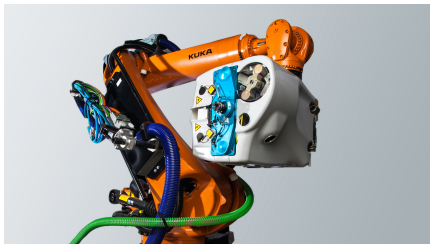# Génération de mouvements pour les robots humanoïdes avec ros-control

Journées Nationales de ROS (Robot Operating System)
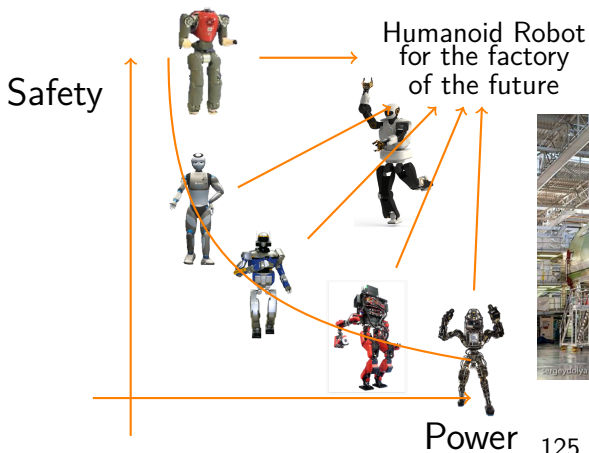
$19 - 20$ Juin 2018, Toulouse, France

O. Stasse, GEPETTO

Costly, heavy,
not user friendly

Safety

Humanoid Robot
for the factory
of the future

Power

125 millions of holes drilled by year

75 % are done *manually*

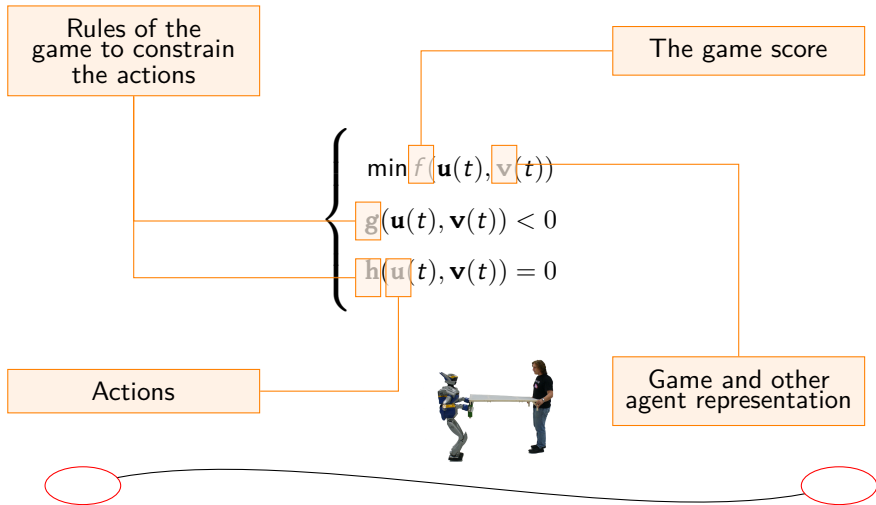**Problem 1** : Variability according to the customers

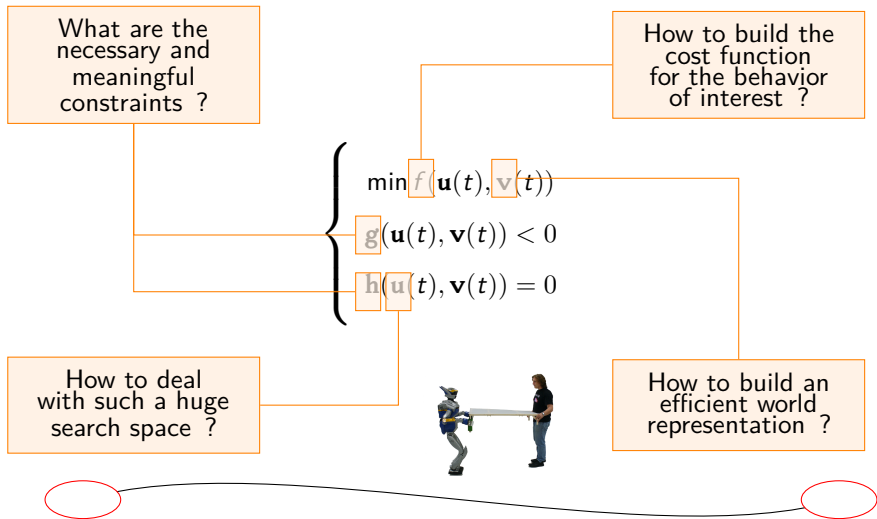**Problem 2** : Variability in the realization (humans)

