The ROS “scorbot_gazebo_tf” package

This is a guide for describing, installing and launching the “scorbot_gazebo_tf” package in a local based Ubuntu computer.

1. Installation

(Comment: remember always to type this command at the beginning in the “catkin_ws” directory in a bash shell: “source devel/setup.bash”. This ensures that ROS environment variables are set and ROS commands will work)

After installing the ROS environment and creating the catkin_ws directory, the “scorbot_gazebo_effort” directory can be installed in “~/catkin_ws/src” directory. Just unzip/copy the package inside that directory. Then the path should contain “~/catkin_ws/src/scorbot_gazebo_tf” folder after.

And not forget to run rosdep: “`rosdep install --from-paths src --ignore-src -r -y`” in the ~/catkin_ws/ directory in a bash shell after installing (unzip) the “scorbot_gazebo_tf” package in the “catkin_ws/src” subfolder. This will install all the necessary ROS packages if they are not yet installed.

Finally, please run “~/catkin_ws/catkin_make” command in a Bash shell to compile the package.

2. Structure

The ROS “scorbot_gazebo_effort” package (directory) has the following structure:

```
.
├── CMakeLists.txt
├── config
│   └── controllers.yaml
├── launch
│   ├── load_robot_description.launch
│   ├── scorbot_join_test.launch
│   └── scorbot_gazebo_tf.rviz
├── meshes
│   ├── base.dae
│   ├── body.dae
│   ├── flange.dae
│   ├── forearm.dae
│   ├── gripper_base.dae
│   ├── gripper_finger.dae
│   ├── scorbot_texture.jpg
│   └── upper_arm.dae
├── package.xml
├── src
│   └── scorbot_direct_kinematics.py
├   └── scorbot_inverse_kinematics_testing_(without_d2).py
└── urdf
```

```
├── common.xacro
├── gazebo.urdf.xacro
├── materials.urdf.xacro
├── scorbot_ER-VII.urdf
└── scorbot.urdf.xacro
```

Here is the description of every directory/file:

CmakeLists.txt: it is a necessary file in ROS to compile the whole package using "catkin_make" command in the "~/catkin_ws" directory. In the second line it is specified the name of the package. In this case: "*project(scorbot_gazebo_tf)*". In the 7th line it is specified the other packages dependencies for compiling. In this case: gazebo_ros, urdf and xacro.

config/controllers.yaml: this an important file that specifies the controllers interface of the robot.

Launch: this directory contains the scripts to run the simulation. The "load_robot_description.launch" is a secondary launch script that it is called/loaded in the main "scorbot_join_test.launch". This last launch file loads the scorbot model for Gazebo. See line 11 (it looks for the "scorbot.urdf.xacro" file):

"command="$(find xacro)/xacro '$(find scorbot_gazebo_effort)/urdf/scorbot.urdf.xacro' fixed:=$(arg fixed)" />"

The "scorbot_join_test.launch" specifies nodes and parameters of the simulation.

Line 4 makes Gazebo appear: <arg name="use_gui" default="true"/>
Line 14 launches the Gazebo environment where the scorbot will be placed. It used a default empty world (<include file="$(find gazebo_ros)/launch/empty_world.launch">).
Line 29 spawns the scorbot in the Gazebo (default empty) world. The "-model Scorbot_ER-VII" is the name that you will see in the Gazebo GUI window.
Line 34 spawns the controller node in Gazebo. This is necessary for a Python script to interact with Gazebo. Names of the controllers should later be refereed in the Python script.

Meshes: are graphical and CAD elements to represent the Scorbot ER VII robot in .urdf format. Without these elements we will see the Scorbot as a plain robot with joints/motors viewed as points and scorbot links viewed as lines. These elements were made by other developers.
These files are mentioned and refered inside the "*urdf/scorbot.urdf.xacro*" file.

package.xml: similar to CMakeLists.txt. These packages are necessary for every ROS package. The most important lines are 51-61, and 72-82. Here are specified the dependencies with other ROS packages. The "rosdep" command mentioned in the "1. Installation" section will look for this file to check dependencies.

src: Here is where Python scripts are placed. There is one demo Python script named "scorbot_direct_kinematics.py" that has the inverse kinematic equations of the Scorbot ER VII. Also a demo Python script with the analytic inverse kinematic equations: "scorbot_inverse_kinematics_testing_(without_d2).py". One important advantage of Python scripts in contrast to C++ programs is that it is not necessary to recompile when a Python script is changed or even a new one is added. Nor it is necessary to change or update the "CmakeLists.txt" file.

