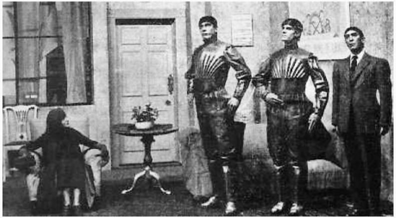
Robots that learn world representations, why and which ones?

Stéphane Caron

March 21, 2023

Researcher at Inria and ENS-PSL

DISCLAIMER



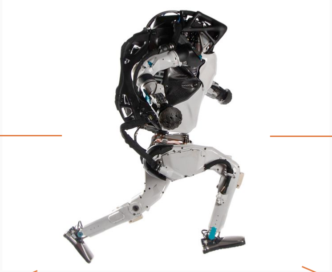
[1] Emanuel Todorov. “Goal directed dynamics”. In: *IEEE International Conference on Robotics and Automation*. 2018.

ROBOTICS IS MULTIDISCIPLINARY

Optimization

Machine Learning

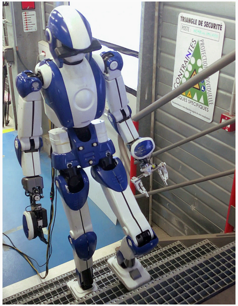
Perception



Control

Electrical Engineering

Mechanical Engineering


TRIANGLE DE SECURITÉ
POSTE
CONTRAINTES
PROTECTION

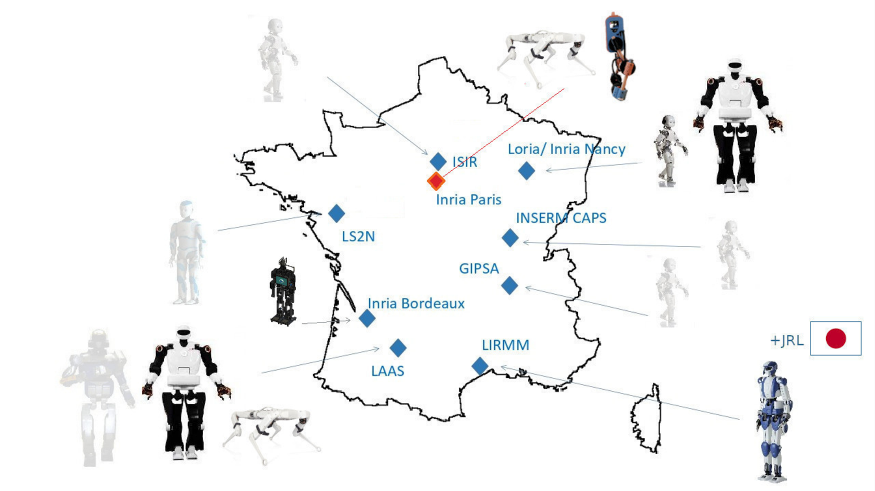
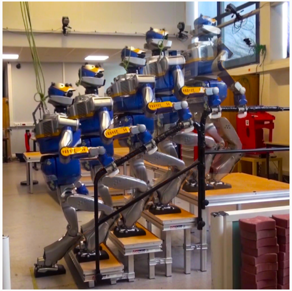


Figure 1: Research labs with legged robots (France, 2023)

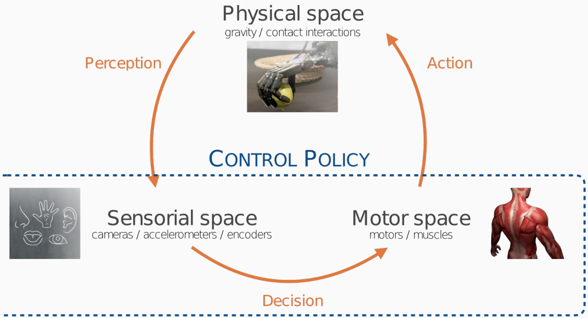
OPTIMIZATION PLAYS A KEY ROLE IN ROBOTICS



$$\begin{aligned} \underset{x}{\text{minimize}} \; & f(x) \\ \text{subject to} \; & g(x) = 0 \\ & h(x) \leq 0 \end{aligned}$$

- Optimal design, physics simulation
- Inverse kinematics, inverse dynamics
- Model predictive control, reinforcement learning
- State observation, estimation (SLAM, etc.)