Problem 3 : Variability in the delivery

Problem 4 : Collaboration with humans

Problem 5 : Certification

Rules of the game to constrain the actions

The game score

$$\min f(\mathbf{u}(t), \mathbf{v}(t))$$

$$\mathbf{g}(\mathbf{u}(t), \mathbf{v}(t)) < 0$$

$$\mathbf{h}(\mathbf{u}(t), \mathbf{v}(t)) = 0$$

Actions

Game and other agent representation

What are the necessary and meaningful constraints ?

How to build the cost function for the behavior of interest ?

$$\begin{cases} \min f(\mathbf{u}(t), \mathbf{v}(t)) \\ \mathbf{g}(\mathbf{u}(t), \mathbf{v}(t)) < 0 \\ \mathbf{h}(\mathbf{u}(t), \mathbf{v}(t)) = 0 \end{cases}$$

How to deal with such a huge search space ?

How to build an efficient world representation ?

Planning : Finding the new global optimum and the basins of attraction

Control : Ensure adaptation to local modification

$$
\begin{cases}
\min f(\mathbf{u}(t), \mathbf{v}(t)) \\
\mathbf{g}(\mathbf{u}(t), \mathbf{v}(t)) < 0 \\
\mathbf{h}(\mathbf{u}(t), \mathbf{v}(t)) = 0
\end{cases}
$$

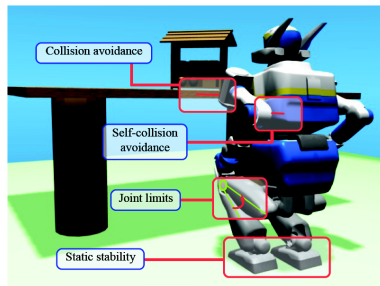[Saidi,IROS 2007]

$\mathbf{f}(t)$: The cost function

$\mathbf{u}(t)$: The control vector

$\mathbf{g}(t)$: The inequality constraints

$\mathbf{h}(t)$: The equality constraints

$\mathbf{v}(t)$: The environment representation

Collision avoidance

Self-collision avoidance

Joint limits

Static stability

$$\begin{cases} \min_{\mathbf{q}(\cdot),\mathbf{u}(\cdot)} \quad \int_0^T L(t,\mathbf{q}(t),\mathbf{u}(t))dt \\ \dot{\mathbf{q}}(t) = f(\mathbf{q}(t),\mathbf{u}(t)), \quad t \in [0,T] \\ \mathbf{q}(0) = \mathbf{q}_0, \ T > 0 \\ 0 \leq \mathbf{gh}(\mathbf{q}(t),\mathbf{u}(t)), \quad t \in [0,T] \end{cases}$$

[Saidi,IROS 2007]
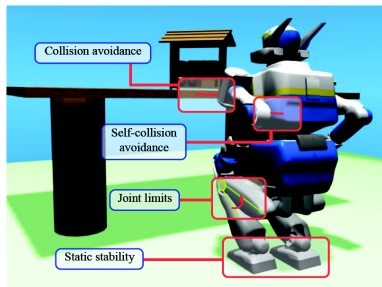
$\mathbf{f}$: The system dynamics

