Feedback on using Open Simulators for humanoid robotics

Journée technique simulation

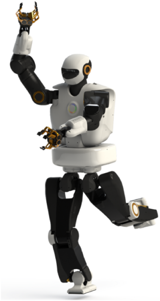
18 Octobre 2017, LAAS, Toulouse, France

O. Stasse, Gepetto,
LAAS-CNRS

Table of Contents

What is a simulator ?

[Ivaldi,ICHR,2014]



System simulators

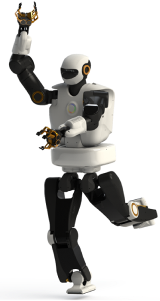
- Simulate sensors, actuators, environment
- Smoothly going from simulation to real robot
- Accurate representation of reality
- Stability

Control simulator

- Real-time
- Catch the robot main dynamics
- To be integrate inside the control loop

What is a simulator ?

[Ivaldi,ICHR,2014]



System simulators

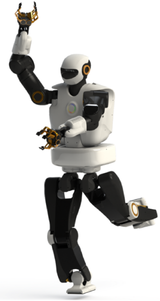
- Gazebo
- Stage
- Morse
- OpenHRP

Control simulator

- MuJoCo
- RBDL
- Pinocchio
- Robotran

What is a simulator ?

[Ivaldi,ICHR,2014]



System simulators

- Gazebo
- Stage
- Morse
- OpenHRP

Dynamics Engine: ODE

Control simulator

- MuJoCo
- RBDL
- Pinocchio
- Robotran

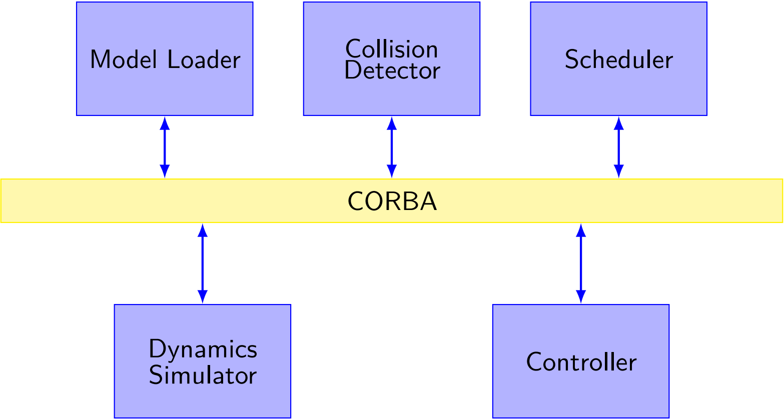
Problem 1: Reality

- Impact
- OpenHRP 2.x: 800 N
- Real robot: 1300 N
- Giving up speed
- Compliant material
- Dynamics
- Vision

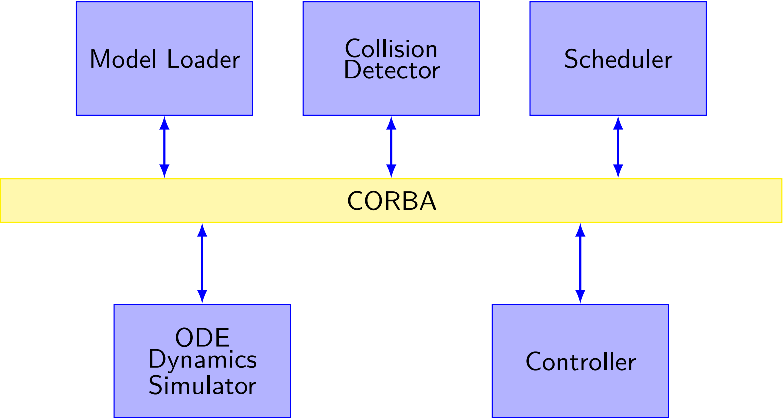
Problem 2: Software flexibility

- From simulation to real robot
- Being able to change parts of the simulator
- Strong middleware
- Sensor simulation

