Report AU423 Project

IPSA - Diaz Raphael - 19 juillet 2020

Summary

I- Line of thought

A- Expectations

The robot must follow the specifications below :

- Must have at least six links.
- An universal joint.
- A prismatic joint.
- A révolution joint.
- Be able to move in an environment (have wheels).
- Be controllable by the operator.

We must also model an appropriate Gazebo environment model for the robot. Use SDF specification is preferred (.world files). The environment modelled can be an approximation of the picture provided, depending on available assets, relevance to the task. Additional constraints (path, scale between robot/environment) will be provided for each student.

The constraints of the project are :

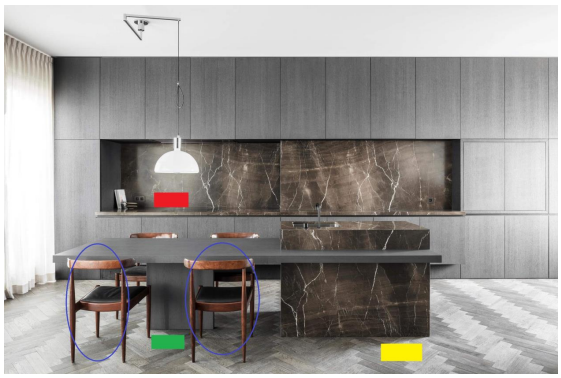
- We can use Ros(v1), gazebo, and RVIZ
- The environment can modelled in any language compatible with Gazebo (URDF, SDF,etc.).
- The project is individual.

We have some resources to do the project :

ROS : https://www.ros.org
Gazebo : http://gazebosim.org
RVIZ : http://wiki.ros.org/rviz
SDF : http://sdfformat.org/spec
Gazebo models : https://github.com/osrf/gazebo_models
3D assets : https://3dwarehouse.sketchup.com/

To do the project, i will use Parralles to work on ubuntu 18.04 and Ros Kinetic.

My environment should be this blueprint :



My robot should start at the yellow box and move around the table to pick up a box over the counter (red box) and deliver it to its destination between the chairs (green box).

B- Line of thought

Before starting my project, i have thought about the way to do my project. First, the 3D environnement. To do it, the use of SketchUP can simplify the modelisation. With this design software, we can export .dae files to import it in gazebo. The .dae file are one of differents extension files which can be import in gazebo. A dae file is a « 3D interchange file used for exchanging digital assets between a variety of graphics programs. It may contain an image, textures, or most likely, a 3D model. The DAE format is based on the COLLADA (COLLAborative Design Activity) xml schema, which is now owned and developed by Autodesk » (source : https://fileinfo.com/extension/dae).

For the robot model, we need to know exactly how work the modelisation of a robot by the use of .xacro files, urdf, .gazebo files etc. For that, the learning of all TPs and lectures about robotics is essential. Before starting work on TPs files, the modelisation of my own robot using only a .sdf files (with no simulation by Rviz and Ros) is the best way to learn.

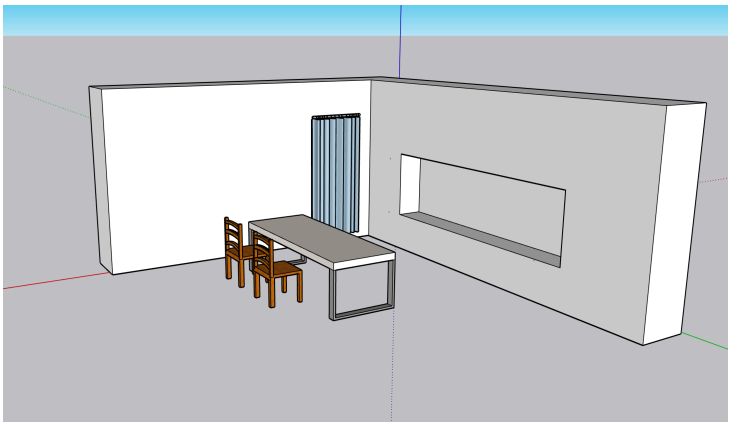
After that, the last step is to do calculation to find the DGM (direct geometric model) and the IGM (inverse geometric model) and implement this in my model.

To do all of that tasks, i need to learn how do wheels model, how have a mobil robot, what is a continuous, prismatic and universal joints etc. I will come back to that later in my report.

