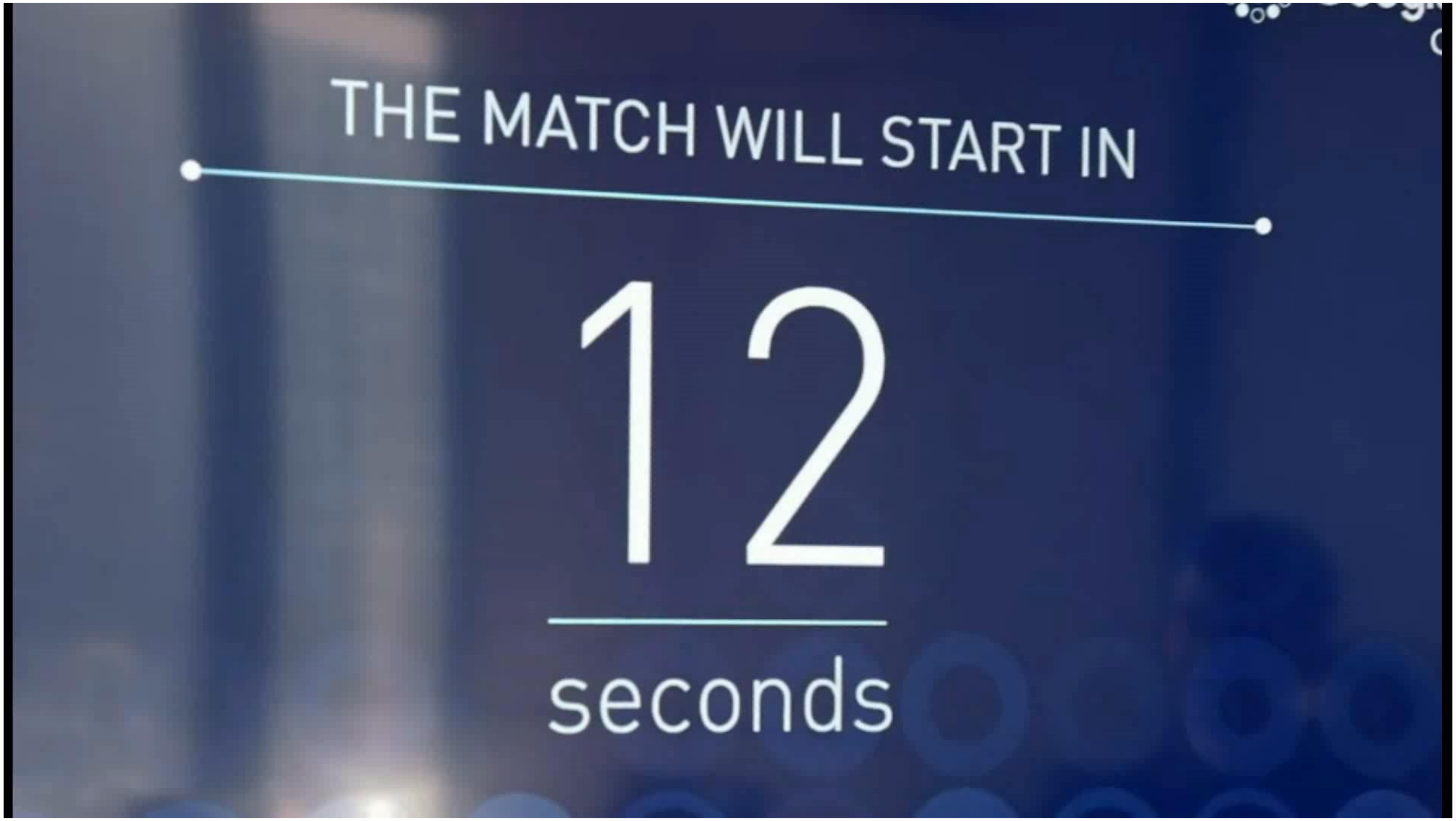
# Robotics

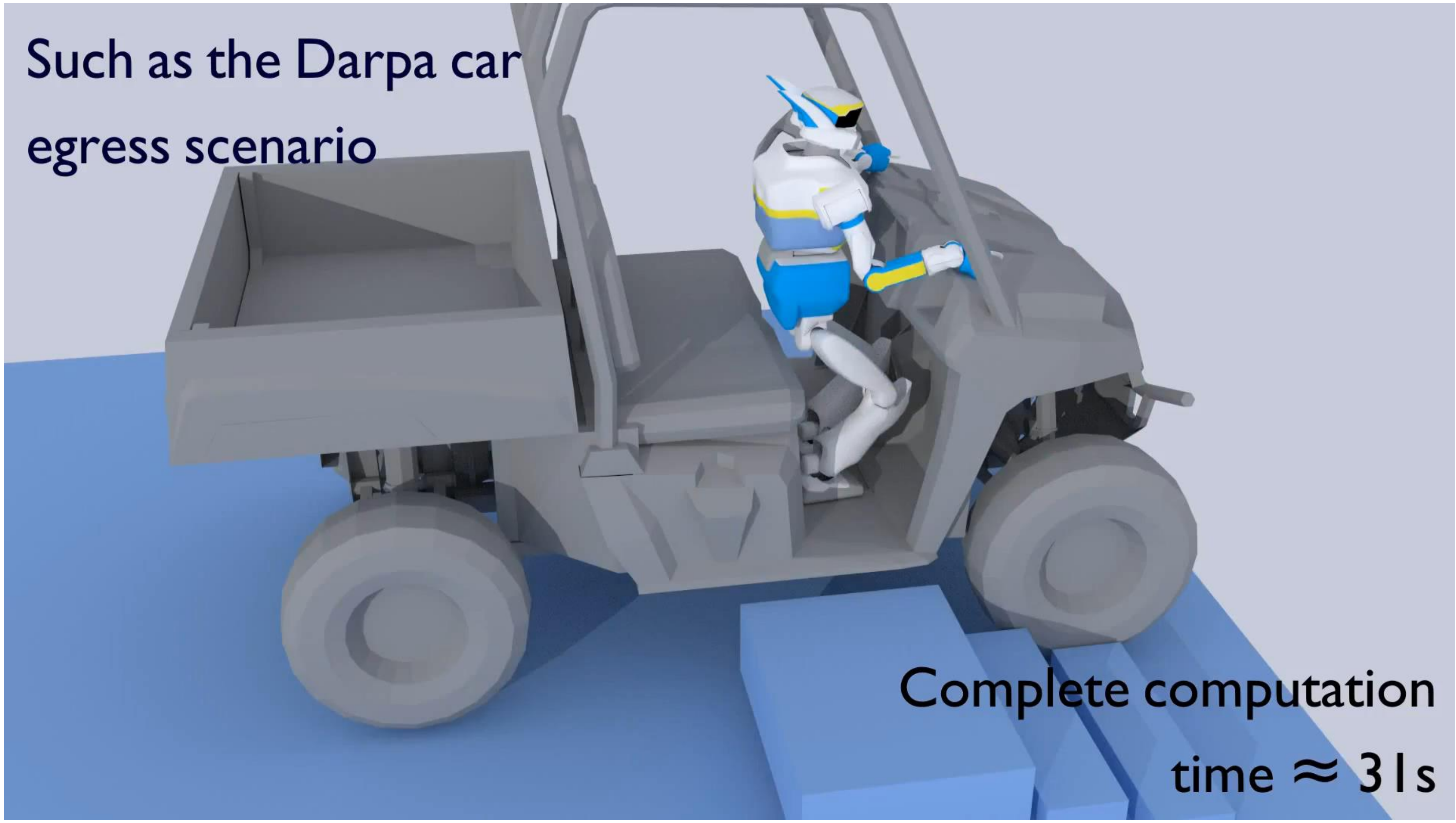
## Motion generation for agile robots

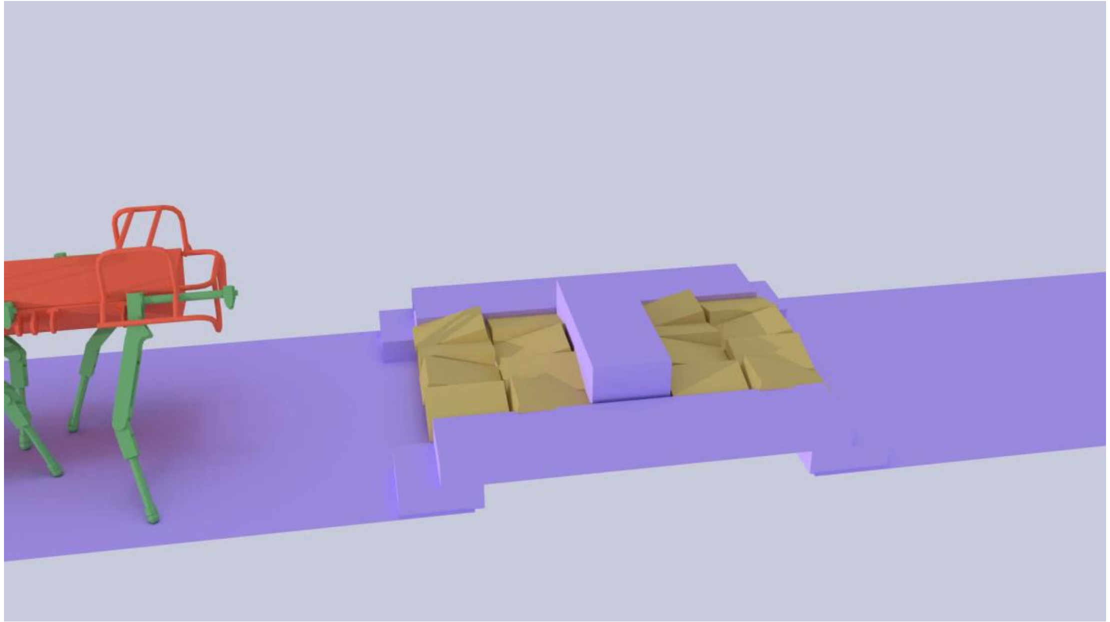
Nicolas Mansard

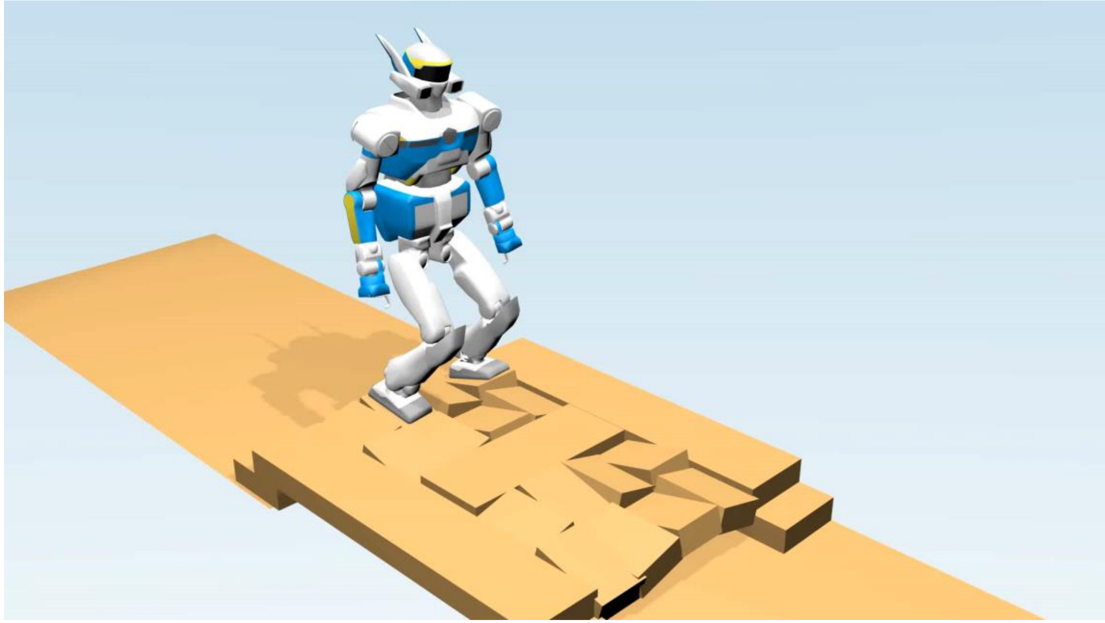
# Robotics as a ML problem?