PERCEPTION-DECISION-ACTION LOOP



Sensorial space
cameras / accelerometers / encoders

sensors data

$$u_t = \pi(y_t)$$

Control policy

Motor space
motors / muscles

motor commands

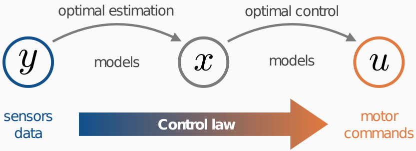
OPTIMAL CONTROL VS POLICY LEARNING

Optimal Control

$$\min_{x(.),u(.)} \int_0^T l_t\left(x(t), u(t)\right) dt$$

$$\dot{x}(t) = f_t(x(t), u(t))$$

$$x(0) = x_0$$

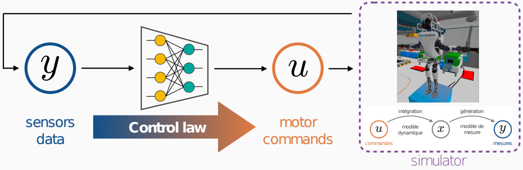


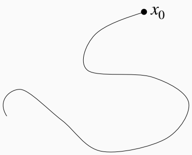
Policy Learning

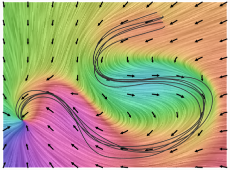
$$\min_{\theta} \mathbb{E}_{\xi}\left[\int_0^T l_t\left(x(t), \pi_\theta(x(t), \xi(t))\right) dt\right]$$

$$\dot{x}(t) = f_t(x(t), \pi_\theta(x_t, \xi_t))$$

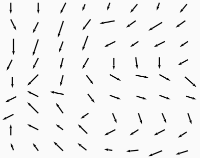
$$x(0) \sim \xi_0$$





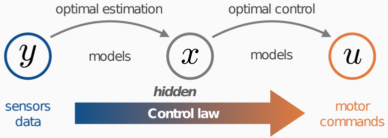


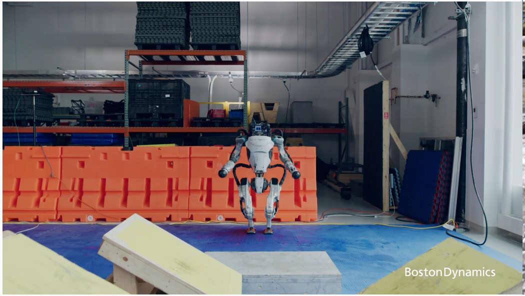
Source: Callinon et al.



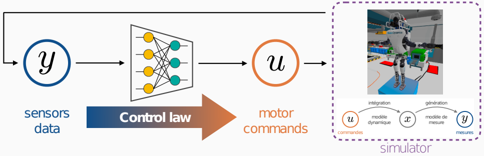
OPTIMAL CONTROL VS POLICY LEARNING: ROBOTS

Optimal Control





Policy Learning



Robotics @ WILLOW

Perception, Learning and Control

THE WILLOW TEAM

Faculties:

* Ivan Laptev (Inria, 100%)
* Justin Carpentier (Inria, 100%)
* Jean Ponce (Inria, 100%)
* Cordelia Schmid (Inria, 50%)
* Stéphane Caron (Inria, 100%)

External collaborators:

* Josef Sivic (CTU)

Post-docs:

* Shizhe Chen
* Alexandre Araujo

Engineers:

* Etienne Arlaud (Inria, SED-Paris)
* Pierre-Guillaume Raverdy (Inria, SED-Paris)

Visitors:

* Kateryna Zorina (CTU Prague)

Assistant:

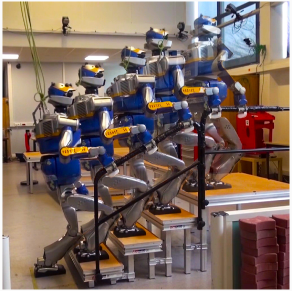
* Julien Guieu

PhD students:

* Yann Labbé
* Antoine Bambade
* Robin Strudel
* Zongmian Li
* Bruno Lecouat
* Yann De Mont Marin
* Pierre-Louis Guhur
* Oumayma Bounou
* Elliot Chane-Sane
* Antoine Yang
* Quentin Le Lidec
* Guillaume Le Moing
* Hugo Cisneros
* Alaaeldin El-Nouby
* Thomas Chabal
* Adrien Bardes
* Wilson Jallet
* Nicolas Chahine
* Elliot Vincent
* Théo Bodrito
* Matthieu Futeral-Peter
* Zerui Chen
* Fabian Schramm