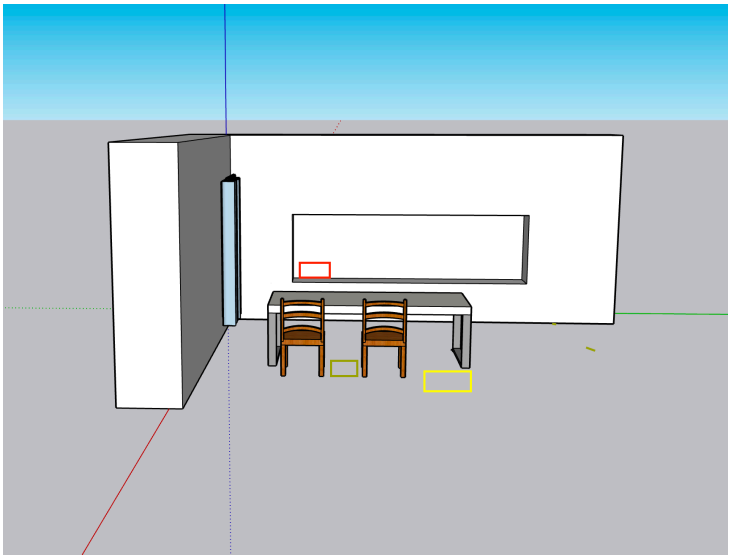
II- 3D environment

A- SkecthUP

First, I have installed a free trial version of skecthUP (https://www.sketchup.com/fr). I have imported different models like table, chairs and curtain. I have created walls. I got this model for my 3D environment :



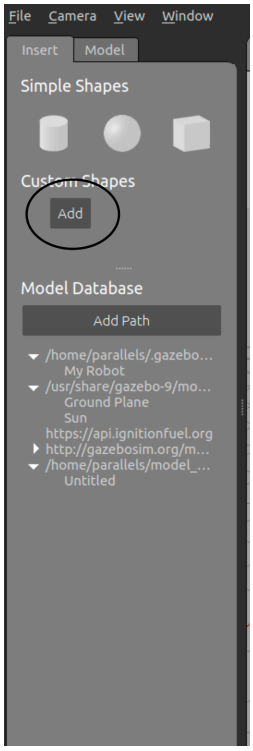
I have reproduced the blueprint to see all the requirements :



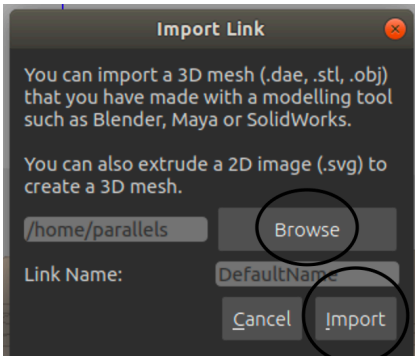
B- Gazebo

Now, with our collada files (.dae) of the 3D environment, i have imported it in gazebo.

I have clicked on « add » in model editor :

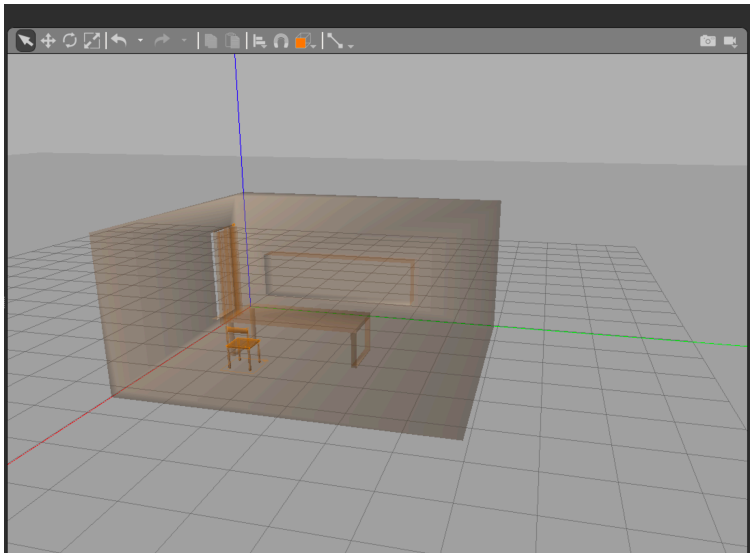


After on « browse and import » :

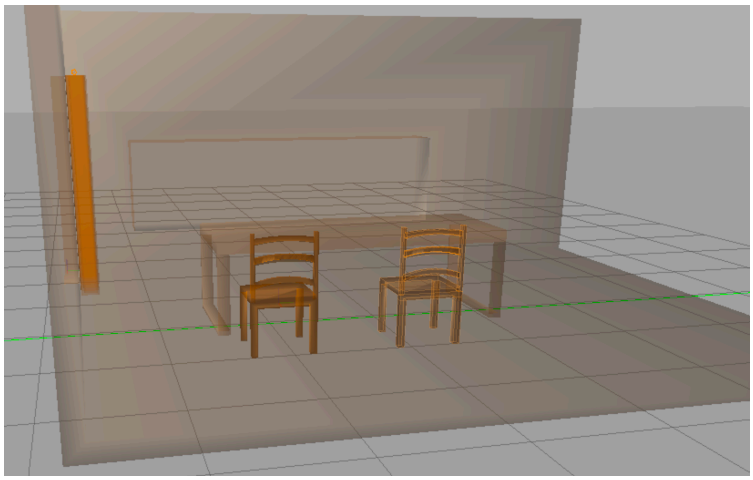


I have selected my file :