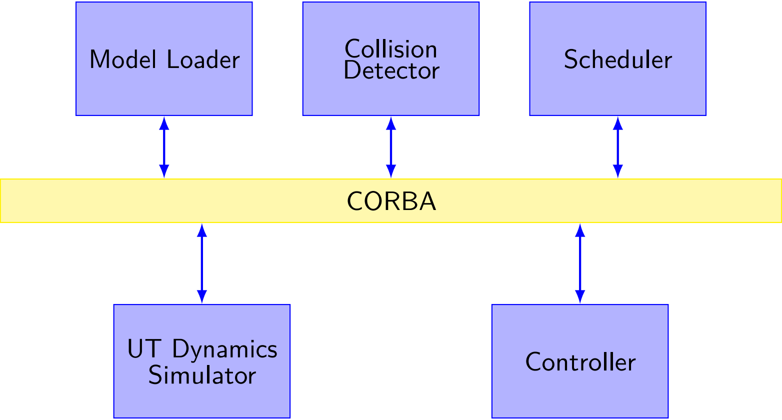
OpenHRP architecture



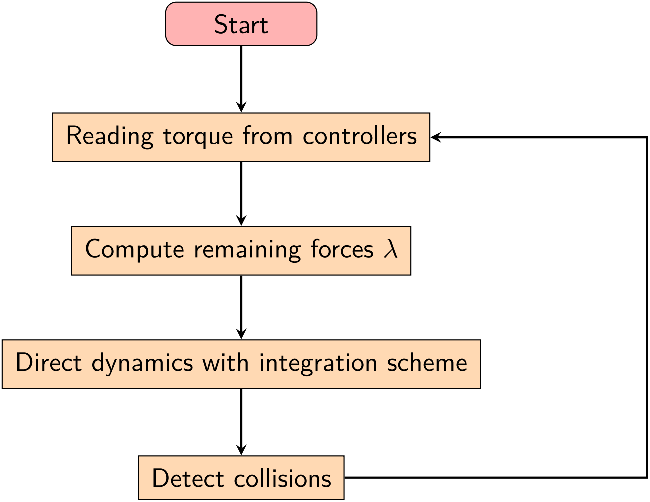
OpenHRP architecture



OpenHRP architecture



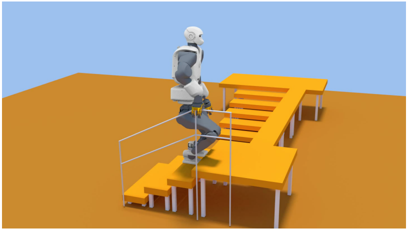
Global sequence - scheduler



OpenHRP AIST Dynamics Simulator

- Uses a mixture of *generalized* and *maximal* coordinates formulation
- The *generalized* coordinates are used to compute the robot state.
- This assumes a perfect rigid body dynamics (true for a high PID control)
- The *maximal* coordinates are used to compute the rest of the world state.
- Specific compliant joint are introduced in the direct dynamics of the robot.

ODE: The problem

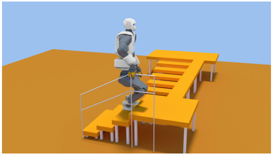


i-th body	
speed	$v_i \in \mathbb{R}^{dim(i)}$
force	$F_i \in \mathbb{R}^{dim(i)}$
acceleration	$\dot{v}_i$
then	$M_i \dot{v}_i = F_i$
	$M \dot{v} = F$

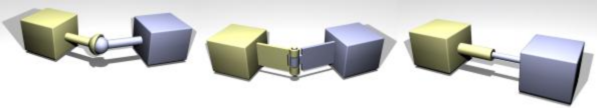
ODE: The problem

i-th constraint

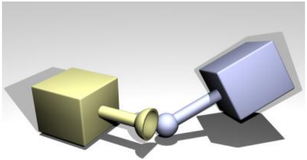
speed	$j_{i1}\dot{v}_1 + \cdots + j_{ik}\dot{v}_k + \cdots + j_{in}\dot{v}_n + c_i = 0$
all together	$J\dot{v} + c = 0$ (1)
	$M\dot{v} = F^c + F^{ext}$
with	$F^c = J^T\lambda$
then	$M\dot{v} = J^T\lambda + F^{ext}$
then	$\dot{v} = M^{-1}J^T\lambda + M^{-1}F^{ext}$
then with (1)	$J(M^{-1}J^T\lambda + M^{-1}F^{ext}) + c = 0$
then	$A\lambda = b$
with	$A = JM^{-1}J^T$ et $b = -(JM^{-1}F^{ext} + c)$



