ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

SEMESTER PROJECT M1

BAXTER MODELING AND SIMULATION

June 16, 2014

Guillaume Clivaz

Project supervisor
Michel Olivier

Professor
Francesco Mondada

Abstract

The robot simulation company Cyberbotics in Lausanne has developed a simulation software called Webots and is interested to add a prototype of the promising robot, recently developed : the Baxter. This is thus the aim of this semester project at EPFL, to develop a first simulation of this industrial robot on Webots.

Acknowledgments

I would like to thanks Mr. Olivier Michel for his support and time during the project, as Mr. Francesco Mondada for his following. I also wish to extend my thanks to the team of *Génération robots* and *Rethink Robotics* for their precious help and advices.

Contents

1 Introduction

This project takes part in a semester project with the Cyberbotics company, situated at the Swiss Federal Institute of Technology in Lausanne. The center of interest of this work is the robot called Baxter. This industrial robot is fairly recent, as its creation has been introduce to the industrial domain since 2012, and its future seems promising. the purpose of this project is thus to collect information on this robot, in order to create a first simulation on the Cyberbotics software, *Webots*.

In this introduction, the report will first present briefly Webots, to get an idea of the software and how to familiarize with it. This first part will also introduce the contacts which have helped us to find informations for this project. Then, through this report, the modeling of the robot will be seen in details, and finally his simulation in a world by a controller.

1.1 Webots

Webots is a software program managed by the company Cyberbotics, located onthe site of the Swiss Federal Institute of Technology in Lausanne. This software is specially developed to create and program simulations of mobile robots in an environment. To learn the the first steps and get accustomed with Webots, a series of several tutorials are proposed on the website of Cyberbotics. These tutorials are a quite good way to begin with Webots. Indeed, they are well explain and especially quite fast, less than 30 minutes for each. As Webots is an rather intuitive program, only the essential needed to begin with Webots is shown in these tutorials. Other robots already on Webots can also give a good inspiration to help for programming, like the e-puck, one of the basic and easiest robot to use, already coded on Webots. This especially help to understand how to program the controller in C.

1.2 Research on Baxter

In parallel with the familiarization of Webots, a search of any informations that could concern the Baxter was leaded. From the search, two main responses came back. The first was from *Rethink Robotics*, which is the company that develops this robot. In their answer, we received a link containing the URDF file, with all the meshes files attached. The second answer came from *Génération Robots*, with the same informations than the one from *Rethink Robotics*. With a script converting URDF files into PROTO, we were able to use most parts of the code they sent us. Concerning the specifications of the Baxter , only the basic informations about sensors and actuators were given, that are also findable on their website.

2 Modeling

Once the Webots initiation and the collect of informations mostly done, the modeling work could start. This part will first present a short description of the Baxter robot and the innovations which comes with this robot to the industrial world. Then, it will detail the robot structure and features, as it is coded on the Webots software.

2.1 The Baxter robot

The Baxter robot has been developed by the company *Rethink Robotics*, founded by Rodney Brooks. This industrial robot has been created to perform easy tasks, like taking objects and load them in a destined place. The final goal that aim this robot would be to replace humans in industry for these examples of repetitive tasks. With the possibilities of buying this kind of robot with a profitability higher than with manoeuvres, the companies would may slowly come back in our developed country.

As it is shown in the figure 1, the Baxter has two arms, each with seven degrees of freedom, and then an end-effector to seize objects. What make this robot so interesting for a industrial company is its ability to learn by itself, and be used by usual worker who does not have a high level of knowledge in programming. Indeed, by moving its arm manually, the Baxter is able to learn and memorize the movement, to repeat it by itself after. Workers just have to train it during some minutes, without programming. It is designed and equipped with sensors to work without cages, in safety conditions for the humans around, and can adapts to variations in the environment around him. People from the company will have the time to concentrate on higher-level tasks, if the Baxter do the easy and repeatable one.[3]

Figure 1: Baxter Robot [1]

2.2 Structure of the Baxter

As it is presented below, the modeling has been done mainly by the provided URDF file from *Rethink Robotics* and *Génération Robots* at this link :

- *https://github.com/RethinkRobotics/baxter_common/tree/master/baxter_description*

The modeling work has then been directly done on a PROTO files, not in the *Scene tree side* bar. The script URDFtoPROTO.py, written in python, convert the URDF files into PROTO. The version Python 2.7 is needed for this, the 3.4 version did not work. From the command, once in the repertory of the script, we just needed to write :

python URDFtoPROTO.py baxter.urdf

with the URDF file in argument. The PROTO file is structured as the robot option in Webots, and is detailed below. On the link given, there were also meshes files, which contains all the 3D shapes.

