University of Liège - Faculty of engineering

Master thesis

Simulation of complex actuators

Author : Hubert Woszczyk
Promotor : Pr. Bernard Boigelot

Master thesis submitted for the degree of
Msc in Electrical Engineering

Academic year 2015-2016

Simulation of complex actuators

Hubert Woszczyk

Abstract

Lorem ipsum dolor...

Acknowledgements

I would like to express my sincere thanks to all those who provided me the possibility to complete this thesis.

First of all, I thank the professor B. Boigelot who made this thesis possible and helped me on various occasions.

I also wish to thank Grégory Di Carlo and Guillaume Lempereur, my fellow students who also worked on the robot.

Contents

List of Figures

Chapter 1

Introduction

1.1 Context

This thesis sprung from the participation of a team of students to the the Robocup competition, a robotics competition where robots play football.

The ultimate goal of Robocup is to create a team of robots able to beat human champions in football by 2050, as illustrated by the following quote :

> By the middle of the 21st century, a team of fully autonomous humanoid robot soccer players shall win a soccer game, complying with the official rules of FIFA, against the winner of the most recent World Cup.

Our team will participle in the kidsize category of the humanoid part of the contest. This means we will have to build a humanoid robot, for the first time at the Montefiore Institute.

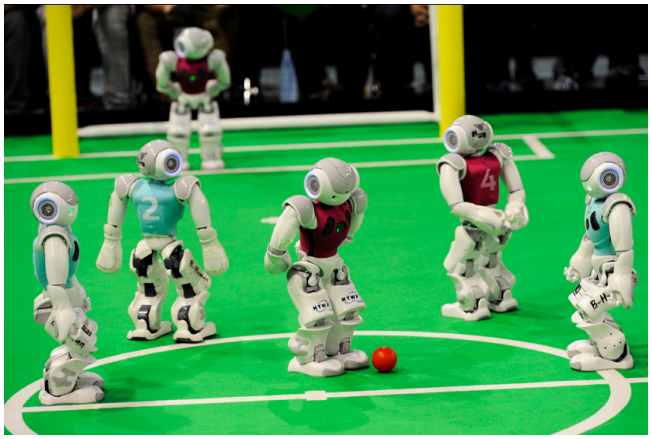

Figure 1.1: Two teams of Nao robots playing against each other in the 2014 edition of RoboCup *[Photo courtesy of RoboCup]*

We thus need a reliable simulation tool to :

1. Test different robot designs and choose the best one, without spending time building a robot in real-life.
2. At a later stage, be able to test control algorithms faster because the real robot is not needed.

1.2 Specifications

The goal of this thesis is to provide the team with a simulation tool and a model able of :

- realistically simulating the dynamics of the robot including springs and dampers
- receiving orders at approximately 300Hz, and we don't really care if the simulation is not in real time.
- the simulated robot receives exactly the same orders as the real robot would.
- visualization

As the robot is still being designed this thesis should give some insights on how to build it.

Chapter 2

Choosing the tools

In this chapter we discuss the choice of V-rep as the simulation tool for this project. We begin by explaining the basics of rigid body dynamics simulation, take a survey of some of the existing simulators and finally test some of them.

2.1 Principles of rigid body dynamics simulation

The section is heavily inspired by [BET14]. The basic idea behind the simulation of physics on a computer is to discretize time and apply the laws of newton to each object in the scene and integrate their acceleration during the timestep. When objects collide, collision.

A rigid body is an idealized solid object which will never change its shape, even under high forces.

The list of physics simulating engines is quite long, but the most popular ones are, in no particular order :

1. Bullet
2. ODE
3. DART
4. Simbody
5. PhysX
6. Havok

2.2 Available simulators

An integrated simulation tool is preferred over a bare-bones physics engine because :

- time would be lost on creating 3D visualization
- time would be lost on writing code to import model
- time would be lost on debugging

Engine	License	Coordinates	Origin	Editor	Solver type
Bullet	Free	Maximal	Games	Blender	Iterative
ODE	Free	Maximal	Simplified robot dynamics, games		Iterative
DART	Free	Generalized	Computer graphics, robot control		
Simbody	Free	Generalized	Biomechamics		
PhysX	Proprietary	Maximal	Games		
Havok	Proprietary	Maximal	Games		

Table 2.1: Features comparison[ETT]

and all that before the actual work could begin.

Blender[Bru04] :

- Uses the Bullet engine
- Scripting via Python, remote control possible through socket
- Very complete modelling tool, in a class of its own.
- Comment : Hard to use because of obscure simulation options and difficulties to correctly set inertias

Gazebo :

- Can use Bullet, Simbody, DART or ODE.
- Scripting via C++
- Uses SDF format for models.
- Comment : Hard to use because model must be in SDF format, which no CAD excepted 3dworks exports to. Furthermore, compiled language takes longer to test.