$\mathbf{u}$: The control vector

$\mathbf{q}$: The system state

$\mathbf{gh}$: The (in)equality constraints

$\mathbf{v}$: The environment representation

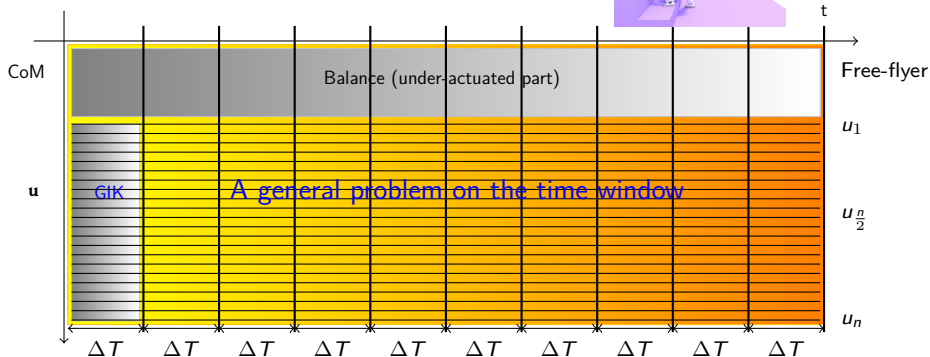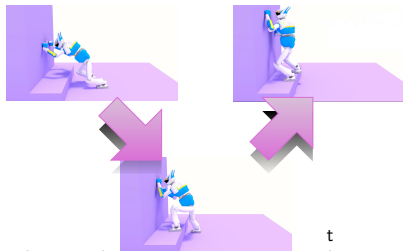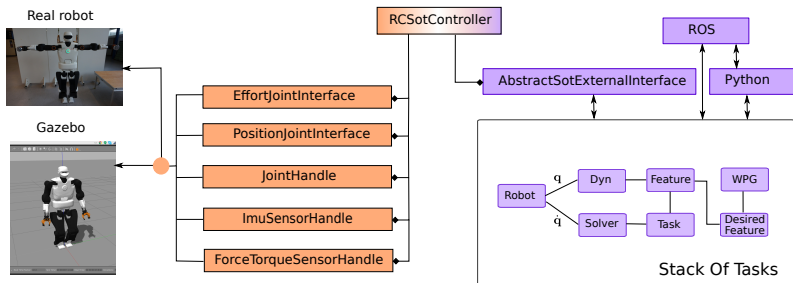$\mathbf{L}$: The cost

- Size of the problem
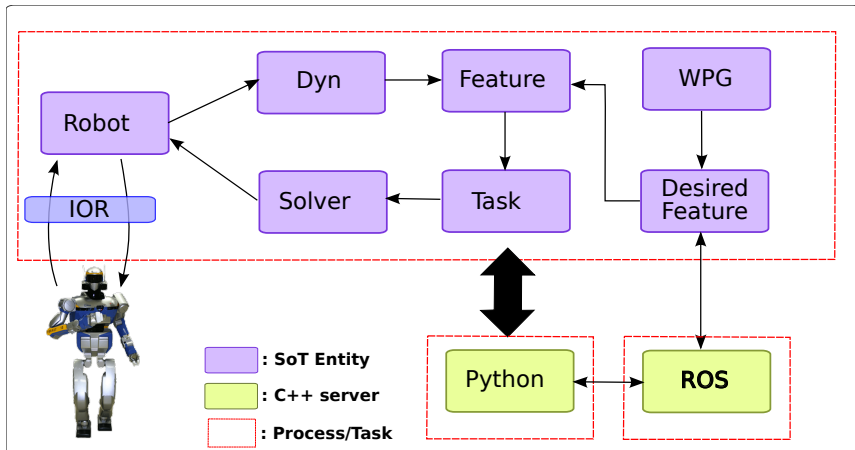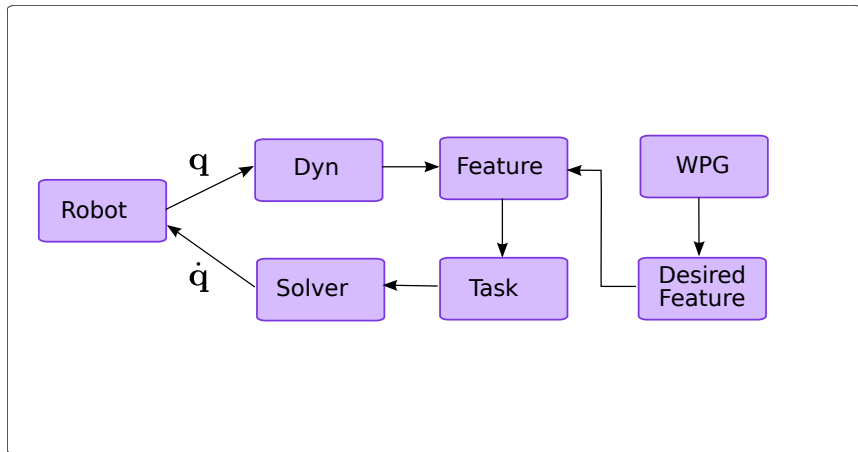
  $1.6 \times 200 \times 30 = 9600$ variables