ODE: The problem

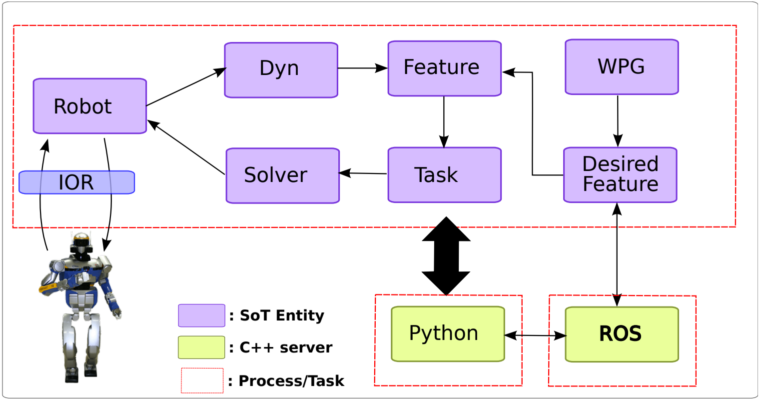


Joints when everythings goes well



Error Reduction Parameter

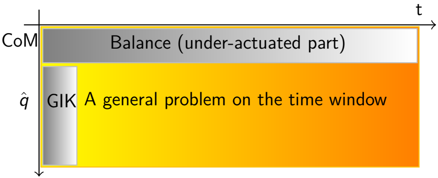
Our software control architecture



OpenHRP: controller component - ROS/Gazebo: roscontrol

Motion generation: the problem

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

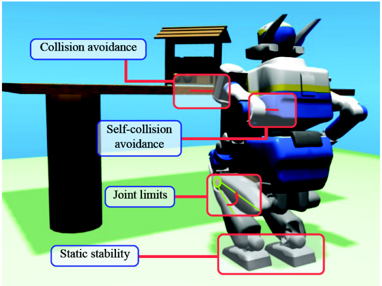


$\mathbf{f}(t)$: The cost function

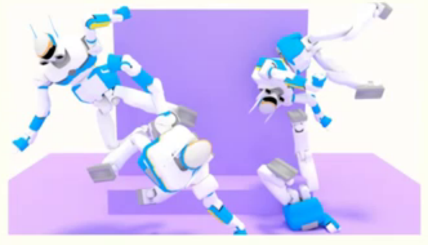
$\mathbf{u}(t)$: The control vector

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

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



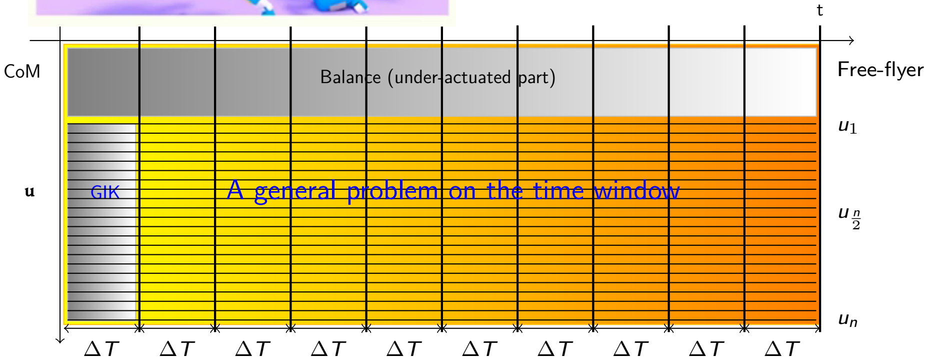
Motion generation



- Size of the problem

$$1.6 \times 200 \times 30 = 9600 \text{ variables}$$

- Non linear constraints
- Discrete nature due to contacts

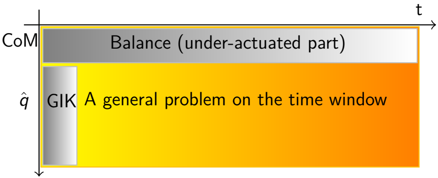


MuJoCo [Koenemann, IROS 2015]

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

Motion generation

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



- Planning and control solve the same problem
  - Planning is looking for a global feasible solution
  - Control is looking for on online sensor grounded local solution
- Planning is too long when simulating the control
- Control can fails
  - Local minima leading to an incomplete behavior
  - Mismatch between the control and the hardware
- Accessibility set [Majumdar, ICRA Best Paper Award 2013]

HPP [IJRR Submitted]

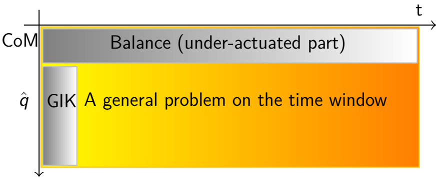
http://stevetonneau.fr/files/publications/ijrr16/video.mp4

Take away messages

- The *embodiment* (mechanical body, limits and controllers) defines the motion capabilities of the robot.
- We need to connect the accessibility set of our controllers to the planner.
- We need an efficient computation of the mechanical quantities
- We need to break down the problem complexity with small but representative problems
- We need to push higher the semantic level of our motion controllers

