

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

Contents	ii
List of Figures	iv
List of Tables	v
1 Introduction	1
1.1 Context	1
1.2 Goals of the project	2
1.3 Structure of the report	2
2 Principles of interactive rigid body simulation	3
2.1 Principles of rigid body dynamics simulation	3
3 Choosing the tools	4
3.1 Available tools	4
3.1.1 Barebone physics engines	4
3.1.2 Simulators	5
3.2 Tested software	6
3.2.1 Blender	6
3.2.2 Gazebo	6
3.2.3 V-Rep	7
3.3 Choice	7
4 Presentation of Blender and V-Rep	8
4.1 Blender	8
4.2 V-REP	9
4.2.1 Servos	9
4.2.2 Joints	9
4.2.3 Sensors (accel, cog)	9
4.2.4 Springs	9
4.3 Simulation settings	9
5 Physical validation	11
5.1 Experimental set-up	11
5.2 Experiment 1	11
5.3 Experiment 2	12
5.4 Conclusion	12
6 Simulation	13
6.1 Simulation setup	13

6.2	Applications	13
6.2.1	Static stability	13
6.2.2	Standing up routines	15
6.2.3	Walking	15
6.3	Influence on robot's design	15
7	Conclusion	18
7.1	Conclusion	18
7.2	Future work	18
7.2.1	Modelling	18
	Bibliography	19
A	Rules	20
B	Design guidelines	21

List of Figures

1.1	Two teams of Nao robots playing against each other	2
2.1	Simulation loop	3
4.1	Front view of the robot's model at the end of the modelling in Blender	8
4.2	View of the robot's model at the end of the modelling in V-REP . . .	10
5.1	Experimental setup	11
5.2	Experimental setup for torque testing	12
5.3	Experimental setup dynamics testing	12
6.1	Simulation principles	13
6.2	Simulation interaction	14
6.3	Initial robot design	15
6.4	Final robot design	16

List of Tables

3.1	Features comparison[ETT15]	5
3.2	Comparison of simulators	5
5.1	Results of experiment 1	12
6.1	Weights and dimensions of the pieces of the robot	15

Chapter 1

Introduction

1.1 Context

This thesis is the product of the participation of a team of students to a robotic contest named RoboCup.

This contest is quite vast and divided into several categories :

- RoboCup Industrial : a category for more industrially focused contests.
- RoboCup@Home : a league for domestic robots that support elder people and stuff.
- RoboCupJunior : more of an initiative that aims to foster robotics interest in children, it helps organize various events for younger minds.
- RoboCup Soccer : historically the first category, centered about humanoid robots playing football in small teams. This is the league we will compete in.

The Robocup Soccer category is further subdivided into :

- Adultsized, for the taller robots
- Teensized, for middle sized robots
- Kidsized, for the smaller robots.

The team will participate to Kidsized subcategory of the Robocup Soccer category. 2016's edition will take place in Leipzig, Germany.

Our participation means that for the first time at the Montefiore Institute, a humanoid robot will be built. We thus need a reliable simulation tool to :

1. Test different robot designs and choose the best one, faster than it would be done by building the designs in real life.
2. At a later stage, be able to test control algorithms faster because the real robot is not needed.

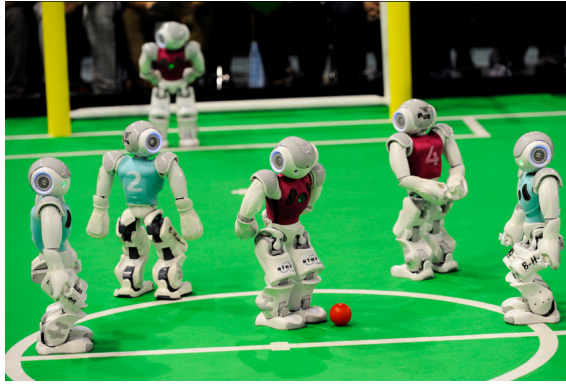


Figure 1.1: Two teams of Nao robots playing against each other in the 2014 edition of RoboCup [*Photo courtesy of RoboCup*]

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 (not necessarily in real-time)
- receiving exactly the same orders as the real robot would.
- visualizing the results of the simulation.