- Non linear constraints

- Discrete nature due to contacts

https ://www.youtube.com/watch ?v=WbsQBPzQakc

$M(\mathbf{q})$

Robot — $\mathbf{q}$ → Dyn → Feature ← WPG

Robot ← $\dot{\mathbf{q}}$ — Solver ← Task ← Desired Feature

$M^*(\mathbf{q})$

$$T(\mathbf{q}, t) = \begin{pmatrix} \mathbf{t}(M^{*-1}(t)M(\mathbf{q})) \\ u_\theta(R^{*-1}(t)R(\mathbf{q})) \end{pmatrix}$$

$$J = \frac{\partial T}{\partial \mathbf{q}}$$

$M(\mathbf{q})$

$\mathbf{q}$

$\dot{\mathbf{q}}$

Robot — Dyn — Feature — WPG

Solver — Task — Desired Feature

$M^*(\mathbf{q})$

$$T(\mathbf{q}, t) = \left(\begin{array}{c} \mathbf{t}(M^{*-1}(t)M(\mathbf{q})) \\ u_\theta(R^{*-1}(t)R(\mathbf{q})) \end{array} \right)$$

$$J = \frac{\partial T}{\partial \mathbf{q}}$$

$M(\mathbf{q})$

Robot	Dyn	Feature	WPG

$\mathbf{q}$

$\dot{\mathbf{q}}$

Solver	Task	Desired Feature

$M^*(\mathbf{q})$

$$\dot{T} = -\lambda T$$

$$\dot{T} = -\lambda T - \frac{\partial T}{\partial t}$$

$$T(\mathbf{q}, t) = \begin{pmatrix} \mathbf{t}(M^{*-1}(t)M(\mathbf{q})) \\ u_\theta(R^{*-1}(t)R(\mathbf{q})) \end{pmatrix}$$

