Lecture 8: Humanoid Robot Control

Nicola Scianca

December 2024

automatic footstep placement

- when using a linear simplified model, the balance condition is expressed as a **linear constraint** on the **ZMP**

$$p_z^{k+i,\text{foot}} \leq \boldsymbol{C}\boldsymbol{x}_{k+i} \leq p_z^{k+i,\max}$$

- if the footprint has size $2d$, its edges can be written as the foot position p_f^{k+i} plus or minus half the foot size

$$p_f^{k+i} - d \leq \boldsymbol{C}\boldsymbol{x}_{k+i} \leq p_f^{k+i} + d$$

- this means that if we consider the foot position as an **optimization variable**, the ZMP constraint is still **linear**

$$-d \leq \boldsymbol{C}\boldsymbol{x}_{k+i} - p_f^{k+i} \leq d$$

automatic footstep placement

- we can formulate an MPC with with the footstep positions as additional optimization variables [Herdt at al.], that still only has linear constraints

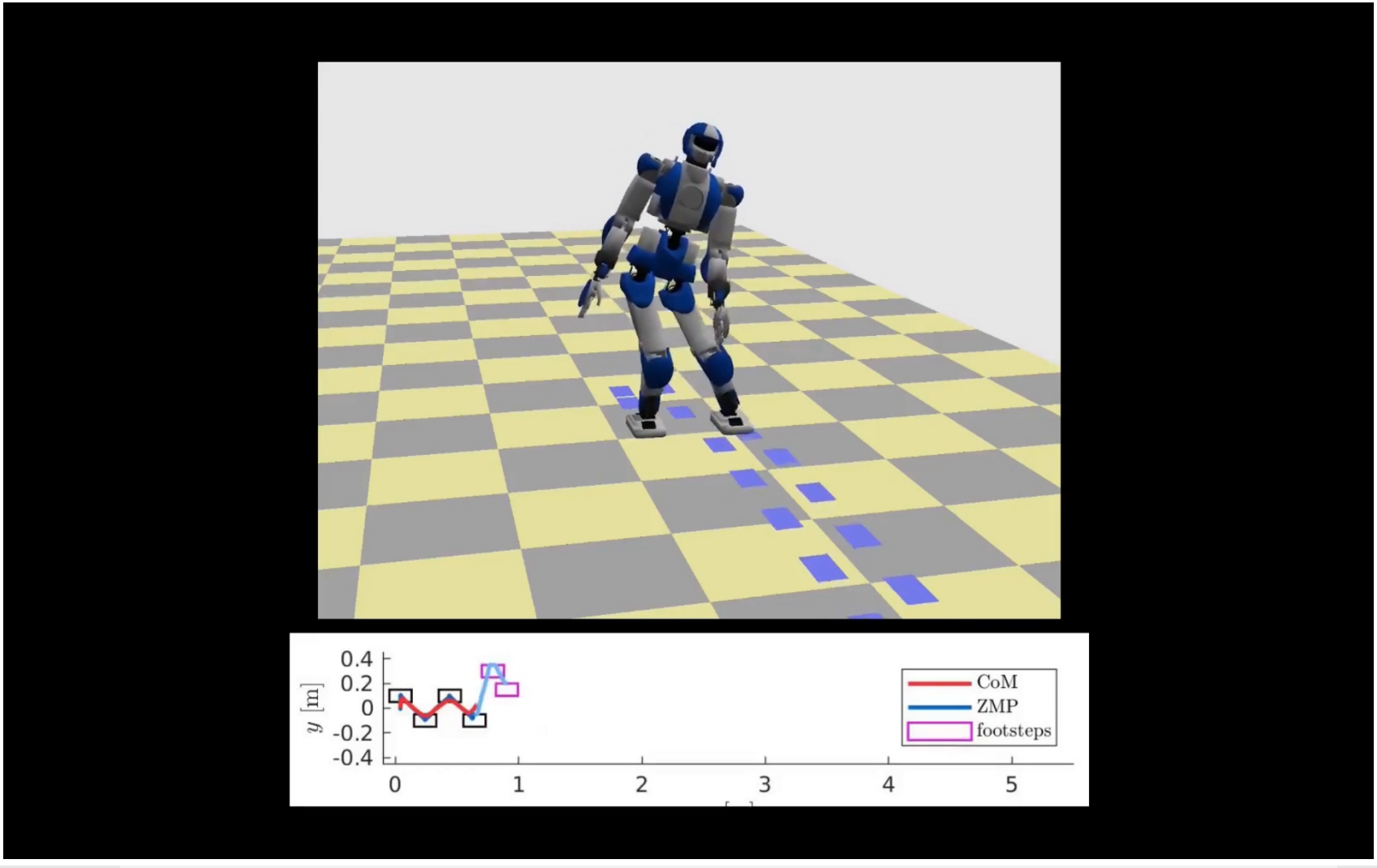
- if balance can't be maintained by simply changing the ZMP and CoM trajectory, the MPC can decide to **move the the foot** to a different location

- this achieves **automatic footstep placement**, and allows the robot to be more robust to disturbances

Herdt et al., "Walking without thinking about it",
International Conference on Intelligent Robots and Systems, 2010