Motion generation: the constraints

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



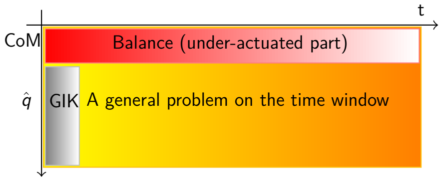
$$\begin{cases}
\mathbf{M}_1(q)\ddot{q} + \mathbf{N}_1(q,\dot{q})\dot{q} + \mathbf{G}_1(q) = \mathbf{T}_1(\mathbf{q})\mathbf{u} + \mathbf{C}_1^\top(q)\lambda & \text{Actuated dynamics of the robot} \\
\mathbf{M}_2(q)\ddot{q} + \mathbf{N}_2(q,\dot{q})\dot{q} + \mathbf{G}_2(q) = \mathbf{C}_2^\top(q)\lambda & \text{Underactuated dynamics of the robot} \\
f(\lambda) \in \mathcal{F} & \text{General balance criteria} \\
\mathbf{u}_{min} < \mathbf{u} < \mathbf{u}_{max} & \text{Torques limits} \\
\hat{q}_{min} < \hat{q} < \hat{q}_{max} & \text{Joints limits} \\
d(\mathcal{B}_i(\mathbf{q}), \mathcal{B}_j(\mathbf{q})) > \epsilon, \forall p(i,j) \in \mathcal{P} & \text{(self-)collisions} \\
\ddot{\mathbf{e}}_i = \dot{\mathbf{J}}_i(q)\dot{q} + \mathbf{J}_i(q)\ddot{q} & \text{Tasks}
\end{cases}$$

Motion generation: the constraints

Pattern generator

Focus on the underactuated part

Model predictive control



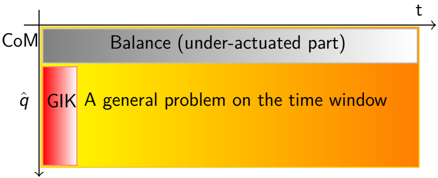
Simplifying the walking problem to control only the CoM reference

$$
\begin{cases}
\mathbf{M}_1(q)\ddot{q} + \mathbf{N}_1(q,\dot{q})\dot{q} + \mathbf{G}_1(q) = \mathbf{T}_1(\mathbf{q})\mathbf{u} + \mathbf{C}_1^\top(q)\lambda & \text{Actuated dynamics of the robot} \\
\mathbf{M}_2(q)\ddot{q} + \mathbf{N}_2(q,\dot{q})\dot{q} + \mathbf{G}_2(q) = \mathbf{C}_2^\top(q)\lambda & \text{Underactuated dynamics of the robot} \\
f(\lambda) \in \mathcal{F} & \text{General balance criteria} \\
\mathbf{u}_{min} < \mathbf{u} < \mathbf{u}_{max} & \text{Torques limits} \\
\hat{q}_{min} < \hat{q} < \hat{q}_{max} & \text{Joints limits} \\
d(\mathcal{B}_i(\mathbf{q}), \mathcal{B}_j(\mathbf{q})) > \epsilon, \forall p(i,j) \in \mathcal{P} & \text{(self-)collisions} \\
\ddot{\mathbf{e}}_i = \dot{\mathbf{J}}_i(q)\dot{q} + \mathbf{J}_i(q)\ddot{q} & \text{Tasks}
\end{cases}
$$

