performance
- main limiting factors:
  - **inertial/modeling errors** (mass/CoM/inertia uncertainty)
  - **discretization error** (coarse integration steps used for real-time NMPC)
  - **computation delay** (the more you iterate, the more delay you introduce)
- takeaway: use the computational budget to get **fast, repeatable updates** rather than highly accurate QP solves

MIT whole-body MPC



https://youtu.be/Xmi7AFGhlFc

ETH whole-body MPC + manipulation

- MPC that optimizes **whole-body motion + contact forces** for both
  - dynamic locomotion
  - manipulation with the arm (object interaction)

- robot model: centroidal dynamics + full kinematics

$$\dot{\boldsymbol{h}}_{\text{com}} = \begin{bmatrix} \sum_{i=1}^{n_c} \boldsymbol{f}_{c_i} + m\mathbf{g} \\ \sum_{i=1}^{n_c} \boldsymbol{r}_{\text{com},c_i} \times \boldsymbol{f}_{c_i} + \boldsymbol{\tau}_{c_i} \end{bmatrix}, \qquad \boldsymbol{h}_{\text{com}} = \boldsymbol{A}(\boldsymbol{q})\dot{\boldsymbol{q}}$$

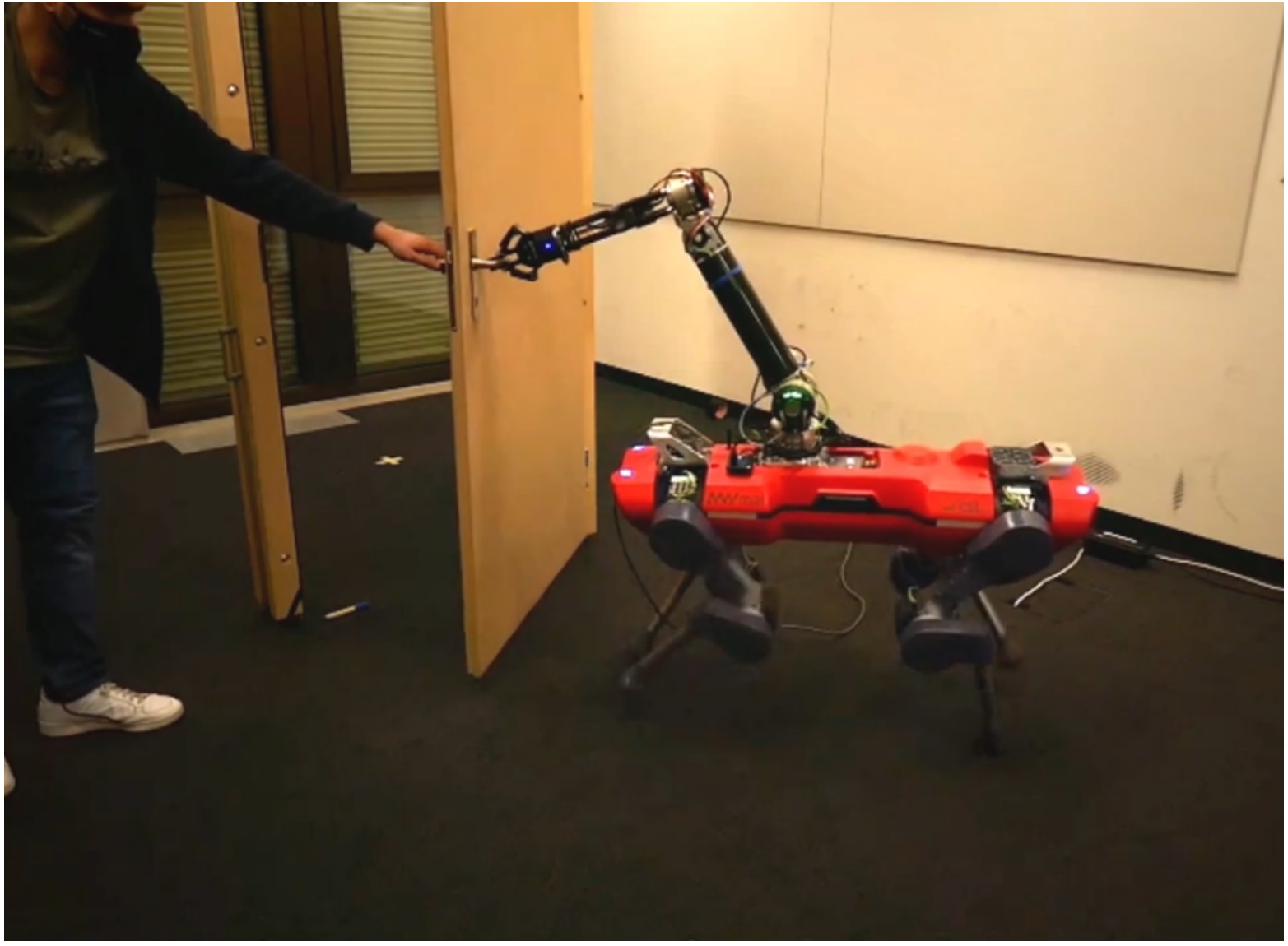
- the whole-body motion contributes to the dynamics via kinematic mapping to the CoM and angular momentum (centroidal map)