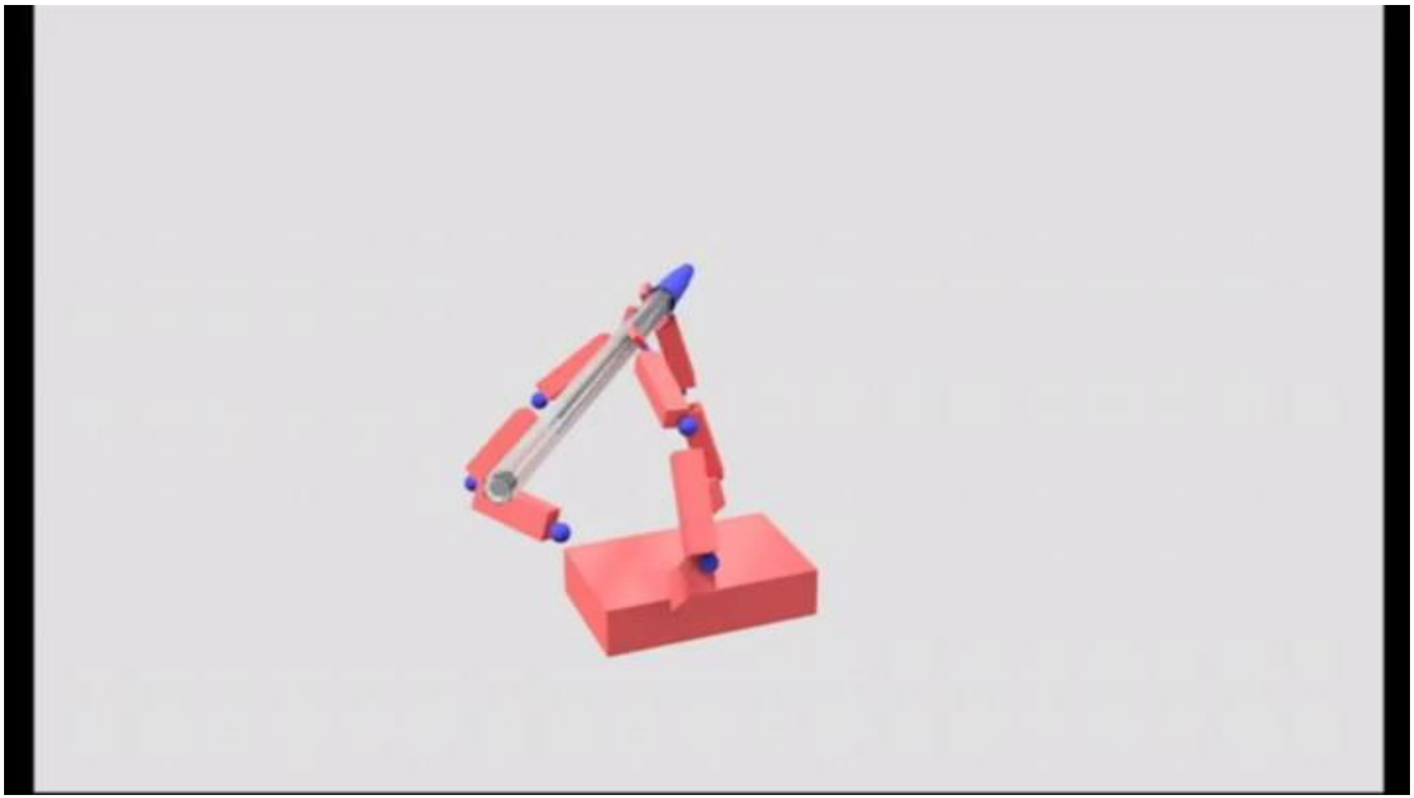



AlphaGo, The Movie


Such as the Darpa car
egress scenario
Complete computation
time ≈ 31s







# End-to-end control?

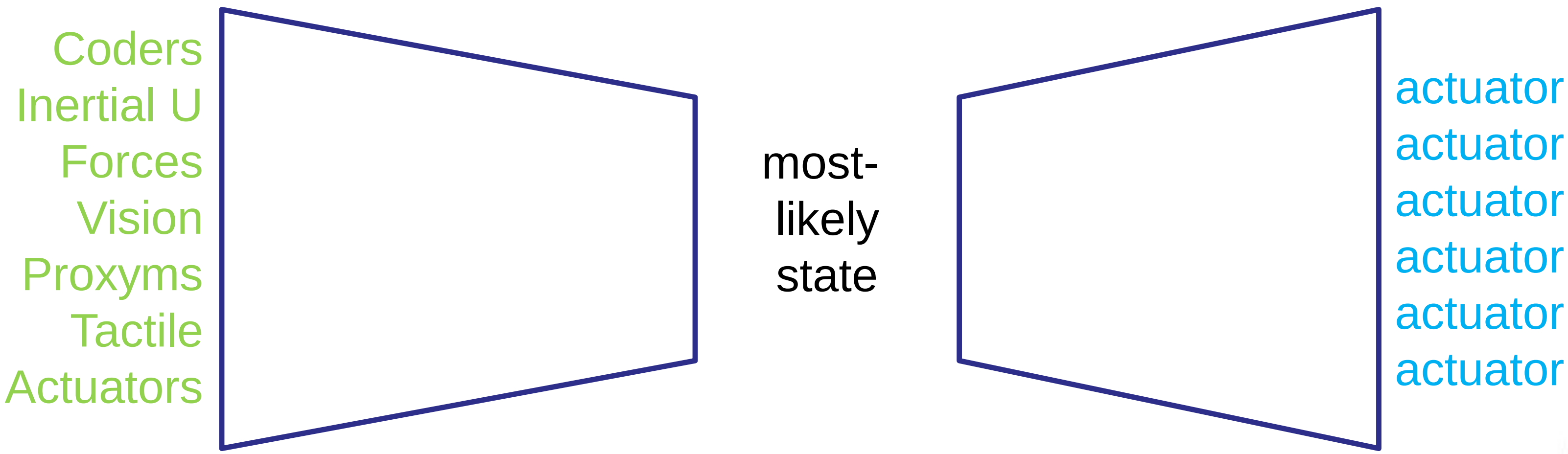
min control sequence — Preview cost

s.t. — Geometry constraint

Dynamic constraint

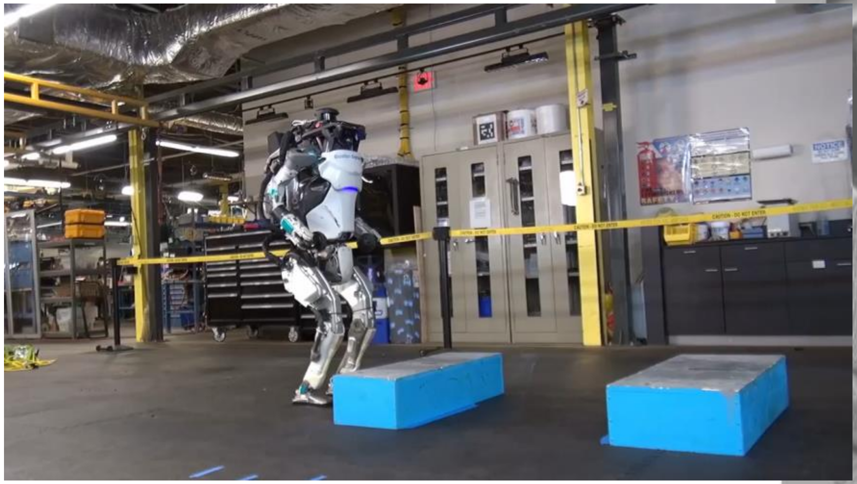
knowing — Past measurements

# End-to-end control?



# Simulation VS embodiment


Hees et al, Deep Mind



Atlas, Boston Dynamics

## Simulation

- End-to-end learning
- Few priors

## Robot

- State+sensor feedback
- Expert-tuned controller

# Sense – plan – act

Planning

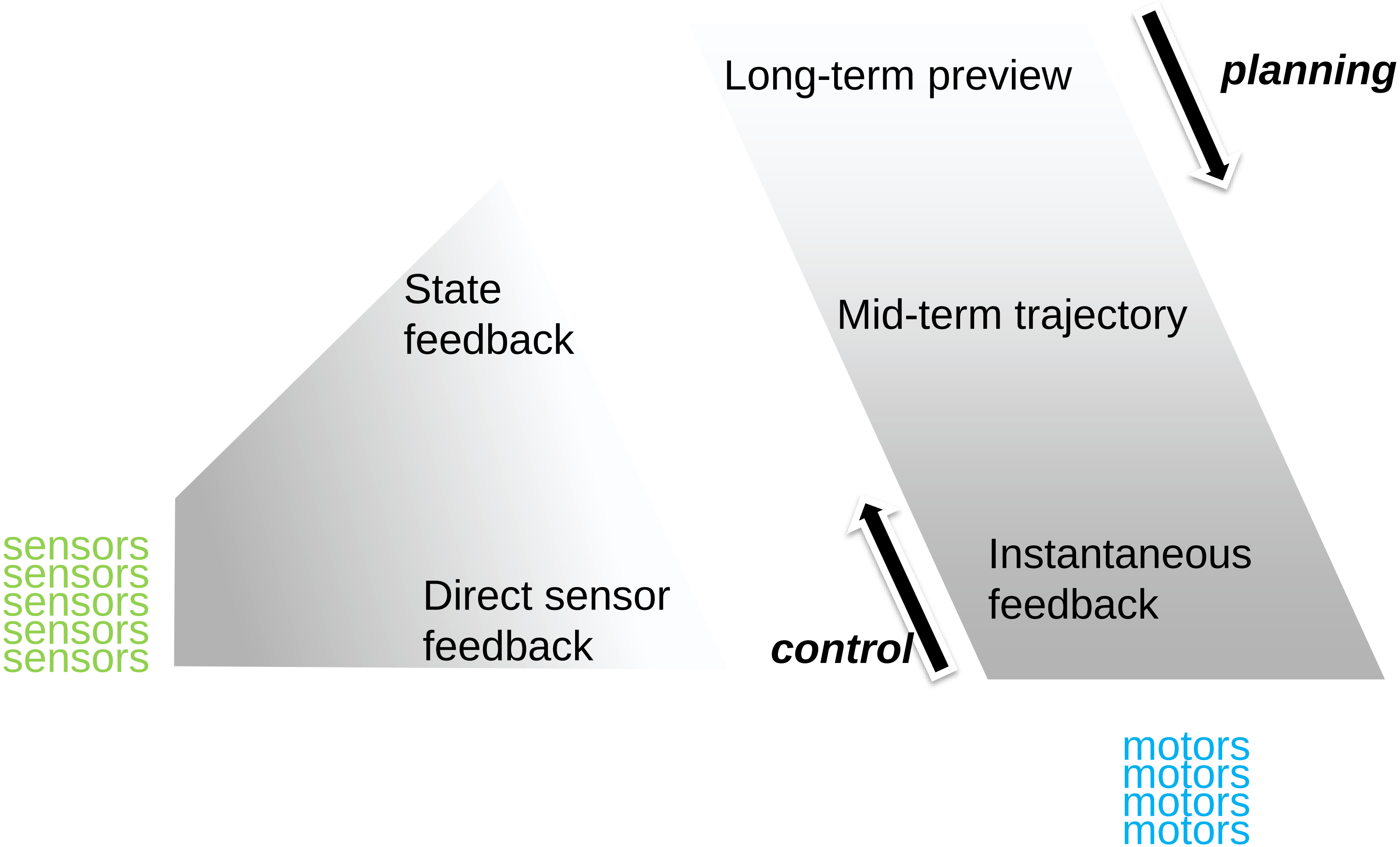
sensors
sensors
sensors
sensors
sensors

Sensing

Control

motors
motors
motors
motors

# Sense – plan – act



# Main messages

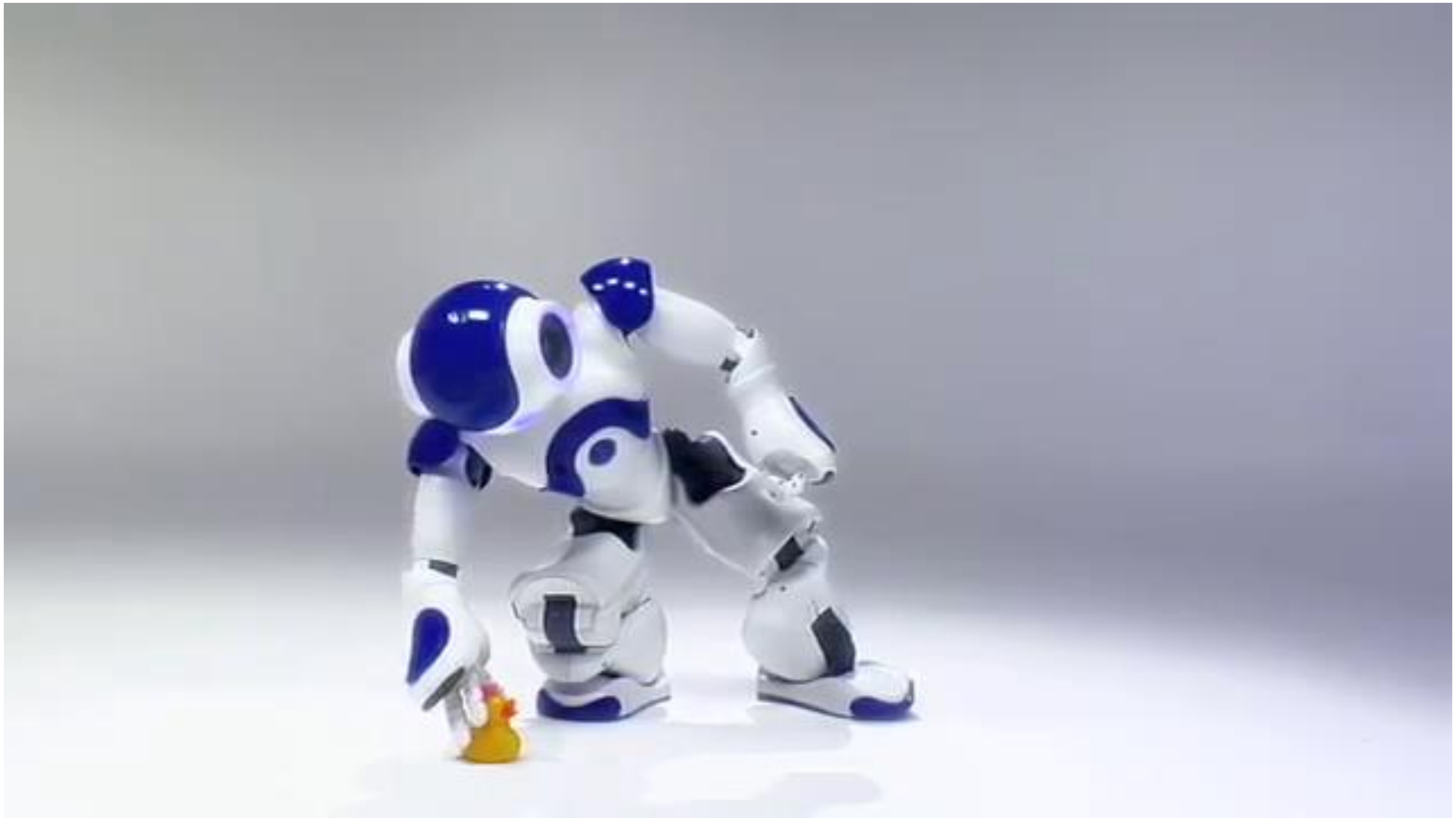
- How to solve robotics through IA?
  - Many subproblems of robotics are well understood
  - IA approaches (yet) hardly reproduces state-of-the-art results
  - We need to understand how to integrate these knowledge in IA methods
- No data directly available in robotics
  - Exploration/planning is difficult
  - Poor generalization using naïve kernel
- Transferring knowledge
  - From planning to control … from simulation to robot

# Outline

- Geometry
  - Model, inverse geometry/kinematics, motion planning
- Dynamics
  - Model and algorithms, contact dynamics, simulation
- Optimal control
  - Predictive control VS policy learning
- Agile robots
  - Practical case study

# Geometry

*Model, inverse geometry/kinematics, motion planning*



# How to construct this motion?

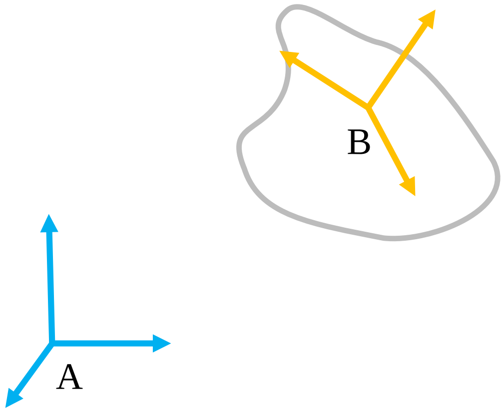
# Rigid-body geometry

- Homogeneous notation

$$^{A}M_{B} = \begin{pmatrix} ^{A}R_{B} & ^{A}\overrightarrow{AB} \\ 0 & 1 \end{pmatrix}$$

- Placement – displacement

$$^{A}x = {}^{A}M_{B}\, {}^{B}x$$




$^{A}M_{B}$ represents the placement.

Other representations possible

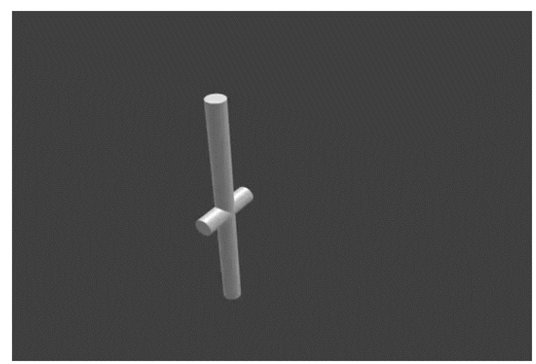
$$^{A}M_{B} \sim \begin{pmatrix} ^{A}\overrightarrow{AB} \\ ^{A}q_{B} \end{pmatrix}$$

# Joint model

- Definition: *a parametric function representing the placement of the output wrt the input.*

- Revolute joint

$$ {}^{I}M_S = \begin{pmatrix} 1 & & & \\ & \cos & \sin & \\ & -\sin & \cos & \\ & & & 1 \end{pmatrix} $$

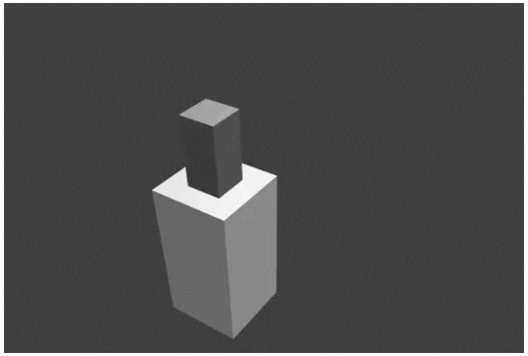


# Joint model

- Definition: *a parametric function representing the placement of the output wrt the input.*

- Prismatic joint

$$
{}^{I}M_S(q) = \begin{pmatrix} 1 & & & q \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{pmatrix}
$$



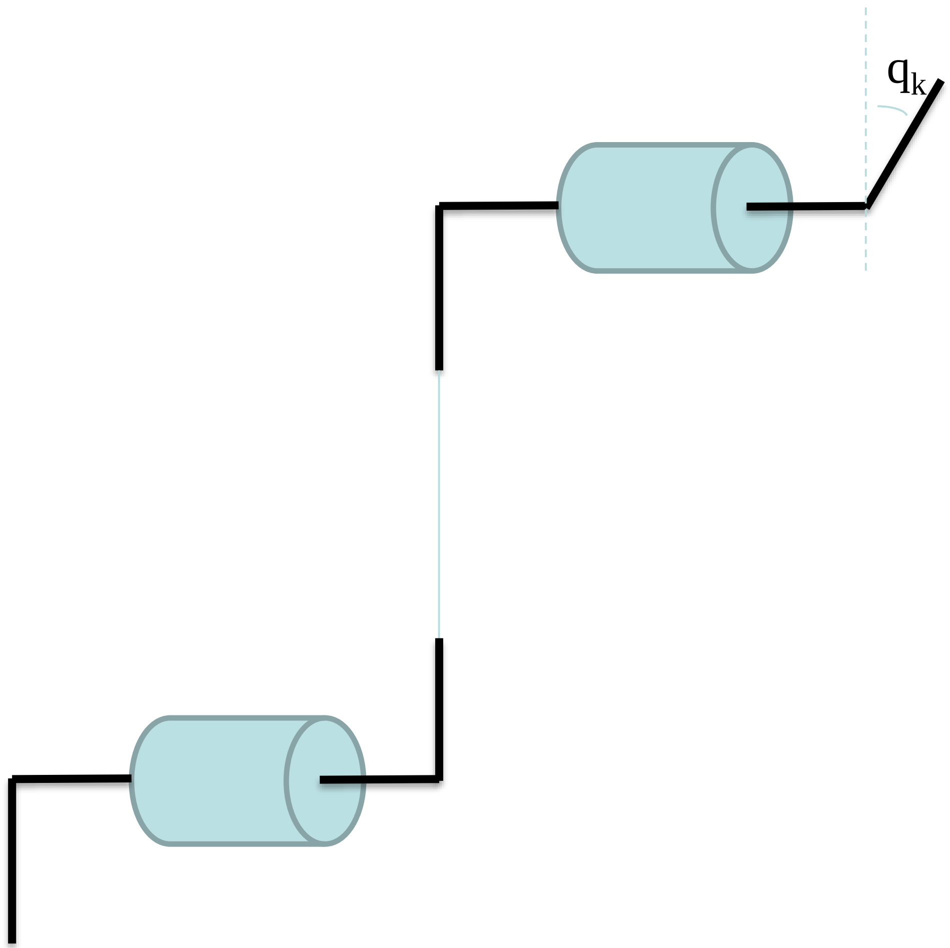
# Joint model

- Definition: *a parametric function representing the placement of the output wrt the input.*

- Ball joint

$$ {}^{I}M_S(q) = \begin{pmatrix} {}^{I}R_S(q) & \\ & 1 \end{pmatrix} $$

# Kinematic model

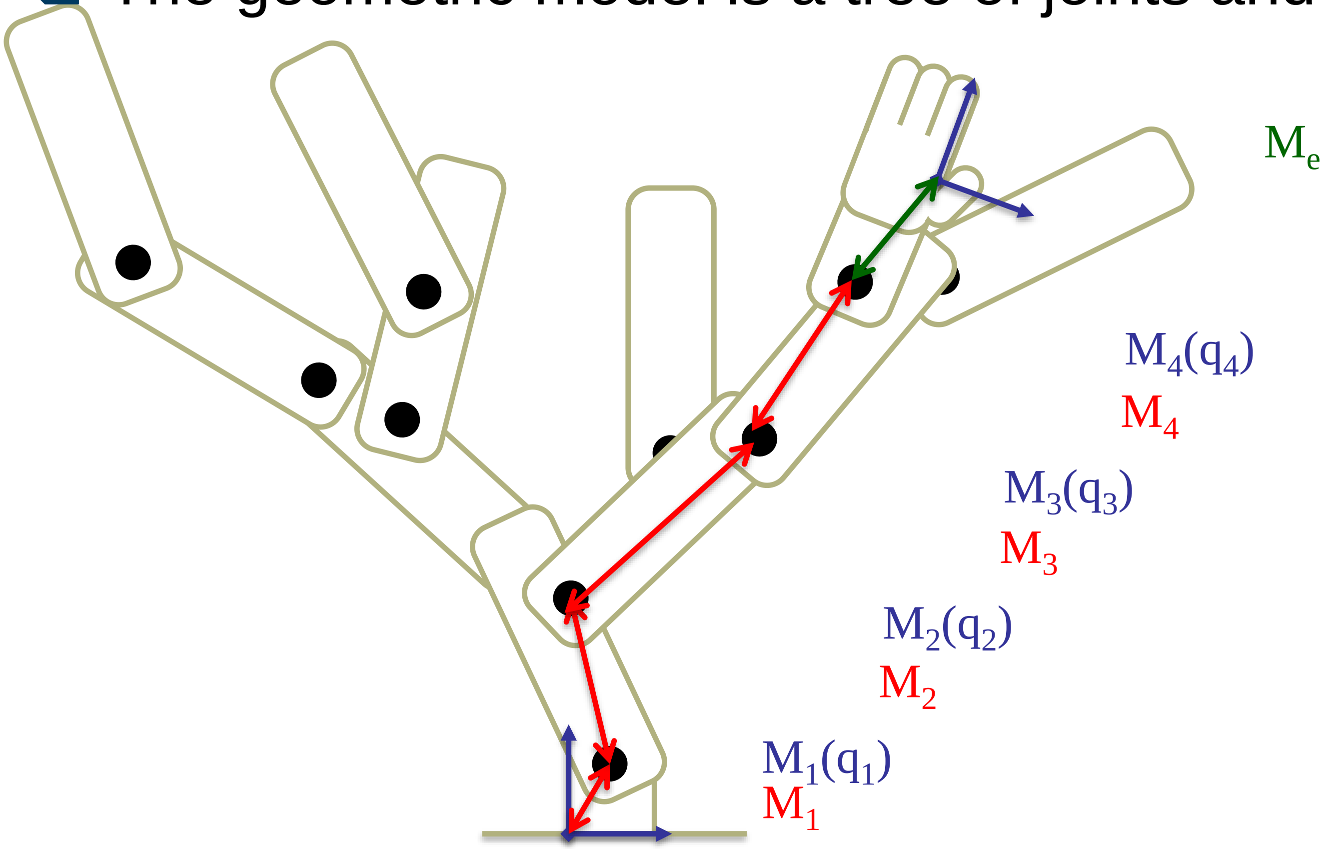



$M_k(q_k)$ Displacement due to the joint

$M_k$ Placement of the joint wrt its parent

# Direct geometry

- The geometric model is a tree of joints and bodies

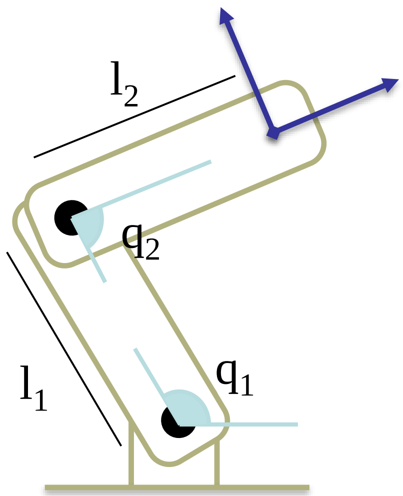



$$M(q) = M_1 \oplus M_1(q_1) \oplus M_2 \oplus \ \ldots \oplus M_4 \oplus M_4(q_4) \oplus M_e$$

# Inverse geometry

- Being given a desired placement $M^*$ …

  what is $q$ such that $h(q) = {}^0M_E = M^*$




$$ {}^0M_E(q) = \begin{bmatrix} l_1 \cos(q_1) + l_2 \cos(q_1+q_2) \\ l_1 \sin(q_1) + l_2 \sin(q_1+q_2) \end{bmatrix} $$

# Inverse geometry

- Solve it using nonlinear programming

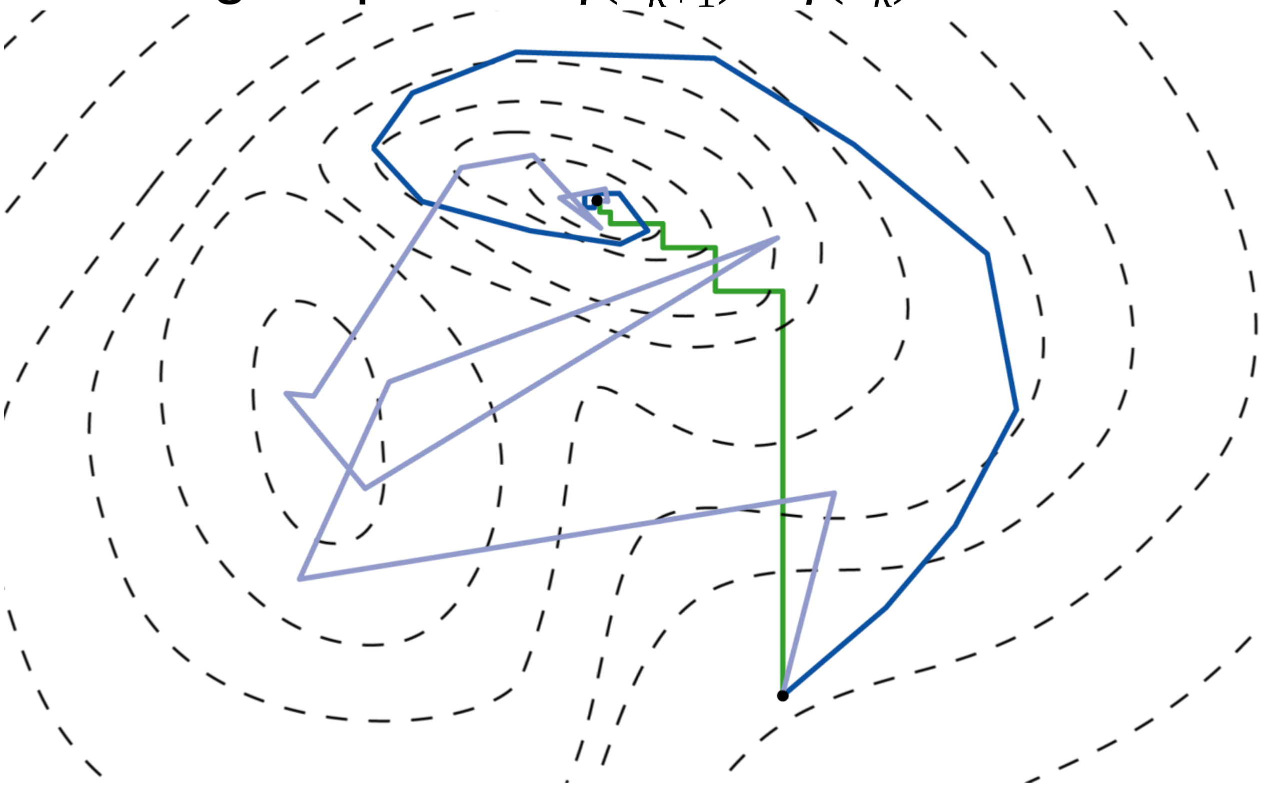
$$\min_{q} \; dist({}^{0}M_E(q)\,,M^*)$$

- Optimization in smooth manifold (Lie groups)

$$\min_{q} \left\| \log( {}^{0}M_E(q)^{-1}\, M^*) \right\|^2$$

# Follow the slope

- Decreasing sequence: $f(x_{k+1}) < f(x_k)$



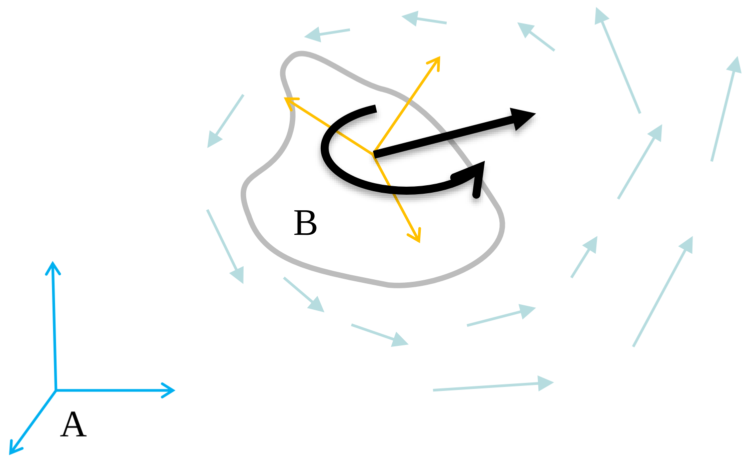
# Rigid-body kinematics

$$\dot{M} = \nu \times M$$

$$\dot{R} = \omega \times R$$

$$= \begin{pmatrix} v \\ \omega \end{pmatrix} \times \begin{pmatrix} R & p \\ 0 & 1 \end{pmatrix}$$

# Rigid-body kinematics



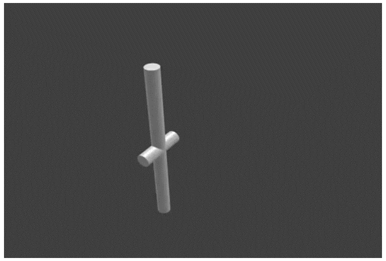

- Velocities are true vectors

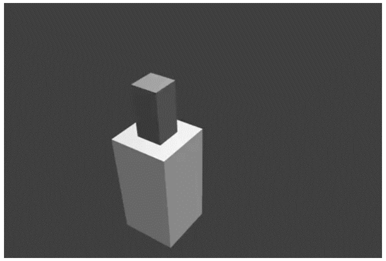
$$^{A}\nu_{AB} = {}^{A}X_{B}\,{}^{B}\nu_{AB}$$

$$^{A}\nu_{AB} = {}^{A}\nu_{AC} + {}^{A}\nu_{CB}$$

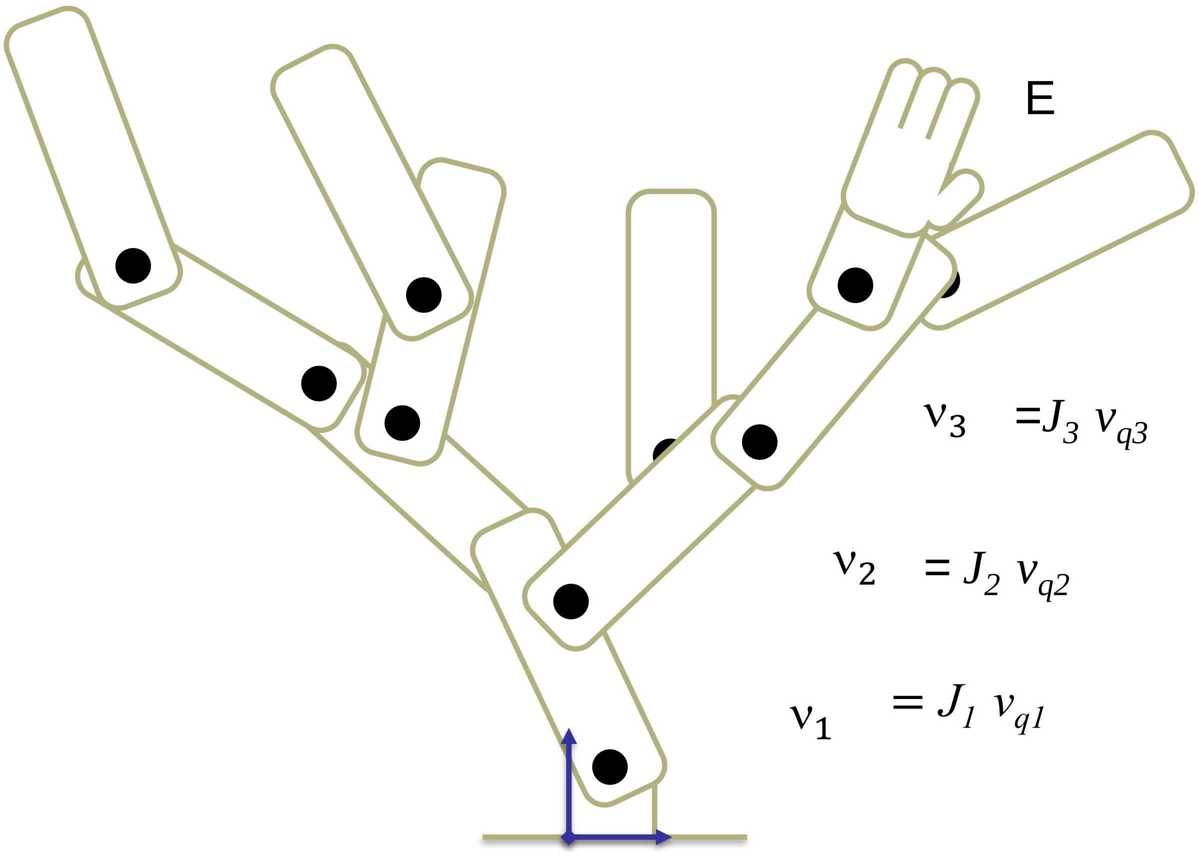
# Joint kinematics

$$\nu = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ v_q \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} v_q \qquad \nu = \begin{pmatrix} 0 \\ 0 \\ v_q \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} v_q \qquad \nu = \begin{pmatrix} 0 \\ 0 \\ 0 \\ v_1 \\ v_2 \\ v_q \end{pmatrix} = \begin{pmatrix} 0&0&0 \\ 0&0&0 \\ 0&0&0 \\ 1&0&0 \\ 0&1&0 \\ 0&0&1 \end{pmatrix} v_q$$