V-Rep:

- Can use Bullet, Newton or ODE.
- Internal scripting in LUA, provides remote API class.
- Can import 3D collada models.
- Comment : Best tool so far because model can be imported and the inertias are easy to control, simulation options as well.

Matlab:

- Analytical modelling
- Mathcode
- No visualization
- Comment : Not adapted because tedious modelling and no visualization and hard to handle friction and difficult to handle other objects.

Simulator	License	Physics engine(s)	Integrated editor	Modelling
Blender	Free	Bullet	Fully fledged	Internal
V-REP	Free (educational license)	Bullet, ODE, Newton, Vortex(10s limit)	Limited	Can import .COLLADA
Gazebo	Free	Bullet, ODE, Simbody, DART	Limited	SDF format
Webots	Proprietary	ODE	None	SDF format
Matlab	Proprietary	None	None	Mathematical

Table 2.2: Comparison of simulators

2.3 Choice

Out of Gazebo, V-Rep and Blender, V-Rep is chosen as the best tool because

- Gives the choice between 3 engines, something blender cannot do
- Makes it easier than Gazebo to create models, because Gazebo uses the URDF format
- Gives better access than blender to the physical options of the simulation (intertias, timestep of engine)
- It is multi-platform.

The physics engine used is Newton Dynamics because simple tests showed it to be the most stable with a high number of joints, with the exception of Vortex but it requires a license to run more than $10s$.

While Blender is not used as the primary simulation tool, it is used in the early phases of the modelling because it is what it does best. More on it in the next chapter.

Chapter 3

Modelling tools usage

This chapter covers the tools used in order to create a model of the robot, from the placement of the servos and joints to the incorporation of accelerometers.

3.1 Blender

Blender is used to :

- simplify the servos, hinges into simple convex shapes that behave better in a physical simulation.
- place all the elements of the robot at their position.
- place position markers for the joints, springs to be added in V-Rep.

An example of the state the model is in after this stage is present on fig. 3.1. When done with this, the model exported in the COLLADA format.

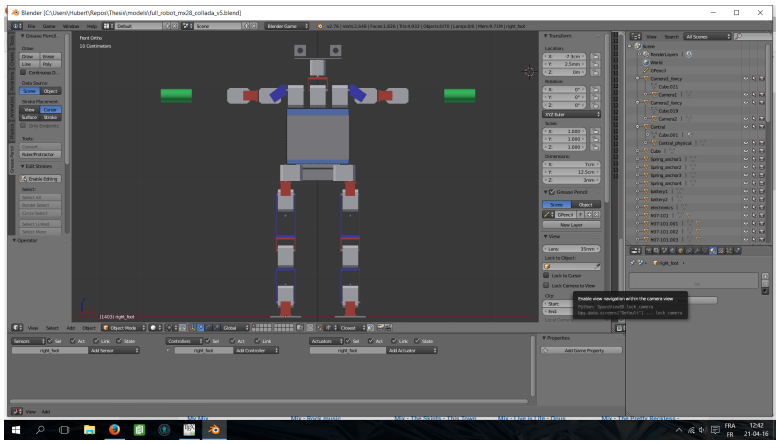

Figure 3.1: Front view of the robot's model at the end of the modelling in Blender. The grey boxes represent the servos, the red pieces are standard Dynamixel frames, the blue are non-standard, the green cylinders are the hands.

3.2 V-REP

The model is finalized by :

- defining the mass and inertia of each piece(compiled in table 5.1) and enabling them for dynamic simulation.
- adding joints between servos. For 2DOF joints, frame are used as intermediates.
- adding vision sensors to simulate cameras.
- adding scripts to simulate sensors (COG, accelerometers).
- adding springs on the legs through the use of prismatic and spheric joints.

3.2.1 Servos

Servos are simulated by joints.

3.2.2 Joints

Spherical joint : 3DOF angular.

Prismatic joint : 1DOF linear.

Revolute joint : 1DOF angular.

3.2.3 Sensors (accel, cog)

The COG is computed through a script inside V-Rep, attached to a piece of the model and made available through the remote interface[Rob16].

3.2.4 Springs

Springs are simulated by prismatic and spherical joints.

An example of what the model looks like at the end of this process is visible on fig. 3.2.