Sleiman et al., "A Unified MPC Framework for Whole-Body Dynamic Locomotion and Manipulation", IEEE RA-L, 2021

ETH whole-body MPC + manipulation

- in locomotion, typically we predefine the **contact schedule**: the timing with which the robot steps
- if we ask the optimizer to also determine contact switches, the problem becomes much harder (**hybrid** dynamic system)
- in this MPC, the authors also predefine the contact schedule for the manipulation task

ETH whole-body MPC + manipulation



https://youtu.be/uT4vpNDzUvI

data-driven methods

- model-based methods are **data-efficient** and interpretable, but rely on:
  - accurate inertial parameters, contact models, actuator models
  - correct handling of impacts, friction, compliance, delays
- data-driven methods try to **learn what is hard to model**:
  - contact-rich stabilizing feedback
  - disturbance recovery and adaptation to unmodeled effects
  - coordination of whole-body motion (arms, torso, feet) for robustness
- key trade-off: **less modeling** $\Rightarrow$ **more data** (and more care about sim-to-real)

deep reinforcement learning

- the basic picture:
  - the policy (controller) π_θ is a **neural network** with parameters θ, that outputs an action $\boldsymbol{a}$ (or a distribution)
  - the robot moves in an environment commanded by the policy (**rollout**) and visits states $\boldsymbol{s}$
  - depending on what it does, it receives a **reward** $r(\boldsymbol{s}, \boldsymbol{a})$

- we collect data, then improve the policy so to maximize the **expected return** (sum of the rewards) using **gradient-based optimization**

$$\max_\theta \ \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t) \right]$$

- future rewards are discounted by a factor γ

deep reinforcement learning

- in RL there are two main **paradigms**
- **on-policy** algorithms (policy gradient, PPO, etc...):
  - collect data under the current policy (one or more rollouts)
  - compute returns for each rollout
  - modify the policy to make some actions more or less probable
- **off-policy** algorithms (DDPG, SAC, etc...)
  - accumulate data (with any policy) into a buffer
  - learn a Q-function that maps states-actions to future rewards
  - train a policy to maximize this Q-function
- off-policy algorithms are typically more data-efficient but can be more unstable because they have more moving parts

deep reinforcement learning

- RL requires a lot of **data**, and we cannot collect it on real robots because they would break
- solution: perform many parallel **simulations** on GPU (Isaac, Mujoco, etc...)
- to improve **sim-to-real**, perform domain randomization (simulations have different parameters)

student/teacher RL

- if rewards are sparse, algorithms could wander forever and **not find** trajectories that give rewards
- observations might be difficult to interpret (camera images, lidar measurements, etc...)
- solution: **cheat!** train a policy in a simulated environment using privileged information (full robot state and map)
- then **distill** a second policy to reproduce the output of the first using less information (e.g., only camera)

student/teacher RL

Radosavovic et al., "Real-world humanoid locomotion with reinforcement learning", Science Robotics, 2024