# Robot kinematics




$$\nu_3 = J_3\, v_{q3}$$

$$\nu_2 = J_2\, v_{q2}$$

$$\nu_1 = J_1\, v_{q1}$$

$$\nu_E = \nu_1 + \nu_2 + \nu_3 + \cdots = \begin{pmatrix} J_1 & J_2 & J_3 & \dots \end{pmatrix} \begin{pmatrix} v_{q1} \\ v_{q2} \\ v_{q3} \\ \dots \end{pmatrix}$$

# Inverse kinematics

- Given the current configuration q
- Find the closest velocity …

$$\min_{v_q} \| J(q) v_q - \nu^* \|$$

$$v_q = J^+ \nu^*$$

- One step in inverse geometry

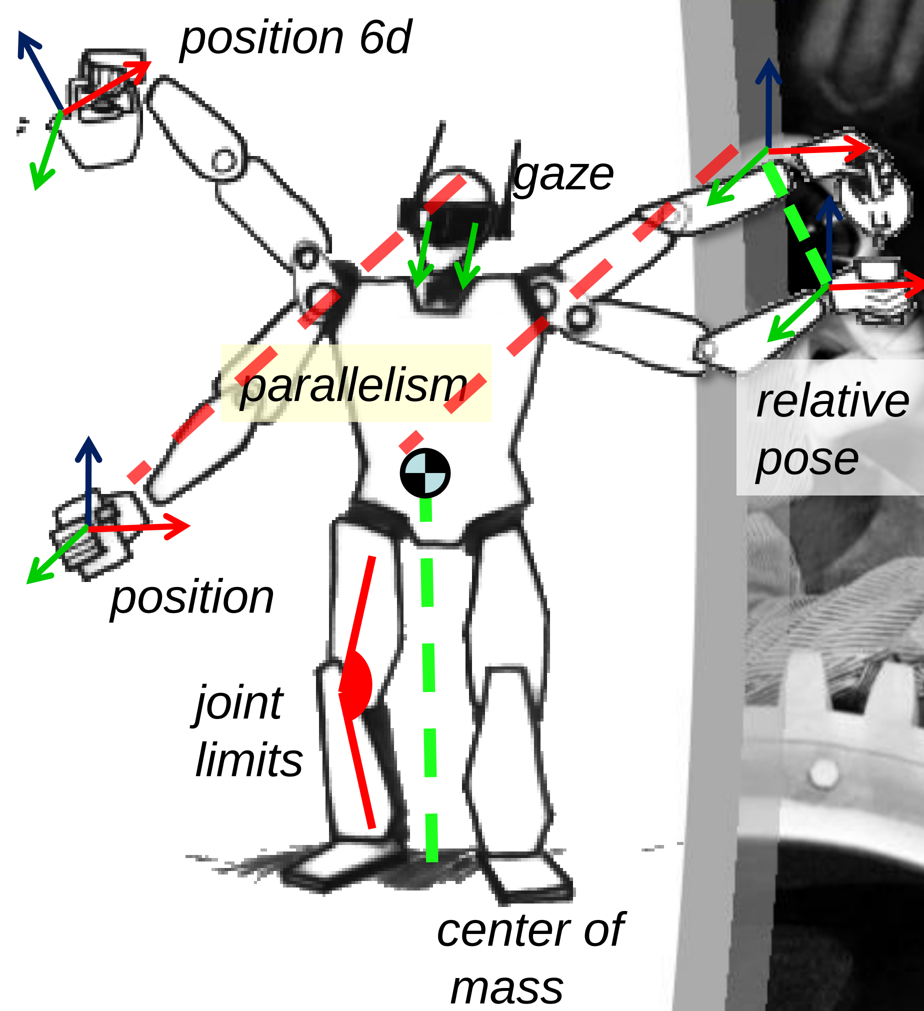
$$\min_q \| r(q) \|^2$$

  - Gradient $= \frac{dr}{dq}^T r^*$ Hessian $= \frac{dr}{dq}^T \frac{dr}{dq} + \varepsilon$
  - Gauss-Newton step $= (\frac{dr}{dq}^T \frac{dr}{dq})^{-1} \frac{dr}{dq}^T r = J^+ r$

# Control in the task space

- Effector position / placement
- Center of mass
- Sensor orientation
- …

- Joint limits
- Field of view
- Balance support
- …



# Redundancy and hierarchy

## First example

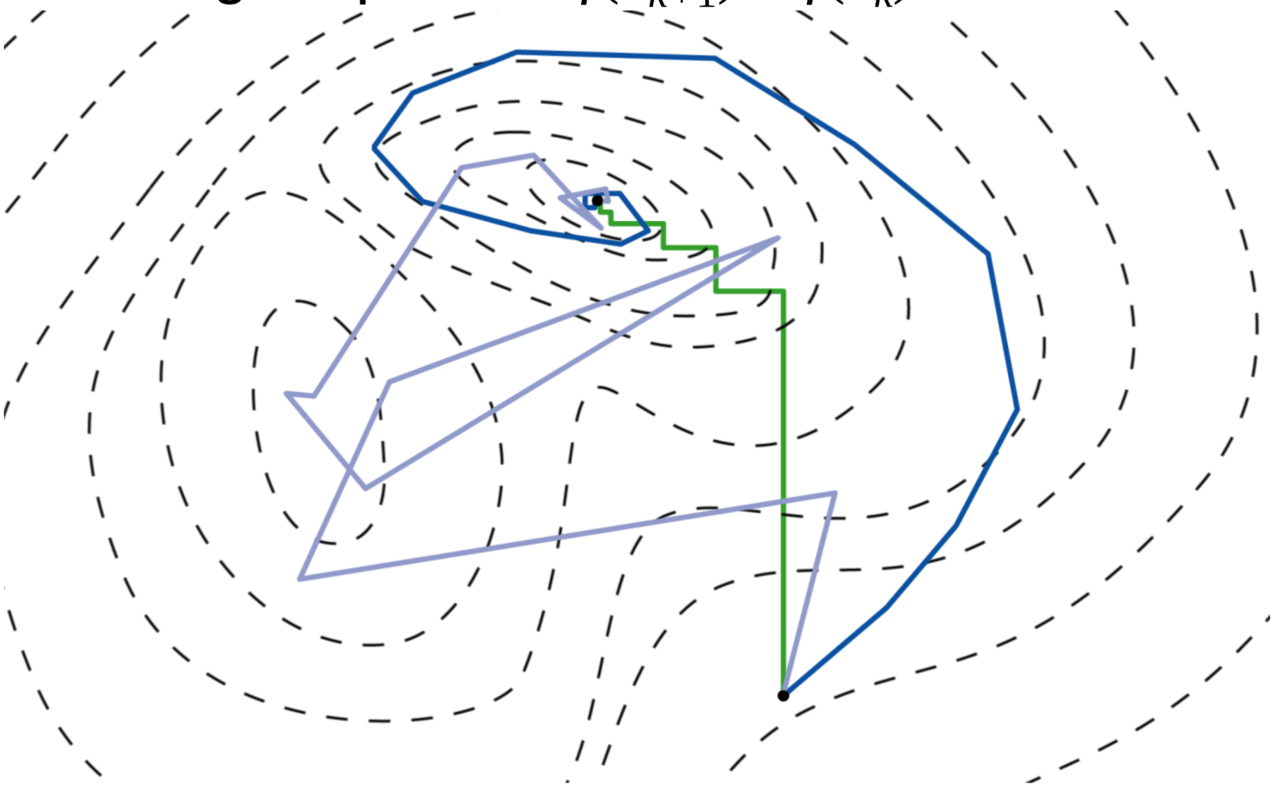
HRP-2 has to walk while avoiding the obstacle and keeping if possible its umbrella vertical with its companion below. This motion is detailed in Part I, Section 5.

Now let us see what happen if an obstacle is on the way.

joint-limits $\prec$ walk $\prec$ bound angle $\prec$ umbrella $\prec$ fov $\prec$ angle

# Follow the slope

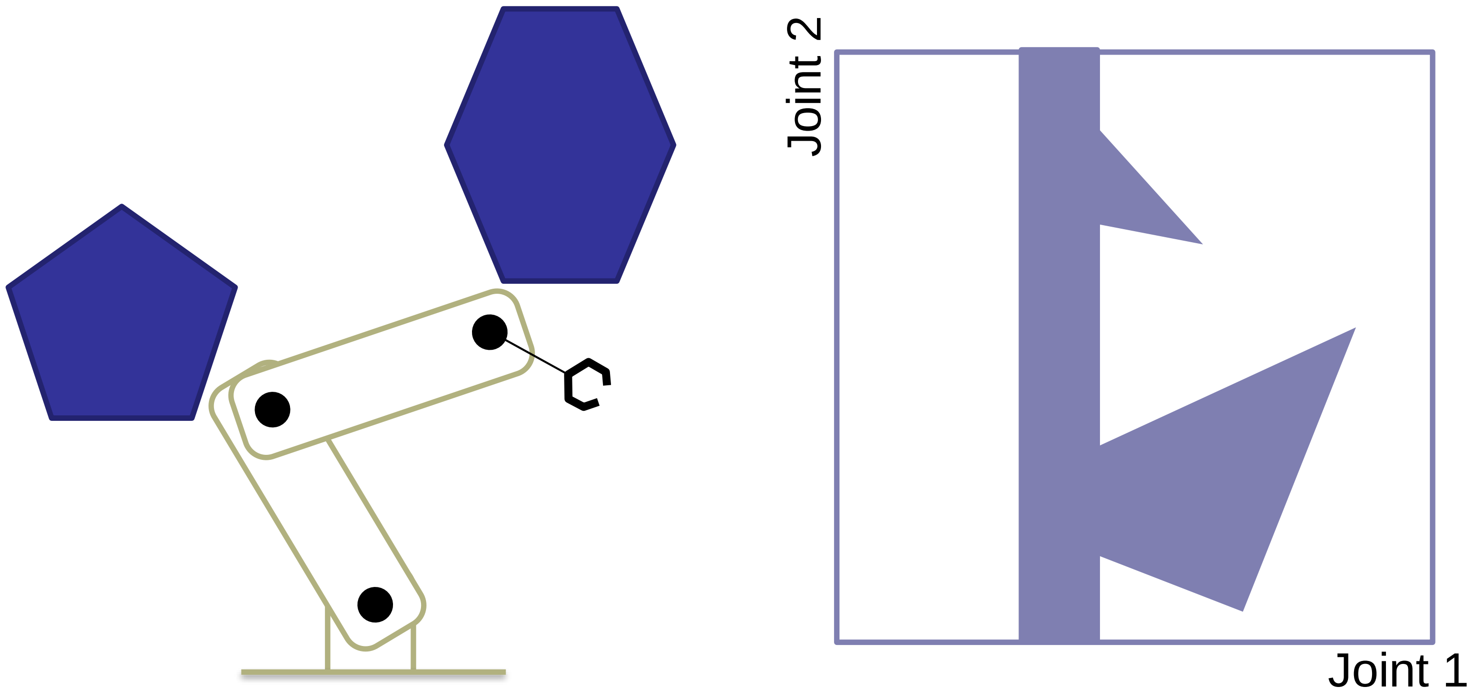
- Decreasing sequence: $f(x_{k+1}) < f(x_k)$



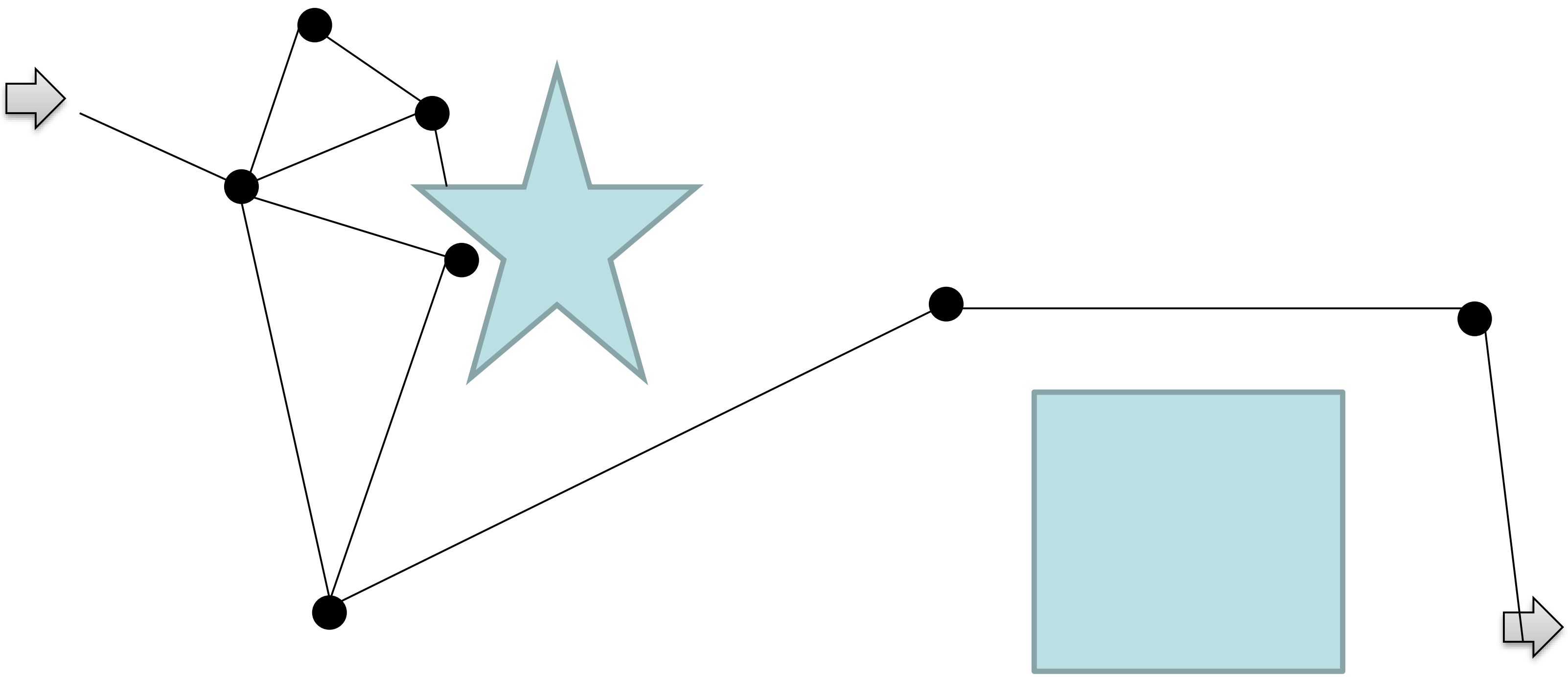
# Explore space

- Monte Carlo Tree Search
  - See R. Munos tomorrow
- Major stakes
  - Discretization
  - Boundedness on derivatives (Lipschitz continuity)

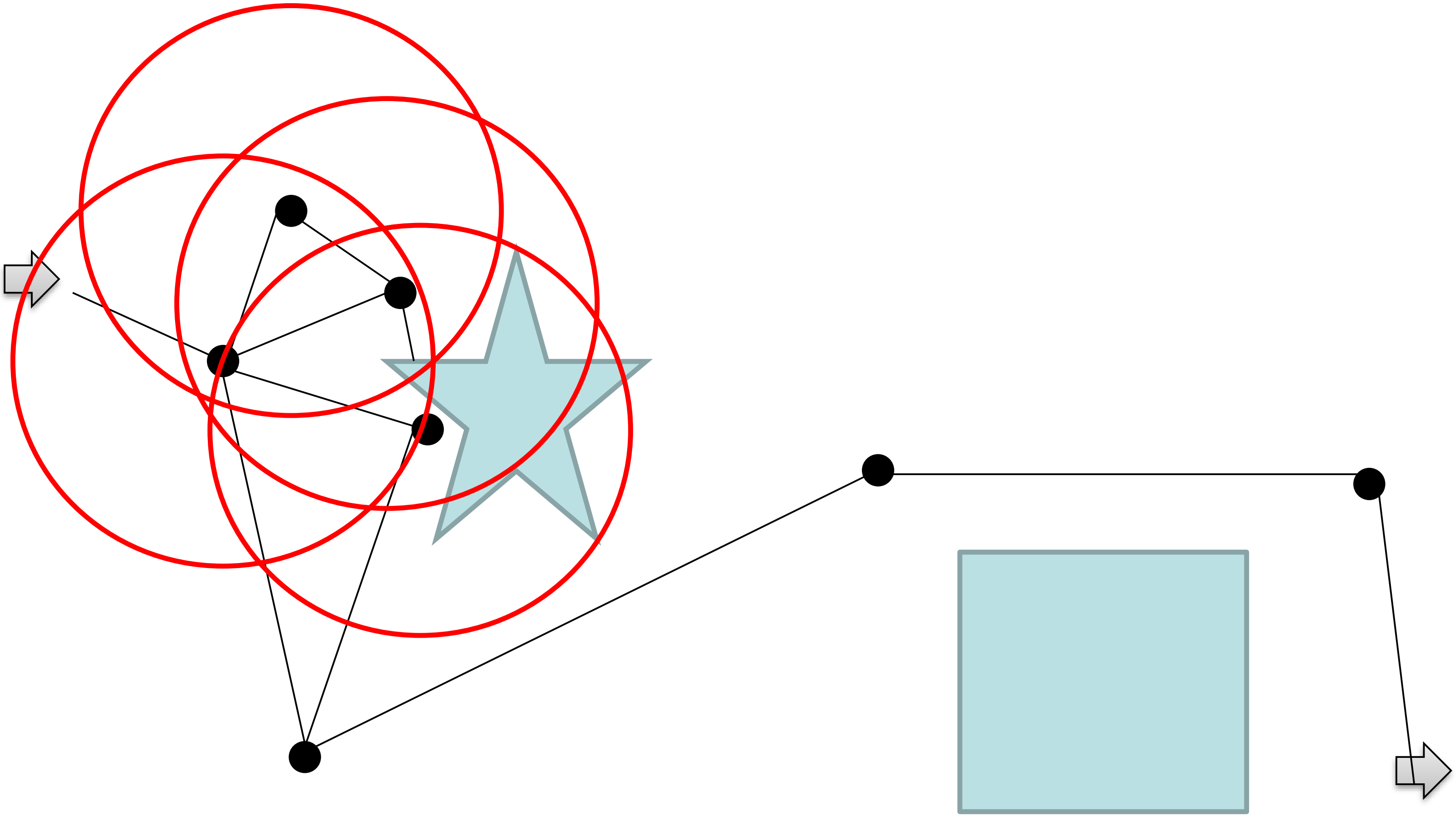
# Collisions!