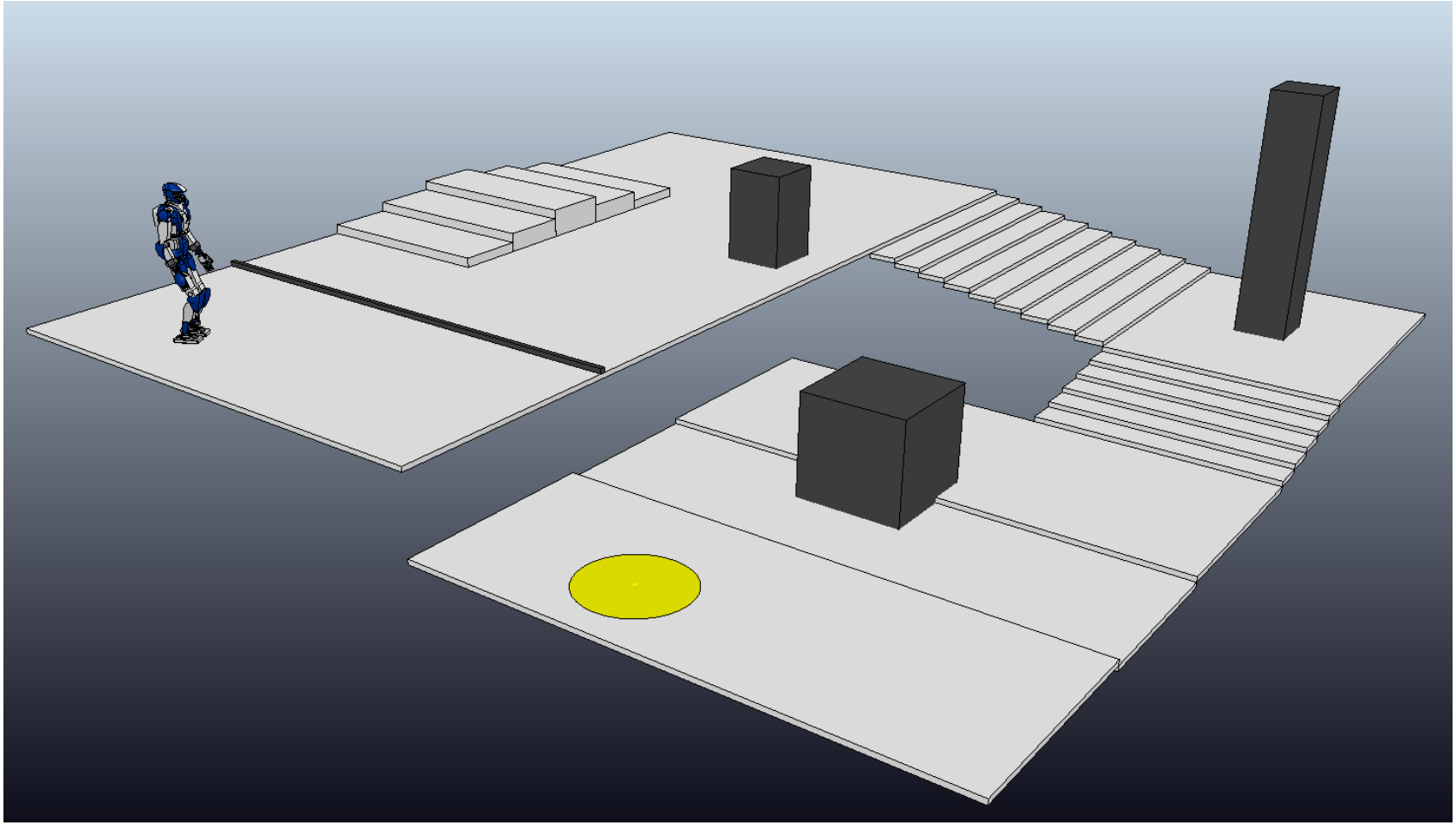
automatic footstep placement

- an example of automatic footstep placement: the robot adapts after getting pushed (not from the previous paper)



locomotion in complex environments

- what if we want the robot to move through a complex environment, with **stairs** and **obstacles**?



locomotion in complex environments

- simple footstep planning or automatic footstep placement are **not enough**: the footsteps must be placed carefully!
- one approach is to use **Rapidly-exploring Random Trees** (RRT), to build a tree of possible paths until we find a path that reaches the goal region [Cipriano et al.]
- we can keep refining the tree and look for the path that satisfies an **optimality criterion** (RRT$*$)

Cipriano et al., "Humanoid motion generation in a world of stairs", Robotics and Autonomous Systems, 2023

locomotion in complex environments

- the other problem we encounter is that the LIP model is not sufficient, because it assumes that the CoM height is constant