`https://youtu.be/eFoBfFhwo18`

RL: typical problems

- a common walking reward is a weighted sum of many terms
- problem: reward design is **difficult**
  - small mistakes $\Rightarrow$ unnatural gaits (asymmetric, hopping, foot sliding)
  - the policy can **hack the reward** rather than learn robust locomotion
  - sometimes the policy can **hack the simulator** and perform very non-physical actions
- pure RL is often not the best/easiest choice

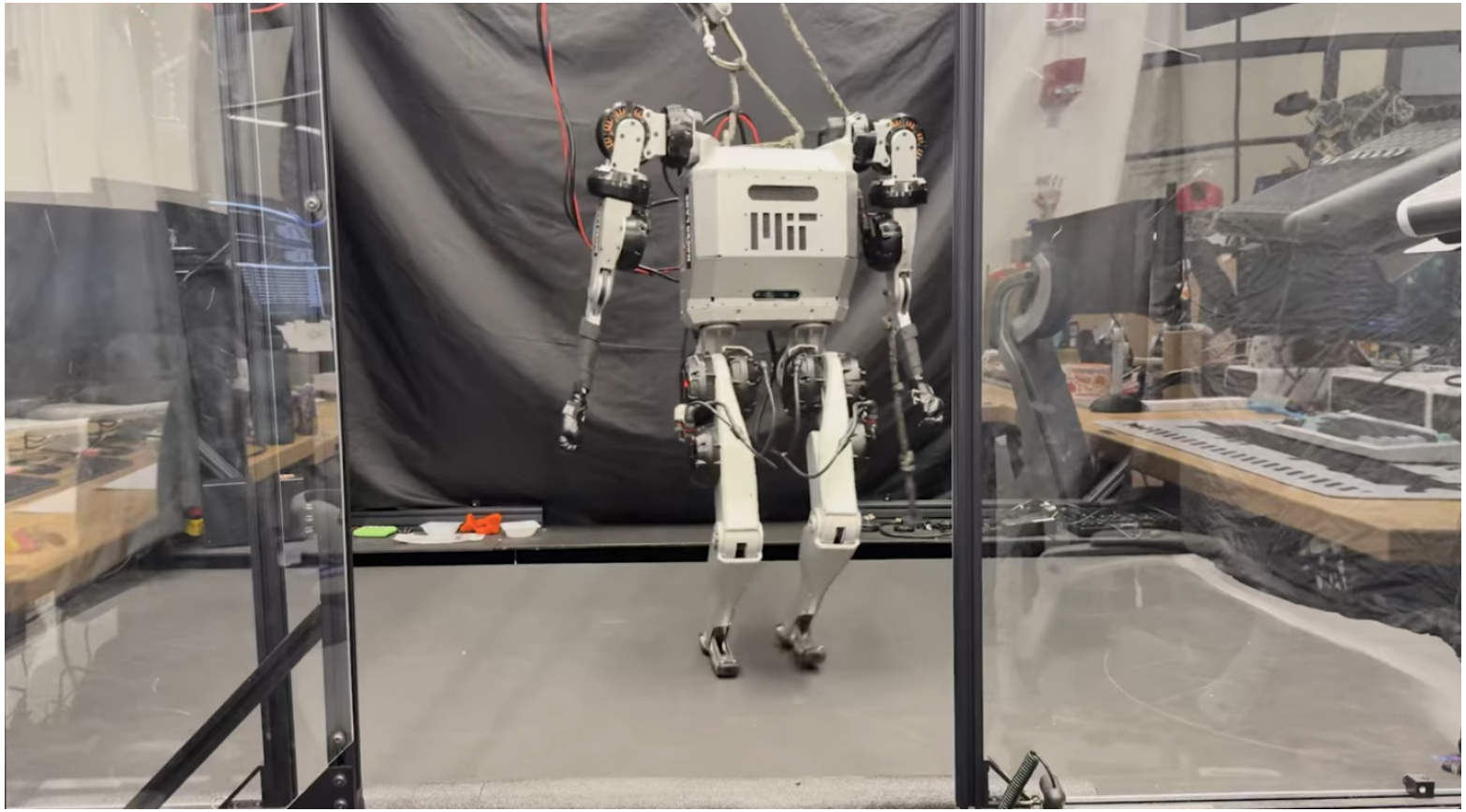
combining MPC and RL

- why learn a policy from scratch when we have working controllers? we could just learn the parts that are difficult to model