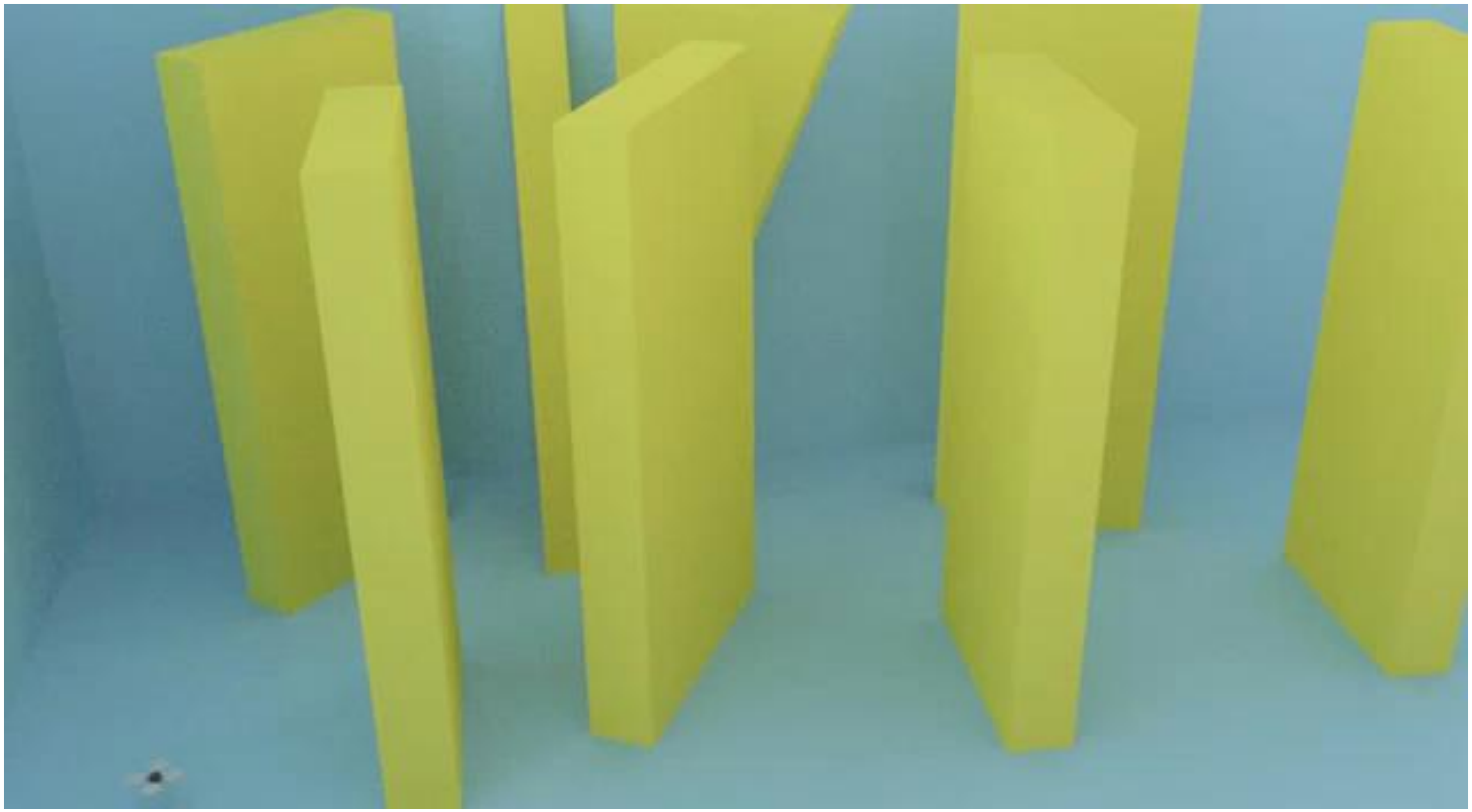
# Random Roadmap

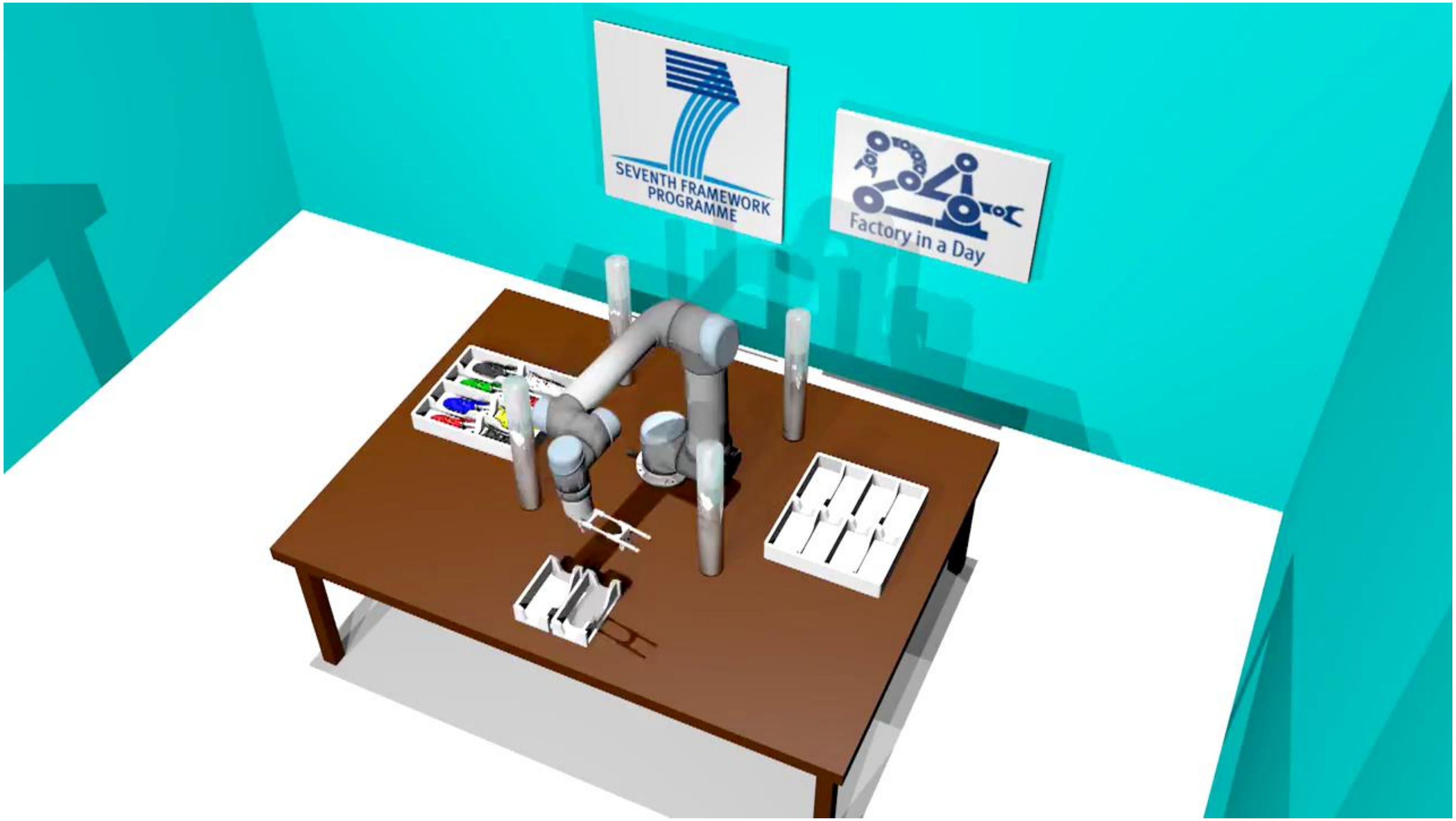


# Random Roadmap



# Random roadmap




SEVENTH FRAMEWORK PROGRAMME
Factory in a Day

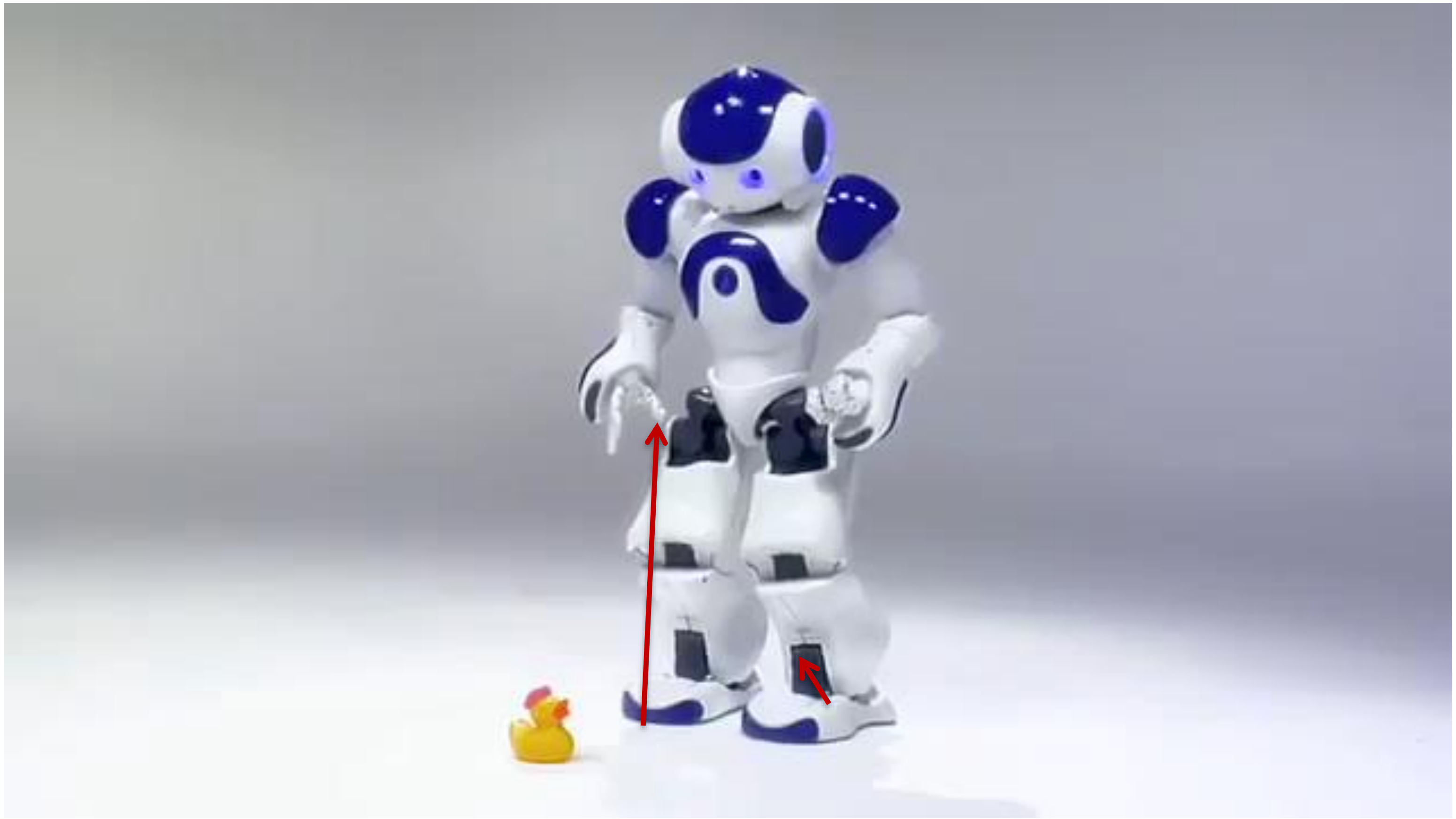
# Random roadmap

# Kinematics of advanced robots


Adept, commercial

# Dynamics

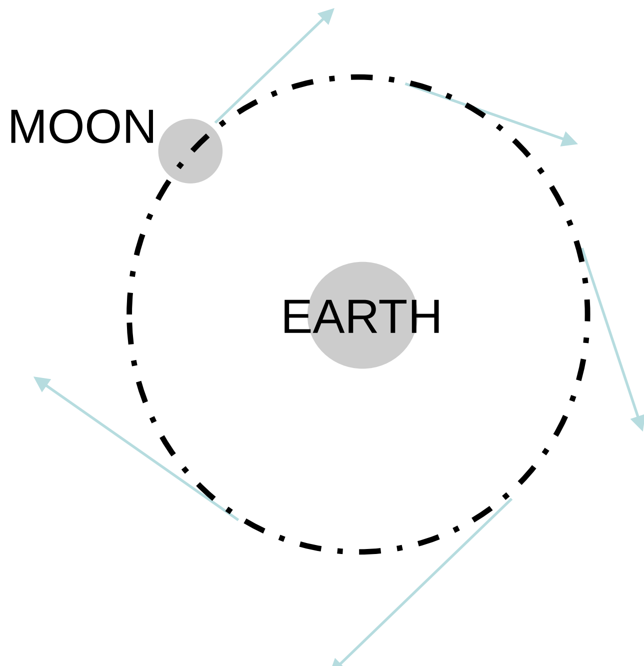
*Model and algorithms, contact dynamics, simulation*



# How to construct this motion?

# Rigid-body dynamics

- Body placement: $M = \begin{pmatrix} R & p \\ 0 & 1 \end{pmatrix}$
- Body velocity: $\nu = \begin{pmatrix} v \\ \omega \end{pmatrix}$
- Body acceleration: $\alpha = \dot{\nu}$




$$\nu = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$\alpha = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

# Rigid-body dynamics

- Body placement: $M = \begin{pmatrix} R & p \\ 0 & 1 \end{pmatrix}$
- Body velocity: $\nu = \begin{pmatrix} v \\ \omega \end{pmatrix}$
- Body acceleration: $\alpha = \dot{\nu}$
- Body force: $\varphi = \begin{pmatrix} f \\ \tau \end{pmatrix}$
- Body inertia: $\Upsilon = \begin{pmatrix} m & 0 \\ 0 & I \end{pmatrix}$

| | | |
|---|---|---|
| Y: Motion | $\rightarrow$ | Force |
| Velocity $\nu$ | $\rightarrow$ | Momentum $\eta$ |
| Acceleration $\alpha$ | $\rightarrow$ | Force $\varphi$ |

# Robot dynamic model

- Kinematic energy

$$E_c = \sum \frac{1}{2} \nu_i^T Y_i \nu_i \qquad (\nu = J\, v_q)$$

$$= \frac{1}{2} v_q^T \underbrace{\left( \sum J_i^T Y_i J_i \right)}_{M(q)} v_q$$

# Robot dynamic model

- Kinematic energy

$$E_c = \frac{1}{2} v_q^T M(q) v_q$$

- Lagrange principle

$$M(q)\dot{v}_q + {v_q}^T R(q) v_q + g(q) = \tau_q$$

mass

acceleration

bias

joint forces

# Dynamics with contact

$$M(q)\dot{v_q} + {v_q}^T R(q) v_q + g(q) = \tau_q$$

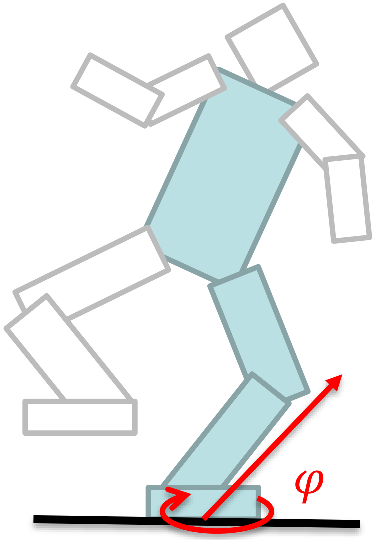
- Joint torques?

$$\tau_q = \tau_{motor} + \tau_{ext}$$

- Motor torque: actuation dynamics
  - Example: the first 6 values are null
- External forces:

$$\tau_{ext} = J^T \varphi$$

$$\nu = J\, v_q \qquad \mathrm{P} = {\tau_{ext}}^T v_q$$



# Simulation

$$M(q)\dot{v_q} + {v_q}^T R(q) v_q + g(q) = \tau_m + J^T \varphi$$

- Rewrite with steps and normal forces

$$v_+ = v_0 + M^{-1} J^T f$$

- Constraints imposed by the contact model
  - Positive forces: $f \geq 0$
  - Positive motion: $J\, v_+ \geq 0$
  - One or the other: $f^T J\, v_+ = 0$
- Linear complementarity problem

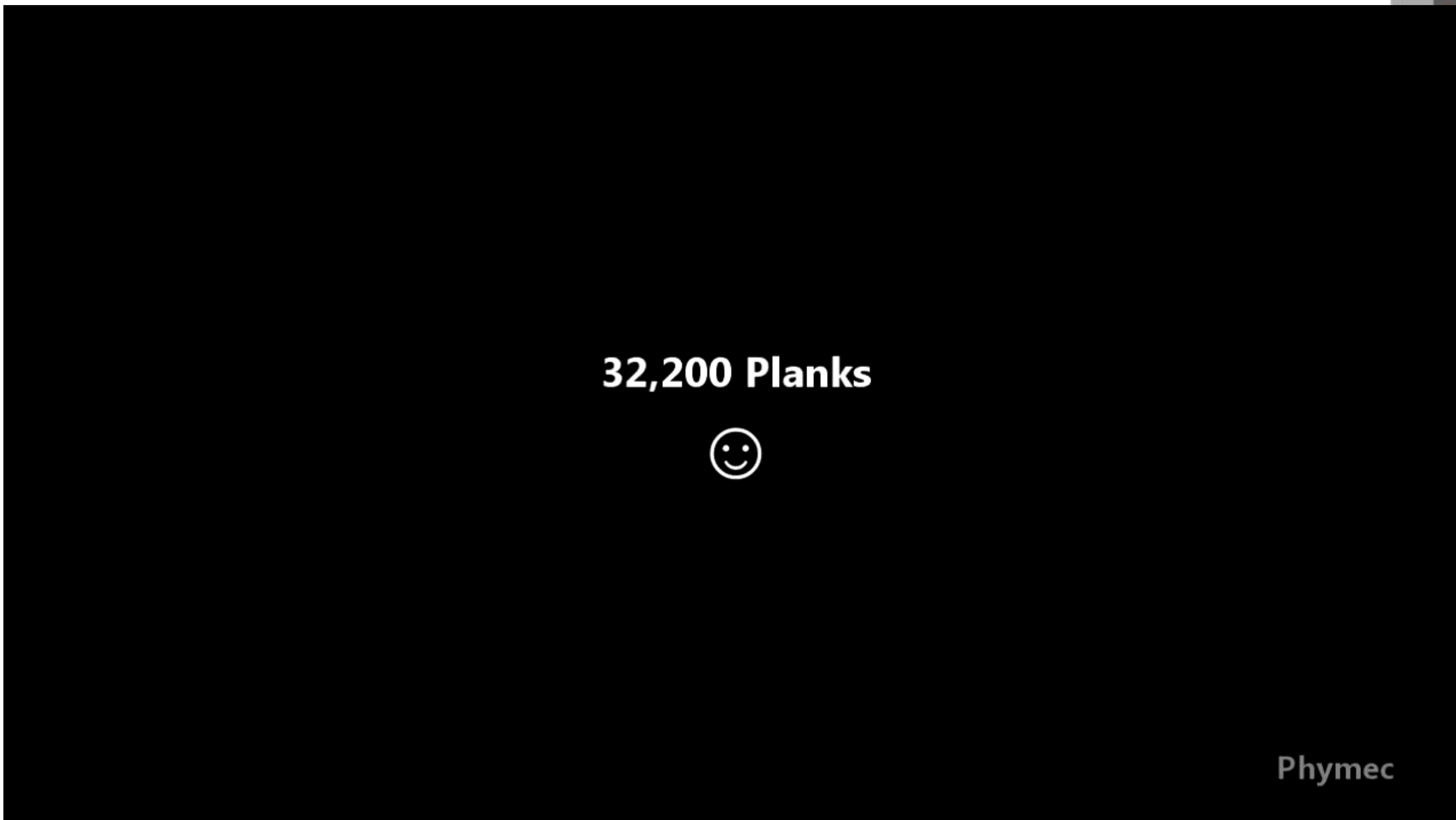
# Force and Lagrange multiplier

- Search $v_+$ and $f$

$$v_+ = v_0 + M^{-1} J^T f$$
$$f \geq 0$$
$$\mathrm{J}\, v_+ \geq 0$$
$$f^T \mathrm{J}\, v_+ = 0$$