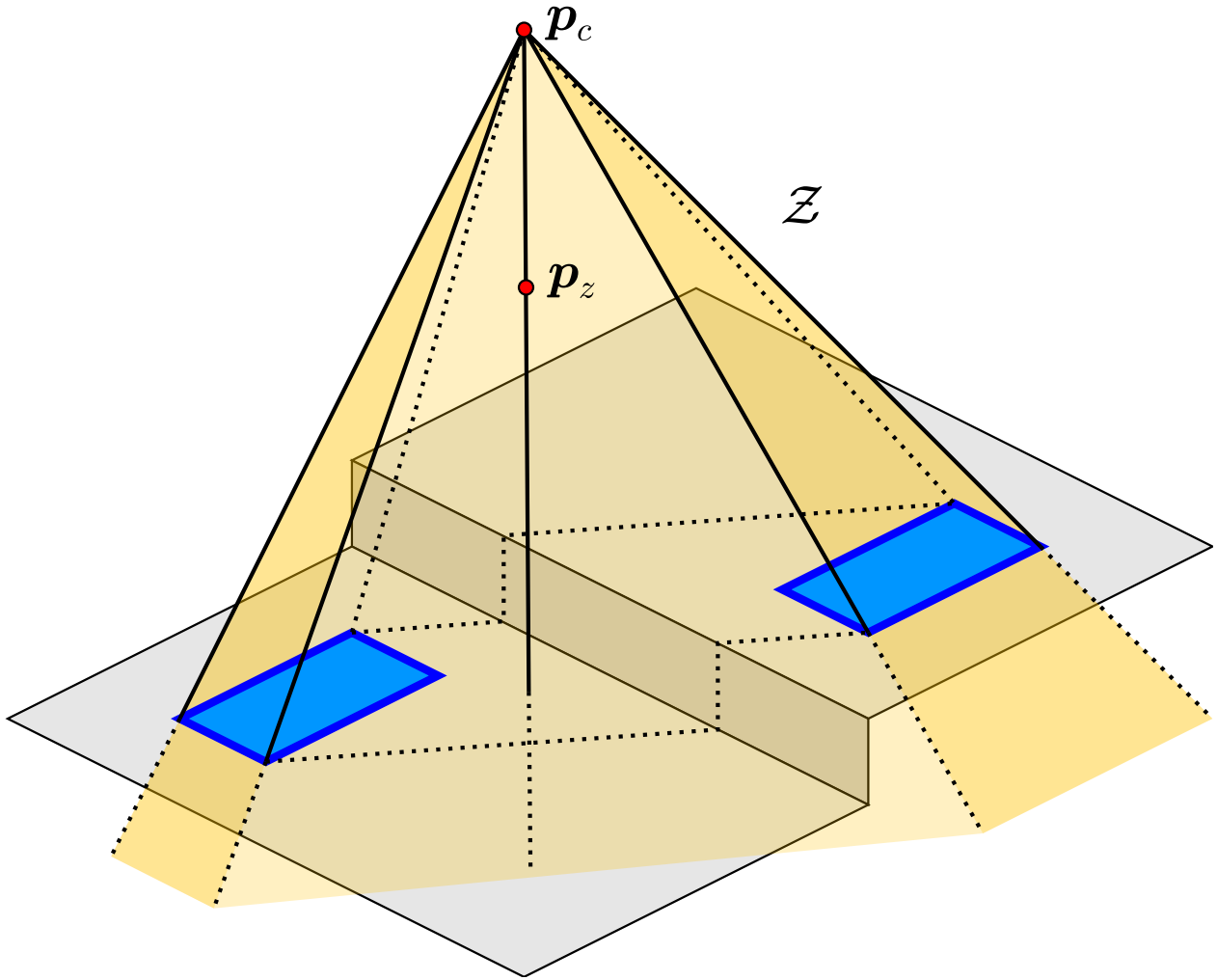
- we have a variant of the LIP model that can generate **3D trajectories** while maintaining linearity

$$\ddot{\boldsymbol{p}}_c = a(\boldsymbol{p}_c - \boldsymbol{p}_z) + \boldsymbol{g}$$

- the only issue with this model is that the ZMP now moves in **3D space**, therefore it's not obvious how "keep the ZMP in the support polygon" should be extended here

3D ZMP dynamics

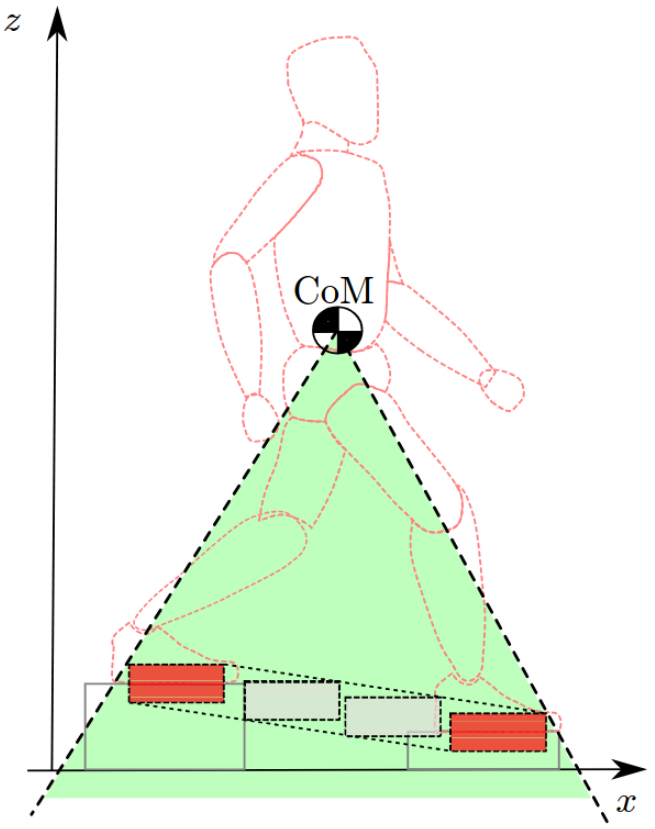
- the equivalent condition states that this 3D ZMP must be inside a **pyramidal region**, with the CoM as its vertex