Urdf: contains the 3D Scorbot robot model for Gazebo. The main file is “scorbot.urdf.xacro” and it is loaded in the “launch/scorbot_join_test.launch” as mentioned before. The others .urdf.xacro files in this directory (“*materials.urdf.xacro*” and “*gazebo.urdf.xacro*”) are loaded/called using this file. Xacro files are just xml files that are expanded during launch files execution or ROS run-time. In this case it expands .urdf files. The “scorbot_ER-VII.urdf” is not really a necessary file and it is not used in the demo. It can be created using this command in a bash shell: “rosrun xacro xacro scorbot.urdf.xacro > scorbot_ER-VII.urdf”.

The most important lines in “scorbot.urdf.xacro” are 22-29. Here the type of joint controls are specified. For example, line 22 we see <xacro:joint_transmission joint="base"/> where. This is expanded using macros in “*gazebo.urdf.xacro*”. Here we can configure the type of hardware joints. It can choose between *joint_transmission* or *joint_pos_transmission*, and that will configure *EffortJointInterface* or *PositionJointInterface*, respectably.

3. Running the demo

These are the steps to run the demo. Tested in Ubuntu 20.04 and ROS Melodic.

Three Bash shells are needed

1) In the first Bash shell type “roscore” to run the main ROS run-time.

2) In the second Bash shell type “roslaunch scorbot_gazebo_tf scorbot_join_test.launch”. This will launch Gazebo. See Fig 1.
Please make sure you have typed the “source devel/setup.bash” in the “catkin_ws” before. This ensures that ROS environment variables are set and ROS commands will work.