- KKT conditions of the Gauss “least-action” principle

$$\min_{v_+} \| v_+ - v_0 \|_{M^{-1}}$$
$$\text{s.t. } \mathrm{J}\, v_+ \geq 0$$



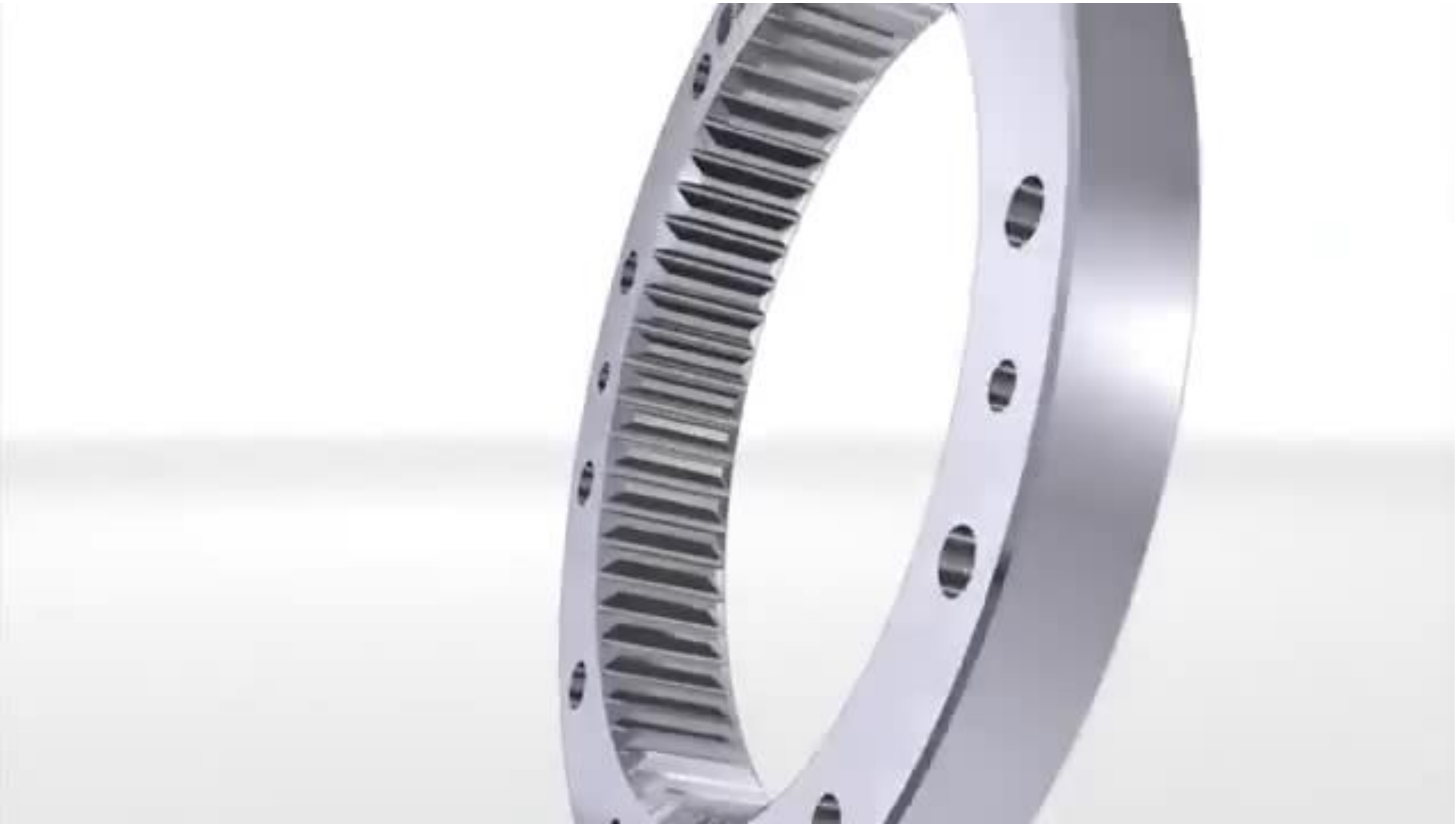

From YouTube

# Task-space inverse dynamics


Wensing&Orin, Ohio

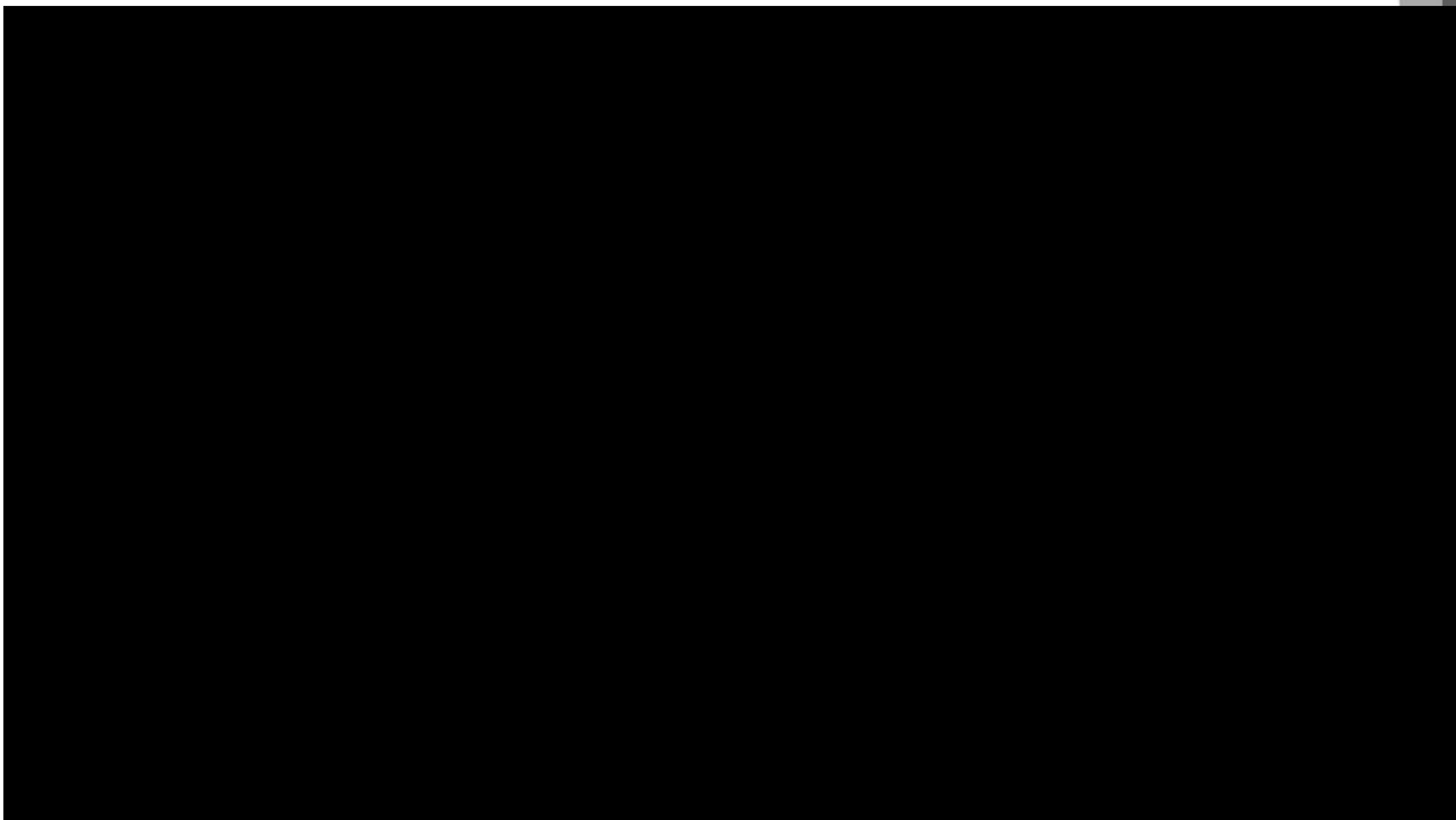

Spot with arm, Boston Dynamics

# Robot actuation



HarmonicDrive, commercial

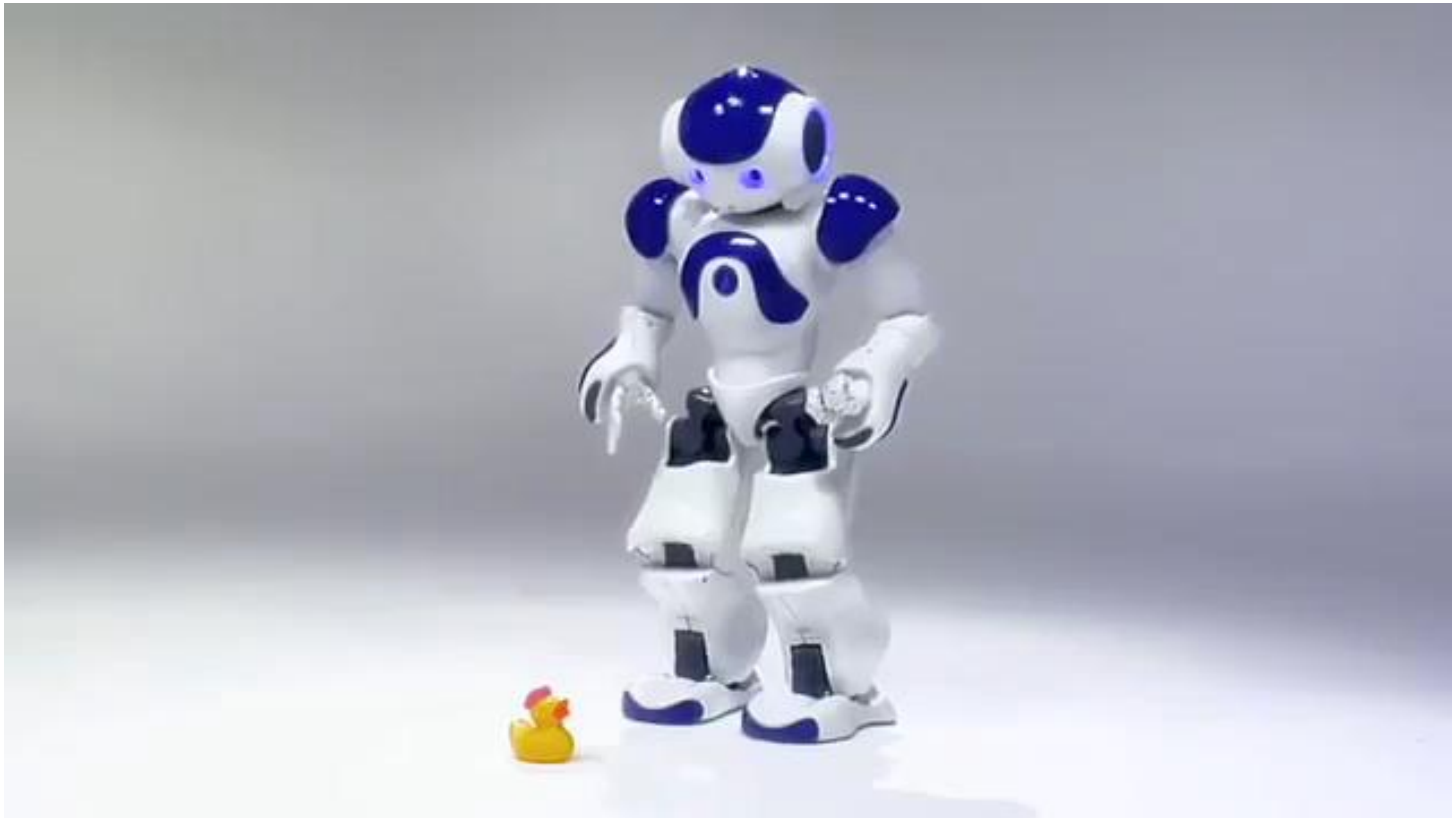
# Robot actuation



Cassie, Hurst (agility robotics), Oregon

# Optimal control

| | | |
|---|---|---|
| *Pontryagine* | | *Belman* |
| *Predictive control* | *VS* | *Policy learning* |
| *Trajectory optim* | | *Policy optimization* |



# How to construct this motion?

# Problem definition

$$\min_{X,U} l_T\big(x(T)\big) + \int_0^T l\big(x(t), u(t)\big)dt$$

$$\text{s.t.} \quad \dot{x}(t) = f(x(t),u(t))$$

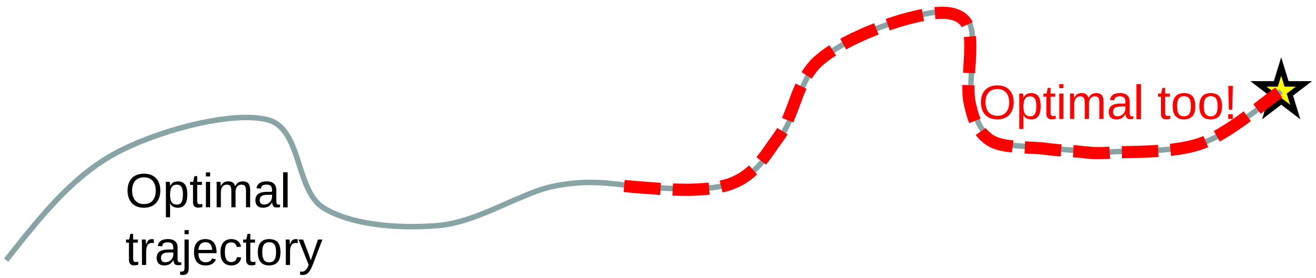
- $X$ and $U$ are functions of t:

  $X: t \in \Re \rightarrow x(t) \in \Re^{\text{nx}}$

  $U: t \in \Re \rightarrow u(t) \in \Re^{\text{nu}}$

- The terminal time $T$ is fixed

# Belman principle and equation




- Given an optimal trajectory …

… subtrajectories are optimal too

# Belman principle and equation

- Value function $V_t(x)$
  - Best score possible from $x,t$

- Belman principle, rewritten

$$V(x,t) = \min_{u_t..u_{t+\Delta t}} \left\{ \int_t^{t+\Delta t} l\big(x(s),u(s)\big)ds + V\big(x(t+\Delta t)\big) \right\}$$

- Passing to $\Delta t \rightarrow 0$

$$V(x,t) = \min_u l(x,u)\,\Delta t$$

$$+ V(x)\Delta t + \frac{dV}{dt}\Delta t + \frac{dV}{dx} f(x,u)$$

- Reordering

$$-\frac{dV}{dt} = \min_u l(x,u) + \frac{dV}{dx} f(x,u)$$

# Optimality principles

- Hamiltonien $H(x,u,p) = l(x,u) + \langle p | f(x,u) \rangle$

- Hamilton Jacobi Belman equation

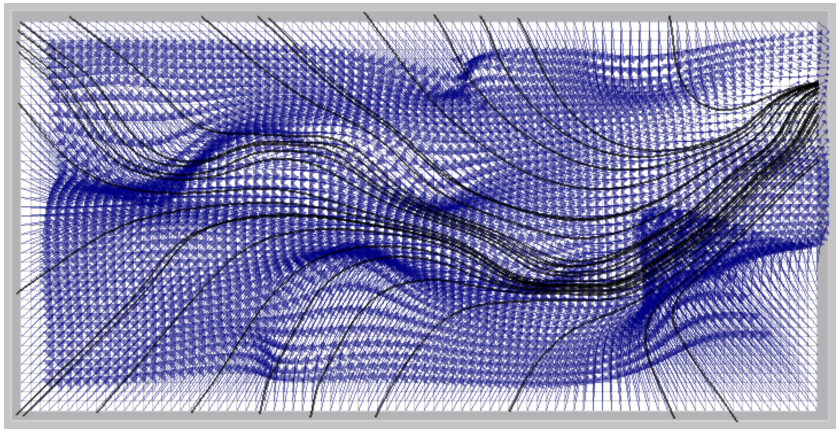
$$u^*(x) = \operatorname*{argmin}_u H\left(x, u, \frac{\partial V}{\partial x}(x)\right)$$

- Pontryagin Maximum principle

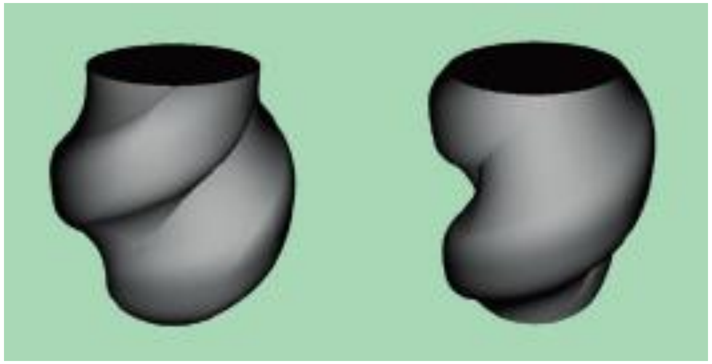
$$u^*(t) = \operatorname*{argmin}_u H\left(x(t), u, p(t)\right)$$

# Optimality principles

- Curse of dimensionality



- Solved in particular cases
  - Nonholonomic car-like robots
  - Reachability sets



# Trajectory (direct) optimization

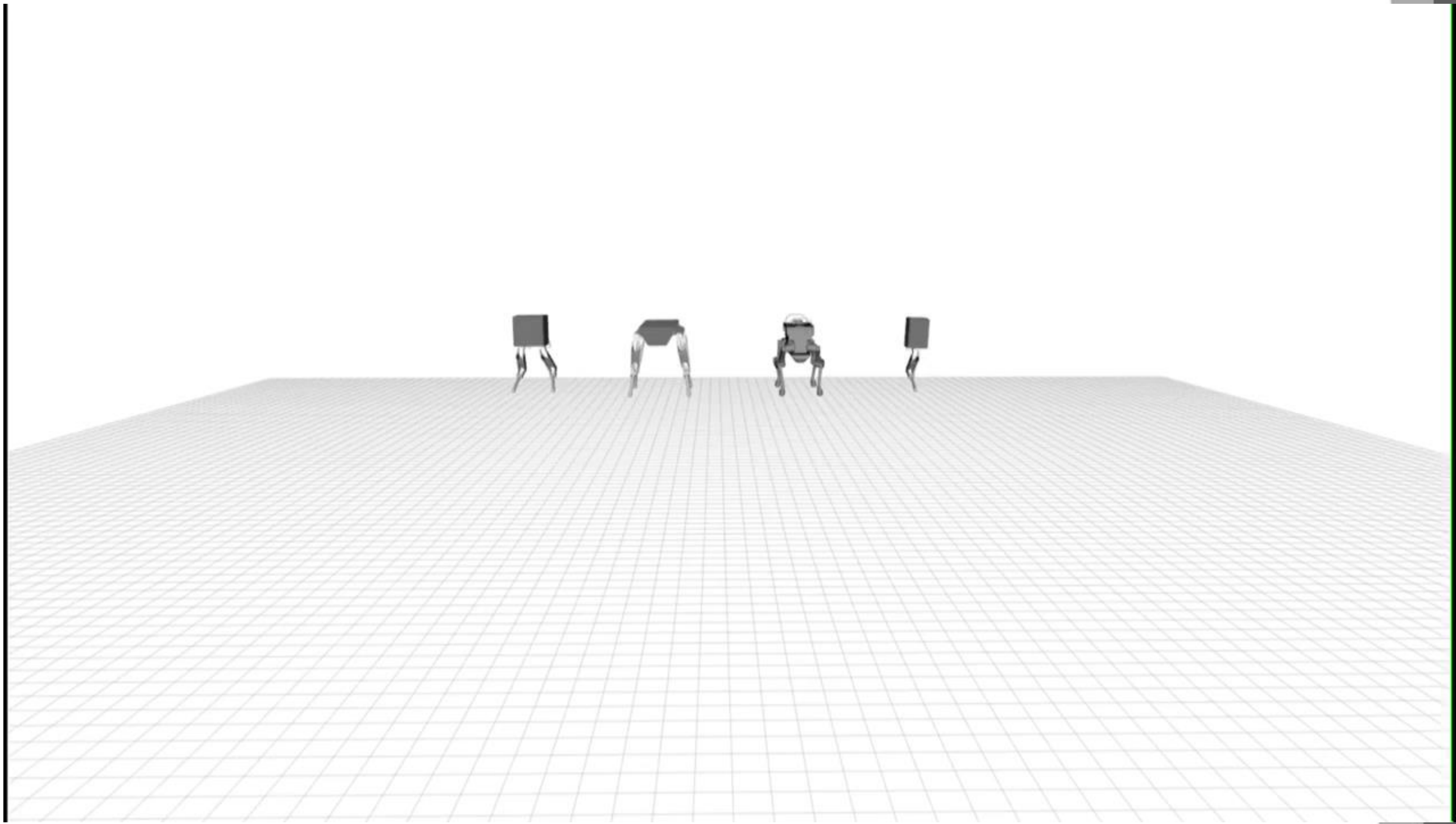
$$\min_{X,U} \; l_T(x_T) + \sum_{t=0}^{T-1} l(x_t, u_t)$$

$$\text{s.t.} \quad x_{t+1} = f(x_t, u_t)$$

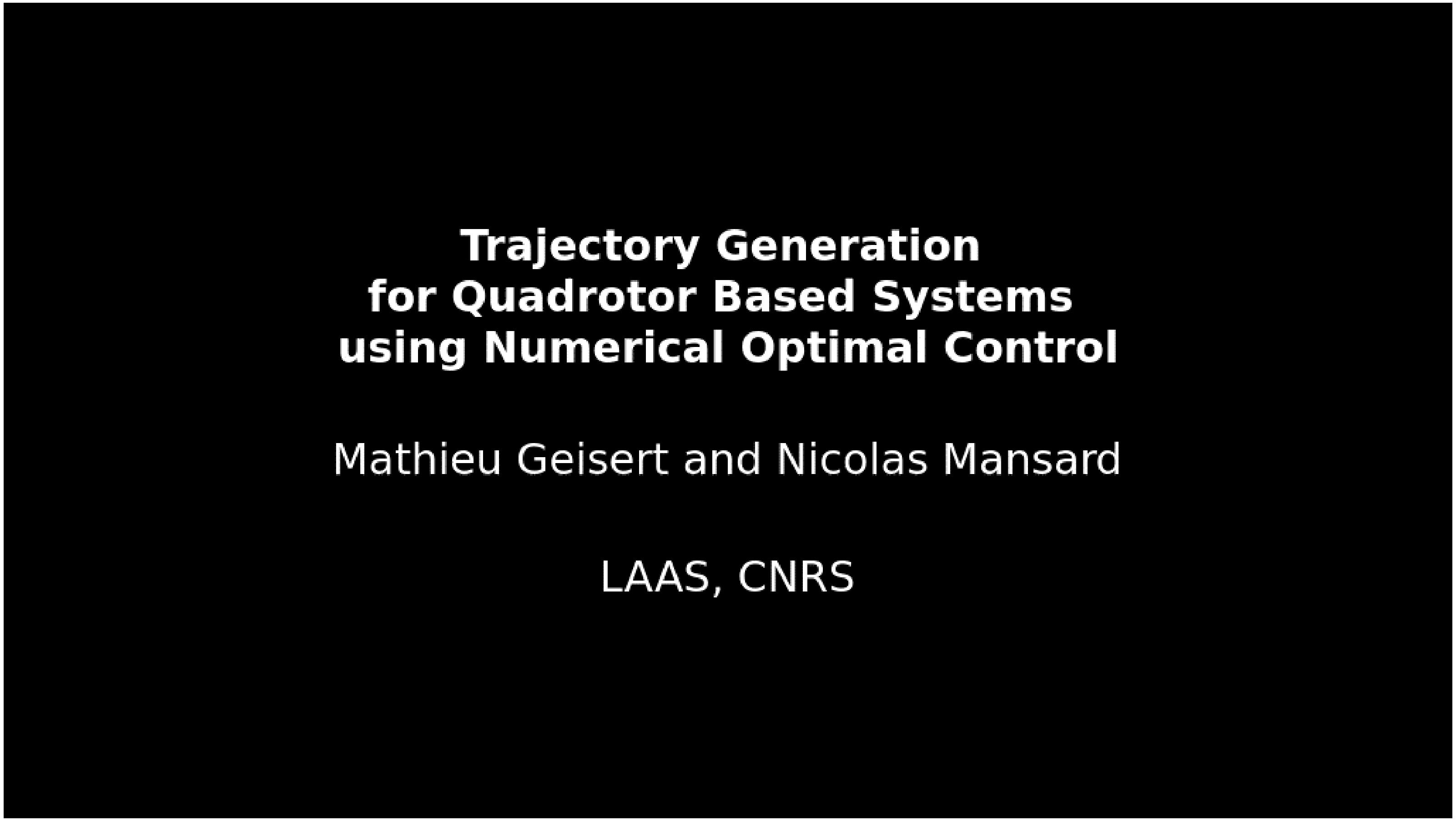
- $X$ and $U$ are vectors of dimension $T{*}nx$ and $T{*}nu$ resp.

$$X = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \qquad U = \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix}$$

- The information in $X$ and $U$ is somehow redundant



Winkler, Buchli et al, ETHZ


Trajectory Generation
for Quadrotor Based Systems
using Numerical Optimal Control
Mathieu Geisert and Nicolas Mansard
LAAS, CNRS

# Shooting and collocation

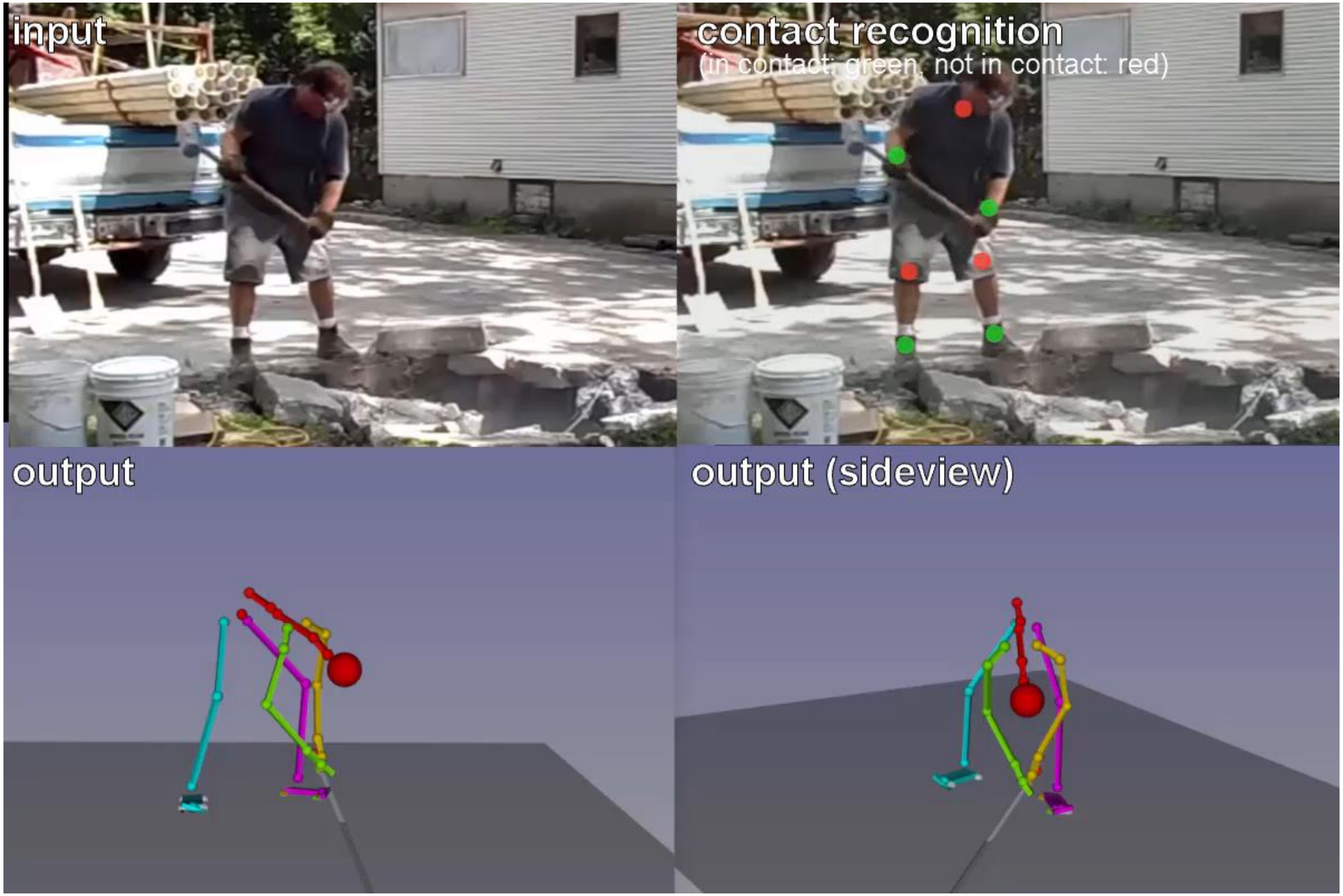
$$\min_{\cancel{X},U} \; l_T(x_T) + \sum_{t=0}^{T-1} l(x_t, u_t)$$

