Learning Cost-Efficient Control Policies with XCSF

Generalization Capabilities and Further Improvement

Jeremie Decock

Supervisor : Olivier Sigaud

UPMC ISIR

September 12, 2013

Paper

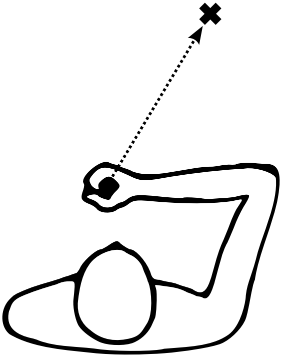
Learning Cost-Efficient Control Policies with XCSF: Generalization Capabilities and Further Improvement. Proceedings of the 13th annual conference on Genetic and evolutionary computation (GECCO'11), ACM Press, publisher. Pages 1235–1242.

- Didier Marin [ISIR]
- Jeremie Decock [ISIR]
- Lionel Rigoux [ISIR]
- Olivier Sigaud [ISIR]

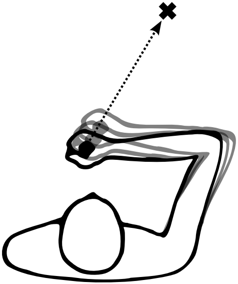
Motor control

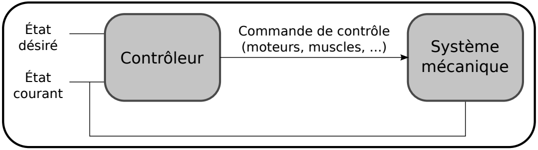
Aim

Let a *mechanical system* go from an initial *state* to a desired state



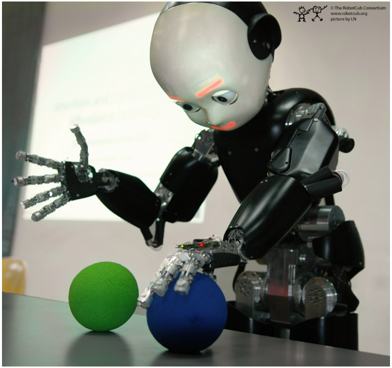
Control loop





Our goal

- Control a complex system
- Generate realist and efficient movements that reproduce human motor properties



Issues

Current techniques fail to fulfil these 2 needs :

Robotics

Complex systems but "unrealistic" and "inefficient" movements

Motor control

Simple systems only (in simulation)

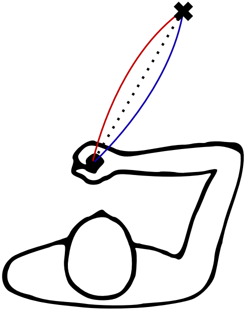
Plan

Overview

Realism and efficiency

We are looking for realistic and efficient movements

- To optimise : choose the "best" movement among those who solve the task



Quasi-Optimal Planning System (QOPS) [Rigoux and Guigon 11]

QOPS has good features, we would like to use it :

- Efficient : it found the best movement even in noisy environment
  - Minimise energetic cost
  - Maximise movement speed
- Realistic : it reproduces known features of human motor control

QOPS controller

QOPS mainly consider mechanical systems activated with muscles

Why muscles driven systems ?

- Robotics is moving towards this kind of actuators
- Interesting features : stiffness regulation
- Get the advantages of elastic muscles : kinetic energy conservation and restitution

QOPS controller

- Use *Pontryagin's minimum principle* (a *calculus of variations* methods) to find the best command $\mathbf{u}^*$ that let the known the current state $\boldsymbol{\xi}$ be closer to the desired state $\boldsymbol{\xi}^*$
- State = joint position and angular velocity ($\boldsymbol{\xi}$)
- Command = muscular activations ($\mathbf{u}$)

$\boldsymbol{\xi}$

$\boldsymbol{\xi}^*$

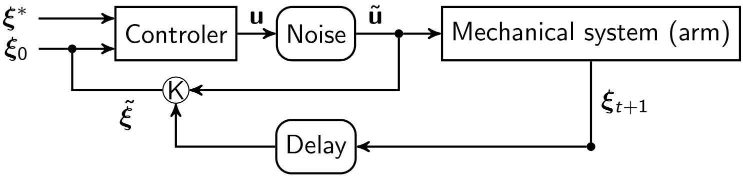
$$\mathbf{u}^* = \arg\min_{\mathbf{u},t_f} \underbrace{\epsilon \sum_{t=0}^{t_f} \Gamma(t_f)\mathbf{u}^2 \; \Delta_t}_{\text{energy cost}} - \underbrace{\rho\, \Gamma(t_f)}_{\text{reward}}$$

$\mathbf{u}^*$

QOPS

QOPS controller

- Deterministic controller
- Noisy environment
- Movements are adjusted (computed) for each time step



QOPS drawback

QOPS make efficient and realistic movements but it's computationally very expensive due to the variational calculus process.

QOPS compute the whole trajectory to reach the desired state considering a deterministic environment. But state and command are noisy so we have compute a new trajectory for each time steps to fit the actual state.