3.3 Simulation settings

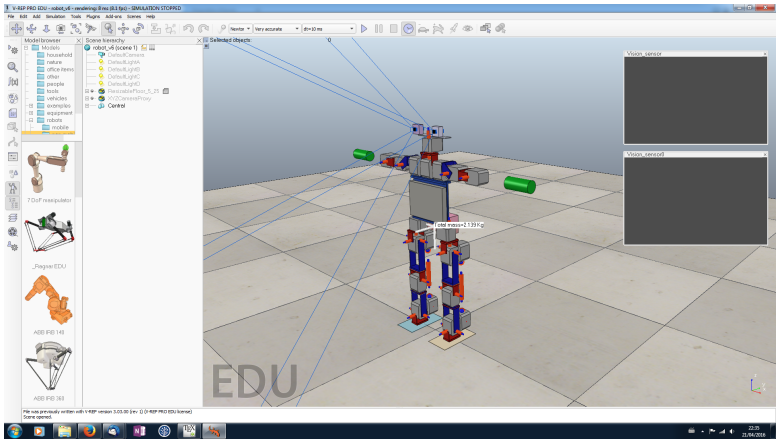

Figure 3.2: View of the robot's model at the end of the modelling in V-REP.

Chapter 4

Physical validation

In this chapter experiments with real servos will be conducted in order to, firstly, tune the parameters of the simulation (servo's characteristics) and, secondly, verify that the simulation correctly predicts the behaviour of a real-life configuration.

4.1 Experimental set-up

The set-up is explained on fig. 4.1. In later experiments a camera will be used to film the motion of the servos and compare it to the results of the simulation that is supposed to predict it.

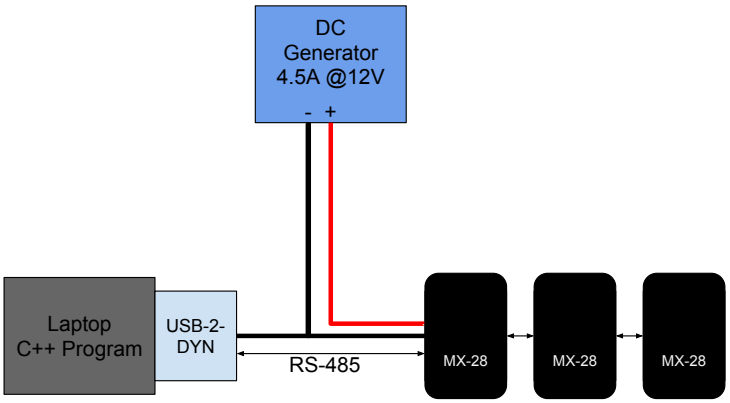

Figure 4.1: Experimental setup : The MX-28 servos are powered by a DC generator and controlled by a laptop equipped with a USB2DYNAMIXEL(USB-2-DYN) device. It converts an USB port into a serial port.

4.2 Experiment 1

The purpose of the first experiment is to test the torque : to that end, a frame is fixed onto a single servo and weighted. The setup is represented on fig. 4.2.

In our case, d was equal to $22.5cm$ and we could reach a weight w of $740g$ at $14.8V$. This equals to a torque of $1.64Ncm$. The complete results are listed in table 4.1.

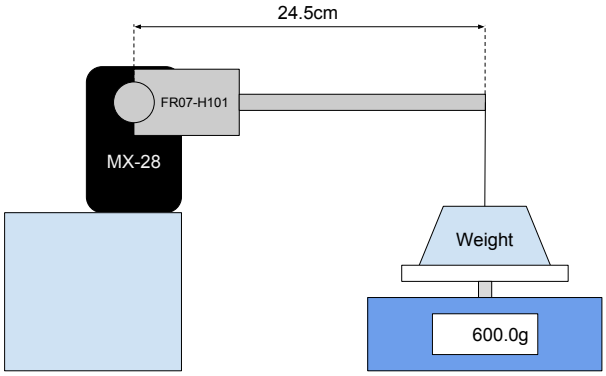

Figure 4.2: Experimental setup for torque testing. A weight w of is suspended at a distance d from the servo, resulting in a applied torque of $w \times g \times d$. The goal consists in finding the weight w for which the servo is unable to lift the arm.

	Stall torque @11.1V $[N.m]$	**Stall torque @12V** $[N.m]$	**Stall torque @14.8V** $[N.m]$
Theoretical	2.1	2.5	3.1
Experimental			1.6

Table 4.1: Experimental stall torques at different tested voltages. Theoretical values taken from [Dyn16]

.

4.3 Experiment 2

In this experiment we will test some simple dynamics. The setup is on fig. 4.3.

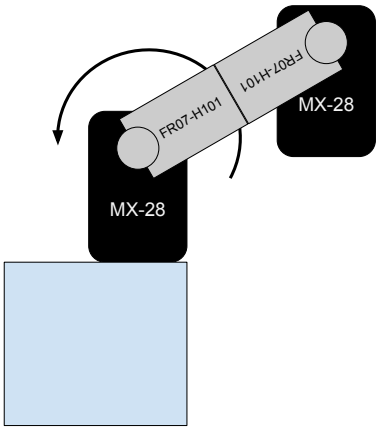

Figure 4.3: Experimental setup for dynamics testing. Two servos are connected together

4.4 Conclusion

Chapter 5

Simulation

In this section we will explain how to use the simulator and how it was used to influence the design of the robot. Finally some simulations will be shown.