~~s.t. $x_{t+1} = f(x_t, u_t)$~~

$$x_t := f_t(x_0, u_0, \ldots, u_t)$$

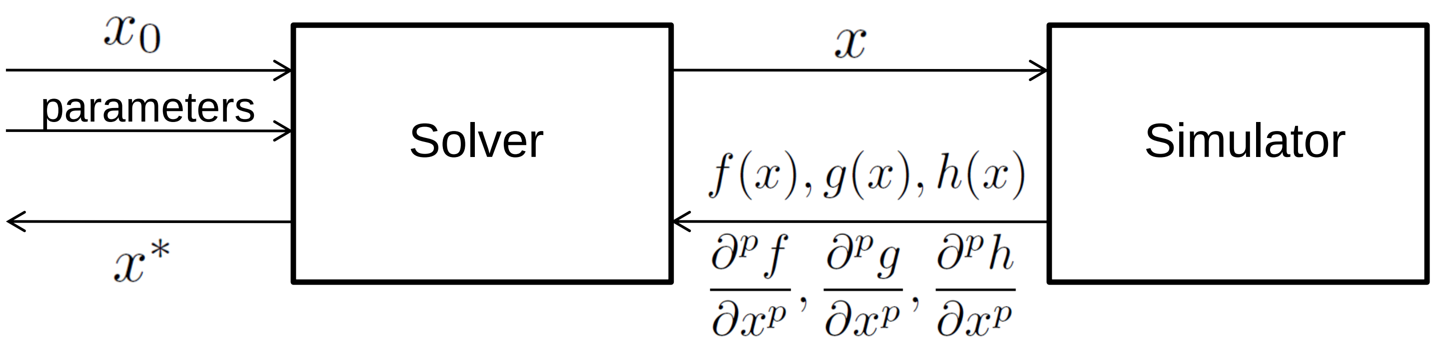
Shooting formulation

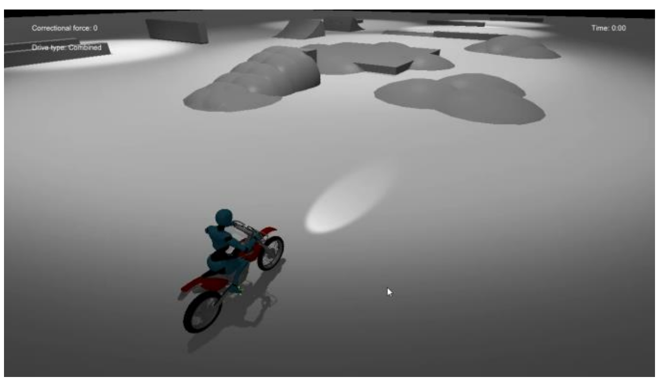
# Control vs estimation




Meet Zongmian Li (first author) at the coffee break

# Optimization



Hamaleinen et al, Aalto

First order method (Hessian not available)
BFGS or sparse Gauss-Newton
~ 10000 variables ~ 1 second available

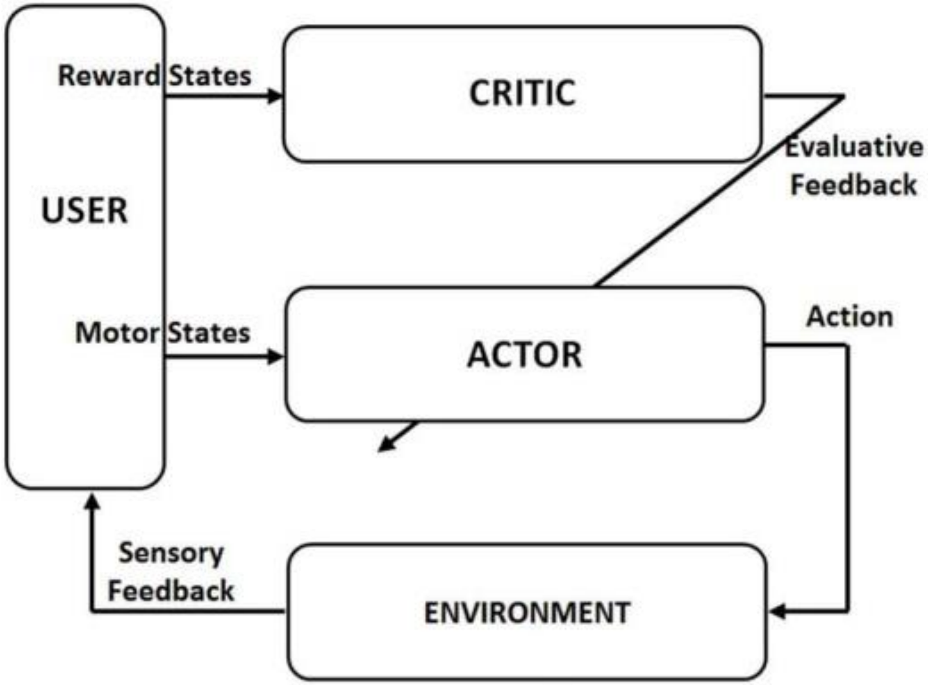
Derivatives?

# Dynamic motor primitives

- Optimize despite evaluation noise
- While carefully exploiting each run


Righetti et al, USC (now NYU)

# Deep reinforcement learning

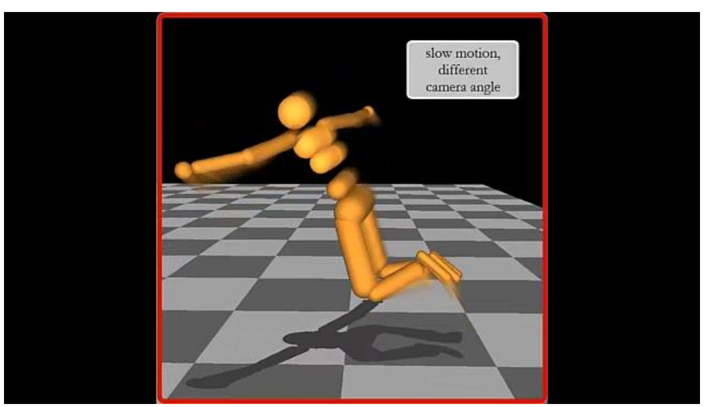



Value function

$$H(x,u) = l(x,u) + V(f(x,u))$$

Policy function

$$u = \Pi(x)$$




Tassa, Todorov et al, UW


Hees, .. , Tassa et al, Deep Mind

# The difficulty is not in learning

- Optimizing a policy boils down to
  - Generate the example dataset (by reinforcement)
  - Exploiting the dataset using machine learning
- The difficulty is in the exploration
- Guided policy search
  - Provide few trajectory examples (demonstration or planning)
  - Reinforce “around” the trajectories

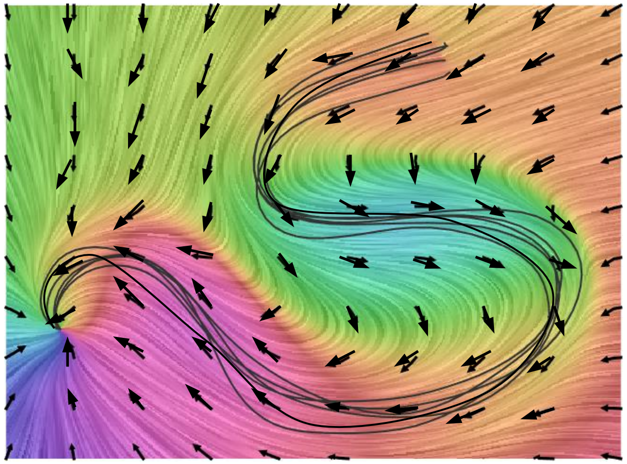
# Guided policy search

Levine et al, UCB/Google

# Optimal control

$$\min_{\substack{X=(Q,\dot{Q}),\\ U=\tau}} \int_0^T l(x_t,u_t)dt$$

so that $\forall t,\ \dot{x}(t) = f(x(t),u(t))$



## Optimizing a trajectory

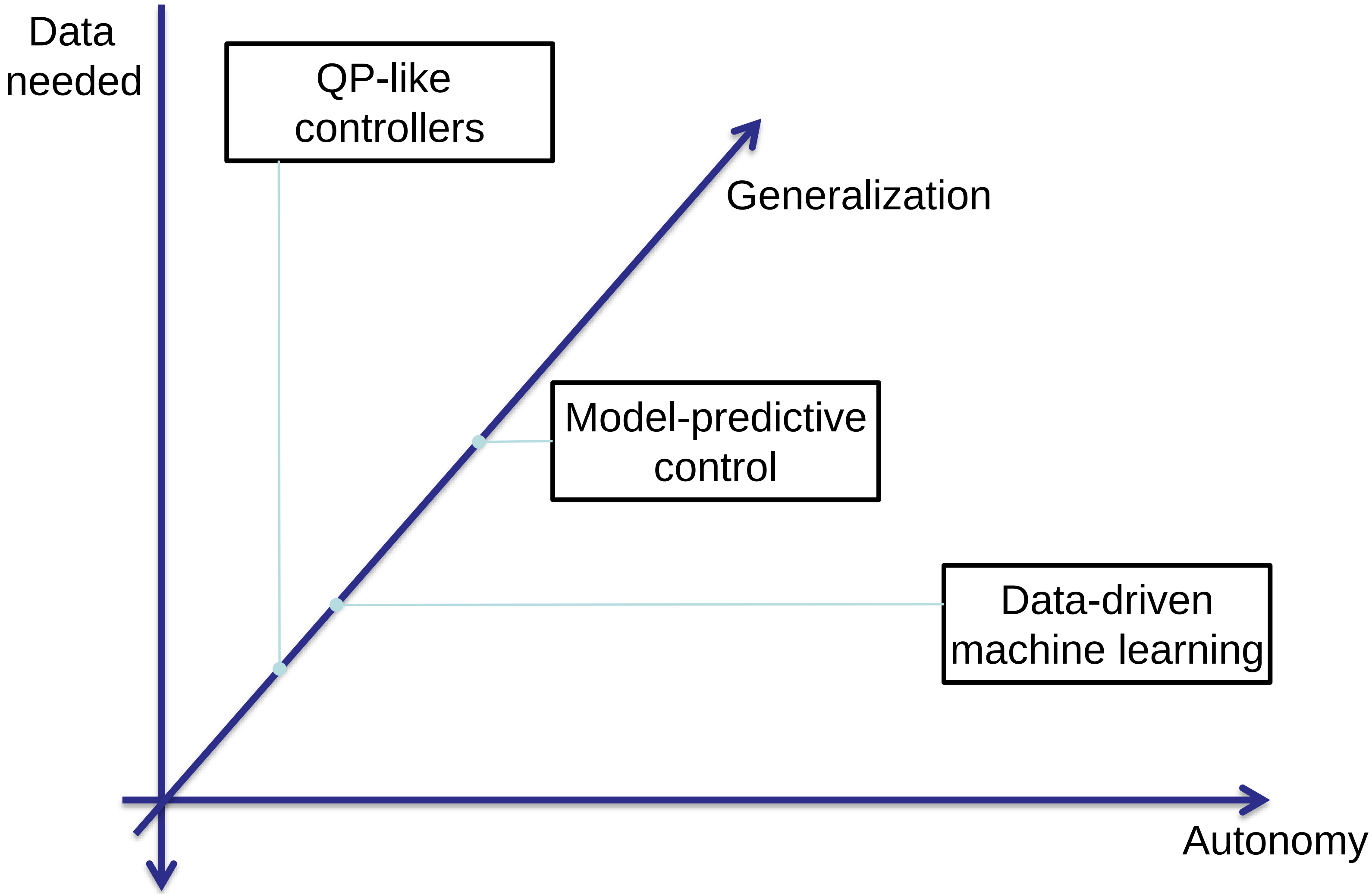
- $U: t \rightarrow u(t)$
- Motion planning

## Optimizing a policy

- $\Pi: x \rightarrow u = \Pi(x)$
- Reinforcement learning

# What kind of model

From Todorov's presentation (25 Nov 2016)



# What is next?

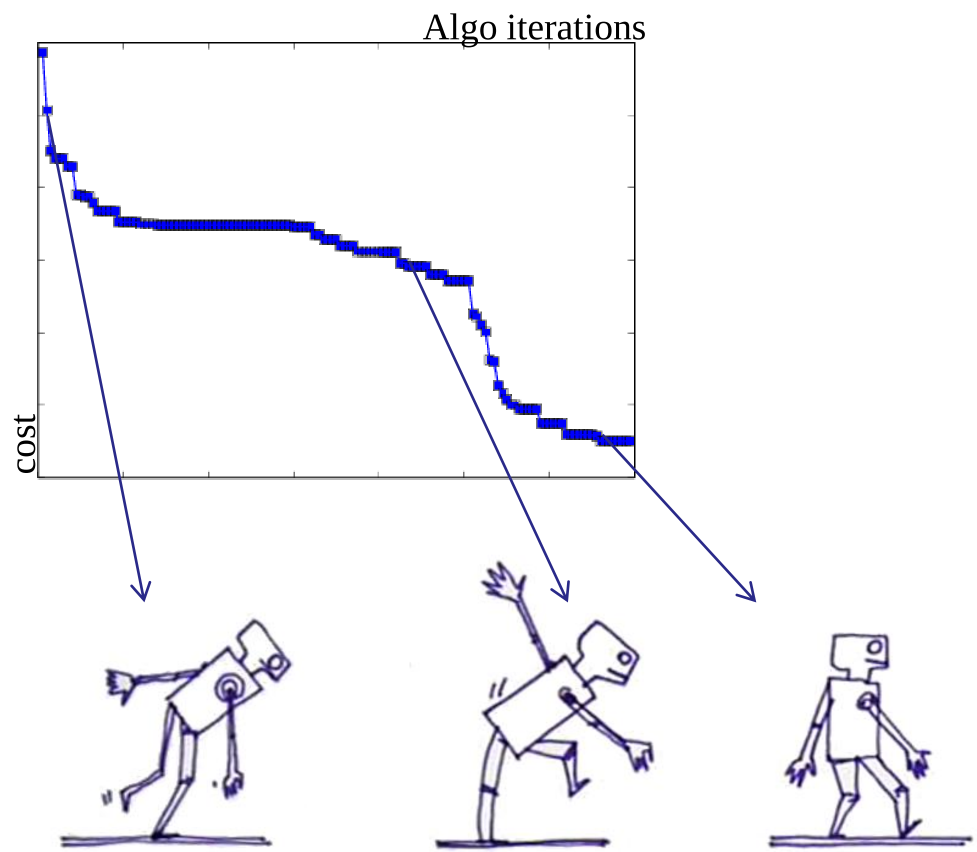
*Planning from and for learning*

*Agile and dexterous robots*

# Example of locomotion

- Motion defined as optimal control

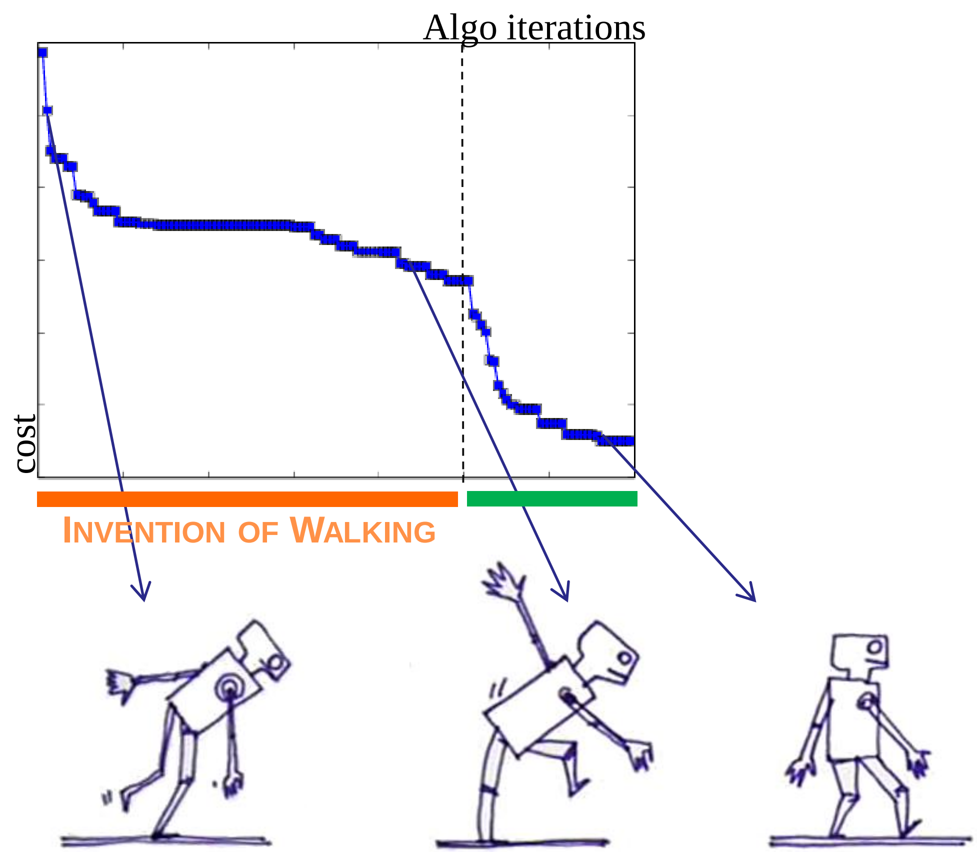
$$\min_{X,U} \int_0^T l(x_t, u_t)\, dt \quad \text{so that} \quad \dot{x}_t = f(x_t, u_t)$$



# Example of locomotion

- Motion defined as optimal control

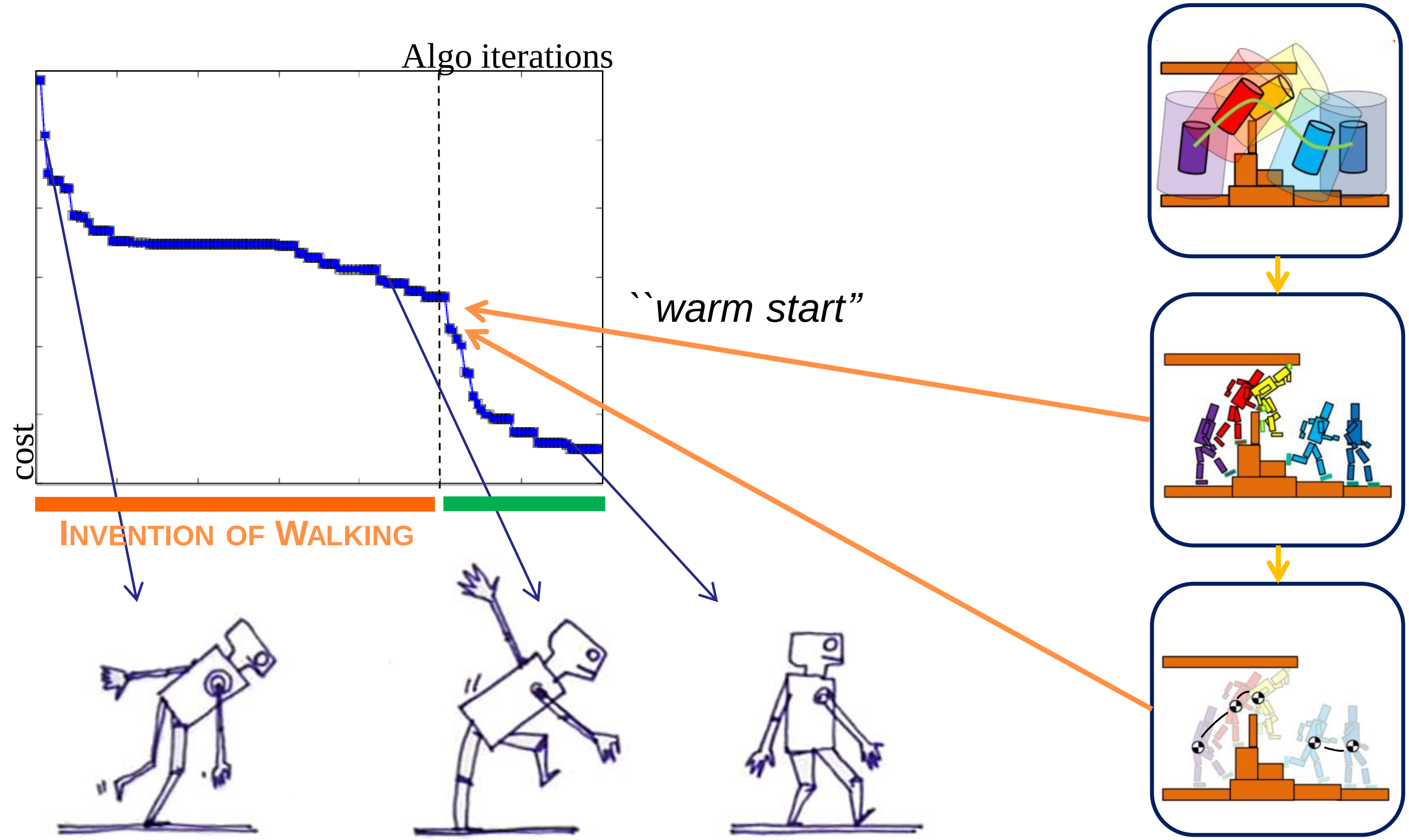
$$\min_{X,U} \int_0^T l(x_t, u_t)\, dt \quad \text{so that} \quad \dot{x}_t = f(x_t, u_t)$$



# Example of locomotion

- Motion defined as optimal control

$$\min_{X,U} \int_0^T l(x_t, u_t)\, dt \quad \text{so that} \quad \dot{x}_t = f(x_t, u_t)$$



