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

List of Tables

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.

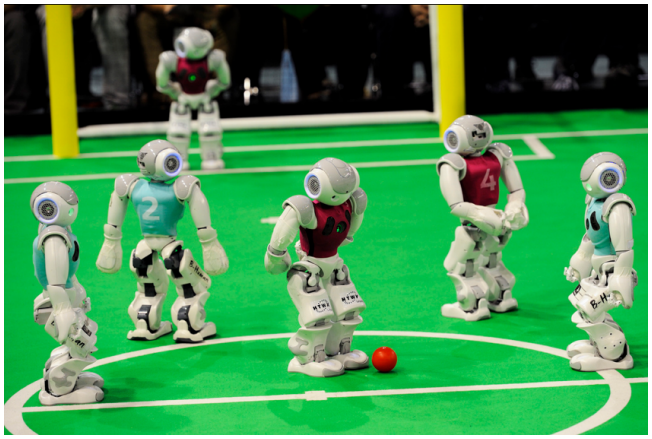

Figure 1.1: Two teams of Nao robots playing against each other in the 2014 edition of RoboCup *[Photo courtesy of RoboCup]*

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 Goals of the project

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.

1.3 Structure of the report

The report will begin with an overview of the available simulation tools and a choice for this project.

The next chapter will detail the chosen tools. The third chapter will be about tuning and verifying the accurateness of the simulation.

The fourth chapter will show some simulation examples and explain how they influence the design of the robot.

The last chapter will be the conclusion where the work will be summed up and future work prospects laid out.

Chapter 2

Principles of simulation

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.

Chapter 3

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.

3.1 Available simulators

3.1.1 Barebone physics engines

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

1. **Bullet :** The most popular open source physics engine as of now, used mainly for games(Rockstar Studios use it for their Grand Theft Auto games series) but also by a lot of tools (NASA tensegrity robotics toolkit, Blender, V-Rep). It supports rigid body and soft body simulation with collision detection and supports both rigid body and soft body constraints. Developed by Ewin Coumans, one of the creators of Havok.

2. **ODE :** Another popular open souce physics engine which is a little older than Bullet but still used by many roboticists for their simulations (V-Rep, Webots, Gazebo). This engine has been used by many commercial games over the years such Call of Juarez series or S.T.A.L.K.E.R.

3. **Newton :** Another open source engine, not quite as popular as Bullet and ODE but nevertheless used for commercial games (Amnesia, Soma). What makes it standout in the crowd is its deterministic constraint solver, as opposed to iterative solvers used by Bullet and ODE.

4. **PhysX :** A proprietary engine used primarily for games because its main focus is performance and not necessarily physical accurateness. Currently owned by Nvidia.

5. **Havok :** Another proprietary engine and a concurrent of PhysX, with virtually no difference between the two. Owned by Microsoft.

Engine	License	Coordinates	Origin	Solver type
Bullet	Free	Maximal	Games	Iterative
ODE	Free	Maximal	Simplified robot dynamics, games	Iterative
PhysX	Proprietary	Maximal	Games	
Havok	Proprietary	Maximal	Games	

Table 3.1: Features comparison[ETT15]

3.1.2 Simulators

In this section we will speak of software that provide a higher level interface to the physics engines we presented earlier. That interface usually adds a visualization and some other useful features.

1. **Blender**[Bru04] is 3D modelling software suite and as such has integrated the Bullet engine to help make more realistic animations. It features the ability to make Python scripts that use that engine to make games or physics simulations. It is cross platform.

2. **Gazebo** is the official simulator for the DARPA Atlas challenge. It features multiple physics engines (Bullet ,Simbody, Dart and ODE), allows custom plugins and uses the SDF format for its models. It is open source and runs natively on Linux systems but needs to be compiled in order to run on Windows or OSX.

3. **V-Rep :** is another simulator that lets you choose the physics engine(Bullet, ODE, Newton, Vortex) and it also allows custom plugins in the form of LUA scripts. It is cross-platform and uses its own format for storing models but can import standard formats(COLLADA, 3ds, etc...). It is free to use for educational purpose.