1.3 Structure of the report

This report begins with an overview of the basics of physics simulation on computers before moving to the chapter that motivates the choice of V-Rep as the main simulation tool for this project.

Then, an in-depth presentation of V-Rep and Blender will be made in chapter 4, before detailing some experiments that were made to verify and improve the accuracy of the model.

Chapter 6 goes into the core of the subject with some simulations that influenced the design of the robot before being used to explore control strategies.

The last chapter will conclude the work by summing up and laying out future prospects.

Chapter 2

Principles of interactive rigid body 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.

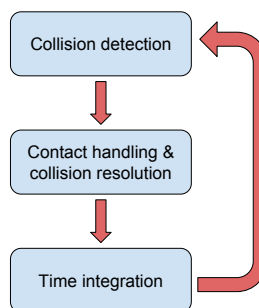


Figure 2.1: Simulation loop

Chapter 3

Choosing the tools

In this chapter we present the reasons for choosing V-Rep as the simulator and Blender as the modelling tool for this project.

3.1 Available tools

3.1.1 Barebone physics engines

The physics engine is the cornerstone of a physics simulation tool. There exist a quantity of them, we present here the most popular ones that have the the following features' set : multi rigid body dynamics, collision detection, joints(hinge, spring-dampers, generic 6DOF¹).

1. **Bullet** : As of now, it is the most popular open source physics engine. Primarily used for games² it is also used in more serious applications such as Blender, V-Rep or NASA's tensegrity robotics toolkit.
2. **ODE** : Open source, a little older than Bullet. Mature for a long time, it is present in a lot of robotics simulation tools (V-Rep, Webots, Gazebo...). It was also designed with games in mind but was influenced by its success in more serious applications.
3. **Newton** : Another open source engine, not quite as popular as Bullet and ODE but nevertheless used for commercial games and simulation. What makes it stand out in the crowd is its deterministic constraint solver, as opposed to iterative solvers used by Bullet and ODE.
4. **PhysX & Havok** : Both are proprietary engines used primarily for games. They won't be further discussed because their focus is on speed rather than accurateness and as such they cannot be used in a simulation that aims to be realistic, as stated by Erez *et al* in [ETT15].

¹DOF : degree of freedom

²Most notably in the Grand Theft Auto(GTA) games series

Engine	License	Coordinates	Solver type
Bullet	Free	Maximal	Iterative
ODE	Free	Maximal	Iterative
Newton	Free	Maximal	Exact

Table 3.1: Features comparison[ETT15]

3.1.2 Simulators

In this section we will speak of software that provides 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)	en-itor	Integrated ed-itor	Modelling
Blender	Free	Bullet		Fully fledged	Internal
V-REP	Free	Bullet, Newton, Vortex(10s limit)	ODE, Vor-	Limited	Can import .COLLADA
Gazebo	Free	Bullet, Simbody, DART	ODE,	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 this section we try and comment on some of the presented options.

3.2.1 Blender

Blender is pleasant to use because the robot's model can be easily changed inside it (Blender is, after all, a modelling tool) and the Python scripting allows fast development. Support for a socket allows an external program to control the simulation and the robot inside it. 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].

Pros :

- Easy modelling.
- Good documentation for Python scripting.

Cons :

- Bullet's simulation parameters are hidden behind an incomplete interface.
- Friction parameters not in a physical value.
- Inertias approximated by identity matrix.

3.2.2 Gazebo

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. That would not be a problem if it could import models easily but that is not the case, 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.

Pros :

- Gives the choice of the physics engine
- Inertias correctly definable
- Friction represented in physical values.

Cons :

- Internal model editor prohibitively limited, cannot be used to create a complicated model. It cannot import popular CAD formats.
- Uses the SDF³ format to store models, making it hard to iterate over robot designs.

³A XML file

3.2.3 V-Rep

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.

Pros :

- Gives the choice of the physics engine.
- Inertias correctly definable.
- Friction represented in physical values.
- Already used at the university.

Cons :

- Limited internal editor, can hardly be used for modelling.

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 features a physics engines does not :

- 3D visualization
- code handling the import of models
- a remote API

All these functionalities would eventually be necessary so a simulator is preferred.