$$J = \frac{\partial T}{\partial \mathbf{q}}$$

$M(\mathbf{q})$

$\mathbf{q}$

Robot — Dyn — Feature — WPG

$\dot{\mathbf{q}}$

Solver — Task — Desired Feature

$M^*(\mathbf{q})$

$$\dot{\mathbf{q}} \triangleq -J^+(\lambda T + \frac{\partial T}{\partial t})$$

$$\dot{T} = -\lambda T$$
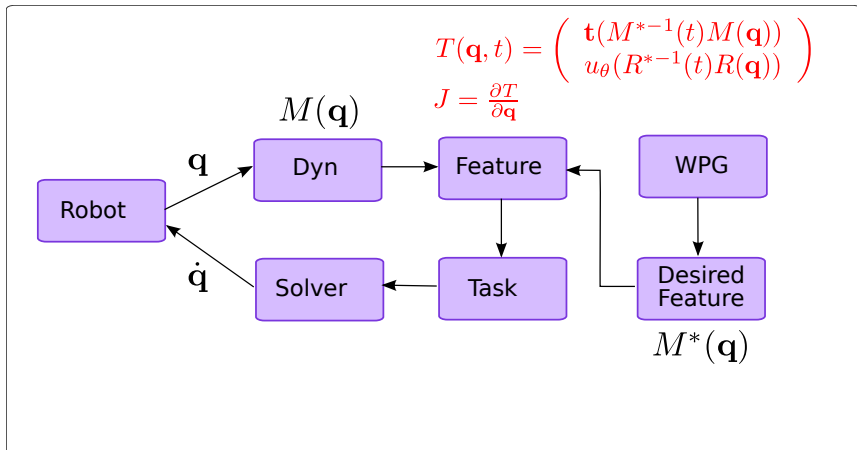
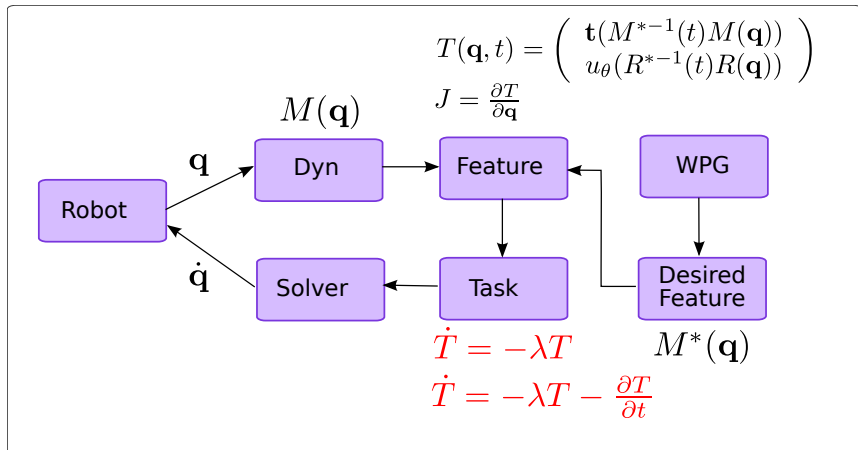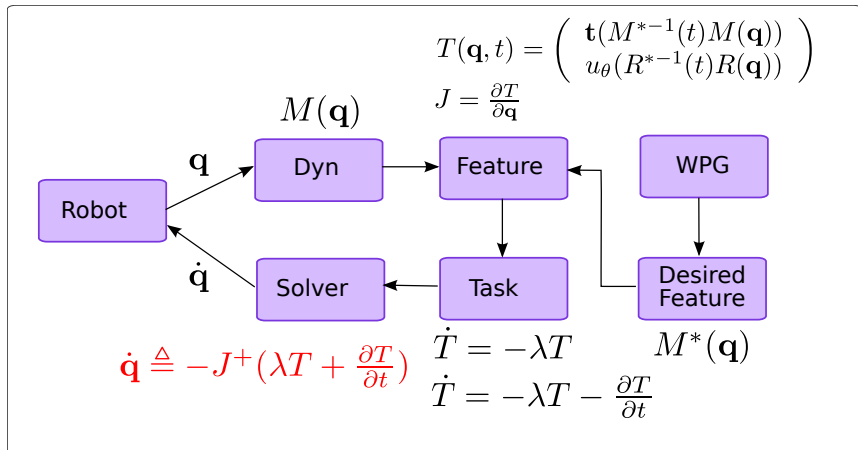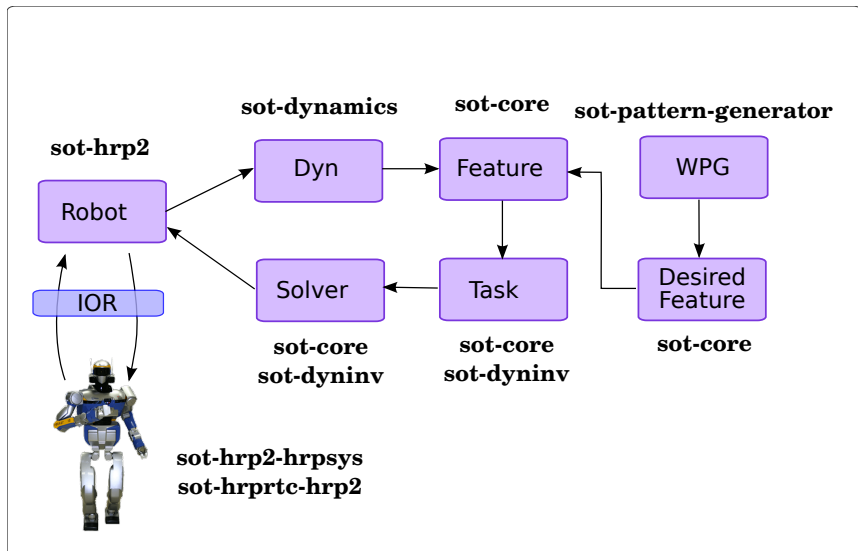$$\dot{T} = -\lambda T - \frac{\partial T}{\partial t}$$

Solvers

Inverse Dynamics

Weighted Pseudo Inverse

- Faster (!?) computation

- Easier to formulate

- Do not guarantee convergence

- Difficulty to tune the weights

- Do not handle properly inequalities

Hierarchical Quadratic Program

- Slower (!?) computation time

- Warranty on priority

- Handle easily inequalities

- Difficult to formulate (here hidden in the solver)

- Known problems with cycles and singularities management

Conclusions

Pros
- Generic to put instantaneous controller together
- Allow code reusability
- Real-time performance
- Adapted to complex applications
- Binary packages support
- Eigen support

Cons
- Research code
- The learning curve seems to be yet steep