4. **Webots :** has virtually the same features as V-Rep but is not free.

5. **Matlab :** Not a dedicated robotics simulator per se but can be used to model the robot analytically and to write simulation code for it.

A summary of the features of each simulator is present on table 3.2.

Simulator	License	Physics engine(s)	Integrated editor	Modelling
Blender	Free	Bullet	Fully fledged	Internal
V-REP	Free	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 3.2: Comparison of simulators

3.2 Tested software

In order to choose the most suited tool some of the presented tools were installed and tested. Here are the results :

- Blender is pleasant to use because the robot's model can be easily changed inside it and the Python scripting allows fast development. Support for a socket allows an external program to control the robot. The internals of the physics are obscured and some interesting object properties, such as inertias, are hard to reach. It is also hard to change the simulation parameters making it difficult to obtain stable results when using a higher number of objects and constraints. Furthermore, support for the game engine, the basis of a simulation project is uncertain [Ble15].
- Gazebo is attractive because it has the support of DARPA and handles multiple physics engines. The main drawback lies in the modelling of the robot to be simulated. It does feature an internal modelling tool but it is too limited to be usable. The difficulty lies in the fact that it uses an xml file to store the parameters of the robot and the only tool that can export models to that format is 3ds max, a commercial product.
- V-Rep also has multiple physics engines available and has a user-friendly interface. It also has an internal modelling tool but there is not much use for it since it allows the import of models in the COLLADA format. It also supports socket communication and even provides code for a client thread in the custom application. The options of the physics engines are also pretty accessible and lots of sensor types are natively supported by the simulator.

3.3 Choice

The first choice to be made is whether we go for a barebone physics engines or a simulator. The former has the advantage of being a highly customizable solution but a simulator provides much a physics engines does not :

- 3D visualization
- code handling models import
- and many other

So a simulator is preferred.

In the case of simulators we must eliminate Webots because it is not free for use. Bullet is nice for little game-like physics but its lack of access to the parameters of the simulation makes it really hard to use in practice. Gazebo suffers from the the use of the SDF format for its models. If a model exists, Gazebo is a good tool but when there is a no stably defined model it is handicap. So we are left with V-rep which is free, multi-platform, gives the choice of the physics engines as Gazebo does but also provides a better modelling workflow which allows us to modify the model of the robot. Another advantage is that this simulator is already used at the university. This choice is further confirmed by [IPPN14] which shows that V-Rep is the highest noted tools amongst roboticists.

Inside V-Rep, we chose 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 $10s$.

Although Blender was not chosen as the primary simulation tool for the project, it shall be used as a modelling tool for the robot as its interface is in another class than V-Rep.

Chapter 4

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.

4.1 Blender

Blender is a free and open source 3D creation software suite. It features modelling, animation, rendering tools and integrates the bullet physics engine.

- 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. 4.1. When done with this, the model exported in the COLLADA format.

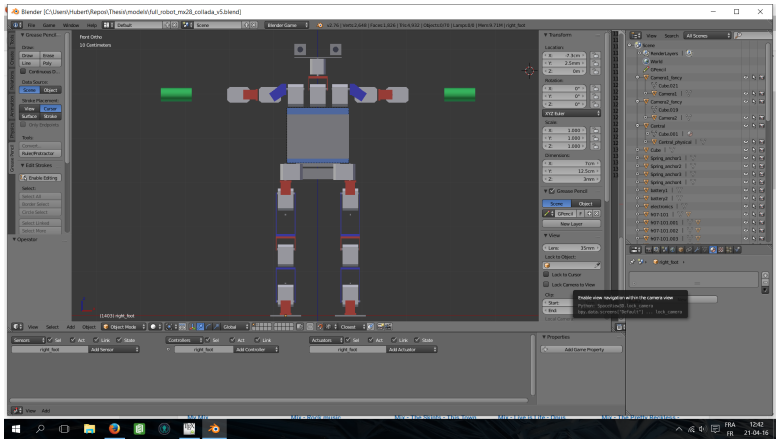

Figure 4.1: Front view of the robot's model at the end of the modelling in Blender. The grey boxes represent the servos, the red pieces are standard Dynamixel frames, the blue are non-standard, the green cylinders are the hands.