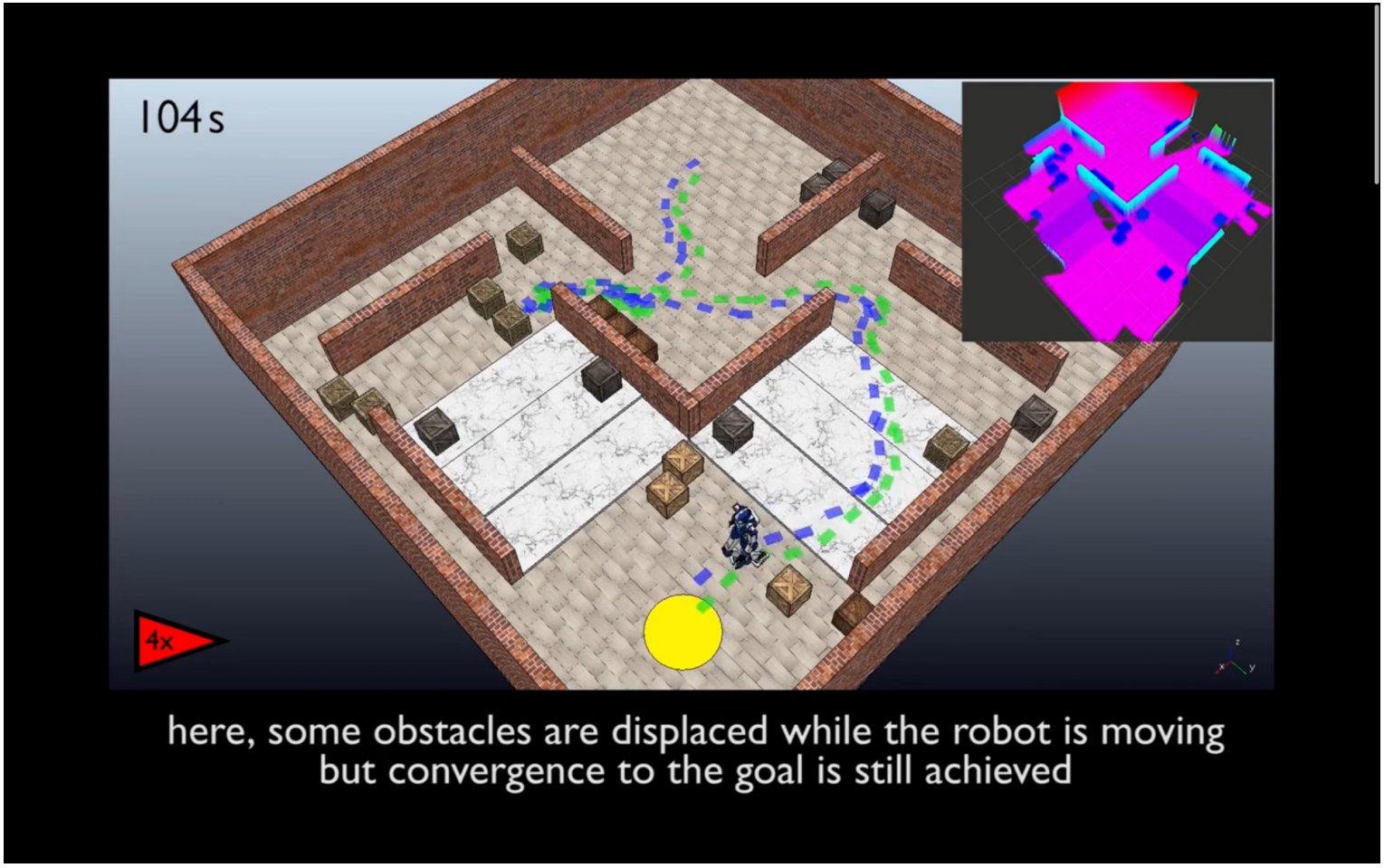
outline

- this balance condition is nonlinear, because the position of the CoM (the top of the pyramid) is a **predicted variable**

- however, we can approximate it by picking a smaller region inside the pyramid, such as for example a **moving box**



locomotion in complex environments



https://www.youtube.com/embed/BF43qUcx4gY

quadruped control using SRBD

- the next example uses **Single-Rigid-Body Dynamics** (SRBD) to control a quadruped [Di Carlo et al.]
- some additional simplifications are performed on the dynamic model we saw
- short prediction horizon (0.33-0.5 s), optimization recomputed at 25-50 Hz using the solver qpOASES

Di Carlo et al., "Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control" , International Conference on Intelligent Robots and Systems, 2018

quadruped control using SRBD

- the SRBD model includes nonlinear terms: we neglect the second term which is usually small

$$\boldsymbol{I}\dot{\boldsymbol{\omega}} + \cancel{\boldsymbol{\omega} \times (\boldsymbol{I}\boldsymbol{\omega})} = \sum_i (\boldsymbol{p}_i - \boldsymbol{p}_c) \times \boldsymbol{f}_i$$

- $\boldsymbol{I}$ represents the inertia tensor in the world frame, which can be written in terms of the (constant) inertia tensor in the local frame as $\bar{\boldsymbol{I}}$ and the rotation matrix of the robot orientation $\boldsymbol{R}$

$$\boldsymbol{R}\bar{\boldsymbol{I}}\boldsymbol{R}^T\dot{\boldsymbol{\omega}} = \sum_i (\boldsymbol{p}_i - \boldsymbol{p}_c) \times \boldsymbol{f}_i$$

