Artificial & Natural Movement

Nicolas Mansard – LAAS-CNRS
Immense progress in 5 years

Unitree quadruped

BostonDynamics Atlas

Cf Le dessous des images “Atlas, le robot star de Boston Dynamics”

Artificial & Natural Movements
Good old control heuristics

... motion programed by tasks

Regularization

Straight torso

Flying foot

Center of mass

Support foot

Task estimate

Robot state

Motor control

Straight torso

Center of mass

Flying foot trajectory

Flat support foot
Whole-body control decision

High-level function objective
*Go that way, bring me that object, open that door*

Automatic motion intelligence

Model of the motion capabilities
*From a known state, this control will bring the robot in that new state*
Predictive control

Decide: future robot trajectory

By optimizing an objective function (eg minimum energy)

Imposing:
- Known initial state
- Known evolution model (simulator)
- ... and other constraints
Predictive control

Original artwork by Michele Carminati, commissioned by Marco M. Nicotra (U. Colorado Boulder)

\[
\begin{align*}
\min_{x=(Q,Q), U=\tau} & \int_{0}^{T} l(x_t, u_t) \, dt \\
\text{so that} & \\
x(0) = \hat{x} \\
\dot{x}(t) = f(x(t), u(t)), \quad \forall t=0..T
\end{align*}
\]

Simulator

Artificial & Natural Movements
Efficient solvers ... 

- Features expected from a good optimal control solver
  - Stable prediction: **multiple shooting**
  - Sparsity: **differential dynamic programming**
  - Strict constraints: **augmented Lagrangian**
  - Our solver incorporates all three!

- Performance on real case studies
  - 4 trot cycles for a quadruped: 8K vars, 12 iterations, **9ms / iter**
  - 2 steps for a humanoid: 12K vars, 18 iterations, **13ms / iter**
... for efficient problems

<table>
<thead>
<tr>
<th>torso</th>
<th>Right leg</th>
<th>Left leg</th>
<th>Right arm</th>
<th>Left arm</th>
<th>Head</th>
</tr>
</thead>
</table>

\[
\begin{bmatrix}
L_{xx} & L_{xx} & -I & F_x^T \\
L_{xx} & L_{xx} & -I & F_x^T \\
L_{uu} & L_{uu} & F_u^T & -I \\
I & -I & F_u & -I \\
F_x & -I & F_u & F_u
\end{bmatrix}
\]

Pinocchio

BSD

BSD-2 License

https://github.com/stack-of-tasks/pinocchio

Justin Carpentier
Inria Paris / PRAIRIE

BSD License
Off-line
Optimization and encoding

\[
\min_{x=(Q,Q), \quad U=\tau} \int_0^T l(x_t, u_t) dt
\]

s.t. \( x(0) = \hat{x} , \quad \dot{x}(t) = f(x(t), u(t)), \quad \forall t=0..T \)

On-line
Whole-body predictive control

High-freq by local convergence

https://github.com/MeMory-of-MOtion/docker-loco3d
Efficient solver ... for efficient problems

Optimize
1 sec of preview
every 1 ms
(2000 variables)

Ludovic Righetti
Artificial & Natural Movements
Efficient solver ... for efficient problems

Artificial & Natural Movements
Efficient solver ... for efficient problems

Artificial & Natural Movements

Thomas Corberes  Steve Tonneau
\[
\min_{x = (Q, \dot{Q}), \quad u = \tau} \int_0^T l(x_t, u_t) dt \\
\text{s.t.} \quad x(0) = \hat{x}, \\
\dot{x}(t) = f(x(t), u(t)), \forall t = 0..T
\]

Trajectory optimization

\( U: t \rightarrow u(t) \)

Motion planning

Policy optimization

\( \Pi: x \rightarrow u = \Pi(x) \)

Reinforcement learning

Artificial & Natural Movements
Memory of motion

\[
\min_{x=(Q,Q), u} \int_0^T l(x_t, u_t) dt \\
\text{s.t.} \quad x(0) = \hat{x}, \quad \dot{x}(t) = f(x(t), u(t)), \quad \forall t=0..T
\]

Belman principle

\[
\Pi(x) = \arg\min_u Q(x, u) \\
Q(x, u) = l(x, u) + Q(x', \Pi(x))
\]
\[
\min_{\mathbf{x}=(q,q'), u} \int_0^T l(x_t, u_t) dt \\
\text{s.t.} \quad x(0) = \hat{x}, \quad \dot{x}(t) = f(x(t), u(t)), \forall t=0..T
\]

- trajectory optimization
- super-linear convergence
- real-time computation
- constraint satisfaction
- local minima (no global policy)
- difficulty with discontinuous dynamics
- no inclusion of multi-modal sensing

- (global) policy optimization
- handles discontinuities
- multi-modal sensing inclusion
- no guaranteed convergence
- little use of model information
- difficult transfer to robots
- no constraint satisfaction

Artificial & Natural Movements
Toward RL solvers with mathematical guarantees

Policy/Value learning

\[
\pi(S) \quad \text{constraint enforcement layer}
\]

state trajectory prediction

value/advantage function \[ V(S) \]

Objective 1
Policy optimization with derivatives
1. policy learning with derivatives
2. value function learning with derivatives
3. adding constraints in RL

Objective 2
Differentiable reinforcement learning with mixed discrete-continuous modes
1. local optimization with differentiable simulators
2. globalization using MCTS

MPC problem

\[
\max_{s_0, \ldots, s_T, a_0, \ldots, a_{T-1}} \mathbb{E}\left[ \sum_{t=0}^{T-1} r(s_t, a_t) + V(s_T) | \phi \right]
\]

s.t.
\[
\begin{align*}
& s_0 = \hat{s} \\
& s_{t+1} \sim P(s_{t+1} | s_t, a_t) = \phi(s_t, a_t) \\
& h(s_t, a_t) \geq 0
\end{align*}
\]

\[ s_0^*, \ldots, s_T^*, a_0^*, \ldots, a_{T-1}^* \]

\[ V^*(s), \nabla V^*(s), \nabla \pi^*(s), \nabla \phi(s, a) \]

policy and value optimization with derivatives

Artificial & Natural Movements
Main messages

We should both **OPTIMISE** and **LEARN** the movements of a robot!

- **Trajectory optimization** is necessary
  - 10,000 variables in 10 ms
  - Accurate convergence, constraints satisfaction, generalization

- **Policy learning** is necessary
  - Globalization using a memory of motion
  - Toward super-linear reinforcement algorithms
ANITI: building hybrid intelligence

Learning

Optimization

Joint laboratories
DYNAMOGRAD

Artificial & Natural Movements