4.2 V-REP

The model is finalized by :

- defining the mass and inertia of each piece(compiled in table 6.1) 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.

4.2.1 Servos

Servos are simulated by joints.

4.2.2 Joints

Spherical joint : 3DOF angular.

Prismatic joint : 1DOF linear.

Revolute joint : 1DOF angular.

4.2.3 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].

4.2.4 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. 4.2.

4.3 Simulation settings

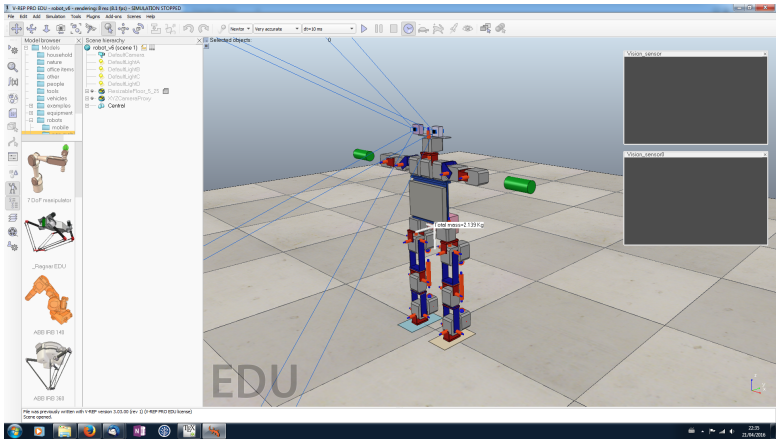

Figure 4.2: View of the robot's model at the end of the modelling in V-REP.

Chapter 5

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.

5.1 Experimental set-up

The set-up is explained on fig. 5.1. 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.

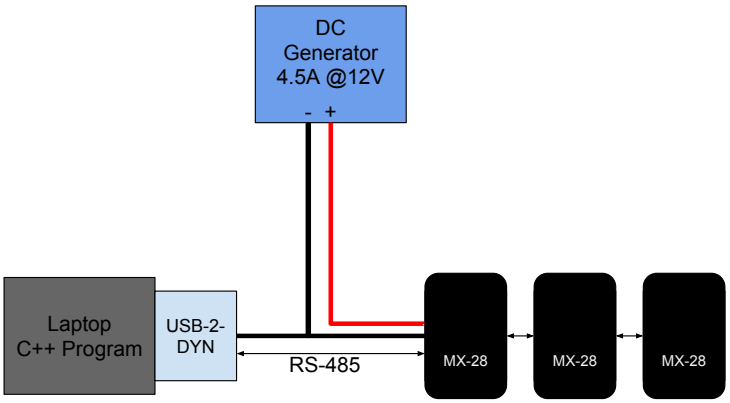

Figure 5.1: Experimental setup : The MX-28 servos are powered by a DC generator and controlled by a laptop equipped with a USB2DYNAMIXEL(USB-2-DYN) device. It converts an USB port into a serial port.

5.2 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. 5.2.

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 5.1.

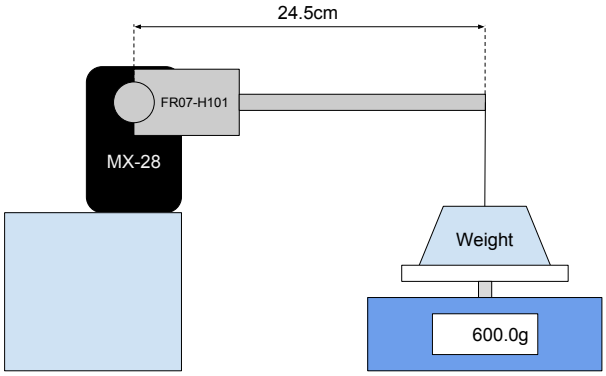

Figure 5.2: Experimental setup for torque testing. A weight w of is suspended at a distance d from the servo, resulting in a applied torque of $w \times g \times d$. The goal consists in finding the weight w for which the servo is unable to lift the arm.