2.2.1 Solids, bounding Objects and Shapes

The robot structure on Webots is a following of solid parts, named *Solids*. As a hierarchical tree, the main solid is taken as a base and other solids extend from this base, called children. If other members follow, they becomes children of the last, etc... The Baxter structure begins with the solid *base* as a base. From it, there are five main members, built as a string of solids :

- The torso, which is the massive central part of the robot.
- The pedestal that carry the whole robot, with four wheels to move it manually.
- The head, with its screen and camera, and one motorized DoF for the screen rotation.
- The right and left arms, composed as a cascade of solids, each separated from the last by an pivot joint, driven by a rotational motor.
- The right and left grippers, with their linearly motorized clamps.

This hierarchical structure came from the *baxter.urdf* file. Each solid had already a given mass, as for its inertial matrix, center of mass and density. In *baxter.proto*, solids with their name finishing with *itb* are commented, like *right_torso_itb, right_arm_itb* and similarly for the left side. These are the interface buttons, called now the navigator. It describe the user input scroll wheel and buttons, including the output LEDs surrounding. As we did not use them for this first simulation, the solids are still there, but commented.

To define the limits of a solid, Webots use *Bounding Objects*, which can have basic form like a capsule or a cylinder, or also a complex form, drawn on a 3D software like Blender or Solidworks. A bounding object in collision with another will be stop by the bounds, except if one solid is the direct children of the other. the Baxter has basic bounding object for every solids, yet there are two complex bounding object available :

- The solid *torso*, which can have the *baselinkcollision* shape, the original shape given in the meshes files, now as a proto.
- The solid *pedestal*, this time with the *pedestallinkcollision* shape.

For a question of simplicity, we did not use these two bounding objects because it could have been too heavy computationally for the simulation.

Finally, a last important feature of a solid is the *Shape* children. It represents the visual form, color, material of the object. The conversion from an URDF to a PROTO file did not convert the shapes given in the meshes files, though no errors had been reported by the script. The shapes were then open in Blender, and registred in WRML file, and then modified as PROTOs. The proto did not convert the color, so I had to had the line *diffuseColor* in the PROTO. To make it easier, the shape had to be split by material in Blender, each material with its own color.

Another weird consequence of the conversion was the non-logical place of the bounding object and the shapes. For the shape, it could be explain because Blender does not have the same referential than Webots, so the object should not be moved on Blender before saving in WRML, and the shape needed to be rotated by a *Transform* around one or more axes. The same was done for the solids, it needed to be rotated every time to get an arm that looked and moved logically. However, the translation did not have to be corrected, so I think this come only because of a difference of referential with the original software were the baxter has been program.