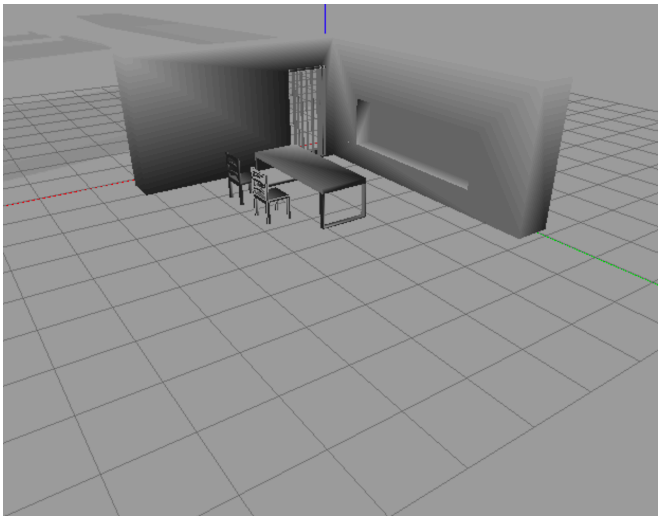
My 3D environment have spawned in gazebo :



But I have had a problem with a chair which didn't spawn. So I have imported a second chair by the same way like my environment model :



After that, I have left the model editor : my final model is here :



C- Problems and Remarks

As you see, the textures are not optimal, it's not the same quality like on SketchUP. For my project it's sufficient, but if I want make an environment more professional and detailed, I think I should use the own gazebo model editor. For a simple simulation, a collada file from sketchUP is the best way to have a 3D environment. I have a problem with the position on Z axis of my table and chairs. I thinks it's because I don't have floor. If I didn't put a floor in my environment, it's because the robot can't move on a floor import from SketchUP.

III- Model of my robot

A- Code and simulation

Like I said previously, I did a model of my robot in a simple sdf file to see how I can do to place wheels and my robot arm. First I have create a .config file :

```
<?xml version="1.0"?>
<model>
  <name>My Robot</name>
  <version>1.0</version>
  <sdf version='1.4'>model.sdf</sdf>

  <author>
   <name>My Name</name>
   <email>me@my.email</email>
  </author>

  <description>
    My awesome robot.
  </description>
</model>
```

After that, I have started with the modelisation of my base named « chassis » :

```
<?xml version='1.0'?>
<sdf version='1.4'>
  <model name="my_robot">


    <static>false</static>
     <link name='chassis'>
            <pose>0 0 .1 0 0 0</pose>

            <collision name='collision'>
              <geometry>
                <box>
                  <size>.8 .4 .1</size>
                </box>
              </geometry>
            </collision>

            <visual name='visual'>
              <geometry>
                <box>
                  <size>.8 .4 .1</size>
                </box>
              </geometry>
            </visual>
          </link>
```

I have add the wheels (you can see an exemple of one wheel, all wheels are similated).

```
<link name="right_wheel">
     <pose>0.30 -0.225 0.1 0 1.5707 1.5707</pose>
     <collision name="collision">
       <geometry>
         <cylinder>
           <radius>.1</radius>
           <length>.05</length>
         </cylinder>
       </geometry>
     </collision>
     <visual name="visual">
       <geometry>
         <cylinder>
           <radius>.1</radius>
           <length>.05</length>
         </cylinder>
       </geometry>
     </visual>
   </link>
```

I have used different tags like « cylinder » ; « radius » ; « length » etc

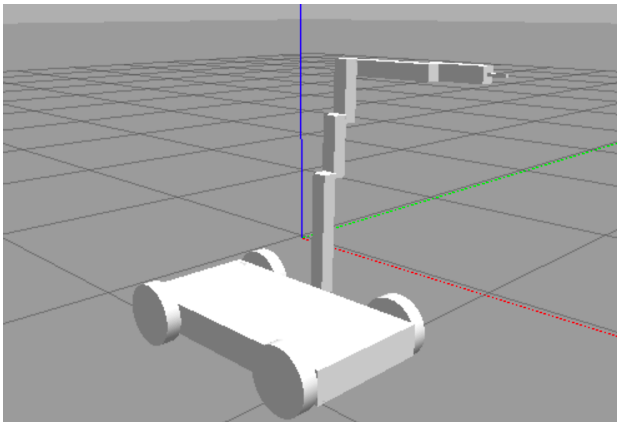
The way to add wheels is the same to add arm. You just have to change the dimension parameters and change the joints parameters. I have used revolute joints to place my wheels, if I want to have turning wheels, I have to put the axis on Y :

```
<joint type="revolute" name="right_wheel_hinge">
  <pose>0 0 0.03 0 0 0</pose>
  <child>right_wheel</child>
  <parent>chassis</parent>
  <axis>
    <xyz>0 1 0</xyz>
  </axis>
</joint>
```