Improved solution

Improved solution

Main idea

Build a fast controller using Machine Learning (ML) and QOPS planning system

The ML system is supposed to :

- learn control policies generated by QOPS that is to say the function $QOPS(\xi_t, \xi^*) = \mathbf{u}_t^*$
- generalize over the whole reachable space based on learning from only a few planned movements
- quickly bring the control vector $\mathbf{u}_t^*$ knowing ξ_t and ξ^*

XCSF Learning Classifier System [Butz 08]

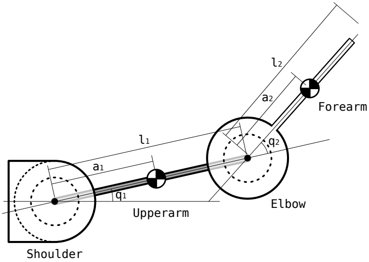
We have selected the eXtended Classifier System for Function (XCSF) to learn control policies

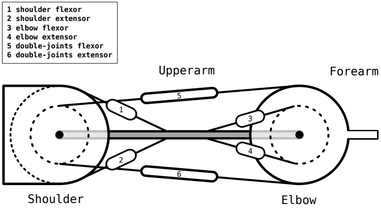
- a Learning Classifier System dedicated to function approximation
- general purpose function approximation tool based on regression mechanisms
- excellent regression capabilities

Experiments and results

Mechanical system modelization

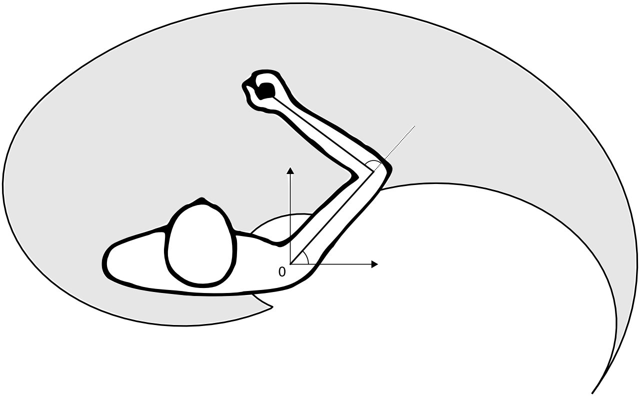
A model from [Li 2008] and [Rigoux et Guigon 11]



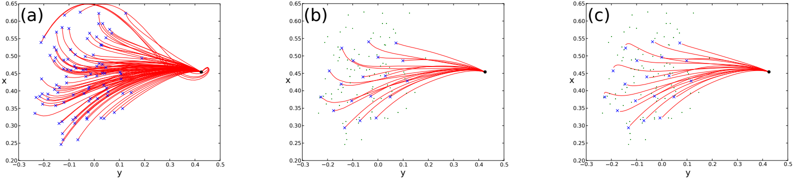


- State : $\xi = \begin{pmatrix} \dot{\mathbf{q}} & \mathbf{q} \end{pmatrix}^T = \begin{pmatrix} \dot{q}_1 & \dot{q}_2 & q_1 & q_2 \end{pmatrix}^T$
- Command : $\mathbf{u} = \begin{pmatrix} u_1 & u_2 & u_3 & u_4 & u_5 & u_6 \end{pmatrix}^T$

Task space

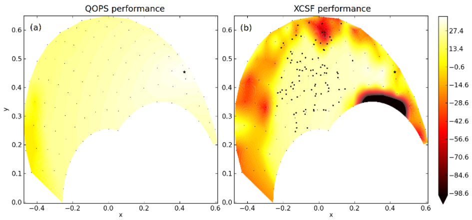


Results



- Trajectories obtained with QOPS for the learning targets(a), testing targets (b) and of the XCSF-based policy for the testing targets (c)
- The starting position is represented by a dot
- The targets are represented by a cross
- In (b) and (c), the dots represents the learning targets

Results



- big dots = learning target positions
- star = starting position

- Performance of the QOPS (a) and the XCSF policy (b) given the target position, obtained by interpolating the performances for the testing targets (smalldots)
- The performance is computed according to this equation :

$$\hat{C}\left(\mathbf{u}_{\{0..t_f\}}, \xi_t\right) = \epsilon \sum \mathbf{u}_t^2 - \rho\, g(\xi_t) \quad (1)$$

Results

Videos

Computation cost

The average running time to get one trajectory :

- QOPS $\approx$ 10 min
- XCSF $\approx$ 2 sec

(Intel Core 2 Duo E8400 @ 3 GHz with 4 GB RAM)

Thank you

Questions ?

Appendix

Parameters

m_i	mass of segment i (kg)
l_i	length of segment i (m)
s_i	inertia of segment i ($kg.m^2$)
d_i	distance between the center of segment i and its center of mass (m)
κ	Heaviside filter parameter
A	moment arm matrix
T	muscular tension
M	inertia matrix
J	Jacobian matrix
C	Coriolis force
τ	segments torque (N.m)
B	damping
u	raw muscular activation (action)
σ_u^2	multiplicative muscular noise
$\tilde{u}$	filtered noisy muscular activation
q^*	target articular position (rad)
q	current articular position (rad)
$\dot{q}$	current articular speed ($rad.s^{-1}$)

Dynamics

$$\boldsymbol{\tau} = \mathbf{A}^T \; \mathbf{f}_{\mathbf{max}} \; \mathbf{u} \tag{2}$$

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q}) \; \ddot{\mathbf{q}} + \mathbf{n}(\mathbf{q}, \dot{\mathbf{q}}) \tag{3}$$

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}(\mathbf{q}) \; \boldsymbol{\tau} - \mathbf{M}^{-1}(\mathbf{q}) \; \mathbf{n}(\mathbf{q}, \dot{\mathbf{q}}) \tag{4}$$