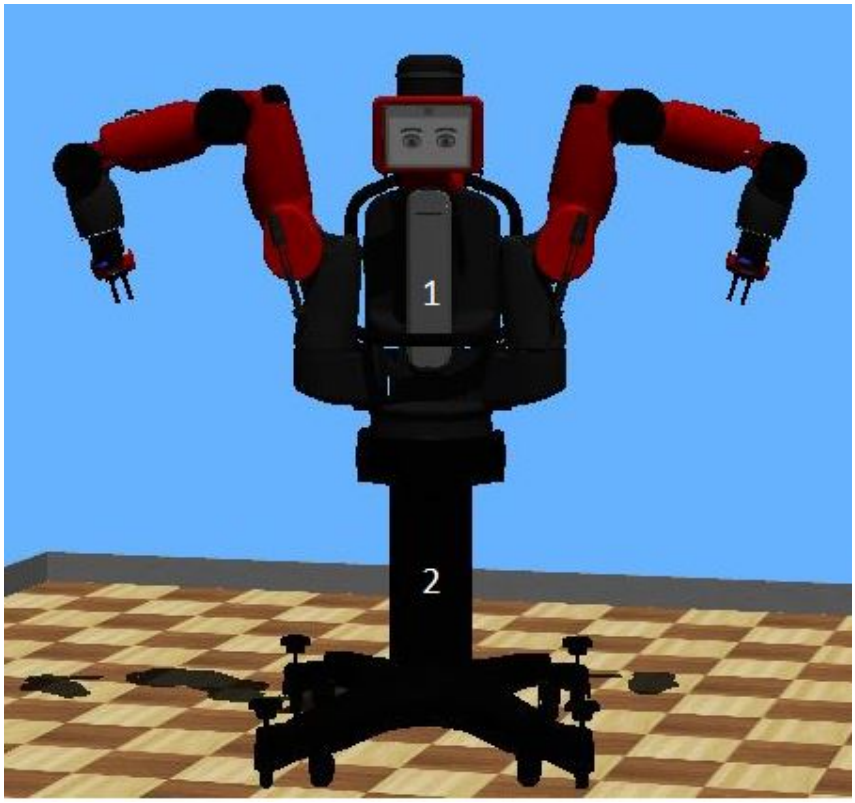

Figure 2: Complete view of the Baxter robot

List of Solids and corresponding Shape

No	Solids	corresponding shape (.proto)
0	base	—-
1	torso	BASE
2	pedestal	PEDESTAL

On the figure 2, we can see the three main solids of the baxter on Webots. These solids does not move or have any actuators and degree of freedom.

Figure 3: View of the Baxter left arm

No	Solids	corresponding shape (.proto)
	left_arm_mount	—-
3	left_upper_shoulder	UPPER_SHOULDER
4	left_lower_shoulder	LOWER_SHOULDER
5	left_upper_elbow	UPPER_ELBOW
6	left_lower_elbow	LOWER_ELBOW
7	left_upper_forearm	UPPER_FOREARM
8	left_lower_forearm	LOWER_FOREARM
9	left_wrist	WRIST

The left arm is shown in the figure 3. The right is symmetrically similar.

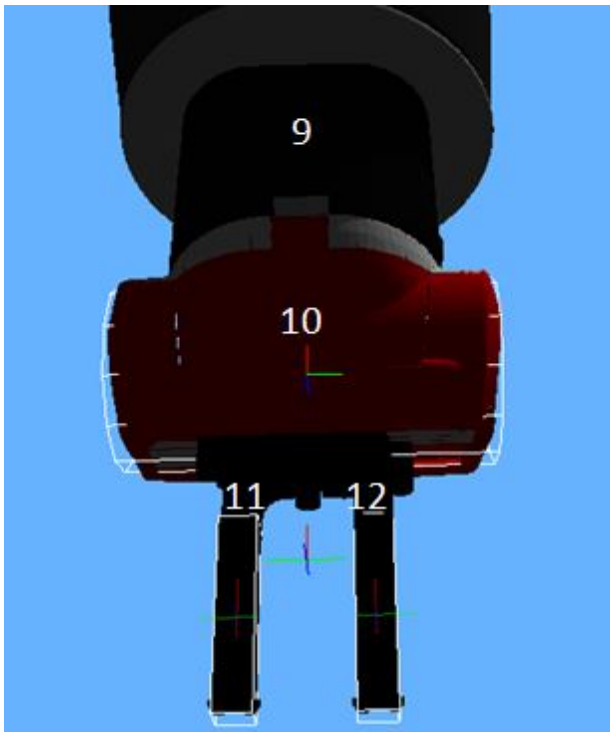

Figure 4: View on the gripper

No	Solids	corresponding shape (.proto)
9	left_wrist	WRIST
	left_hand	—-
10	left_gripper_base	ELECTRIC_GRIPPER
11	left_gripper1	F0030517
12	left_gripper2	F0030517

The gripper with the end of the wrist is shown in the figure 4. In reality, the finger, or clamps of the gripper are interchangeable. The shape of nine different forms of fingers are registered into PROTO files, but for the simulation, only one kind was used.