I put five arms to see how the model can be viable :

```
<link name='arm4'>
     <pose>0.15 0.375 0.85 0 0 0</pose>

     <collision name='collision'>
       <geometry>
         <box>
           <size>.3 .05 .05</size>
         </box>
       </geometry>
     </collision>

     <visual name='visual'>
       <geometry>
         <box>
           <size>.3 .05 .05</size>
         </box>
       </geometry>
     </visual>
   </link>


<joint type="revolute" name="arm4_hinge">
  <pose>-0.15 0 0 0 0 0</pose>
  <child>arm4</child>
  <parent>chassis</parent>
  <axis>
   <xyz>0 1 0</xyz>
  </axis>
</joint>
```

My training robot was over :



After I have started to consider how my project robot model will be. I have decided to remove one arm and enlarge my first arm to have a robot with properties more realistic.

I didn't start directly my robot with the code of my training robot because of the difference of language between my training file and the AU423 files used during TPs classes. I did many researchs to know how I can adapt my code. Finally I have noted some difference between the two codes :

```
<link name="base_link">
  <visual>
    <origin xyz="0 0 0"/>
    <geometry>
      <box size="${base_width} ${base_width} ${base_height}"/>
    </geometry>
    <material name="black"/>
  </visual>
  <collision>
    <origin xyz="0 0 0"/>
    <geometry>
      <box size="${base_width} ${base_width} ${base_height}"/>
    </geometry>
  </collision>
  <xacro:box_inertial mass="4" width="${base_width}" height="${base_height}" depth="${base_width}"/>
  </link>
```

```
<?xml version='1.0'?>
<sdf version='1.4'>
  <model name="my_robot">


    <static>false</static>
     <link name='chassis'>
            <pose>0 0 .1 0 0 0</pose>

            <collision name='collision'>
              <geometry>
                <box>
                  <size>.8 .4 .1</size>
                </box>
              </geometry>
            </collision>

            <visual name='visual'>
              <geometry>
                <box>
                  <size>.8 .4 .1</size>
                </box>
              </geometry>
            </visual>
          </link>
```

For the base (« chassis or « base_link »), the first code (see above) use xacro to define geometry parameters. I had inertial properties to can launch my robot on Rviz and Gazebo. The joint system for the base is similar between the two codes.

After have modified my code to obtain it in a good language, i have this :

For the base :

```
<link name="base_link">
  <visual>
    <origin xyz="0 0 0"/>
    <geometry>
      <box size="${base_width} ${base_width} ${base_height}"/>
    </geometry>
    <material name="black"/>
  </visual>
  <collision>
    <origin xyz="0 0 0"/>
    <geometry>
      <box size="${base_width} ${base_width} ${base_height}"/>
    </geometry>
  </collision>
  <xacro:box_inertial mass="4" width="${base_width}" height="${base_height}" depth="${base_width}"/>
  </link>
```

For the arms (it's an exemple for one arm, the code for others arms are similated).

```
<link name="arm1">
  <visual>
    <origin xyz="${arm1_length/2} 0 ${-arm1_width/2}"/>
    <geometry>
      <box size="${arm1_length} ${arm1_width} ${arm1_width}"/>
    </geometry>
    <material name="red"/>
  </visual>
  <collision>
    <origin xyz="${arm1_length/2} 0 ${arm1_width/2}"/>
    <geometry>
      <box size="${arm1_length} ${arm1_width} ${arm1_width}"/>
    </geometry>
  </collision>
  <xacro:box_inertial mass="0.7" width="${arm1_width}" height="${arm1_width}" depth="${arm1_length}"/>
</link>

<joint name="base_link_to_arm1" type="continuous">
  <parent link="base_link"/>
  <child link="arm1"/>
  <origin xyz="0 ${base_width/2} 0" rpy="${pi/2} 0 0"/>
  <axis xyz="0 0 1"/>
  <limit effort="10" lower="${-pi}" upper="0" velocity="1"/>
</joint>
```