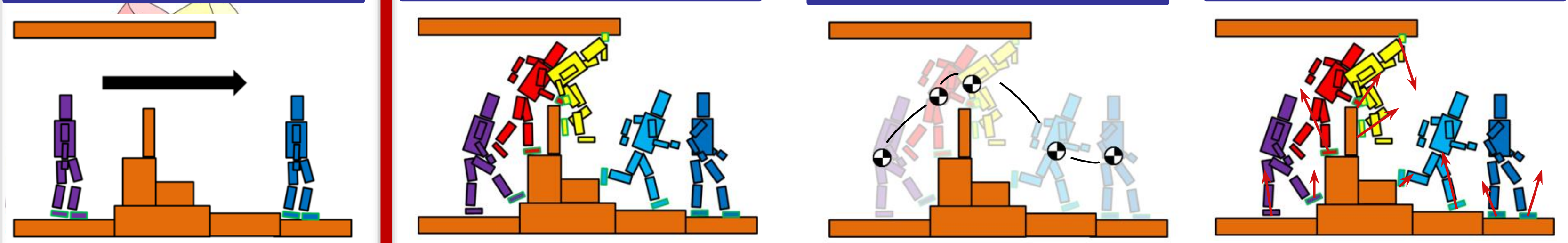
# Workflow

Reachability planner

Sequence of contacts

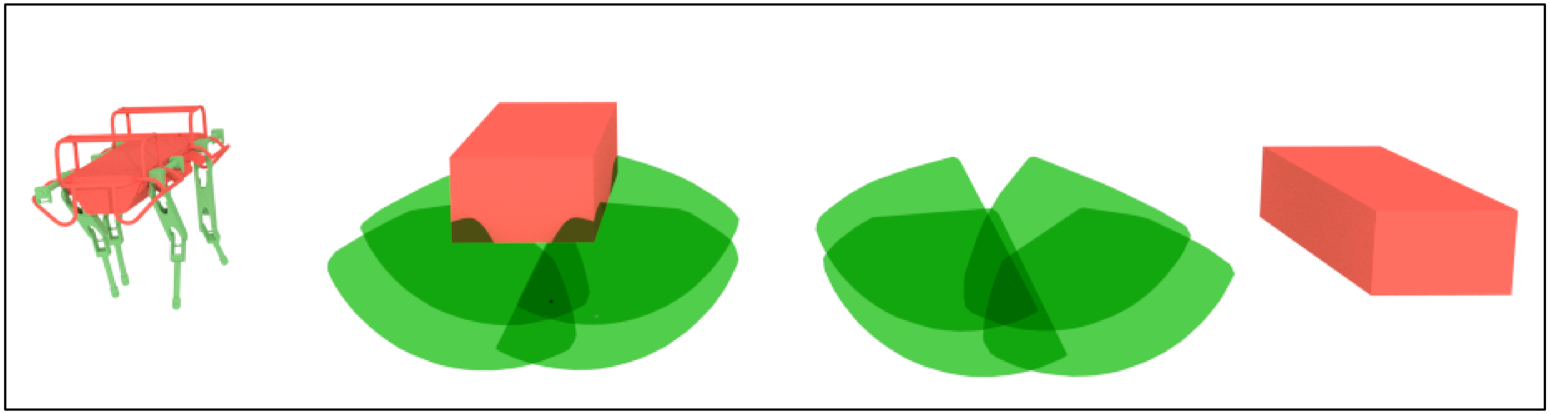
3D Pattern generator

Whole-body force control

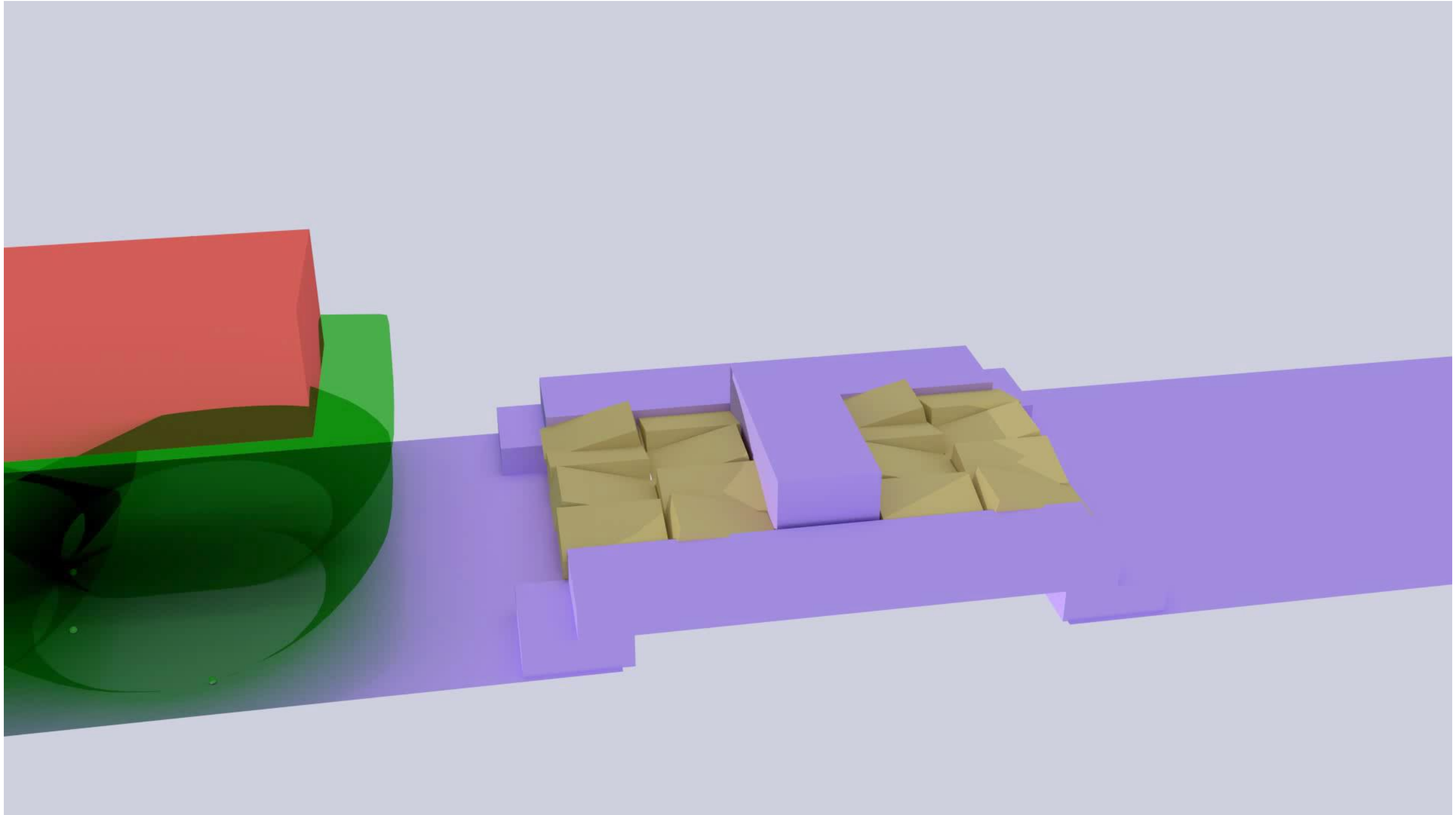


# Reachability-based planner

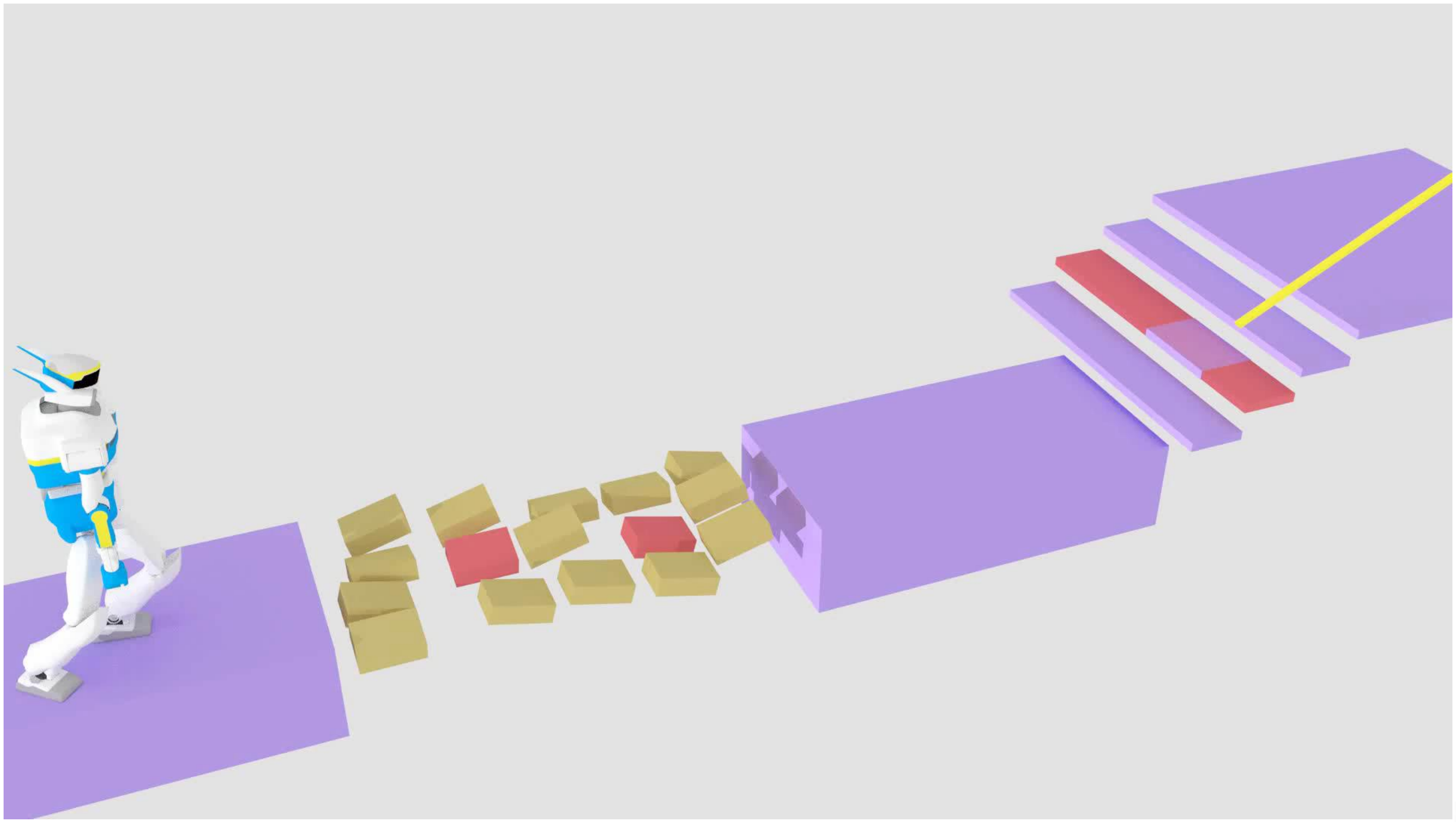
- Necessary vs sufficient cond. for contact feasibility
  - Obstacles **in reachable workspace**...
  - … while **avoiding collisions**



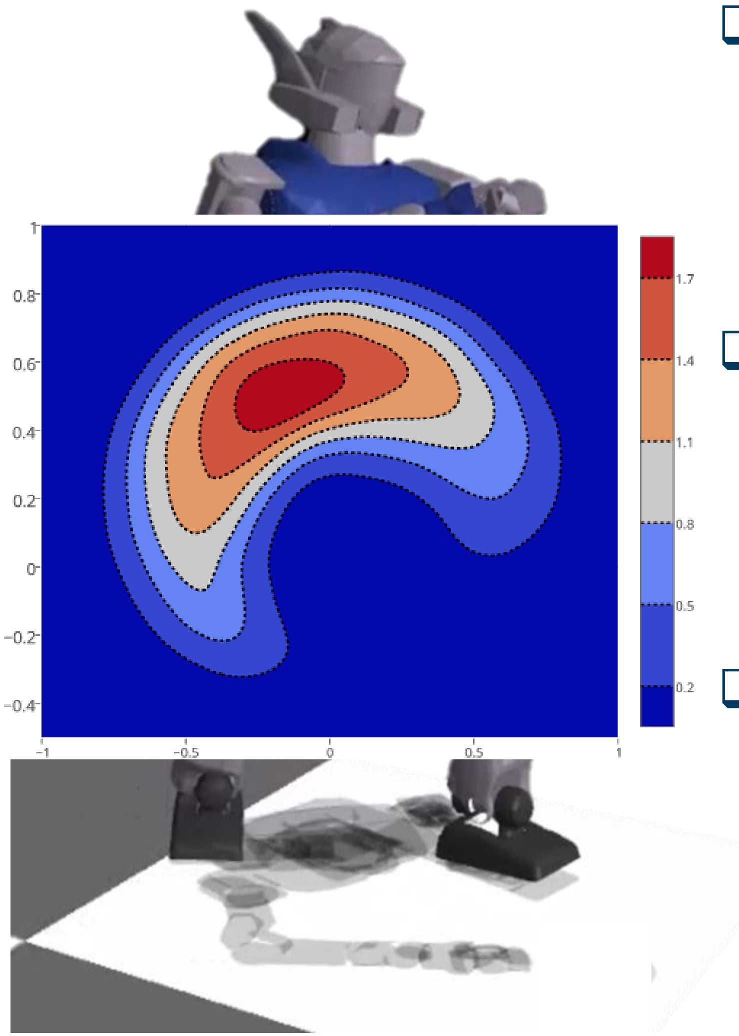
# Reachability-based planner



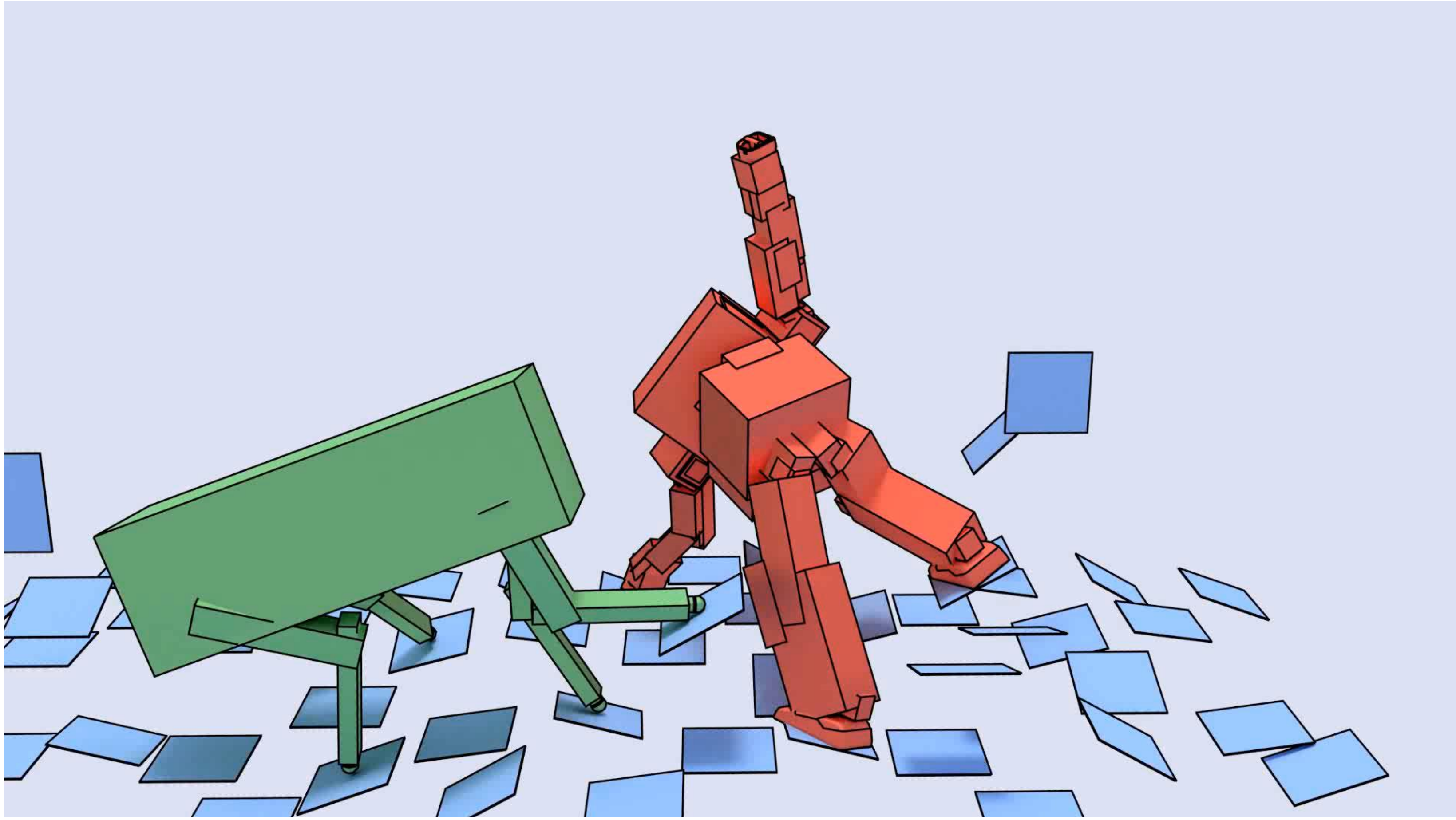
# Real-time contact planner

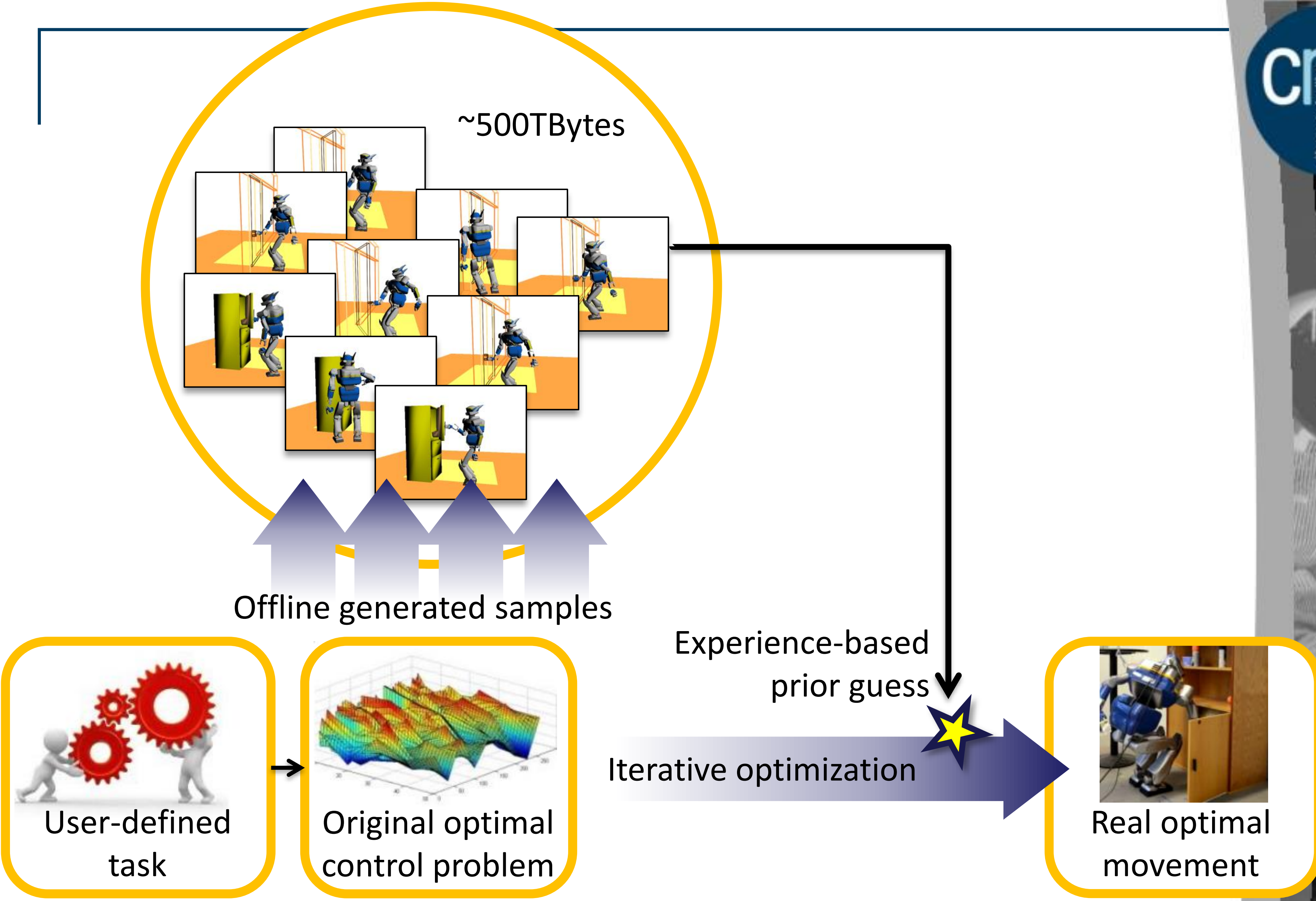


# Whole-body "proxy" constraints

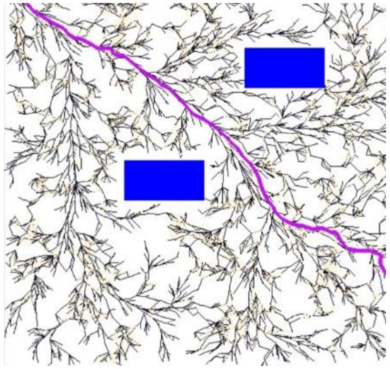



- COM limits
  - Learned from off-line random sampling
- Footstep timings
  - Another variables in the OCP
  - No serious additional cost
- Angular momentum
  - Model of the limits (proxy) needed

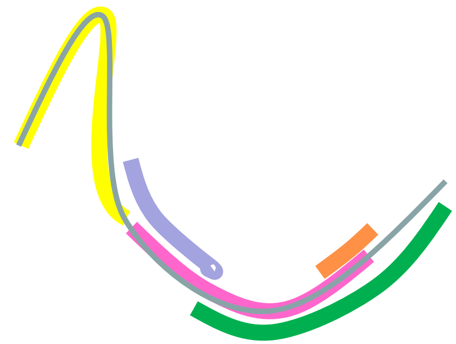



~500TBytes
Offline generated samples
Experience-based
prior guess
User-defined
task
Original optimal
control problem
Iterative optimization
Real optimal
movement

# Roadmap/approximation co-training



Sampling

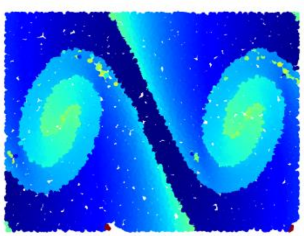


Kino-dynamic
Probabilistic Roadmap
*30-50 states, dense connect*

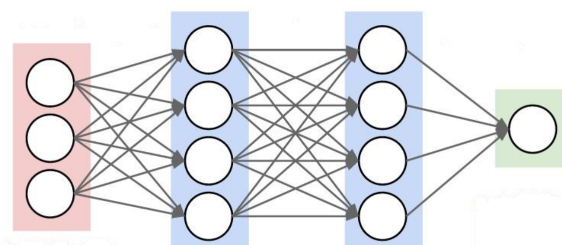
Dataset of subtrajectories
*10-100k items*

Roadmap
extension

Regression
(stoch.grad.)