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

.

5.3 Experiment 2

In this experiment we will test some simple dynamics. The setup is on fig. 5.3.

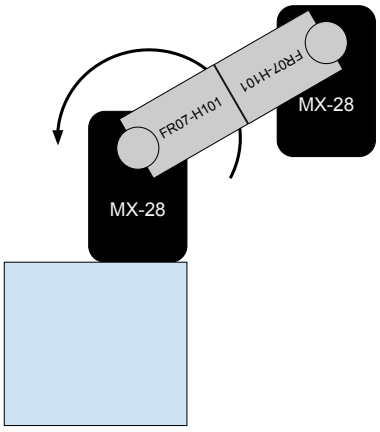

Figure 5.3: Experimental setup for dynamics testing. Two servos are connected together

5.4 Conclusion

Chapter 6

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.

6.1 Simulation setup

The basic idea of the simulation is presented on fig. 6.1.

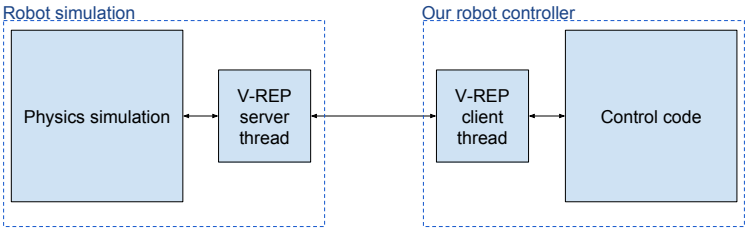

Figure 6.1: The idea is to let V-Rep simulate the robot and control with the actual code that runs on the robot.

The architecture of the system is explained on fig. 6.2. 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.

6.2 Applications

6.2.1 Static stability

The first application is simply to build a model of the robot and test if it is able to stand upright on its own.

The modelling begins in Blender where pieces are simplified/made convex and placed to create the structure of the robot. The model is then exported (in COLLADA) and imported into V-Rep.

In V-Rep the different elements of the robot are dynamically enabled and given mass, accordingly to the values listed in table 6.1. Then, joints (motor controlled with control loop

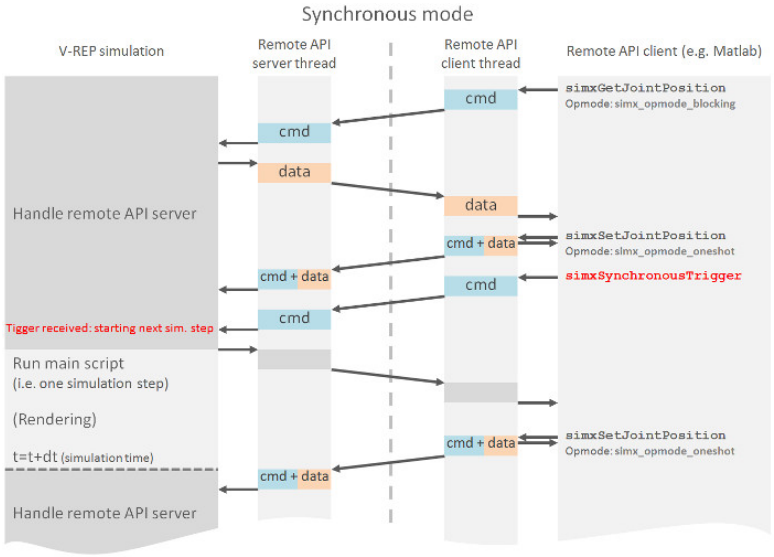

Figure 6.2: Typical interaction between the simulator and the control code. The simulation runs on two threads : the simulation and the server thread. The server threads can receive orders from a client thread which is controlled by a custom application of our own.

activated) are added to simulate the behaviour of the servos. Their maximal torque is set to 1.6, the maximum torque developed my Mx-28 servos as shown by our earlier experiments (table 5.1).