For the wheels it was more complicated, I did intensive research's to have wheels in the good position and with the capacity to turn.

```
<link name="right_wheel">
  <visual>
    <origin xyz="0 ${-right_wheel_radius/2} 0" rpy="${pi/2} 0 0"/>
    <geometry>
      <cylinder radius="0.05" length="0.05"/>
    </geometry>
    <material name="blue"/>
  </visual>
  <collision>
    <origin xyz="0 ${-right_wheel_radius/2} 0" rpy="${pi/2} 0 0"/>
    <geometry>
      <cylinder radius="0.05" length="0.05"/>
    </geometry>
  </collision>
  <!-- <xacro:box_inertial mass="0.1" width="1" height="1" depth="1"/>-->
  <inertial>
    <mass value="0.2"/>
    <origin rpy="0 1.5707 1.5707" xyz="0 0 0"/>
    <inertia ixx="3" ixy="0" ixz="0" iyy="3" iyz="0" izz="3"/>
  </inertial>
</link>

<joint name="base_link_to_right_wheel" type="continuous">
  <parent link="base_link"/>
  <child link="right_wheel"/>
  <origin xyz="${base_width/2} ${-base_width/2} ${-base_width/4} "/>
  <axis xyz="0 1 0"/>
  <limit effort="10000" velocity="1000"/>
  <joint_properties damping="0" friction="0" />
</joint>
```

As you can see, I have used « continuous » joints instead of « revolute » joints, to have a rotation more realistic with capacitate to turn. The big difference between the code of arms and wheels it's for inertial properties : for arms, I have used the professor formula. But for wheels I did my own research and I have find the good formula :

$$ixx = \frac{m * (3 * r * r + l * l)}{12}$$

$$iyy = \frac{m * r * r}{2}$$

$$izz = \frac{m * (3 * r * r + l * l)}{12}$$

$$iyz = 0$$

$$ixy = 0$$

$$ixz = 0$$

With m=mass ; r=radius ; l=length.

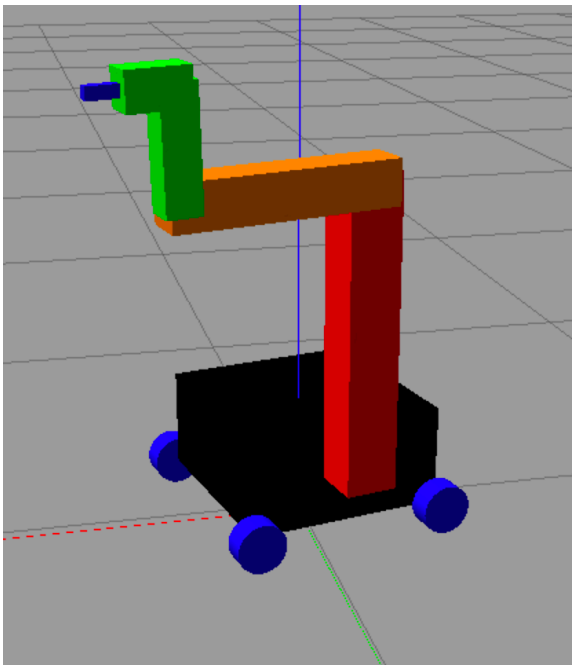
After have a good code with no errors, I have modified the transmission and gazebo files to have a model wich will be launch in gazebo and Rviz :

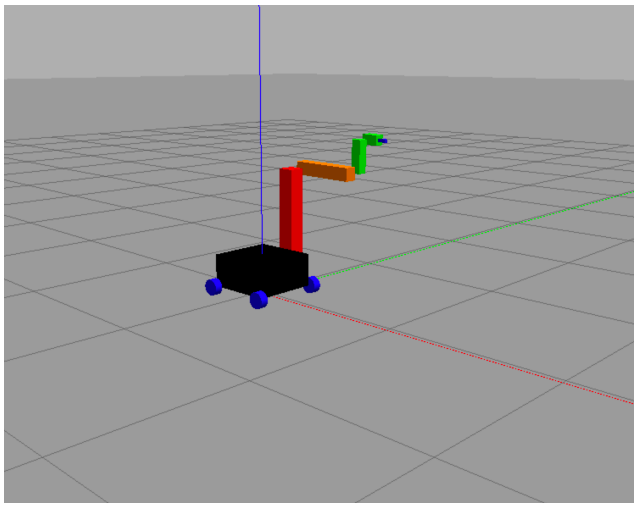
```
 <transmission name="left_wheel_trans">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="base_link_to_left_wheel">
    <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
  </joint>
  <actuator name="motor6">
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>


 <transmission name="left_back_wheel_trans">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="base_link_to_left_back_wheel">
    <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
  </joint>
  <actuator name="motor7">
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>


<transmission name="right_wheel_trans">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="base_link_to_right_wheel">
    <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
  </joint>
  <actuator name="motor8">
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>
```

```
<gazebo reference="left_wheel">
  <material>Gazebo/Blue</material>
  <mu1>100.0</mu1>
  <mu2>50.0</mu2>
</gazebo>

<gazebo reference="right_wheel">
  <material>Gazebo/Blue</material>
  <mu1>100.0</mu1>
  <mu2>50.0</mu2>
</gazebo>

<gazebo reference="right_back_wheel">
  <material>Gazebo/Blue</material>
  <mu1>100.0</mu1>
  <mu2>50.0</mu2>
</gazebo>


 <gazebo reference="left_back_wheel">
  <material>Gazebo/Blue</material>
  <mu1>100.0</mu1>
  <mu2>50.0</mu2>
</gazebo>
```

You can see some exemples of this files.