- the authors approximate $\boldsymbol{R}$ with the rotation matrix around the z-axis $\boldsymbol{R}_z$, as the other rotations are usually small
- $\boldsymbol{R}_z$ is assumed to be the yaw of the planned footsteps, so the model becomes **linear time-varying**

quadruped control using SRBD

- the MPC is written as

$$\min_{\boldsymbol{x},\boldsymbol{u}} ||\boldsymbol{x}_{k+1} - \boldsymbol{x}_{k+1}^{\text{ref}}||^2 + ||\boldsymbol{u}_k||^2$$

$$\text{s.t. initial state constraint}$$

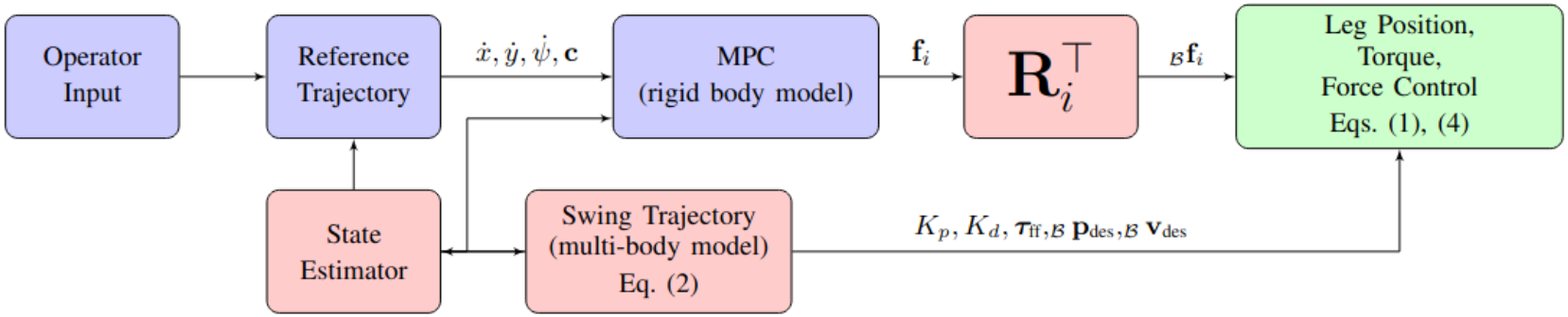
$$\text{dynamics constraints}$$

$$\text{force constraints}$$

$$\text{force selection constraints}$$

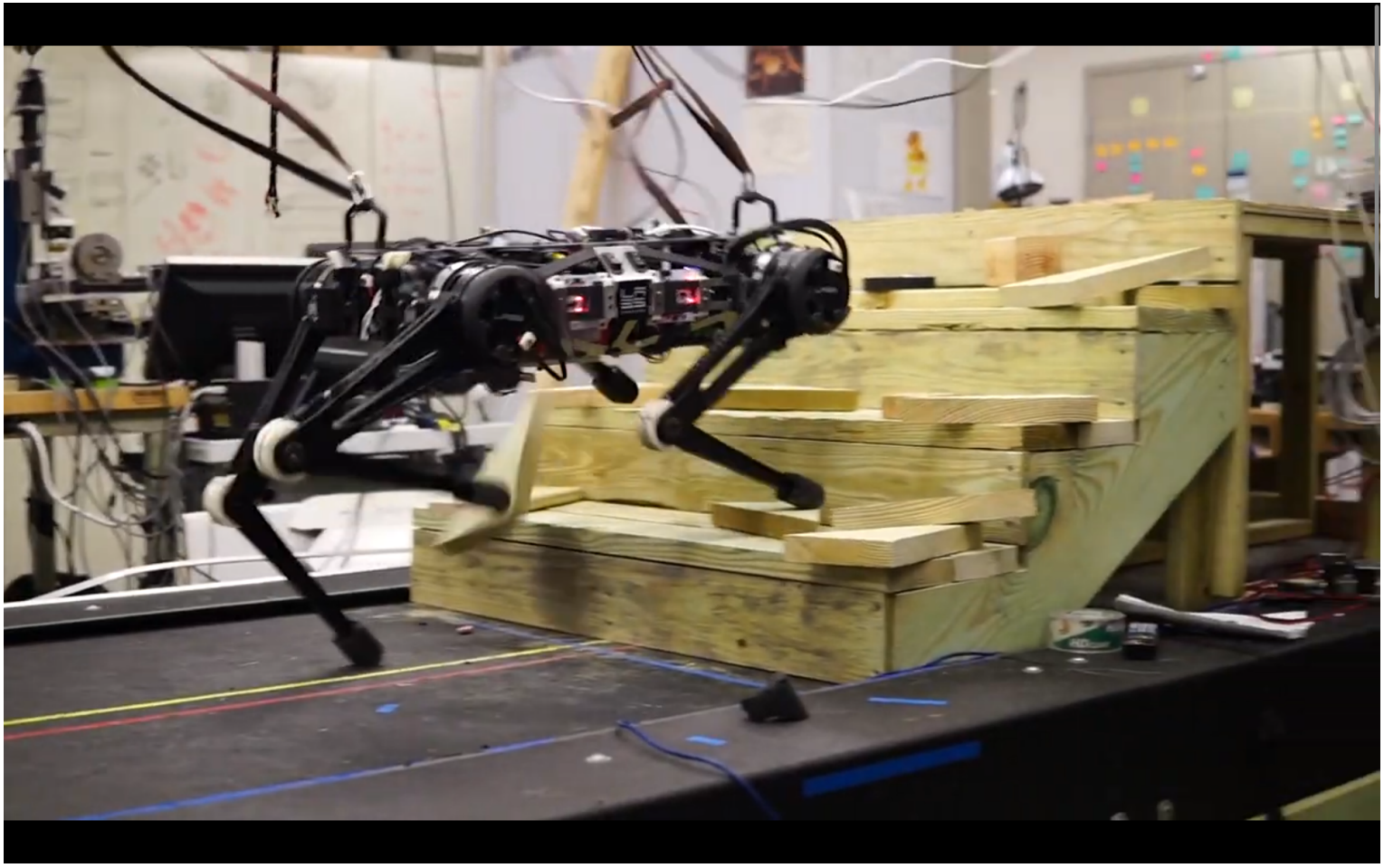
- **force constraints** are the usual unilaterality + friction (no contact torques because the quadruped feet are **point-like**)
- **force selection constraints** impose that the feet not in contact with the ground have **zero force**