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 6.1: 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.

The springs on the leg are simulated two spherical joints and one prismatic joint set to spring-damper mode.

The servos of the robot are simply ordered to hold their initial angle and the simulation determines that the robot can indeed stand upright without any active stabilization.

6.2.2 Standing up routines

This section is heavily inspired by [SSB06]

6.2.3 Walking

6.3 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 first design is visible on fig. 6.3. It was plagued by stability problems, overcomplicated arms and simulation difficulties.

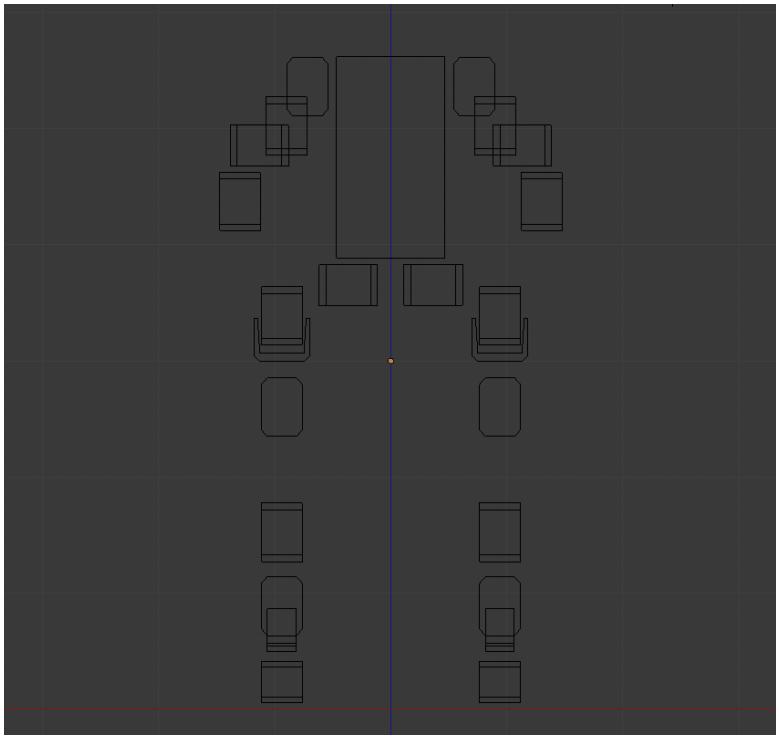

Figure 6.3: First robot design. Arms use 4 servos each, making it quite heavy.

The final design, visible on fig. 6.4 has better stability, wider movement possibilities and can stand up and walk more easily.

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.