After that, i had a good model in gazebo :





The next step it was to have a **mobile** robot. For that, I have used a plugin : «differential_drive_controller » wich file named «libgazebo_ros_diff_drive.so ».
This plugin is « a model plugin that provides a basic controller for differential drive robots in Gazebo. You need a well defined differential drive robot to use this plugin » (source :http://gazebosim.org/tutorials?tut=ros_gzplugins). I have used two differential drive controller plugin because one plugin can just manage two opposite wheels. I have this code :

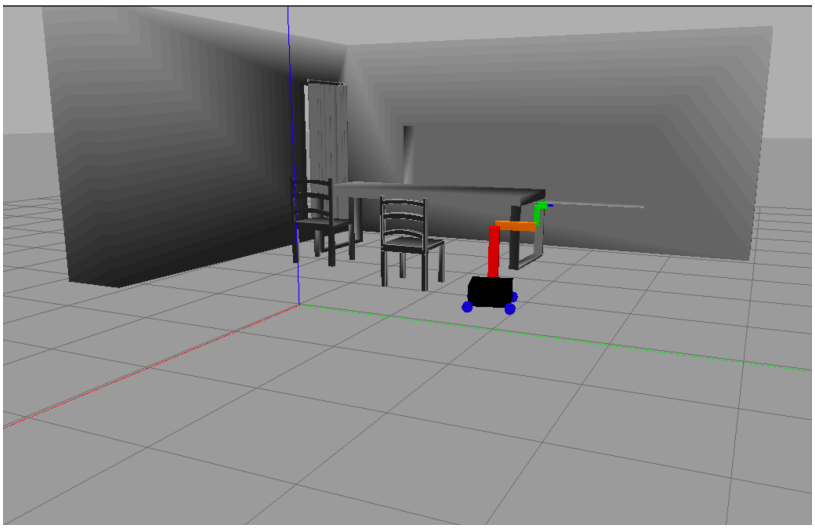
```
<gazebo>
  <plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so">
    <robotNamespace>/robot_ipsa</robotNamespace>
  </plugin>
  <plugin filename="libgazebo_ros_diff_drive.so" name="differential_drive_controller">
    <alwaysOn>true</alwaysOn>
    <updateRate>20</updateRate>
    <publishWheelTF>true</publishWheelTF>
    <leftJoint>base_link_to_left_wheel</leftJoint>
    <rightJoint>base_link_to_right_wheel</rightJoint>
    <wheelSeparation>0.5380</wheelSeparation>
    <wheelDiameter>0.1</wheelDiameter>
    <torque>0.5</torque>
    <commandTopic>cmd_vel</commandTopic>
    <odometryTopic>odom</odometryTopic>
    <odometryFrame>odom</odometryFrame>
    <robotBaseFrame>base_link</robotBaseFrame>
  </plugin>
  <plugin filename="libgazebo_ros_diff_drive.so" name="differential_drive_controller">
    <alwaysOn>true</alwaysOn>
    <updateRate>20</updateRate>
    <publishWheelTF>true</publishWheelTF>
    <leftJoint>base_link_to_left_back_wheel</leftJoint>
    <rightJoint>base_link_to_right_back_wheel</rightJoint>
    <wheelSeparation>0.5380</wheelSeparation>
    <wheelDiameter>0.1</wheelDiameter>
    <torque>0.5</torque>
    <commandTopic>cmd_vel</commandTopic>
    <odometryTopic>odom</odometryTopic>
    <odometryFrame>odom</odometryFrame>
    <robotBaseFrame>base_link</robotBaseFrame>
  </plugin>
</gazebo>
```

To have a mobile robot using this plugins, it is necessary to install the associated package (source : http://wiki.ros.org/diff_drive_controller).