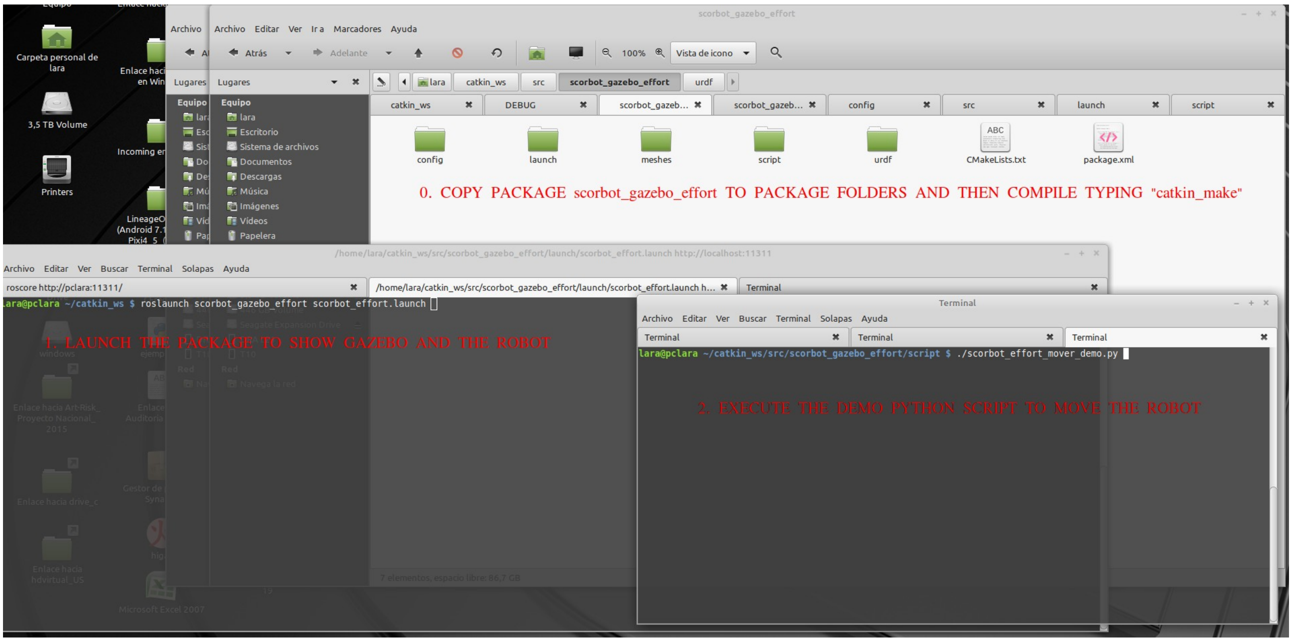

Fig 1. Launch Gazebo.

3) Run the demo script. Go to the “*~/catkin_ws/src/scorbot_gazebo_tf/src*” directory. In the third Bash shell type “./scorbot_direct_kinematics.py”. See Fig 2.
You can also type anywhere in Bash “rosrun scorbot_gazebo_tf scorbot_direct_kinematics.py”.

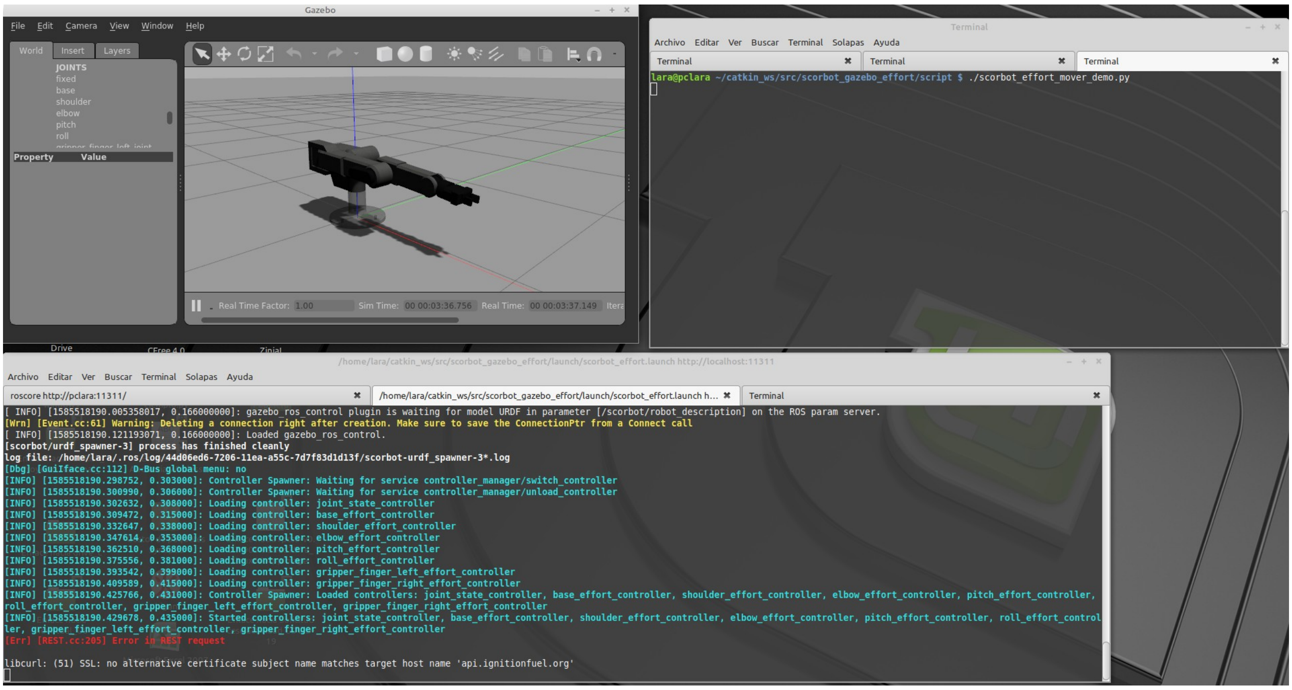

Fig 2. Demo is running: Gazebo is launched and Python script is also running.