From the available simulators(Blender, Gazebo and V-Rep) the best seems to be V-Rep. This choice is further confirmed by [\[IPP14\]](#) 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 joints⁴.

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 a complement for V-Rep.

⁴with the exception of Vortex but it requires a license to run more than 10s

Chapter 4

Presentation of Blender and V-Rep

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.

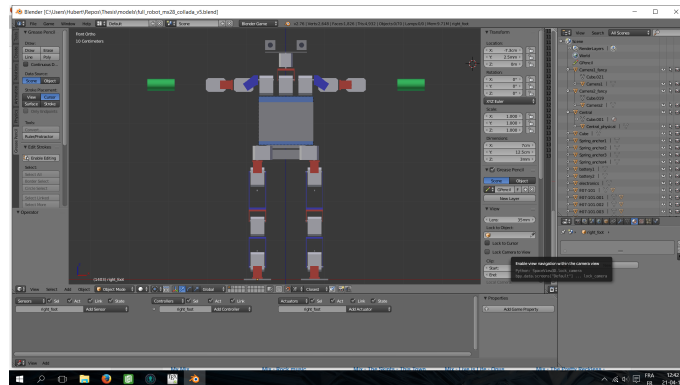


Figure 4.1: Front view of the robot’s model at the end of the modelling in Blender. The grey boxes represent the servos, the red pieces are standard Dynamixel frames, the blue are non-standard, the green cylinders are the hands.

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

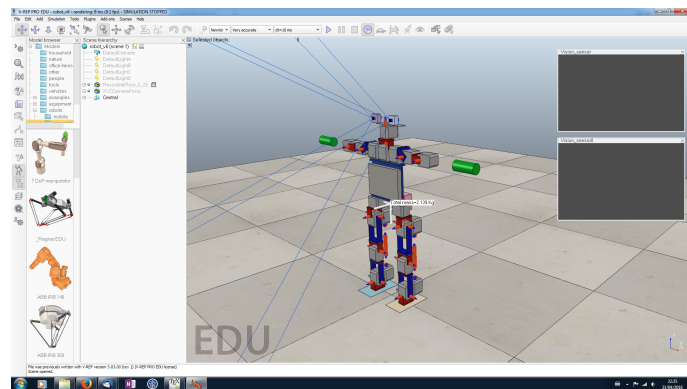


Figure 4.2: View of the robot's model at the end of the modelling in V-REP.

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.

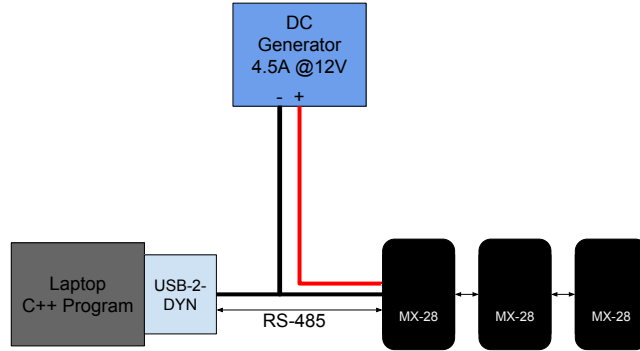


Figure 5.1: Experimental setup : The MX-28 servos are powered by a DC generator and controlled by a laptop equipped with a USB2DYNAMIXEL(USB-2-DYN) device. It converts an USB port into a serial port.

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.

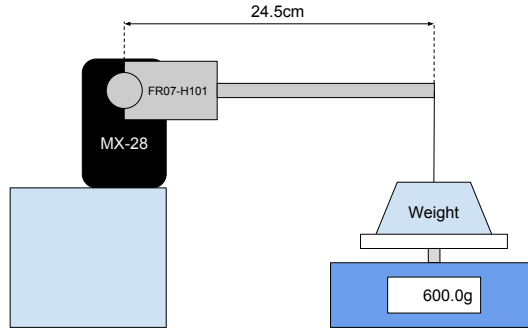


Figure 5.2: Experimental setup for torque testing. A weight w is suspended at a distance d from the servo, resulting in an applied torque of $w \times g \times d$. The goal consists in finding the weight w for which the servo is unable to lift the arm.