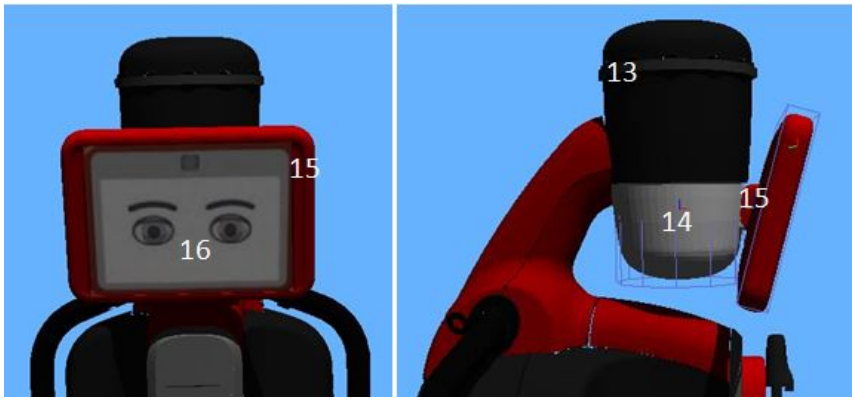

Figure 5: View on the head, screen, and sonar ring

No	Solids	corresponding shape (.proto)
13	sonar_ring_solid	Cylinder form
14	base_head	BASE_HEAD
	dummyhead1	—-
15	screen	SCREEN

The head, screen and sonar solids are indicated in the figure 5. The sonar ring beholds the device sensor. The screen hold the display device, indicated with the number 16, and which is coded in the shape SCREEN.proto. The function of the dummyhead solids is still not known, may this is an actuator.

2.2.2 Actuators

Each arm, from the base to the wrist, has 7 pivot links, which give a total for the arm mobility of 7 degrees of freedom. Each degree is driven and controlled by a rotational motor. The gripper, with the couple of pliers, are not included in the 7 degrees of freedom.