- **Optimization:** numerical solvers for robotics (AB, WJ)
- **Simulation:** contact simulation and non-smooth differentiation (QLL, LM)
- **Perception:** object pose estimation, world representations (YL)
- **Control:** model predictive control for manipulation and locomotion (WJ)
- **Learning:** reinforcement learning, interplay with optimal control (FS)

OPTIMIZATION PLAYS A KEY ROLE IN ROBOTICS



$$\begin{aligned} \underset{x}{\text{minimize}}\ & f(x) \\ \text{subject to}\ & g(x) = 0 \\ & h(x) \leq 0 \end{aligned}$$

- Optimal design, physics simulation
- **Inverse kinematics**, inverse dynamics
- Model predictive control, **reinforcement learning**
- State observation, estimation (SLAM, etc.)

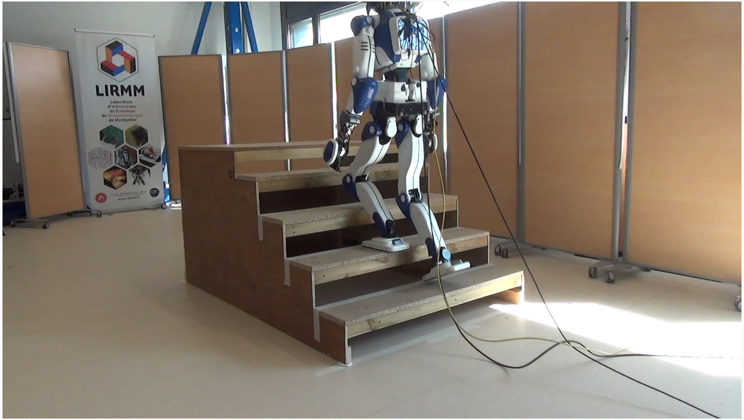
Inverse kinematics

Make a robot **move** (motion) to achieve some **tasks** (control).

Examples:

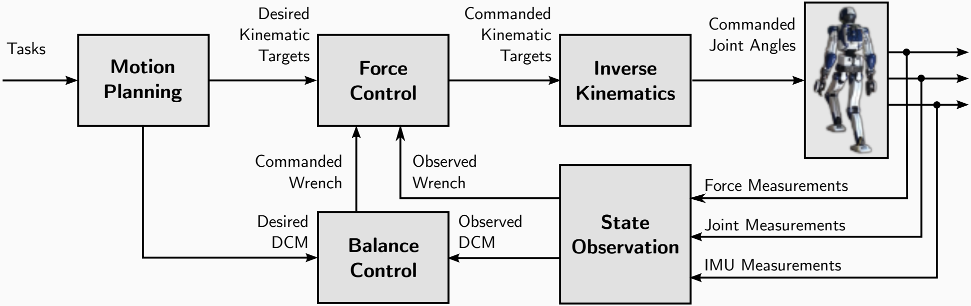
- Locomotion: change position w.r.t. the world
- Manipulation: change the pose of an object w.r.t. the robot
- Folding: change the configuration of a deformable object
- Breaking: add free-flyer joint to another system ;)

Key part of the work: task formulation.



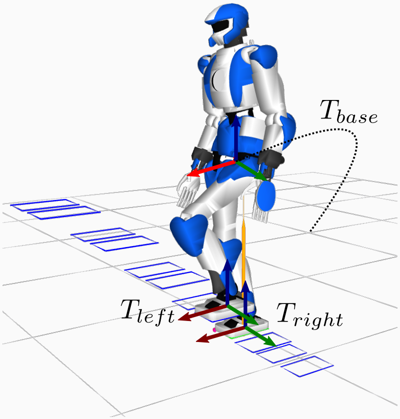
Video: https://www.youtube.com/watch?v=vFCFKAunsYM

CONTROLLER PIPELINE



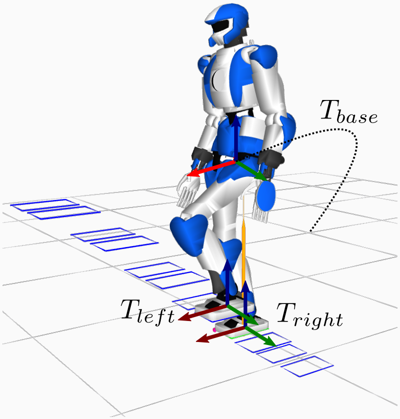
[2]Stéphane Caron, Abderrahmane Kheddar, and Olivier Tempier. "Stair Climbing Stabilization of the HRP-4 Humanoid Robot using Whole-body Admittance Control". In: *IEEE International Conference on Robotics and Automation*. May 2019.