	Stall @11.1V	torque [N.m]	Stall @12V	torque [N.m]	Stall @14.8V	torque [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.

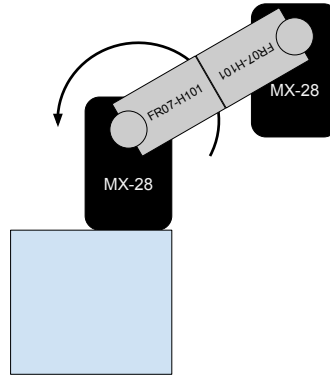


Figure 5.3: Experimental setup for dynamics testing. Two servos are connected together

5.4 Conclusion

Chapter 6

Simulation

In this section we explain how to use the simulator and how some simulations influenced the design of the robot.

6.1 Simulation setup

The basic idea is to use V-Rep as a simulation platform and use TCP/IP to have another program control the simulation from outside. This gives us great implementation flexibility and we can substitute the real robot by the simulation model easily. This is represented on fig. 6.1. For this thesis, the control code will be written in MATLAB.

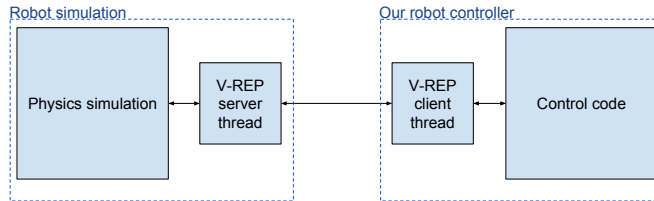


Figure 6.1: V-Rep simulates the robot while an external program sends order to the robot over TCP/IP thanks to the client/server thread provided by V-Rep.

Furthermore, the simulation will operate in the synchronous operation mode. That is, each simulation timestep must be triggered by the control code, allowing precise control of the robot. Figure 6.2 presents how V-Rep and the control code interact in synchronous mode.

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.

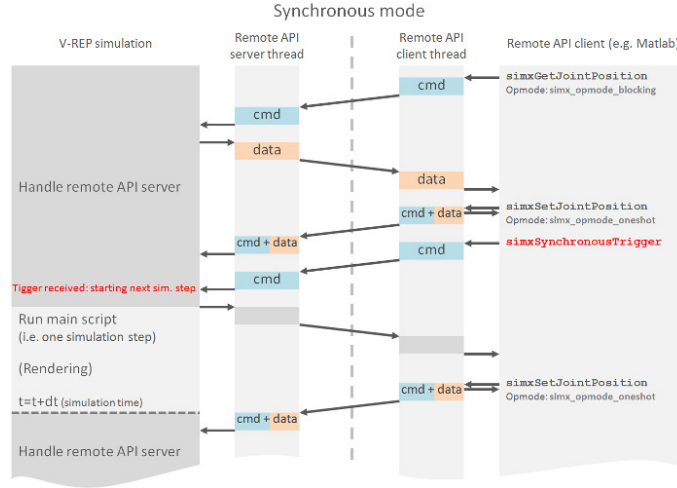


Figure 6.2: Typical interaction between the simulator and the control code. The simulation runs on two threads : the simulation and the server thread. The server threads can receive orders from a client thread which is controlled by a custom application of our own. The simulator waits for a trigger before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

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.

The feet are approximated by a thicker block in order to help with the collision detection. They are also given a high friction.

The torque of the servos is computed from the maximal torque of the DC motor and the reduction ratio of the gears.

$$\begin{aligned}
 Torque &= TorqueMotor \times ReductionRatio \\
 &= 3.67e^{-3} \times 193 \\
 &= 0.7083Nm
 \end{aligned}$$

In section 5.2 we determined that the maximum torque the servo was actually able to produce was $1.7Nm$ so to represent this we choose to set the maximum torque of the servos to $1Nm$.

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

The springs on the leg are simulated by 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.

Module	Weight [g]	Density [kg/m ³]	Dimensions [mm × mm × 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.

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.