After all this steps, I had a mobile robot. To launch the simulation, we have to write this line in terminal : « roslaunch description_robosa simu.launch » and in an other terminal, I have to write this : « run teleop_twist_keyboard teleop_twist_keyboard.py ». To have this command working, we should install this package telepop_twist_keyboard (source : http://wiki.ros.org/teleop_twist_keyboard). With this package you can control the robot with your keyboard :

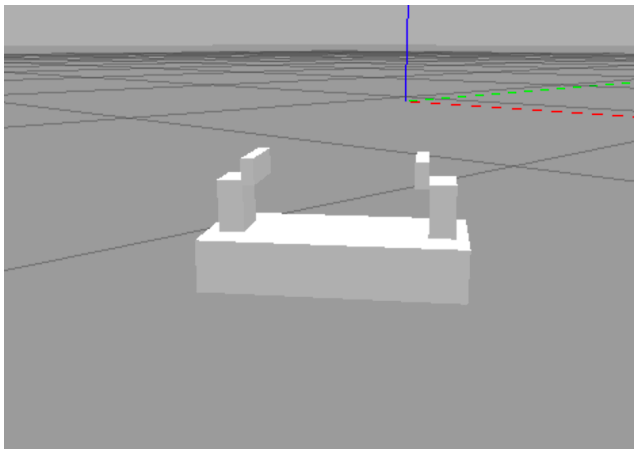
```
Moving around:
   u    i    o
   j    k    l
   m    ,    .

For Holonomic mode (strafing), hold down the shift key:
---------------------------
   U    I    O
   J    K    L
   M    <    >

t : up (+z)
b : down (-z)

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit

currently:      speed 0.5       turn 1.0
```

Here, you can see my robot in my environment :



B- The gripper

I have decided to create a gripper to fixe it at the end of the arm. So I have used two prismatic joints to have a movement aloung the axis X. But I have some difficulties to fixe it a the end of the arm. I have a problem with the X axis, when I launch the simulation, the little grippers move out the base of my gripper. I have tried to change all the inertial parameters but it's not the problem.

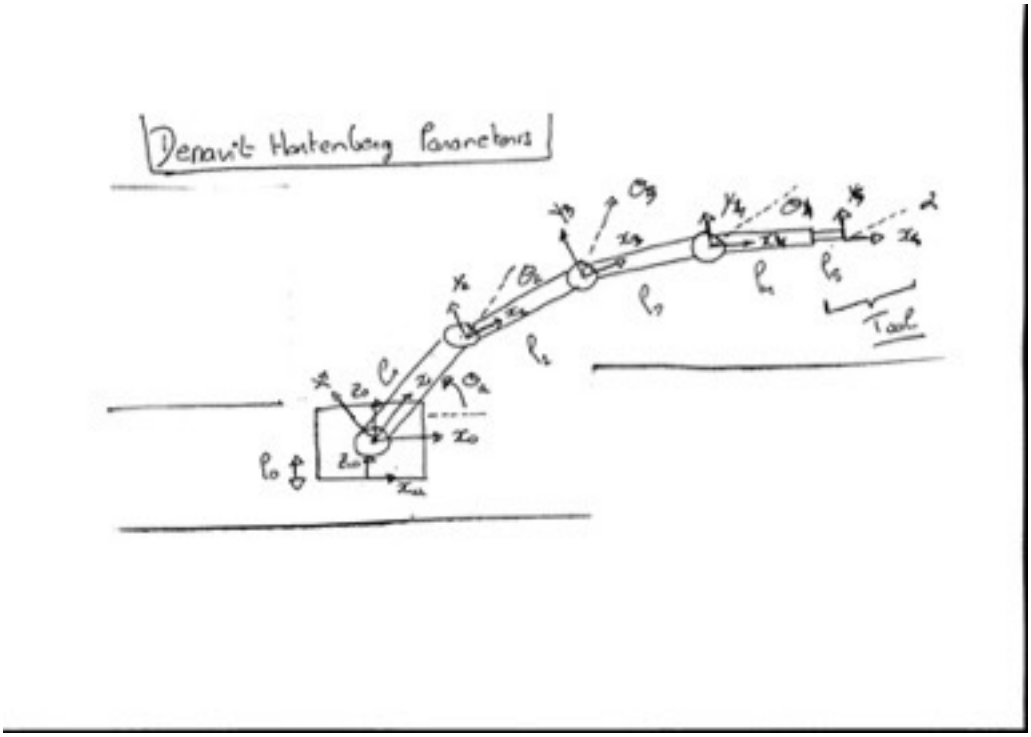


C- Problems and remarks

During this part, I had many problems with my model, my code and the gripper. I have a mobile robot but I think I have some errors of dimensions and parameters. The robot can roll and turn, but the physics is not perfect and I have some bugs. The arm can move and be in different positions, but I don't know if it's possible for the arm to pick up a box. When I launch the simulation, I have some errors in Rviz. It's normal. Because for launch Rviz, the model should have a « world » frame which is fixed. But if I do that, the mobile robot can't move in Gazebo.

IV- Calculations

The objectives of this part is to determine and implemante the inverse geometric model.