- **Goal:** move to fulfill a set of tasks
- **Configuration space:** $q \in \mathcal{C}$ is a manifold
- **Joints:** q_i in sub-manifolds, *e.g.* $q_i \in SO(2)$
- **Foot task:** move ${}_0T_{foot}(q) \in SE(3)$ to ${}_0T_{foot*}$
- **Error:** $e_i(q) = \log_6({}_{foot}T_{foot*}(q))$
  - Goal: $\forall i, e_i(q) \to 0$
- **Jacobian:** $J_i(q) = \nabla_q e_i(q)$
- **Dynamics:** $\dot{e}_i(q) = J_i(q)v = -\alpha e_i(q)$
  - No constraint $\Rightarrow e_i(q) \to 0$ exponentially
- **Constraints:** $q \in \mathcal{C}_{free} \subset \mathcal{C}$
- **Multi-tasks:** weighted / lexicographic optim.



$$\begin{aligned} \underset{v}{\text{minimize}} \quad & \textstyle\sum_i \|J_i(q)v + \alpha e_i(q)\|^2_{W_i} \\ \text{subject to} \quad & G(q)v \leq h(q) \\ & A(q)v = b(q) \end{aligned}$$

- Gauss-Newton approximation
- Velocity v is **both** the step and motor command: moving is thinking
- No line search!
- Levenberg-Marquardt damping, but numerical instability when $J_i(q)$ singular



QUADRATIC PROGRAMMING

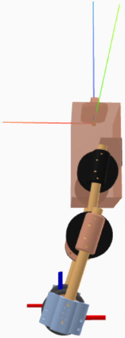
$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \frac{1}{2} x^T P x + q^T x \\ \text{subject to} \quad & Gx \leq h \\ & Ax = b \end{aligned}$$

```python
from qpsolvers import solve_qp

M = np.array([[1., 2., 0.], [-8., 3., 2.], [0., 1., 1.]])
P = M.T @ M  # this is a positive definite matrix
q = np.array([3., 2., 3.]) @ M
G = np.array([[1., 2., 1.], [2., 0., 1.], [-1., 2., -1.]])
h = np.array([3., 2., -2.])
x = solve_qp(P, q, G, h, solver="proxqp")
```

Setup: `pip install proxsuite qpsolvers`

TRY IT ON ANY ROBOT

```python
from pink import solve_ik
from loop_rate_limiters import RateLimiter

rate = RateLimiter(frequency=200)
dt = rate.period
for t in np.arange(0., 42., dt):
    update_targets(tasks, t, dt)
    v = solve_ik(q, tasks, dt, solver="proxqp")
    q.integrate(v, dt)
    rate.sleep()
```



Setup: `pip install loop-rate-limiters proxsuite pin-pink`

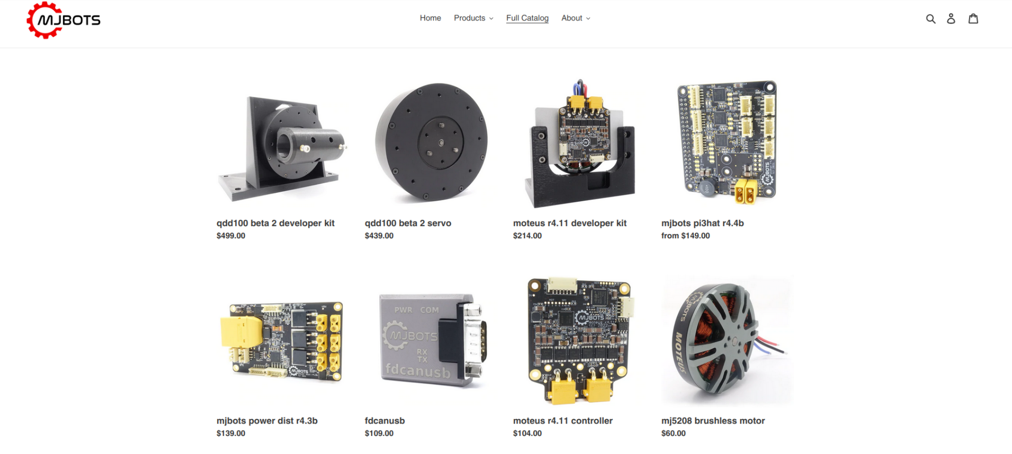
REALLY, ANY ROBOT

List: `pip install robot_descriptions`

Reinforcement learning

Data!