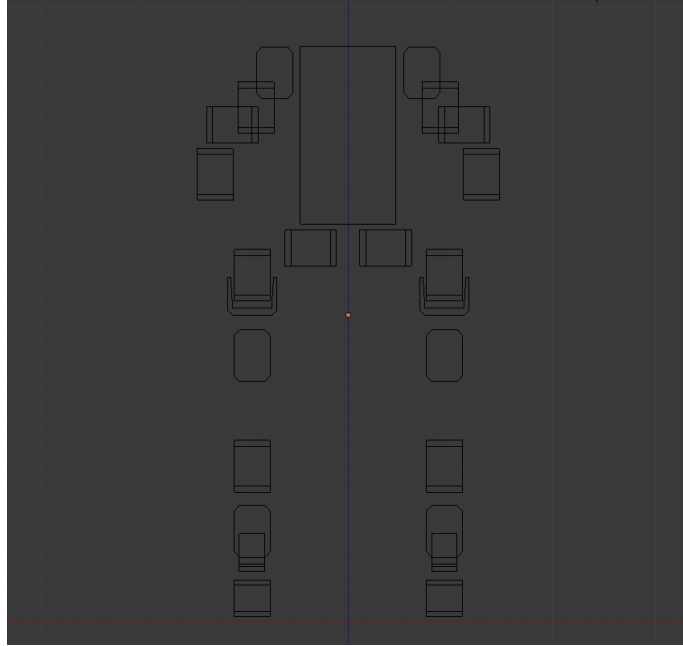


Figure 6.3: First robot design. Arms use 4 servos each, making it quite heavy.

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

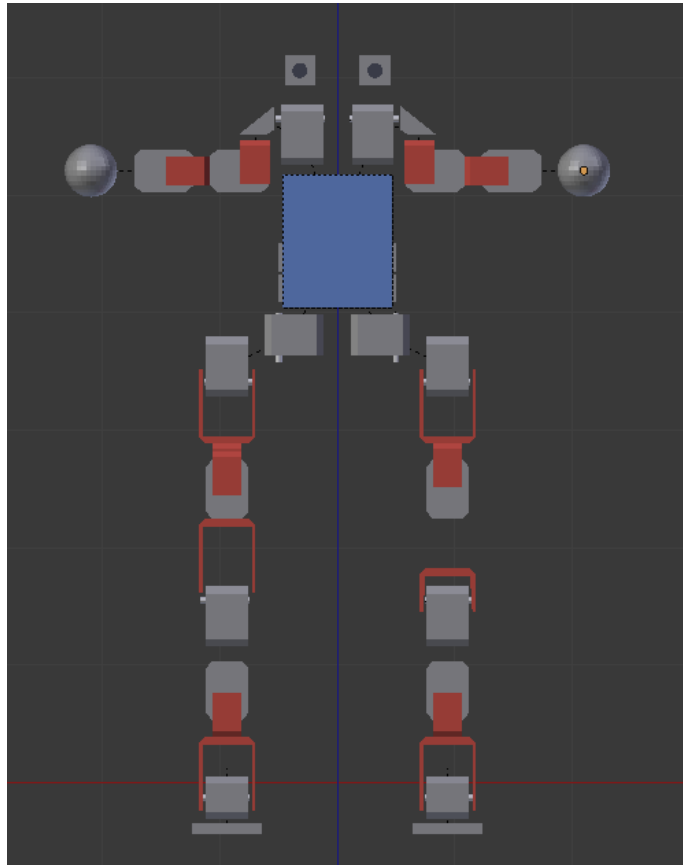


Figure 6.4: Final robot design. Arms now use 3 servos. The feet and the hips use a different configuration to have wider movement possibilities and bring down the center of gravity.

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 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] Coppelias Robotics. V-rep user manual 3.3.0, 2016. [Online; accessed 18-April-2016].
- [SSB06] Jörg Stücker, 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 torso 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.

Appendix B

Design guidelines

For a dynamic simulation several design restrictions must be considered :

- use pure convex as much as possible, they are much more stable and faster to simulate. When a more complex shape is used, approximate it with several convex shapes.
- use reasonable sizes, neither not too small nor too big. Thin shapes may behave strangely.
- when using joints, keep the ratio of the masses below 10. Otherwise, the joint may have large orientation/position errors.