Motion generation: the constraints

Inverse dynamics

- Focus on the inertia matrix
- Forces
- Complete constraints

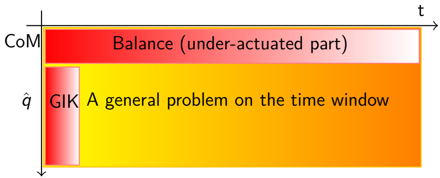


$$
\begin{cases}
\mathbf{M}_1(q)\ddot{q} + \mathbf{N}_1(q,\dot{q})\dot{q} + \mathbf{G}_1(q) = \mathbf{T}_1(\mathbf{q})\mathbf{u} + \mathbf{C}_1^\top(q)\lambda & \text{Actuated dynamics of the robot} \\
\mathbf{M}_2(q)\ddot{q} + \mathbf{N}_2(q,\dot{q})\dot{q} + \mathbf{G}_2(q) = \mathbf{C}_2^\top(q)\lambda & \text{Underactuated dynamics of the robot} \\
f(\lambda) \in \mathcal{F} & \text{General balance criteria} \\
\mathbf{u}_{min} < \mathbf{u} < \mathbf{u}_{max} & \text{Torques limits} \\
\hat{q}_{min} < \hat{q} < \hat{q}_{max} & \text{Joints limits} \\
d(\mathcal{B}_i(\mathbf{q}), \mathcal{B}_j(\mathbf{q})) > \epsilon, \forall p(i,j) \in \mathcal{P} & \text{(self-)collisions} \\
\ddot{\mathbf{e}}_i = \dot{\mathbf{J}}_i(q)\dot{q} + \mathbf{J}_i(q)\ddot{q} & \text{Tasks}
\end{cases}
$$

Motion control: Instantaneous control

Inverse dynamics

- Focus on the inertia matrix
- Forces
- Complete constraints

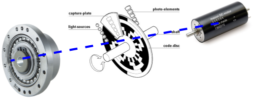


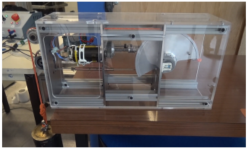
Ignore the actuator dynamics

$$\begin{cases} \mathbf{M}_1(q)\ddot{q} + \mathbf{N}_1(q,\dot{q})\dot{q} + \mathbf{G}_1(q) = \mathbf{T}_1(\mathbf{q})\mathbf{u} + \mathbf{C}_1^\top(q)\lambda & \text{Actuated dynamics of the robot} \\ \mathbf{M}_2(q)\ddot{q} + \mathbf{N}_2(q,\dot{q})\dot{q} + \mathbf{G}_2(q) = \mathbf{C}_2^\top(q)\lambda & \text{Underactuated dynamics of the robot} \\ f(\lambda) \in \mathcal{F} & \text{General balance criteria} \\ \mathbf{u}_{min} < \mathbf{u} < \mathbf{u}_{max} & \text{Torques limits} \\ \hat{q}_{min} < \hat{q} < \hat{q}_{max} & \text{Joints limits} \\ d(\mathcal{B}_i(\mathbf{q}), \mathcal{B}_j(\mathbf{q})) > \epsilon, \forall p(i,j) \in \mathcal{P} & \text{(self-)collisions} \\ \ddot{\mathbf{e}}_i = \dot{\mathbf{J}}_i(q)\dot{q} + \mathbf{J}_i(q)\ddot{q} & \text{Tasks} \end{cases}$$

Actuator dynamics

Classical actuator | SEA - Romeo - [IROS 2017] | Mc Kibben Muscle - [IROS 2016]





Actuator dynamics: necessity to control

$$
\begin{cases}
\phi(\mathbf{u}) = \tau_j & \text{Actuators dynamics} \\
\mathbf{M}_1(\mathbf{q})\ddot{q} + \mathbf{N}_1(\mathbf{q},\dot{q})\dot{q} + \mathbf{G}_1(\mathbf{q}) = \mathbf{T}_1(\mathbf{q})\tau_j + \mathbf{C}_1^\top(\mathbf{q})\mathbf{f} & \text{Actuated dynamics of the robot} \\
\mathbf{M}_2(\mathbf{q})\ddot{q} + \mathbf{N}_2(\mathbf{q},\dot{q})\dot{q} + \mathbf{G}_2(\mathbf{q}) = \mathbf{C}_2^\top(\mathbf{q})\mathbf{f} & \text{Underactuated dynamics of the robot} \\
f(\mathbf{f}) \in \mathcal{F} & \text{General balance criteria} \\
\mathbf{u}_{min} < \mathbf{u} < \mathbf{u}_{max} & \text{Torques limits} \\
\hat{q}_{min} < \hat{q} < \hat{q}_{max} & \text{Joints limits} \\
d(\mathcal{B}_i(\mathbf{q}), \mathcal{B}_j(\mathbf{q})) > \epsilon, \forall p(i,j) \in \mathcal{P} & \text{(self-)collisions} \\
\ddot{\mathbf{e}}_i = \dot{\mathbf{J}}_i(\mathbf{q})\dot{q} + \mathbf{J}_i(\mathbf{q})\ddot{q} & \text{Tasks}
\end{cases}
$$

Torque control with HRP-2

A. Del Prete, T. Flayols

Conclusion - Take away messages

- We need better Dynamics Engine
- We need better Contact model
- We need better Vision Rendering
- We need better Actuator simulator
- My best wish : Aist Dynamic Simulator + Blender Rendering + Gazebo