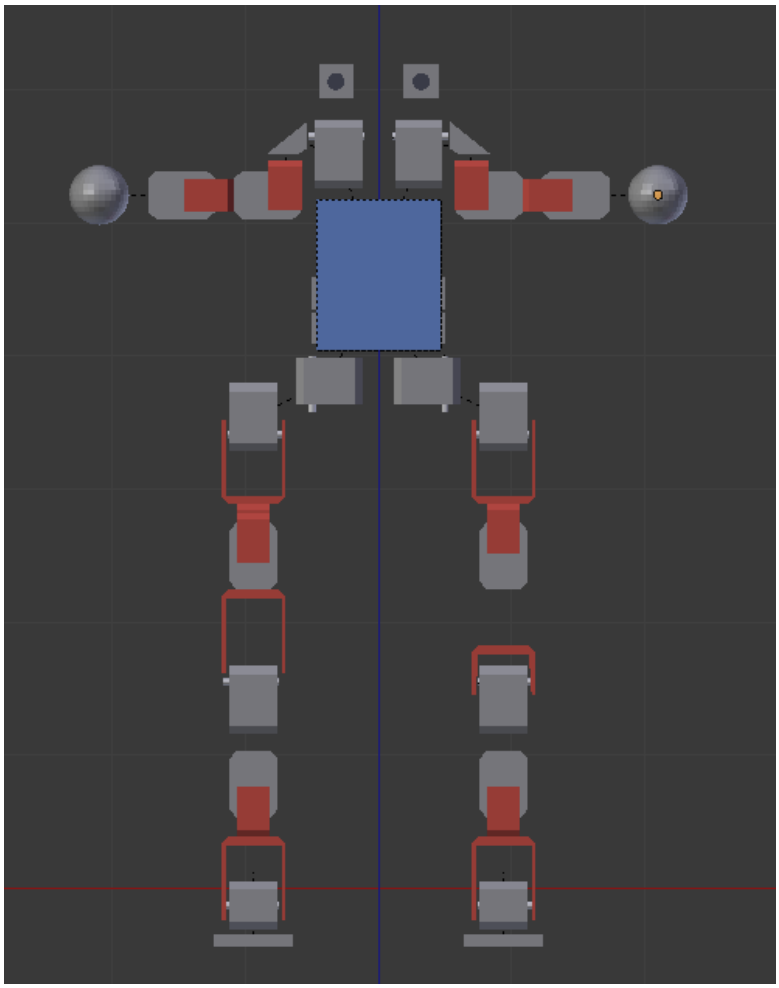

Figure 6.4: Final robot design. Arms now use 3 servos. The feet and the hips use a different configuration to have wider movement possibilities and bring down the center of gravity.

Chapter 7

Conclusion

7.1 Conclusion

This is the conclusion to my work.

7.2 Future work

7.2.1 Modelling

As of now it is still uncertain if Blender shall continue to support the COLLADA format (as explained in [Ble15]). In the negative, another tool should be chosen to perform the modelling.

The springs also need some work, as of now they are just there as a proof of concept but their parameters will need to be tuned.

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.

[Ble15] Blender. 2.8 project developer kickoff meeting notes, 2015. [Online; accessed 01-May-2016].

[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].

[ETT15] Tom Erez, Yuval Tassa, and Emanuel Todorov. Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx. *International Conference on Robotics and Automation*, 2015.

[IPPN14] Serena Ivaldi, Jan Peters, Vincent Padois, and Francesco Nori. Tools for simulating humanoid robot dynamics: a survey based on user feedback. In *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*, pages 842–849. IEEE, 2014.

[Rob15] Robocup. Robocup soccer humanoid league rules and setup, 2015.

[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.

Appendix A

Rules

The robots that participate in the kidsize competition must respect the following characteristics[Rob15]:

1. $40cm \leq H_{top} \leq 90cm$.
2. Maximum allowed weight is $20kg$.
3. Each foot must fit into a rectangle of area $(2.2 \cdot H_{com})^2/32$.
4. Considering the rectangle enclosing the convex hull of the foot, the ratio between the longest side of the rectangle and the shortest one, shall not exceed 2.5.
5. The robot must fit into a cylinder of diameter $0.55 \cdot H_{top}$.
6. The sum of the lengths of the two arms and the width of the tor so at the shoulder must be less than $1.2 \cdot H_{top}$. The length of an arm is defined as the sum of the maximum length of any link that forms part of the arm. Both arms must be the same length.
7. The robot does not possess a configuration where it is extended longer than $1.5 \cdot H_{top}$.
8. The length of the legs H_{leg}, including the feet, satisfies $0.35 \cdot H_{top} \leq H_{leg} \leq 0.7 \cdot H_{top}$.
9. The height of the head H_{head}, including the neck, satisfies $0.05 \cdot H_{top} \leq H_{head} \leq 0.25 \cdot H_{top}$. H_{head} is defined as the vertical distance from the axis of the first arm joint at the shoulder to the top of the head.
10. The leg length is measured while the robot is standing up straight. The length is measured from the first rotating joint where its axis lies in the plane parallel to the standing ground to the tip of the foot.