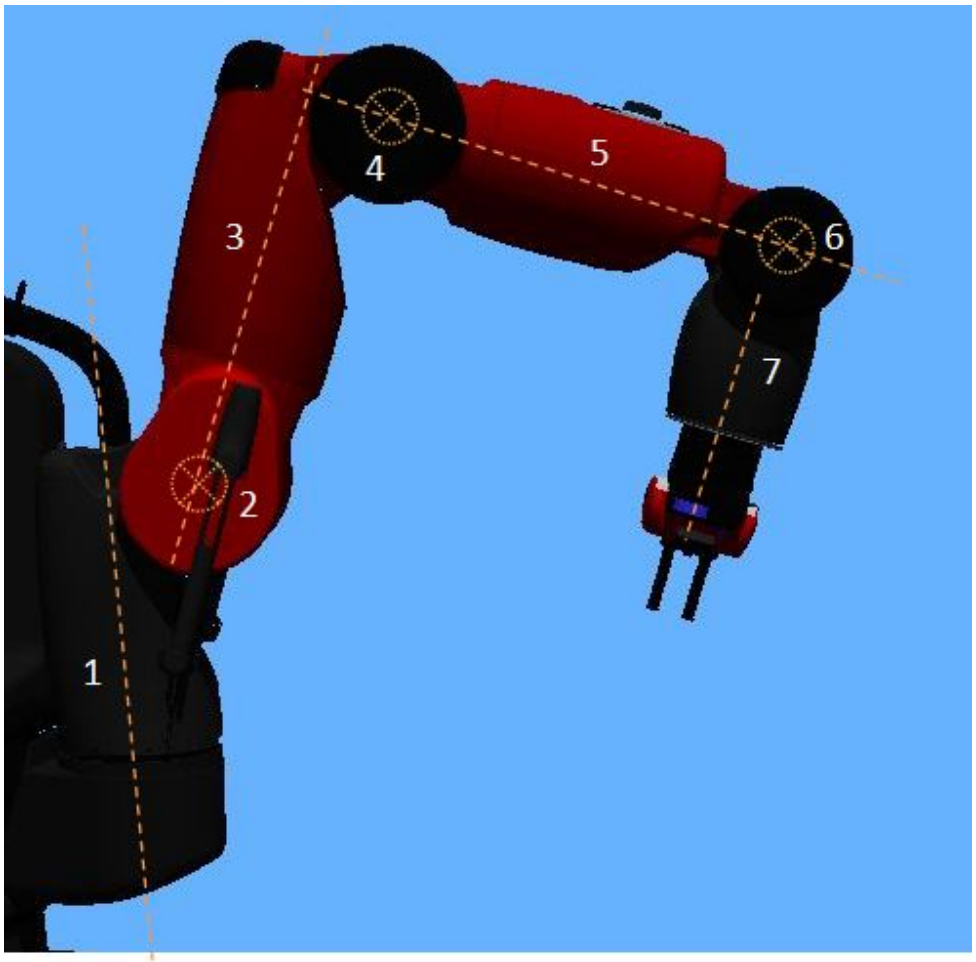

Figure 6: Axis for the rotation for the 7 degree of the arms

No	name of the actuator	Max Torque [Nm]
1	left_s0	50
2	left_s1	50
3	left_e0	50
4	left_e1	50
5	left_w0	15
6	left_w1	15
7	left_w2	15

The figure 6 indicates the axis of rotation for the seven rotational motor in the arm. Maybe because of a problem in the conversion URDF to PROTO, the range of the angles were not really logical. Some motors could almost not move. I thus have shifted the range, always keeping the same difference between the max and min. The gripper have both two linear motors, with a range from -0.02 to 0.02 in Webots, with a maximum torque I have let at 15 [Nm]. Finally, the head has one degree of freedom because of its rotational motor *head_pan*, with a maximum torque non-indicate, and a range over 180°.

2.2.3 Sensors

The Baxter robot has been given many types of sensors like distance sensors, and is also equipped with cameras. There are a front camera and a sonar ring around the head, to detect human presence, and react to it, for more safety conditions of work. The hands are also equipped with cameras. The movement of the robot are thus vision-guided with more precision than with only the head camera. Infra-red distance sensors can also be found in the hands.

Cameras The conversion from the URDF to PROTO file did not add exactly the sensors. All the sensors are under *Solid* form, so the sensors specifications come from the datasheets of *Rethink Robotics*. There are three cameras on the Baxter, one per hand and the last on the head part. Only one kind of camera specification was found in the documentation, so it is supposed in the simulation that the three cameras are similar. The resolution maximum of the camera is 1280x800 pixels.

Infra-red distance sensors These distance sensors are situated in the hands. Their range of detection goes from 4 to 40 [cm].

Sonar-ring This sensor is situated almost at the top of the robot. It can detect by ultrasound any object or human over 360 degree. However, the range of this sensor did not figure in the data sheet, so in the simulation, the sensor can detect from 0.05 to 50 m, which were the values in the baxter.urdf file.

Screen The Baxter also has a screen, which can react to the task given and indicate by many facial expression what it does, or if there was a misunderstand in the task given. The screen resolution is 1024 x 600 pixels.

3 Simulation

Once that the robot is created, Webots permits to move it for a demo, through a simulation. Most of the video and demonstration of the Baxter on internet show the robot taking object and put them somewhere else, on a table. This is the kind of work this industrial robot is sense to achieve, easy repetitive movements that it could learn and proceed after a training, by itself. The demo is then based on this, that is taking object and move them to a desired place.

Figure 7: Baxter Robot in its world stacking cubes

The difficulty of such movements comes from the mobility of the arms. One arm counts seven degrees of freedom, more the end-effectors. A wanted position can be reached with many orientations. As each motors give a rotation, the arm of the robot is difficult to move linearly. If the task is to size an object, we need to move the arm as the two fingers of the end-effector have the object between them, without knocking the object over. A solution is to implement an inverse geometric model, with the coordinates of the positions and orientations given, and as a result, the motors angle values. But because of a lack of time at the end of the project,

this solution was not used. Instead, the motors angles were found experimentally. To seize the object, the arm movement was decomposed in two. First the fast move until close from the wanted position, and finally a last move really slow and as linear as possible.

During the simulation, the Baxter was then able to seize rectangular objects, one by the side and the other from the top, and to stack the second one on the first, without knocking them over. Actually, seizing objects is one of the quality of the Baxter that could be shown during the simulation, not as its other main interesting quality. Indeed, in reality, a worker can take manually the arm of the Baxter, make the move with it, and the Baxter will learn and repeat it individually, but this was avoided in the simulation because of a lack of time, knowledge on the topic, and mainly because we directly give the angles to each motors in the controller.