quadruped control using SRBD



- an operator gives a command with a videogame controller, which is used to build a simple reference trajectory
- the MPC provide the inputs $\boldsymbol{u}$, i.e, the **contact forces**
- a **whole-body controller** finds the **joint torques** to realize these contact forces and the kinematic tasks for the legs

quadruped control using SRBD



https://www.youtube.com/embed/q6zxCvCxhic

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** $(\bar{\boldsymbol{q}}_0, \dots, \bar{\boldsymbol{q}}_N)$, $(\bar{\boldsymbol{\nu}}_0, \dots, \bar{\boldsymbol{\nu}}_N)$

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

- 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

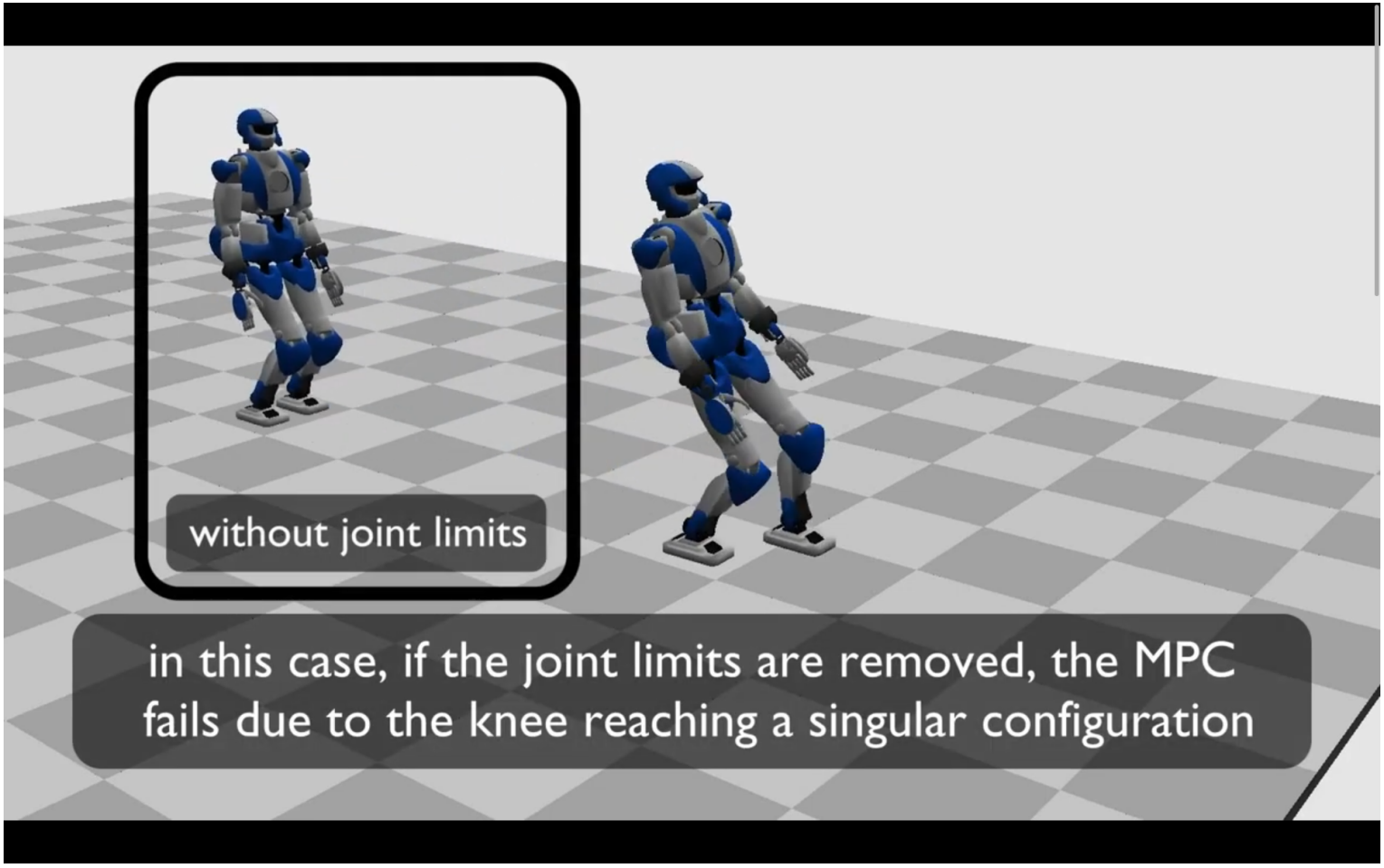
- the MPC looks like

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

  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

MPC using a whole-body dynamic model