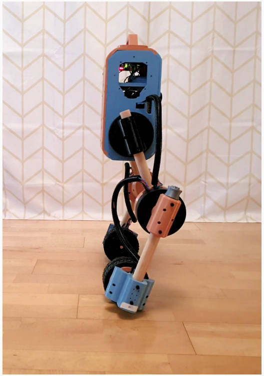
MJBOTS STACK



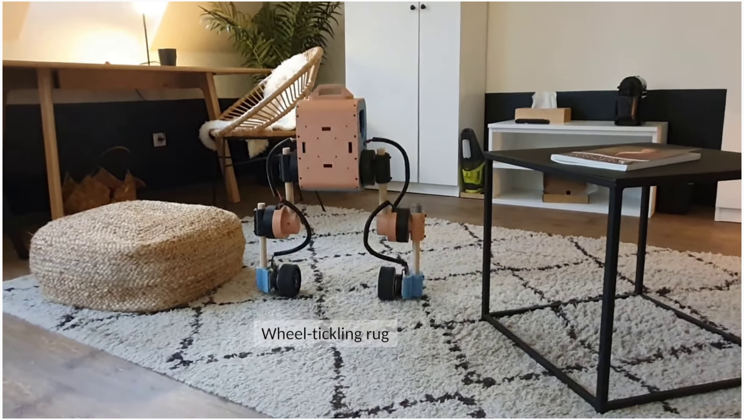
Store: `https://mjbots.com/`

Wheeled biped:

- Joints: 6 (hips, knees, wheels)
- Total mass: 5.4 kg
- Computer: Raspberry Pi
- Print time: $\leq$ 100 h
- Actuators + electronics: 3,000 €
- **Autonomy:** 3–4 h with 5.0 Ah battery
- Sensors: IMU, joint position, velocity, torque
- Extensibility: 12 actuators, external sensors



https://hackaday.io/project/185729-upkie-homemade-wheeled-biped-robot

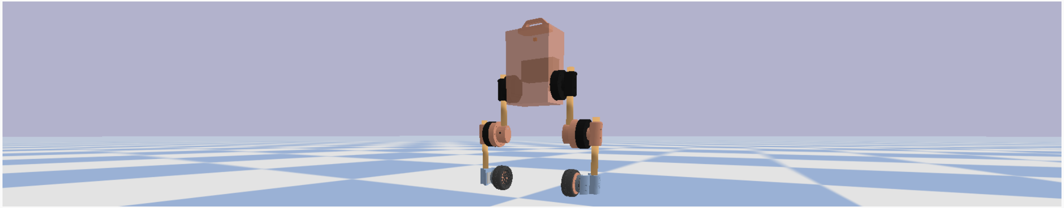


Video: https://www.youtube.com/watch?v=NO_TkHGS0wQ

Software that comes batteries included:

```
git clone https://github.com/tasts-robots/upkie_locomotion.git
cd upkie_locomotion
./tools/bazelisk run -c opt //agents/ppo_balancer:train
```

Bazel will download and build everything (no installation required on Linux).



Repository: https://github.com/tasts-robots/upkie_locomotion

```python
import gym
import upkie_locomotion.envs

upkie_locomotion.envs.register()
env = gym.make("UpkieWheelsEnv-v1")  # same for sim or real robot
observation = env.reset(seed=42)
action = 0.0 * env.action_space.sample()
for step in range(1_000_000):
    observation, reward, done, _ = env.step(action)
    if done:
        observation = env.reset()
    pitch = observation[0]
    action[0] = 10.0 * pitch
env.close()
```

Setup: `pip install upkie_locomotion`

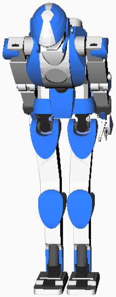
World representations?

ANOTHER SOURCE OF DATA

TAKEAWAYS

1. **Optimization** plays a key role in robotics
    - Come test your algorithms!
2. Building robots has become 10x **cheaper**
    - Come get relevant problems and data!
3. **Open source** hardware and software
    - Just `pip install`, no strings attached

THANK YOU FOR YOUR ATTENTION!



Bibliography

[CKT19] Stéphane Caron, Abderrahmane Kheddar, and Olivier Tempier. "Stair Climbing Stabilization of the HRP-4 Humanoid Robot using Whole-body Admittance Control". In: *IEEE International Conference on Robotics and Automation*. May 2019.

[Tod18] Emanuel Todorov. "Goal directed dynamics". In: *IEEE International Conference on Robotics and Automation*. 2018.