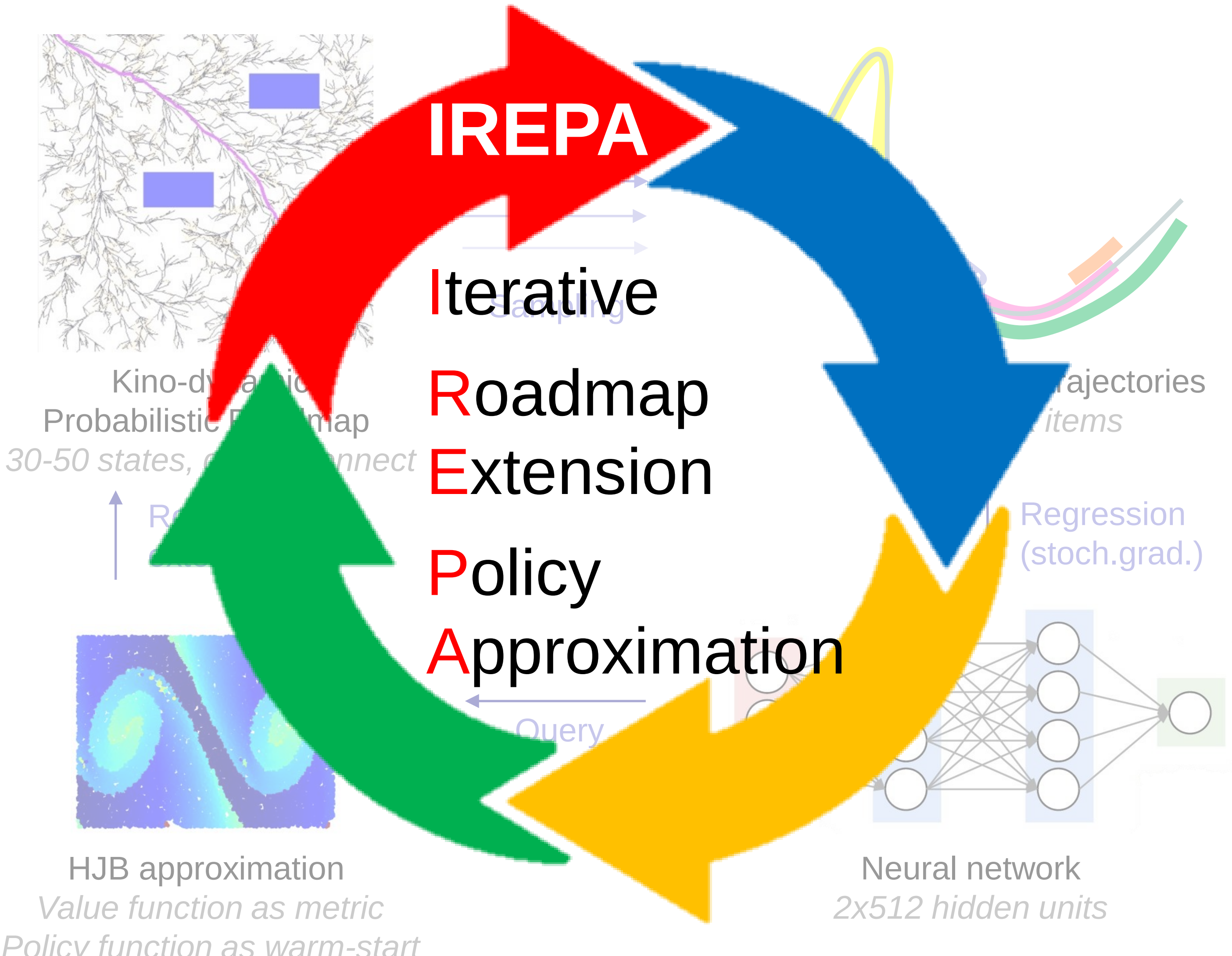
Query



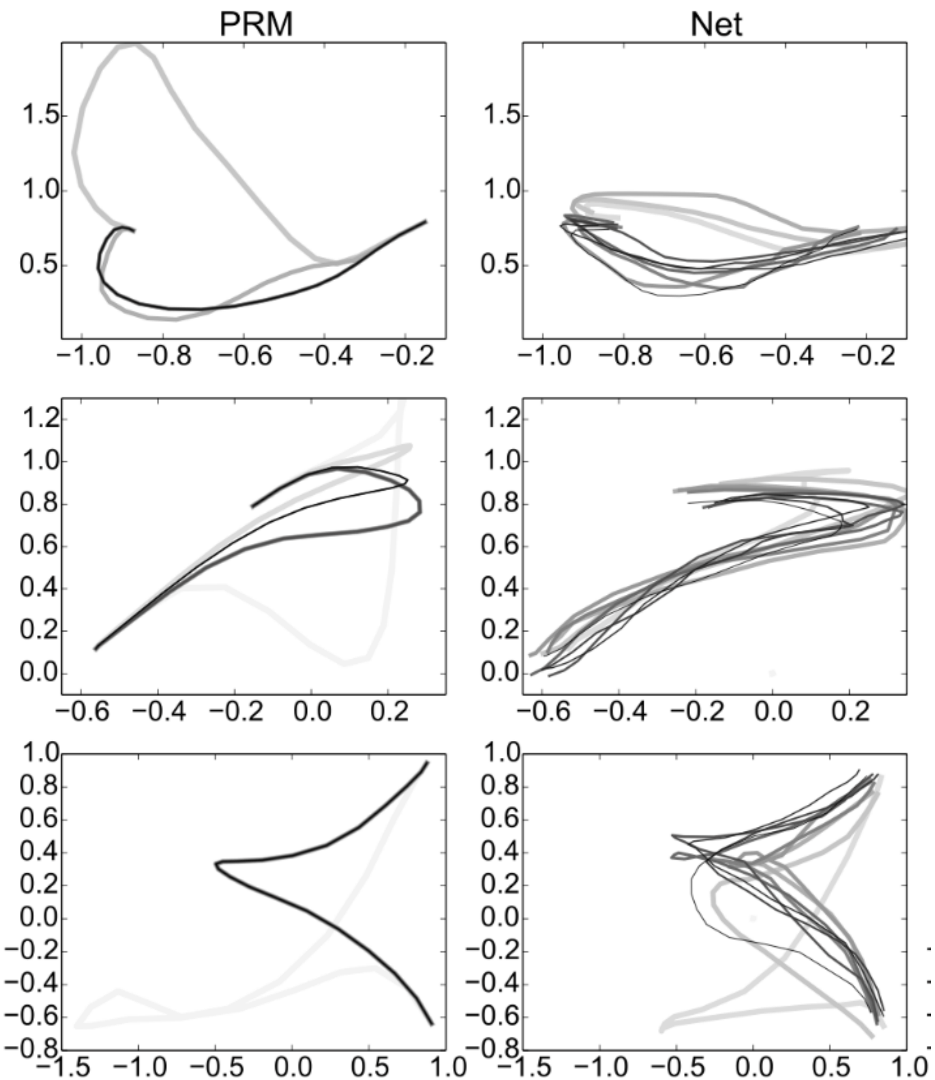
HJB approximation
*Value function as metric*
*Policy function as warm-start*

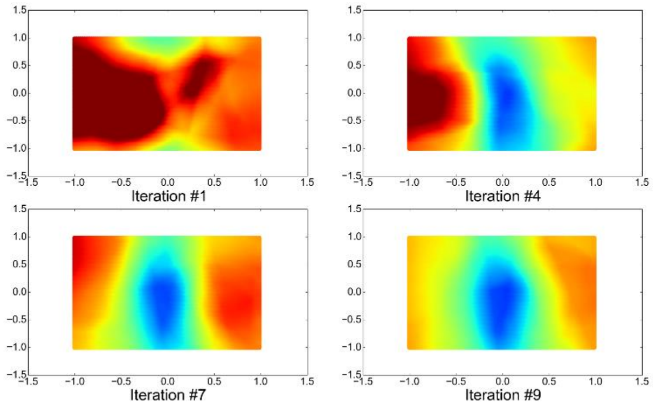
Neural network
*2x512 hidden units*

# Roadmap/approximation co-training

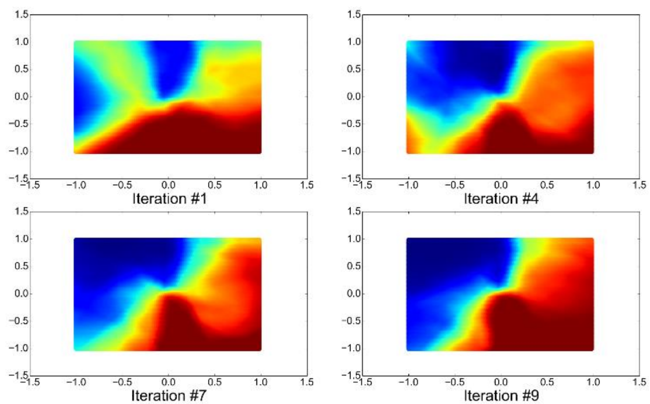



# IREPA: iterations



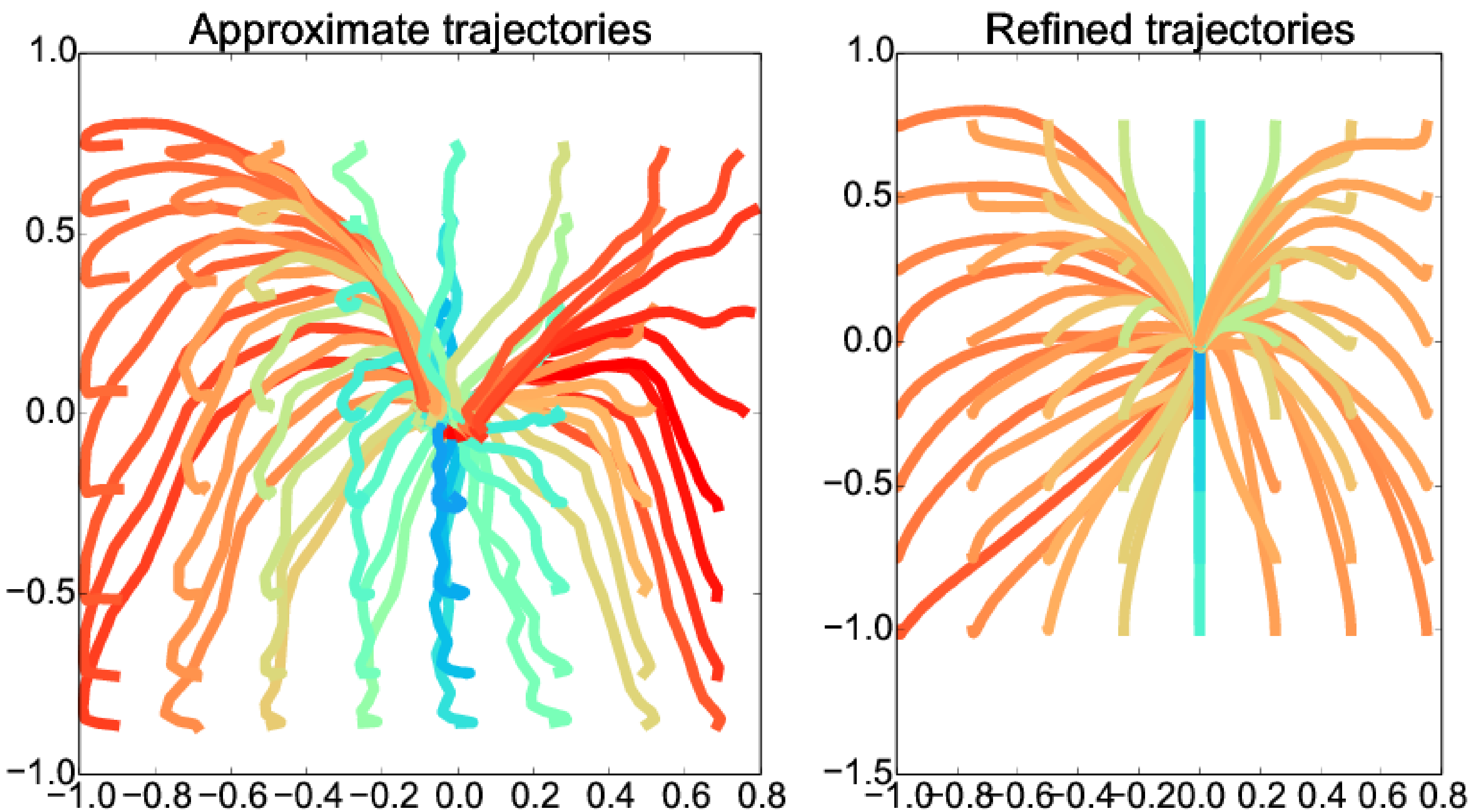




Value network

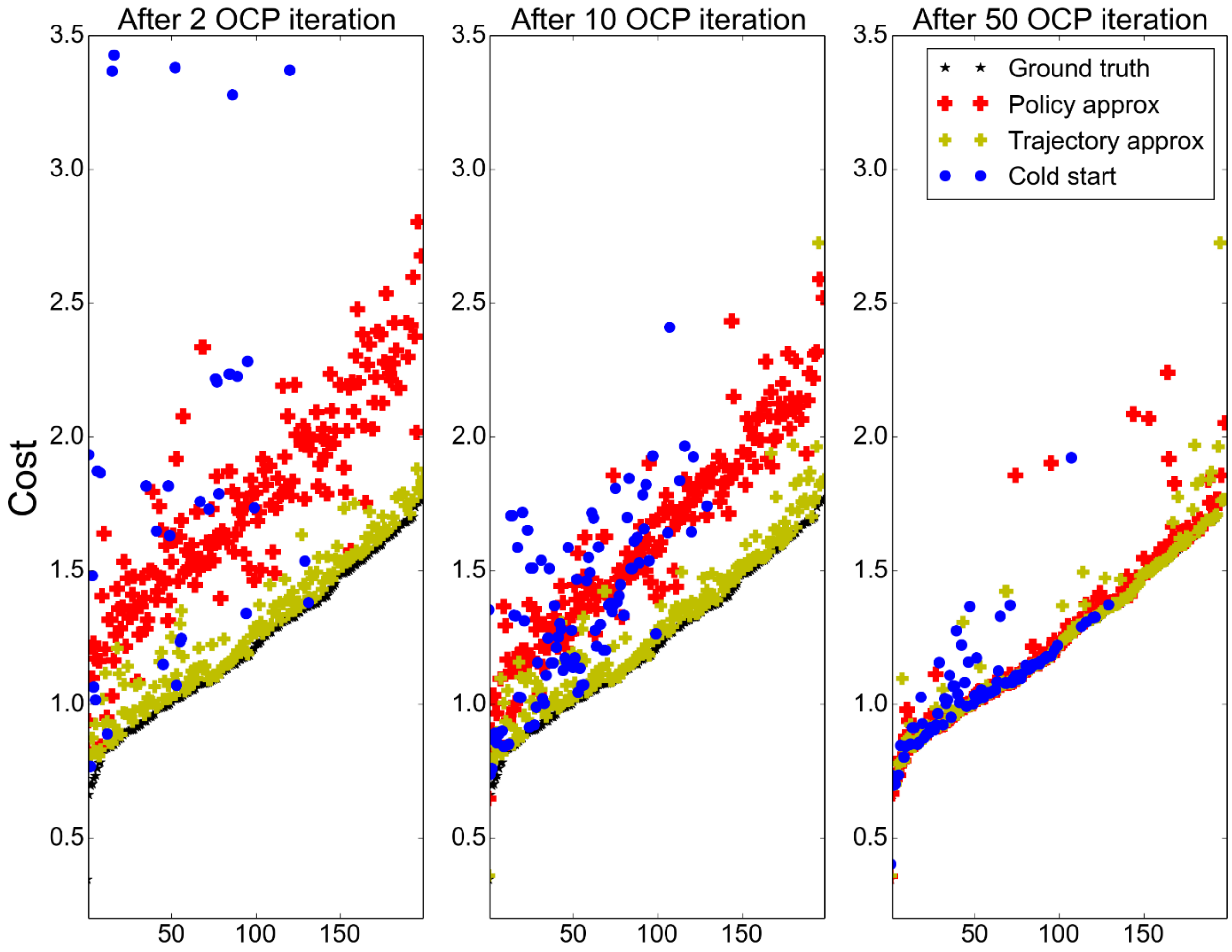


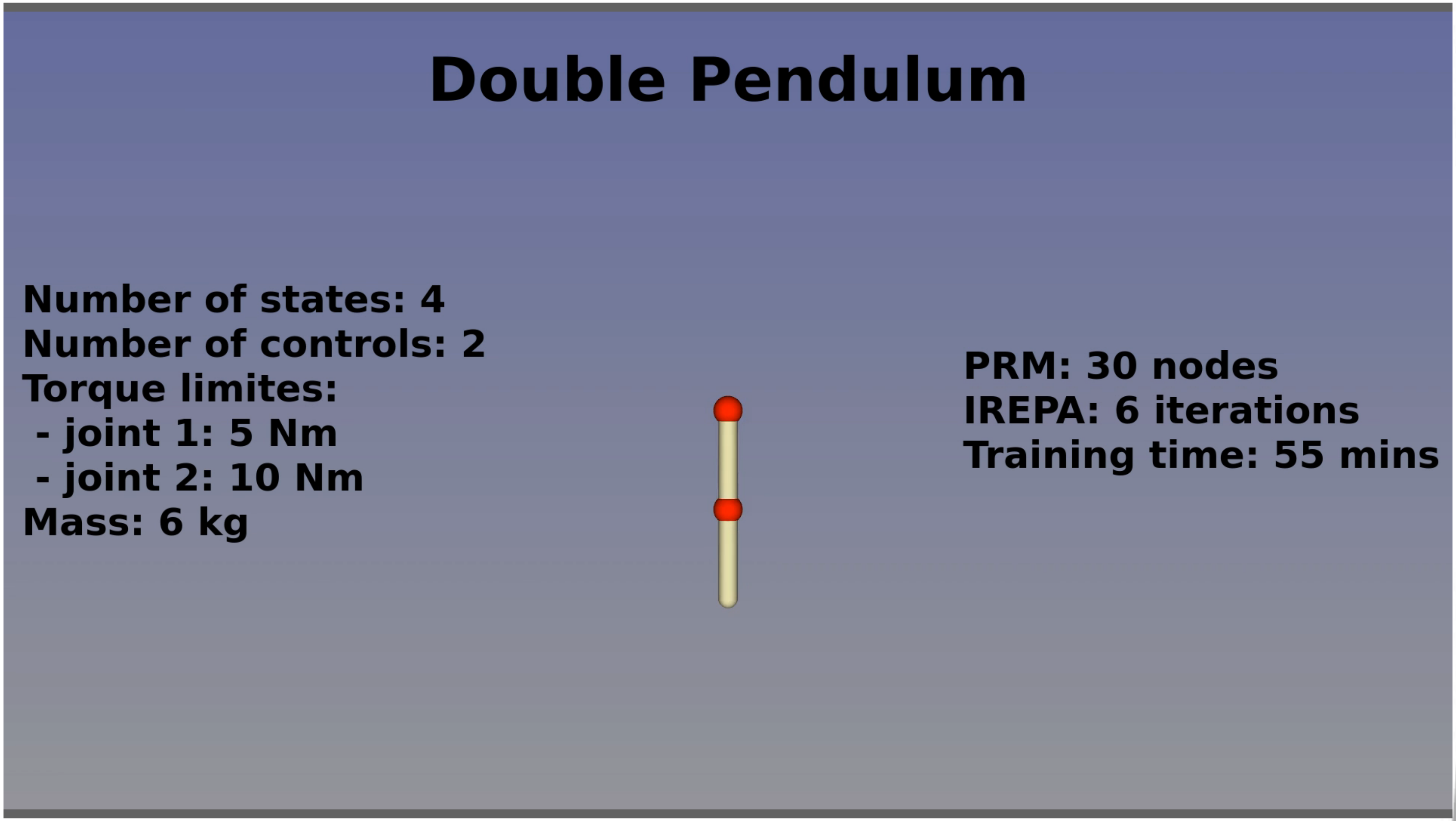

Policy network

# IREPA with MPC



# IREPA with MPC



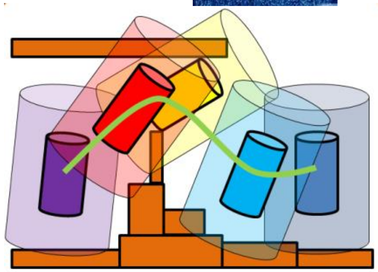

Double Pendulum
Number of states: 4
Number of controls: 2
Torque limites:
- joint 1: 5 Nm
- joint 2: 10 Nm
Mass: 6 kg
PRM: 30 nodes
IREPA: 6 iterations
Training time: 55 mins

# Main messages

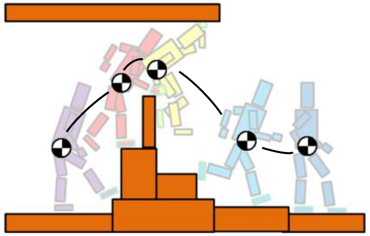
- Memorization is necessary
  - But it is likely not so difficult
- Exploration + transfer  are the difficult points
  - Include sensing and mechanical design
  - Not formulated as canonical IA problems (yet)
- Several well-understood excellent models
  - Equivalent to convolution in vision?
  - Fully-connected ReLU networks are not reasonable
- Your algorithm on a robot: so much additional work
  - But so much additional fun when it works

# Team – join it!

Steve
Tonneau

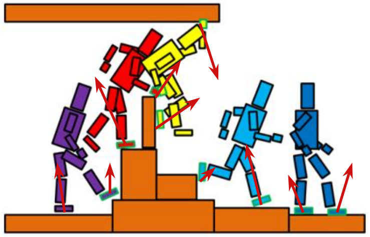


Justin
Carpentier

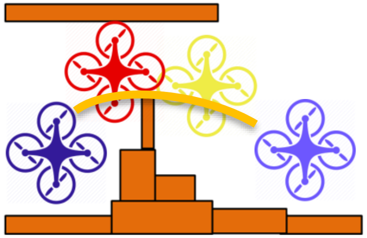


Andrea
Del Prete

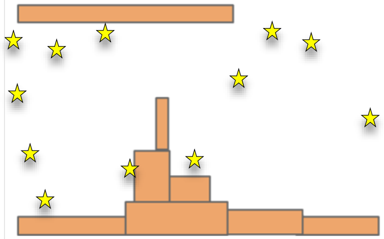
Thomas
Flayols



Joint project with
Olivier Stasse
and
Florent Lamiraux



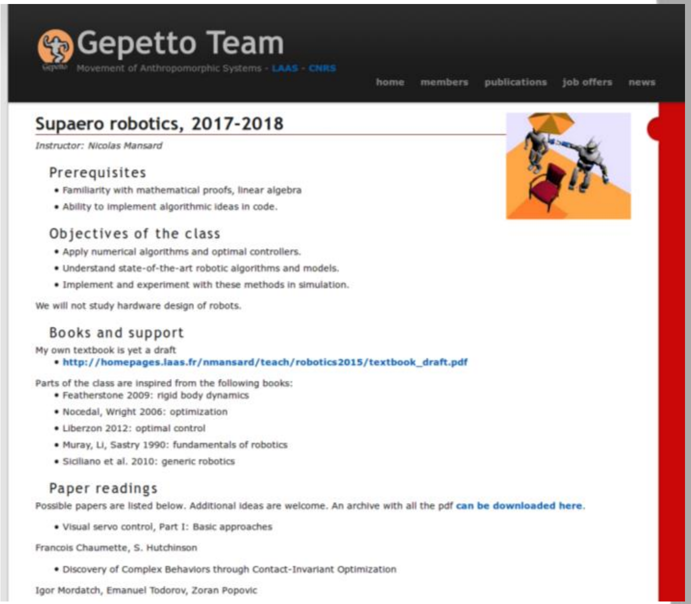
Mathieu
Geisert



Joan Sola
UPC LAAS

# Resources

- Materials from master classes
  http://projects.laas.fr/gepetto/index.php/Teach/Supaero2018



Gepetto Team
Movement of Anthropomorphic Systems - LAAS - CNRS
home members publications job offers news

Supaero robotics, 2017-2018

*Instructor: Nicolas Mansard*

Prerequisites

- Familiarity with mathematical proofs, linear algebra
- Ability to implement algorithmic ideas in code.

Objectives of the class

- Apply numerical algorithms and optimal controllers.
- Understand state-of-the-art robotic algorithms and models.
- Implement and experiment with these methods in simulation.

We will not study hardware design of robots.

Books and support

My own textbook is yet a draft

- http://homepages.laas.fr/nmansard/teach/robotics2015/textbook_draft.pdf

Parts of the class are inspired from the following books:

- Featherstone 2009: rigid body dynamics
- Nocedal, Wright 2006: optimization
- Liberzon 2012: optimal control
- Muray, Li, Sastry 1990: fundamentals of robotics
- Siciliano et al. 2010: generic robotics

Paper readings

Possible papers are listed below. Additional ideas are welcome. An archive with all the pdf can be downloaded here.

- Visual servo control, Part I: Basic approaches

Francois Chaumette, S. Hutchinson

- Discovery of Complex Behaviors through Contact-Invariant Optimization

Igor Mordatch, Emanuel Todorov, Zoran Popovic

- Other text books
  - Featherstone 2009: rigid body dynamics
  - Liberzon 2012: optimal control
  - Muray 1990: fundamentals of robotics
  - Siciliano 2010: generic robotics