- 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

MPC using a whole-body dynamic model

- 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**

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

MPC using a whole-body dynamic model

- 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 $\boldsymbol{w}_d$

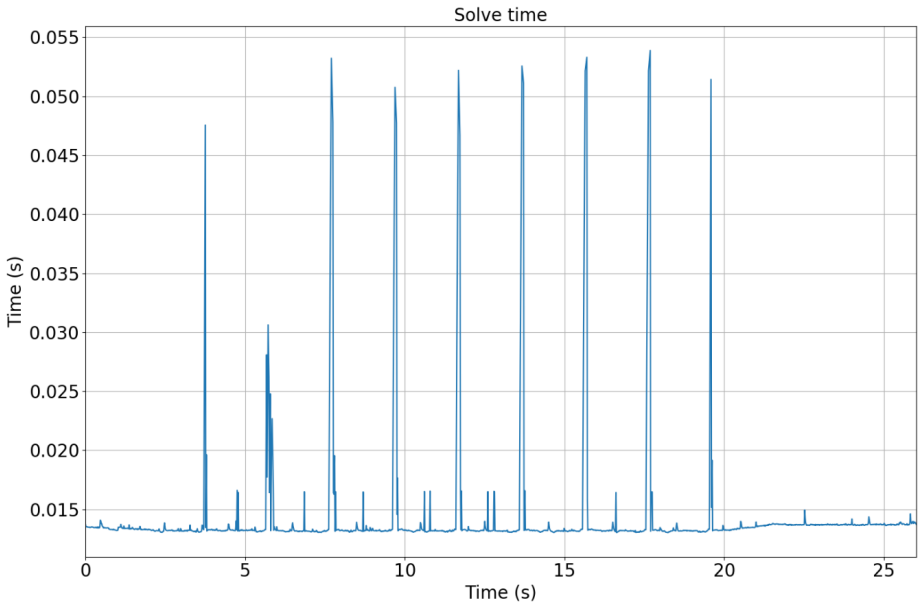
$$\begin{aligned} f_z &\geq 0, \\ |f_x| &\leq \bar{\mu} f_z, \\ |f_y| &\leq \bar{\mu} f_z, \\ |\tau_x| &\leq d_y f_z, \\ |\tau_y| &\leq d_x f_z, \end{aligned} \qquad \Longrightarrow \qquad ||\boldsymbol{A}(\boldsymbol{w} - \boldsymbol{w}_d(t))||^2$$

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

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

MPC using a whole-body dynamic model

- **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

MPC using a whole-body dynamic model

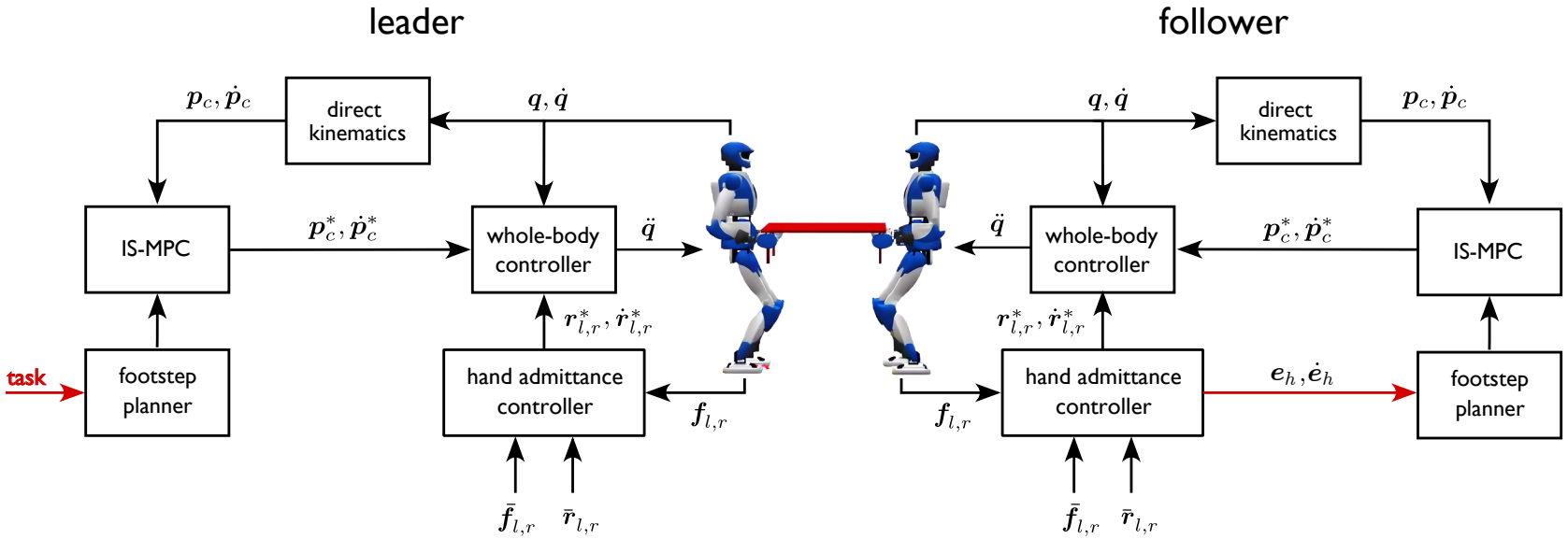
https://peertube.laas.fr/w/r4uA52hrEHfErjdTN5GyZV