The simulation can be done completely on a controller, coded in a C-file. Webots has also another tool, the *Motion Editor*, which permit to test and save series of movement.

3.1 Motion Editor

The *Motion Editor* is a tool used to registered a series of movements of the actuators. The sequence of new positions of the motors/actuators is then registered in a *file.motion*, which can be reused on the *Motion Editor*, or loaded by the controller with the function *load_motion_files()* and *start_motion(WbMotionRef motion)*. In the case of the Baxter's demo, the different sequences used to move the actuators are not loaded from the motion file in the controller, due to a better speed control with the controller code, as it is explained after. However, the *Motion Editor* tool was very useful to find the right actuators values for the desired move, easier and faster than to revert the simulation after having modified and compiled the code.

3.2 Controller baxter.c

This C-file controls every sensors, motors or other tools that could be equipped on the robot. The first function that is called in the main after the initialisation is *find_and_enable_device()*, so the controller enable all the devices of the Baxter. Once this is done, the movement part of the demo begins. The different joints are registered in five different arrays :

- *right_arm_motors* and *left_arm_motors* , both containing the seven rotational motors of each arms.
- *right_gripper_motors* and *left_gripper_motors* , both containing the two linear motors of each hands.
- *head_pan_motor* for the rotational motor of the head.

To move one arm by example, an array is created with seven values corresponding to the angles for the different motors and the *right_arm_motors* take each values one by one in a loop FOR, by the two functions *wb_motor_set_position* and *wb_motor_set_velocity*.

Orders can be given from the keyboard during the simulation, once that the keyboard has been enabled in the code. As long as nothing is entered with the keyboard, a loop WHILE continue to run and check. Once it finds an keyboard input, the code enter in a SWITCH instruction. As it was explain at before just before, we need sequence of movement with different velocities to not knock the object over. These are the keyboard main options for the demo :

- the keyboard key 'i', to move both arms to their initial position.
- the keyboard key 'r', to first to move the right arm toward the cube after opening the gripper. Then it takes the cube and close the gripper, to move it somewhere else, open the gripper and goes back first slowly and linearly, and then quickly to the initial position.
- the keyboard key 'l', to first to move the left arm toward the cube after opening the gripper. Then it takes the cube from the top and close the gripper, to move it over the other cube, already displaced. The gripper opens and the arm goes back, again first slowly and linearly, and then quickly to the initial position.

The function *wb_robot_step(20 * TIME_STEP)* is used to make the robot wait a desired time between two other instructions, otherwise the robot directly goes to the second instruction values. More case could be used with the SWITCH, but these are the cases used in the demo.

4 Conclusion

This project was mainly a work on the software Webots, so I get used to a part of it, with its good features and also difficulties. I personally found Webots quite interesting concerning the simulation part of the robot in his world. Many robots are already presents, ready to use for a simulation. It was the first simulation software I had used, but I got a good feeling of it. The control of a robot seems quite realistic, with a wide choice of parameters, sensors, actuators,... to use and do. The controller coded in C is easy to get accustomed to, especially with the help of examples.

However, I met some difficulties during the modeling of the robot. I lost a considerable time during this part of the project because of some problems with PROTO files. By example, when an error was found in the *baxter.proto*, Webots could not response and I had to close it, find the error by myself and save the file, and then I could reopen the world on Webots. I thus think the part of the software concerning the creation of the robot, and the PROTO files could be improved.

Concerning the project, it has finally in a main part get close of its principal objective, which were to create a version of the Baxter robot on Webots. The response and help received from other companies was really helpful. Even if I met problem coming from the conversion of the URDF file, the main structure of the robot was given.

Now that the robot has been created on Webots, an interesting continuation of the project would be to improve the controller. By example, it would be interesting for the Baxter to detect the object and take it by visual control. Also, *Rethink Robotics* and *Génération Robots* were ready to help from the beginning. Now that a first simulation has been done, they may could be more interested and send more informations for further.

References

[1] Rethink Robotics. Baxter research robot.

[2] Rethink Robotics. Brochure ”baxter for packaging”.

[3] Rethink Robotics. Baxter arm and hardware specifications v2-3, 2013.