- **residual RL**: control is $\boldsymbol{u} = \boldsymbol{u}_{\text{mpc}} + \Delta \boldsymbol{u}_{\theta}$
  - the model-based part $\boldsymbol{u}_{\text{mpc}}$ generates basic motion
  - the learned residual compensates unmodeled effects (contacts, compliance, delays)

- motivation: easier to train/interpret/modify

combining MPC and RL



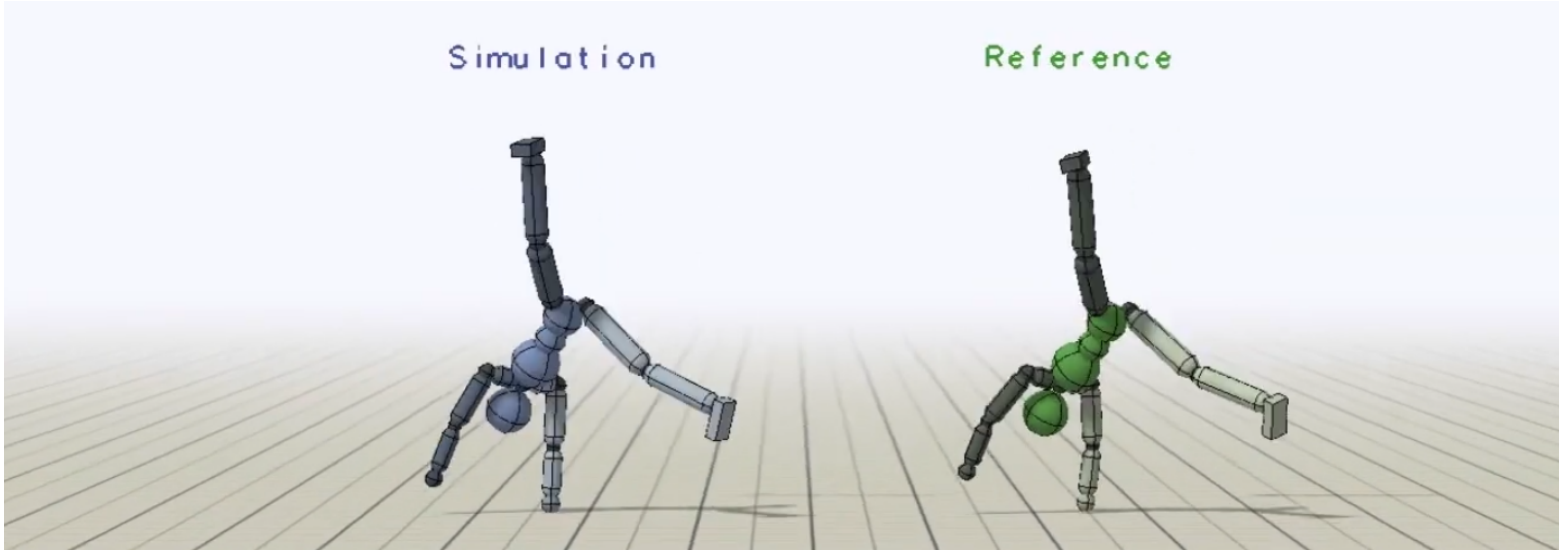
Jeon et al., "Residual MPC: Blending Reinforcement Learning with GPU-Parallelized Model Predictive Control", preprint, 2025 https://youtu.be/L2NrPD4yiMs

imitation learning

- idea: add an **imitation objective** to stabilize learning and improve style
- motion can be collected with motion capture devices from humans, or identified from video
- humans and robots are different: motions need **retargeting** to match the robot kinematic structure
- benefit: much less reward hacking; policies learn **humanlike gaits** and still discover **recoveries**

Peng et al., "DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills", SIGGRAPH, 2018

imitation learning



https://youtu.be/vppFvq2quQ0

imitation learning

https://youtu.be/05GphCrjx98