not only locomotion

- these lessons have been mostly concerned with locomotion, which is the most challenging task
- however, the goal is to use legged robots to perform useful **real-life tasks**, such as transportation, manipulation, etc...
- let's see an example of **cooperative transportation** of a heavy object using MPC

not only locomotion

- the collaboration takes place between two robots: a **leader** who knows the task and a **follower** who is unaware
- the controller is **decentralized**, so the robots move independently of one another
- the follower measures **interaction forces** and simply tries to maintain balance



not only locomotion

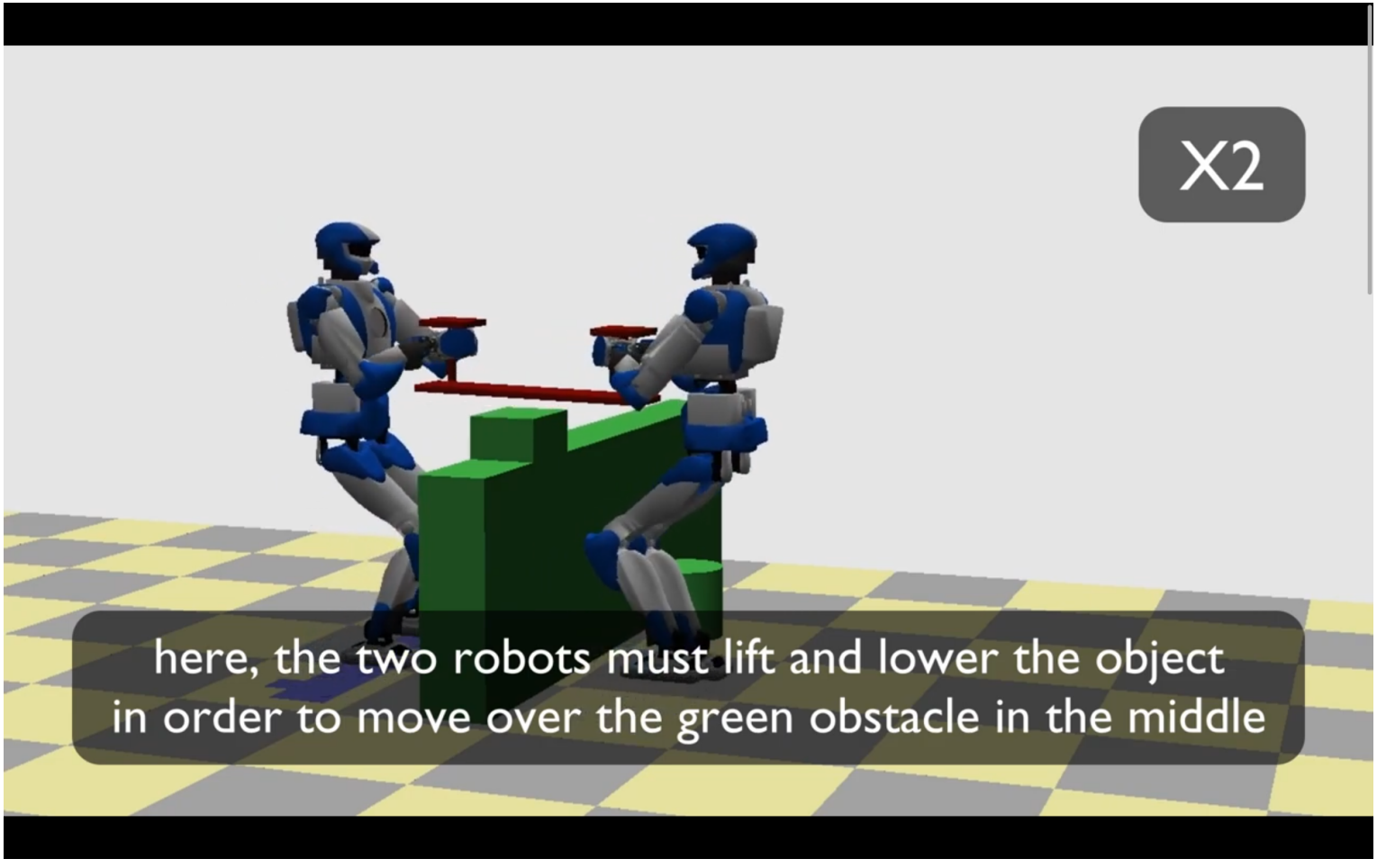
- both robots use **admittance control** to have a compliant grasp of the transported object

$$\underbrace{\boldsymbol{M}\ddot{\boldsymbol{r}}_i + \boldsymbol{C}\dot{\boldsymbol{r}}_i + \boldsymbol{K}(\boldsymbol{r}_i - \bar{\boldsymbol{r}}_i)}_{\text{virtual spring-mass-damper}} = \boldsymbol{R}_{\mathcal{F}}^T(\underbrace{\boldsymbol{f}_i}_{\text{measured interaction force}} - \bar{\boldsymbol{f}}_i)$$

- when the admittance displaces the hands with respect to a neutral position, the **footsteps are replanned** in the same direction

- if a robot gets pushed, first it reacts with the **hands**, then with the **whole body**

not only locomotion



https://www.youtube.com/embed/ChfhpSJFqg0