5.1 Simulation setup

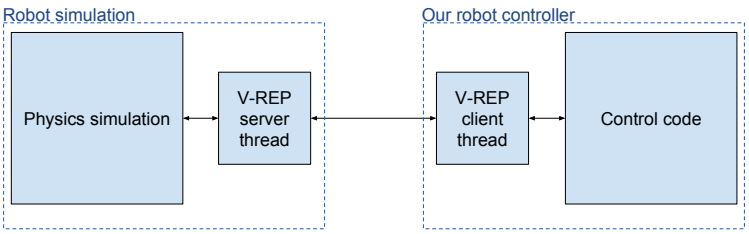

Figure 5.1: Schematic view of the simulation system. V-Rep handles solely the simulation of the robot and the control is established through a socket link with an external application which in our case is a MATLAB script.

The architecture of the system is explained on fig. 5.2. We choose to operate in the synchronous operating mode : before V-REP simulates a timestep it waits for a trigger, allowing us to precisely control the robot.

5.2 Applications

5.2.1 Static stability

5.2.2 Standing up routines

This section is heavily inspired by [SSB06]

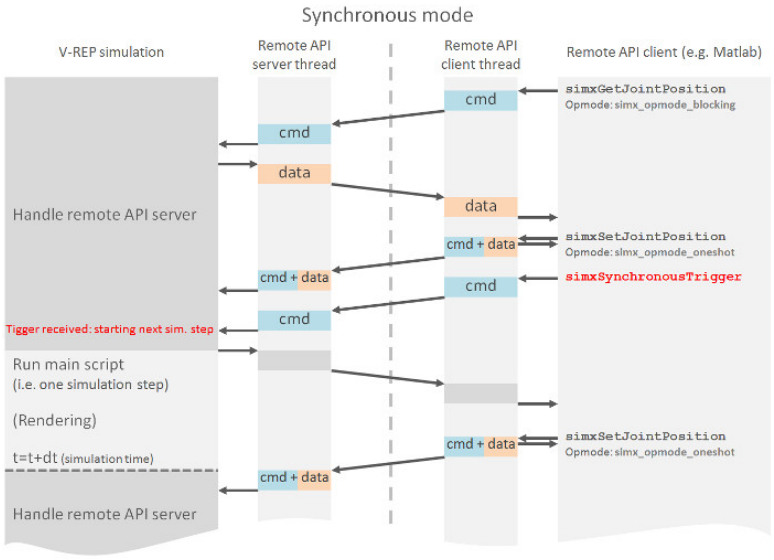

Figure 5.2: Explanation of the architecture of the simulation. The simulation runs on two threads : the simulation and the server thread. The server threads can receive orders from a client thread which is controlled by a custom application of our own.

Module	**Weight** $[g]$	**Density** $[kg/m^3]$	**Dimensions** $[mm \times mm \times mm]$
Odroid C-2	40		85.0 x 56.0 x 10.0
Li-Po battery	188	2304	103.0 x 33.0 x 24.0
Mx-28R	72	1150	35.6 x 50.6 x 35.5
LI-USB30-M021C	22	2200	26.0 x 26.0 x 14.7
Frame Fr-07		1200	
Frame Fr-101-H3	7	1200	

Table 5.1: Weights and dimensions of the pieces of the robot. The density is useful for the automatic computation of the weight and inertia of the pieces in V-REP.

5.2.3 Walking

5.3 Influence on robot's design

The simulator helped shape the robot through simulations that unveiled serious design problems (inability to stand after a fall, inability to walk).

The final dimensions of the robot respect the rules of the contest:

- Height : 61.3cm
- Height of COM : 34cm
- Height of legs : cm
- Height max is $< 1.5 \times 61.3$.
- Foot area is cm^2.

Chapter 6

Conclusion

This is the conclusion to my work.

Bibliography

[BET14] Jan Bender, Kenny Erleben, and Jeff Trinkle. Interactive simulation of rigid body dynamics in computer graphics. In *Computer Graphics Forum*, volume 33, pages 246–270. Wiley Online Library, 2014.

[Bru04] Herman Bruyninckx. Blender for robotics and robotics for blender. *Dept. of Mechanical Engineering, KU Leuven, Belgium*, 2004.

[Dyn16] Dynamixel. Robotis e-manual v1.27.00, 2016. [Online; accessed 18-April-2016].

[ETT] Tom Erez, Yuval Tassa, and Emanuel Todorov. Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx.

[Rob16] Coppelia Robotics. V-rep user manual 3.3.0, 2016. [Online; accessed 18-April-2016].

[SSB06] Jörg Stückler, Johannes Schwenk, and Sven Behnke. Getting back on two feet: Reliable standing-up routines for a humanoid robot. In *IAS*, pages 676–685, 2006.