$S_{i\to i+1}$	a_i	α_i	d_{i+1}	θ_{i+1}
$S_{w\to 0}$	0	0	l_0	0
$S_{0\to 1}$	0	$\pi/2$	d_{y_0}	θ_1
$S_{1\to 2}$	l_1	0	d_{y_1}	θ_2
$S_{2\to 3}$	l_2	0	d_{y_2}	θ_3
$S_{3\to 4}$	l_3	0	d_{y_3}	θ_4
$S_{4\to 5}$	$l_4 + l_5$	0	d_{y_4}	0

We now determine the homogeneous matrices of transformations for each joint :

$$T_{w\to 0} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_{l_0} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{0\to 1} = \begin{bmatrix} C_{\theta_1} & -S_{\theta_1} & 0 & 0 \\ 0 & 0 & -1 & -d_{y_0} \\ S_{\theta_1} & C_{\theta_1} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{1\to 2} = \begin{bmatrix} C_{\theta_2} & -S_{\theta_2} & 0 & l_1 \\ S_{\theta_2}C_0 & C_{\theta_2}C_0 & 0 & 0 \\ 0 & 0 & 1 & C_O d_{y_1} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{2\to 3} = \begin{bmatrix} C_{\theta_3} & -S_{\theta_3} & 0 & l_2 \\ S_{\theta_3}C_0 & C_{\theta_3}C_0 & 0 & 0 \\ 0 & 0 & 1 & C_O d_{y_2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{3\to 4} = \begin{bmatrix} C_{\theta_4} & -S_{\theta_4} & 0 & l_3 \\ S_{\theta_4}C_0 & C_{\theta_4}C_0 & 0 & 0 \\ 0 & 0 & 1 & C_O d_{y_3} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{4\to 5} = \begin{bmatrix} 1 & 0 & 0 & l_4 + l_5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & C_O d_{y_4} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

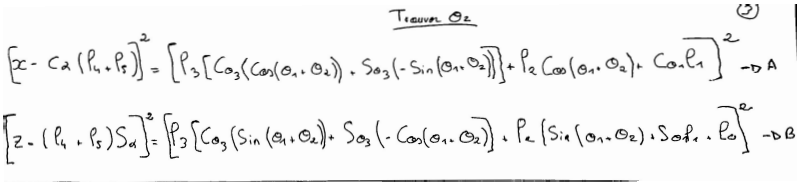
We can therefore deduce the direct geometric model:

$$x = l_3[C_{\theta_3}(C_{\theta_1}C_{\theta_2} - S_{\theta_1}C_{\theta_2}) + S_{\theta_3}(-C_{\theta_1}S_{\theta_2} - S_{\theta_1}C_{\theta_2})] + l_2[C_{\theta_1}C_{\theta_2} - S_{\theta_1}S_{\theta_2}] + C_{\theta_1}l_1 + (l_4 + l_5)[C_{\theta_4}[C_{\theta_3}(C_{\theta_1}C_{\theta_2} - S_{\theta_1}S_{\theta_2}) + S_{\theta_3}(-C_{\theta_1}S_{\theta_2} - S_{\theta_1}C_{\theta_2})] + S_{\theta_4}(-S_{\theta_3}(C_{\theta_1}C_{\theta_2} - S_{\theta_1}S_{\theta_2}) + C_{\theta_3}(-C_{\theta_1}S_{\theta_2} - S_{\theta_1}C_{\theta_2}))]$$

$$y = -d_{Y_4} - d_{Y_3} - d_{Y_2} - d_{Y_1} - d_{Y_0}$$

$$x = (l_4 + l_5)[C_{\theta_4}[C_{\theta_3}(S_{\theta_1}C_{\theta_2} + C_{\theta_1}S_{\theta_2}) + S_{\theta_3}(-S_{\theta_1}S_{\theta_2} - C_{\theta_1}C_{\theta_2})] + S_{\theta_4}(-S_{\theta_3}(S_{\theta_1}C_{\theta_2} + C_{\theta_1}S_{\theta_2}) + C_{\theta_3}(-S_{\theta_1}S_{\theta_2} + C_{\theta_1}C_{\theta_2}))] + l_3[C_{\theta_3}(S_{\theta_1}C_{\theta_2} + C_{\theta_1}C_{\theta_2}) + S_{\theta_3}(-S_{\theta_1}S_{\theta_2} + C_{\theta_1}C_{\theta_2})] + l_2[S_{\theta_1}C_{\theta_2} + C_{\theta_1}S_{\theta_2}] + S_{\theta_1}l_1 + l_0$$

I didn't finish my calculations because I have some difficulties to reduce my equations. Below, this is a photo of my last step of calculation of the inverse geometric model.



I have used the same step of the tp calculation's to do this. I didn't write it in LateX in my report because the lines are to big for one page.

V- Conclusion

This project was complicated and difficult, I don't finish and all tasks are not fulfilled but I successfully did some tasks and I have learned how modelise mobile robot with wheels in a environment. I think, with more time, I will be able to finish all tasks and have a program which can move the robot automatically to a position and pick up a boxe. My gazebo installation had many bugs and problems, like Ubuntu 18.04 on Parallels.